République Algérienne Démocratique et Populaire

Université Saad dahlab Blida

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option : Ingénierie de logiciel

Implémentation d'une méthode de compression d'image médicale dans un système embarqué (FPGA)

Réalisé par :

Rahali Abdenacer

Promoteurs

Mme F.Alim CDTA
Mr F.Talbi CDTA

Encadrer par

Mr Djilali Nahal USDB

Année universitaire 2016-2017

Remerciement

Je tiens à remercier Dieu et Dieu de me donner du courage, de la patience et de la violence pour accomplir cet humble travail.
À la fin de ce travail, j'aimerais exprimer ma profonde appréciation et mes sincères remerciements à:

Mr.Talbi de me avoir encadré au sein de l'équipe de recherche, AC2 (Architectures pour la classification et cryptographie), de la division ASM et à leur tête Mme D.ALIM, Chef d'équipe, pour me avoir accueilli et de me avoir permis de travailler dans de bonnes conditions.

j'exprimé ma profonde gratitude et mes remerciements les plus sincères à Mr.Djilali Nahal mon cou promoteur au département d'Informatique à la Faculté des Sciences de l'Université de Blida, pour son aide, ses conseils, ses encouragements, à qui je dit merci pour votre haute sympathie, votre patience.

Je souhaite aussi remercier mesdames et messieurs les membres du jury pour leur précieux temps accordé à l'étude de ma mémoire.

Je tenais à remercier également l'ensemble des enseignants Qui ont Contribué à mon formation universitaire.
Que soient enfin remerciés tous ceux et celles qui ont bien Voulu d'une façon ou d'une autre participer à élaboration de ce modeste travail.

Dédicaces

J'ai le grand plaisir de dédier ce modeste travail

A:

Ceux qui m'ont encouragé dans ma vie la source de l'amour,
La lumière de mes yeux, mes très chers parents :

Ma mère ***ZOHRA*** *et mon père* ***mouloud***
Que DIEU me les garde et les protège

*A mes sœur et frères (**I**dris, **I**kram, **A**dberahim, **O**kba, **K**hadijda)*

*A mes chères amies (**S**ofian, **A**bdelmalk et **R**abiz)*

*A mes collègues (**F**atma et **S**ara)*

A tous mes enseignants

Sans oublier tous mes amis de L'arbaa, en particulier mes camarades des groupes Informatique.

Nacer Rahali

Résumé

L'objectif principal de ce travail est d'implémenter sous l'environnement Vivado 2015.4 sur un circuit FPGA (Field Programmable Gate Array) une méthode hybride entre les ondelettes et les réseaux de neurones dont le but est la compression des images médicales IRM. Pour cela nous avons étudié la transformée en ondelettes basée sur l'approche du lifting schème et l architecture du carte SOM dan le but est de l'implémenter sur un circuit FPGA (ZedBoard).

Abstract

The main objective of this work is to implement under the Vivado 2015.4 environment on a FPGA (Field Programmable Gate Array) circuit a hybrid method between wavelets and neural networks whose goal is the compression of medical images MRI. For this purpose we have studied the wavelet transform based on the lifting schema approach and the SOM map architecture in order to implement it on an FPGA (ZedBoard) circuit.

ملخص

الهدف الرئيسي من هذا العمل هو تنفيذ تحت بيئة فيفادو 2015 علا دارة (حقل برمجة بوابة مجموعة) طريقة هجينة بين الموجيات والشبكات العصبية التي تهدف إلى ضغط الصور الطبية لتصوير بالرنين لهذا الغرض قمنا بدراسة تحويل الموجيات على أساس نهج مخطط الرفع و سوم خريطة المغنطيسي الهندسة من أجل تنفيذها على الدائرة fpga (زدبوارد)

Table de la matière

Table de figures

Table de tableaux

Liste des abréviations:

AL	**All Programmable**
Bpp	**bits per pixel**
DCT	**Discrete Cosine Transform**
DDR	**Double Data Rate**
DWT	**Discrete Wavelet Transform**
Elf	**Executable and Linkable Format**
FPGA	**Field Programmable Gate Array**
GLA	**General Lloyd Algorithme**
Hdl	**Hardware Design Language**
ILDWT	**Inverse Lifting Discrete Wavelet Transform**
JTAG	**Joint Test Action Group**
LDWT	**Lifting Discrete Wavelet Transform**
PSNR	**Peak Signal to Noise Ratio**
PL	**Programmable Logic**
PS	**Processing System**
QS	**Quantification Scalaire**
QV	**Quantification Vectorielle**
RGB	**Red, green, blue**
SSIM	**Structural Similarity**
SOM	**Self Organizing Maps**
USB	**Universal Serial Bus**

Introduction générale

Les images font partie intégrante de la vie humaine, l'image s'intègre naturellement dans notre environnement quotidien, nous synthétisons des images à des fins artistiques ou autres. Nous générons et interprétons des images à des fins scientifiques comme les images médicales.

Les images médicales ne cessent de se développer en donnant une représentation de plus en plus précise des différentes parties du corps humain. Cependant plus l'image est précise plus la quantité des données engendrées est grande. Les images médicales tel que l'IRM, l'image tomographique et l'échocardiographie 3D dynamique sont de plus en plus utilisées, elles sont considérées parmi les techniques les plus performantes en imagerie médicale, mais elles produisent des données volumiques, ainsi que la plupart des antécédents médicaux d'un patient doivent être conservés et stockés, car la législation exige que tous les renseignements sur les soins de santé enregistrés soient conservés pendant une certaine période (généralement de 5 à 10 ans) [41] avant de pouvoir être supprimés. Ainsi, l'hôpital doit faire face à des exigences de stockage très élevées, d'où la nécessité de leur compression pour des fins de stockage et de transport via des réseaux de télécommunication.

Depuis plusieurs années, de nombreuses techniques de compression des images sont mis en œuvre, il existe deux types de compression: *Compression sans perte* où la reconstruction des données est identique à l'original mais le taux de compression est très faible et la *compression avec perte* où le taux de compression est très élevé mais avec perte d'information.

Parmi les transformations qui ont été utilisées pour la compression d'images, on peut citer la transformation de Karhunen-Loéve (KLT), la transformation en cosinus discrète (DCT) et les transformations en ondelettes discrètes (DWT). Contrairement à la KLT et à la DCT, les transformations en ondelettes discrètes (DWT) demandent moins de calculs, et cela présente un avantage à cette dernière. Les travaux récents montrent l'efficacité d'utilisation de la LDWT (Lifting discrete wavelet transform) pour l'implémentation hardware.

La compression avec les Algorithmes neuronaux fait partie des méthodes de compression d'images irréversibles (avec perte), les approches utilisant les réseaux de

neurones artificiels pour le traitement intelligent des données semblent être très prometteuses, ceci est essentiellement dû à leurs structures offrant des possibilités de calculs parallèles ainsi que l'utilisation du processus d'apprentissage permettant au réseau de s'adapter sur les données à traiter. Pour la quantification, les méthodes d'apprentissage compétitif permettent de construire rapidement un dictionnaire. Avec les cartes auto-organisées on dote le dictionnaire d'une structure topologique robuste pour le système.

Il n'est pas toujours probable de faire fonctionner le système par l'utilisation d'un logiciel utilisé sur un ordinateur à usage général puisque les ressources de mémoire, de CPU et de périphériques dans les ordinateurs sont limitées. Dans la plupart des applications de traitement d'image, des dizaines d'opérations sont effectuées sur chaque pixel. Que ces opérations soient effectuées par des processeurs à vocation universelle entraînant séquentiellement des conséquences négatives en termes de consommation de ressources et de performance. Cependant, les FPGA (Field ProgrammingGateArrays) ont la capacité d'opérer de manière parallèle en termes de matériel, ce qui les distingue des processeurs traditionnels. De cette façon, les opérations sont divisées en étapes sur une carte FPGA et plusieurs opérations peuvent se faire simultanément.

L'objectif de 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 d'une méthode qui utilise les transformées en ondelettes et les réseaux de neurones de Kohonen pour la compression des images médicales IRM sur un circuit programmable de type **FPGA (Carte ZedBoard).**

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

Le premier chapitre état de l'art.

Le deuxième chapitre présente une étude sur les ondelettes et réseau de neurone pour la compression des images.

Le troisième chapitre porte sur la théorie de fonctionnement de méthode de compression dans ce mémoire (d'une méthode hybride entre les transformées en ondelettes et les réseaux de neurones pour la compression des images médicales avec pertes.), et leurs implémentations software sous l'outil MATLAB.

Le dernier chapitre est consacré à l'implémentation hardware sur un circuit programmable de type FPGA (la carte ZedBoard), les résultats de simulation ainsi que les performances temporelles.

Nous terminons par une conclusion générale.

Chapitre 1: Etat de l'art de la compression des images

1.1 Introduction

L'imagerie médicale regroupe les moyens d'acquisition et de restitution d'images du corps humain à partir de différents phénomènes physiques tels que l'absorption des rayons X, la résonance magnétique nucléaire, la réflexion d'ondes ultrasons ou la radioactivité .

Ces technologies ont révolutionné la médecine grâce au progrès de l'informatique en permettant de visualiser indirectement l'anatomie, la physiologie ou le métabolisme du corps humain. Développées comme outil diagnostique [1].

De ce fait, le traitement d'image est devenu une discipline nécessaire, mettant en place un ensemble de techniques permettant d'améliorer leur qualité ou d'extraire des informations et de compressée des donnée

La compression d'image est une application de la compression de donnée sur des images numériques. Cette compression a pour utilité de réduire la redondance des données d'une image afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou la transmettre rapidement

Ce chapitre sera divisé en deux axes, le primer s'intéresse aux prétraitements d'images (notions de base), et le second aux techniques de compression de ces dernières

1.2 L'image

L'image est une représentation d'une personne ou d'un objet peut être obtenue soit à partir de capteurs optiques (caméra, scanner…) ou créée à partir de logiciels C'est aussi un ensemble de structure d'informations qui, après affichage su l'écran, a une signification pour l'œil humain [2].

Chaque point de l'image peut être représentée par une fonction f (x, y, λ, t) ou x et y sont les coordonnées spatiales d'un point de l'objet décrit dans un système de coordonnées cartésiennes, λ est la longueur d'onde rayonnée par le point et t le temps[3].

1.2.1 Image numérique

Pour faire un traitement sur une image avec l'outil informatique, il faut quelle soit numériser. L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés pixels. La numérisation d'une image est la conversion de celle-ci en une image numérique (c'est-à-dire à partir d'une image à distribution continue d'intensité, vers une image numérique) représentée par une matrice bidimensionnelle à valeur numérique f(x, y) où x, y correspondent aux coordonnées cartésiennes (discrètes) d'un point de l'image, et f(x, y) la valeur de ce point [4].

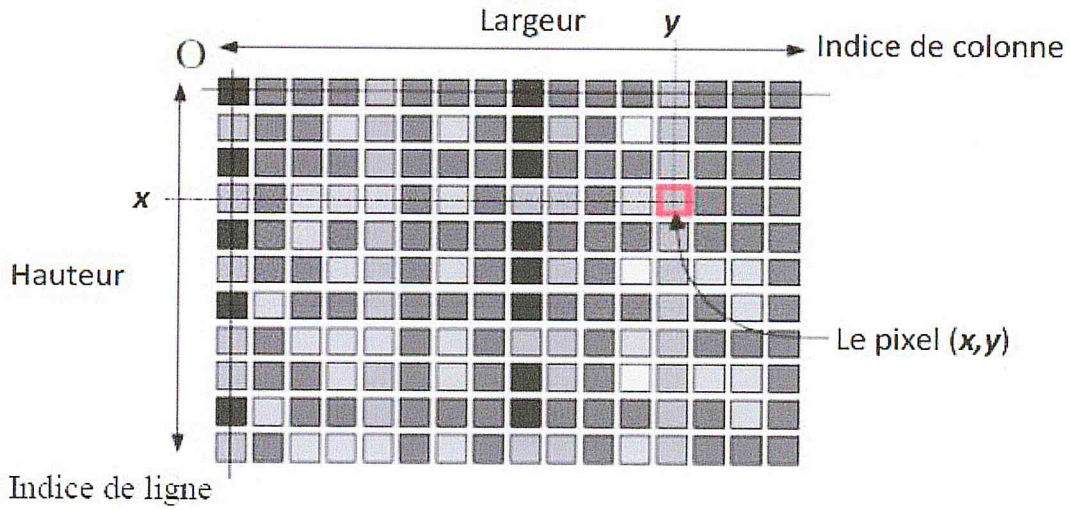


Figure 1.1 : Image numérique

1.2.2 Format d'images

Il existe deux formats :

1.2.2.1 Format matricielle

Une image matricielle (ou bitmap) est une image constituée d'un ensemble de points : les pixels. Chaque point porte des informations de position et de couleur. utilisées dans le domaine du traitement et de l'analyse d'images ; ce sont celles qui seront décrites dans ce support [5].

1.2.2.2 Format vectorielle

Les images vectorielles sont composées de formes géométriques qui vont pouvoir être décrites d'un point de vue mathématique. Utilisées principalement dans le monde du graphisme et de la conception assistée par ordinateur [5].

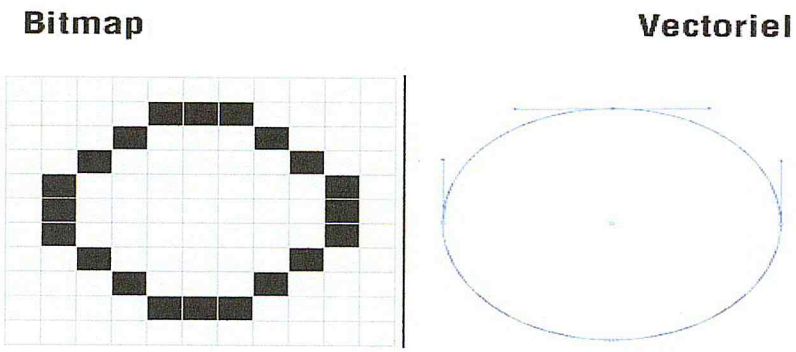


Figure 1.2 : Format matriciel et vectoriel

1.2.3 Type d'images

1.2.3.1 Image binaire

Avec ce mode, il est possible d'afficher uniquement des images en deux couleurs: noir et blanc. Chaque pixel peut donc avoir deux couleurs possibles : soit noir ou soit blanc.

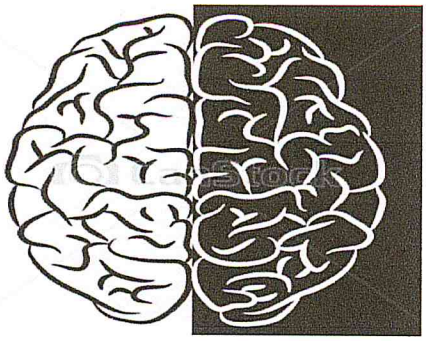


Figure 1.3 : Image binaire

1.2.3.2 Image en niveau de gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Donc pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255.

Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la " couleur " de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux [6].

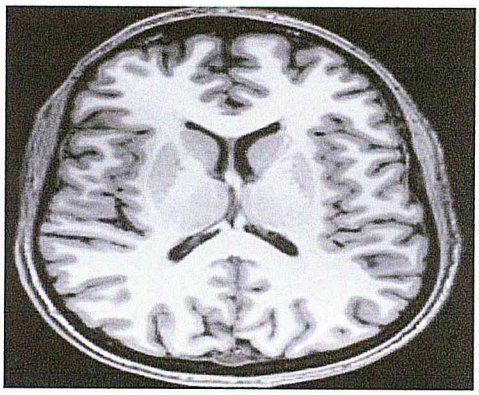


Figure 1.4 : Image au niveau de gris

1.2.3.3 Image couleur

L'œil humain analyse la couleur à l'aide de trois types de cellules photo réceptrices *'les cônes'*. Ces cellules sont sensibles aux basses, moyennes, ou hautes fréquences (rouge, vert, bleu). Pour représenter la couleur d'un pixel, il faut donc donner trois nombres, qui correspondent au dosage de trois couleurs de base : Rouge, Vert, Bleu. On peut ainsi

représenter une image couleur par trois matrices chacune correspondant à une couleur de base [2].

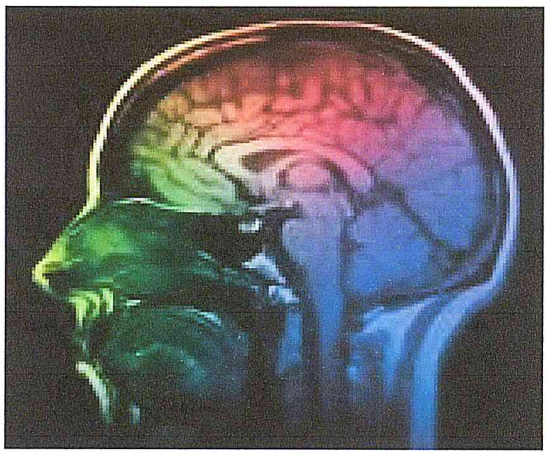


Figure 1.5 : Image couleur

1.2.4 Caractéristiques de l'image

Comme nous l'avons vu, l'image est un ensemble structuré d'informations parmi ses caractéristiques nous pouvons citer les paramètres suivants:

1.2.4.1 Pixel

Le pixel est l'abréviation du mot « Picture élément », est une unité de surface permettant de définir la base d'une image numérique. Il matérialise un point donné (x, y) du plan de l'image. L'information représentée par le pixel est le niveau de gris (ou la couleur) prélevée à l'emplacement correspondant dans l'image réelle [4].

La différence entre image monochrome et image couleur réside dans la quantité d'informations contenue dans chaque pixel, par exemple dans une image couleur (RVB : Rouge, Vert, Bleu) la valeur d'un pixel est représentée sur trois octets pour chaque couleur.

1.2.4.2 Voisinage

Le voisinage d'un pixel est formé par l'ensemble des pixels qui se situent autour de ce même pixel. On distingue deux types de voisinage :

- **Voisinage à 4 :** On ne prend en considération que les pixels qui ont un côté commun avec le pixel considéré.

- **Voisinage à 8 :** On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré

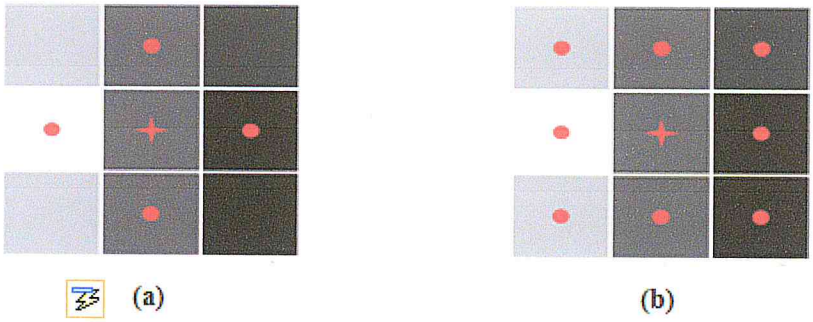


Figure 1.6 : (a) voisinage à 4, (b) voisinage à 8

1.2.4.3 Niveau de gris

C'est la valeur d'intensité lumineuse d'un pixel. Cette valeur peut aller du noir (0) jusqu'au blanc (255) en passant par les nuances qui sont contenues dans l'intervalle [0, 255]. Elle correspond en fait à la quantité de la lumière réfléchie.

Pour 8 bits, on dispose de 256 niveaux de gris dont. Plus le nombre de bit est grand plus les niveaux sont nombreux et plus la représentation est fidèle [5].

1.2.4.4 Définition

La définition est le nombre de points (ou pixels) que comporte une image numérique en largeur et en hauteur. On l'exprime en donnant le nombre de pixels en hauteur et en Largeur (exemple : 1600x1200) [7].

1.2.4.5 Résolution

La résolution est le nombre de pixels par unité de longueur. est exprimée le plus souvent en ppp (point par pouces) ou en dpi (dots per inch), avec: 1 pouce = 2.54 cm.

La résolution définit la netteté d'une image et sa qualité d'affichage à l'écran. Plus la résolution est grande (c'est-à-dire plus il y a de pixels dans une longueur de 1 pouce), plus votre image est précise dans les détails [7].

Résolution = définition / dimension.

1.2.4.6 Contour

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes [8].

1.2.4.7 Luminance

C'est le degré de luminosité des points de l'image. Est substitué au mot brillance, qui correspond à l'éclat d'un objet. Une bonne luminance se caractérise par :

1. Des images lumineuses (brillantes).

2. Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir, ces images entraînent des pertes de détails dans les zones sombres ou lumineuses [6].

1.2.4.8 Histogramme

L'histogramme est un vecteur Chaque élément de ce vecteur (noté h(i)) représente le nombre de pixels de l'image possédant le niveau de gris « i » (sa fréquence d'apparition), on peut donc assimiler l'histogramme à la densité de probabilité des intensités lumineuses. Puisque le nombre des pixels est généralement assez grand, on peut alors normaliser l'effectif de chaque niveau de gris en divisant chaque terme du tableau par la surface du plan exprimée en nombre total des pixels de l'image. De ce fait, chaque case h(i) du vecteur représente la probabilité d'avoir l'intensité « i ».

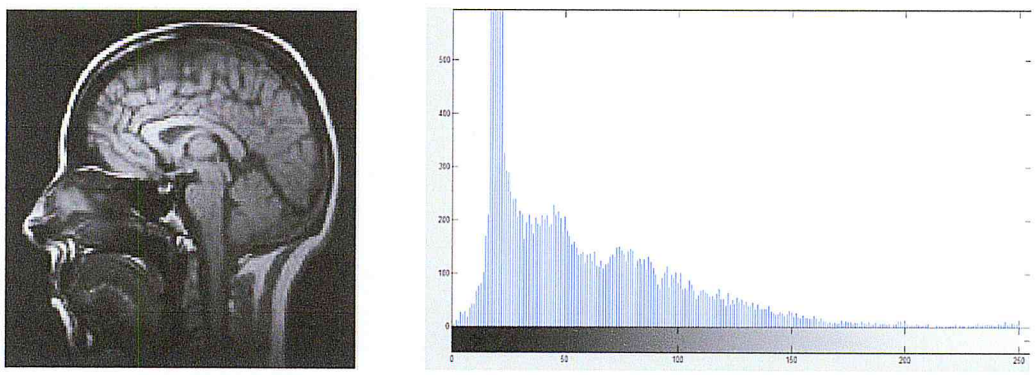


Figure 1.7 : L'histogramme

1.3 La compression

La compression a pour but de réduire le nombre de bits codant une image. Cela est possible grâce à une redondance de l'information généralement présente dans une image. Cette redondance peut être d'origine statique, spatial ou encore fréquentielle. Toutes les techniques de compression d'image essayent en général de tirer de cette redondance.

On différencie généralement les méthodes de compression selon la distorsion, ou la perte d'informations, qui est engendrée. Les méthodes réversibles exploitent uniquement le principe de la réduction de la redondance et n'engendrent pas de perte. Les méthodes irréversibles définissent une représentation approximative de l'information et occasionnent une perte [18].

1.4 Les méthodes de la compression

Il existe deux types de compression:

1.4.1 La compression sans perte

Ces méthodes dites réversibles permettent de coder les données de manière plus concise dans un fichier. Elles réduisent les redondances dans les données, sans en modifier le contenu.

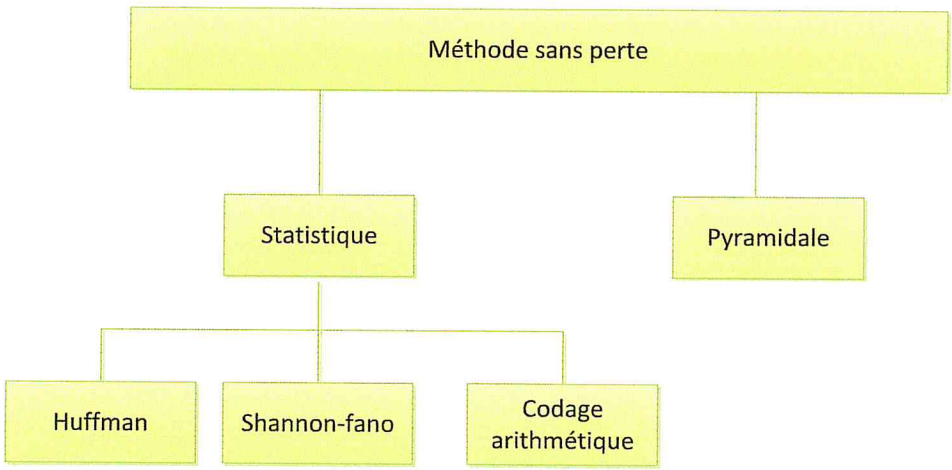


Figure 1.8 : méthodes de compression sans perte

1.4.1.1 Hufman

Le codage d'Huffman permet d'associer à chaque symbole un code optimisé. La probabilité d'occurrence du symbole dans le message est prise en compte en associant un code le plus court possible pour un symbole fréquent [18].

➢ **Procédure de codage**

1. Les probabilités d'occurrence de chaque message sont déplacées dans une liste dans un ordre décroissant.
2. Les deux probabilités les plus faibles sont identifiées en fin de la liste.
3. La somme des deux probabilités est placée a sa place dans la liste triée elle constitue un nœud parente les deux enfants son retirés de la liste.
4. Le chemin « enfant de plus faible probabilité, parent » est codé par un 1, l'autre par un 0.
5. La procédure reprend a l'étape 2 jusqu'à ce qu'il ne reste plus qu'une probabilité dans la liste.

Soit un message de 36 caractères, composé des caractères A, B, C, D et E qui apparaissent selon les fréquences suivantes:

- Symboles : A B C D E
- Fréquences : 7 6 5 14 4

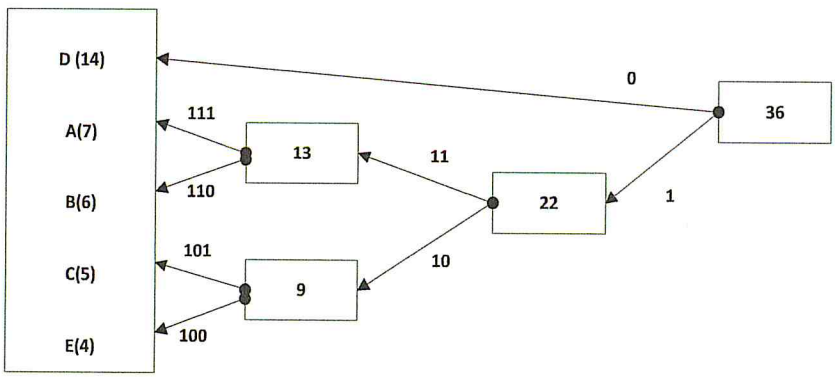


Figure 1.9 : Principe de Huffman

Le codage obtenu est donc :

- D (occurrence =14) : 0.
- A (occurrence =7) : 111.
- B (occurrence =6) : 110.
- C (occurrence =5) : 101.
- E (occurrence =4) : 100.

1.4.1.2 Shannon-Fano

Ce codage suit le même principe que le codage d'Huffman, à la différence que l'algorithme part des racines et aboutit aux feuilles par divisions successives. Il classe les symboles par ordre de probabilité décroissante. L'ensemble est divisé en deux sous-ensembles de probabilités aussi proches que possibles. A chacun des deux ensembles est affecté le code 0 ou 1. Puis, le processus est réitéré jusqu'à ce que chaque ensemble ne soit composé que d'un seul élément.

Le procédé de Shannon-Fano construit un arbre descendant à partir de la racine, par divisions successives. Le classement des fréquences se fait par ordre décroissant, ce qui suppose une première lecture du fichier et la sauvegarde de l'entête. Le principe est le suivant :

➢ **Procédure de codage**

1. Classer les n fréquences non nulles par ordre décroissant.
2. Répartir la table des fréquences en deux sous tables de fréquences proches. Poursuivre l'arborescence jusqu'à ce que toutes les fréquences soient isolées.
3. Attribuer dans l'arborescence le bit 0 à chaque première sous table.
4. Attribuer aux symboles les codes binaires correspondant aux bits de description 1 de l'arborescence.

1.4.1.3 Codage arithmétique

Cette méthode attribue à une suite de symboles une valeur réelle. Elle consiste à diviser l'intervalle des réels [0,1[en sous-intervalles, dont les longueurs sont fonctions des probabilités des symboles.

L'exemple ci-dessous montre la construction d'un message eaii! selon la table de partitions réalisée pour l'alphabet a, e, i, o, u.

Symboles	Probabilité	Partition initiale
a	0,2	$[0 \quad 0,2[$
e	0,3	$[0,2 \quad 0,5[$
i	0,1	$[0,5 \quad 0,6[$
o	0,2	$[0,6 \quad 0,8[$
u	0,1	$[0,8 \quad 0,9[$
!	0,1	$[0,9 \quad 1[$

Figure 1.10 : Le principe de codage arithmétique

Au fur et à mesure du codage, la longueur de l'intervalle diminue en tenant compte du modèle et du sous-intervalle précédent.

- On débute avec l'intervalle [0 1[.
- Le premier symbole à coder est e, on obtient l'intervalle [0, 2 0, 5[.
- Le second symbole est a, ce qui donne l'intervalle [0, 2 0, 26[.
- Le troisième symbole i forme l'intervalle [0, 23 0, 236[, Finalement tout réel dans l'intervalle [0, 23354 0, 2336[codera le message eaii! [19].

Le codeur arithmétique est très répandu dans la littérature, il est notamment utilisé dans le standard Jpeg2000.

1.4.2 La compression avec perte

Elle consiste en une « Réduction » de l'information basée sur notre propre limite humaine à percevoir ces médias. Puisque l'œil ne perçoit pas nécessairement tous les détails d'une image, il est possible de réduire la quantité de données de telle sorte que le résultat soit très ressemblant à l'original, voire identique, pour l'œil humain.

Le schéma général souvent utilisé pour décrire le fonctionnement des algorithmes de compression est celui présenté dans la figure (1.11).

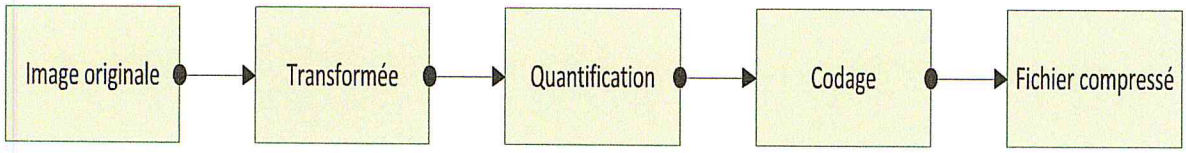


Figure 1.11 : Schéma générale de la compression avec pertes

1.4.2.1 La transformée

Les transformations les plus utilisées en compression, que ce soit pour les images fixes ou les séquences d'images, sont la Transformée en Cosinus Discrète (DCT), la Transformée en Ondelettes Discrète (DWT). Quelque soit la transformation, la variance (l'énergie) des composantes transformées est concentrée sur les basses fréquences et les composantes de faible variance, concentrées sur les hautes fréquences sont annulées par quantification.

1.4.2.1.1 L'histoire de l'ondelette

La première transformation en ondelettes - le nom n'est pas encore utilisé - est proposée par ***Haar*** en 1910 ; il serait plus judicieux de parler alors de «pré-ondelette».

La transformée en ondelettes est un outil qui découpe les données, les fonctions sous les opérateurs en composantes fréquentielles suivant une résolution adaptée à l'échelle. Les précurseurs conscients de cette technique ont été des mathématiciens (***Calderon*** 1964), des physiciens (***Aslaken*** et ***Klauderen*** 1968, ***Paul*** en 1985), et surtout des ingénieurs (ou des chercheur de sciences pour l'ingénieur) comme ***Esteban*** et ***Galand*** (1977), ***Smith*** et ***Barnwell*** (1986), ***Vetterli*** (1986). Nous pourions parler dans leur cas de «pré-ondelette». Mais le premier à avoir utilisé la méthode et le premier à avoir proposé le nom d'ondelettes fut ***Jean Morlet*** (1983) [20].

Le problème traité par ***Morlet*** était celui de l'analyse de données issues des ondes sismiques effectués pour des recherches géologiques ; ces données faites de nombreuses transitoires sont particulièrement adaptées à une technique d'analyse conservant la notion de localisation de l'événement tout en fournissant une information sur son contenu fréquentiel ce qui est tout l'intérêt de ce type de transformation. Les résultats obtenus par ***Morlet*** et formalisés par le physicien ***Alex Grossmann*** ont rapidement éveillé l'attention de nombreux chercheurs et bien tôt des bases mathématiques solides ont été mises en place faisant apparaître la notion de base orthogonale (***Y.Meyer*** 1985), d'analyse multi résolution (***S.Mallat***

1989) et d'ondelettes à support compact (***I.Daubechies*** 1988). Les ondelettes modernes étaient nées [21].

1.4.2.1.2 Intérêt d'ondelettes

La plupart des signaux du monde réel ne sont pas stationnaires, et c'est justement dans l'évolution de leurs caractéristiques (statistiques, fréquentielles, temporelles, spatiales) que réside l'essentiel de l'information qu'ils contiennent. Les signaux vocaux et les images sont à ce titre exemplaire. Or l'analyse de Fourier propose une approche globale du signal, les intégrations sont faites de moins l'infini à plus l'infini, et toute notion de localisation temporelle (ou spatiale pour des images) disparaît dans l'espace de Fourier; il faut donc trouver un compromis, une transformation qui renseigne sur le contenu fréquentiel tout en préservant la localisation afin d'obtenir une représentation temps/fréquence ou espace/échelle du signal [22].

1.4.2.2 La quantification

La quantification intervient dans la conversion analogique-numérique, Mais elle est aussi présente en compression d'image avec pertes. Dans ce Contexte il s'agit alors de supprimer l'information non visible (ou peu importante).

La quantification d'une grandeur x consiste a l'approximer par une valeur y choisie dans un ensemble fini **C = f {yi i = 1; .....N}**, **C** est le dictionnaire et les valeurs de reconstruction **yi** sont des valeurs codes. Quand on quantifie une seule valeur à la fois (pixel, coefficient de transformé), on parle de quantification scalaire. Quand on quantifiée plusieurs valeurs a la fois, x est un vecteur, les yi sont appel les vecteurs codes et il S'agit de quantification vectorielle.

La QV est donc une généralisation de la Quantification scalaire [33].

1.4.2.2.1 La quantification scalaire

Les techniques de compression d'images exploitent généralement la redondance statistique présente dans l'image. La quantification associe à une variable continue x une variable discrète y pouvant prendre un nombre plus faible et fini de valeurs.

Un exemple de quantification figure (1.12), où l'on distingue les niveaux de quantification y_i (avec $1 \leq i \leq N$), et les seuils de quantification x_j (avec $0 \leq j \leq N$), qui

délimitent les intervalles de quantification. Un quantificateur est défini par l'ensemble de ses niveaux et de ses seuils de quantification.

Shannon a montré qu'il était toujours possible d'améliorer la compression de données en codant des vecteurs plutôt que des scalaires.

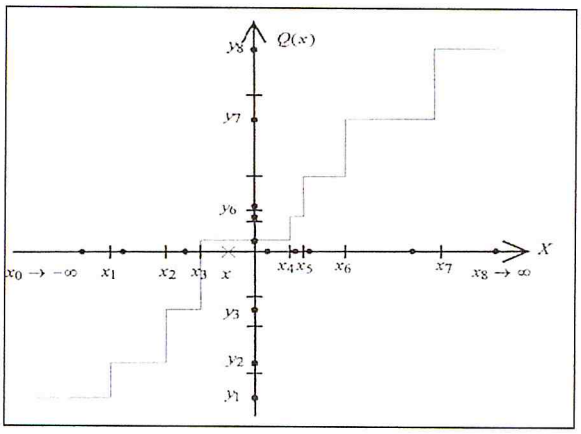


Figure 1.12 : Principe de QS

L'inconvénient de la quantification scalaire est qu'elle ne permet pas d'exploiter la corrélation spatiale qui existe entre les différents pixels de l'image.

1.4.2.2.2 Quantification vectorielle

La quantification vectorielle VQ-(Vector Quantization) a été développée par Gersho et Gray et elle fait aujourd'hui l'objet de nombreuses publications dans le domaine de la compression numérique. Le principe de la quantification vectorielle est issu du travail de Shanon qui montre qu'il était toujours possible d'améliorer la compression de données en codant non pas des scalaires, mais des vecteurs. Un quantificateur vectoriel Q associe à chaque vecteur d'entrée $X_i = (x_j, j=1\ldots k)$ un vecteur $Y_j = (y_j, j=1\ldots k) = Q(X_i)$, ce vecteur Y_i (qui correspond à une distance euclidienne minimum) étant choisi parmi un dictionnaire (code book) de taille finie. La *VQ* produit de meilleurs résultats que la *SQ* en terme de taux de compression, néanmoins la *VQ* nécessite un codage complexe et de grandes capacités de mémoire en plus elle est très gourmande en temps de calcule [31].

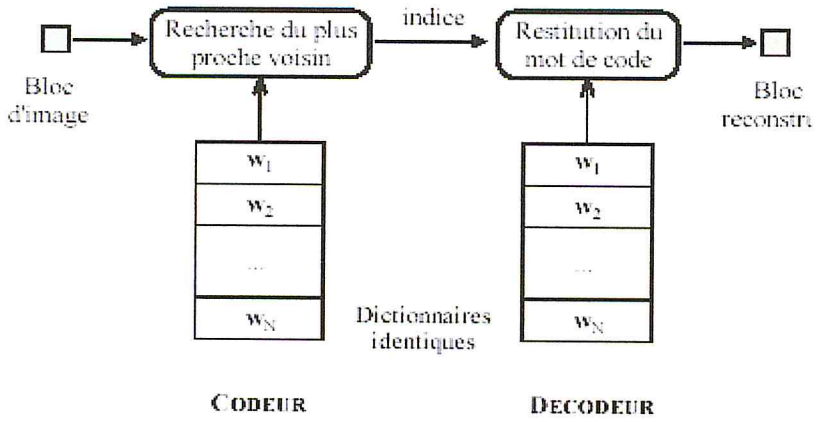


Figure 1.13 : Principe de QV

1.4.2.2.3 Construction de dictionnaire

La construction de dictionnaire est considérée comme un problème d'optimisation. Les algorithmes de construction fonctionnent à partir d'un dictionnaire initial qu'ils modifient itérativement en cherchant à minimiser la distorsion moyenne.

1.4.2.3 Les algorithmes non-neuronaux

Il existe deux classes d'algorithmes de construction de dictionnaire. Dans la première, le dictionnaire n'est modifié qu'après traitement de la base d'apprentissage entière. On parle alors de traitement par lot car les vecteurs d'apprentissage ne sont pas traités individuellement. .Dans la seconde famille, on modifie un ou plusieurs vecteurs codes pour chaque vecteur d'apprentissage.

Trois algorithmes sont présentés ci-dessous : le GLA, le splitting et l'agrégation constructive progressive.

1.4.2.3.1 Algorithme de Lloyd généralise (GLA)

L'algorithme de Lloyd généralise (GLA) est l'algorithme le plus simple, auquel on se réfère pour comparer d'autres méthodes. Il est si répandu qu'on le rencontre dans la littérature sous différentes appellations : k-means ou Linde-Buzo-Gray(LBG) [18] .Il procède par amélioration successive d'un dictionnaire initial par application de deux conditions

d'optimalité. Ces conditions sont nécessaires et suffisantes pour minimiser la distorsion moyenne. La première définit la partition optimale pour un dictionnaire donné. C'est la condition du plus proche voisin.

Inconvenant de cet algorithme est la détermination du dictionnaire initial qui est un problème pour construction de dictionnaire.

1.4.2.3.2 Splitting

Cet algorithme a pour objectif d'être invariant par rapport au dictionnaire initial. Pour cela, on applique le GLA sur des tailles croissantes de dictionnaire. Le dictionnaire initial ne contient qu'un vecteur code. Afin de minimiser la distorsion moyenne, sa valeur est celle du centroide de la base d'apprentissage. Ce premier vecteur code est éclate en deux autres. Les deux nouveaux vecteurs codes forment un dictionnaire de deux éléments, sur lequel on exécute le GLA afin d'optimiser le positionnement des vecteurs codes. Une fois ce dictionnaire stabilisé, on éclate de nouveau chacun des vecteurs codes afin de produire un dictionnaire à quatre éléments sur lequel sera exécuté le GLA. Le processus est itèrè jusqu'a atteindre le nombre N (une puissance de deux) voulu de vecteurs codes.

Il a pour objectif la rapidité par le parcours de la base d'apprentissage en une seule passe. Pour cela, les vecteurs codes et le partitionnement de la base d'apprentissage sont dénis progressivement en parcourant la base d'apprentissage.

1.4.2.3.3 QV adaptative par réapprentissage

Une manière d'améliorer la qualité de reconstruction consiste à faire évoluer le dictionnaire au cours du temps par un apprentissage sur des blocs. Comme le dictionnaire initial est déjà assez proche de l'optimum, cet apprentissage est généralement plus court qu'un apprentissage complet. Comme pour les algorithmes de construction de dictionnaire statique,

Le changement de dictionnaire peut porter sur tous les vecteurs codes, ou seulement sur quelques-uns [31].

1.4.2.4 Les algorithmes neuronaux

Les algorithmes neuronaux d'apprentissage dit compétitif sont particulièrement adaptés au problème de la construction de dictionnaire. Dans le domaine des réseaux de neurones artificiels, ils entrent dans la catégorie des apprentissages non supervisés. Cela signifie qu'ils se forment une représentation de l'espace d'entrée à partir de la simple observation des vecteurs d'exemple, sans information a priori [31] sur la sortie attendue du réseau.

Les approches utilisant les réseaux de neurones artificiels pour le traitement intelligent des données semblent être très prometteuses, ceci est essentiellement dû à leurs structures offrant des possibilités de calculs parallèles ainsi que l'utilisation du processus d'apprentissage permettant au réseau de s'adapter sur les données à traiter [17].

1.5 Les critères d'évolution de méthode de compression

Ils donnent une mesure de performance de la méthode de compression utilisée. Les principaux critères d ´évaluation de toute méthode de compression sont :

- Le taux de compression.
- La qualité de reconstruction de l´image.
- La rapidité du codeur et du décodeur.
- La robustesse aux erreurs de transmission.

1.5.1 Le taux de compression

Le rapport de compression est l'une des caractéristiques les plus importantes de toutes les méthodes de compression, il est défini comme :

$$\mathbf{Rc} = \frac{\textbf{Espace mémoire occupé par les données non compressées}}{\textbf{Espace mémoire occupé par les données compressées}} \qquad (1.1)$$

On défini alors le taux compression (pourcentage) par :

$$\mathbf{Tc} = \mathbf{100x}\left(\mathbf{1} - \frac{\mathbf{1}}{\mathbf{Rc}}\right) \qquad (1.2)$$

1.5.2 L'évolution de la distorsion

On utilise deux méthodes, une subjective l'autre objective :

1- La méthode subjective est basée sur des testes psycho-visuels de l'œil humaine. En imagerie médicale, l'avis d'un médecin spécialiste est indispensable pour confirmer la validité de l'image compressée vis-à-vis le diagnostique.

2- Pour la méthode objective, on utilise souvent le calcul de l'erreur quadratique moyenne MSE (Mean Square Error), elle est définie par la moyenne des écarts au carré entre les pixels de l'image originale (Xi) et l'image reconstruite (xi) comme suit : $$\mathbf{MSE} = \frac{1}{R}\sum_{i=1}^{R}(X_i - x_i)^2 \quad (1.3)$$

- R : représente la résolution de l'image (nombre total de pixels)

On peut alors définir **Le rapport crête signal sur bruit** PSNR (Peak Signal to Noise Ratio) qui mesure la fidélité de la compression puisque il relie le MSE a la dynamique de l'image :

$$\mathbf{PSNR} = 10 \times \log_{10}\left(\frac{D^2}{MSE}\right) \mathbf{dB} \quad (1.4)$$

- D : est la valeur maximum que pourrait prendre un pixel ; $D = 2^{BD} - 1$

  BD (Bit Depth) : nombre de bit sur le quel les pixels sont codés.

1.5.3 Similarité structurelle

Basé sur la distorsion structurelle, le SSIM (Structural Similarity) est utilisé comme indicateur de la qualité de l'image compressée, il fournit un moyen pour quantifier la similarité perceptuelle entre deux images.

Les méthodes traditionnelles sont basées sur le calcul d'erreur entre les valeurs de chaque pixel d'une image distordue et une autre référence, par contre le SSIM mesure la similarité de structure de ces deux images. Ceci donne de très bons résultats, comme présenté dans l'article [34].

Le SSIM des images x et y est défini à partir de la moyenne (μ), la variance (σ) de chaque image, et aussi de leur covariance, comme suit :

$$\mathbf{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\,cov_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (1.5)$$

Avec : C_1 et C_2 sont des constantes dans : $C_1 = (K_1L)^2$ et $C_2 = (K_2L)^2$

L est la dynamique des valeurs des pixels. (Soit 255 pour une image codée sur 8 bits). En utilisant l'expression SSIM précédente sans les constantes C_1 et C_2, les valeurs calculées deviennent instable. Ce problème a été résolu en ajoutant ces deux petites constantes (Calculées on posant $K_1 = 0.01$ et $K_2 = 0.03$).

Le SSIM satisfait les conditions suivantes :

- La symétrie : SSIM (x,y) = SSIM (y,x)
- SSIM (x,y) ≤ 1.
- Unicité du maximum : SSIM (x,y) = 1 si et seulement si x = y.

Le SSIM est calculé (en utilisant une fenêtre courante qui parcourt l'ensemble de l'image pixel-par-pixel) pour chaque bloc de l'image de même taille que la fenêtre utilisée. Cette opération nous donne la carte SSIM (une matrice de dimension inférieure) qui représente la carte qualité de l'image distordue (en supposant que l'autre image a une qualité parfaite). Finalement, le SSIM moyen (moyenne de la carte SSIM) est utilisé pour évalué la qualité globale de l'image.

1.6 Conclusion

La connaissance et la compréhension des différentes caractéristiques d'une image permettent d'augmenter la qualité de celle-ci en appliquant les méthodes de traitement appropriées telles que l'amélioration la restauration et la compression.

Dans ce chapitre nous avons présenté des méthodes dédiées à la compression des images selon leurs approches et mettre en évidence la diversité de ces techniques utilisées. La diversité des techniques est représentative de la complexité du problème de compression d'images IRM, ces difficultés sont liées, d'une part, aux objets traités et, d'autre part, aux particularités des images IRM.

Le chapitre suivant est consacré à l'étude la méthode destinée à la compression des images médicales Dans notre travail (les ondelettes et réseau de neurone).

Calculer cette fonction $c_f(a,b)$ est faire l'analyse de f par l'ondelette ψ. La fonction f est alors décrite par ses coefficients d'ondelettes. Ils mesurent les fluctuations à l'échelle a, de la fonction f.

Cette transformée consiste donc à décomposer un signal en une famille de fonctions localisées en temps et en fréquence.

2.2.1 Transformée en ondelettes continues

Les transformées continues sont obtenues en prenant le facteur d'échelle a et le pas de translation b dans l'ensemble des nombres réels, ces transformées sont évidemment très redondantes car l'espace-temps-fréquence est parcouru continûment, ce type de transformation ne peut, dans la pratique, être effectué que de façon approximative et il y a toujours en fait une discrétisation du calcul qui est opérée [22].

Dans le cas unidimensionnel, la TOC d'un signal f(x) est donc obtenue par l'expression de (eq 2.2) :

$$\forall(a,b) \in R^* \times R, W_f(a,b) = \frac{1}{a}\int_R f(x)\,\Psi^*\left(\frac{x-b}{a}\right)dx \qquad (2.2)$$

Avec:

- ψ (x) est l'ondelette mère analysante.
- a est le paramètre d'échelle.
- b est le paramètre de position.
- W_f (a,b) est le résultat de la transformée donné par un ensemble des coefficients appelés coefficients d'ondelette de la fonction f(x) dans l'espace-temps-échelle ou espace-échelle.

$$f(x) = C_\Psi^{-1}\int_{-\infty}^{+\infty} W_f(a,b)\,\Psi_{a,b}(x)\,\left(\frac{da\,db}{a^2}\right) \qquad (2.3)$$

2.2.1.1 Exemple d'ondelettes de base

Il existe de nombreuses formes d'ondelettes, le choix de l'ondelette optimale dépend de l'application envisagée. Il convient de bien cerner le problème à étudier et d'identifier le type de transformée à utiliser (continue ou discrète). En analyse d'image, il est souvent utile d'avoir une certaine redondance pour avoir plus d'informations. L'utilisation de la transformée en ondelettes continue est alors conseillée. Si on veut un calcul exact, alors les

ondelettes à support compact sont indiquées. On voit donc qu'on ne peut parler d'une ondelette idéale, adaptée à tous les cas.

2.2.1.2 Ondelette de Haar

Elle est définie par [15] :

$$\begin{cases} \Psi(x) = 1 \quad si \ 0 < x < ½ \\ \Psi(x) = -1 \quad si \quad ½ < x < 1 \\ \Psi(x) = 0 \quad sinon \end{cases} \tag{2.4}$$

La fonction échelle de ***Haar*** :

$$\begin{cases} \varnothing(x) = \varnothing(2x) + \varnothing(2x-1) \\ \Psi(x) = \varnothing(2x) - \varnothing(2x-1) \end{cases} \tag{2.5}$$

C'est-à-dire :

$$\begin{cases} \varnothing(x) = 1 \quad si \ 0 < x < ½ \\ \varnothing(x) = -1 \quad si \quad ½ < x < 1 \\ \varnothing(x) = 0 \quad sinon \end{cases} \tag{2.6}$$

- **Propriétés**

C'est une ondelette ortho-normale à support compact, symétrique. Elle permet d'obtenir une reconstruction exacte du signal. Elle est utilisée à la fois pour les transformées continue et discrète [23].

- **Intérêt**

Cette ondelette est très simple et facile à implémenter. De plus elle est à support compact. Le calcul de la transformée de Fourier est donc exact [23].

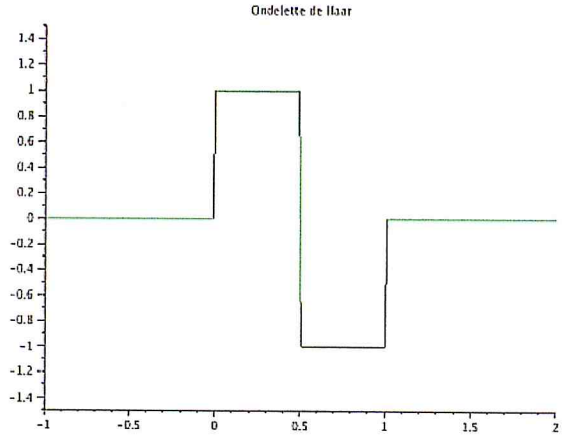


Figure 2.1 : Ondelette de Haar

2.2.1.3 Ondelette de *Daubechies*

L'ondelette de *Daubechies* est la famille la plus connue des ondelettes orthonormales. Ses ondelettes sont généralement dénommées par le nombre de coefficients a k non nuls, on parler donc d'ondelettes *Daubechies4*, *Daubechies6*, etc.

- **Propriétés**

Quand l'ordre augmente, les supports grandissent ainsi que la régularité des ondelettes [15].

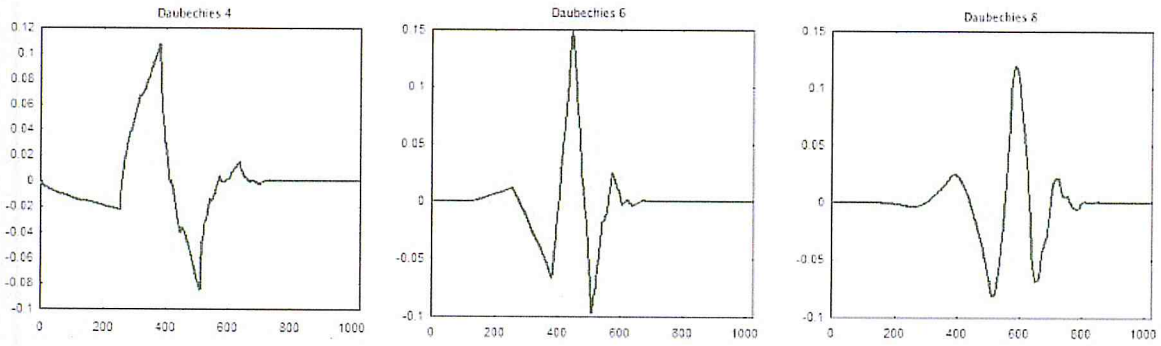


Figure 2.2 : Ondelette de Daubechies

➢ **Intérêt**

La mathématicienne ***Ingrid Daubechies*** a cherché dans ses travaux à concilier deux contraintes respectives: l'orthogonalité de la base d'ondelettes et la compacité du support de l'ondelette-mère. Ce qui implique que toute ondelette de la base est à support compact et donc que le calcul de la transformée en ondelettes est exacte. De plus, elle a imposé à ses ondelettes une troisième condition: avoir n moments nuls [22].

2.2.2 Transformée en ondelettes discrètes

La transformée en ondelettes continues présentée précédemment est obtenue en prenant le facteur d'échelle **a** et le pas de translation **b** dans l'ensemble des nombres réels. Ce type de transformation ne peut être effectué dans la pratique que de façon approximative, et il y a toujours, en fait, une discrétisation du calcul, qui est opérée. ***Morlet*** a formulé des bases construites par une discrétisation dyadique de ces paramètres sur le modèle suivant [24]

$$a=2^{-j}, b=k2^{-j} \text{ avec: } j = 1,2,\ldots..,j-1 \text{ et } k= 1,2\ldots..2^{-j}-1 \qquad (2.7)$$

Et l'ensemble des fonctions d'ondelettes analysantes seront donc

$$\begin{cases} \Psi_{(a,b)}(t) = \frac{1}{\sqrt{2}}\left(\frac{t-b}{a}\right) \\ \Psi_{(j,k)}(t) = 2^{\frac{j}{2}}.\Psi(2^{j}t-k) \\ \Psi_{(a,b)}(t) = \Psi_{(j,k)}(t) \end{cases} \qquad (2.8)$$

La transformée en ondelettes discrètes (TOD) de la fonction **f**(t) est donc en fonction de **j** et **k**, au lieu de **a** et **b**, respectivement, et entraîne un ensemble de coefficients d'ondelettes (détail) :

$$C_{j,k} = \int_{-\infty}^{+\infty} f(t)\,\Psi_{(j,k)}(t)dt \qquad (2.9)$$

Comme les fonctions d'ondelettes, il y a encore un autre ensemble de fonctions appelées fonctions d'échelle (t) qui donnent par convolution avec φ **f**(t), l'ensemble des coefficients d'approximation :

$$A_{j,k} = \int_{-\infty}^{+\infty} f(t).\emptyset(x)_{j,k}(t).dt \qquad (2.10)$$

Les fonctions d'ondelettes (détail) et d'échelles (approximation) établissent un algorithme de décomposition multi-résolution. La fonction d'ondelette est orthogonale à la fonction d'échelle à un indice d'échelle particulière **J**. Ainsi, les informations contenues dans les coefficients d'approximation d'un indice d'échelle **J** ne sont pas répétées dans les coefficients d'ondelettes. De plus, les coefficients d'ondelettes à un indice d'échelle donnée sont en fonction de la fonction d'échelle de niveau inférieur. Par conséquent, les fonctions d'approximation à un indice d'échelle donnée peuvent être reconstruites en utilisant la fonction d'approximation et les coefficients de détail de l'indice supérieur [24].

$$F_j(x) = C_{j+1}(x) + A_{j+1}(x) \tag{2.11}$$

Ainsi, les fonctions d'ondelettes et d'échelle ont la capacité d'exprimer une fonction dans une résolution inférieure. La fonction de décomposition de l'équation précédente peut s'écrire sous la forme.

$$f_0(x) = C_1(x) + A_1(x) \tag{2.12}$$

$$f_0(x) = C_2(x) + A_2(x) + A_1(x) \tag{2.13}$$

$$f_0(x) = C_3(x) + A_3(x) + A_1(x) + A_2(x) \tag{2.14}$$

La transformée en ondelettes discrètes (TOD) est devenue un outil très polyvalent de traitement de signal, après l'introduction de la représentation multi-résolutions des signaux basée sur la décomposition en ondelettes en 1987 [24].

Stéphane Mallat a mis en avant une certaine catégorie de décompositions en ondelettes, qui peuvent être réalisées numériquement en un temps très court, par « une transformée en ondelettes rapide », constituée d'une cascade de filtres passe-bas et passe-haut suivie par des opérations de sous échantillonnages par un facteur de deux.

2.2.2.1 Extension de transformée en ondelettes

Il existe plusieurs façons pour étendre la transformation en ondelettes pour un signal 2D, tel que l'image. Le procédé classique consiste à obtenir la fonction d'échelle 2D $\varphi(t)$ et la

1. décomposer d'autres sous-bandes du niveau n–1 que (n−1) LL, et ainsi former des sous-niveaux de décomposition

2. décomposer une sous-bande uniquement ligne par ligne ou colonne par colonne, et ainsi former uniquement 2 sous-bandes de niveau n pour une sous-bande donnée du niveau n−1.

3. faire un choix différent pour les points 1 et 2 à chaque résolution n.

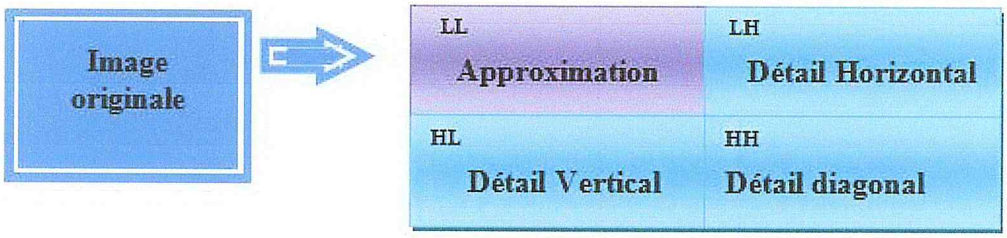


Figure 2.3 : Décomposition d'une image en sous-banes

2.2.3 La transformé en ondelettes et l'approche lifting scheme

À l'origine, le but du lifting scheme, introduit par Sweldens, était de proposer un procédé de construction d'ondelettes bi-orthogonales dont les moments s'annulent pour des ordres de plus en plus élevés : « un ascenseur » (lift) vers de hauts moments nuls [27].

C'est une alternative intéressante au schéma de filtrage convolutif classique de la transformée, car beaucoup moins complexe. En effet, le nombre d'opérations est divisé par un rapport allant jusqu'à deux en comparaison avec un schéma classique.

Le lifting scheme consiste en plusieurs étapes, comme est montré dans la figure (2.4). En premier lieu, le signal d'entrée « X » est partitionné en plusieurs composantes. Généralement ce dernier est dyadique et les échantillons d'indices pairs « X_e » et d'indices impairs « X_o » sont séparés.

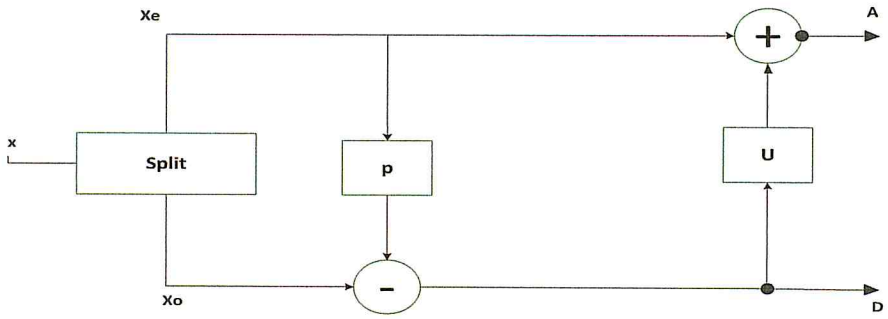


Figure 2.4 : Banc d'analyse

Puis, un opérateur de prédiction est appliqué aux échantillons pairs et le résultat est soustrait aux échantillons impairs. Cette opération élémentaire, appelée étape de prédiction (predict) ou pas primal, donne une erreur de prédiction:

$$D = x_o - P_1(x_e). \qquad (2.16)$$

L'étape de mise à jour (update), ou pas dual, du lifting scheme, modifie les échantillons pairs avec l'erreur de prédiction. Le signal mis à jour s'exprime ainsi :

$$A = x_e + U_1(d_1). \qquad (2.17)$$

La transformé inverse est réalisée par l'inverse de schémas de la figure (2.4) come est mntré dans la figure (2.5), de droite à gauche en commutant les signes de l'opération de prédiction (perdict) et de mis à jour (update) [28].

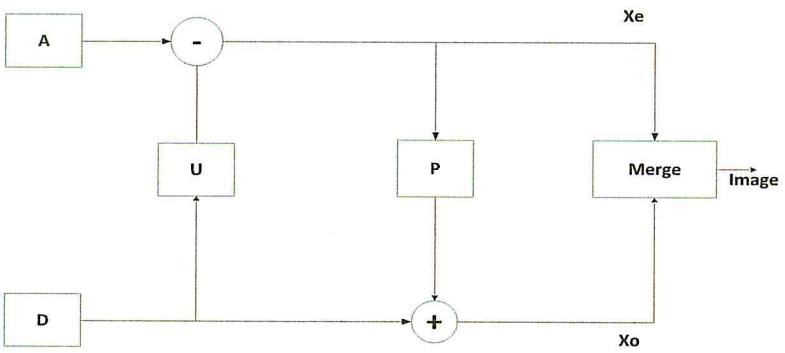


Figure 2.5 : Banc de synthèse

2.2.3.1 L'avantage de lifting scheme

- Le lifting permet une transformation "in place", allégeant l'implémentation pour la transformée en ondelettes rapides. Ceci dit, pas besoin d'allouer un espace mémoire auxiliaire.

- L'usage du lifting permet particulièrement de construire les transformée en ondelettes non linéaires: on peut par exemple faire les transformées entier entiers. ceci est important pour les implémentations matérielles et pour le codage d'image.

- Toute transformation faite avec le lifting est immédiatement inversible et la transformée inverse a exactement le même coût en complexité, que la transformée elle même.

- Le lifting permet une adaptation de la transformée en ondelette.

- Le lifting permet une construction des ondelettes sans faire usage de la transformée de Fourrier. Ceci signifie qu'il peut être utilisé pour construire les ondelettes qui ne sont pas nécessairement translatés ou dilatés d'une fonction (on parle d'ondelettes de seconde génération).

- Le lifting est plus souple pour les non puristes mathématiciens, car ne fait pas appel aux notions de la transformée de Fourrier, et peut être de la sorte facilement introduite, uniquement par ses arguments dans le domaine spatial.

- Enfin, le lifting expose le parallélisme inhérent de la transformée en ondelettes. Toutes les opérations d'un pas de lifting peuvent être faites totalement en parallèle [29].

2.3Réseau neuronal de Khonen (carte SOM)

Une carte de Kohonen (ou carte auto-organisee, SOM pour Self-Organising Map) permet de représenter un espace de grande dimension par une projection non linéaire sur un espace De dimension réduite. Kohonen s'est inspiré du cortex [32] qui projette sur sa surface les localisations des différents sens et muscles. Dans le cortex sensoriel par exemple, les régions du corps sont projetées de sorte que des stimuli appliques en des zones proches font réagir des neurones proches dans le cortex. Une cellule de la carte calcule simplement une distance, par exemple la distance euclidienne, entre ses entrées et les poids synaptiques associés.

Les cartes auto-organisatrices de Kohonen sont des types de réseaux qui s'inspirent de la modélisation des systèmes de perception, tels que la vue ou l'ouïe, chez les mammifères.

Dans ces systèmes il y a réception des signaux, puis traitement à l'intérieur du système nerveux. Leur principale propriété est de pouvoir coder sur des neurones voisins des signaux d'entrée qui se ressemblent

2.3.1 Architecture de réseau

Les « cartes auto-organisatrices de Kohonen » sont constituées d'une grille (uni- ou bidimensionnelle). Dans chaque noeud de la grille se trouve un « neurone ». Chaque neurone est lié à un vecteur référent, responsable d'une zone dans l'espace des données (appelé encore espace d'entrée).

Dans une carte auto-organisatrice, les vecteurs référents fournissent une représentation discrète de l'espace d'entrée. Ils sont positionnés de telle façon qu'ils conservent la forme topologique de l'espace d'entrée. En gardant les relations de voisinage dans la grille, ils permettent une indexation facile (via les coordonnées dans la grille). Ceci s'avère utile dans divers domaines, comme la classification de textures, l'interpolation entre des données, la visualisation des données multidimensionnelles.

Soit A la grille neuronale rectangulaire d'une carte auto-organisatrice. Une carte de neurones assigne à chaque vecteur d'entrée un v∈ V un neurone r ∈ *A* désigné par son vecteur de position$\vec{r}$, tel que le vecteur référent *wr* est le plus proche de V [33] [15].

Mathématiquement, on exprime cette association par une fonction:

$$\theta w : V \longrightarrow A$$

$$r = \theta w\ (v) = \arg\min ||v - wp|| \qquad (2.18)$$

Cette fonction permet de définir les applications de la carte.

- ***quantificateur vectoriel*:** on approxime chaque point dans l'espace d'entrée par le vecteur référent le plus proche par :

$$w_r = \phi_w^{-1}\ (\phi w(v)) \qquad (2.19)$$

- ***classificateur :*** en utilisant la fonction $r = \phi w(v)$ on assigne à chaque neurone de la grille une étiquette correspondante à une classe, tous les points de l'espace d'entrée qui se projettent sur un même neurone appartiennent à la même classe. Une même classe peut être associée à plusieurs neurones.

2.3.2 Algorithme d'apprentissage :

Après une initialisation aléatoire des valeurs de chaque neurones on soumet une à une les données à la carte auto adaptative. Selon les valeurs des neurones, il y en a un qui répondra le mieux au *stimulus*. Celui dont la valeur sera la plus proche de la donnée présentée. Alors ce neurone sera gratifié d'un changement de valeur pour qu'il réponde encore mieux à un autre *stimulus* de même nature que le précédent. Par là même, on gratifie aussi un peu aussi les neurones voisins du gagnant avec un facteur multiplicatif du gain inférieur à un. Ainsi, c'est toute la région de la carte autour du neurone gagnant qui se spécialise. En fin d'algorithme, lorsque les neurones ne bougent plus, ou très peu, à chaque itération, la carte auto organisatrice recouvre toute la topologie des données [33].

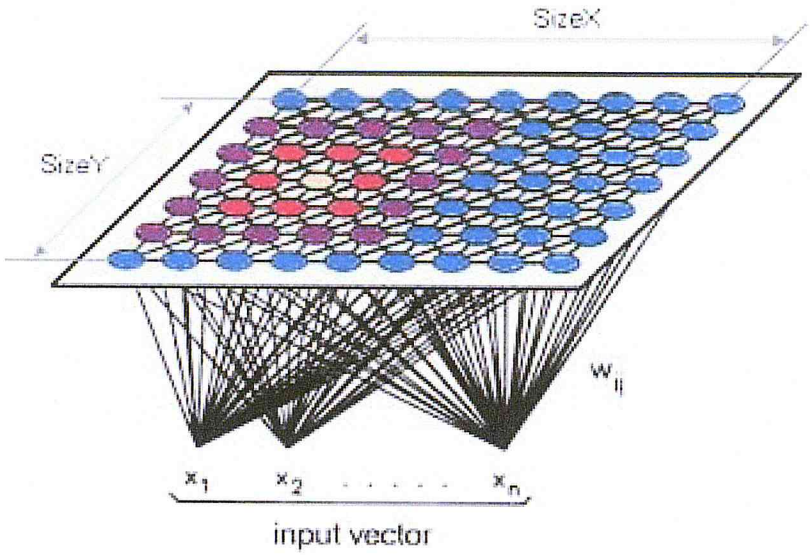


Figure 2.6 : Architecture de KSOM

2.3.2.1 **Formalisation mathématique**

En premier lieu, la grille doit être initialisée de manière aléatoire.

Un cycle d'apprentissage est constitué des étapes suivantes :

1) Présenter un vecteur d'entrée associé à un stimulus à la grille.

2) Trouver le nœud gagnant (ou *winner*). C'est l'unité dont le vecteur associé est le plus similaire au vecteur d'entrée.

||entrée – neurone *winner*|| = Mini ||entrée – neurone i || (2.20)

3) Modifier les poids *Wi* du nœud gagnant, ainsi que ceux de son entourage, de manière à ce que les vecteurs associés (les vecteurs de poids) «se rapprochent d'avantage» du vecteur d'entrée présenté à la grille. La règle de modification est la suivante :

$$\begin{cases} wi(t+1) = wi(t) + h(r,t)\left(pi - wi(t)\right) & \text{si } i \text{ voisinage} \\ wi(t+1) = wi(t) & \text{si } i \text{ voisinage} \end{cases} \quad (2.21)$$

Avec :

- $h(r,t) = (t).v(t)$.
- (t) : *Le taux d'apprentissage.*
- $v(t)$: *la fonction de voisinage*

4) Faire décroître la taille de la zone de voisinage des nœuds gagnants (la zone qui contient les neurones subissant la transformation).

5) Faire décroître le coefficient d'apprentissage, (t), qui contrôle l'importance des modifications appliquées aux vecteurs de poids.

La modification des vecteurs associés aux unités se fait de manière différente selon la position des nœuds par rapport à l'unité gagnante. Le nœud gagnant sera celui dont le vecteur subira le plus de modifications, tandis que les unités plus éloignées seront moins affectées. La fonction de voisinage v(r) va être maximale pour r=0 et décroître quand r croît, (quand on s'éloigne du nœud gagnant) [33].

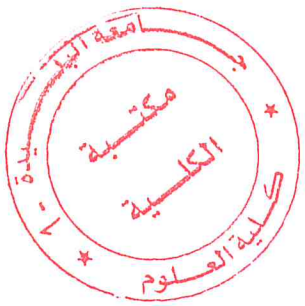
2.3.2.2 Avantage de carte SOM

L'algorithme de Kohonen profite des relations de voisinage dans la grille pour réaliser une discrétisation dans un temps très court. On suppose que l'espace n'est pas constitué de zones isolées, mais de sous-ensembles compacts. Donc en déplaçant un vecteur référent vers une zone, on peut se dire qu'il y a probablement d'autres zones dans la même direction qui doivent être représentées par des vecteurs référents. Cela justifie le fait de déplacer les neurones proches du vainqueur dans la grille dans cette même direction, avec une amplitude de déplacement moins importante. L'algorithme présente des opérations simples ; il est donc très léger en termes de coût de calculs [33].

2.4Conclusion

Contrairement aux méthodes classiques qui ont montré leurs limites, les réseaux de neurones ont montré leurs tendances à s'adapter à des problèmes complexes grâce à leur grande capacité de calcul et d'apprentissage. Ils sont l'objet d'utilisation dans les différents domaines .Le grand avantage caractérisé dans les réseaux de neurones de Kohonen est que ces derniers sont léger en cout et en calcule et sont portable dans différents domaines, ce qui les rend les plus simples à utiliser et les plus rapides.

Dans le prochain chapitre, nous allons discuter les résultats obtenus en appliquant la technique des réseaux de neurone (carte SOM) sur les images médicale de type IRM cérébrale, et la mesure des performances de la méthode de compression utilisée.



Chapitre 3:
Implémentation software de la méthode proposée.

3.1 Introduction

Dans notre travail, nous avons implémenté une méthode hybride entre les réseaux de neurone de Kohonen et la transformée en ondelettes pour la compression d'images IRM, dans ce chapitre nous présentions un aperçus théorique de la méthode proposée, leur implémentation software avec l'outil de développement Matlab et les résultats obtenus. Ces résultats sont satisfaisants par apport de critères d'évolution : taux de compression, PSNR et SSIM.

3.2 Chaine de compression des images IRM par la méthode proposée

La compression d'images de manière générale se fait en trois étapes : transformée, quantification et codage.

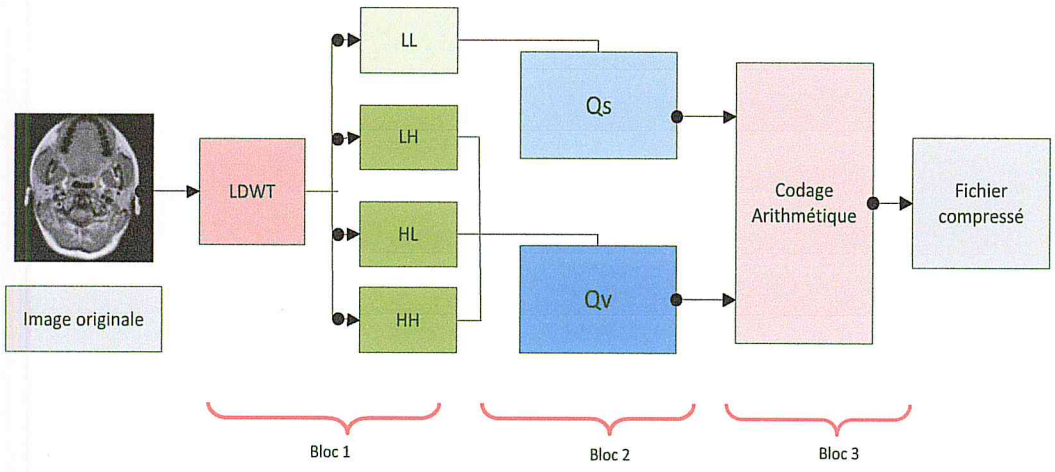


Figure 3.1 : Chaine de compression

L'implémentation software de la méthode proposée pour la compression d'image IRM est montrée en figure 3.1. Donc, après avoir lu le fichier DICOM (l'image originale), nous lui subissons une transformée en ondelettes par l'approche de lifting scheme (bloc 1), obtenant ainsi quatre sous-bandes : l'image LL, celle qui contient les basses fréquences (l'approximation de l'image originale), les images LH, HL, HH qui contiennent les haute fréquences (les détails de l'image originale, vertical, horizontal et diagonal).

Ensuite, une quantification scalaire est appliquée pour l'image LL. Pour les images LH, HL, HH nous appliquons une quantification vectorielle. La quantification (bloc 2) de ces quatre sous-bandes est appliquée par l'algorithme de Kohonen, en utilisant deux dictionnaires (un pour la QS et l'autre pour la QV) qui sont aussi réalisés avec l'approche de la carte SOM de Kohonen. Quatre mapps qui sont produites à la fin de la quantification (une mapp pour l'image LL, et trois mapps pour les images LH, HL, HH).

Pour supprimer la redondance qui se trouve dans les quatre mapps et les rendre dans un seul fichier, on applique un codage arithmétique (bloc 3).

A la fin de processus, un fichier compressé est obtenu qui peut être décompressé par la suite si le médecin a besoin de l'image originale, la procédure suivie pour la récupération de l'image originale est l'inverse de la procédure suivie pour la compression.

L'implémentation de cette méthode est réalisée avec l'outil de développement MATLAB R2015a.

3.2.1 Implémentation de bloc 1 : la transformée

L'étape de la transformé consiste à décomposé l'image en quatre sous-bandes (hautes fréquences et en basses fréquences), nous avons implémenté un algorithme d'ondelettes par l'approche de lifting scheme qui consiste en trois principales étapes : séparation, prédiction et mis à jour, l'organigramme suivie par notre algorithme est montré dans les figures 3.2 et 3.3.

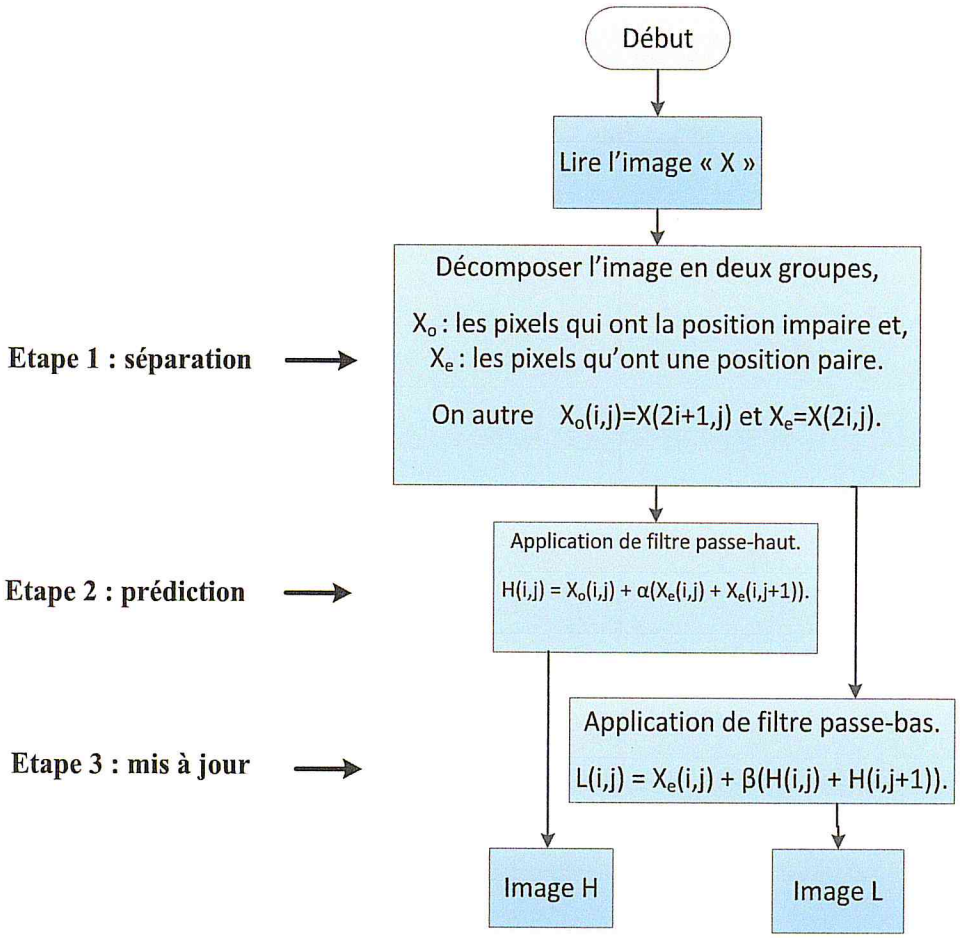


Figure 3.2: Application des filtres selon les lignes de l'image (horizontalement).

Dans la figure 3.2, les coefficients de filtre haut et filtre bas sont appliqués sur l'image selon les lignes ce qui permet d'obtenir les deux images 'L' qui représente l'approximation et 'H' qui représente les détails de l'image 'X', pour avoir les quatre sous-bandes il faut appliquer ce filtre verticalement (selon les colonnes) sur les deux images 'L' et 'H'.

L'application de filtre sur l'image 'L' nous permet d'obtenir l'approximation 'LL' et le détail 'LH' de l'approximation de l'image originale (image 'L'). l'application de filtre sur l'image 'H' nous permet d'obtenir les deux sous-bande 'HL' et 'HH'. La figure 3.3 montre l'organigramme de l'algorithme qu'on a utilisé pour décomposer l'image 'L' et l'image 'H'.

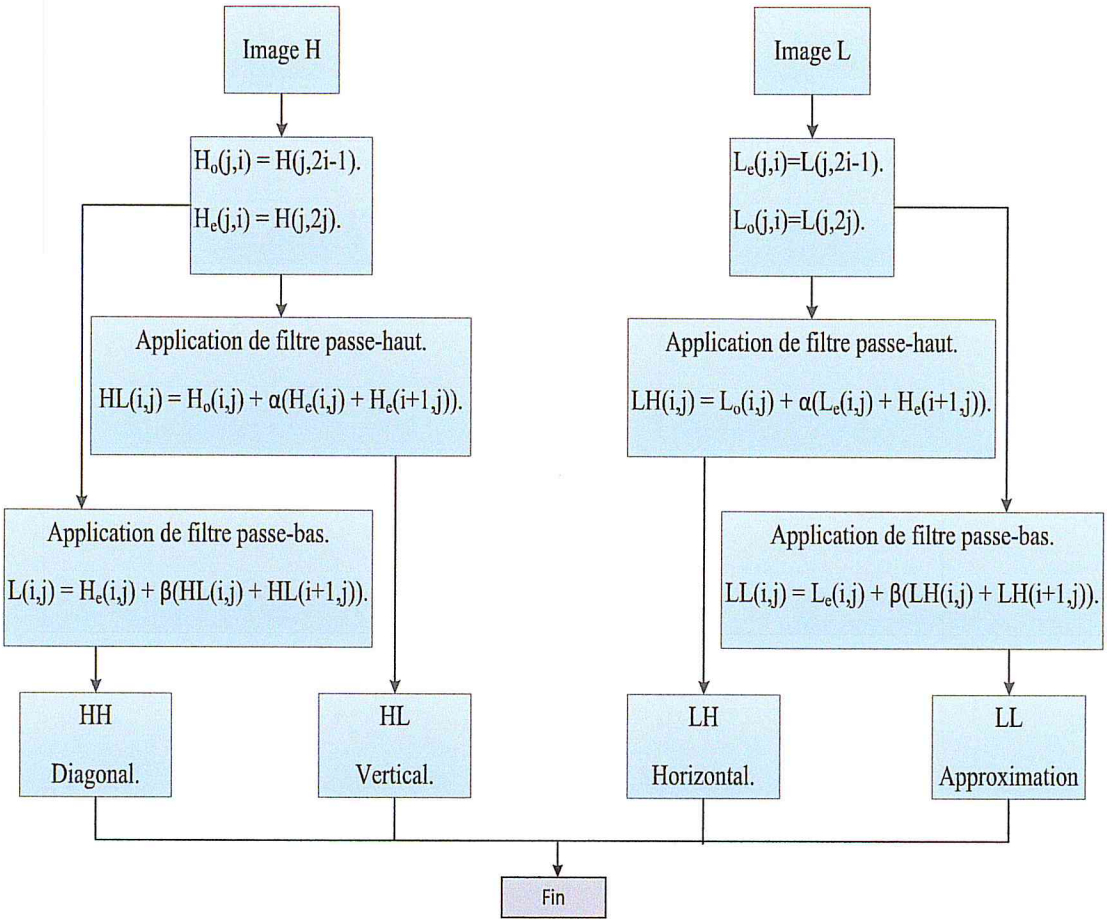


Figure 3.3: Application des filtres selon les lignes de l'image (horizontalement).

Dans ce travail, on a choisi d'implémenter deux filtre : filtre de Haar et le filtre de Le Gall pour pouvoir choisir le quel on va l'implémenter sur le circuit FPGA.

3.2.1.1 Implémentation de filtre de Haar

Pour le filtre de haar, les deux équations utilisés [35] dans l'algorithme pour l'étape de prédiction et de mis à jour sont eq 3.1 et eq 3.2 :

- Filtre haut (étape de prédiction)

$$H(i,j) = Xo(i,j) - Xe(i,j). \qquad \text{eq (3.1)}$$

- Filtre bas (étape de mis à jour)

$$L(i,j) = Xe(i,j) + H(i,j)/2. \qquad \text{eq (3.2)}$$

Nous avons appliqué la transformée d'ondelette par le filtre de haar selon l'approche lifting scheme sur des images médicales, nous avons obtenu les résultats des figures 3.4 et 3.5.

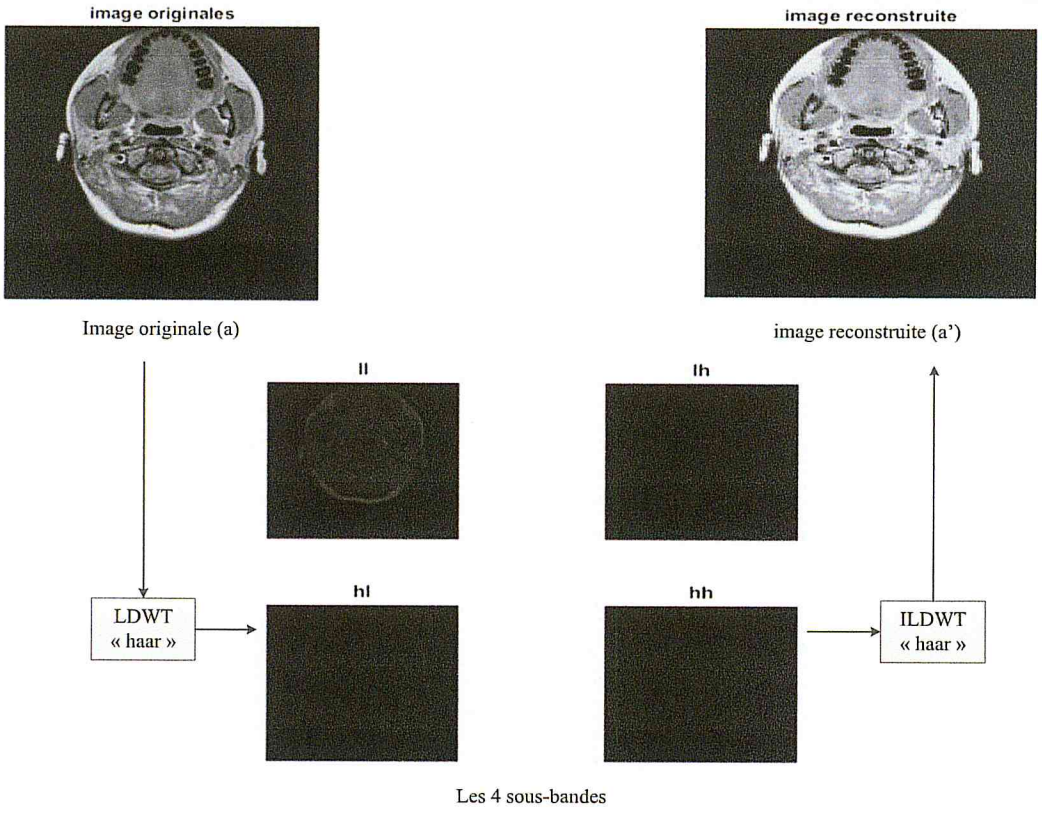


Figure 3.4: Application de filtre Haar sur l'image (a) et l'image reconstruite (a').

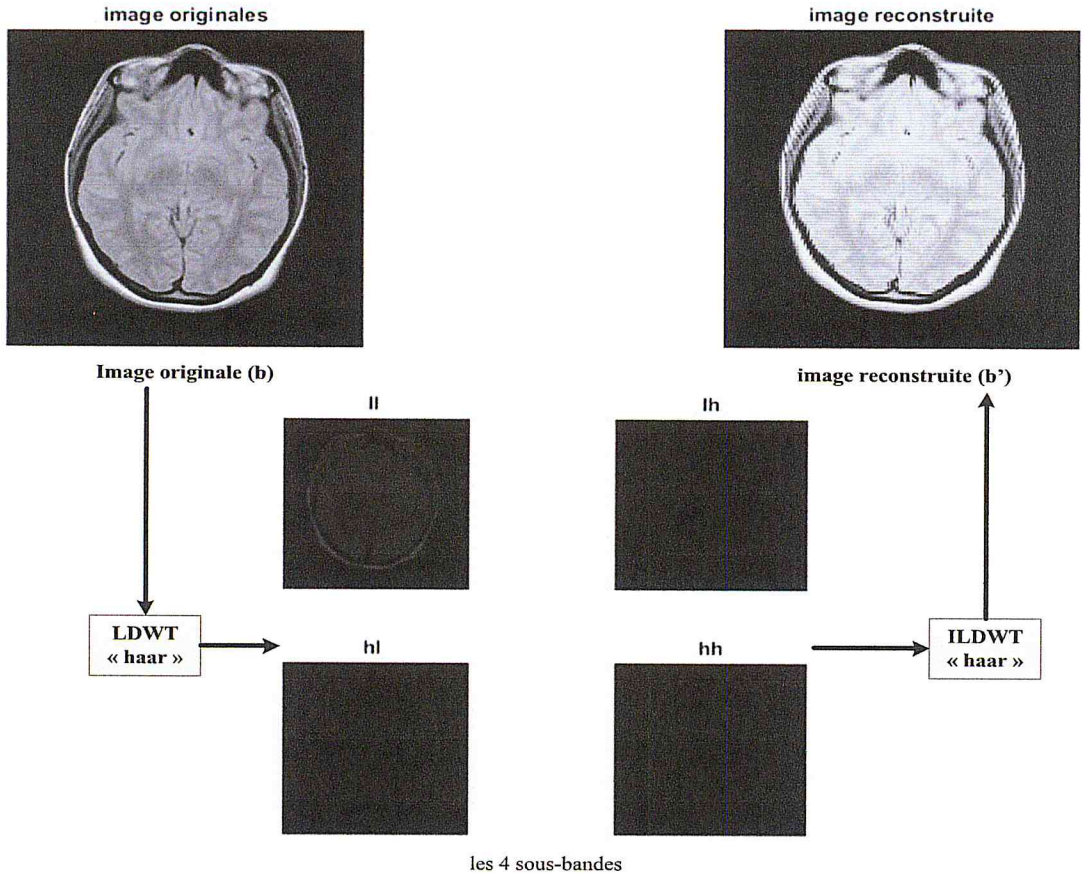


Figure 3.5 : Application de filtre Haar sur l'image (b) et l'image reconstruite (b').

Le tableau (3.1) montre le résultat de l'application de filtre de Haar par l'approche de lifting scheme en calculant le PSNR et le SSIM sur quelques images de test.

Images	Image (a)	Image (b)	Image (c)	Image (d)	Image (e)
PSNR (dB)	15.1495	13.1507	14.4551	13.6559	16.5697
SSIM	0.5768	0.5255	0.5900	0.5470	0.6492

TABLEAU 3.1 : RESULTAT D'IMPLEMENTATION SOFTWARE DE FILTRE DE HAAR

Le filtre de Le Gall consiste de 5 coefficients pour le filtre passe-bas et 3 coefficients pour le filtre passe-haut :

- Filtre passe-bas L: {-1/8 , 2/8 , 6/8 , 2/8 , -1/8}[28].
- Filtre passe-haut H: {-1/2 , 1 , -1/2} [28].

Pour l'implémentation de filtre de Le Gall, les équations de filtre de banc sont les suivantes [28] [36]:

- Prédiction :

$$H(2i-1,j) = \mathrm{X(2i-1,j)} - [\frac{\mathrm{X(2i-2,j)} + \mathrm{X(2i,j)}}{2}] \qquad \text{eq (3.3)}$$

- Mis à jour :

$$(2i,j) = \mathrm{X(2i,j)} - [\frac{\mathrm{H(2i-1,j)} + \mathrm{H(2i+1,j)}}{4}] \qquad \text{eq (3.4)}$$

Comme résultat les figures 3.6 et 3.7 montre l'application de filtre de Le Gall selon l'approche de lifting scheme sur l'image originale et son image reconstruite par l'application de filtre inverse.

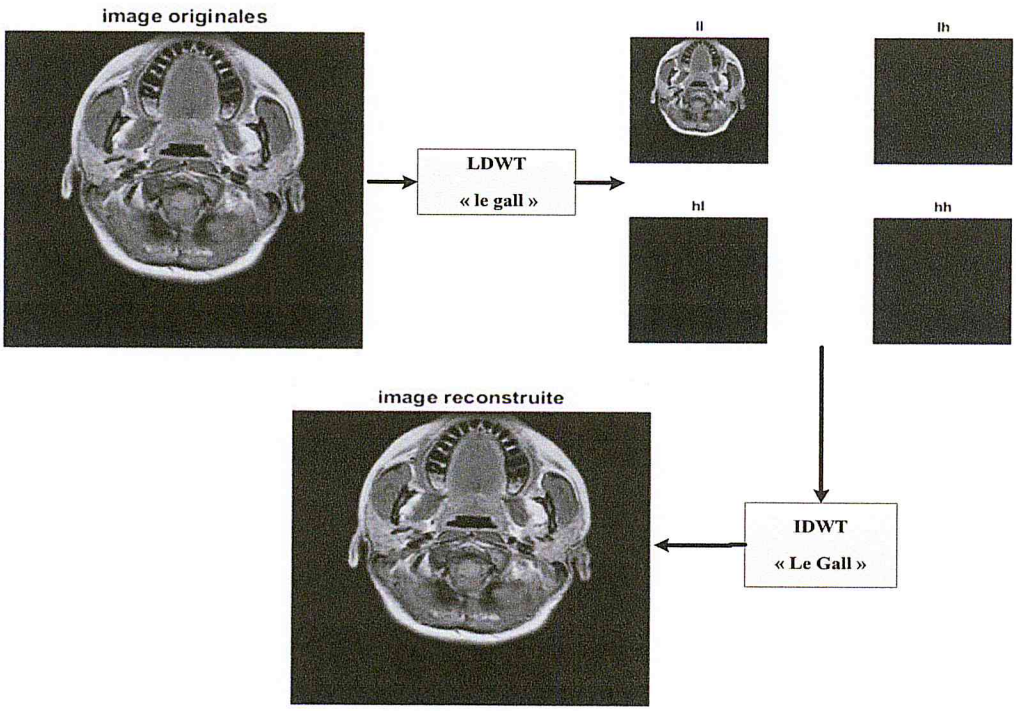


Figure 3.6 : Résultat de l'application de filtre Le Gall sur l'image originale « a », les 4 sous-bandes et leur image reconstruite.

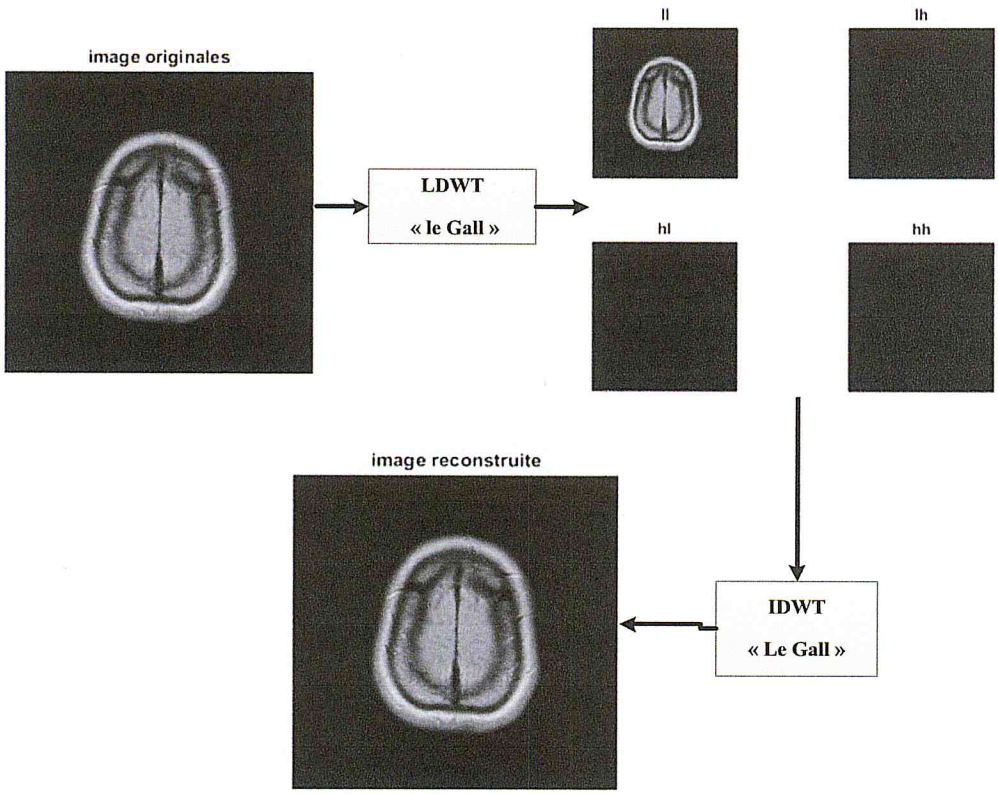


Figure 3.7 : Résultat de l'application de filtre Le Gall sur l'image originale « b », les 4 sous-bandes et leur image reconstruite.

Pour bien montrer l'efficacité de filtre Le Gall le tableau 3.2 représente les valeurs de PSNR et SSIM calculé à partir de l'image originale et l'image reconstruite par l'application de l'ondelette inverse selon l'approche de lifting scheme.

- **Remarque :** les images utilisées pour tester le filtre Le Gall sont les même qui sont utilisées pour le filtre Haar.

Images	Image (a)	Image (b)	Image (c)	Image (d)	Image (e)
PSNR (dB)	329.8659	329.8323	329.4311	329.6565	331.1677
SSIM	1	1	1	1	1

TABLEAU 3.2 RESULTAT DE L'APPLICATION DE FILTRE LE GALL PAR L'APPROCHE DE LIFTING SCHEME.

On compare les résultats de l'application de filtre de Haar par l'approche de lifting scheme sur les images médicales avec celle de filtre de Le Gall, on remarque que le filtre Le Gall donne le meilleur résultat, par conséquence le filtre le Gall est choisit pour la construction de dictionnaire et la quantification ainsi pour l'implémentation hardware de l'algorithme de compression.

3.2.2 Implémentation de bloc 2 : Quantification

Après l'étape de transformé suit une autre étape de quantification (comme est montré dans la figure 3.1). la quantification sert à approximer à une grandeur 'X' une valeur 'y_i' de l'ensemble 'C' (dictionnaire), ou 'i' est appelé indice de quantification [37].

Le principe de quantification vectorielle (QV) est montré dans la figure 2.9, ainsi que pour le principe de quantification scalaire (QS) est le même principe en changent seulement le bloc d'image par un seul pixel de l'image.

Dans cette étape nous avons utilisé l'algorithme de Kohonen (KSOM) pour quantifier les 4 sous-bandes, ainsi que l'algorithme de Kohonen est utilisé pour la construction des dictionnaires (un pour la QS et l'autre pour la QV).

3.2.2.1 Organigramme de la quantification

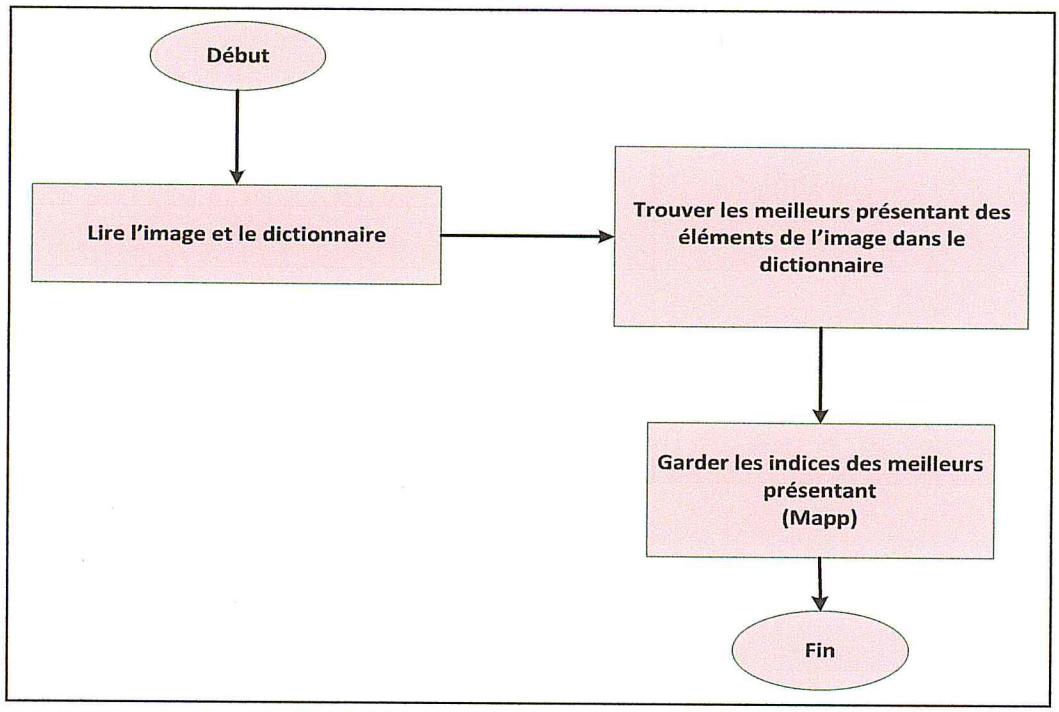


Figure 3.8 : L'organigramme de la quantification.

3.2.2.2 Algorithme de QS

- **Etape 1 :** lire l'image X (M x L), et le dictionnaire Dic (N x N).

- **Etape 2 :** présenter un élément de l'image.

  For(i=1; i<M ; i++)
  For(j=1 ; j<L ; j++)

- **Etape 3 :** initialise la distance minimal à **:**

  Dis_min=$\sqrt{Dic(1)^2 - X(i,j)^2}$.

- **Etape 4 :** calcule la distance euclidienne entre le pixel de l'image et l'élément de dictionnaire

  for (k=0 ; k<N*N ; K++)
  Distance= $\sqrt{Dic(k)^2 - X(i,j)^2}$.
  if (Distance < Dis_min)
  Dis_min = Distance ;

- **Etape 5 :** garde l'indice de voisin le plus proche (le meilleur présentant).

3.2.2.3 Algorithme de QV

Les mêmes étapes que la QS seulement que les éléments de dictionnaire sont des cellules, alors les éléments présentées de l'image sont des blocs de même taille que les éléments de dictionnaire.

Donc l'équation pour calculer la distance euclidienne devient la somme des pixels de la matrice résultante de la soustraction de bloc de l'image et l'élément de dictionnaire.

3.2.2.4 La quantification inverse

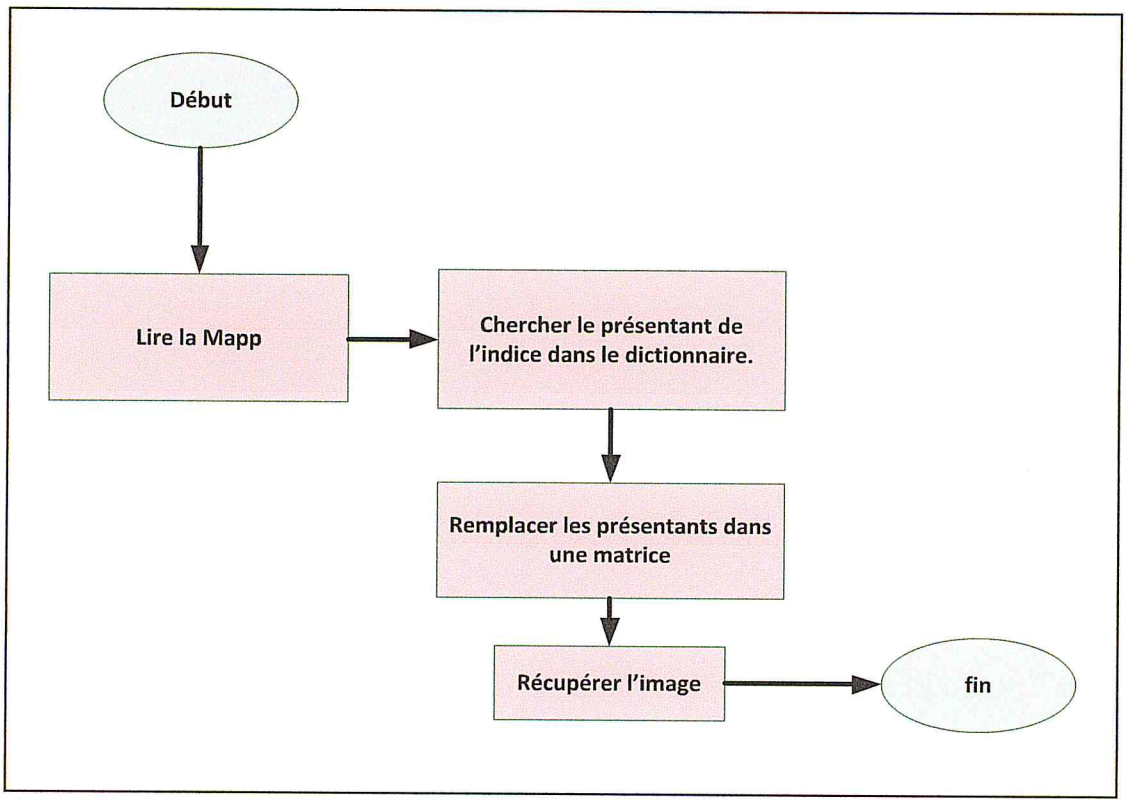


Figure 3.9 : L'organigramme de la quantification inverse

3.2.3 Construction de dictionnaire

Le dictionnaire est l'un des éléments de la quantification le plus important, et pour que l'image reconstruite après la chaine de décompression soit sans distorsion, et avec un taux de compression élevé il faudra un dictionnaire robuste. L'algorithme de Kohonen permet de construire un dictionnaire robuste, ayant plusieurs avantages [37] [38] [39].

3.2.3.1 Dictionnaire de la QS

Le dictionnaire de la QS est constitué par une carte de K-SOM de taille (10x10). L'algorithme de construction suit l'organigramme de la figure 3.10.

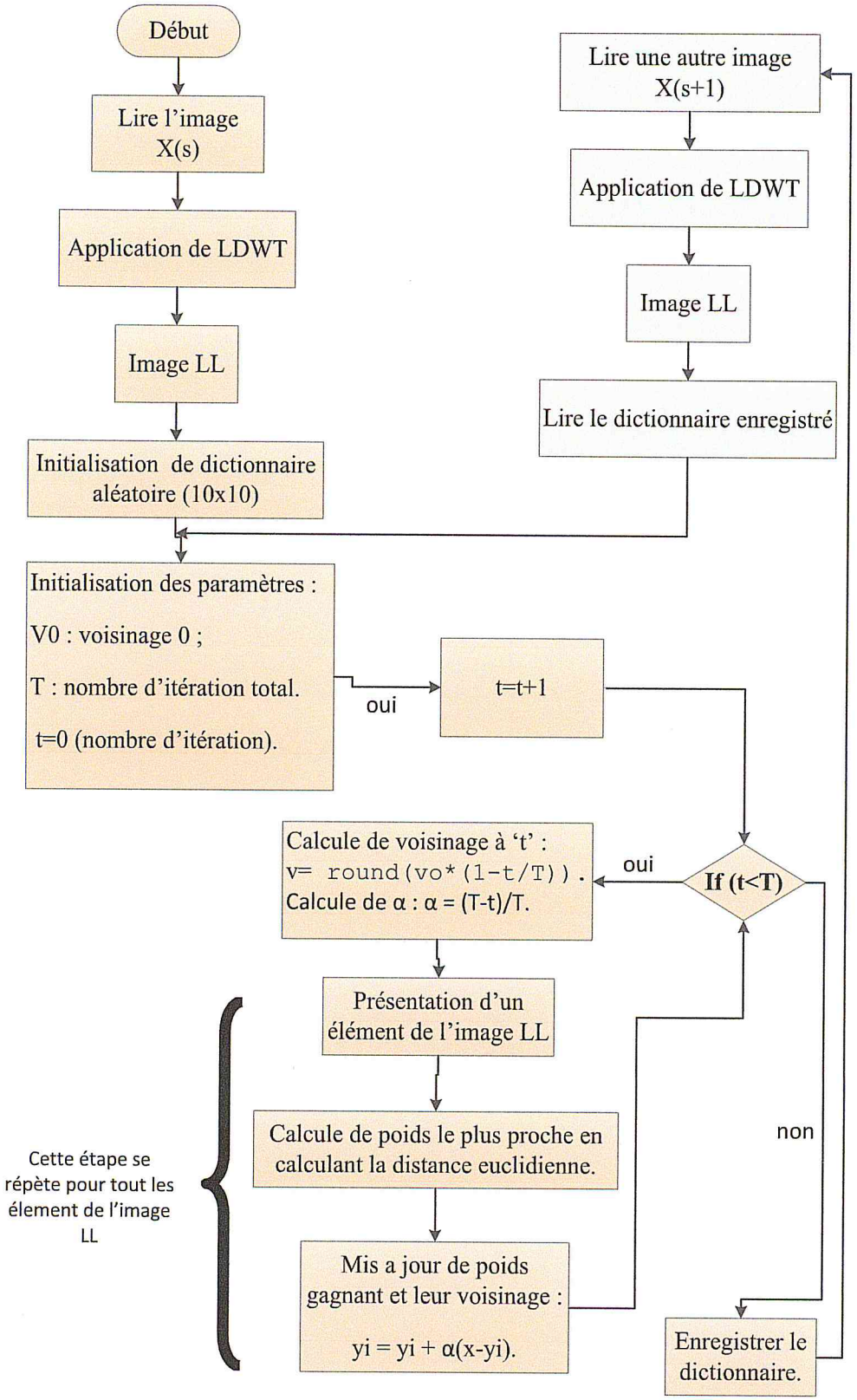


Figure 3.10: Organigramme de construction de dictionnaire de la QS

L'algorithme de construction de dictionnaire pour la QS donne comme résultat la figure 3.11.

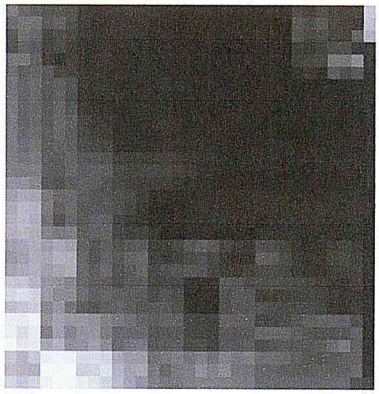


Figure 3.11: Dictionnaire de QS.

3.2.3.2 Dictionnaire de la Qv

Le dictionnaire de la QV est constitué par des neurones sous forme des blocs de taille (2x2). L'algorithme de son construction suit l'organigramme suivant :

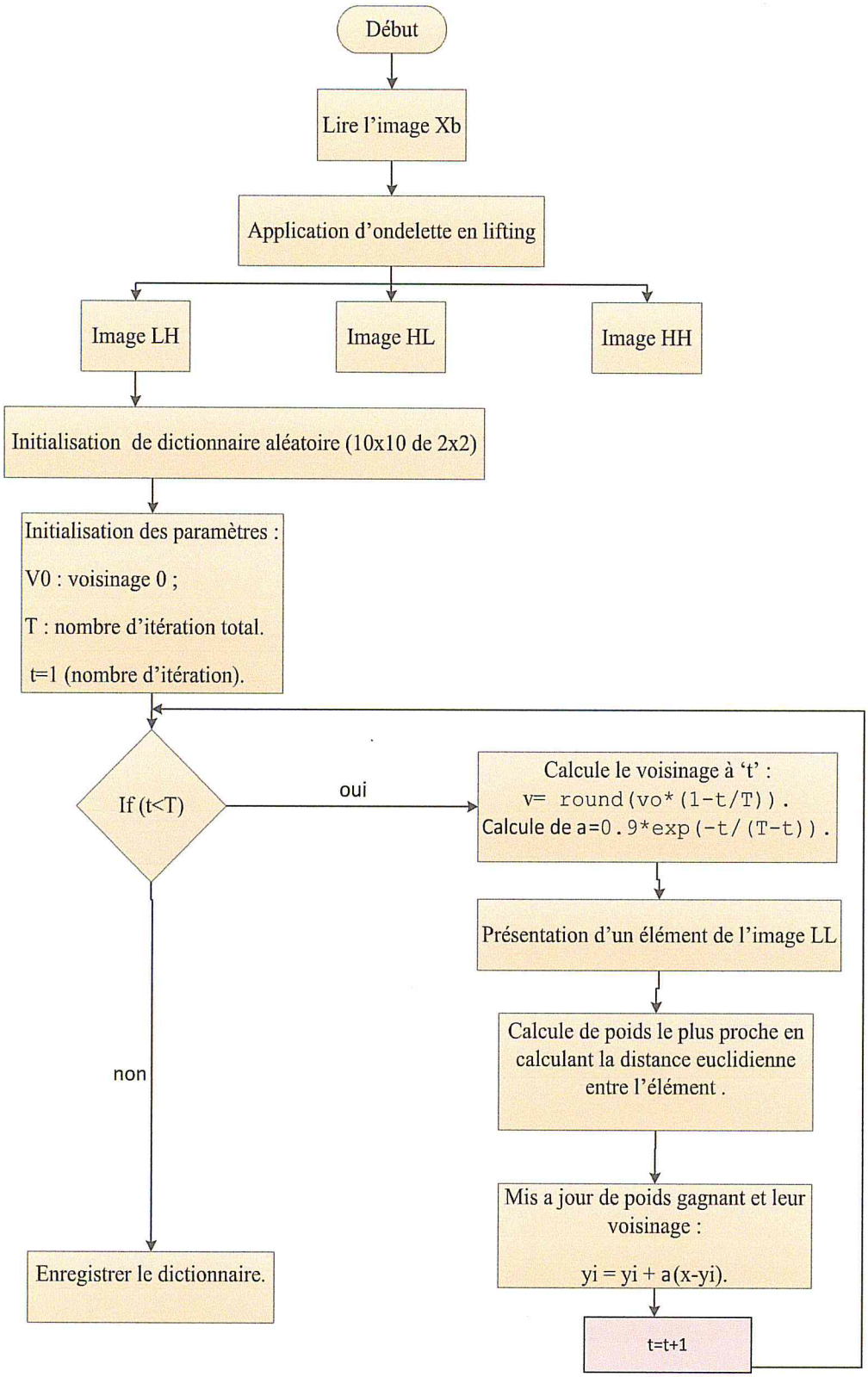


Figure 3.12 : Organigramme de construction de dictionnaire de QV

- **Remarque :** Après l'enregistrement de dictionnaire, on applique le même algorithme pour le reste de sous-bande mais dans cette étape au lieu d'initialiser un dictionnaire aléatoire on utilise le dictionnaire qu'est déjà enregistré. On répit cet algorithme pour toues les images qu'on a pour la construction de dictionnaire.

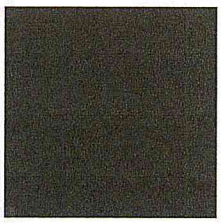


Figure 3.13: Dictionnaire de la QV.

3.3 Résultat de la méthode proposée

Notre algorithme est testé sur plusieurs images médicales, les résultats obtenus en termes de critères d'évolution de compression PSNR, SSIM, taux de compression sont présentés dans le tableau 3.3 :

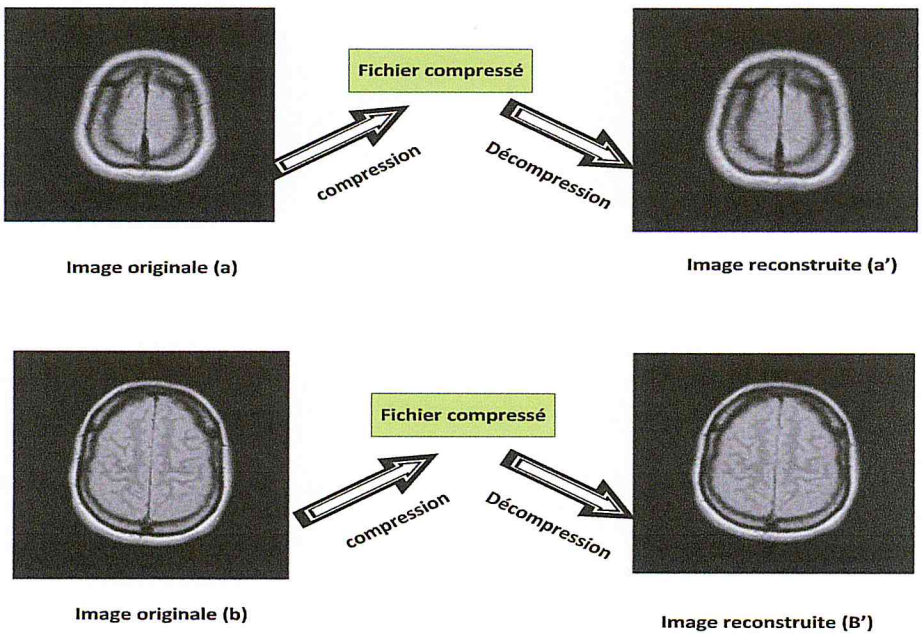


Figure 3.14 : Résultat de l'implémentation software de la méthode proposée

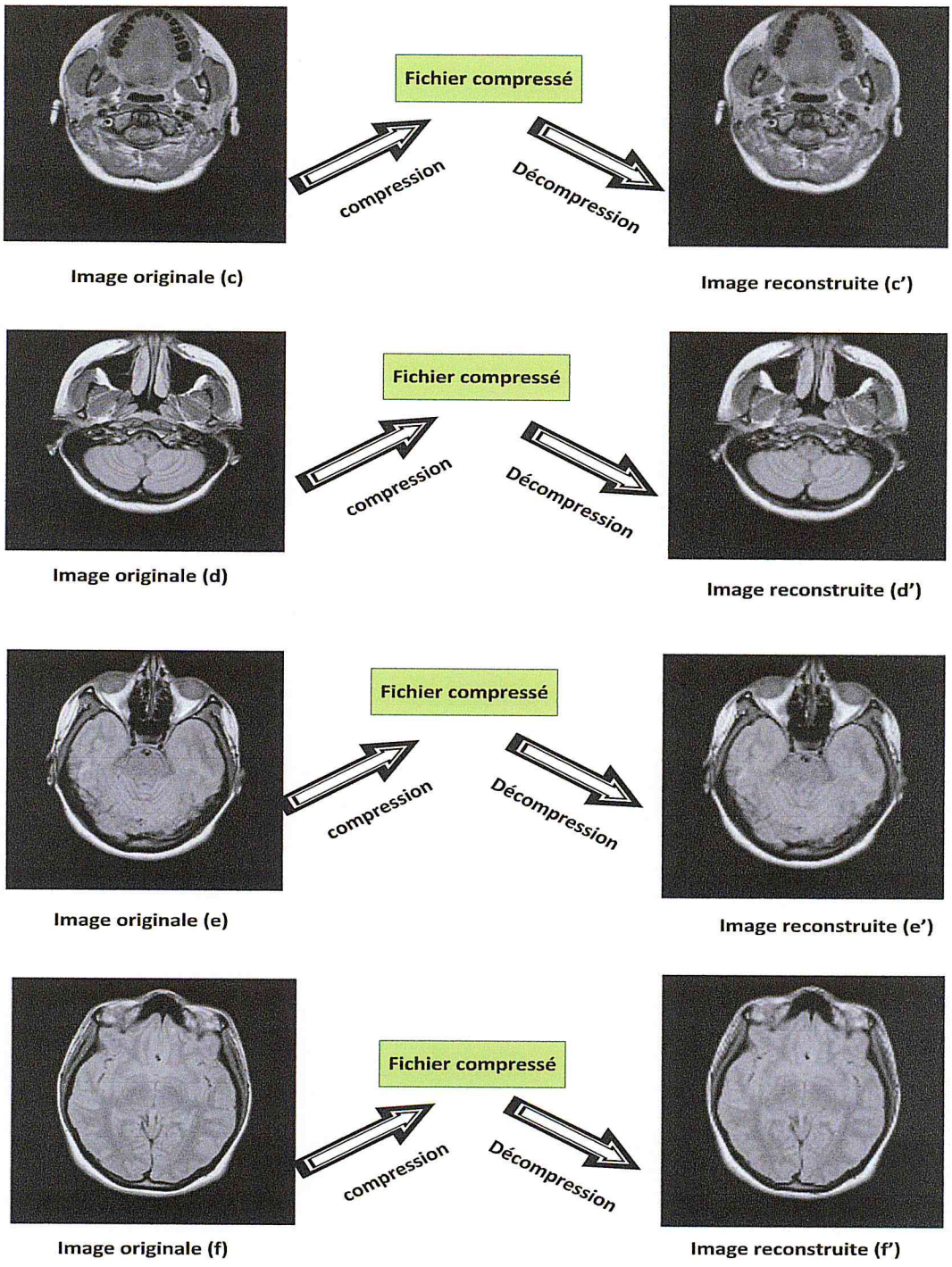


Figure 3.15 : Résultat de l'implémentation software de la méthode proposée (à gauche les images originales et à droite les images reconstruites)

Image	Taille de l'image originale	PSNR (db)	SSIM	Taille de fichier compressé	Taux de compression.
a	(256x256) 113Ko	36,7432	0.8920	15 Ko	8,67 : 1
b	(256x256) 113Ko	32.7789	0,9037	15Ko	8,67 : 1
c	(256x256) 113Ko	31.9040	0.8974	15ko	8,67 : 1
d	(256x256) 113Ko	35,7501	8,66	15Ko	8,67 : 1
e	(256x256) 113Ko	31.5624	0.8948	15ko	8,67 : 1
f	(256x256) 113Ko	31.4328	0.9000	15ko	8,67 : 1

TABLEAU 3.3 RESULTAT DE L'IMPLEMENTATION SOFTWARE DE L'APPLICATION.

3.4 Conclusion

A travers ce chapitre, nous avons présenté l'algorithme de la méthode proposée dans le but de la compression des images IRM et les résultats obtenus après simulation de code sur l'outil de développement MATLAB.

Le rapport de PSNR, de SSIM et de taux de compression montre que les résultats obtenus sont satisfaisants.

Le chapitre quatre est consacré à l'implémentation de l'approche proposée sur une carte FPGA ZedBoared de la famille ZYNQ.

Chapitre 4 :
Implémentation hardware

4.1 Introduction

L'objectif principal dans ce chapitre est l'implémentation matérielle de la méthode proposée dans le but de la compression des images IRM sur un circuit FPGA (Field Programmable Gate Array) de la famille ZYNQ « ZedBoard », l'algorithme est développé à l'aide de deux outils: Vivado design suite 2015.4 et le SDK 2015.4.

Par la suite, une interface graphique qu'était développée par l'outil de développement JAVANetBeans IDE 8.1 qu'il communique avec la partie PS (processing system) de la carte FPGA ZedBoard par le protocole RS232.

Dans ce chapitre, nous avons commencé par la présentation de la carte utilisée pour le développement de l'application, l'architecture globale de notre algorithme et les résultats obtenus.

4.2 Description de la carte ZedBoard

La ZedBoard est une carte d'évaluation et de développement basé sur la plate-forme de traitement extensible Xilinx Zynq-7000. Combinant un système de traitement dual core ARM Cortex-A9 (PS) avec une partie programmables Logic (PL) qui contient 85 000 cellules de la Sériés-7, le Zynq-7000 AP SoC (all programmable system on chip) peut être ciblé pour une utilisation large dans de nombreuses applications. Le mélange robuste des périphériques embarqués et des capacités d'expansion de la ZedBoard en fait une plateforme idéale pour les concepteurs débutants et expérimentés [40].

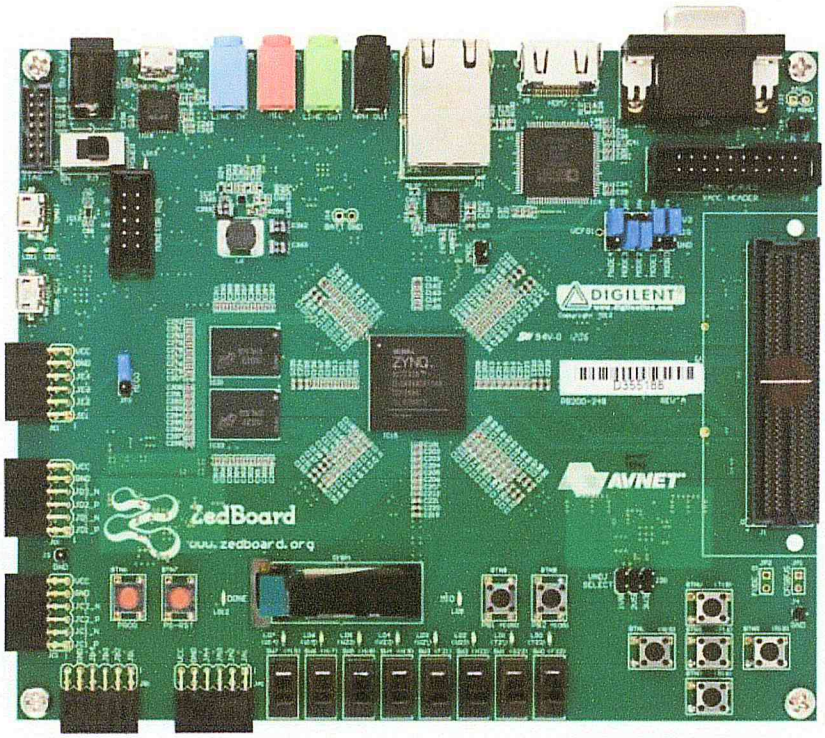


Figure 4.1 : la carte ZedBoard

4.2.1 Caractéristiques de la carte ZedBoard

- Processeur :
  - Zynq ™ -7000 AP SoC XC7Z020-CLG484-1.
- Mémoire :
  - 512 Mo de DDR3.
  - 256 Mo Quad-SPI Flash.
  - Carte SD de 4 Go.

- La communication :
  - Programmation USB à bord JTAG.
  - Ethernet 10/100/1000.
  - USB OTG 2.0 et USB-UART.
- Connecteurs d'extension
  - Connecteur FMC-LPC (68 E / S à une seule fin ou 34).
  - 5 en-têtes compatibles Pmod ™ (2x6).
  - En-tête Agile Mixed Signaling (AMS).
- Clocking
  - Source d'horloge 33.33333 MHz pour PS
  - oscillateur à 100 MHz pour PL
- Afficher
  - Sortie HDMI prenant en charge 1080p60 avec 16 bits, YCbCr, couleur 4: 2: 2
  - Sortie VGA (couleur de résolution 12 bits)
  - Affichage OLED 128x32
- Configuration et débogage
  - Interface USB-JTAG intégrée
  - Connecteur JTAG du câble de la plate-forme Xilinx
- E / S à usage général
  - 8 DEL utilisateur
  - 7 boutons poussoirs
  - 8 commutateurs DIP [40]

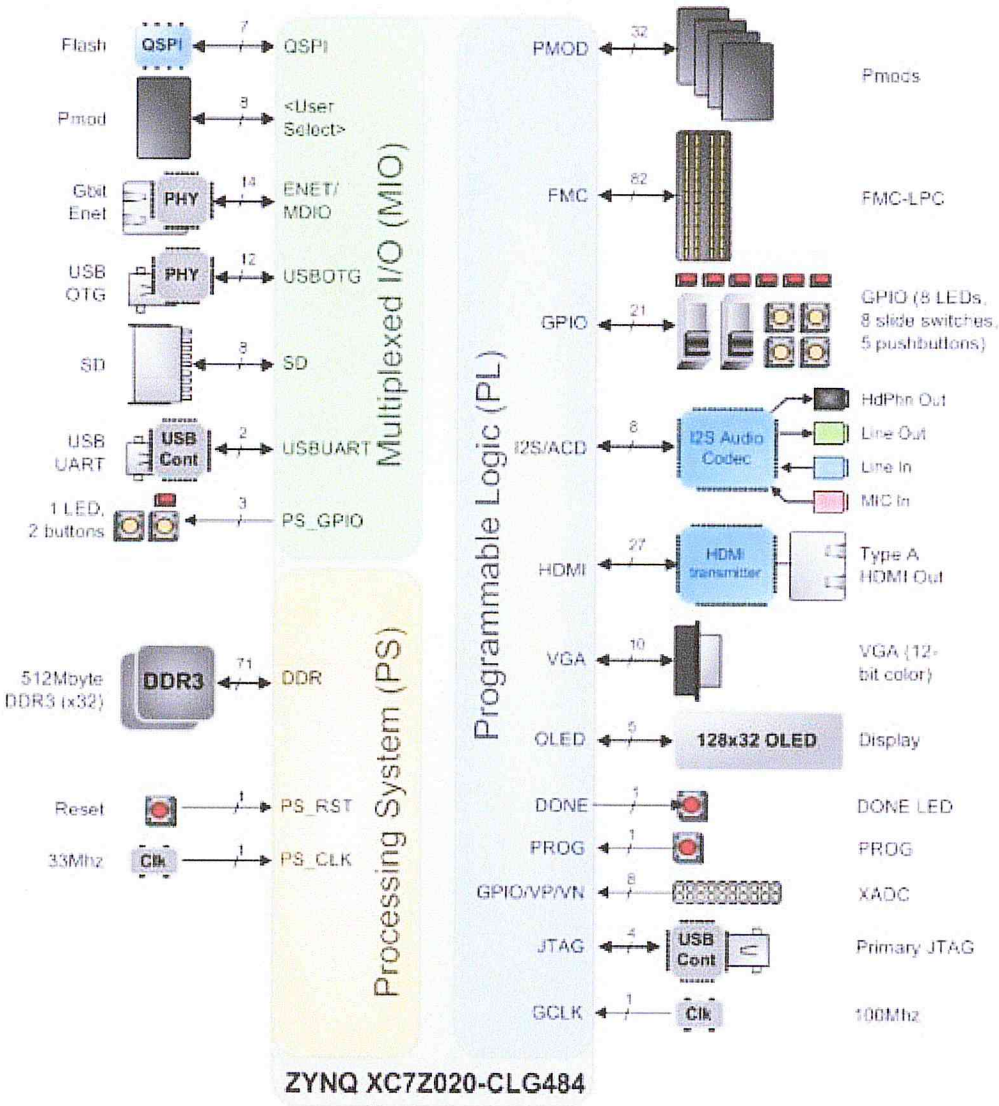


Figure 4.2 : Diagramme de blocs de matériel ZedBoard.

4.2.2 Système de traitement PS (processing system)

La figure 4.3 montre les composantes principales de la partie PS de la carte FPGA ZedBoard tel que : le processeur ARM Coretex A9 dual core, la mémoire DDR3, Jtag, USB.

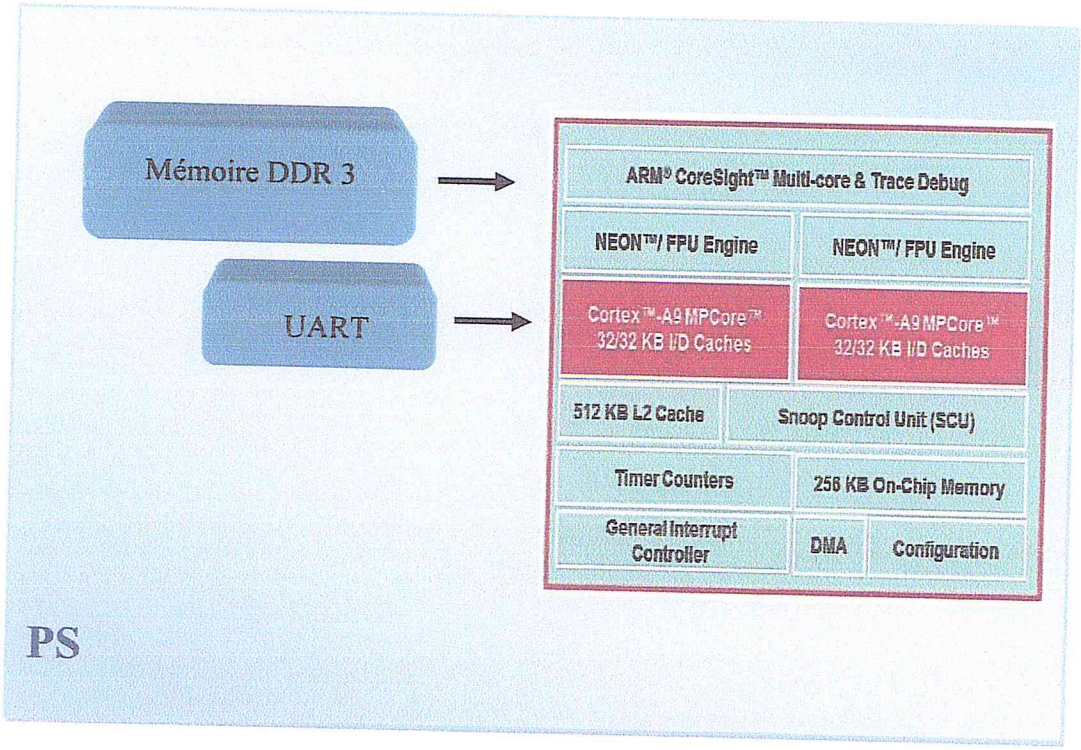


Figure 4.3 : Système microprocesseur (PS).

4.2.1.1 Performances de processeur ARM cortex A9

L'ARM Cortex-A9 est un processeur performant de type tablette fonctionnant jusqu'à 800 MHz. Il possède un cache L1 de 32 kilo octets sur les données et sur les adresses. Il y a deux coeurs A9 dans la Zynq.

La Snoop Control Unit (SCU) connecte les deux Cortex-A9 à la mémoire système et au cache L2. Il maintient la cohérence entre les deux caches L1 données. Il arbitre entre les deux processeurs pour l'accès au cache L2. Il fournit un accès à la ROM/RAM interne. Il fournit aussi un accès à l'ACP (Accelerator Coherence Port) pour la logique programmable.

Le cache L2 de 512 kilo octets est unifié et performant (8 voies associatives, write-back et write-through). Il cache la mémoire DDR ainsi que les périphériques PS et PL [42].

4.2.1.2 Le contrôleur de la mémoire DDR 3

. Le contrôleur de le mémoire DDR3 SDRAM est utilisé pour offrir un accès partagé de la mémoire DDR3 aux noyaux ARM et autres composants. Il dispose de 4 ports esclaves du bus AXI (Advanced eXtensible Interface) pour offrir l'accès aux noyaux ARM via leur cache L2 commun.

4.2.1.3 Les périphérique USB

4.2.1.3.1 USB-UART

La carte ZedBoard possède un pont USB-UART connecté à un périphérique PS, pour la communication en série : tels que le protocole RS232 utilisé pour la transmission et la réception des données.

4.2.1.3.2 JTAG

Il existe plusieurs solutions pour debugger le programme dans la partie ARM ainsi que la logique programmable avec Vivado. Nous utiliserons la solution mono câble JTAG avec le ARM Debug Access Port (DAP) placé en tête dans la chaine JTAG Xilinx. Cela nous permettra aussi de télécharger le bitstream dans la PL.

4.3 Description de l'architecture globale de l'implémentation hardware

Ce travail consiste a l'implémentation d'une méthode hybride ondelettes et réseaux de neurones pour la compression des images IRM et créer une interface graphique sur PC. L'interface graphique communique avec la partie PS de la carte ZedBoard par le protocole de transmission en série « RS232 » (annexe 1) qui permet la transmission des pixels des images 8 bits par 8 bits de l'interface graphique vers la partie PS de la carte Zedboard pour le traitement. La figure 4 .4 représente l'architecture globale de notre application.

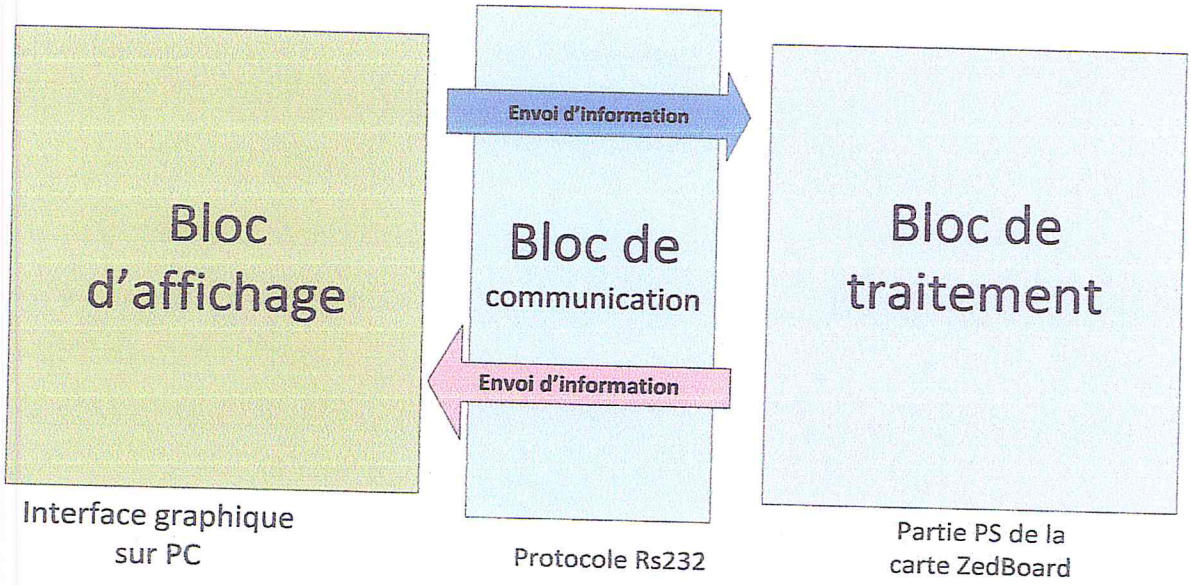


Figure 4.4 : Schémas synoptique de l'architecture globale

Le traitement de compression se fait au niveau de la partie PS de la carte FPGA, le procédé utiliser pour l'implémentation et la configuration de la carte est détaillée à la figure 4.5 qui décrite l'organigramme de notre architecture la liaison entre l'interface graphique et l'algorithme de compression et les différentes parties de l'implémentation de la méthode.

Le protocole RS232 est schématisé sous forme de flèches qui lient les deux principales parties de l'implémentation.

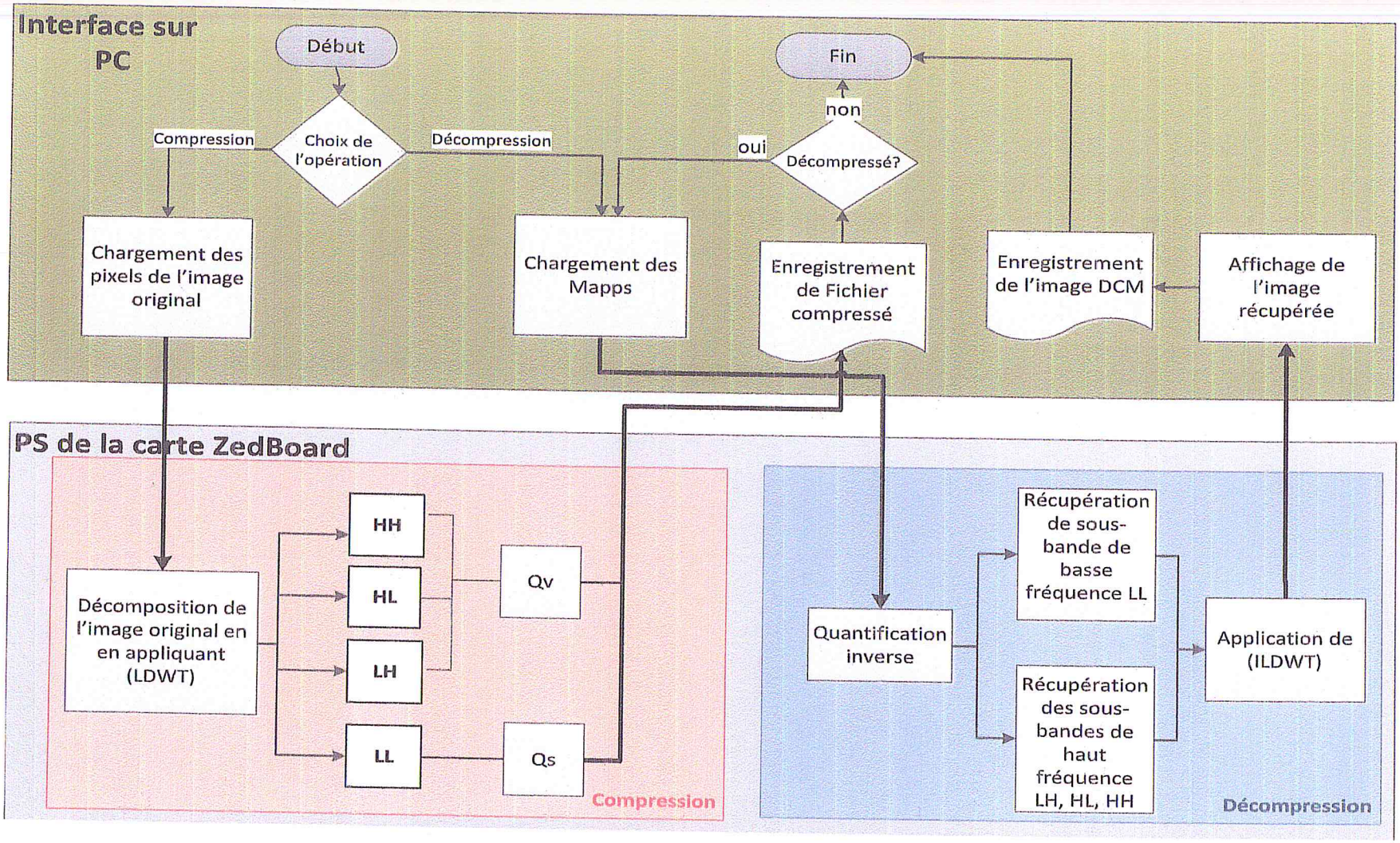


Figure 4.5 : L'organigramme de la l'architecture globale.

4.3.1 Bloc de traitement

L'algorithme de compression et de décompression est implémenté dans la partie PS de la carte ZedBoard, en utilisant un seul microprocesseur **(ARM Cortex-A9).**

La figure 4.6 montre les différents périphériques a configuré pour notre application :

- UART : permet la connexion en série à l'ordinateur,
- l'horloge interne : l'horloge avec laquelle le système sera cadencé,
- mémoire DDR dans laquelle est sauvegardée l'application à exécuté par le PS.

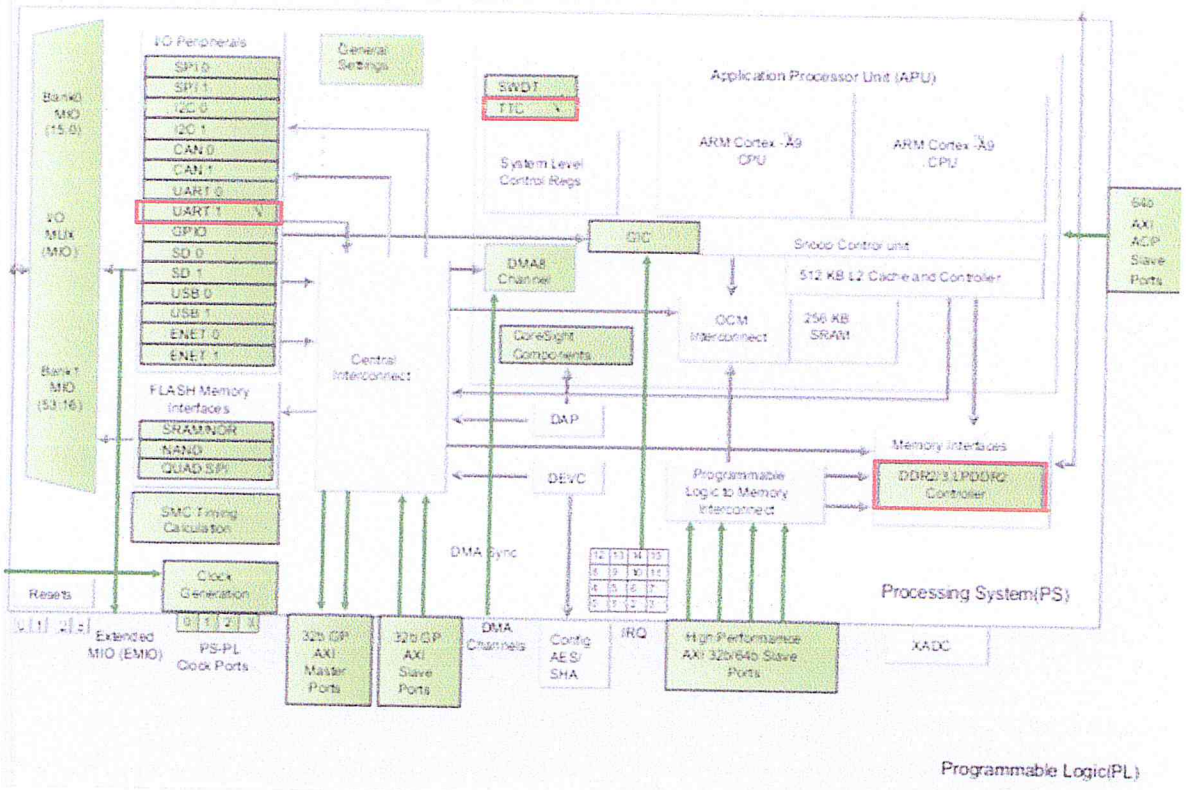


Figure 4.6 : Configuration des périphériques

Après avoir configuré la plateforme hardware qui inclut les paramètres d'horloge, du la mémoire DDR3 et le module URAT en utilisant l'outil de développement Vivado design suite 2015, on génère la plateforme et l'exporte vers SDK pour le développement de l'application.

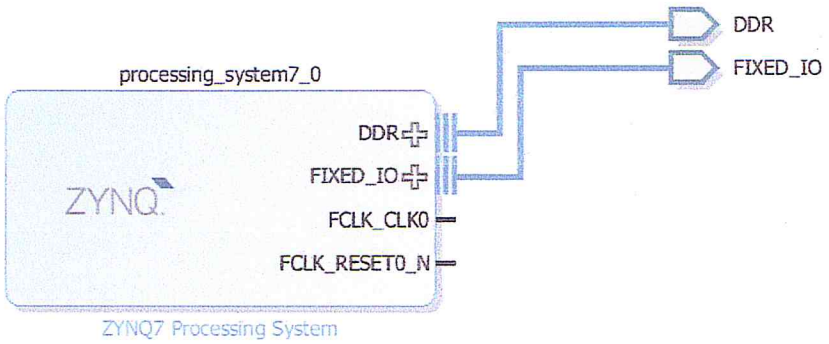


Figure 4.7 : La partie PS utilisée.

Après avoir configuré la plateforme hardware qui inclut les paramètres d'horloge, du la mémoire DDR3 et le module URAT en utilisant l'outil de développement Vivado design suite 2015.4, on génère le fichier HDL Wrapper dont le rôle est d'instancier le système processeur (PS) ; ensuite, on crée le Bitstream. Après la création de top module, un fichier .hdf (Hard Description File) qui contient le code d'initialisation de la partie PS et les paramètres d'initialisation pour l'interface DDR, les horloges et les MIOs (Mixed in/out) va être exporté vers SDK (Soft Developpement Kit), sur cette environnement qu'on va crée notre application en utilisant le langage C embarqué, on génère le BSP Standalone (Board Support Package), ensuite on passe à la création de notre application, après l'exécution de programme un fichier elf va être crée automatiquement celui qui va être importé vers la carte FPGA (PS de ZedBoard).

La figure 4.8 montre le flot de conception et de l'implémentation d'une application sur une carte FPGA de la famille Zynq.

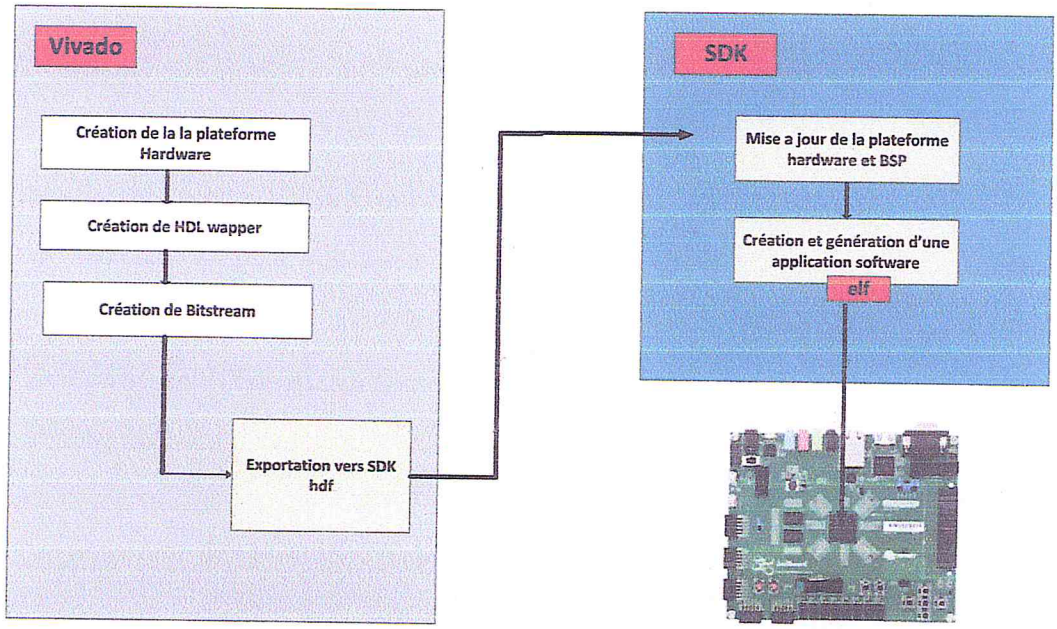


Figure 4.8 : Flot de conception d'une application.

4.3.2 Bloc d'affichage : réalisation de l'interface

L'interface graphique qui sert à faciliter l'utilisation de l'application pour la compression des images IRM est réalisée par le langage JAVA (annexe 1) par l'outil Netbeans IDE 8.1.

Notre interface est composée principalement de trois parties figure 4.9 :

- La barre de menu.
- La barre d'affichage des images.
- Barre de commentaire.

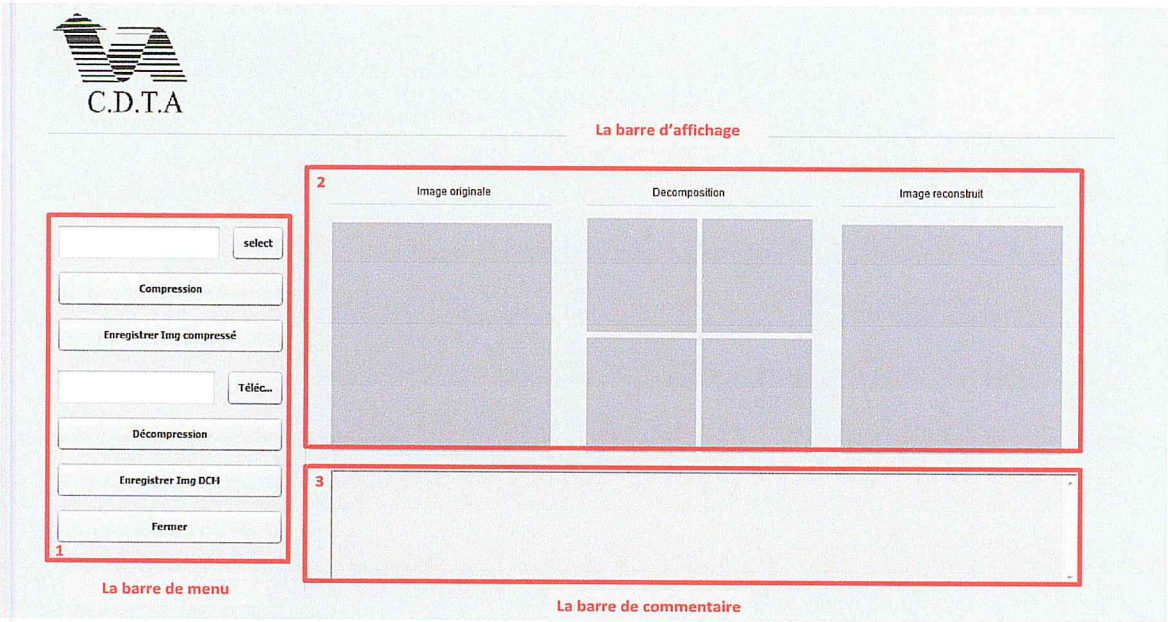


Figure 4.9 : l'interface graphique de l'application.

- **La barre de menu** : La figure 4.10 montre la barre de menu et ses fonctionnalités :

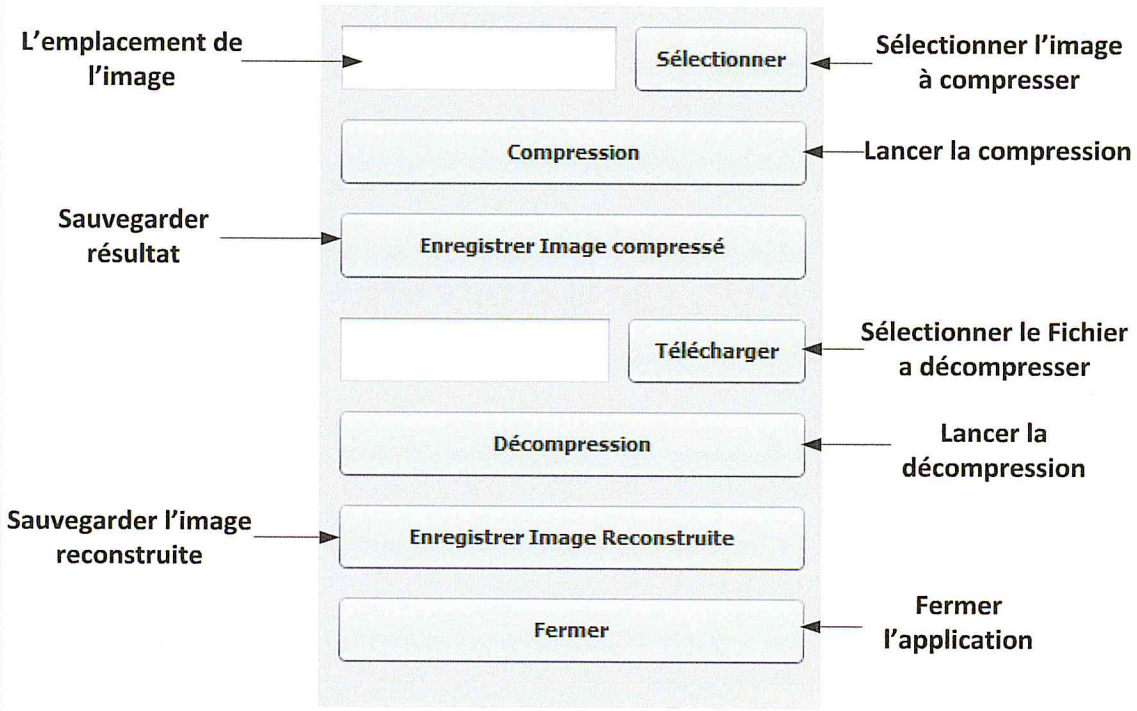


Figure 4.10 : La barre de menu

- **La barre d'affichage :** La figure 4.11 montre la barre d'affichage et ses fonctionnalités.

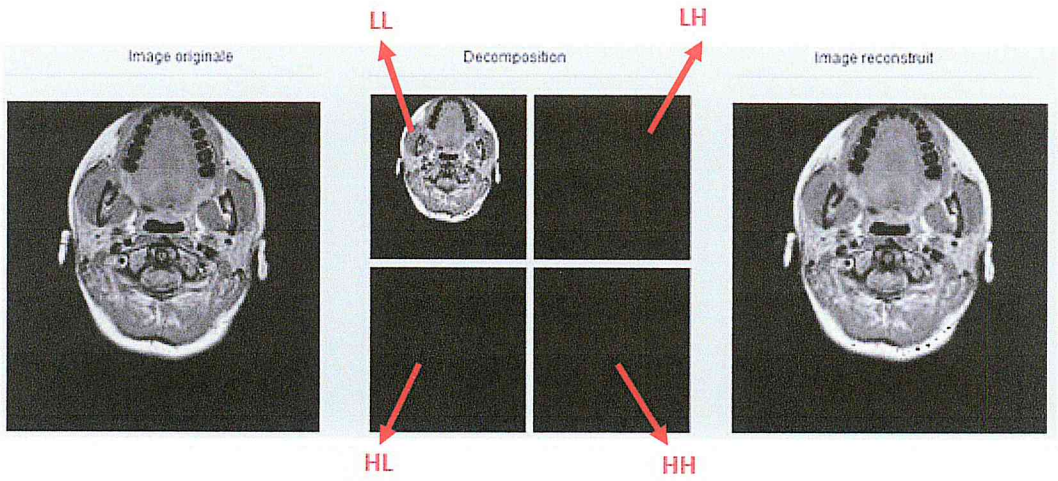


Figure 4.11 : la barre d'affichage

La troisième partie consiste à afficher des commentaires qui montrent l'opération en cours ou dans le cas d'une erreur.

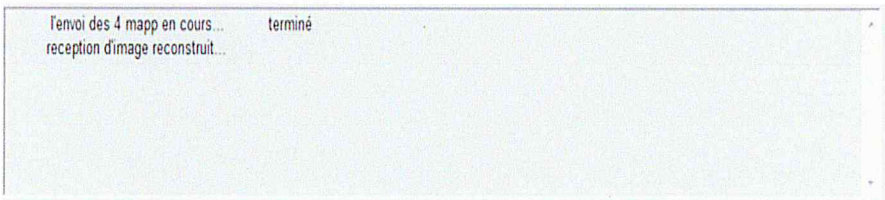


Figure 4.12 : La barre de commentaire.

4.3.3 Bloc de communication

Pour l'envoi d'information de l'interface vers la carte (la partie PS) et vice versa on a utilisé le protocole de transmission en série RS232.

4.4 Résultat de l'implémentation

Le tableau 4.1 montre le résultat en temps d'exécution de l'algorithme dans la partie PS, et la figure 4.13 représentes les ressources des différentes composantes utilisées ainsi que la puissance sur de la carte ZedBoard.

Le tableau 4.2 montre les images IRM reconstruite sur l'interface et celles qui sont reconstruites sur MATLAB, pour pouvoir les comparer.

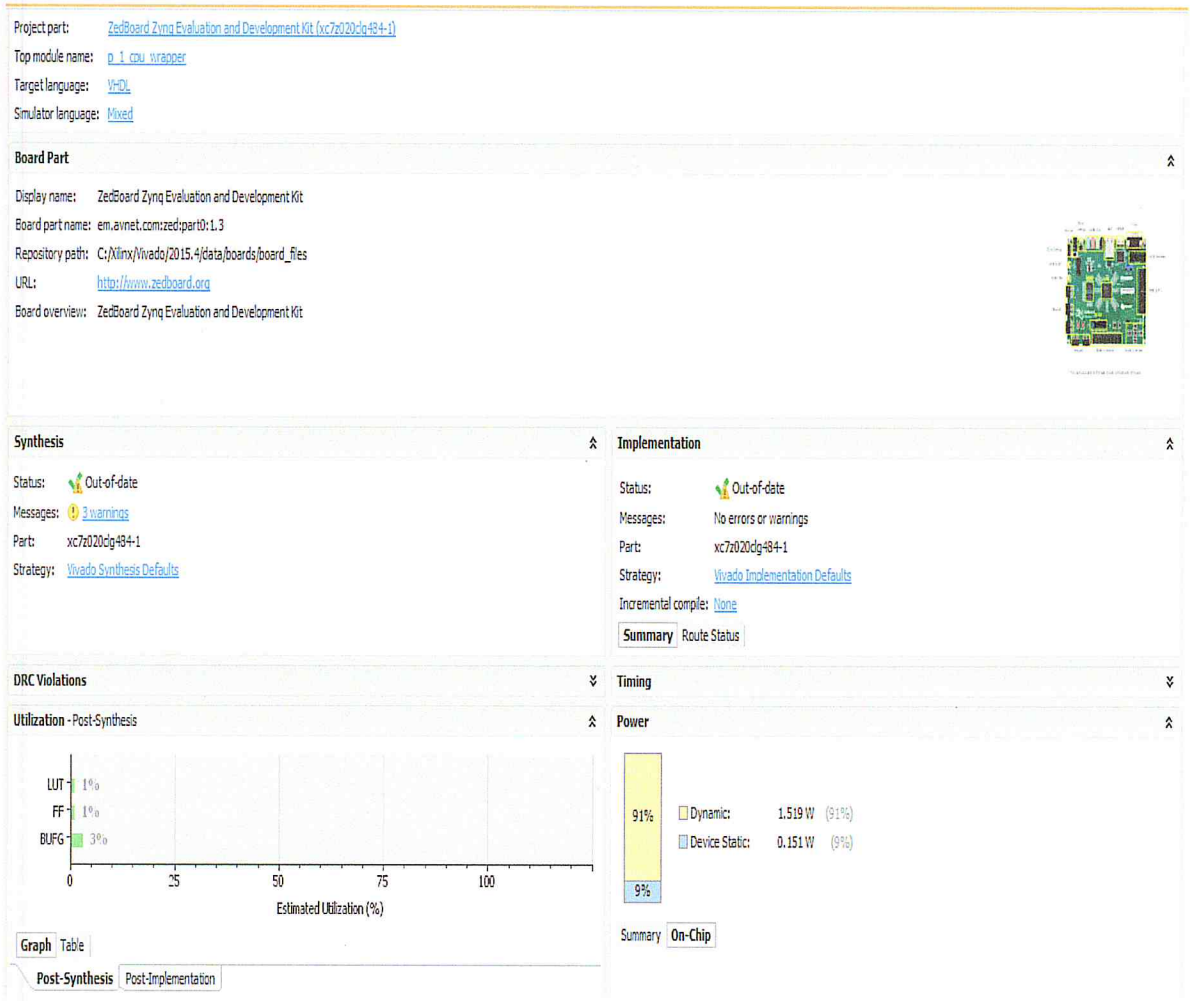
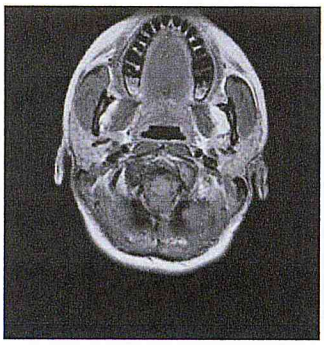
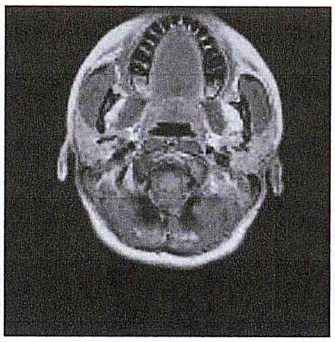
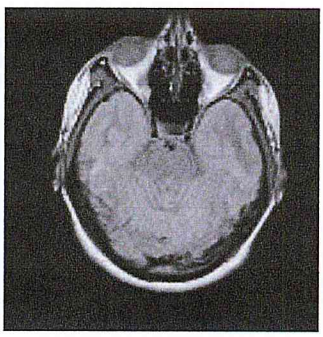


Figure 4.13 : Résultat de l'implémentation de l'application

	LDWT	Qs	Qv	Q-inverse	ILDWT
Le temps (us)	3 371	88 394	129 979	927.6	3 190.34

TABLEAU 4.1 : TEMPS D'EXECUTION DE DEFERENTE PARTIE DE L'ALGORITHME

- Le temps d'exécution de l'algorithme complet est: 225 968.9 (us).

Image originale	Image reconstruite sur MATLAB	Image reconstruite sur l'interface
		
		

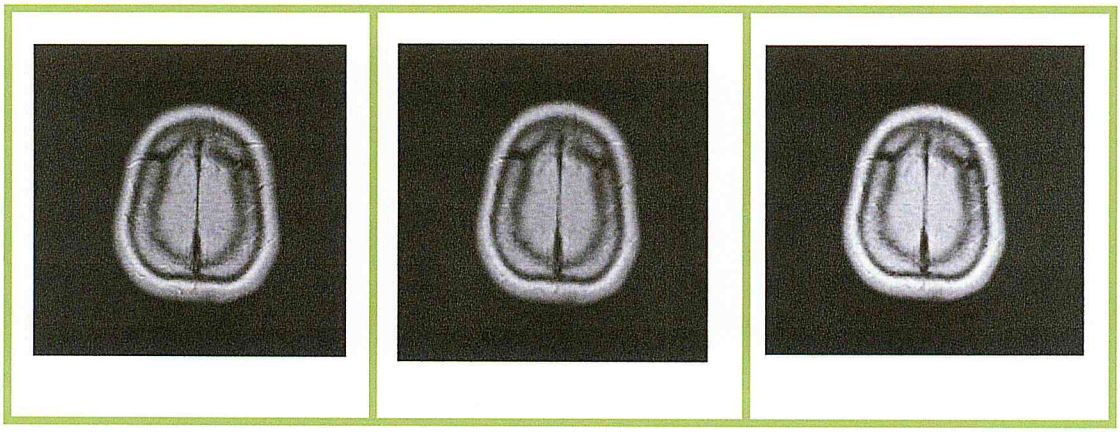


TABLEAU 4.2 : COMPARAISON ENTRE L'IMAGE RECONSTRUITE AVEC MATLAB ET L'IMAGE RECONSTRUITE AVEC L'INTERFACE

4.5 Conclusion

Dans ce chapitre nous avions présenté la méthodologie de conception de notre architecture, nous sommes passés par plusieurs étapes chacune d'entre elles utilisent un outil bien spécifique. On a présenté l'implémentation hardware sur une carte FPGA ZedBoard de la famille ZYNQ avec la création d'une interface graphique associée dans le but de la compression des images IRM, nous avion donné les résultats de simulation de quelques images ainsi les ressources utilisées pour notre application les résultats obtenus sont satisfaisants montre l'efficacité de la méthode proposée.

Conclusion générale

De nombreuses méthodes de compression des images existent. L'une des techniques qui a acquis une attention particulière et une reconnaissance grandissante au sein de la communauté scientifique sont les réseaux de neurones et la transformée en ondelettes.

Dans ce mémoire, nous avons étudié la transformée en ondelettes basée sur l'approche du Lifting scheme, dans le but de décomposer l'image en quatre sous bande, suivie de l'étape de quantification par la carte SOM de Kohonen ; et leurs implémentation matérielle sur un circuit FPGA de la famille ZYNQ (ZedBoard).

Le premier chapitre est concentré sur la connaissance et la compréhension des différentes caractéristiques d'une image, ainsi que les différentes méthodes d'acquisition des images médicales, cela permet d'augmenter la qualité de celle-ci en appliquant les méthodes de traitement appropriées telles que la compression par les différentes méthodes utilisées.

Dans le deuxième chapitre, nous avons étudié les différentes méthodes de la compression des images on se basant sur la transformée en ondelettes et l'utilisation de la carte SOM de Kohonen pour la quantification et la construction de dictionnaire.

Dans la troisième partie, nous avons présenté l'implémentation software sous l'outil de développement MATLAB de la méthode de compression appliqués sur des images IRM. Les résultats obtenus au niveau de la simulation montrent l'efficacité de l'algorithme basé sur les ondelettes et les réseaux de neurones.

Le quatrième chapitre est consacré à l'implémentation de la procédure proposé dans le troisième chapitre sur une carte FPGA et la création d'une interface graphique sur PC.

Ce travail aura pour perspective une étape de prétraitement des images médicales bruités et l'intégration dans un système embarquée pour la télémédecine et sa sécurisation ;

Les transformées en ondelettes et les réseaux de neurones demeurent des outils largement utilisés ; de ce fait, l'amélioration des performances et de leurs exécution constitue une préoccupation majeure dans le contexte d'un traitement massif de l'information et la réalisation des IPs génériques pour d'autres applications pour le traitement d'images.

Bibliographie

[1] D.Pierrick HORDÉ, « Imagerie médicale-définition », Sante-Medecine (santemedecine.commentcamarche.net) le journal des femmes, Juin 2014.

[2] S.benfriha et S.hamel, « Segmentation d'image par Coopération région-contours », Mémoire Master Université KasdiMerbah-Ouargla Professionnel, Domaine : Informatique et Technologiede l'Information, 2016.

[3] M.Sandeli, « Traitement d'images par des approches bio-inspirées application à la segmentation d'images », Mémoire de Magister en informatique université Constantine 2. 2014.

[4] Rafael C. Gonzalez et Richard E.Woods, « Digital image processing », article Bibliothèque du Congrès Catalogage-en-publication2008

[5] Mohammed Ben Abdallah, « Outils de compression et de crypto compression: applications aux images fixes et vidéo », thèse doctorat Faculté des Sciences, 4 Avenue Ibn Battouta ,2007.

[6] Didier Müller, « Traitement d'images », cours, 2juin 2016

[7] Edmond Fernandez, « Caracteristiques techniquesdes images numeriques », Archives Nationales. Outre-Mer, avril 2010.

[8] Ibrahim Guelzim, « Contributions aux traitements d'images perspectives et omnidirectionnelles par des outils statistiques », thèse de doctorat faculté des sciences rabat – maroc, informatiques et télécommunications, 12 mai 2012.

[9] Sèmiyou Ayélé Osseni, « Nano-plateformes hybrides multimodales pour l'imagerie médicale », thèse de doctorat l'université de Toulouse, 14 décembre 2012.

[10] Juliette Selb, « Source virtuelle acousto-optique pour l'imagerie des milieux diffusants », thèse de doctorat de l'université Paris XI, 2002.

[11] Julien Nauroy, « Traitements interactifs d'images radiologiques et leurs applications cliniques. Informatique », thèse de doctorat Université Paris Sud - Paris XI, 2010.

[12] D.Le Bihan, J.-F. Mangin, C. Poupon, livre pédagogique, « l'imagerie médical » Réalisation : Agence Gimmik - Janvier 2017.

[13] Guillaume Trébuchet, « Segmentation par contours actifs de séquences de vélocimétrie IRM Application aux artères carotides », thèse doctorat Nantes Angers Le Mans, 27 septembre 2013.

[14] Haz-Edine Assemlal, « Traitement et analyse d'images IRM de diffusion pour l'estimation de l'architecture locale des tissus », thèse doctorat Université de Caen, 2010. Français.

[15] Olfa Kanoun, mohamed Hedi Kallel & Mohamed Salim Bouhlel, « Adaptativité du quantificateur vectoriel à la compression d'images médicales par réseau de neurones », article institut supérieure de biotechnologie d'sfax (isbs), université d'sfax, 2007.

[16] A. Pinti,.Hédoux , E.Watelain,G.Kemoun, B.Boluix, « Comparaison à partir d'IRM de caractéristiques biomécaniques de membres inférieurs sains et pathologiques », septembre 2000.

[17] Nacéra Benamrane, Zakaria Benahmed Daho, Jun Shen, « Compression des images medicales fixes par réseau de neurones », Université des Sciences et de la Technologie d'Oran 2012.

[18] ZEROUAl DJAZIA, « Implémentation d'un environnement parallèle pour la compression d'images a l'aide des fractales", thèse université de Batna, 2016.

[19] Andràs Cziho, « Quantification vectorielle et compression d'image. Application à l'imagerie médicale », Thèse Université de Rennes I, Mai 1999.

[20] F.Truchete, « Ondelettes pour le signal numérique », Edition Hermès, Paris 1998.

[21] M.Bergounioux, « Mathématiques pour le traitement du signal », Cours et Exercices corrigés, 2ème édition, Dunod 2012.

[22] F.Truchete, « Ondelettes pour le signal numérique », livre, edition Hermès, Paris 1998

[23] Jean-Pol Guillement, « Analyse de Fourier - Ondelettes», Cours D´épatement de Mathématiques, Nantes 2010/2011

[24] NulHaq k.hayat S.h.sherazi et W.puech, «Segmentation through dwt and adaptive morphological closing », article institute comsats, 2 /7/2013

[25] R.Soulard, « Ondelettes analytiques et monogènes pour la représentation des Images couleur », Thèse de Doctorat de l'université de Poitier, 2012.

[26] Gwénolé Quellec, « Traitement du Signal Indexation et fusion multimodale pour la Recherche d'information par le contenu. Application aux bases de données d'images médicales », Thèse de doctorat, Université de Rennes I, 2008.

[27] Hocine Bekkouche, « Synthèse de banc de filtre adapté, application à la compression des images ». Thèse de doctorat université Paris-sud XI. 2007.

[28] MARIA E. ANGELOPOULOU AND PETER Y. K. CHEUNG, Département de génie électrique et électronique, Imperial College de Londres. KONSTANTINOS MASSELOS. Département de l'informatique et de la technologie, Université du Péloponnèse, Greece. YIANNIS ANDREOPOULOS Département de génie électronique, Queen Mary University of London, « Implementation and Comparison of the 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs». 2007.

[29] DJ.Jannesquin, R.dirigé, « Implémentation de la Transformée en Ondelettes par l'approche *Lifting Scheme* sur GPU », 2015.

[30] Mr Gas, « Carte de Kohonen pour la quantification de fonction», mémoire master SDI, université pierre- marie curie, septembre 2007.

[31] C. Foucher, « Algorithmes neuronaux et non neuronaux de construction de dictionnaire pour la quantification vectorielle en traitement d'images », thèse en traitement du signal d'Université de Rennes, Décembre 2002.

[32] M.Talibi-Alaoui, R.Touahni, A.Sbihi, « Classification des Images Couleurs par association des transformations Morphologiques aux Cartes de Kohonen» Laboratoire Image et Reconnaissance des Formes, Université Ibn Tofail. Maroc Cari, 2004.

[33] Kouadri Boudjelthia Mohamed, «Réseau de Kohonen les Carte Auto-Organisatrices », thèse master Université des Sciences et de la Technologie d'Oran USTO M.B, 2012.

[34] Zhou Wong, Alan C.Bovik, Hamid R.Shiekh, Eero P.Simoncelli, « Image quality assessment : From Error Visibility To Structural Similarity ». IEEE. 2004.

[35] Haider Ismael Shahadi, Razali Jidin, Wong Hung Way et Yasir Amer Abbas, « Efficient FPGA Architecture For Dual Mode Integer Haar Lifting Wavelet Transform Core ». Département de génie électronique, Université de Babylon, Hila, Babil, Iraq. Université nationale Tenaga, Malizia. 2014.

[36] Mr.Murali Mohan.S et Dr. P.Sathyanarayana, « Modified Lifting Scheme Algorithm for DWT with Optimized Latency & Throughput and FPGA Implementation for Low Power &

Area ». Dept. De ECE, SVCETChittoor, A.P., India, College of Engineering S.V.University, Tirupati, A.P., India. 2016.

[37] Christophe Foucher et Gilles Vaucher, « compression d'images et réseau de neurone », Sup_elec, _Equipe _Electronique, Traitement du Signal et Neuromim_etisme (ETSN), avenue de la Boulaie, BP 28, 35511 Cesson Sévigné Cédex.

[38] Christophe Amerijckx, Michel Verleysen, Philippe Thissen, et Jean-Didier Legat, *Member, IEEE.* « Image Compression by Self-Organized Kohonen Map ». IEEE, 1998.

[39] A. Mebarki, M. Mahmoudi, N. Benamrane. « Compression des images animées suivant une approche basée sur la quantification vectorielle ». Equipe Images-Vision (Laboratoire SIMPA)–USTOMB,2006.

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

[41] Gregorio Bernabé, Jose M. García, José González. « A lossy 3D wavelet transform for high-quality compression of medical video», déppartement de l'ingénieur et de technologie de computer, université de Murcia, centre de recherche d'intel de Barcelon, Espain, 2009.

Annexe 01

➢ La technologie FPGA

Les FPGA (Field Programmable GateArrays 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 à fin 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

➢ 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).

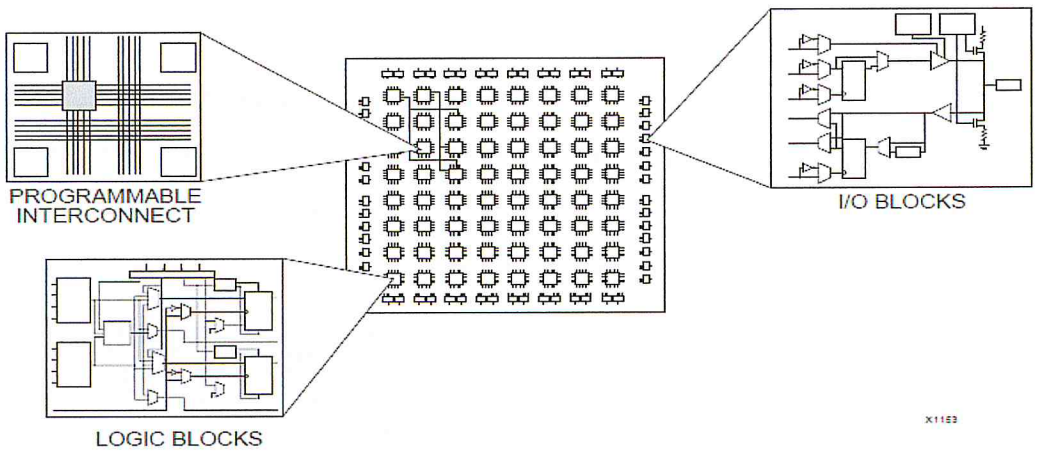


Figure 1 : Architecture générale d'un FPGA.

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)

Annexe 01

-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.

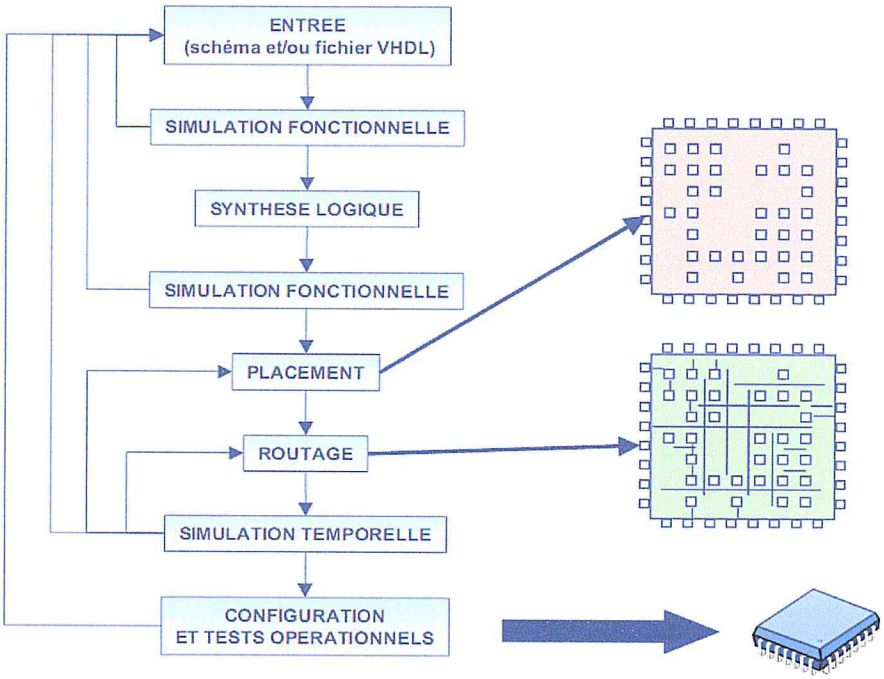


Figure 4: Flot de la conception d'un circuit FPGA.

➢ 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.

Annexe 01

➢ 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.

➢ 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

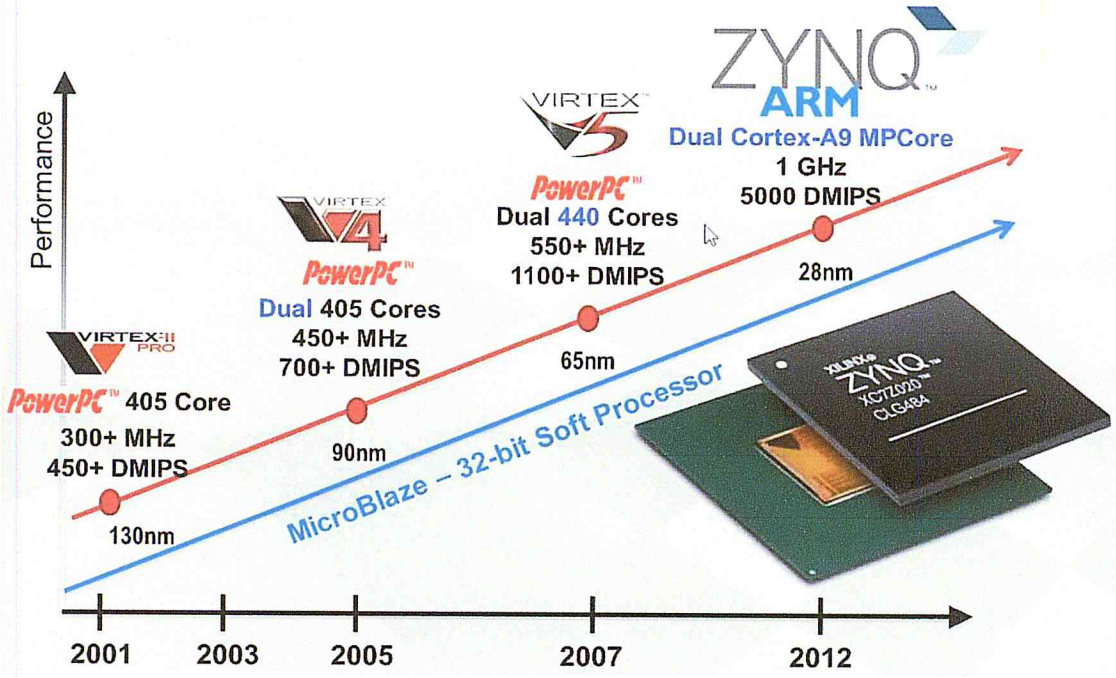


Figure 5 : Héritage de Xilinx.

➢ **Flot de conception entre la partie PS et PL**

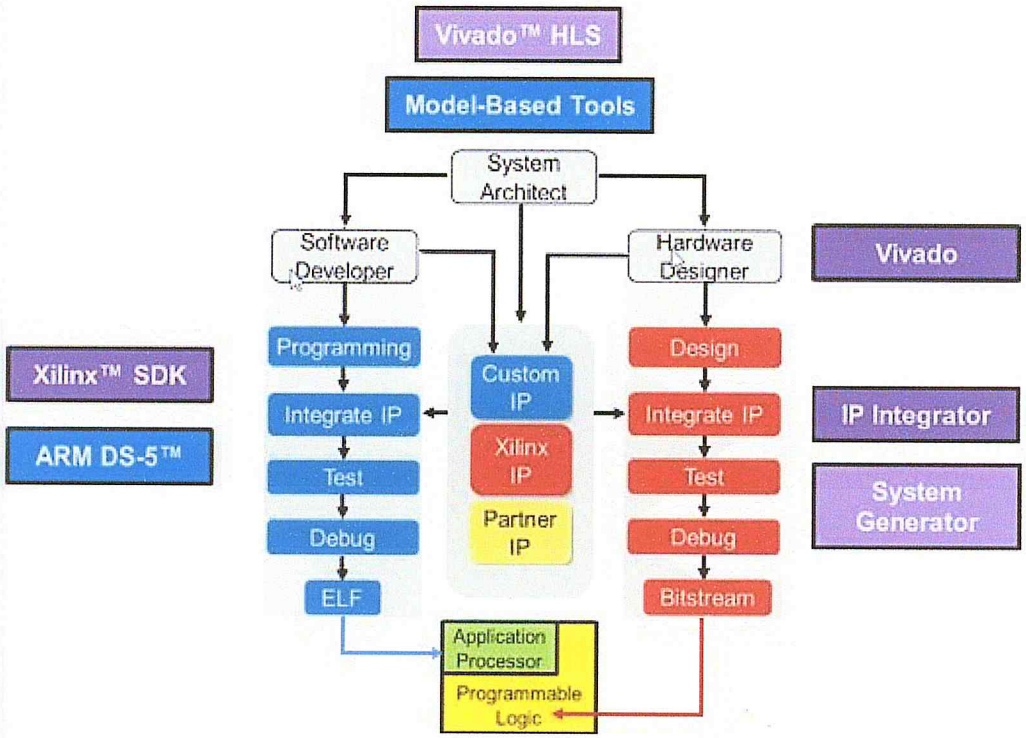


Figure 5: Flot de conception entre la partie PS et PL

Annexe 02

➢ Le standard Jpeg

Jpeg est l'acronyme de «Joint Photographic Experts Group» qui est le comité ayant établi la norme Jpeg ISO 10918-1 [ISO/IEC/10918-1/ 1994]. Ce standard a vu le jour en 1992. Il est devenu le format le plus populaire pour le stockage de photographies numériques, il offrait une meilleure compression que tous les formats bitmap connus à l'époque.

Le standard avait pour objectif de concevoir une nouvelle norme de compression, avec les différentes contraintes suivantes :

• La norme Jpeg doit être très proche des nouvelles techniques de compression, en termes de taux de compression, qualité de restitution et de temps de calcul.

• Jpeg doit pouvoir compresser tout type d'images réelles (tailles différentes, images multi-composantes).

• L'algorithme doit être implémenté sans problèmes sur une grande gamme de CPUs et sur des cartes plus spécialisées ;

• Le codage doit pouvoir être séquentiel, progressif, sans perte et hiérarchique, définissant il existe 4 modes de fonctionnement.

➢ Le standar Jpeg2000

Le comité Jpeg, conjointement avec le comité JBIG (Joint Bilevel Image ExpertsGroup), a lancé un appel d'offre pour un nouveau standard de compression d'image en 1997. Le but était de définir un nouveau standard, en proposant de nouvelles caractéristiques, et en améliorant les performances des caractéristiques existantes dans les standards précédents. . Le standard de Jpeg2000 [ISO/IEC/15444-1/ 2000] pour valider ces différents points :

• Améliorer les performances pour la compression très bas débit.

• Proposer un algorithme assurant à la fois une compression avec perte et une compression sans perte.

• Permettre la transmission progressive de l'image.

• Permettre la sélection d'une région d'intérêt (ROI), et son codage avec moins de distorsion.

• Renforcer la robustesse aux erreurs.

• L'image est décomposée en plusieurs composantes. L'image et ses composantes sont partitionnées en blocs qui seront compressés séparément. Les valeurs de chaque bloc sont centrées par rapport à la moyenne du bloc.

•Une transformation discrète d'ondelette DWT.

➢ Digital Image Communications in Medicine (DICOM)

La NEMA (National Electrical Manufacturers Association) ont développé en 1992 le standard DICOM pour faciliter l'interconnexion des systèmes d'imagerie médicale aux réseaux. Ce format dispose de protocoles d'échange et d'une interface de communication, soit par l'OSI (Open Systems Interconnect), soit par TCP/IP (Transmission Control Protocol/ Internet Protocol). En plus de l'image numérique, une information texte est fournie par le format, et renseigne sur l'examen effectué. DICOM autorise une compression de l'image par le format Jpeg.

Annexe 03

➢ Le protocole RS232

Dans une communication série RS232, les bits sont envoyés les uns à la suite des autres sur la ligne en commençant par le bit de poids faible. La transmission s'appuie donc sur le principe des registres à décalage. La transmission se fait octet par octet :

- pas d'horloge transmise
- Nécessité de rajouter un bit de "START" ('0' logique) avant l'octet à transmettre, et un bit de "STOP" ('1' logique) après l'octet à transmettre.
- La norme RS232 prévoit également la possibilité de rajouter un autre bit juste avant le bit de STOP :
  - Bit de parité
  - ou un 2ème bit de STOP

- 10 ou 11 bits sont transmis au registre à décalage qui assure la transmission en commençant par le bit de poids faible.

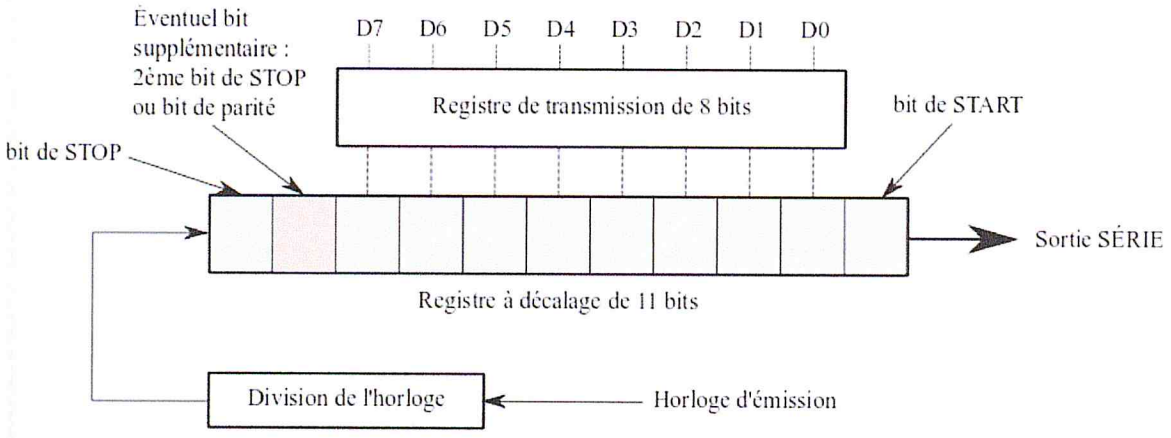


Figure 7 : Principe de protocole RS232

Annexe 03

➢ Langage Java

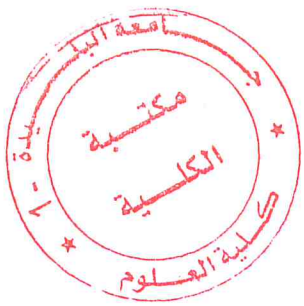
Java est un langage orienté objet : un programme Java est centré complètement sur les objets et fournit un ensemble prédéfini de classes facilitant la manipulation des entrées-sorties, la programmation réseau, système, graphique…

Le langage Java est distribué : il est conçu pour développer des applications en réseau, les manipulations des objets distants ou locaux se font de la même manière.

Le langage Java est robuste et sûr : il est fortement typé, il élimine bien des erreurs d'incohérence de type à la compilation et ne supprime pas tous les problèmes de sécurité mais le réduit fortement.

Le langage Java est interprété : un programme Java n'est pas compilé en code machine, il est transformé en code intermédiaire interprété.

Le langage Java est portable et indépendant des plates-formes : un Interface d'exécution ne doit ni dépendre de l'architecture matérielle, ni du système d'exploitation.



Annexe 04

➢ **Programme VHDL**

```
entity p_1_cpu_wrapper is
 port (
  DDR_addr : inout
STD_LOGIC_VECTOR ( 14 downto 0 );
  DDR_ba : inout
STD_LOGIC_VECTOR ( 2 downto 0 );
  DDR_cas_n : inout STD_LOGIC;
  DDR_ck_n : inout STD_LOGIC;
  DDR_ck_p : inout STD_LOGIC;
  FIXED_IO_ps_clk : inout STD_LOGIC;
  FIXED_IO_ps_porb : inout
STD_LOGIC;
  FIXED_IO_ps_srstb : inout
STD_LOGIC
 );
end p_1_cpu_wrapper;

architecture STRUCTURE of
p_1_cpu_wrapper is
 component p_1_cpu is
 port (
  DDR_cas_n : inout STD_LOGIC;
  DDR_cke : inout STD_LOGIC;
  DDR_ck_n : inout STD_LOGIC;
  .
  .
  .
  FIXED_IO_ps_porb : inout
STD_LOGIC
 );
 end component p_1_cpu;
begin
p_1_cpu_i: component p_1_cpu
   port map (
    DDR_addr(14 downto 0) =>
DDR_addr(14 downto 0),
    DDR_ba(2 downto 0) => DDR_ba(2
downto 0),
    DDR_cas_n => DDR_cas_n,
    DDR_ck_n => DDR_ck_n,
    .
    .
    .
    FIXED_IO_ps_srstb =>
FIXED_IO_ps_srstb
  );
end STRUCTURE;
```

Annexe 04

➢ Programme MATLAB de la méthode proposée

```
close all; clear all; clc;

load dicqs_plus_d_appr.mat;

load dictQV_mo.mat

[x
y]=uigetfile('C:\Users\me\Desktop\traveau
x mémoire\diction\*dcm');

f=[y x];

image=dicomread(f);

figure; imshow(image)

% application de filtre legall lifting scheme
sur l'image

[a h v d]=legall_lifting(image); title('image
originales');

%% application de la QS pour la bande LL

aapp=erq1(a,dic);

%% application de la QV pour la bande
LH , HL et HH

happ=erqv2(h_n,qv_h);

vapp=erqv2(v_n,qv_h);

dapp=erqv2(d_n,qv_h);

%%reconstruction de la bande 'LL'

for i=1:z(1)

  for j=1:z(2)

  a_re(i,j)=dic(aapp(i,j));

  end

end

%reconstruction de bandes 'LH', 'HL','HH'

sb=size(qv_h{1,1});

n=size(h);

for i=1:sb(1):n(1)

  for j=1:sb(2):n(2)

  h_re(i:i+sb(1)-1,j:j+sb(2)-
1)=qv_h{happ((i+sb(1)-1)/sb(1),(j+sb(2)-
1)/sb(2))};

  v_re(i:i+sb(1)-1,j:j+sb(2)-
1)=qv_h{vapp((i+sb(1)-1)/sb(1),(j+sb(2)-
1)/sb(2))};

  d_re(i:i+sb(1)-1,j:j+sb(2)-
1)=qv_h{dapp((i+sb(1)-1)/sb(1),(j+sb(2)-
1)/sb(2))};

  end

end

% application de l'ondelette inverse de le
gall

image_re=legall_inverse(a_re,h_re_n,v_re
_n,d_re_n);

%% affichage de l'image reconstruite

figure; imshow(image_re)

%%calcule de PSNR

peaksnr_image_re=psnr(image_re,image)

%% Calacule de SSIM

sss=ssim(image_re,image)
```

Annexe 04

➢ Code en C

```
while (1){
        int i=inValeur16Bit(); //choix de l'opperation
        if(i==1) //compression
        {       charg();
                dwt();
                Qs();
                Qv();
                env_band(); }
        if(i==2) //decompression {
                rec();
                dwt_inv();
                env_inv();          }
        if(i==3)
                env_mapp();
        if(i==4){
        resv_mapp();
        }}
return 0;
}
```

Annexe 04

➢ Code en Java

```
public class read_image {

    static float[][] M;

    public static DICOM dcm = new
DICOM();

    public static ImageProcessor imgPro;

    public static short[] pixels;

    public static int n, m;

    public static void read() throws
NoSuchPortException,
PortInUseException,
UnsupportedCommOperationException,
IOException {

        //------rad DICOM Image-----

        dcm.open(Interface.s);

        imgPro = dcm.getProcessor();

        pixels = (short[]) imgPro.getPixels();

        n = imgPro.getHeight();

        m = imgPro.getWidth();

        //----convertion de vecteur a matric et
normalisation----------

        int i, j;

        M = new float[n][m];

        for (i = 0; i < n; i++)

            for (j = 0; j < m; j++)

                M[i][j] = pixels[i * n + j] / 1023;

    }

}
```

Annexe 04

// commenication rs232

```
public void outValeur16Bit(int valeur) {
    byte b;
    try {
        for (int i = 0; i <= 1; i++) {
            b = (byte) (valeur >> (8 * i));
            out.write(b);
        }
        out.flush();
    } catch (IOException e) {
        // TODO Auto-generated catch
block
        e.printStackTrace();
    }
}
```

```
public int inValeur16Bit() {
    int valeur = 0;//byte b;
    for (int i = 0; i <= 1; i++)///3 o lieu 4
    {
        try {
            valeur |= in.read() << (i * 8);
        } catch (IOException e) {
            // TODO Auto-generated catch
block
            e.printStackTrace();
        }
    }
    return valeur;
}
```