UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

ARCHITECTURE DE TRANSFORMÉE DE COSINUS DISCRÈTE SUR DEUX DIMENSIONS SANS MULTIPLICATION ET MÉMOIRE DE TRANSPOSITION

Mémoire de maîtrise
Spécialité : génie électrique

Alexandre DUGAS

Jury : François MICHAUD (directeur)
Chon-Tam LE DINH (codirecteur)
Jean ROUAT
Daniel DALLE

Sherbrooke (Québec) Canada Novembre 2012

À ma conjointe, Lai-Lin Ng et à mes parents

RÉSUMÉ

Au cours des dix dernières années, les capacités technologiques de transmission vidéo rendent possible une panoplie d'applications de télésanté. Ce média permet en effet la participation de médecins spécialisés à des interventions médicales ayant lieu à des endroits distants. Cependant, lorsque ces dernières se déroulent loin des grands centres, les infrastructures de télécommunication n'offrent pas un débit assez important pour permettre à la fois une transmission d'images fluides et de bonne qualité. Un des moyens entrepris pour pallier ce problème est l'utilisation d'encodeur et décodeur vidéo (CODEC) permettant de compresser les images avant leur transmission et de les décompresser à la réception.

Il existe un bon nombre de CODEC vidéo offrant différent compromis entre la qualité d'image, la rapidité de compression, la latence initiale de traitement et la robustesse du protocole de transmission. Malheureusement, aucun n'est en mesure de rencontrer simultanément toutes les exigences définies en télésanté. Un des problèmes majeurs réside dans le délai de traitement initial causé par la compression avec perte du CODEC.

L'objet de la recherche s'intéresse donc à deux CODEC qui répondent aux exigences de délais de traitement et de qualité d'image en télésanté, et plus particulièrement pour une application de téléassistance en salle d'urgence. L'emphase est mise sur les modules de quantification des CODEC qui utilisent la transformée en cosinus discrète. Cette transformée limite la transmission des images vidéo fluide et quasi sans délais en raison des délais de traitement initiaux issus des nombreuses manipulations arithmétiques qu'elle requiert.

À l'issu de la recherche, une structure efficace de la transformée en cosinus est proposée afin de présenter une solution au temps de latence des CODEC et ainsi de répondre aux exigences de télécommunication en télésanté. Cette solution est implémentée dans un CODEC JPEG développé en VHDL afin de simuler un contexte d'application réelle.

Mots-clés : Transformée de cosinus discrète, JPEG, compression d'image, FPGA, Réseau de neurones artificiels à propagation avant

REMERCIEMENTS

Ce travail est le résultat de la mise en application de divers théorèmes mathématiques et théories relatives au traitement numérique des signaux. Cette aventure a été partagée par ma conjointe, ma famille et mes amis, à qui je dois un support inconsidéré et des appuis qui m'ont permis d'avancer dans les moments les plus difficiles de cette recherche.

Je désire aussi adresser mes plus sincères remerciements au professeur Chon Tam Le Dinh et au professeur François Michaud en ce qui concerne leur support financier (via un financement par le programme ACELP/3IT du Département de génie électrique et de génie informatique de l'Université de Sherbrooke), technique et moral dont ils ont fait preuve tout au long de mon travail. Leur apport méthodologique et leur analyse critique de mon travail m'ont permis de livrer une contribution au domaine de la recherche en traitement d'image.

En terminant, je tiens à remercier Mme Danielle Gagnée et Mme Claudia Carbonneau du secrétariat du Département de génie électrique et génie informatique ainsi que Mme Linda Simoncelli pour le soutien administratif conféré pendant mes travaux.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LEXIQUE

Terme technique	Définition
H.261	Standard de compression vidéo pour la vidéoconférence, issu des recommandations de l'*International Telecommunication Union* (ITU).
H.263	Standard de compression vidéo pour la vidéoconférence, issu des recommandations de l'*International Telecommunication Union* (ITU).
H.264	Standard de compression vidéo connu aussi sous le nom d'AVC, ce CODEC incorpore de la compensation de mouvement lors de la compression de la vidéo.
Structure Papillon	Un papillon est une structure qui découle de l'algorithme Cooley-Turkey. Ce dernier consiste à fractionner une transformée en plusieurs sous-transformées de façon récursive dans le but de paralléliser les manipulations mathématiques.

LISTE DES SYMBOLES

Symbole	Définition
bpp	Bits par pixel
fps	*Frame per second*, il s'agit de la cadence vidéo en champ par seconde (trame / seconde).
Gb	Gigabit
Gb/s	Gigabits par seconde
Go	Giga-octet
Go/s	Giga-octets par seconde
kb	Kilobit
kb/s	Kilobits par seconde
Ko	Kilooctet
Ko/s	Kilooctets par seconde
Mb	Mégabit
Mb/s	Mégabits par seconde
MHz	Mégahertz
Mo	Mégaoctet
Mo/s	Mégaoctets par seconde
ms	milliseconde
ns	nanoseconde

LISTE DES ACRONYMES

Acronyme	Définition
CODEC	Il s'agit d'une paire de Codeur et d'Encodeur servant à effectuer un traitement de signal quelconque sur une image, un flux audio, ou autres.
DCT	*Discrete Cosine Transform*, ou Transformée en cosinus discrète, est une transformée mathématique servant la plupart du temps à la décorrélation d'un signal. Le résultat de cette transformée est l'expression du signal dans un nouvel espace transformé.
DCT-2D	Version de la DCT appliquée sur deux dimensions. Elle est employée surtout en traitement d'image.
DFT	*Discrete Fourier Transform*, ou Transformée de Fourier Discrète, est la version discrète de la transformée de Fourier.
DWT	*Discrete Wavelet Transform*, ou transformée en ondelettes discrètes, est une transformée servant à identifier la localisation et l'information fréquentielle reliée à un signal.
FFNN	*Feed Forward Neural Network*, ou réseau de neurones artificiels à propagation avant.
FPGA	*Field Programmable Gate Array* est une technologie de circuit intégré qui permet la mise en œuvre de circuits logiques à partir d'un champ de portes logiques programmables.
FT	*Fourier Transform*, ou transformée de Fourier, est une transformée servant à transposer l'information temporelle d'un signal au domaine fréquentiel.
GPU	*Graphic Processing Units* se dit d'une unité de processeur dédiée au traitement graphique.
ITU	*International Telecommunication Union.*
JPEG	*Joint Photographic Expert Group*.
JPEG2000	*Joint Photographic Expert Group 2000.*
MJPEG	*Motion Joint Photographic Expert Group*. Il s'agit d'un CODEC vidéo.
MPEG-1	*Moving Picture Expert Group* -1. Il s'agit d'un standard de compression vidéo.
MPEG-2	*Moving Picture Expert Group* -2. Il s'agit d'un standard de compression vidéo.
MSE	*Mean Square Error*, soit l'erreur quadratique moyenne.
PAR	*Place And Route*. Il s'agit d'une étape de la synthèse FPGA.
PSNR	*Peak Signal to Noise Ratio*, soit le rapport entre signal maximum et le bruit.

RLE	*Run-Length Encodin*g est une méthode de codage sans perte visant à compresser les séries de symboles identiques.
RTSS	Réseau de télécommunications socio-sanitaire.
SIF	*Source Input Format*
SMPTE	*Society of Motion Picture and Television Engineers*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very-High-Speed Integrated Circuits*

CHAPITRE 1
INTRODUCTION

L'utilisation de la vidéo dans les domaines de la santé est en plein essor depuis la dernière décennie. Elle devient de plus en plus utilisée pour des applications telles que la biométrie ou les opérations médicales dirigées à distance. De ces applications ressortent aussi des contraintes liées à la fois à la localisation des équipements de traitement vidéo et aux ressources matérielles disponibles pour effectuer le traitement des images. Prenons l'exemple du traitement des images vidéo dans une application de télérobotique mobile. Dans ce contexte, la vidéo est souvent utilisée à des fins de perception de l'environnement et du positionnement du robot dans ce dernier. Dans certains cas, les images doivent être transmises à un utilisateur distant qui le contrôle à partir d'un lien Ethernet ou autre. Comme le canal de communication possède une bande passante limitée et que cette dernière n'est pas seulement dédiée au transfert des images, la vidéo doit être compressée à l'aide d'un codeur. Plus la compression est importante, plus le nombre de bits requis pour représenter l'image est petit. En revanche, le temps requis pour effectuer la compression et la décompression augmente, et la qualité de l'image reconstruite est, en règle générale, diminuée.

La situation décrite est propre à un problème typique de transmissions d'image. L'application qui requiert l'utilisation de la vidéo exige un niveau de qualité de reconstruction d'image défini et une vitesse de traitement qui repose sur la cadence désirée des images vidéo. Les exigences de qualité et de vitesse de traitement sont assurées par la largeur de bande du canal de communication et par le codeur. Dans l'optique où l'un ou l'autre de ces éléments ne peut répondre adéquatement en termes de débit et de rapidité d'exécution, la chaîne de traitement vidéo ne peut rencontrer les exigences de l'application car il y a alors perte d'information ou apparition d'un délai dans la transmission [Chang, 2000].

Un système de téléassistance en salle d'urgence est sujet aux mêmes contraintes. Il fut établi que [Auclair Beaudry, 2009] :

- Les images vidéo transmises doivent satisfaire la contrainte de faible débit afin qu'elles puissent être transférées dans des médiums de communication restreints du calibre de celui d'une ligne téléphonique. Dans le contexte de la téléopération au Québec, il faudrait que le débit des images soit d'environ 384 kb/s [Le Dinh, 2009a] pour s'assurer que les images puissent être transférées via le Réseau de Télécommunications Socio-Sanitaire (RTSS) [Lemieux *et Al.,* 2007], liant les grands centres aux régions (qui ne disposent pas alors de réseau de télécommunication à large bande).

- La qualité des images reconstruites ne peut être compromise [Devi Gukhool, 2009] [Auclair Beaudry, 2009].

- Les délais associés à la compression et à la décompression du flux vidéo doivent être presque imperceptibles du point de vue de l'utilisateur final [Lemieux *et Al.,* 2007].

Il ne reste plus qu'un seul élément pouvant faire preuve de flexibilité pour rendre possible la téléopération dans les régions éloignées, soit la paire Codeur-Décodeur (CODEC) vidéo, responsable de la compression et de la décompression des images.

[Lemieux *et Al.,* 2007] fait état d'expériences visant à apporter des solutions viables aux contraintes imposées par la téléassistance en salle d'urgence. Ces expériences sont réalisées à l'aide d'un CODEC vidéo T3000 de Tandberg. Ce dernier est destiné à des applications de vidéoconférence. Or, la téléassistance en salle d'urgence, impliquant la transmission simultanée de deux flux vidéo [Lemieux *et Al.,* 2007], est une application ayant des exigences différentes en termes de débit vidéo et de qualité de reconstruction des images. Les applications de vidéoconférence conventionnelles font l'hypothèse qu'un flux vidéo est réservé à la transmission des diapositives de présentation, et que l'autre est dédié au conférencier. [Auclair Beaudry, 2009] explique d'ailleurs que ce type de CODEC est inapproprié pour l'application de téléassistance en salle d'urgence car celle-ci requiert une bande passante de même largeur sur les deux canaux pour la transmission d'images provenant des deux caméras. De plus, le CODEC T3000 repose sur les standards H.263 et H.261 qui constituent le

minimum recommandé pour la téléconférence [Keith, 2005]. L'architecture est similaire à celle du *Motion Joint Photographic Expert Group* (MJPEG), ce qui rend difficile la dégradation progressive de la vidéo pour accommoder la bande passante du canal [Keith, 2005]. Les débits anticipés par le T3000 sont de l'ordre de 500 kb/s à 1.5 Mb/s, ce qui dépasse la capacité du Réseau de Télécommunications Socio-Sanitaire (RTSS).

D'autres solutions sont présentes sur le marché. Le MAKO-HD ou l'INTERN de *HaiVision* et *Tandberg* en sont quelques-unes. Bien que la qualité d'image soit respectée par ces solutions, le réel problème demeure la bande passante du canal requise par ces équipements. Ces derniers utilisent les CODEC H.264, employés abondamment dans le domaine de la télédiffusion. La bande passante requise s'étale alors de 500 kb/s jusqu'à 10 Mb/s [Keith, 2005]. Ces solutions sont viables si un lien de communication large bande est dédié au transfert de vidéo et si le réseau de communication n'est pas soumis aux fluctuations d'un trafic public. D'autre part, la majorité de ces produits proposent un temps de latence de l'ordre de 300 ms à 500 ms. Ces délais sont acceptables pour de la télédiffusion, mais inappropriés dans le cadre de l'application de téléassistance en salle d'urgence où un spécialiste de la santé agit en rétroaction, en temps réel, sur une intervention médicale.

Des travaux ont alors été entrepris à l'Université de Sherbrooke sur l'élaboration de deux CODEC logiciels qui répondent à ces contraintes [Auclair Beaudry, 2009], [Devi Gukhool, 2009] et proposant des solutions prometteuses en termes de qualité des images reconstruites et de débit des images compressées. Elles offrent la possibilité de modifier le débit et la qualité de l'image selon les exigences d'une application en télésanté. Toutefois, comme il s'agit de solutions logicielles, elles ne sont pas adaptées pour fonctionner sur du matériel dédié au traitement d'image. Il est effectivement peu recommandable d'utiliser un système ordiné comme ceux employés lors du développement de ces deux CODEC pour effectuer les tâches de compression du flux vidéo [Brice, 2005]. Les systèmes d'exploitation de ces derniers sont sujets à des améliorations continues et ne présentent pas une plateforme stable pour un éventuel déploiement. Une grande lacune de ces systèmes est le traitement séquentiel des opérations requises pour la compression des images. Ces opérations impliquent une

accumulation de l'image en mémoire [Stallings, 2006] qui se traduit éventuellement par des délais perceptibles. Les temps de latence restent alors trop élevés pour une application directe.

Aucune des solutions précédentes ne semble donc répondre entièrement aux trois contraintes énoncées dans l'introduction. Sur le plan matériel, les composantes d'un système ordiné ne sont pas organisées pour accélérer le traitement de la vidéo. Les composantes de ces derniers reflètent plutôt une application multimédia, à l'exception des accélérateurs graphiques (GPU) qui sont dédiés à la vidéo. Ces derniers sont toutefois optimisés pour effectuer des tâches de compression propres au H.264 et autres standards vidéo qui ne répondent pas aux besoins mentionnés dans l'introduction [Stallings, 2006]. L'idée est donc d'effectuer une mise en œuvre des algorithmes développés par [Auclair Beaudry, 2009] [Devi Gukhool, 2009] sur du matériel dédié à la tâche de compression. Il faut éviter le traitement séquentiel des images et effectuer une refonte de ces algorithmes présentant des performances adéquates. Le but ultime est d'arriver à atteindre les performances de traitement en temps réel avec des méthodes de compression et décompression répondant aux spécifications des applications visées en télésanté.

1.1 Spécifications globales

La notion de traitement en temps réel des images est relative à l'application finale du CODEC. Les CODEC commerciaux tels le H.264 n'offrent pas cette possibilité de traitement en temps réel. Des retards de l'ordre de la seconde sont observables dans des circonstances normales d'opération du CODEC. Il est donc primordial de définir ce qu'est la notion de temps réel dans le présent travail de maîtrise pour que cette dernière rejoigne les spécifications propres à un contexte de téléassistance en salle d'urgence. Sans faire référence au débit résultant de la compression, le temps de traitement en temps réel doit aussi correspondre au temps de traitement de tous les pixels d'une image à une cadence donnée. Il est donc nécessaire de définir un format d'image avant de définir la contrainte de traitement à temps réel.

1.1.1 Le format d'image

Le format d'image retenu pour la compression est le *Source Input Format* (SIF), tel que défini selon la norme *Moving Picture Expert Group-1* (MPEG-1). Ce dernier se caractérise par les résolutions et la cadence (fps) suivantes [Keith, 2005] :

- **Amérique** - 352V × 240H NTSC - 30 fps - progressif[1]
- **Europe et autres régions** - 352V × 288H PAL - 25 fps - progressif

Les lettres H et V correspondent respectivement à la dimension horizontale et verticale de l'image en pixel. Le SIF est un format vidéo qui découle de la norme CCIR-610 (maintenant BT.601) régissant la télévision standard utilisée par les télédiffuseurs. La norme BT.601 définit une résolution de 720 × 480 pixels actifs[2] entrelacés ou 720 × 576 pixels totaux entrelacés (actifs et non actifs) [Keith, 2005]. Le SIF constitue aussi un ensemble de résolutions du format MPEG-1, format utilisé par les caméras numériques et autres équipements de télédiffuseurs pour le transport de contenu vidéo. Les formats MPEG-1 sont englobés par les niveaux régis par MPEG-2 qui est le flux de transport par excellence des télédiffuseurs [Brice, 2005].

En utilisant le standard SIF comme image d'entrée du CODEC, il est possible de rendre ce dernier compatible aux différents réseaux de télédiffusion internationaux. Comme le débit du SIF – 25 fps est équivalent à celui du SIF – 30 fps, le CODEC est automatiquement compatible avec plusieurs régions du globe sans nécessiter d'adaptation[3]. Un avantage supplémentaire à utiliser le standard SIF est que le débit requis par ce dernier dans le cadre du projet est légèrement supérieur à celui d'une petite caméra de résolution 640 × 480 entrelacée [Keith, 2005]. L'image reconstruite par le CODEC peut donc provenir de différents

[1] Signifie que les lignes de l'image sont transmises dans l'ordre progressif (1,2,…*N*+1) et non pas entrelacées tel que le signal NTSC d'une télévision.
[2] Un pixel actif signifie un pixel visible. Une image contient des pixels visibles.
[3] En prenant le nombre total de pixels de la résolution nord américaine multiplié par la cadence d'image, le résultat obtenu est le même que si le calcul était effectué avec la résolution européenne.

équipements ayant une résolution (débit) plus faible et être directement acheminée sans traitement additionnel sur un moniteur ou une télévision [Keith, 2005].

En résumé, le choix du format SIF est stratégique pour l'application visée. Le CODEC peut utiliser n'importe quel équipement de projection respectant les standards de *Society of Motion Picture and Television Engineers* (SMPTE) ou de l'*International Telecommunication Union* (ITU), et donc être utilisé dans des circonstances ou les équipements visuels dédiés à une application de télésanté ne sont pas présents. Le format SIF constitue un sous-ensemble des formats MPEG-1 et MPEG-2 qui sont utilisés couramment comme flux de transport d'image dans les réseaux de télécommunication. Cela assure donc une compatibilité entre le CODEC et plusieurs équipements de capture ou de transport vidéo. Enfin, le format SIF permet d'obtenir un débit d'images constant, peu importe que la cadence vidéo soit de 30 fps ou de 25 fps [Keith, 2005].

1.1.2 Délais introduits par le traitement

Pour assurer une fluidité adéquate des images, la cadence des images vidéo doit être d'au moins 12 images par seconde [Brice, 2005]. Or, pour obtenir une bonne qualité d'image sans avoir de papillotement, la cadence doit correspondre au minium à 24 images par seconde, et ce, en mode progressif [Brice, 2005]. Comme le format d'image SIF est préconisé dans la présente démarche, il est nécessaire d'utiliser une cadence de 30 images par seconde dans un cas limite. Ces données viennent fixer le temps de latence minimum toléré. Les images provenant de deux caméras de format SIF sont constituées d'un équivalent de 259200 pixels après un prétraitement suggéré par [Le Dinh, 2009a]. Le délai minimum tolérable est d'une image par trame complète. Cela correspond à environ 34 ms. En ayant deux flux vidéo à traiter de façon simultanée pour fournir une image stéréoscopique au spécialiste, ce délai se doit d'être divisé par deux. Le CODEC dispose alors de 17 ms pour effectuer le traitement d'image. La compression et la décompression doivent s'effectuer dans ce délai, sans quoi la fluidité de la vidéo sera compromise. Il ne s'agit pas d'une limite à atteindre en termes de délai, mais plutôt d'un seuil à ne pas dépasser. Le calcul effectué ne prend pas en

considération les délais liés au canal de transmission et à la synchronisation des modules du CODEC[4]. Cette recommandation du délai de traitement est plus sévère que l'ensemble des propositions de [Auclair Beaudry, 2009]. Les délais associés à une conversation ne peuvent être appliqués à un contexte de télésupervision en temps réel.

1.2 CODEC considérés

À la lumière des résultats portant sur les travaux de [Auclair Beaudry, 2009] et [Gukhool, 2007] en termes de *Peak Signal-to-Noise Ratio* (PSNR), trois CODEC sont candidats pour la compression d'image dans le cadre de la téléassistance en salle d'urgence : un CODEC JPEG2000 (*Joint Photographic Expert Group committee in 2000*); un CODEC JPEG (*Joint Photographic Expert Group*) et un CODEC de type *Feedforward Neural Network* (FFNN). Les restrictions liées au format d'image et aux délais de traitement conduisent directement à une restriction de débit des CODEC énumérés. En effet, pour que ces derniers soient employés en téléopération, ils doivent pouvoir fournir un débit de traitement de 62,208 Mb/s au minimum pour les pixels entrant.

Bien que ces algorithmes de compression respectent les contraintes liées au débit des images compressées et à la qualité des images reconstruites, ils n'offrent pas un encodage suffisamment puissant pour transmettre les images requises en temps réel sur une bande passante de canal plus faible que 384 kb/s. C'est le cas des implémentations logicielles et matérielles de ces CODEC car les limitations de performance sont liées à l'architecture même de ces derniers. L'objet de la recherche n'est donc pas de proposer des modifications intrinsèques aux CODEC quant à la nature des manipulations des images vidéo, mais de proposer une solution pour accélérer le traitement effectué par ces derniers. À cette fin, une analyse du débit en fonction du PSNR pour les CODEC envisagés est présentée à l'issue des travaux afin de mesurer la qualité des images reconstruites ainsi que la vitesse de traitement.

[4] Les modules de compression et de décompression du CODEC sont de part et d'autre du canal de transmission.

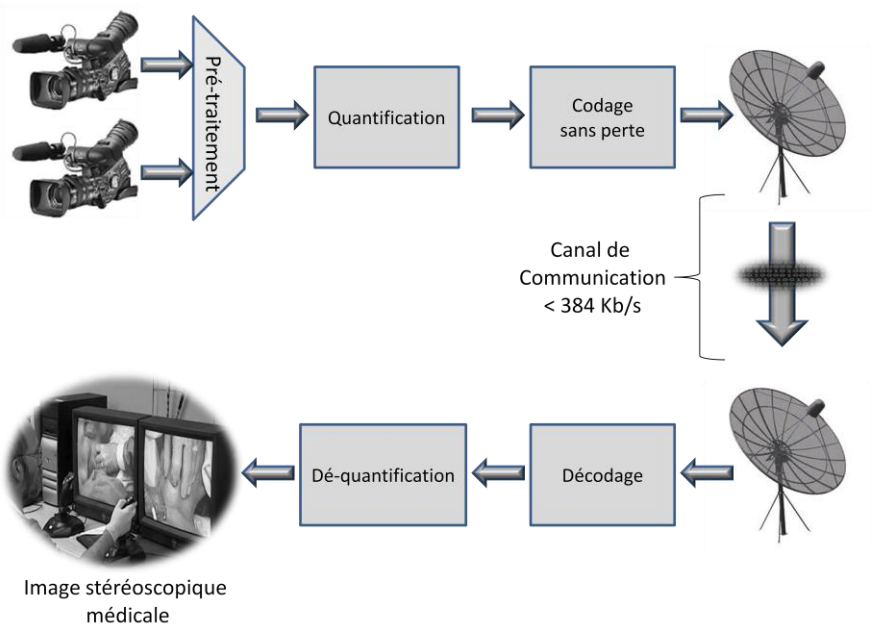

Figure 1.1 - Chaîne de traitement d'une image vidéo

Comme le démontre la figure 1.1, un CODEC vidéo est un ensemble de manipulations mathématiques visant à réduire la taille d'une image en dégradant cette dernière, ou en effectuant du codage sans perte. Ces manipulations s'effectuent dans un ordre séquentiel. La vitesse du CODEC dépend donc de l'élément le plus lent dans la chaîne. En règle générale, ce dernier est le quantificateur. Dans le cas des CODEC considérés, les quantificateurs sont à base de transformée de cosinus discrète sur deux dimensions (DCT-2D) pour le JPEG ou le FFNN, et à base de transformée en ondelettes discrètes (DWT) pour le JPEG2000.

Les quantificateurs sont sujets à introduire un problème de récursivité, débouchant éventuellement sur un temps de traitement inapproprié dans un contexte de téléassistance en salle d'urgence. Ils comportent différentes opérations matricielles ou de filtrage qui se traduisent nécessairement par des opérations mathématiques séquentielles lorsque le CODEC est implémenté dans un système ordiné. Afin de diminuer le temps de traitement du CODEC, il est important de favoriser une implémentation maximisant le parallélisme des opérations matricielles. Une implémentation de CODEC sous une technologie de *Field Programmable Gate Array* (FPGA) est un bon point de départ en soit. Même si des GPU peuvent effectuer des tâches de compression correctement en étant programmés adéquatement, le problème

réside dans la compatibilité de la programmation avec les technologies récentes. Les versions précédentes d'une même famille de GPU possèdent souvent des modules d'interface incompatibles avec les nouvelles versions. Par souci de compatibilité à plus long terme, il est préférable de développer les algorithmes en *VHSIC Hardware Description Language* (VHDL), car il s'agit d'un langage de modélisation et non pas un langage d'utilisation qui requiert une révision adaptée au matériel utilisé [Ashenden, 2001] [Ashenden, 2008]. Ce langage mène directement à la synthèse des algorithmes du CODEC sous forme de logique combinatoire et séquentielle. Plusieurs opérations peuvent ainsi être réalisées en parallèle, débouchant sur un temps de latence beaucoup plus faible qu'un processeur traditionnel non spécialisé si la synthèse cible un FPGA.

Un élément restant à définir est le choix du quantificateur à optimiser. À cet effet, il est nécessaire d'avoir une vue d'ensemble des opérations des trois CODEC candidats afin d'évaluer la complexité de ces derniers. Même si l'objectif est de s'attarder aux quantificateurs, il faut tout de même considérer la longueur totale de la chaîne de traitement pour rencontrer la contrainte de traitement temps réel, sans quoi la recherche ne peut pas s'appliquer au domaine de la téléassistance en salle d'urgence. Cette analyse permet aussi de déterminer le quantificateur retenu qui, une fois optimisé, permettra d'atteindre les performances souhaitées.

La figure 1.2 illustre le traitement effectué par les trois CODEC candidats. Il faut noter que le prétraitement dans le cas des trois CODEC est identique. L'analyse porte alors uniquement sur le quantificateur et le codage sans perte.

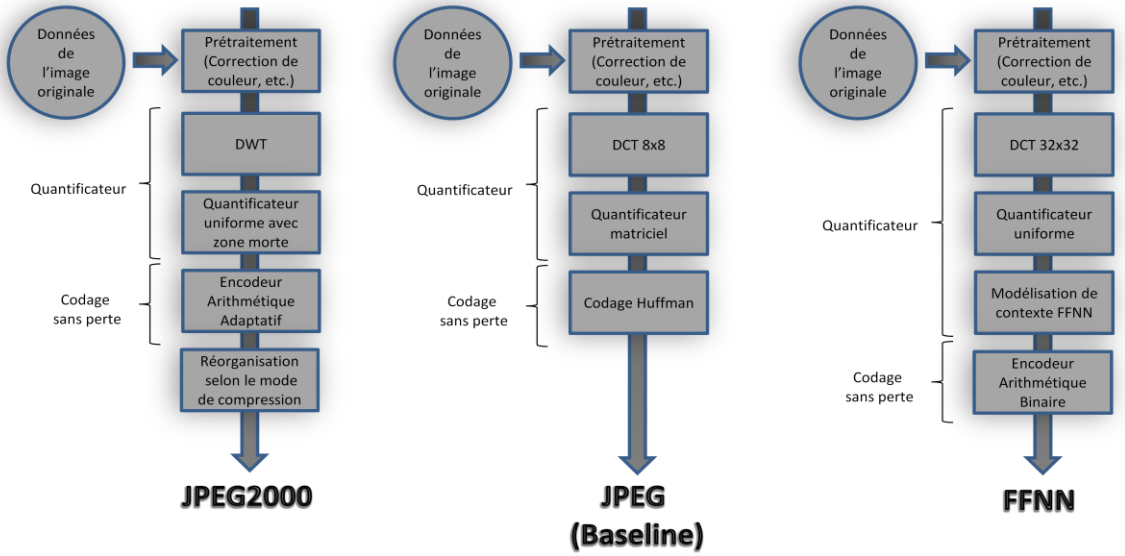

Figure 1.2 - Traitement d'image effectué par les trois CODEC

1.2.1 Analyse du JPEG2000

Le JPEG2000 est un CODEC éprouvé dont les performances et les avantages sont bien connus. Il offre des rapports de compressions allant de 40:1 jusqu'à environ 100:1 et procure aussi une qualité de reconstruction d'image qui dépasse de nombreux CODEC à base de transformées. Il permet la reconstruction à débit variable en fonction des régions d'intérêt d'une image, la reconstruction partielle d'une image et ne nécessite pas de fonction mathématique pour réduire les artéfacts de compression [Skodras, 2000] [Skodras, 2001]. Les étapes suivantes sont réalisées séquentiellement pour compresser l'image dans le format JPEG2000:

Quantificateur - première partie. L'image est sectionnée en tuiles carrées de $N \times N$ pixels, de mêmes dimensions et sur sa pleine grandeur. Plusieurs options existent à ce niveau, dont celle de ne pas segmenter l'image. Ce sont ces tuiles qui forment la plus petite unité de reconstruction d'image. Elles peuvent avoir n'importe quelle dimension allant jusqu'à la taille complète de l'image.

Quantificateur - deuxième partie. Les tuiles sont indépendamment traitées par une transformée par ondelettes discrètes (DWT) sur deux dimensions. Si la transformée se doit d'être réversible, l'ondelette Le Gall 5/3 est employée. Dans le cas contraire, l'ondelette Daubechies 9/7 est utilisée. La transformée est couplée à un module de quantification scalaire uniforme avec zone morte pour former le quantificateur du JPEG2000. Ce sont les coefficients issus de la DWT qui sont quantifiés et envoyés au module de codage sans perte.

Codage sans perte - première partie. Les coefficients résultant du module de quantification sont encodés grâce à un encodeur arithmétique adaptatif. Cet encodage est binaire et est basé sur des conditions de contexte. Comme il s'agit d'un codeur entropique, l'encodage repose sur la probabilité de rencontrer ces conditions de contexte. Un symbole est alors produit en fonction de la condition rencontrée et de la probabilité d'occurrence de cette dernière. Ainsi, chaque tuile d'image quantifiée peut être représentée par un prisme ayant une surface égale à celle de la tuile et une profondeur égale au nombre de bits utilisés pour la résolution binaire utilisée. Il s'agit donc d'un cube binaire tel qu'illustré à la figure 1.3. Chaque plan binaire du cube est parcouru bit par bit avec l'encodeur arithmétique pour déterminer les conditions de contextes rencontrées. Le fait qu'une condition soit rencontrée ou non influence le symbole produit par l'encodeur. Le symbole prend la forme d'une trame binaire qui peut-être tronquée à plusieurs niveaux, permettant ainsi une plus grande granularité du PSNR en respectant un débit donné. Le codeur arithmétique adaptatif effectue plusieurs itérations d'encodage sur un même cube binaire.

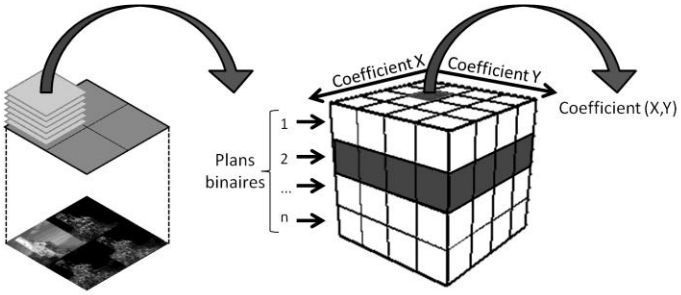

Figure 1.3 - Cube binaire

Légende : Une tuile de l'image à la fois est traitée par une DWT. Les coefficients résultant de cette transformée sont quantifiés. La figure 1.3 illustre une tuile d'image traitée par DWT, dont le premier cadran (la composante basse fréquence - LL) est segmentée en plans binaires. Le premier plan binaire au bas de la pile est constitué de l'ensemble des bits les moins significatifs de chaque coefficient quantifié. Le dernier plan au haut de la pile est formé des bits les plus significatifs de chaque coefficient. On peut donc représenter chaque cadran d'une DWT par un cube binaire. Ce dernier est codé par un encodeur arithmétique adaptatif, plan par plan et coefficient par coefficient. Les indexes X et Y servent à situer un coefficient en particulier.

Codage sans perte - deuxième partie. La dernière étape du JPEG2000 consiste à organiser la trame binaire encodée de façon à optimiser le PSNR en fonction du débit dans le but d'effectuer un décodage multi-résolution.

En effectuant l'analyse des opérations mentionnées, il est possible d'anticiper les délais relatifs au traitement, de même que l'utilisation de mémoire requise pour effectuer le traitement. La transformée DWT nécessite au maximum un délai d'une image car la taille des tuiles peut s'étendre jusqu'à la dimension de l'image complète. Or, comme la vidéo se transmet ligne par ligne, de la gauche vers la droite, il est nécessaire d'attendre jusqu'à la dernière ligne transmise avant d'effectuer une transformée sur deux dimensions. En fonction de la cadence de transmission des images, le JPEG2000 occasionne ici un premier délai de 33 ms si la taille des tuiles correspond à la taille des images. Il est donc nécessaire de fournir l'espace mémoire requis pour l'accumulation de la première image, soit environ 2,1 Mb dans le présent contexte de traitement d'image. Le deuxième délai par importance provient du codeur arithmétique. Chaque plan binaire d'une tuile peut être recalculé jusqu'à trois fois par ce dernier. Cela prend du temps, et requiert de la mémoire. La figure 1.4 résume les étapes importantes du JPEG2000 et approxime les délais causés par les étapes clefs.

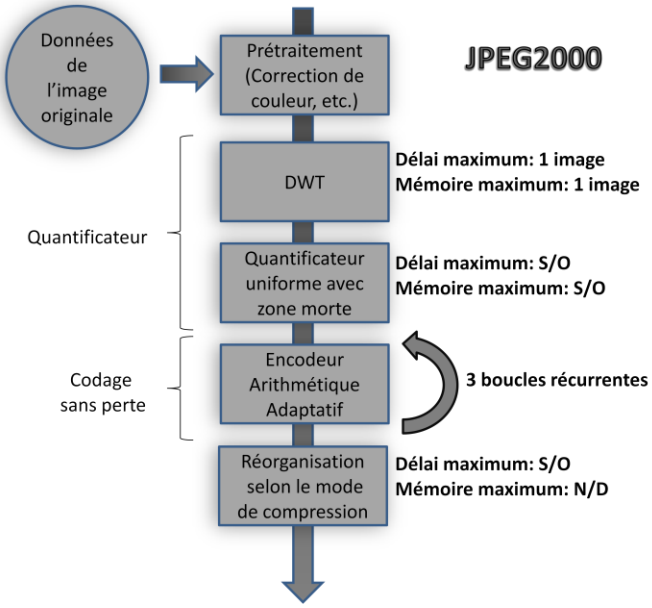

Figure 1.4 - Traitement d'image réalisé par le JPEG2000

De part le fait que le CODEC JPEG2000 permet la segmentation en tuile de taille égale à l'image, force est de constater que les délais d'encodage et de décodage combinés peuvent atteindre deux images, soit 66 ms de délais initial. Ce délai ne compte pas le temps pris par le CODEC pour effectuer le traitement. La taille standard des tuiles utilisée par plusieurs firmes détentrices de propriété intellectuelle est toutefois de 32 × 32 pixels [Marcellin, 2000].

Il est possible d'avoir une bonne idée des performances de ce CODEC en consultant le site de certains détenteurs de propriété intellectuelle œuvrant dans le domaine de la vidéo. Intopix fournit une fiche technique pour son codeur JPEG2000 développé pour un FPGA Virtex-5 ™ de la compagnie Xilinx [Intopix, 2011]. Intopix certifie que son implémentation occasionne un délai de deux images à l'encodage et une image au décodage. Il est donc question d'un délai total de 99 ms, ce qui dépasse largement les tolérances exigées par une application de téléassistance en salle d'urgence. En revanche, la vitesse de traitement d'une telle implémentation de CODEC est de 250 Mb/s, ce qui est plus que suffisant pour la quantité de données exigées.

1.2.2 Analyse du JPEG

Le JPEG est le résultat des travaux conjoints du CCITT et de l'ITU-T, qui débutèrent en 1986. Ce standard de compression est formellement connu sous l'appellation ISO/IEC_JTC1/SC29/WG10. Les normes régissant ce standard sont respectivement ISO/IEC 10918-1 et CCITT T.81. Elles sont en vigueur depuis 1992. Les implémentations du JPEG sont nombreuses car il s'agit d'un standard bien connu. Dans le présent document, le terme JPEG est associé au type JPEG fondamental développé initialement. La figure 1.2 permet de déduire qu'il s'agit du CODEC le plus petit des trois en termes d'étapes de traitement. Voici, dans l'ordre, les étapes prescrites par la norme T.81 [CCIT, 1993].

Quantificateur. L'image est reçue ligne par ligne et accumulée en tuile de 8 × 8 pixels. Une transformée DCT sur deux dimensions est appliquée sur chaque tuile, transposant ainsi les données dans un domaine fréquentiel. Il s'en suit une quantification matricielle appliquée directement sur les coefficients de la DCT. Ces deux opérations constituent le quantificateur du JPEG.

Codage sans perte. Une fois les données quantifiées, la matrice contient plusieurs 0 en apparence épars. Les données sont réorganisées de façon à maximiser l'occurrence de ces 0. Ces séries sont codées à l'aide d'un algorithme de *Run-Length Encoding* (RLE) modifié. Ces trames RLE sont encodées à leur tour via un codeur entropique de type Huffman. Ce codeur est en réalité une table de correspondance qui génère un symbole uniquement décodable en fonction d'un symbole RLE. Cette section forme ainsi le codage sans perte du JPEG. La figure 1.5 résume les principales étapes du JPEG ainsi qu'une approximation des délais causés par les modules principaux.

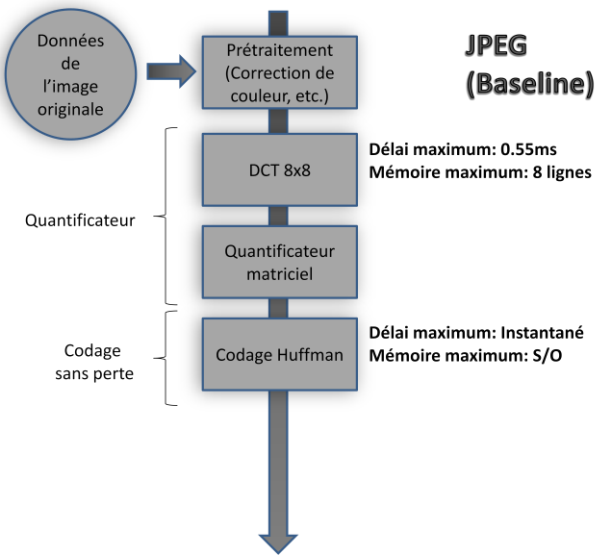

Figure 1.5 - Traitement d'image réalisé par le JPEG

Le premier élément marquant du JPEG est l'utilisation d'une faible quantité de mémoire. Ce dernier nécessite uniquement huit lignes de vidéo afin de commencer l'encodage. Il est donc question d'un délai initial de 1/60 d'image SIF, soit 0,55 ms. Il y a donc une marge de manœuvre suffisante pour rencontrer les contraintes de traitement en temps réel exigées par la téléassistance en salle d'urgence, soit 17 ms pour l'encodage et le décodage combinés. Plusieurs individus[5] ou entreprises[6] réalisent déjà des CODEC JPEG, mais ces derniers sont mal caractérisés ou très peu optimisés pour un contexte d'application médicale. Il y a donc place à amélioration pour atteindre la spécification minimale de débit du CODEC, soit 62,208 Mb/s. Le JPEG est donc prometteur d'un point de vue mémoire. Il faut toutefois optimiser le quantificateur afin que ce dernier introduise le moins de délai possible. Comme solution, l'encodage Huffman est une table de correspondance, qui s'exécute très rapidement.

[5] Il existe des implémentations JPEG pour FPGA disponibles sur le site www.opencores.org. Il s'agit de codes VHDL ou Verilog qui peuvent être employés sous licence *Lesser General Public License* (LGPL). Ce sont des contributeurs de opencores.org qui ont développés ces projets. Il y a 6 projets JPEG sur ce site seulement.
[6] À titre d'exemple seulement, Barco-Silex et CAST Inc. sont deux fournisseurs d'implémentations JPEG pour les FPGA de la compagnie Xilinx. Les fiches techniques de ces CODEC sont disponible sur le site www.xilinx.com.

1.2.3 Analyse du FFNN

FFNN est étroitement basé sur les travaux de [Ponomarenko, 2005] et [Ponomarenko, 2007]. Il se situe à mi-chemin entre le JPEG et le JPEG2000, à l'exception du codage entropique [Auclair Beaudry, 2009]. Il comporte un quantificateur reposant sur la DCT-2D comme le JPEG mais un étage de quantification uniforme qui s'apparente au JPEG2000. La quantification uniforme s'effectue sur un plan binaire comme le CODEC AGU [Ponomarenko, 2005] [Ponomarenko, 2007]. Le codage sans perte s'effectue à l'aide d'un codeur arithmétique binaire comme le JPEG2000 mais la différence réside dans l'estimation des probabilités d'occurrence des conditions de contexte. L'estimation repose sur un perceptron (neurone artificiel) ayant déjà convergé. Cela a pour avantage d'éliminer les itérations d'encodage requises pour déterminer la fréquence d'occurrence des conditions de contexte. Les principales fonctions de traitement sont résumées à la figure 1.6.

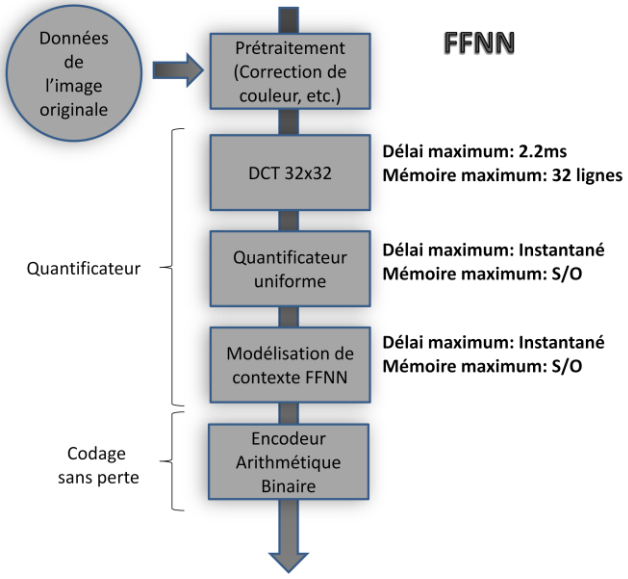

Figure 1.6 - Traitement d'image réalisé par le FFNN

Quantificateur. L'image est reçue ligne par ligne et formatée en tuiles de 32 × 32 pixels. De la même façon que le JPEG, une DCT-2D est appliquée sur chaque tuile. La quantification est réalisée sur les plans binaires. Elle s'effectue en deux temps. Il y a la simplification de coefficients isolés, c'est-à-dire en enlevant les coefficients qui n'ont que des 0 comme voisins et la suppression de plans binaires. Cette suppression engendre une perte de données, mais diminue aussi la quantité d'information à traiter par le codeur sans perte. On retrouve sommairement le même principe dans le CODEC JPEG2000 à l'exception qu'il est effectué lors du formatage des données de sortie. Le codeur sans perte du JPEG2000 doit donc traiter des données qui ne seront pas transmises. Le quantificateur du FFNN est en ce point plus optimal que celui du JPEG2000.

Codage sans perte. Le codage sans perte du FFNN est aussi optimisé. De la même façon que le JPEG2000, FFNN utilise la modélisation de contexte pour estimer la probabilité d'occurrence d'un bit. Or, cette modélisation de contexte est acheminée vers un perceptron qui se charge de traduire le contexte en probabilité d'occurrence. Cette dernière est utilisée par un codeur arithmétique. La modélisation de contexte et le perceptron agissent comme un prédicteur. Le codage sans perte est alors plus efficace que le codeur arithmétique adaptatif du JPEG2000 car il n'y a plus de récurrence.

Il n'existe aucune implémentation matérielle du FFNN. De plus, il n'existe aucune donnée sur la quantité de mémoire employée ou la précision requise pour les calculs intermédiaires du quantificateur. Il est sage d'anticiper qu'en raison des tuiles de 32 × 32, le délai initial de traitement du FFNN est de 32 lignes, et donc de 2,22 ms. Le débit maximal qu'il est possible d'anticiper dépend alors de la DCT-2D et du codeur arithmétique. Le perceptron procure un très faible délai de part son implémentation [Pedroni, 2004]. Comme la contrainte de traitement en temps réel impose un maximum de 17 ms pour la chaîne complète du CODEC, il reste un total de 7,5 ms pour effectuer l'encodage d'une image et l'équivalent pour la décoder.

1.2.4 Choix du quantificateur cible

À la lueur des comparaisons entre les différents CODEC candidats, il est évident que le JPEG2000, malgré sa puissante capacité d'encodage, ne permet pas d'atteindre les contraintes liées à l'application de téléassistance en salle d'urgence. Par conséquent, les seules alternatives susceptibles de déboucher sur une solution viable à court terme sont les CODEC FFNN et JPEG. Comme le cœur de la recherche vise l'optimisation des modules de quantification de ces CODEC, en raison du nombre élevé de manipulations mathématiques s'y retrouvant, l'objet de la recherche cible spécifiquement l'optimisation de la transformée DCT-2D sous une implémentation de technologie FPGA.

Afin de faire une analyse de performance réelle, les résultats de l'optimisation de la DCT-2D sont mis en application sous forme d'un CODEC JPEG en langage VHDL. Il est alors plus facile d'effectuer des comparatifs entre plusieurs méthodes d'optimisation lorsque ces dernières se trouvent en un contexte similaire à celui du problème soulevé dans l'introduction. Si les mesures d'optimisation s'avèrent adéquates et offrent des performances acceptables pour l'application de téléassistance en salle d'urgence, le système de téléassistance en salle d'urgence bénéficiera alors d'un CODEC opérationnel adapté à son contexte d'utilisation, mais aussi d'une première partie d'un CODEC potentiellement plus performant, le FFNN.

1.3 Organisation du document

Le chapitre 2 présente une revue de littérature sur la DCT et les méthodes d'optimisation de cette dernière. Les différents algorithmes concurrents sont présentés, de même que diverses architectures employées pour réaliser une DCT-2D sous une implémentation matérielle.

Le chapitre 3 traite de l'environnement de test et de la méthodologie employée pour effectuer les calculs de performance dans un environnement matériel tel qu'un FPGA. Il est question de l'architecture des bancs d'essai et des constructions réalisées pour mesurer la performance des algorithmes DCT-2D. Il est aussi question des méthodes d'analyse de performance, ou des

différentes métriques qui servent à mesurer la performance des architectures proposées pour l'optimisation.

Le chapitre 4 présente les résultats et les performances des implémentations DCT-2D mises à l'épreuve. Une discussion et une conclusion suivent immédiatement aux chapitres 5 et 6.

CHAPITRE 2
ALGORITHMES DES DCT-II RAPIDES

L'utilisation d'une transformée dans le traitement signal signifie qu'il faut décorréler un signal d'entrée, et donc diminuer l'information requise pour représenter ce dernier [Sayood. 2005]. Une transformée permet aussi de minimiser la représentation totale de l'entropie, et par conséquent, minimise la bande passante requise pour transmettre des données sur un canal de transmission. La notion de « compaction d'énergie » est employée ici pour illustrer la distribution de la variance d'un signal entrant dans les coefficients résultant d'une transformée. Plus la variance d'un signal traité par la transformée se trouve distribuée dans un petit nombre de coefficients du domaine de la transformée, plus la compaction d'énergie est importante et plus l'entropie sera minimisée. La transformée la plus optimale en termes de compaction d'énergie est la transformée de Karhunen-Loéve (KLT). Or, un inconvénient majeur de cette transformée est qu'elle requiert des informations statistiques sur le signal d'entrée pour effectuer la décorrélation. Les vecteurs de base sont choisis en fonction de la matrice des covariances du signal. Ainsi, si les informations statistiques du signal changent, la KLT doit être recalculée pour effectuer un traitement optimal [Britanak *et Al.,* 2007], [Sayood. 2005]. Comme les données statistiques d'un signal sont rarement disponibles au moment de la décorrélation, il est nécessaire de se tourner vers d'autres transformées. Parmi les plus populaires, il y a la transformée de Fourier (FT), la transformée de cosinus (CT), la version discrétisée de la FT (DFT), l'algorithme optimisé de la DFT (FFT) et la DCT.

La DCT est une transformée mathématique très utilisée dans le traitement d'image [Sayood. 2005]. Plusieurs raisons sont à l'origine de ce choix, notamment le fait qu'elle utilise une représentation réelle des nombres et qu'elle est non complexe, comme dans le cas de la FFT [Britanak *et Al.,* 2007]. Pour un ordre *N* élevé, elle est asymptotiquement équivalente à la KLT en terme de compaction d'énergie si, toutefois, le signal d'entrée peut être modélisé par une source de Markov ayant un grand coefficient de corrélation [Sayood. 2005]. La plupart des images naturelles peuvent être modélisées par une telle source car la valeur d'un pixel

comporte un grand niveau de corrélation avec les pixels voisins. L'avantage premier d'utiliser la DCT est qu'elle demeure, contrairement à la KLT, statique peu importe les informations statistiques du signal [Britanak *et Al.*, 2007], [Sayood. 2005]. C'est la raison pour laquelle la DCT est plus utilisée dans le traitement d'image, faute d'avoir accès aux données statistiques des images entrantes dans un CODEC. La figure 2.1 illustre un exemple de la différence entre le degré de compaction de l'énergie de l'image Lenna en employant la FFT et la DCT. Il est évident que l'énergie est davantage compactée en regardant le nuage de points brillants qui découle de la DCT.

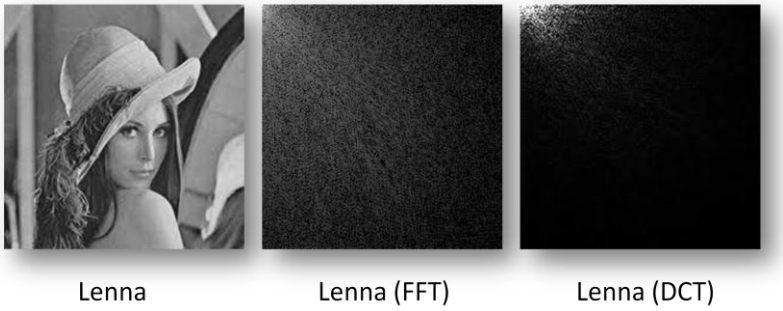

Figure 2.1 - Compaction de l'énergie de la photo Lenna en fonction de la FFT et DCT

Il existe au total une série de huit DCT, soit quatre de type pair et quatre de type impair. Comme les CODEC effectuent un traitement sur des blocs de dimension paire, la présente recherche se concentre donc sur les types pairs, soir la DCT-I, DCT-II, DCT-III et DCT-IV. Bien que chacune de ces transformée fasse partie de la famille des transformées sinusoïdales unitaires [Britanak *et Al.*, 2007], elles offrent des performances de compaction d'énergie différente les unes des autres. La transformée optimale en ce sens est la DCT-II. L'équation 2.1 régit les vecteurs de base de cette dernière. La variable *k* représente le numéro du vecteur de base (indice des lignes) et la variable *l* symbolise le numéro de l'échantillon du vecteur de base (indice des colonnes) à générer [Britanak *et Al.*, 2007], [Pillai, 2002a]. Il s'agit donc d'une matrice dont les lignes constituent les vecteurs de base. Le terme $\sqrt{\frac{2}{N}}\sigma_k$ est un facteur de normalisation servant à rendre la matrice unitaire. Finalement, *N* correspond à la dimension de la DCT-II à générer.

$$(C_N^{II})_{lk} = \sqrt{\frac{2}{N}}\sigma_k \cos\left[\left(l + \frac{1}{2}\right)\frac{k\pi}{N}\right], \qquad k, l = 0, 1, \ldots, N-1 \tag{2.1}$$

La DCT n'est pas une version discrète de la CT comme la DFT est la version discrète de la FT [Ali, 2003]. Elle constitue plutôt un ensemble de vecteurs de base représentant le mouvement oscillatoire discrétisé d'une masse et d'un ressort à diverses fréquences [Britanak *et Al.*, 2007]. Cette observation est une caractéristique propre à la famille des DCT et des transformées de sinus discrètes (DST). La figure 2.2 illustre les différents vecteurs de base et la notion de discrétisation de ces derniers.

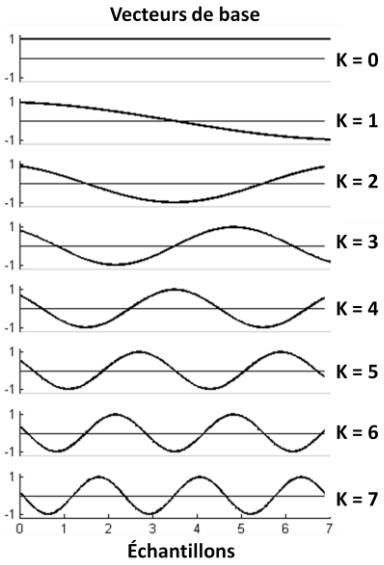

Figure 2.2 - Vecteurs de base de la DCT

En se référant à l'équation 2.1 et à la figure 2.2, il est possible d'en déduire une matrice $N \times N$ de la transformée qui, multipliée par un signal, fournit les coefficients de ce même signal dans le domaine de la transformée. Si cette matrice est appliquée sur les deux dimensions d'une image, soit une fois dans le sens des lignes et une fois sur le sens des colonnes, la version deux dimensions de cette transformée est obtenue. L'équation 2.2 illustre la matrice de la DCT-II et par le fait même, la version discrétisée de la figure 2.2.

$$C_8^{II} = \frac{1}{2}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & -\sin\frac{\pi}{16} \end{bmatrix} \quad (2.2)$$

Cette matrice ne peut pas être appliquée de façon optimale dans sa forme native. En plus de la précision requise pour réaliser les calculs, une matrice DCT-II de dimension N requiert N^2 multiplications et $N(N\text{-}1)$ additions pour un total de $2N^2$ opérations arithmétiques élémentaires. Il est donc nécessaire d'effectuer des simplifications, arrondissements et optimisations pour envisager une implémentation matérielle plus efficace de la DCT-II. La section 2.1 présente quelques alternatives en vue d'une simplification du calcul de la DCT-II.

2.1 Algorithmes FAST-DCT

Les algorithmes FAST-DCT constituent un ensemble de simplifications applicables à la matrice DCT-II. La représentation et l'étude de la transformée par l'algèbre vectoriel permet d'obtenir des équations plus rapides à calculer.

2.1.1 FAST-DCTII

Le premier algorithme de type FAST pour la DCT date de 1977 [Chen, 1977], [Britanak *et Al.*, 2007]. Ce dernier repose sur la factorisation récursive de la matrice des coefficients de la DCT. Il s'agit d'une première méthode pour effectuer la DCT-II de façon directe sans avoir recours à une forme de récursivité dans les calculs. La matrice des coefficients de la FAST-DCTII est obtenue à l'aide de multiplications de matrices de rotation, de portions des vecteurs de base de la DCT-IV et de matrices de permutation tel que présenté à l'équation 2.3.

$$C_N^{II} = B_N \begin{pmatrix} \hat{C}_{\frac{N}{2}}^{II} & 0 \\ 0 & \hat{C}_{\frac{N}{2}}^{IV} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix} \text{ où } J_{\frac{N}{2}} = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \tag{2.3}$$

B_N est une matrice de permutation permettant de réorganiser les coefficients en ordre binaire inversé, $\hat{C}_{\frac{N}{2}}^{II}$ est la matrice DCT-II de dimension $\frac{N}{2}$ ayant ses rangées en ordre binaire inversé, $\hat{C}_{\frac{N}{2}}^{IV}$ est la matrice DCT-IV de dimension $\frac{N}{2}$ ayant ses rangées et colonnes en ordre binaire inversé, et $J_{\frac{N}{2}}$ est une matrice identité renversée. L'ordre binaire inversé pour une matrice de permutation N = 8 est illustré à l'équation 2.4 de même que les manipulations binaires effectués sur les indexes d'une matrice identité pour obtenir B_N.

$$\begin{array}{l} 0\text{b}000 \Rightarrow 0\text{b}000 \\ 0\text{b}001 \Rightarrow 0\text{b}100 \\ 0\text{b}010 \Rightarrow 0\text{b}010 \\ 0\text{b}011 \Rightarrow 0\text{b}110 \\ 0\text{b}100 \Rightarrow 0\text{b}001 \\ 0\text{b}101 \Rightarrow 0\text{b}101 \\ 0\text{b}110 \Rightarrow 0\text{b}011 \\ 0\text{b}111 \Rightarrow 0\text{b}111 \end{array} \qquad B_N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Cette multiplication matricielle ne sert qu'à illustrer l'ensemble des manipulations mathématiques qui peuvent être parallélisées pour éliminer la récursivité. Il en résulte un

ensemble de structures papillon[7] imbriquées les unes dans les autres. Il s'agit d'une structure très efficace pour une future implémentation sur silicium. Les opérations mathématiques sont parallélisées et permettent de bénéficier des avantages d'une structure *pipeline*. La factorisation matricielle des coefficients de la DCT est à la base de tous les autres algorithmes de type FAST développés à la section 2.2. Bien que la parallélisation des manipulations mathématiques soit un résultat bénéfique de la structure développée par [Chen, 1977], cette dernière n'élimine pas les multiplications, de même que la représentation à virgule flottante ou virgule fixe nécessaire aux calculs. La quantité d'opérations mathématiques pour une DCT-II de taille *N* est toutefois réduite à [Britanak *et Al.*, 2007]:

$$2N\ \log_2 N = \left[\frac{N}{2}\log_2 N\right] \text{ multiplications} + [N \log_2 N] \text{ additions} \tag{2.5}$$

La figure 2.3 expose l'ordre dans lequel les calculs sont effectués. Les matrices de permutation sont en fait des structures papillons. Les matrices de rotation prennent la forme d'un papillon incorporant des changements de signe et des multiplications.

[7] Un papillon est une structure qui consiste à fractionner une transformée en plusieurs sous-transformée de façon récursive dans le but de paralléliser les manipulations mathématiques.

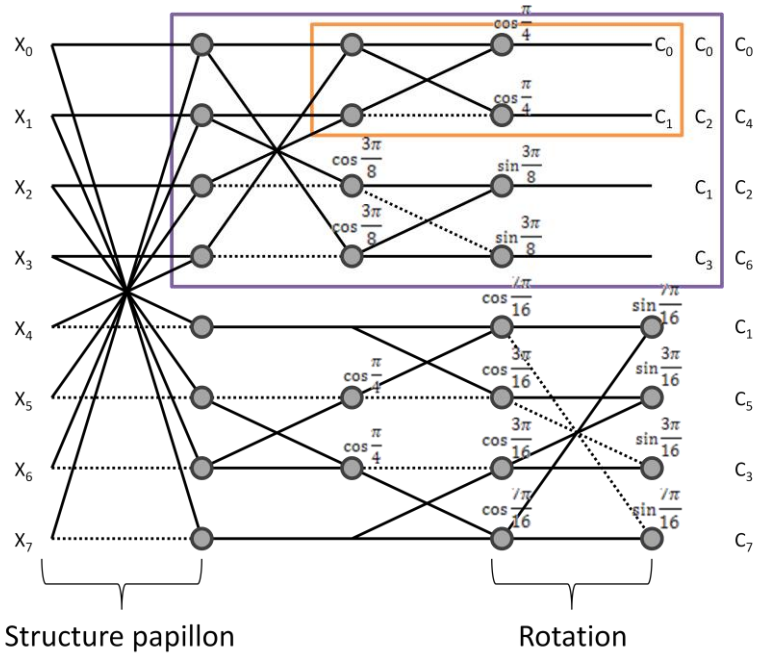

Figure 2.3 - Structure de l'algorithme FAST-DCTII

Légende : La structure présentée permet de traiter deux, quatre ou 8 pixels simultanément. Le plus petit encadré regroupe les opérations nécessaires pour une DCT-II en 2 points, l'encadré moyen permet une DCT-II en 4 points et la totalité de la figure permet une DCT-Ii sur 8 points. Les *Xi* représentent la valeur d'un pixel entrant et les *Ci* sont les coefficients sortant de la DCT-II. Les lignes pointillées représentent des soustractions et les lignes solides sont des additions. Cette structure est équivalente à l'équation 2.3.

2.1.2 DCT-II via WHT

La DCT-II est une transformée paire et dont la base est orthonormée. Cette propriété propre à plusieurs autres transformées permet de calculer les coefficients de la DCT en utilisant une autre transformée paire et de base orthonormée plus simple. Il est donc possible de calculer les coefficients de la DCT-II en utilisant la transformée de Walsh-Hadamard (WHT) $\hat{W}_N$ et une matrice de transformation intermédiaire T_N. L'accent circonflexe signifie qu'une matrice ou un vecteur est en ordre binaire inverse. L'équation 2.6 exprime la manipulation matricielle à réaliser pour obtenir les coefficients $\hat{c}^{II}$ de la DCT-II ou les coefficients $\hat{w}$ de la WHT à partir d'une transformée $\hat{C}_N^{II}$ ou $\hat{W}_N$ et d'un vecteur entrant transposé x^T. Comme la matrice $\hat{W}_N$ (WHT) est une matrice orthogonale, $\hat{W}_N^T \hat{W}_N$ est alors une matrice identité *I*. La substitution est donc possible comme l'expose l'équation 2.7. L'équation 2.8 démontre donc qu'il est possible,

par substitution de 2.6 dans 2.7, d'obtenir une matrice de transformation T_N . Cette matrice de transformation permet d'arriver aux coefficients DCT-II si elle est multipliée par les coefficients transposés $\hat{w}^T$ issus d'une WHT du signal x^T, tel que démontré dans l'équation 2.6. Les vecteurs de base de la WHT consistent en une série de 1 ou -1 tel que l'équation 2.9. Les manipulations mathématiques ne consistent alors qu'en de simples additions pour cette section. Les avantages de cette structure résident dans une réduction des matrices de rotation et des permutations binaires [Britanak *et Al.*, 2007].

$$\hat{c}^{II} = \hat{C}_N^{II} x^T, \qquad \hat{w} = \hat{W}_N x^T \tag{2.6}$$

$$\hat{c}^{II} = \hat{C}_N^{II} \hat{W}_N^T \hat{W}_N x^T = T_N \hat{w}^T \tag{2.7}$$

$$T_N = \hat{C}_N^{II} \hat{W}_N^T \tag{2.8}$$

$$W_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix} \tag{2.9}$$

$$T_8 = \begin{pmatrix} 1 & 0 & & 0 & \\ 0 & 1 & & & \\ & & U_2 & 0 & \\ & 0 & & 0 & U_4 \end{pmatrix} \tag{2.10}$$

$$U_2 = \begin{pmatrix} cos\frac{\pi}{8} & sin\frac{\pi}{8} \\ -sin\frac{\pi}{8} & cos\frac{\pi}{8} \end{pmatrix} \tag{2.11}$$

$$U_4 = P_4 \begin{pmatrix} cos\frac{\pi}{16} & 0 & 0 & sin\frac{\pi}{16} \\ 0 & cos\frac{3\pi}{16} & sin\frac{3\pi}{16} & 0 \\ 0 & -sin\frac{3\pi}{16} & cos\frac{3\pi}{16} & 0 \\ -sin\frac{\pi}{16} & 0 & 0 & cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} U_2 & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & U_2 \end{pmatrix} P_4 \qquad (2.12)$$

L'avantage premier d'une telle structure est qu'elle élimine deux permutations de coefficients en comparaison à FAST-DCTII. De plus, le nombre d'angles de rotation est réduit tout en fournissant une structure plus régulière. Les équations 2.10 à 2.12 exposent les détails de la matrice T_8 utilisée dans les équations 2.7 et 2.8 pour obtenir les coefficients de DCT-II. S'il était question de produire une DCT-II avec N = 16, la matrice T_{16} serait composée d'une matrice U_8, U_4 et U_2. On peut alors constater que la matrice de transformation T_N est une structure régulière composée de matrices de rotations U_N et de matrices de permutation P_N telles qu'illustrées à l'équation 2.12. La matrice de permutation P_4 est illustrée à l'équation 2.13.

$$\begin{matrix} 0b00 \Rightarrow 0b00 \\ 0b01 \Rightarrow 0b10 \\ 0b10 \Rightarrow 0b01 \\ 0b11 \Rightarrow 0b11 \end{matrix} \qquad P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.13)$$

2.1.3 DCT-II via Fourier

Le calcul de la DCT-II en utilisant la transformée de Fourrier se base sur le même concept que le calcul de la DCT-II impliquant la transformée de Walsh-Hadamard. Il n'y a aucun gain à employer cette méthode dans le cas d'une optimisation de la DCT. Elle présente une augmentation des multiplications et l'implémentation est moins directe qu'avec la transformée de Walsh-Hadamard [Britanak *et Al.*, 2007].

2.1.4 Split-Radix DCT-II

L'algorithme Split-Radix [Britanak *et Al.,* 2007] permet d'atteindre la limite inférieure du nombre de multiplications théoriques nécessaires pour réaliser une DCT-II. Il n'y a pas une grande différence entre le résultat de cette factorisation et l'algorithme FAST-DCTII présenté au début de la section. La différence réside en premier lieu sur l'emploi de matrices de rotations différentes et l'ordre dans lequel les manipulations sont réalisées. Un autre fait intéressant de cette factorisation est qu'elle présente un calcul de DCT-II à deux et quatre coefficients dans une DCT-II de huit coefficients. Tout comme FAST-DCTII, aucune simplification ou arrondissement n'est effectuée.

2.2 Algorithmes FAST sans multiplication

Tel qu'il a été mentionné dans les algorithmes FAST à la section 2.1, le problème lié à la rapidité d'une transformée dépend du nombre de multiplications nécessaires lors des calculs et de la parallélisation de ces derniers. Les algorithmes de type FAST règlent le problème de parallélisation, mais n'éliminent pas les multiplications. Il existe des algorithmes de type FAST sans multiplication [Jie, 2001] et il est primordial d'étudier ces algorithmes dans le but d'accélérer le traitement de la DCT-II.

La section 2.1 illustre que les principaux algorithmes FAST découlent de la factorisation matricielle éparse. Cette factorisation se traduit éventuellement par le produit de matrices papillon et de matrices de rotation telle qu'à l'équation 2.14. Les matrices papillon sont des matrices comportant des termes 1 ou -1. Il est difficile de les optimiser davantage. En revanche, il a été démontré que les matrices de rotation peuvent être factorisées en produit de matrices élémentaires Gauss-Jordan. Ces matrices élémentaires peuvent se décomposer à nouveau en produit de matrices triangulaires et diagonales en employant la factorisation PLUS [Britanak *et Al.,* 2007]. Ainsi, il est possible de décomposer une matrice de rotation en un produit de matrices triangulaires. Cela met en évidence que la DCT-II est soumise aux

simplifications de l'algèbre linéaire et vectoriel si elle est considérée comme une matrice. Ces simplifications débouchent sur des structures plus efficaces en termes de matériel. La figure 2.4 illustre l'équivalence graphique d'une matrice de rotation dont l'angle est de $\frac{3\pi}{8}$.

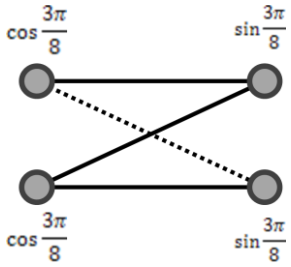

Figure 2.4 - Matrice de rotation $\frac{3\pi}{8}$

L'équivalence d'une telle structure sous forme matricielle est illustrée à l'équation 2.12.

$$G_{\frac{3\pi}{8}} = \begin{pmatrix} cos\frac{3\pi}{8} & sin\frac{3\pi}{8} \\ -sin\frac{3\pi}{8} & cos\frac{3\pi}{8} \end{pmatrix} \tag{2.14}$$

En employant la factorisation PLUS [Britanak *et Al.*, 2007]sur la matrice montrée à l'équation 2.12, une structure LUL est obtenue telle que démontré à l'équation 2.13.

$$G_{\frac{3\pi}{8}} = \begin{pmatrix} 1 & 0 \\ -\frac{1-cos\frac{3\pi}{8}}{sin\frac{3\pi}{8}} & 1 \end{pmatrix} \begin{pmatrix} 1 & sin\frac{3\pi}{8} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1-cos\frac{3\pi}{8}}{sin\frac{3\pi}{8}} & 1 \end{pmatrix} \tag{2.15}$$

Cette factorisation PLUS signifie tout simplement que la matrice de rotation est factorisée successivement en une matrice de permutation *P*, une matrice triangulaire inférieure *L*, une matrice triangulaire supérieure *U* et une matrice triangulaire inférieure ou supérieure *S*. Dans le cas spécifique de la matrice de rotation démontrée à l'équation 2.11, la matrice de permutation *P* est une matrice identité et la matrice S est une matrice triangulaire inférieure *L*. Une structure LUL est ainsi obtenue.

2.2.1 Structure LUL

Les structures LUL sont à la base des deux prochains algorithmes optimaux dont il est question dans cette recherche. La factorisation des matrices de rotation permet d'extraire une structure de type *lifting* des algorithmes FAST. Cette structure permet en soit une décomposition et reconstruction parfaite du signal car elle repose uniquement sur des manipulations de nombres entiers. Plus important encore, cette structure n'emploie que des additions et élimine toute multiplication.

Les termes des matrices de rotation peuvent être approximés par des nombres fractionnaires dyadiques, c'est-à-dire qui reposent sur une base de puissance deux. Ces nombres fractionnaires peuvent à leur tour être réduits sous la forme d'une somme minimale d'entiers. Un exemple est illustré à la figure 2.5. Il s'agit de la même structure présentée à l'équation 2.13. En multipliant les matrices par un vecteur d'entrée et en exposant graphiquement chaque manipulation mathématique, une structure de type *lifting* est obtenue [Le Dinh, 2009b].

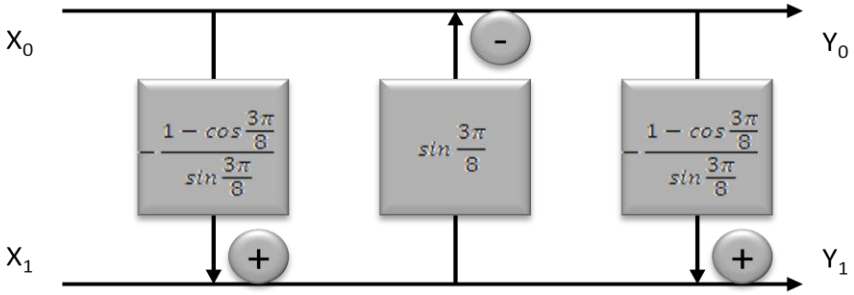

Figure 2.5 - Structure *lifting*

Cette structure prend tout son sens dans l'optique où les manipulations mathématiques sont approximées par une représentation entière minimale. La structure résultante est illustrée à la figure 2.6 et son inverse à la figure 2.7. Ce sont sur ces bases que reposent les algorithmes présentés aux points 2.2.2 et 2.2.3.

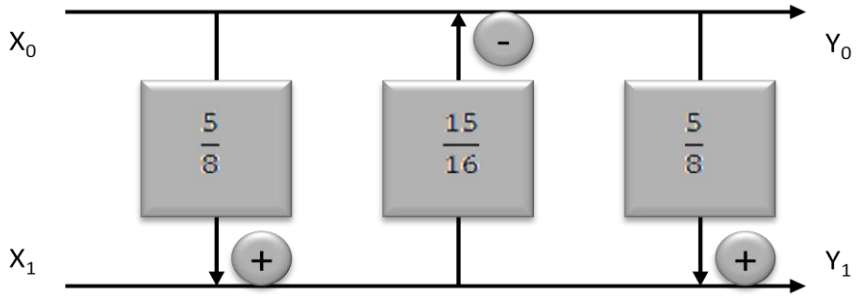

Figure 2.6 - Structure *lifting* approximée

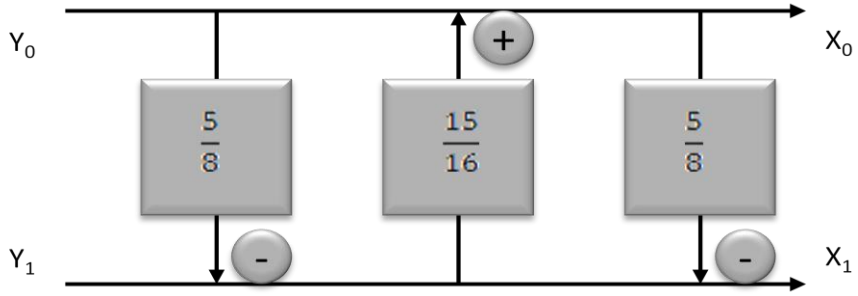

Figure 2.7 - Structure *lifting* approximée inverse

2.2.2 BinDCT-II

La BinDCT fait parti des algorithmes de DCT-II ne nécessitant pas de multiplicateurs [Tran, 2000]. En réalité, la multiplication s'effectue à l'aide d'entiers, et si possible à l'aide d'entiers issus de puissances dyadiques. Bien que la multiplication soit toujours présente, elle peut être réalisée de façon matérielle à l'aide de décalages binaires. Ces derniers ne requièrent aucune opération dans un circuit intégré tel un FPGA. Ces approximations des multiplications ont un effet négatif sur la performance de reconstruction des images. L'erreur quadratique moyenne (MSE) est plus élevée qu'avec la DCT-II réalisée à l'aide de multiplicateurs à point flottants. Il y a cependant plusieurs avantages propres à cet algorithme : la plage dynamique peut être directement calculée à partir de la sous-bande DC; la BinDCT hérite des mêmes avantages propres à la DCT-II conventionnelle; l'algorithme n'emploie que des entiers pour effectuer des opérations mathématiques. Cela signifie qu'il est possible d'atteindre des performances à toutes fins pratiques équivalentes à la DCT-II en utilisant une représentation des nombres

binaires plus restreinte [Britanak *et Al.*, 2007]. Il est important de noter que la BinDCT est issue de l'algorithme FAST-DCTII auquel la technique de factorisation LUL a été appliquée.

2.2.3 IntDCT-II

L'algorithme IntDCT-II est issu de l'algorithme DCT-II via WHT auquel la factorisation LUL a été appliquée. L'algorithme IntDCT-II comporte davantage d'additions que son homologue BinDCT. Or, ses performances sont supérieures en termes de compaction d'énergie et de MSE. Le degré de compaction d'énergie peut se mesurer à l'aide du gain de codage de la transformée. Ce gain de codage est exprimé à l'équation 2.16 et consiste au rapport de la moyenne arithmétique des variances des coefficients de la transformée sur la moyenne géométrique de ces mêmes coefficients. N correspond au nombre de coefficients et σ_i^2 correspond à la variance du $i^{\text{ème}}$ coefficient de la transformée. Le tableau 2-1 fournit un comparatif sommaire des performances entre les deux algorithmes à structure LUL. Malgré une hausse substantielle des opérations mathématiques, la IntDCT-II est un meilleur choix d'implémentation pour ses performances. Les 28 rotations binaires exigées par l'algorithme sont transparentes dans un FPGA et les 54 additions, une fois parallélisées, ont un faible impact sur la vitesse de l'algorithme [Britanak *et Al.*, 2007].

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{\frac{1}{N}}} \tag{2.16}$$

Tableau 2-1 Performances et ressources des trois DCT-II

Type de DCT-II	*Erreur quadratique moyenne*	*Gain de codage*	*Manipulations mathématiques*
DCT-II	-	8.82591	-
BinDCT-IIS	8.533458×10^{-5}	8.81855	41Add / 18 Rotations B.
IntDCT-II	5.467468×10^{-5}	8.82393	54Add/ 28 Rotations B.

2.3 Structure efficace pour une DCT-II 2D

La DCT est généralement employée sur deux dimensions en traitement d'image. Elle porte alors le nom de DCT-2D. Afin de représenter les vecteurs de base d'une telle transformée, il faut imaginer la fonction décrite à l'équation 2.1 transposée sur les deux dimensions d'une image. La figure 2.8 illustre ce concept.

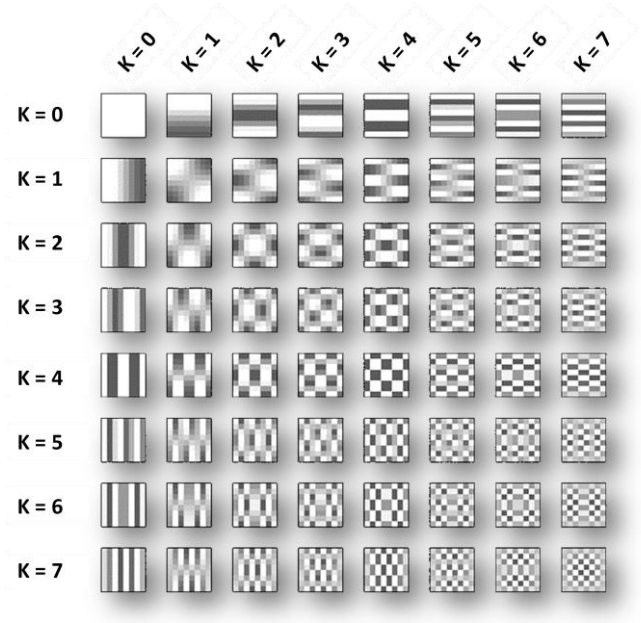

Figure 2.8 - Vecteurs de base de la DCT-II 2D

La fonction représentant les vecteurs de base est alors représentée par l'équation 2.17. Les variables k_1 et k_2 caractérisent respectivement les vecteurs de base sur les dimensions horizontale et verticale. Les termes n_1 et n_2 sont les numéros d'échantillon des vecteurs de base des dimensions horizontale et verticale. Le terme *N* représente l'ordre de la DCT-II 2D.

$$X_{k1,k2} = \sum_{n1=0}^{N_1-1} \sum_{n2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\left(n_1 + \frac{1}{2}\right)\frac{k_1\pi}{N_1}\right] \cos\left[\left(n_2 + \frac{1}{2}\right)\frac{k_2\pi}{N_2}\right], k = 0, \dots, N-1 \qquad (2.17)$$

Il existe beaucoup d'essais d'implémentation différents de DCT-II sous FPGA. Il est toutefois difficile de comparer de façon exhaustive les différences et les similitudes de ces derniers. Pour comparer adéquatement les implémentations, nous devons posséder des informations relatives à la technologie, au placement de la logique de même que le nombre exact de multiplicateurs, blocs additifs ou autres éléments susceptibles d'altérer les performances de l'algorithme implémenté. Ces informations sont rarement disponibles. Ainsi, la majorité des réalisations sous FPGA sont souvent inutilisables, tel le cas de [Gharge *et Al.,* 2007]. La comparaison se réalise alors au niveau des architectures employées pour réaliser les DCT-II. Il s'agit alors d'une comparaison arbitraire de concepts qui se base sur le nombre d'opérations à réaliser. Il n'existe toutefois aucune garantie d'atteindre les résultats présentés.

Ces architectures sont développées pour une application spécifique qui ne convient pas nécessairement au FFNN ou au JPEG. Les requis apportés par l'application de téléassistance en salle d'urgence consistent en la rapidité de traitement, un faible délai initial de traitement et une reconstruction fidèle des données. Il faut à tout prix diminuer les tampons présents entre les étages de la transformée et éviter d'avoir recours à des machines à états pour le contrôle de la DCT-II. Il faut donc éviter les architectures proposées par [Ruiz *et Al.*, 2005], les unités de contrôle qui ralentissent le *pipeline* [Gustavo, 2001] ou encore des méthodes d'encryption qui permettent une reconstruction sans erreur [Vassil *et Al.,* 2004]. Les DCT entières proposent déjà des structures à reconstruction parfaite ainsi que des architectures *lifting*. Il n'y a donc aucun gain à employer des méthodes d'encryption ou des unités de contrôle du *pipeline* pour cadencer ce dernier. Il est important d'avoir une reconstruction parfaite et une représentation compacte des coefficients pour la transmission sur canal. Or, si nous changeons la représentation des coefficients de DCT, cela affectera la compression RLE du JPEG qui suit éventuellement la quantification. Cela aura donc un effet néfaste sur les performances du codage entropique, ce qui se traduit par une augmentation du débit pour un même PSNR.

En ce qui a trait aux coefficients mêmes de la DCT-II, plusieurs auteurs tels que [Liang, 2001], [Trac, 2000] et [Britanak *et Al.,* 2007] proposent des coefficients susceptibles d'améliorer les performances de l'algorithme sous divers aspects. Comme la transformée est statique, il est possible d'utiliser une matrice de coefficients optimaux satisfaisants. La

combinaison de ces techniques nous permet donc d'obtenir un cœur DCT soit à une dimension ou à deux dimensions en regard à la propriété de séparabilité de la transformée [Britanak *et Al.*, 2007], [Syed, 2003]. Le changement de coefficients n'altère aucunement les performances des algorithmes en autant que ces derniers demeurent des entiers. Or, en ce qui a trait à la propriété de séparabilité illustrée à l'équation 2.18, il s'agit du noyau fondamental permettant de développer facilement un algorithme à deux dimensions à partir des algorithmes FAST entrevus dans la section 2.1 et d'un produit de matrice Kronecker. Ainsi, les sections 2.3.1 à 2.3.2 proposent deux architectures concurrentes possibles.

$$\begin{pmatrix} C_0^{II} \\ C_4^{II} \\ C_2^{II} \\ C_6^{II} \\ C_1^{II} \\ C_5^{II} \\ C_3^{II} \\ C_7^{II} \end{pmatrix} = (C_8^{II} \otimes C_8^{II}) \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} \tag{2.18}$$

2.3.1 Architecture 2D FAST

Comme il l'a été démontré dans les sections 1.1 et 1.2, il est primordial que la transformée DCT soit implémentée de façon à maximiser le débit, et donc diminuer le temps de traitement. L'architecture proposée par [Roman C *et Al.*, 2006] peut être considérée comme une avenue utilisable par une application de téléassistance en salle d'urgence. Selon l'expérience menée par les auteurs, un débit de 1719 Mb/s est atteignable avec la technologie Virtex2 Pro de Xilinx. La technologie préconisée dans le cadre de la présente recherche est la Virtex5 de Xilinx [Xilinx Inc., 2010]. Cette technologie est mature et est de loin plus performante que les technologies Virtex2 et Virtex4 sur les plans de la fréquence d'horloge et de la quantité de ressources disponibles. Un gain en vitesse de traitement est donc envisageable. La structure de [Roman C *et Al.*, 2006] permet d'accélérer le traitement de la DCT sur deux dimensions, et ce, sans tampon intermédiaire.

La stratégie à employer dans le cas de la DCT est donc d'utiliser un modèle IntDCT tel que proposé par [Raymond, 2006]. Or, l'implémentation finale s'effectue en utilisant l'architecture de [Roman C *et Al.*, 2006]. La figure 2.9 illustre une telle architecture.

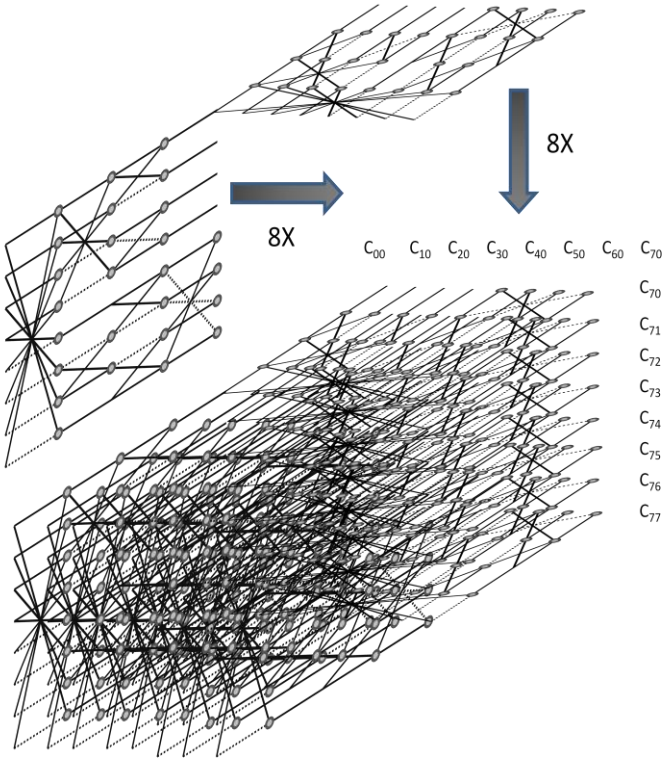

Figure 2.9 - Architecture 2D sans multiplicateur ou mémoire avec un algorithme FAST

Il est important de préciser que cette architecture traite 64 pixels simultanément. Seuls les coefficients en périphérie sont indiqués sur la figure 2.9. En réalité, un carré de 8 × 8 coefficients est produit pour chaque groupe de 8 × 8 pixels entrant. Les deux transformées illustrées en haut du prisme de DCT-II servent à illustrer l'agencement des 16 transformées qui constituent le prisme. À partir de ce point, un tel algorithme porte le nom de FAST Sans Multiplicateur DCT-II (FSM-DCTII).

2.3.2 Architecture 2D Xilinx

[Pillai, 2002a] proposent une architecture DCT-II en employant des unités de multiplication dédiées (DSP48E). Ces unités sont des accumulateurs configurables qui peuvent effectuer des multiplications dont le résultat ne dépasse pas 48 bits de résolution. Elles peuvent fonctionner à une fréquence maximum de 450 MHz dans le cas du Virtex-5 grade 3. Il est important de considérer une telle architecture puisqu'elle incorpore des ressources présentes dans le FPGA. Malheureusement, il est impensable d'employer une structure similaire [Roman C *et al.*, 2006] car cela impliquerait l'emploie de 112 multiplicateurs pour une DCT-II à deux dimensions et 240 multiplicateurs pour réaliser la transformée inverse [Pillai, 2002a]. En utilisant les propriétés matricielles de la DCT-II, il est possible de simplifier le nombre de multiplications en effectuant des permutations de lignes et de colonnes. La figure 2.10 expose ces manipulations.

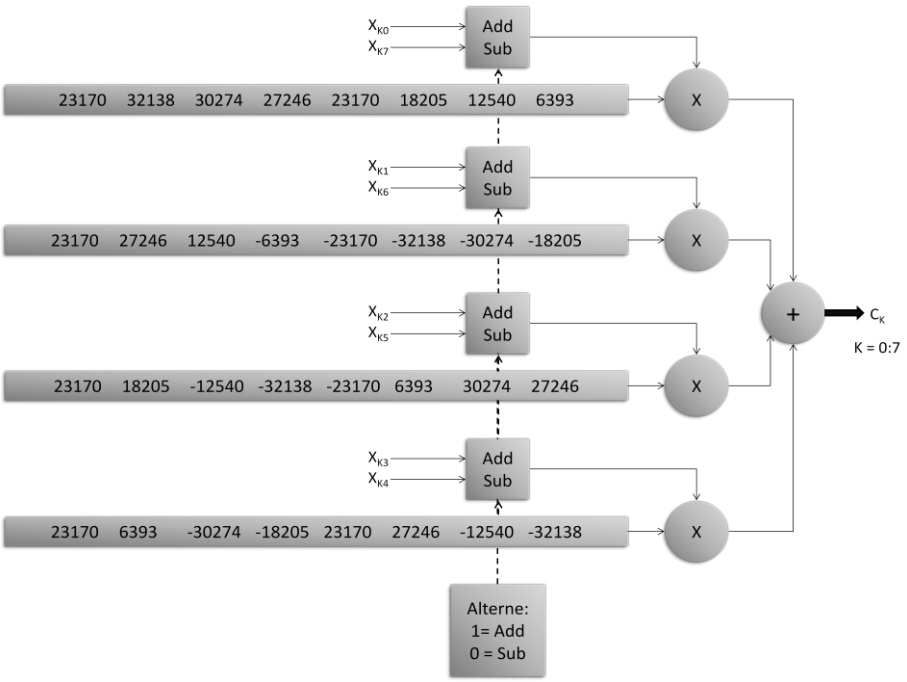

Figure 2.10 - Algorithme proposé par [Pillai, 2002a]

Légende : La structure présentée traite en entrée un vecteur X_i pour $i = 0$ à 7. Pour chaque indice pair des coefficients de sortie de la DCT-II (0 est considéré comme pair), il y a addition des coefficients d'entrée. Dans le cas des indices impairs, la soustraction est effectuée. Chaque résultat d'addition ou de soustraction (selon l'indice de sortie) est multiplié par le coefficient 16 bit du vecteur de coefficients DCT-II. Les 4 vecteurs présentés se lisent de la gauche vers la droite. Pour l'indice de sortie k = 0, le coefficient 16 bit sera 23170 pour tous les vecteurs de coefficients DCT-II. Chaque coefficient sera multiplié par l'addition d'une paire de X_i.

Les coefficients employés dans la figure 2.10 sont ceux de l'équation 2.2 ramenés sur une précision de 16 bits. Il faut lire les valeurs séquentiellement de la gauche vers la droite pour faire une correspondance directe avec l'équation 2.2.

Même s'il est évident qu'un tampon sera requis entre les deux étages de DCT-II, il est difficile de prévoir quelle sera la fréquence d'opération de l'architecture de [Pillai, 2002a] une fois cette dernière implantée dans le FPGA. Comme le traitement s'effectue par groupe de huit pixels au lieu de groupe de 64 pixels, il est raisonnable d'anticiper un traitement huit fois plus lent de cette architecture en comparaison avec l'architecture 2D FAST. Or, l'implémentation de [Roman C *et Al.,* 2006] est une DCT-II 2D de dimension 4 × 4. Une fois cet algorithme porté à une dimension de 8 × 8, rien ne garantit une fréquence d'opération aussi élevée dans le FPGA. Il est donc capital d'essayer l'architecture 2D de [Pillai, 2002a]. Cette architecture est appelée XIL-DCTII dans la suite du document et est exposée à la figure 2.11.

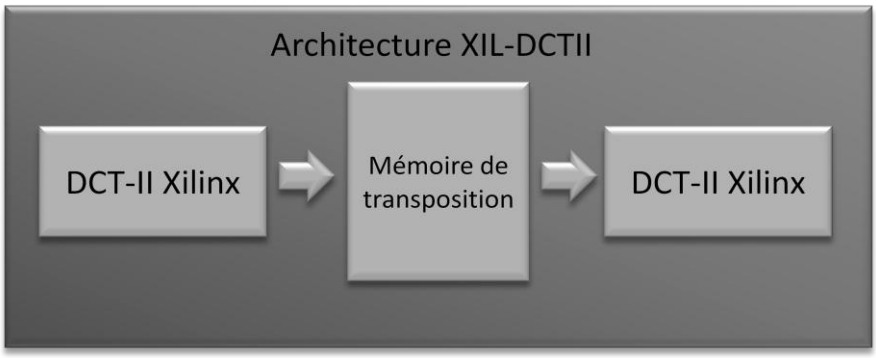

Figure 2.11 - Architecture proposée par [Pillai, 2002a] pour une DCT-II 2D

Légende : L'architecture XIL-DCTII est formée de deux éléments identiques de DCT-II présentés à la figure 2.10. La première DCT-II tel que proposée par [Pillai, 2002a] traite une image dans le sens des lignes. La seconde DCT-II traite les coefficients résultants dans le sens des colonnes. Or, comme la première DCT-II traite l'image entrante une ligne à la fois, il est nécessaire d'employer une mémoire de transposition pour récupérer les coefficients ligne par ligne et envoyer ces derniers colonne par colonne vers la seconde DCT-II.

CHAPITRE 3
MÉTHODOLOGIE ET ENVIRONNEMENT D'ESSAI

Le chapitre 2 indique qu'il est nécessaire de valider deux types d'architectures, soit la FSM-DCTII et la XIL-DCTII. FSM-DCTII repose sur l'algorithme FAST DCT-II via la transformée de Walsh-Hadamard, tandis que la XIL-DCTII repose sur une optimisation de la multiplication matricielle des vecteurs de base dans le cas spécifique de la DCT-II. La FSM-DCTII est entièrement expérimentale et constitue le cœur de la présente recherche. XIL-DCTII fait l'objet d'une note d'application (XAPP610) de la compagnie Xilinx Inc. [Pillai, 2002a] mais ne contient aucune spécification en matière de MSE, de résolution des manipulations mathématiques et de vitesse de traitement.

Ces deux algorithmes sont développés en VHDL pour une implémentation sous FPGA. Tel que décrit au chapitre 1, il y a de nombreux avantages à employer cette technologie. Il est nécessaire de paralléliser les opérations des algorithmes FAST si l'objectif de l'optimisation est la vitesse de traitement. Un processeur ou microcontrôleur exécute du code de façon séquentielle alors qu'un FPGA parallélise des structures logiques matérielles, soit des bascules, portes OU, portes ET, etc. Le parallélisme de ces structures débouche sur un autre avantage majeur : toute division par une puissance dyadique peut être représentée par un agencement de rotations binaires [Britanak *et Al.*, 2007]. Ces dernières peuvent être facilement contournées dans un FPGA, augmentant du même coup les performances par rapport à un processeur traditionnel.

Ce chapitre traite donc des méthodes employées, des calculs et analyses effectués dans le but de déterminer le niveau de performance des structures FSM-DCTII et XIL-DCTII dans un environnement VHDL et comparer ces derniers à une référence. Le choix de cette référence doit être justifié afin de donner la crédibilité nécessaire aux résultats qui servent essentiellement à situer les algorithmes développés par rapport à ce qui se fait actuellement. De plus, dans un soucis de transparence et pour permettre de tirer profit de l'expérimentation

menée par ce projet de recherche, il est primordial de définir le cadre d'essais des algorithmes DCT-II 2D, les outils mathématiques de performance employés et le point de référence.

3.1 Analyse de performance

L'analyse de performance doit s'effectuer à plusieurs niveaux. Il est question non seulement d'évaluer la qualité de la transformée par rapport à la référence, mais aussi d'évaluer la vitesse de traitement et le délai de traitement initial. Ces deux derniers éléments débouchent directement sur une spécification de débit en Mb/s et de latence en portion de champ d'image ou en ms. Il est aussi important de quantifier les ressources monopolisées par les deux structures en test. Il est possible d'atteindre de très bonnes performances de traitement avec des ressources matérielles illimitées. Or, une telle supposition viendrait dévaloriser l'objet de la recherche.

3.1.1 Qualité de reconstruction de la DCT-II

Il existe plusieurs méthodes pour évaluer la qualité de traitement de la DCT-II. En règle générale, le MSE de la transformée test est un indicateur abondamment employé comme métrique. L'équation 3.1 est la définition de l'erreur $\boldsymbol{e}$ entre la transformée de référence $\mathrm{U_N}$ et la transformée approximée $\hat{\mathrm{U}}_\mathrm{N}$. tel que le définit [Britanak *et Al.*, 2007]. Un signal **x** est appliqué à la différence des transformées. Comme le MSE peut être défini comme l'équation 3.2 où σ^2 est la variance d'une population et N est la taille de cette dernière, il est possible de définir le MSE en substituant la variance σ^2 par l'espérance E de la différence entre la valeur anticipée $\mathrm{U_N}$ de la transformée et la valeur obtenue par l'approximation $\hat{\mathrm{U}}_\mathrm{N}$ et pondérée par l'entrée **x**. L'équation 3.3 est donc obtenue en remplaçant l'équation 3.1 dans l'équation 3.2.

$$\boldsymbol{e} = \left(\mathrm{U_N} - \hat{\mathrm{U}}_\mathrm{N}\right)\mathbf{x} = \mathrm{D}\mathbf{x} \tag{3.1}$$

$$\mathrm{MSE} = \frac{\sigma^2}{N} = \frac{1}{N} E\left(\left(\left(\mathrm{U_N} - \hat{\mathrm{U}}_\mathrm{N}\right)\mathbf{x}\right)^2\right) \tag{3.2}$$

$$\in = \frac{1}{N} E[\mathbf{e}\mathbf{e}^{\mathrm{T}}] \tag{3.3}$$

Le calcul du MSE demande donc un signal et une référence comme facteur en entrée. S'il est envisagé qu'une DCT-II expérimentale soit comparée à une DCT-II de référence en termes de MSE, cela implique le choix d'un signal de référence de variance connue. Il y a là un problème ennuyeux car le type de signal d'entrée aura un impact direct sur la métrique que représente ici le MSE [Bi, 2003]. Dans le cadre de la recherche, il est nécessaire d'avoir une indépendance absolue entre le signal de test et les performances de la DCT-II de test. [Britanak *et Al.,* 2007] proposent une méthode pour calculer le MSE indépendamment du signal choisi. Les performances démontrées par [Britanak *et Al.,* 2007] dépendent d'un signal d'entrée que l'auteur ne définit pas, mais dont la matrice des covariances R_X est connue. [Britanak *et Al.,* 2007] calcule le MSE à partir de la différence entre les coefficients de la DCT-II et ceux de la DCT approximée tel que montré à l'équation 3.4. La trace constitue la sommation des éléments de la diagonale de la matrice entre accolades.

$$\in = \frac{1}{N} Trace\,\{\mathrm{D}R_X\mathrm{D}^{\mathrm{T}}\} \tag{3.4}$$

Malheureusement, il n'y a aucune information concernant la résolution binaire employée ou la précision des calculs avec lesquels la DCT-II référence est réalisée. Cette précision affecte le MSE surtout si la plage dynamique est importante.

Ce calcul de différence entre la DCT-II de référence et la transformée expérimentale n'est pas une méthode valable pour évaluer les performances de l'algorithme développé. Certains algorithmes permettent une reconstruction parfaite ou possèdent une capacité de décorrélation supérieure, mais présentent un MSE plus élevé lorsque comparés à la référence. C'est le cas des transformées effectuant des calculs sous base entière. Même s'il existe une légère différence entre les coefficients de ces transformées et la référence, elles permettent une reconstruction parfaite une fois couplée avec la transformée inverse correspondante. Il est donc évident qu'une telle transformée ne peut être évaluée seule en comparaison avec une référence. Il est préférable de l'évaluer conjointement avec la transformée inverse

correspondante et mesurer les performances de la paire. Une section portant sur la qualité de reconstruction traite de cet aspect à la page 49.

Même en adoptant une telle méthodologie de comparaison, le calcul de performance ne peut être isolé du signal employé lors des calculs. Il est donc recommandable d'effectuer une moyenne des résultats sur une gamme de signaux variés afin d'avoir une idée générale des écarts entre la DCT-II expérimentale et la référence. Les performances obtenues contiennent les erreurs propres à la transformée et à son inverse, ce qui est beaucoup plus significatif dans le cas du présent projet que d'établir le MSE sur la transformée simplement. Ainsi, au lieu d'employer seulement le MSE, les calculs de performances sont basés sur le PSNR calculé à partir d'une image où la transformée et son inverse sont appliqués séquentiellement. L'image résultante doit être de la même résolution binaire que l'image entrante pour ne pas fausser les résultats du PSNR. Ainsi, le MSE est calculé à partir d'une image sur laquelle une transformée de référence et son inverse sont appliqués, soit l'image résultante $I(i,j)$. Le même processus est réalisé avec la transformée test, dont l'image résultante est $K(i,j)$. L'image est de taille m pixels par n pixels. Le calcul du MSE est montré à l'équation 3.5. Le PSNR résultant est montré à l'équation 3.6. MAX_I^2 constitue le maximum de la plage dynamique de l'image élevé au carré. Dans le cas de l'environnement de test présent, il s'agit de la valeur 255^2.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{3.5}$$

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{3.6}$$

Il est donc question ici de qualité de reconstruction, et non pas nécessairement de qualité de traitement de la DCT-II. Le traitement de la DCT-II, même s'il se rapproche du traitement réalisé par une DCT-II de référence, n'implique pas nécessairement une reconstruction parfaite. Dans le contexte de traitement d'image mentionné au chapitre un, la chaîne formée par la transformée et son inverse fournissent une certaine qualité de reconstruction comparée à un chaîne de référence. La résolution binaire de l'image d'entrée est de huit bits, et telle est la

résolution de l'image de sortie. Les calculs de performance sont effectués à l'aide d'une quantité importante de signaux variés dans le but d'établir le niveau de performance moyen des nouveaux algorithmes développés. Nous obtenons alors un niveau de performance général, sans biais associé à un signal en particulier. Ce calcul de qualité de reconstruction s'effectue sur une DCT-II à deux dimensions, car il s'agit de la façon dont la DCT-II sera employée dans le contexte de traitement d'image dans le cadre de l'application de téléassistance en salle d'urgence.

3.1.2 Vitesse de traitement

La vitesse de traitement est un autre paramètre qui doit être tenu en compte dans l'analyse des performances de l'algorithme développé. Il débouche directement sur une valeur de débit lorsque le nombre de bits traités par coup d'horloge est connu. L'équation 3.7 permet d'obtenir le débit d'un algorithme, $v_{Traitement}$, en fonction de la fréquence d'opération et le nombre de pixels de huit bits traités.

$$v_{Traitement} = F_{opération} \cdot N_{Pixel} \cdot 8 \tag{3.7}$$

Une vitesse de traitement inférieure à 62,208 Mb/s rend le CODEC inutilisable dans le cadre de l'application de téléassistance en salle d'urgence. Cela fixe la limite inférieure de débit à laquelle doivent se soumettre les algorithmes expérimentaux. Le débit fournit par la DCT-II expérimentale se calcule une fois la mise en place de l'algorithme dans le FPGA. Ce sont les résultats du routage (PAR) qui indiquent la fréquence d'opération maximale de la structure instanciée. Cette vitesse de traitement prend aussi en considération le traitement sur deux dimensions.

3.1.3 Délai initial

Le délai initial de traitement est un autre facteur contraint par l'application de téléassistance en salle d'urgence. Il s'agit du délai requis par l'algorithme pour accumuler un nombre suffisant d'échantillons afin de pouvoir commencer un traitement. Ainsi, un algorithme peut fournir un débit très élevé, mais demander une accumulation de données trop importante pour être employée dans le cadre de l'application. Si cette accumulation de données est supérieure à 17 ms en temps, l'algorithme n'effectue pas un traitement en temps réel. Ce délai initial peut dépendre de plusieurs facteurs. Les principaux sont la longueur du chemin de traitement, la quantité de piles FIFO présentes sur le chemin et finalement l'architecture employée. Ainsi, une architecture nécessitant une mémoire intermédiaire de transposition, tel que l'algorithme XIL-DCTII, est plus susceptible d'engendrer un délai de traitement initial important en comparaison au FSM-DCTII dont les opérations mathématiques sont entièrement distribuées.

3.2 Environnement d'essai

La figure 3.1 présente une vue d'ensemble de l'environnement d'essai. Ce dernier est constitué de deux couches logicielles. La première regroupe un ensemble de scripts servant à la génération de vecteurs d'essai et de rapports de performance des algorithmes testés. Cette couche est implémentée sous MATLAB ® de la compagnie MatWorks ®. Un premier ensemble de fonctions effectue l'extraction d'image à partir de fichiers BMP et organise les trames de données pour l'envoi vers le CODEC. Un fichier d'échange en format ASCII est par la suite généré à l'aide des vecteurs de test et de données issus de la fonction d'extraction d'image. Ce fichier est transmis à la seconde couche logicielle codée en VHDL. Finalement, un autre fichier d'échange contenant les résultats de l'encodage est récupéré par un ensemble de fonctions responsables de la génération de rapport sous MATLAB ®.

La seconde couche logicielle est développée en langage VHDL et est employée sous MODELSIM ® de Mentor Graphics ®, un simulateur et compilateur VHDL. Afin que les deux couches logicielles communiquent ensemble, le fichier d'échange créé lors de l'extraction

d'image est interprété par un protocole de lecture implémenté en VHDL. La machine principale du banc d'essai interprète les valeurs ASCII pour adresser correctement les vecteurs de tests à l'unité expérimentale. Cette unité expérimentale est composée de l'encodeur et du décodeur d'images formant un CODEC qui est expliqué en détail aux sections 3.2.1 et 3.2.2. Les résultats sont retournés vers MATLAB ® via un autre fichier d'échange qui est interprété par l'ensemble de fonctions responsables de la génération de rapport. Les résultats et comparaisons sont alors produits.

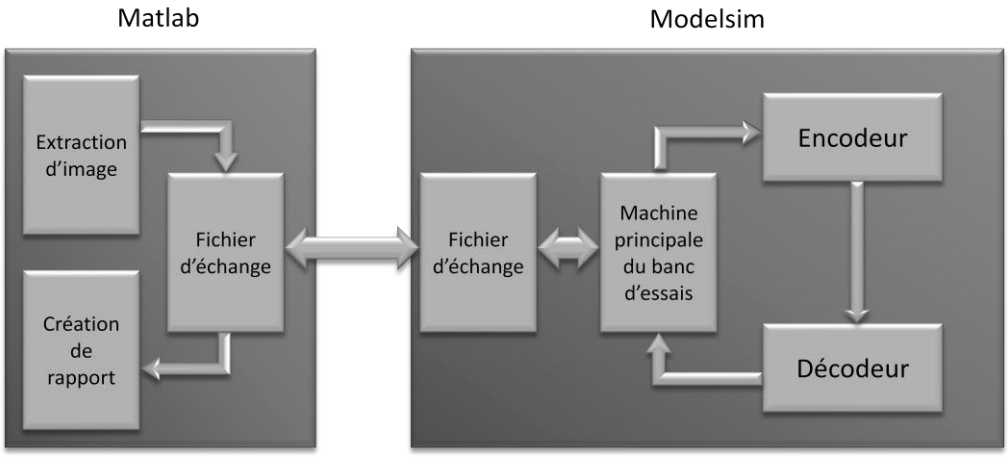

Figure 3.1 - Schéma de concept du banc d'essai

3.2.1 Banc d'essai VHDL des algorithmes

Les algorithmes DCT-II expérimentaux sont mis à l'épreuve dans un contexte de traitement d'image réel. Ils sont placés comme quantificateur et déquantificateur au centre d'un CODEC JPEG développé en VHDL. Ce CODEC émule le traitement réalisé par un véritable CODEC JPEG mais ne produit pas les entêtes de fichier JFIF propres au CODEC commercial [Hamilton, 2009]. Cette méthode permet non seulement d'entrevoir s'il y a une dégradation notoire du PSNR, mais aussi de constater s'il y a augmentation ou diminution du débit de l'image résultante. Si les algorithmes constituent une bonne approximation de la DCT-II de référence, il ne devrait pas y avoir de différences notables au niveau de l'encodage JPEG et sur le PSNR résultant du décodage.

Le banc d'essai sert avant tout de plateforme de mesure pour établir les performances des algorithmes développés en rapport avec les exigences d'une application de téléassistance en salle d'urgence. Quatre paramètres doivent être observés afin de spécifier correctement les algorithmes DCT-II 2D testés. Dans l'ordre, il y a le délai initial qui spécifie si l'algorithme peut être employé dans un CODEC en temps réel. Il y a la mesure de qualité de reconstruction servant à établir si les deux algorithmes sont équivalents en termes de traitement d'image. La vitesse de traitement est une métrique identifiant si l'algorithme maintient une vitesse de fonctionnement suffisante pour traiter le débit de données présenté au chapitre 1. Finalement, une mesure de ratio de compression est réalisée pour déterminer si les algorithmes évalués proposent un traitement identique lorsque employés dans un CODEC JPEG.

Délai initial. Le calcul du délai initial de traitement s'effectue dans le banc de test VHDL, à même le simulateur. Un fanion unique est envoyé dans l'encodeur JPEG et récupéré à la sortie du bloc DCT-II 2D expérimental. Il est donc possible d'obtenir un délai de traitement initial en termes de coups d'horloge pour chaque algorithme DCT-II 2D mis à l'épreuve. En connaissant la fréquence d'opération du module, il est possible de connaître le délai de traitement initial de l'architecture. La figure 3.2 illustre ce concept. Un train de pixels auquel un fanion unique est ajouté est transmis à l'encodeur. Au moment de l'insertion du fanion unique, un module de calcul de délai de traitement reçoit une consigne de départ. Le module de calcul du délai agit comme un chronomètre. Une fois que le fanion unique se trouve en sortie du module DCT-II 2D, le chronomètre s'arrête. La mesure de délais en coup d'horloge est alors connue.

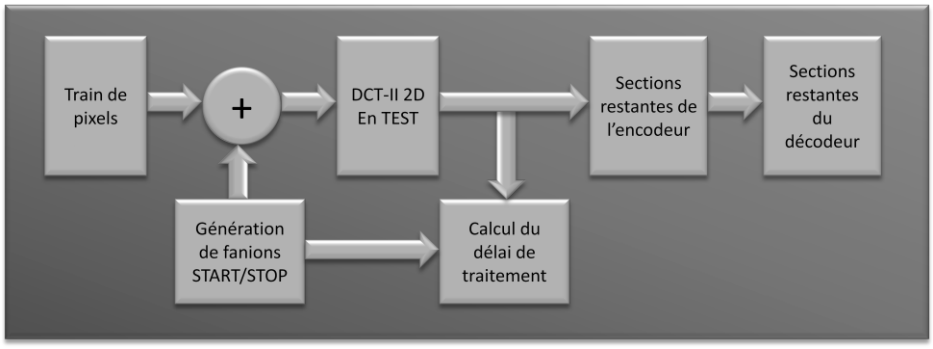

Figure 3.2 - Shéma de concept du calcul de délais initiaux

Qualité de reconstruction. Tel qu'il l'a été mentionné à la section 3.2.1, la méthode privilégiée pour évaluer la qualité de reconstruction de la DCT-II expérimentale est de comparer une séquence de traitements transformée/transformée inverse avec une référence. La référence employée dans le cadre du calcul de performance est la DCT-II de MATLAB ®. Cette dernière est calculée dans un environnement à point flottant et les résultats sont ramenés sur huit bits de précision. Cet arrondissement masque certaines différences potentielles entre la référence et la DCT-II expérimentale. Or, les données en provenance de l'image comportent une précision de huit bits. Il en est de même pour les données résultantes du décodage. Si l'erreur liée aux manipulations mathématiques est inférieure à la moitié du bit le moins significatif (LSB) de l'image résultante, elle n'aura aucun impact sur l'image de sortie. Si l'erreur apportée par la transformée expérimentale n'est pas perceptible pour l'utilisateur final en raison des arrondissements, elle effectue donc, à toute fin pratique, un traitement résultant similaire à la transformée de référence, dans le cadre présent travail de recherche. Or, seules les erreurs supérieures à un demi-LSB sont retenues et prises en compte dans le calcul du PSNR. La figure 3.3 démontre la méthode employée pour comparer les DCT-II 2D à la référence. La DCT-II 2D de référence et son inverse sont générées sous MATLAB ®. La DCT-II 2D expérimentale est entièrement codée en VHDL et exécutée sous MODELSIM ® . Les deux DCT-II 2D reçoivent les mêmes données provenant des fonctions d'extraction de MATLAB ®. Un fichier d'échange comprenant les résultats du banc d’essai de la DCT-II 2D expérimentale est généré par MODELSIM ® . Une fonction de comparaison sous MATLAB ® évalue l'algorithme de référence et l'algorithme expérimental.

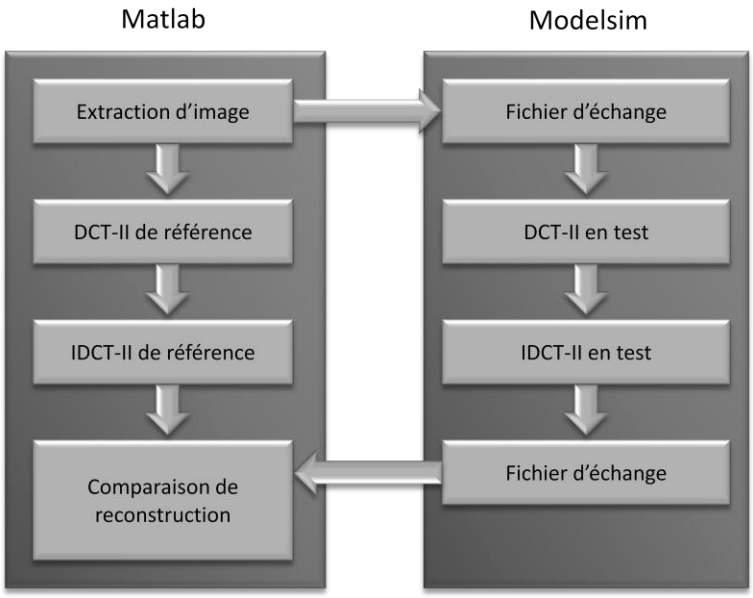

Figure 3.3 - Banc de comparaison de DCT-II 2D.

Débit - Vitesse de traitement. Le débit peut être anticipé en simulation, mais il ne sera validé qu'au moment du *Place And Route* (PAR), lors de la synthèse des modules dans le FPGA. Le PAR détermine les chemins électriques employés dans le FPGA, et donc la fréquence maximale à laquelle un module peut fonctionner. La méthode employée pour la validation de la vitesse de traitement consiste donc à compiler les fichiers sources en langage VHDL en utilisant l'outil de synthèse du manufacturier du FPGA. Dans le cas du présent projet, le FPGA employé est un Virtex-5™ SX50T de la compagnie Xilinx. L'outil de synthèse est inclus dans la suite logicielle *Integrated Software Environment* (ISE Design Suite 12.3™) de la même compagnie. Le compilateur fournit un bilan des ressources utilisées de même que la fréquence d'opération maximale du module. Cette fréquence, multipliée par le nombre d'échantillons traités par coup d'horloge, fournit une mesure de débit. La même procédure est répétée pour chacune des DCT expérimentales.

Débit - Bit par pixel. Le débit présenté dans le dernier paragraphe est associé à la vitesse de traitement de l'algorithme. Il existe toutefois une autre notion de débit associée à l'image résultante. Ce débit est calculé en bit/pixel (bpp) et définit la taille binaire associée à chaque pixel. Il s'agit en quelque sorte d'une extension du facteur de compression exprimée autrement. L'équation 3.8 illustre le calcul du débit. Ce dernier utilise la taille totale de l'image compressée $(Nbit)$ en bits et le nombre de pixels total $(Npixel)$ de l'image originale.

$$débit = \frac{Nbit}{Npixel} \tag{3.8}$$

3.2.2 Codec JPEG en VHDL

Comme le mentionne la section 3.3.1, l'expérimentation se déroule dans un cadre propre au JPEG. Ce cadre n'est pas absolument nécessaire dans l'optique où seul des comparatifs en termes de vitesse de traitement ou de délai initial de traitement d'une paire de transformées sont réalisés. Or, il est désirable non seulement de connaître les performances en terme de PSNR, mais aussi de voir comment ces performances affectent le débit de l'image résultante. Les algorithmes qui emploient une structure *lifting* permettent, en règle générale, une reconstruction parfaite du signal encodé. Or, cette reconstruction parfaite n'implique aucunement que les coefficients intermédiaires resteront identiques à la référence. Dans notre cas, cette déviation peut avoir un impact sur le JPEG. Comme le codage sans perte réside dans une table de correspondance de symboles[8], une déviation des coefficients de la DCT-II expérimentale peut engendrer une augmentation ou une diminution systématique du débit de l'image résultante. La présente section porte donc sur l'implémentation en VHDL du JPEG employé dans le cadre de l'expérience. Les étapes qui suivent constituent l'ensemble des manipulations couvertes par la norme T.81 et adaptées en VHDL. Seul l'encodeur est couvert ici. Le décodeur est une copie conforme des mêmes manipulations mais réalisées à l'inverse. Il est à noter que les points suivants n'exposent pas tout sur la compression JPEG, mais

[8] Encodage Huffman.

présentent une vue d'ensemble de chacune des transformations du JPEG associées à une structure VHDL pour la réalisation.

Transformée. Il s'agit de la première étape du JPEG. Une transformée DCT est appliquée en bloc de taille 8 × 8 sur les pixels entrant. Le scan vidéo s'effectue ligne par ligne. Pour appliquer la transformée, il est donc nécessaire d'accumuler au minimum huit lignes complètes dans un accumulateur. La transformée peut être remplacée par différentes implémentations (XIL-DCTII ou FSM-DCTII) servant à des fins expérimentales. Le bloc de transformée réalisé en VHDL est schématisé à la figure 3.4. Le train de pixels entrant dans le CODEC est accumulé au moyen d'un accumulateur de lignes. Cet accumulateur est en fait un bloc mémoire qui permettra de sortir les pixels de la vidéo en bloc de 8 × 8 pixels. Ces blocs sont ensuite traités par le bloc de DCT-II 2D. Cette DCT-II 2D peut être expérimentale ou une référence, tout dépendant de la nature des tests à réaliser. Les données sortant du bloc de DCT-II 2D sont dirigés vers le quantificateur qui est le module suivant. La plage dynamique des coefficients de sortie est de 12 bits. La totalité de la plage dynamique des valeurs intermédiaires est conservée pour les calculs.

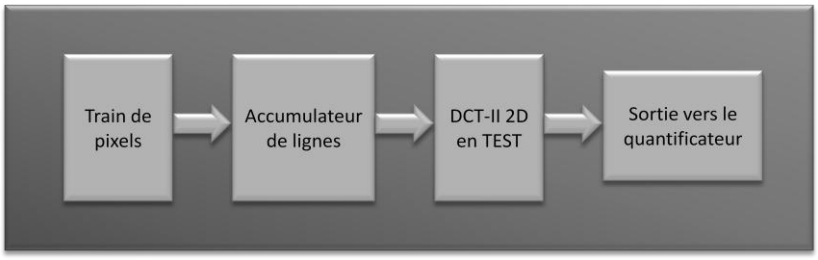

Figure 3.4 - Schéma du bloc DCT-II 2D du CODEC VHDL

Quantification. La quantification [Sayood. 2005] suit immédiatement le bloc de DCT-II 2D. Il s'agit d'une quantification de type matriciel. Chaque coefficient est numéroté séquentiellement de 1 à 64. À chacun de ces derniers est associé un pas de quantification. Dans l'exemple donné à la figure 3.5, le coefficient 1 de la DCT-II 2D se voit attribuer un pas de quantification de 16. Plus ce pas est petit, plus la granularité de la quantification est fine. À l'inverse, plus le pas est grand, plus la quantification est

grossière. En grossissant ou en diminuant le pas de quantification de certains coefficients, il est possible d'obtenir un compromis entre la qualité de l'image reconstruite et le débit de cette dernière.

Coefficient DCT-II 2D								Matrice de quantification de luminance							
1	2	3	4	5	6	7	8	16	11	10	16	24	40	51	61
9	10	11	12	13	14	15	16	12	12	14	19	26	58	60	55
17	18	19	20	21	22	23	24	14	13	16	24	40	57	69	56
25	26	27	28	29	30	31	32	14	17	22	29	51	87	80	62
33	34	35	36	37	38	39	40	18	22	37	56	68	109	103	77
41	42	43	44	45	46	47	48	24	35	55	64	81	104	113	92
49	50	51	52	53	54	55	56	49	64	78	87	103	121	120	101
57	58	59	60	61	62	63	64	72	92	95	98	112	100	103	99

Figure 3.5 - Matrice des pas de quantification

Légende : La figure associe l'index du coefficient de DCT-II 2D (Coefficient DCT-II 2D) à un pas de quantification (Matrice de quantification de luminance). Ainsi, si la valeur du coefficient 1 (indexe = 1) de la DCT se situe entre 0 et 15, la sortie quantifiée sera de valeur 0. Or, si ce même coefficient se situe entre 16 et 31, la sortie quantifiée aura la valeur 1.

La figure 3.6 illustre le schéma de concept du module de quantification matricielle en VHDL. La réalisation de celui-ci s'effectue à l'aide d'une mémoire dynamique (RAM) de sélection et d'une mémoire statique de quantification (ROM). La RAM possède une plage d'adressage de 0 à 63 correspondant à l'ensemble des index possibles des coefficients DCT-II 2D d'entrée (voir les indexes possibles à la figure 3.5). Chaque case de la RAM contient une adresse de sélection de la ROM. Cette adresse de sélection associe à un pas de quantification de la ROM, tel que présenté à la figure 3.5, à un index de coefficient DCT-II 2D d'entrée. Chaque case de la ROM contient un pas de quantification compris entre 0 et 255. Pour quantifier, il est nécessaire diviser les coefficients DCT-II 2D entrant par la valeur du pas de quantification associé à ce dernier. Le coefficient DCT-II 2D d'entrée est donc multiplié par l'inverse du pas de quantification qui est sélectionné par la case mémoire de la RAM. L'adresse de la case mémoire de la RAM correspond à l'index d'entrée de ce même coefficient DCT-II 2D. Les résultats sont envoyés vers le module suivant, soit le Zigzag. Une seule ROM de 64 adresses aurait été suffisante pour réaliser la sélection des pas de quantification. Cependant, l'ajout d'une RAM permet une plus grande flexibilité de sélection.

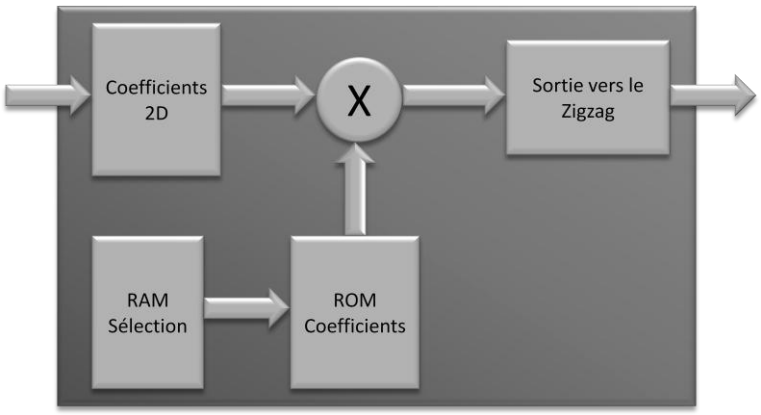

Figure 3.6 - Module de quantification matricielle

Réorganisation Zigzag. Tel que vu à la figure 3.5, plus l'index du coefficient est élevé, plus le pas de quantification est élevé. Ainsi, la probabilité d'occurrence de la valeur 0 augmente avec le numéro de coefficient de la DCT. L'idée de la réorganisation Zigzag est de créer de longues trames de 0. Ces dernières sont efficacement encodées par le codeur sans perte à la toute fin du CODEC JPEG. La figure 3.7 expose clairement l'ordre dans lequel les coefficients de la DCT-II 2D, une fois quantifiés, sont réorganisés. L'exemple illustre un bloc de 8 × 8 coefficients.

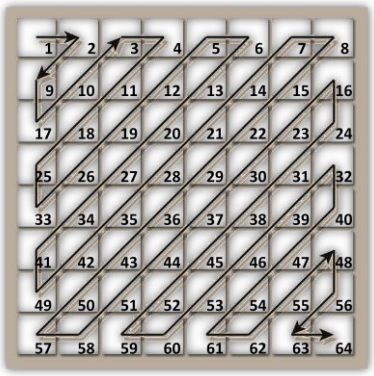

Figure 3.7 - Réorganisation Zigzag

Légende : La figure 3.7 démontre l'ordre dans lequel les coefficients quantifiés de la DCT-II 2D sont réorganisés. Dans l'ordre, les premiers coefficients à être transmis sont 1, 2, 9, 17, 10, 3, ..., etc. par rapport à l'ordre conventionnel qui aurait été 1, 2, 3, 4, 5 ,6 , ..., etc.

Il est nécessaire d'avoir un délai d'un bloc de 8 × 8 coefficients afin de pouvoir réorganiser convenablement ces derniers. Le module VHDL associé à la réorganisation Zigzag est présenté à la figure 3.8. Il emploie une mémoire à deux ports (DPRAM) pour l'accumulation des données. Un compteur d'adresse est synchronisé avec les données entrantes et sélectionne la plage adressable de la mémoire allant de 0 à 63. Une mémoire ROM se charge de sélectionner les bonnes données au bon moment en sortie. Il y a donc un délai minimum d'un bloc de 8 × 8 coefficients.

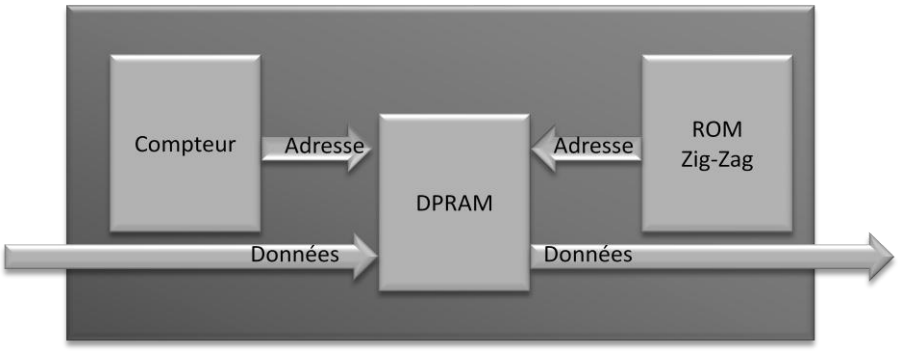

Figure 3.8 - Structure de la réorganisation Zigzag

Encodage RLE. L'encodage RLE [Sayood. 2005] n'est pas une version traditionnelle du RLE. Il cible précisément les séries de 0, dont la fréquence des occurrences est maximisée par la réorganisation Zigzag. Le symbole résultant de l'encodage se divise en trois parties. La première comporte quatre bits et symbolise le nombre de zéros précédant la prochaine valeur non nulle. La seconde est un groupe de quatre bits qui définit la dimension du coefficient encodé. La dernière constitue la valeur d'amplitude encodée du coefficient. La figure 3.9 illustre un exemple d'encodage RLE en fonction d'une trame reçue en provenance du module Zigzag. Il est à noter que la longueur des mots sortant du module d'encodage RLE est variable en raison de la valeur résiduelle des coefficients encodés.

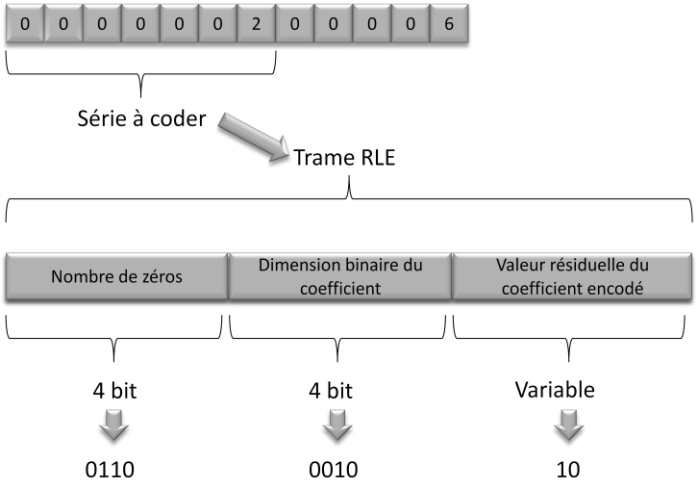

Figure 3.9 - Codage sans perte RLE

Légende : La figure 3.9 prend comme exemple une série de six coefficients DCT-II 2D quantifiés à zéros suivis d'un coefficient quantifié égal à deux. La trame RLE indique donc le nombre de zéros précédent la première valeur non nulle rencontrée. Cette valeur tient sur quatre bits et correspond à six dans l'exemple. La seconde information codée dans la trame RLE est la dimension binaire du coefficient. Cette information tient aussi sur quatre bits. Pour toutes les valeurs comprises entre -3 et 3 à l'exception de -1 et 1, cette dimension binaire est égale à deux. En effet, avec deux bits seulement, il est possible de représenter la série de valeurs suivantes : -3, -2, 2, 3. Dans l'exemple, le chiffre suivant la série de zéros est le deux. L'index de la série de valeur -3, -2, 2, 3 correspondant au chiffre deux est la valeur deux. La valeur -2, à titre d'exemple seulement, correspond à l'index 1. La valeur résiduelle de la trame RLE, quant à elle, correspond au complément à1 du coefficient non nulle à coder. Cette valeur résiduelle est de longueur variable.

L'exemple précédent implique un coefficient AC [CCIT, 1993], c'est-à-dire un coefficient différent du coefficient DC (0,0) de la DCT. Le cas spécifique du coefficient DC emploie une technique similaire d'encodage, à l'exception du fait qu'aucun 0 ne précède ce coefficient et que la valeur résiduelle est obtenue en soustrayant la valeur précédente du coefficient DC de la valeur actuelle. Cela revient donc à encoder la différence entre le bloc DCT actuel et le précédent. Ce bloc est réalisé en VHDL à l'aide d'une machine à états. Tel qu'expliqué au chapitre 1, nous cherchons à diminuer le nombre de traitements séquentiels. Il est malheureusement nécessaire d'en employer un ici.

Encodage Huffman. L'encodage Huffman [Sayood, 2005] est la deuxième étape du codage sans perte. Ce codage se fait à l'aide d'une table de correspondance et des deux premières parties du codage RLE. Un code unique et à décodage unique est généré à partir du nombre de 0 précédant le prochain mot binaire non nul et de la dimension binaire de ce dernier. La figure 3.9 illustre le code Huffman généré à partir de l'exemple de la figure 3.10. L'encodeur transmet donc sur le canal un code Huffman suivi des chiffres binaires formant la valeur résiduelle du coefficient encodé.

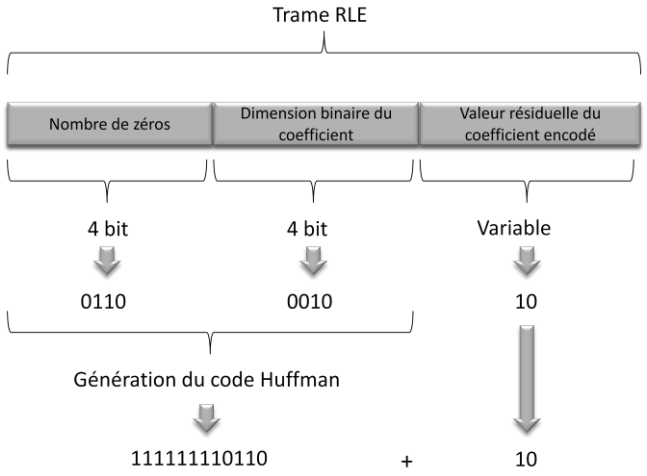

Figure 3.10 - Génération du code Huffman

Légende : La figure 3.10 reprend l'exemple de la figure 3.9. Les deux premières composantes de la trame RLE sont associées à un symbole Huffman prédéterminé dans une table. Le symbole dépend du nombre de zéros précédant une valeur non-nulle et la dimension binaire de ce premier coefficient non-nulle. La valeur résiduelle est simplement concaténée à la suite du code de Huffman.

Ce module emploie un multiplexeur codé en plusieurs niveaux afin de maximiser la vitesse d'opération. La figure 3.11 ne montre pas la concaténation de la valeur résiduelle du coefficient encodé mais illustre la sélection du code approprié en fonction du nombre de 0 et la dimension binaire de la trame RLE. Les codes de Huffman sont déjà emmagasinés dans une mémoire. Le sélecteur les aiguille au bon moment vers la sortie.

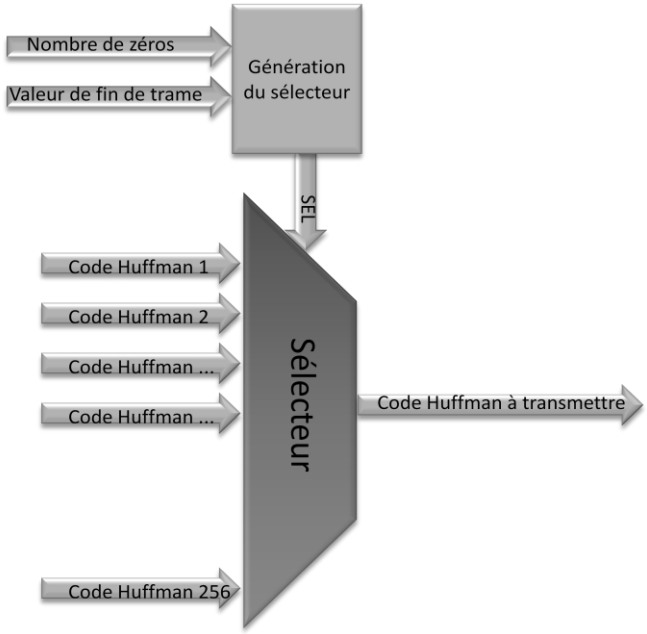

Figure 3.11 - Implémentation du codage Huffman en VHDL

Codec complet. L'annexe B décrit, en termes de fichiers, l'architecture du banc d'essai complet en VHDL. Chaque fichier renferme des fonctions et processus nécessaires à l'interface des modules du CODEC JPEG. L'interface MATLAB ® ainsi que les fonctions de cette dernière ne sont pas décrites ici. Afin d'avoir une vue d'ensemble de l'agencement des modules VHDL du CODEC, la figure 3.12 résume les interconnections décrites à la présente section. Le banc d'essai est responsable de la lecture des stimuli et de l'envoi des résultats via des fichiers d'échange. Il comprend un encodeur et un décodeur JPEG séparés par une pile *First-In-First-Out* (FIFO). Le Décodeur fonctionne à une vitesse plus élevée que l'encodeur pour éviter une congestion potentielle dans la pile FIFO.

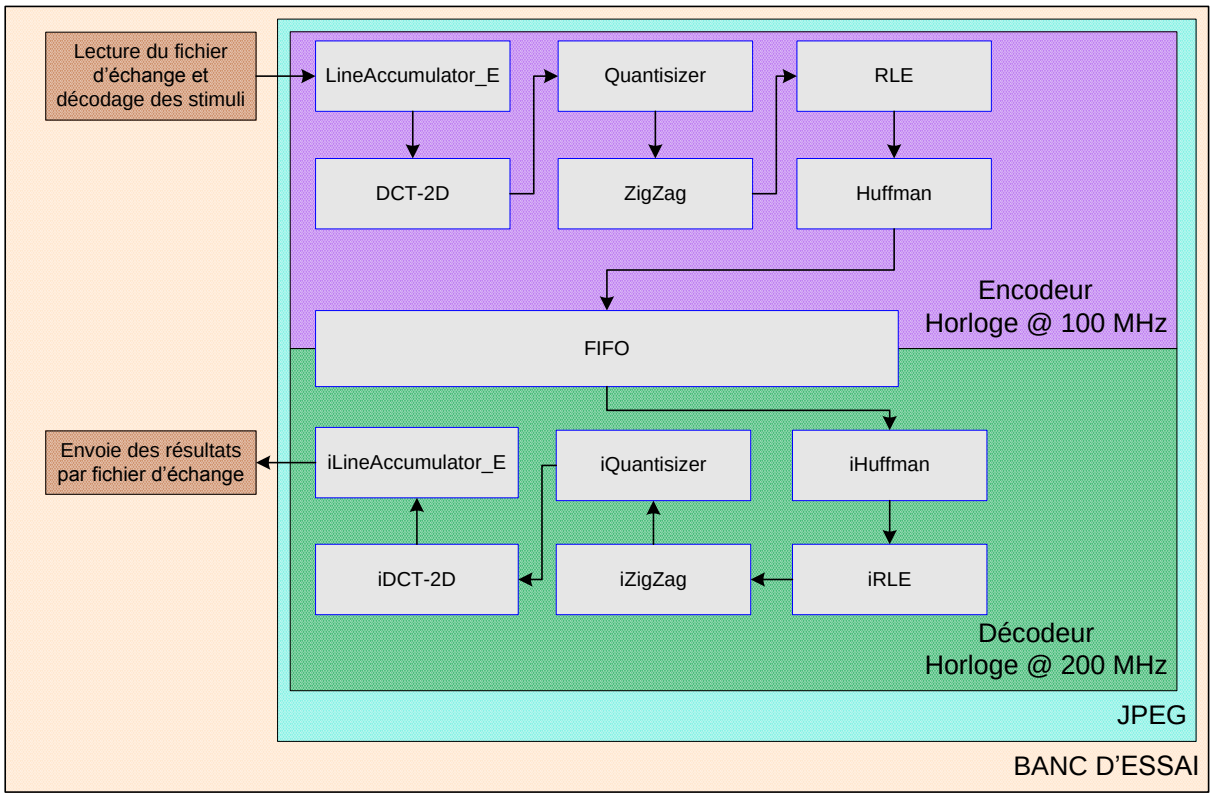

Figure 3.12 - Vue d'ensemble du banc d'essai VHDL

Le chapitre suivant expose les résultats en termes de vitesse de traitement, délais initiaux de traitement ainsi que des ressources matérielles employées par chacun des deux algorithmes mis à l'essais dans le cadre du présent projet de recherche. Il est aussi question des performances de ces algorithmes en terme de reconstruction d'image lorsque ces derniers sont employés dans le cadre d'un CODEC JPEG. Le CODEC JPEG et l'environnement d'essai utilisés correspondent à la figure 3.12

CHAPITRE 4
RÉSULTATS

Ce chapitre expose les résultats de comparaison entre les deux algorithmes proposés au chapitre 2, soit XIL-DCTII et FSM-DCTII. Ces derniers sont comparés en fonction de la démarche proposée au chapitre 3. En premier lieu, il est question des performances en termes de vitesse de traitement. Ensuite, les performances en termes de délais initiaux sont mises en relief. Par la suite, les performances de reconstructions sont présentées lors de la comparaison des CODEC JPEG incorporant les deux différentes alternatives vues au chapitre 2 ainsi que la DCT-II 2D de référence. Une section supplémentaire traite des ressources matérielles utilisées à la réalisation des deux algorithmes. En dernier lieu, les résultats d'implémentation sous JPEG matériel sont exposés.

4.1 Vitesse de traitement

La vitesse de traitement des deux algorithmes se traduit par une spécification de chacun en termes de débit en Mb/s. Le tableau 4-1 illustre la vitesse de ces derniers, une fois leur compilation réalisée au sein d'ISE 12.3. Cette vitesse doit être multipliée par le nombre de bits traités à chaque coup d'horloge. Nous savons, de par leur architecture, que XIL-DCTII et FSM-DCTII traitent respectivement 8 pixels et 64 pixels simultanément. Ce nombre de pixels par coup d'horloge doit être multiplié par une résolution de 8 bits. Il est donc évident que le débit de FSM-DCTII dépasse largement celui du XIL-DCTII malgré que cette dernière utilise une fréquence d'horloge environ 1.5 fois plus élevée. Le tableau 4-2 illustre les performances en termes de débit de chacun des algorithmes et présente aussi le seuil d'opération minimal exigé dans le cadre du projet de l'application de téléassistance en salle d'urgence. FSM-DCTII est donc supérieur à XIL-DCTII en termes de vitesse de traitement malgré une vitesse d'opération moins élevée. Les deux algorithmes sont toutefois suffisamment performants dans le cadre du projet de l'application de téléassistance en salle d'urgence.

Tableau 4-1 Fréquence d'opération des algorithmes développés sous Virtex-5™ SX50T

Algorithme	Fréquence d'opération de la structure développée (MHz)
XIL-DCTII	360,4
FSM-DCTII	200,7

Tableau 4-2 Débit des structures développées sous Virtex-5™ SX50T

Algorithme	Débit maximal de la structure développée (Gb/s)
Seuil visé	0,062
XIL-DCTII	2,883
FSM-DCTII	12,842

4.2 Délai initial de traitement

L'évaluation du délai de traitement est en partie expérimentale et en partie théorique. Il est nécessaire d'avoir la fréquence d'opération des architectures mais aussi de connaître le délai en termes de coups d'horloge entre l'entrée du premier pixel et la sortie du premier coefficient. Ce délai peut-être calculé à partir de la simulation sous MODELSIM ® .

De par l'architecture des deux algorithmes en compétition, il est possible d'anticiper que XIL-DCTII aura un délai de traitement plus élevé que FSM-DCTII en raison de la mémoire de transposition qui force une accumulation de 64 pixels au minimum. Or, de par la vitesse de traitement une fois implanté sous FPGA, il se pourrait que l'algorithme FSM-DCTII démontre un délai de traitement plus important en raison du nombre élevé d'étages de traitement. Le tableau 4-3 montre le délai de traitement en coups d'horloge pour chacun des algorithmes proposés. En reprenant les résultats présentés au tableau 4-1 et en les combinant aux résultats du tableau 4-3, le délai initial en ns est obtenu au tableau 4-4. À la lueur des résultats présentés dans les tableaux, il est possible d'en conclure que FSM-DCTII est plus rapide que XIL-DCTII.

Tableau 4-3 Délais initiaux de coups d'horloge sous Virtex-5™ SX50T

Algorithme	Délais initiaux de traitement (coups d'horloge)
XIL-DCTII	94
FSM-DCTII	16

Tableau 4-4 Délais initiaux de traitement sous Virtex-5™ SX50T

Algorithme	Délais initiaux de traitement (ns)
XIL-DCTII	260,8
FSM-DCTII	79,7

4.3 Mise en place avec JPEG

Cette section présente les résultats en termes de PSNR des différents CODEC entrevus dans l'introduction. Des images de références sont compressées et décompressées en employant le JPEG2000, FFNN et différentes versions du CODEC JPEG. Les CODEC JPEG n'ont qu'un seul élément qui les distingue, soit la DCT-II 2D. Les différentes versions du JPEG sont JPEG standard (JPEG_STD), JPEG Point Fixe avec Multiplicateur (JPEG_FX) et JPEG avec manipulations entières (JPEG_INT).

4.3.1 JPEG_STD

JPEG_STD est une version du JPEG telle qu'implémentée sous MATLAB ®. Les calculs sont réalisés en point flottant en suivant une méthode de quantification associée à l'*Independent JPEG Group* (IJG). L'architecture générale demeure la même que celle proposée par les comités ISO/IEC, mais la matrice de quantification est basée sur un facteur appelé qualité $Q_{facteur}$. Ce facteur est d'ailleurs employé par la vaste majorité des implémentations JPEG, mais la façon de le calculer est propre aux développeurs du CODEC. La méthode de calcul employée par IJG est aussi utilisée par *Microsoft Paint*, *The Gimp*, *Infranview* et certaines caméras présentes sur le marché. Or, d'autres manufacturiers tels que *Apple* et *Adobe* emploient un facteur de qualité différent. Ce facteur de qualité modifie la matrice de

quantification afin d'augmenter ou de diminuer la résolution binaire servant à quantifier les coefficients de la DCT-II 2D. Le pas de quantification de la matrice est décrit par les équations 4.1 et 4.2. Le facteur de qualité $Q_{facteur}$ sert à déterminer le facteur d'élargissement $S_{facteur}$ nécessaire au calcul de la nouvelle matrice des pas de quantification. Le tableau 4-5 correspond aux pas de quantification standards proposés par la norme T.81 pour une qualité de 75. Le tableau 4-6 correspond aux nouveaux pas de quantification $\Delta_{(u,v)}$ pour un facteur de qualité $Q_{facteur}$ de 80. Les pas de quantification sont alors plus petits, ce qui donne une granularité plus fine d'échantillonnage et donc, une meilleure qualité d'image reconstruite.

$$\Delta_{(u,v)} = int\left(\frac{Q_{table(u,v)} \cdot S_{facteur} + 50}{100}\right) \tag{4.1}$$

$$S_{facteur} = \begin{cases} int\left(5000/Q_{facteur}\right) si\ \ Q_{facteur} < 50 \\ 200 - 2 \cdot Q_{facteur}\ \ si\ \ Q_{facteur} \geq 50 \end{cases} \tag{4.2}$$

Tableau 4-5 Pas de quantification selon la norme T.81

Luminance, Qualité = 75							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tableau 4-6 Pas de quantification pour une qualité de 80

Luminance, Qualité = 80							
6	4	4	6	10	16	20	24
5	5	6	8	10	23	24	22
6	5	6	10	16	23	28	22
6	7	9	12	20	35	32	25
7	9	15	22	27	44	41	31
10	14	22	26	32	42	45	37
20	26	31	35	41	48	48	40
29	37	38	39	45	40	41	40

4.3.2 JPEG_FX

JPEG_FX emploie la même méthode de quantification que IJG. Il s'agit d'une des versions de développée dans le cadre du présent projet de recherche. L'architecture du CODEC repose sur la norme T.81 mais la DCT-II 2D employée pour JPEG_FX est la réalisation VHDL de XIL-DCTII. Ainsi, les performances montrées plus loin se différencient uniquement par le type d'implémentation de la DCT-II 2D.

4.3.3 JPEG_INT

Le JPEG INT est similaire en tout point au JPEG_FX à l'exception encore une fois de l'implémentation de la DCT-II 2D. L'algorithme FSM-DCTII est employé ici.

Afin de démontrer la comparaison entre les différents CODEC, il est nécessaire d'employer un débit d'image reconstruite similaire dans tous les cas. Or, comme le facteur de qualité $Q_{facteur}$ régit ce débit dans le cas du JPEG, il faut spécifier le facteur de qualité employé lors de la comparaison. Ce facteur de qualité change en fonction de l'image entrante et par conséquent, le PSNR change aussi. Le tableau 4-7 spécifie le facteur de qualité $Q_{facteur}$ employé pour chaque image et pour chaque débit utilisé comme point de comparaison.

Tableau 4-7 Facteur de qualité employé à différents débits

	Débit (bpp)			
	0.125	0.25	0.5	1
Image	Qualité			
Lenna	Q2	Q12	Q40	Q78
Barbara	Q1*	Q7	Q20	Q59
Baboon	Q1*	Q5	Q11	Q31
Goldhill	Q2	Q9	Q24	Q62
Peppers	Q1*	Q10	Q34	Q73

Une étoile à côté du facteur de qualité signifie qu'il aurait fallu un facteur de qualité inférieur à ce dernier pour obtenir le débit spécifié. Or, un débit inférieur à 1 est invalide. Cela signifie

donc que le JPEG ne peut encoder l'image entrante à un débit égal à ce qui est spécifié dans le tableau. De façon conceptuelle, il est possible d'entrevoir que la limite inférieure de débit résultant pour le JPEG se situe entre 0.125 bpp et 0.25 bpp. La figure 4.1 illustre à droite la limite d'encodage de l'image Lenna avec un encodage JPEG dont le facteur de qualité est à deux, soit 0,125 bpp. Pour le même débit, l'image produite par le CODEC FFNN est présentée à gauche. Le FFNN dispose donc d'une marge de manœuvre supérieure à celle du JPEG lorsqu'il s'agit d'encoder à faible débit. Le tableau 4-8 illustre les performances en termes de PSNR pour l'ensemble des CODEC évalués de même que pour les versions du CODEC JPEG incorporant les architectures expérimentales de DCT-II 2D. Ce format de tableau provient de [Auclair Beaudry, 2009]. Cette disposition des données permet de réaliser une comparaison rapide entre différents CODEC. Il est possible d'en déduire que le FFNN est un CODEC supérieur au JPEG en ce qui à trait à la qualité de reconstruction des images et au rapport de compression. Il est aussi intéressant de constater qu'il n'y a que très peu de différences entre toutes les implémentations du JPEG développées pour l'expérimentation. Cela signifie donc que le traitement DCT-II 2D FSM-DCTII est équivalent à celui de XIL-DCTII et à celui de la DCT-II 2D de référence.

Figure 4.1 - Lenna 0.125 bpp avec FFNN (gauche) et Lenna 0.125 bpp avec JPEG (droite)

Tableau 4-8 PSNR des CODEC selon différents débits

		Débit (bpp)			
		0.125	0.25	0.5	1
Image	Codeur	PSNR (dB)			
Lenna	FFNN	31.21	34.31	37.31	40.22
	J2000	31.02	34.15	37.27	40.33
	JPEG_STD	23.59	31.09	35.13	38.25
	JPEG_FX	24.27	31.16	35.24	38.21
	JPEG_INT	24.27	31.16	35.22	38.23
Barbara	FFNN	27.21	30.47	34.47	39.04
	J2000	25.87	28.89	32.87	38.07
	JPEG_STD	22.38	24.87	28.34	33.89
	JPEG_FX	22.39	24.98	28.55	33.82
	JPEG_INT	22.40	24.96	28.57	33.82
Baboon	FFNN	21.70	23.46	25.90	29.48
	J2000	21.68	23.18	25.57	29.11
	JPEG_STD	19.92	21.52	23.67	26.54
	JPEG_FX	19.93	21.46	23.50	26.57
	JPEG_INT	19.93	21.47	23.50	26.57
Goldhill	FFNN	28.70	30.88	33.46	36.85
	J2000	28.49	30.53	33.24	36.54
	JPEG_STD	23.74	28.29	31.45	34.41
	JPEG_FX	23.76	28.33	31.41	34.41
	JPEG_INT	23.74	28.34	31.41	34.41
Peppers	FFNN	30.65	33.1	35.25	38.04
	J2000	30.79	33.54	35.80	38.17
	JPEG_STD	24.29	30.13	33.82	36.12
	JPEG_FX	24.30	30.11	33.82	36.11
	JPEG_INT	24.30	30.09	33.80	36.09

4.4 Ressources matérielles

Les ressources matérielles employées pour chaque algorithme expérimental sont propres à la technologie FPGA employée et à l'outil de synthèse. Ainsi, ces résultats sont des outils de comparaison des implémentations, mais ne doivent pas être perçus comme étant une référence en absolu. Une autre technologie de FPGA ou une autre révision d'outil de synthèse pourraient fournir des résultats substantiellement différents. Il est important de connaître l'ampleur des ressources matérielles employées par chacun des algorithmes pour entrevoir la possibilité d'utiliser ces derniers dans des petits systèmes avec peu de ressources. Les deux sections suivantes exposent la consommation de ressources de chacun des algorithmes mis à l'épreuve.

4.4.1 XIL-DCTII

XIL-DCTII emploie des multiplicateurs. Cela a donc pour effet de concentrer le design VHDL aux alentours de ces derniers. La série SX des Virtex-5 de Xilinx voit ses tuiles DSP48E disposées en colonnes de part et d'autres de l'arbre de distribution d'horloge Nord-Sud. Cette disposition force donc le design à emprunter un étalement Nord-Sud en suivant la disposition des DSP48E. La figure 4.2 présente la disposition de l'algorithme XIL-DCTII dans le FPGA V5SX50T. À titre de comparaison, la figure 4.3 illustre la disposition de l'architecture XIL-DCTII avec une transformée issu de FSM-DCTII. Le but est de démontrer que le placement des composantes dépend de la localisation des multiplicateurs dans le cas où l'algorithme DCT-II employé nécessite des tuiles DSP48E.

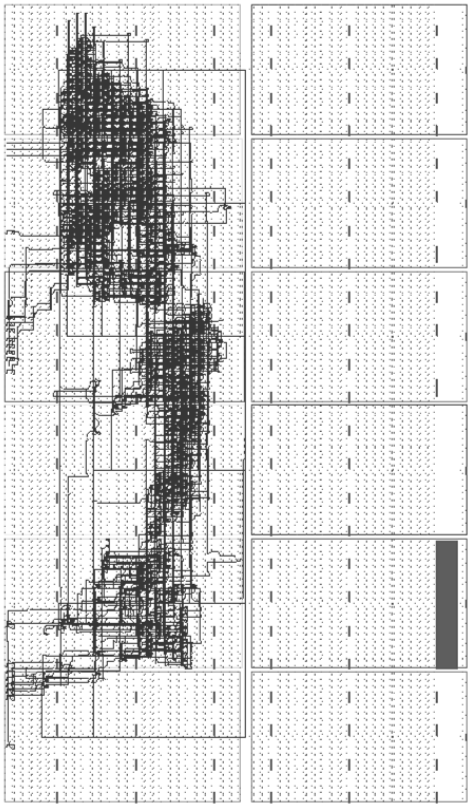

Figure 4.2 - Disposition de XIL-DCTII dans le V5 SX50T

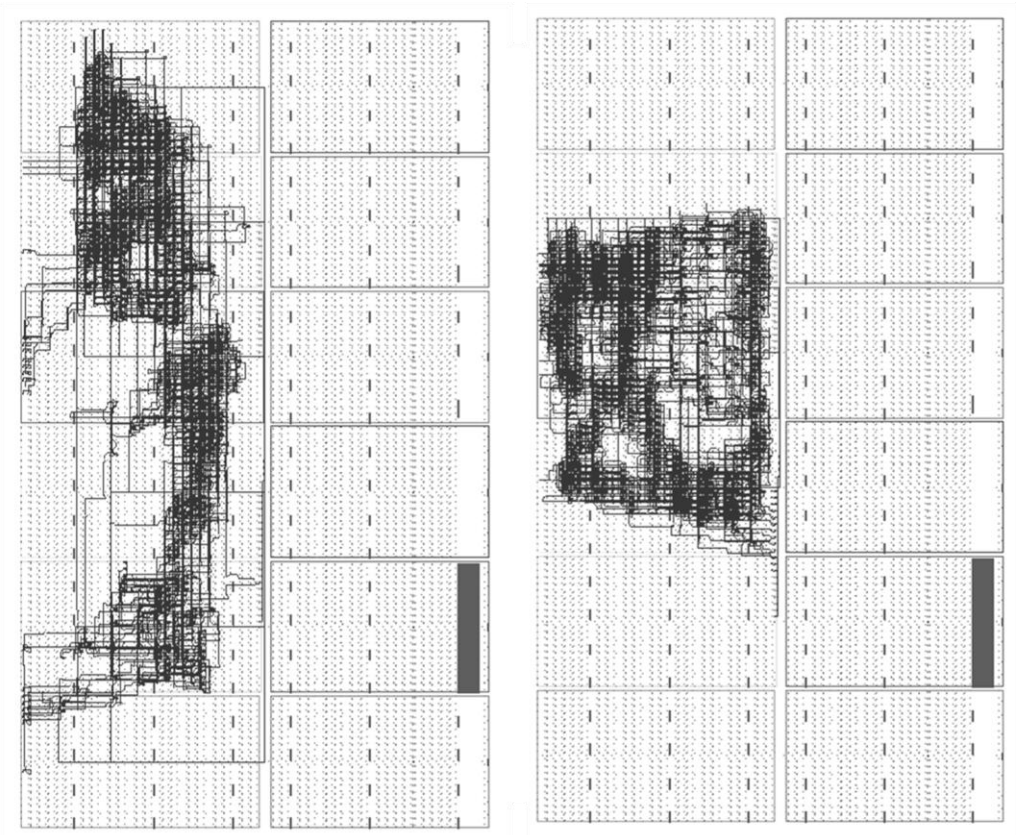

Figure 4.3 - Comparaison du placement entre XIL-DCTII (gauche) et FSM-DCTII (droite)

Les ressources employées par l'implémentation de XIL-DCTII sont présentées au tableau 4-9. Il s'agit d'une copie conforme du rapport de synthèse d'ISE 12.3.

Tableau 4-9 Ressources employées par XIL-DCTII dans un Virtex-5™ SX50T

Nom de la ressource (Configuration)	**Quantité utilisée**	**Quantité disponible**	**Niveau d'utilisation**
Registres de slice	1088	32640	3%
Lut de slice	599	32640	1%
Lut (logique)	421	32640	1%
Lut (Mémoire)	178	12480	1%
DSP48E	34	288	11%

En somme, la structure XIL-DCTII occupe 3% des registres disponibles dans les *Slice* du FPGA et 1% des *Look-Up Table* (LUT) disponibles dans le Virtex-5™ SX50T. Les termes entre parenthèses signifient la configuration de la ressource ciblée. Ainsi, il est clair que la majorité des LUT employées est vouée à une application de mémoire et d'accumulateurs en raison de la structure de l'algorithme. XIL-DCTII occupe une petite empreinte sur la surface du silicium. Cela se fait au détriment des unités de multiplications utilisées. 11% de ces dernières sont employées, ce qui en fait de XIL-DCTII un algorithme plus dispendieux. Plus un FPGA comporte de DSP48E, plus le prix de ce dernier est onéreux.

4.4.2 FSM-DCTII

L'algorithme FSM-DCTII n'est pas synthétisable dans le FPGA employé dans le cadre de la présente recherche. Il n'y a pas assez de ressources matérielles pour réaliser une telle structure. Au lieu d'employer un V5SX50T comme c'est le cas dans le présent projet, il faudrait augmenter le nombre de LUT et passer au FPGA supérieur suivant, soit le V5SX95T.

Tableau 4-10 Ressources employées par FSM-DCTII dans un Virtex-5™ SX50T

Nom de la ressource (Configuration)	Quantité utilisée	Quantité disponible	Niveau d'utilisation
Registres de slice	25760	32640	78%
Lut de slice	44208	32640	135%
Lut (logique)	44208	32640	135%
Lut (Mémoire)	--	12480	0%
DSP48E	--	288	0%

Le tableau 4-10 fait état des ressources nécessaire pour implémenter l'algorithme FSM-DCTII pour une taille de 8 × 8. Deux éléments sont remarquables dans les résultats présentés. Le premier étant le nombre total de LUT et de registres utilisés. La structure *lifting* employée dans le cadre de FSM-DCTII débouche sur une augmentation drastique de l'utilisation de ces

ressources. La cause première est la duplication des signaux servant à approximer des multiplications. Non seulement ces derniers doivent être instanciés autant de fois qu'une addition ou soustraction est effectuée, mais ils comportent chacun une plage dynamique considérable. La puissance de l'algorithme est liée à la parallélisation des opérations. Il s'agit de sa force et de son inconvénient majeur, car cela requiert une quantité importante de logique. Le second élément à remarquer est l'absence totale de LUT configurés en mémoire. Cela témoigne d'une structure comportant un délai de traitement minimum.

4.5 Extension pour DCT de grande taille

La DCT-II 2D mise en application dans le cadre de l'implémentation JPEG en VHDL a nécessité une plage dynamique de 48 bits pour réaliser les calculs impliquant des coefficients DCT-II à 16 bits et des pixels en entrée de 8 bits dans le cas de XIL-DCTII. Le FSM-DCTII nécessite autant de précision pour accomplir un même travail. Les multiplicateurs dédiés sont remplacés par des structures *lifting*, mais la précision requise pour obtenir les coefficients reste la même dans les deux implémentations.

Dans un effort visant à mesurer la possibilité d'employer l'algorithme FFNN pour traiter des images médicales, une version à 32 coefficients de l'algorithme FSM-DCTII est développée sous MATLAB ®. Afin de trouver les coefficients de la structure *lifting*, le problème est posé sous forme d'une fonction d'optimisation à 32 dimensions. Les coefficients sont comparés à la DCT-II de référence de MATLAB ® et approximés pour être représentés sous forme entière. Il s'agit donc d'une DCT-II à 32 coefficients qui doit être appliquée sur deux dimensions. L'équation matricielle de la matrice de transformation T32 régissant cette DCT-II est décrite via l'équation 4.3. Cette dernière est composée des matrices de rotation U_2 jusqu'à U_{16} suivies de matrices de permutation.

$$T_{32} = \begin{pmatrix} 1 & 0 & & & 0 & \\ 0 & 1 & & & & \\ & & U_2 & 0 & 0 & 0 \\ & 0 & 0 & U_4 & 0 & 0 \\ & & 0 & 0 & U_8 & 0 \\ & & 0 & 0 & 0 & U_{16} \end{pmatrix} \tag{4.3}$$

Un exemple impliquant la matrice U_2 est donné pour illustrer la structure de *lifting* résultante dans le cas de l'approximation développée sous MATLAB ® aux équations 4.4 à 4.6.

$$U_2 = \begin{pmatrix} cos\frac{\pi}{8} & sin\frac{\pi}{8} \\ -sin\frac{\pi}{8} & cos\frac{\pi}{8} \end{pmatrix} \tag{4.4}$$

$$G_{\frac{\pi}{8}} = \begin{pmatrix} 1 & -\frac{1-cos\frac{\pi}{8}}{sin\frac{\pi}{8}} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ sin\frac{\pi}{8} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1-cos\frac{\pi}{8}}{sin\frac{\pi}{8}} \\ 0 & 1 \end{pmatrix} \tag{4.5}$$

$$G_{\frac{\pi}{8}} = \begin{pmatrix} 1 & p_{\frac{\pi}{8}} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ u_{\frac{\pi}{8}} & 1 \end{pmatrix} \begin{pmatrix} 1 & p_{\frac{\pi}{8}} \\ 0 & 1 \end{pmatrix} \tag{4.6}$$

Les coefficients approximés pour l'ensemble des matrices de rotation se trouvent au tableau 4-11.

Tableau 4-11 Coefficients pour une DCT FSM-DCTII de largeur 32

Structure de *lifting*	Approximation dyadique
$p_{\frac{\pi}{8}}$	25/128
$u_{\frac{\pi}{8}}$	49/128
$p_{\frac{\pi}{16}}$	13/128
$u_{\frac{\pi}{16}}$	25/128
$p_{\frac{3\pi}{16}}$	39/128
$u_{\frac{3\pi}{16}}$	71/128
$p_{\frac{\pi}{32}}$	7/128
$u_{\frac{\pi}{32}}$	13/128
$p_{\frac{3\pi}{32}}$	19/128
$u_{\frac{3\pi}{32}}$	37/128
$p_{\frac{5\pi}{32}}$	1/4
$u_{\frac{5\pi}{32}}$	15/32
$p_{\frac{7\pi}{32}}$	23/64
$u_{\frac{7\pi}{32}}$	81/128
$p_{\frac{\pi}{64}}$	7/128
$u_{\frac{\pi}{64}}$	3/64
$p_{\frac{3\pi}{64}}$	9/128
$u_{\frac{3\pi}{64}}$	19/128
$p_{\frac{5\pi}{64}}$	1/8
$u_{\frac{5\pi}{64}}$	31/128
$p_{\frac{7\pi}{64}}$	11/64
$u_{\frac{7\pi}{64}}$	43/128
$p_{\frac{9\pi}{64}}$	29/128
$u_{\frac{9\pi}{64}}$	27/64
$p_{\frac{11\pi}{64}}$	35/128
$u_{\frac{11\pi}{64}}$	33/64
$p_{\frac{13\pi}{64}}$	21/64
$u_{\frac{13\pi}{64}}$	19/32
$p_{\frac{15\pi}{64}}$	49/128
$u_{\frac{15\pi}{64}}$	43/64

Il est possible, avec ces résultats, d'avoir un aperçu de la résolution binaire nécessaire pour effectuer une telle transformée. En commençant par la WHT, la DCT-II de 32 coefficients nécessite une WHT de même taille. La résolution binaire est donc celle des échantillons d'entrée à laquelle vient s'additionner cinq bits supplémentaires. La plage dynamique est alors de 13 bits uniquement pour la WHT. Les coefficients doivent par la suite être multipliés par la matrice T32 présenté à l'équation 4.3. Au maximum, il est évident que le dernier coefficient est un de ceux qui impliquent le plus d'opérations de rotation. Pour une DCT-II de 32 coefficients, il y aura un total de quatre rotations sur ce dernier. Ces quatre rotations impliquent un total de 12 rotations binaires successives de sept bits vers la droite. Il est donc assuré que la plage dynamique nécessaire est donc de 84 bits pour cet étage. Les additions et soustractions propres à la structure de *lifting* sont négligées dans les calculs. Pour en connaître le nombre exact, il faudrait représenter chaque coefficient en représentation entière minimale. Il est toutefois assuré de trouver au minimum huit soustractions et quatre additions. Sans la WHT, la plage dynamique de la matrice T32 est alors de 96 bits au minimum. Pour obtenir une matrice T32 sur deux dimensions, il est donc nécessaire d'avoir au minimum 192 bits de plage dynamique.

Une telle structure est difficilement réalisable sans congestionner un FPGA. Il faudrait d'ailleurs distribuer dans le temps les opérations mathématiques d'une telle structure pour éviter des problèmes électriques et diminuer la fréquence de l'horloge afin de rencontrer les contraintes de placement. Il faut aussi noter qu'il est impensable d'employer des entiers ou autres types de déclarations définies par le standard IEEE-1076 car de telles structures ne sont pas définies pour une précision de 192 bits. Il faudra donc développer des additionneurs et soustracteurs pour une plage dynamique élevée afin de réaliser les opérations de base couvertes par la DCT-II 32 × 32. La réalisation de FSM-DCTII de taille 8 × 8 a démontré que ce genre de problème est bien réel. Même pour une implémentation de petite taille, un FPGA de taille supérieure à celle employée dans le cadre du projet est nécessaire.

4.6 Résultats sous JPEG matériel

Le banc d'essai VHDL incorporant un CODEC JPEG bénéficie de la rapidité de traitement de la nouvelle architecture proposée pour la DCT-II 2D. La majorité des délais de traitement ne proviennent plus de la DCT-II 2D en particulier, mais bien des mémoires intermédiaires de type FIFO et des accumulateurs de ligne. La figure 4.4 témoigne de la simulation du fonctionnement du CODEC JPEG matériel et des performances d'une telle implémentation. Il s'agit d'une vue à l'écran des chronogrammes du banc de test sous l'application MODELSIM ® . Chaque ligne représente un train binaire entre les modules de l'encodeur et du décodeur. Cette figure est aussi disponible en annexe C avec de plus amples détails sur la nature des signaux affichés.

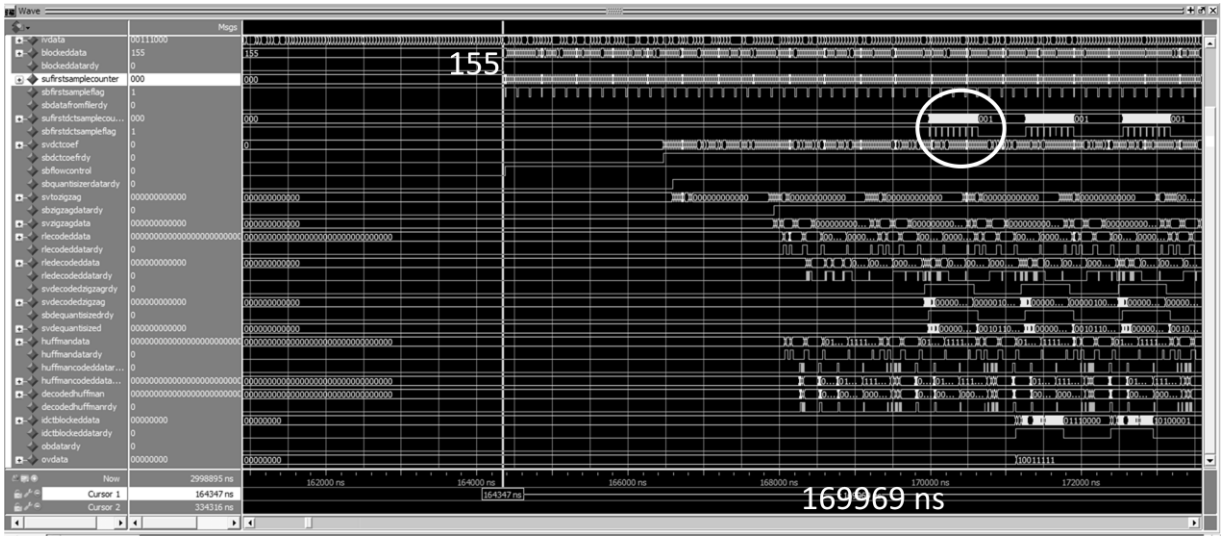

Figure 4.4 - Simulation MODELSIM ® du CODEC matériel

Sur cette même figure 4.4, le nombre 155 constitue la valeur des quatre premiers pixels de l'image Lenna sur le flux de pixel entrant dans le CODEC. Les premiers chronogrammes illustrent donc le chemin d'entrée des données. Le cercle illustre le traitement par paquets de huit lignes nécessaires au traitement par DCT-II 2D suivant les accumulateurs de ligne. Ces paquets sont visibles parce que la transformée DCT-II 2D traite ces derniers plus vite que la vitesse à laquelle ils proviennent du canal de communication. Ce délai d'accumulation est de

loin supérieur au traitement quasi-instantané de la DCT-II 2D. Finalement, la valeur de 169969 ns est le temps requis pour traiter l'image de Lenna en noir et blanc, soit 262144 pixels. Il est donc question d'un délai de 1/16 d'image à une résolution supérieure à celle suggérée par l'application de téléassistance en salle d'urgence, et ce pour l'encodage et le décodage combinés. Cette implémentation de CODEC serait une excellente candidate pour l'application en raison de son traitement en temps réel.

La version matérielle du JPEG développée conjointement avec la nouvelle architecture DCT-II dispose d'une mémoire de quantification. Cette mémoire peut-être reprogrammée de façon à changer le ratio de compression. Dans le cadre du présent projet, cette reprogrammation est nécessaire pour obtenir un comparatif avec les autres CODEC à différents débits. La figure 4.5 expose l'image de Lenna à différents débits de comparaison produits par le CODEC matériel. Du fragment A à D, les débits sont respectivement de 0,125 bpp, 0,25 bpp, 0,5 bpp et 1 bpp. En regardant les résultats du CODEC JPEG, il peut être anticipé que la limite de compression de ce dernier se situe entre 0,125 bpp et 0,25 bpp.

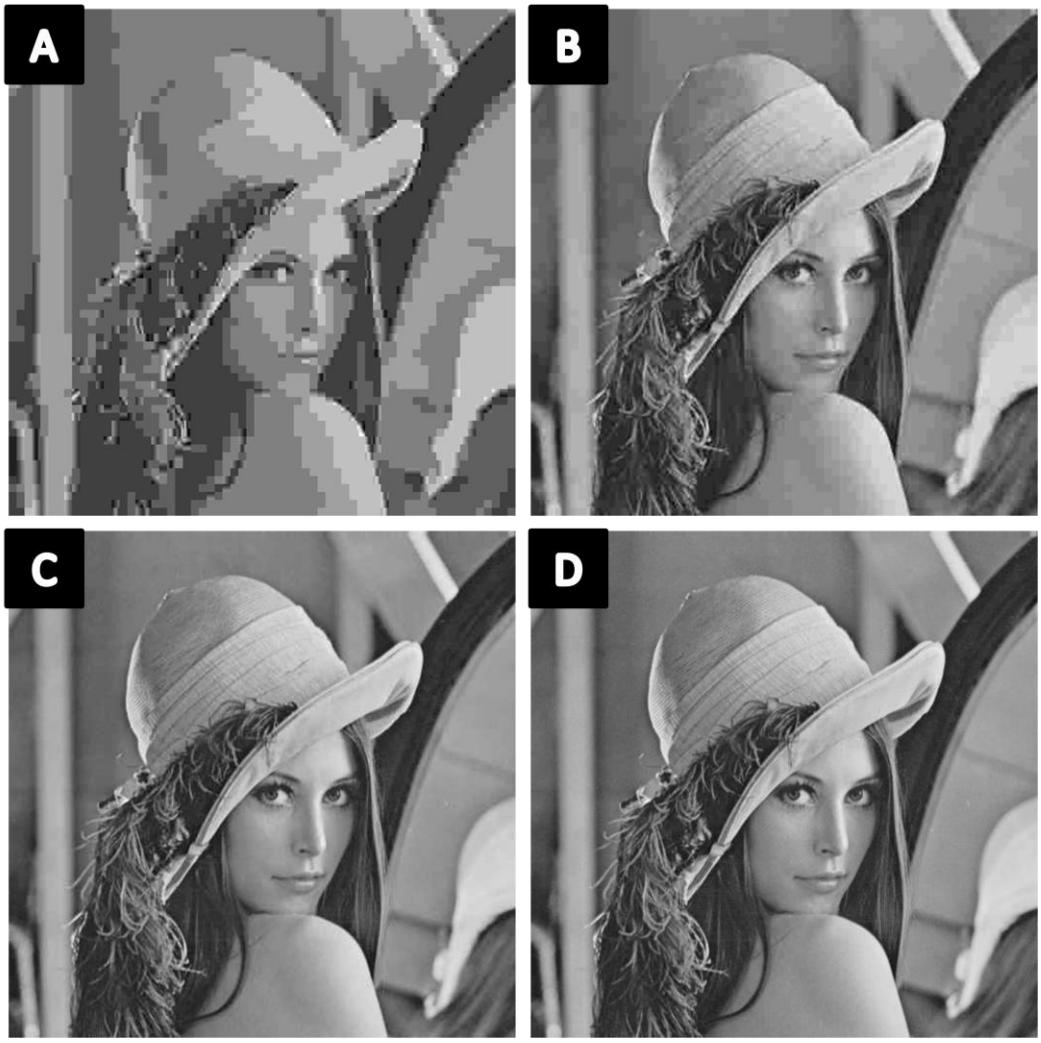

Figure 4.5 - Lenna 0,125 bpp (A), 0,25 bpp (B), 0,5 bpp(C) et 1 bpp(D).

CHAPITRE 5
DISCUSSION

Les résultats de recherche présentés au chapitre 4 sont prometteurs à plusieurs niveaux et il est possible d'en tirer plusieurs observations liées aux problèmes soulevés aux chapitres 1 et 3. Une première série d'observations porte sur les performances des algorithmes XIL-DCTII et FSM-DCTII. Une seconde traite des performances générales des deux CODEC candidats JPEG et FFNN.

5.1 Performance des algorithmes DCT-II 2D

Le chapitre 4 présente les résultats sous différents aspects de l'implémentation des algorithmes XIL-DCTII et FSM-DCTII. Ces derniers sont repris ici et commentés afin de mettre en relief les avantages et inconvénients majeurs de ces algorithmes.

5.1.1 Vitesse de traitement

Malgré une vitesse d'horloge inférieure à celle de l'algorithme XIL-DCTII, la vitesse de traitement de FSM-DCTII est impressionnante. Elle surclasse les performances obtenues par [Roman C *et al.*, 2006] qui proposèrent un des fondements de cette même architecture. Sa force réside dans la capacité à traiter 64 pixels simultanément. Cet avantage se traduit donc par un débit élevé pour une fréquence d'horloge raisonnable. Cet algorithme est donc un excellent candidat pour une version faible coût et faible puissance de FPGA. Une vitesse de traitement de 12,842 Gb/s est suffisante pour traiter simultanément quatre flux vidéo de 3 Gb/s. Chacun de ces flux vidéo possède une cadence d'image de 60 fps pour une résolution de 1920 × 1080 pixels. Même si l'algorithme XIL-DCTII n'est pas aussi performant, il faut toutefois noter que les deux algorithmes dépassent la contrainte fixée au chapitre 1, soit celle d'un débit de 62,208 Mb/s.

5.1.2 Délai de traitement

FSM-DCTII est aussi le plus performant en termes de délai de traitement. Le délai de 79,7 ns est environ 200 000 fois plus petit que le délai de 17 ms imposé par les spécifications de traitement en temps réel chiffrées au chapitre 1. La structure à deux dimensions est distribuée de part et d'autres de la transformée. En traitant 64 échantillons simultanément, il n'y a aucune nécessité d'avoir un tampon de transposition pour traiter les échantillons selon une autre dimension. Le délai initial de traitement est donc réduit considérablement. Bien que les deux algorithmes dépassent largement les objectifs de temps de latence initial, un algorithme ne nécessitant aucun tampon est un choix judicieux pour une implémentation de CODEC faible coût et petite surface de silicium. Encore une fois, les deux algorithmes dépassent les spécifications du chapitre 1 mais le FSM-DCTII est de loin supérieur.

5.1.3 Ressources matérielles

Il n'y a pas de spécifications concernant la quantité de ressources matérielles employées. Il n'y a pas de limite fixée quant au nombre de LUT employées ou de multiplicateurs utilisés. Or, il est impensable de développer des algorithmes nécessitant des ressources illimitées. En ce sens, les deux algorithmes implémentés sont aux antipodes en termes de stratégie de développement.

- XIL-DCTII est un algorithme nécessitant peu d'espace de silicium mais requiert la présence de quelques multiplicateurs dédiés de haute performance, c'est-à-dire les tuiles DSP48E. Il est donc normal de l'employer dans un FPGA comportant un grand nombre de modules ou dans lequel nous désirons implémenter plusieurs copies de l'algorithme.
- FSM-DCTII est un algorithme de grande performance. L'idée n'est pas nécessairement d'en avoir plusieurs copies dans un même FPGA, mais bien de multiplexer temporellement les accès à ce dernier. En ce sens, il ne requiert que de la logique distribuée, aucun accumulateur mémoire et aucune tuile spécialisée telles les DSP48E. Il s'agit donc d'un algorithme pouvant s'implémenter dans un FPGA bon marché sans ressources particulières. C'est ainsi une approche faible coût offrant de grandes

performances. Si le but est toutefois de minimiser la surface de silicium employée, XIL-DCT semble plus approprié.

5.1.4 Extension pour DCT de grande taille

L'introduction au chapitre 2 démontre la quantité de multiplicateurs requis pour réaliser une DCT-II 2D avec XIL-DCTII et de façon conventionnelle. Malheureusement, il semble que le problème majeur des transformées de grande taille soit la plage dynamique des coefficients de sortie. Il s'agit d'un frein à leur implémentation dans un système dédié. FSM-DCTII et XIL-DCTII souffrent tous deux de cet inconvénient lorsque portées à une dimension supérieure à 8 × 8.

5.2 Performance générale des CODEC

Cette section traite de la performance des différentes versions du CODEC JPEG implémentés dans le cadre de l'expérience. Une revue de la performance du FFNN est aussi mise en contraste par rapport aux résultats obtenus de l'implémentation de différents types de DCT-II 2D.

5.2.1 Comparaison entre les JPEG

Il est intéressant de constater que d'employer une version particulière de DCT-II ou une autre a très peu d'effets sur le PSNR de l'image résultante provenant du CODEC JPEG. Il est clair que les coefficients dans le domaine de la transformée sont différents, mais cela a peu d'impact sur le CODEC général en raison de la quantification sévère employée pour atteindre un faible débit. En ce sens, l'approximation employée de la DCT-II, que ce soit par le biais de XIL-DCTII ou FSM-DCTII, est équivalente à la DCT-II dans le contexte d'une mise en application sous JPEG à faible débit. La figure 5.1 donne une comparaison visuelle des différents PSNR à 0,25 bpp. Le CODEC FFNN fournit une image plus agréable à l'œil, ce qui concorde avec les

travaux de [Auclair Beaudry, 2009]. Or, en regardant les trois versions de JPEG, il est possible de constater que les différentes DCT-II essayées dans le cadre de l'expérience n'ont que très peu d'effets visuels et se comportent sensiblement de la même façon.

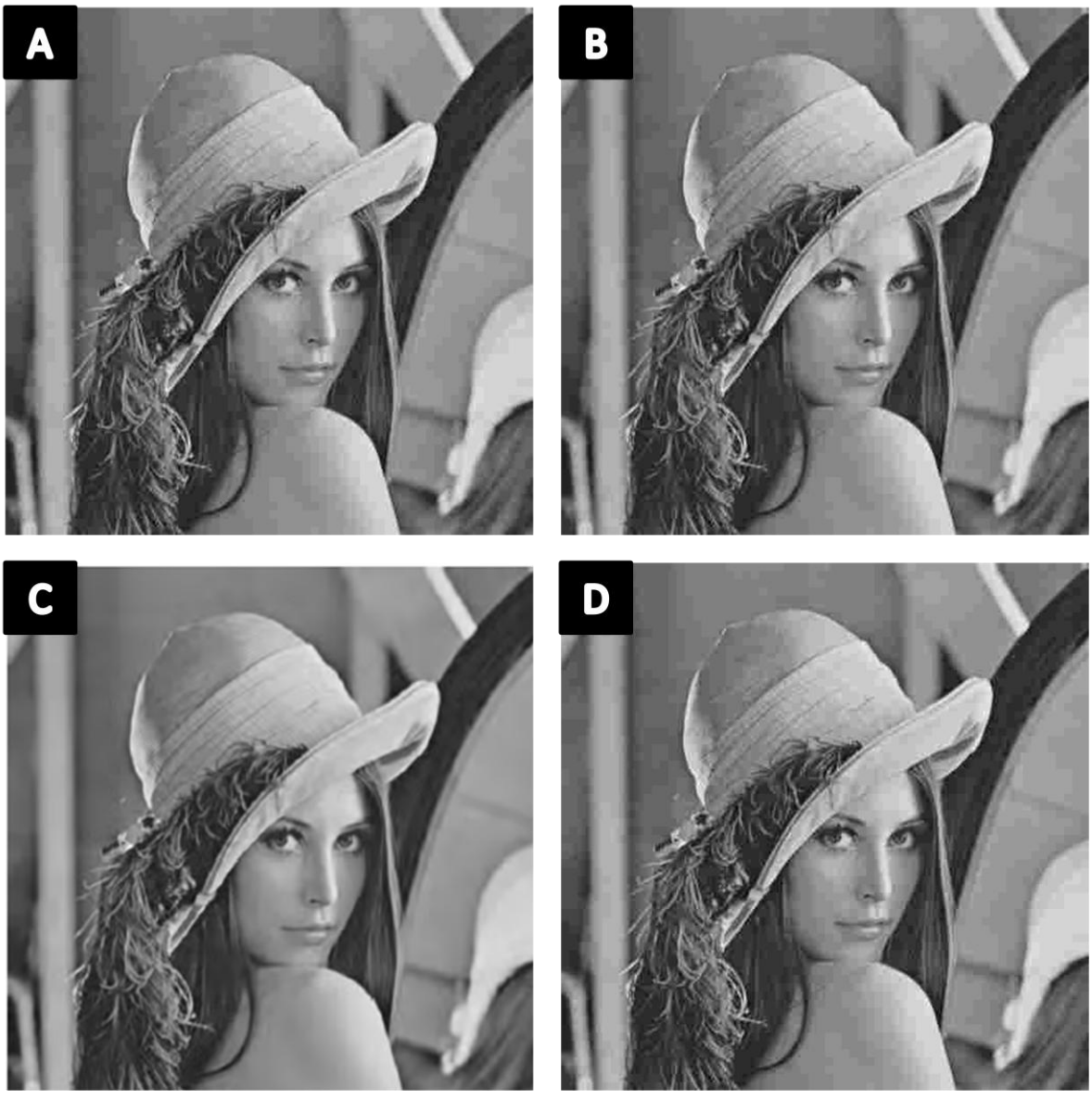

Figure 5.1 - Reconstruction JPEG_STD (a), JPEG_FX (b), FFNN (c) et JPEG_INT (d) à 0,25 bpp

5.2.2 Performances de FFNN

Les travaux [Auclair Beaudry, 2009] démontrent les performances d'un tel CODEC en termes de PSNR et de débit de l'image encodée. Tel qu'expliqué à l'introduction du chapitre 1, deux goulots d'étranglement sont présents dans ce dernier, soit un au niveau de la DCT-II 32 × 32 et un second au niveau du codage entropique. La présente expérience démontre que l'algorithme FSM-DCTII 8 × 8 est un excellent candidat en termes de vitesse de traitement, délai de traitement et ressources employées pour un CODEC. L'expérience ne permet pas de vérifier si le FFNN bénéficie des mêmes qualités s'il est implémenté conjointement au FSM-DCTII. Il a toutefois été démontré que FSM-DCTII porté à une dimension de 32 × 32 représente un défi de réalisation important en raison de la plage dynamique des coefficients de sortie d'une telle transformée. Le CODEC FFNN devrait être réévalué afin de vérifier son fonctionnement lorsqu'une DCT-II 8 × 8 ou 16 × 16 est employée. Si les performances en termes de PSNR, débit résultant, débit de traitement et délai de traitement étaient acceptables, le FFNN aurait la capacité de surclasser les CODEC JPEG et JPEG2000 au niveau des performances de PSNR et rapidité de traitement. Même s'il advenait qu'il soit moins rapide que le JPEG au niveau de son exécution, l'important est de respecter les contraintes de débit de traitement et de délai de traitement.

CHAPITRE 6
CONCLUSION

L'objectif de la recherche consistait à optimiser l'implémentation, dans un CODEC vidéo, de la transformée DCT-II 2D afin de rencontrer les exigences d'une application de téléassistance en salle d'urgence. Cette transformée est directement responsable des délais de traitement initiaux, limitant la transmission des images vidéo fluide et quasi sans délais.

Les résultats présentés au chapitre 5 démontrent qu'il est possible de respecter les contraintes de traitement imposés par la téléassistance en salle d'urgence et que les solutions avancées en matière de DCT-II 2D peuvent être implémentées dans un CODEC matériel. Il est donc possible de relever trois contributions théoriques et deux contributions pratiques des travaux réalisés dans le cadre du présent projet.

La première contribution théorique est d'avoir développé un nouveau concept d'implémentation de DCT-II 2D de haute performance. Si la taille de la DCT-II 2D requise par l'application finale exige que cette dernière soit de 8 × 8 ou plus petite, il est possible d'employer l'architecture FSM-DCTII avec les avantages qui en découlent. Il est donc possible d'utiliser ce nouveau concept dans des CODEC commerciaux tels le JPEG, le MPEG2 et même le H.264.

La seconde contribution théorique est d'avoir mis en lumière une méthode correcte de comparaison des DCT-II. La majorité des résultats présentés dans plusieurs travaux de recherche font état d'une comparaison entre les coefficients des différentes DCT-II pour en faire ressortir une erreur quadratique moyenne basée sur une DCT-II de référence. D'autres résultats proposent d'effectuer un balayage fréquentiel pour déterminer les performances des DCT-II sur une portion de leur plage d'utilisation. Les présents travaux de recherche ont démontré qu'une méthode appropriée permettant de comparer les différents algorithmes de DCT-II est d'employer ces derniers conjointement à leur inverse et de comparer le signal

original avec le signal reconstruit. Cela permet de prendre en considérations toutes les erreurs de traitement sur la chaîne complète de l'encodage et d'évaluer les transformées dans un domaine commun à toutes. Or, il est important de noter que cette approche ne permet pas, à elle seule, de juger de l'incidence des écarts entre les différentes implémentations de DCT-II, mais plutôt de fournir un juste référentiel de comparaison.

Enfin, la troisième contribution théorique a permis d'établir les limites d'utilisation des algorithmes FAST DCT-II sans multiplication dans le cadre d'une application 2D. Le compromis offert FSM-DCTII est de remplacer des unités de multiplication coûteuses par des registres et additionneurs. Malheureusement, cette substitution devient inefficace pour des dimensions de DCT-II 2D supérieures à 8 × 8. La surface de silicium employée devient trop importante par rapport à l'utilisation de quelques unités de multiplication. Cet aspect est souvent négligé dans le cadre de plusieurs recherches où l'efficacité n'est mesurée qu'avec des transformées de petites dimensions.

Comme contribution pratique, la première est d'avoir déterminé la quantité de ressources nécessaires à l'implémentation du nouveau type de DCT-II 2D pour la taille conventionnelle de 8 × 8 et d'avoir prédit la plage dynamique nécessaire à une implémentation de taille 16 × 16. De plus, contrairement à de nombreux travaux sur le sujet qui n'élaborent aucun calcul pour des DCT-II 16 × 16 et 32 × 32, les bancs d'essai générés au cours du présent travail permettent le calcul de coefficients pour des ordres supérieurs de FSM-DCTII. Le tableau 4-11 illustre les résultats de calculs pour un ordre supérieur à 8 × 8.

La seconde est de livrer un CODEC JPEG complet de haut calibre pouvant être utilisé dans le domaine de la transmission vidéo et comme point de départ solide pour une implémentation matérielle future du FFNN. Cette plateforme peut servir à la continuation de la recherche dans le cadre de l'application de téléassistance en salle d'urgence pour le développement de CODEC à base de DCT-II 2D.

Comme travaux futurs, malgré une grande rapidité d'exécution, ce qui en fait un CODEC temps réel hors pair, les travaux de [Gukhool, 2009] et [Auclair Beaudry, 2009] démontrent que le JPEG n'est pas adéquat dans le contexte de l'application de téléassistance en salle

d'urgence, et ce pour plusieurs raisons. Le FFNN est en soi un excellent CODEC en partie grâce à la simplicité de ses opérations mais aussi parce qu'il requiert une faible quantité de mémoire. Or, pour le rendre parfaitement adéquat à l'application en temps réel, il serait nécessaire d'approfondir les points suivants:

DCT-II de grande taille. Il n'y a aucun doute sur la performance de l'algorithme FSM-DCTII. Cependant, il est primordial d'optimiser ce dernier afin de le rendre portable sur des DCT-II de dimensions supérieures à 8. Il est nécessaire de comprendre l'impact de la plage dynamique des coefficients de sortie sur la plage dynamique réelle utilisée afin de minimiser la largeur binaire des coefficients. En connaissant la plage dynamique désirée des coefficients de sortie, il est possible, dans le cas particulier d'une compression à faible débit, de réduire la plage dynamique des opérations requises sans dévier numériquement de la transformée de référence. Cette recherche permettrait entre autre d'employer le CODEC FFNN sans modifications majeures de ce dernier.

Fonction de quantification du FFNN. Il manque un paramètre important pour compléter le CODEC FFNN. Il s'agit d'une fonction de décimation qui régit le comportement du quantificateur du FFNN afin d'adapter la quantification en fonction du débit désiré. Cette fonction doit fixer une plage dynamique des coefficients de la DCT-II 2D et déterminer le nombre de plans binaires à supprimer en vue d'atteindre le débit fixé. Il s'agit d'une fonction similaire à la fonction de qualité du JPEG. Une telle fonction impose donc la plage dynamique maximale des coefficients de la DCT-II 2D et ouvre donc la porte à une optimisation de cette dernière.

ANNEXE A

IMAGES DE TEST

Figure A.1 - Lenna

Figure A.2 - Barbara

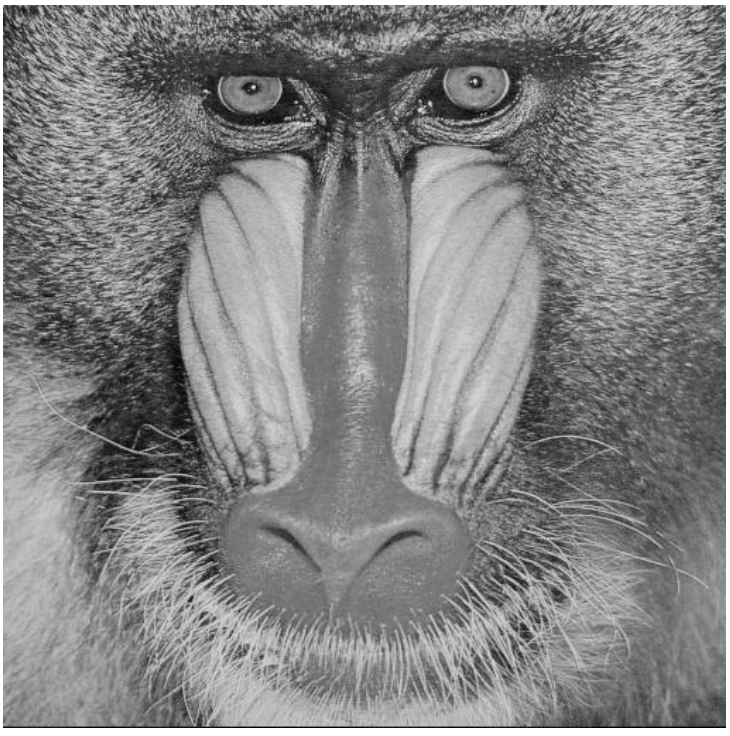

Figure A.3 - Baboon

Figure A.4 - Goldhill

Figure A.5 - Peppers

ANNEXE A

ANNEXE B

STRUCTURE DU BANC D'ESSAI

La figure B.1 symbolise la hiérarchie des principaux fichiers `vhdl` et cœurs `xco`. Elle doit être lue dans le sens des flèches.

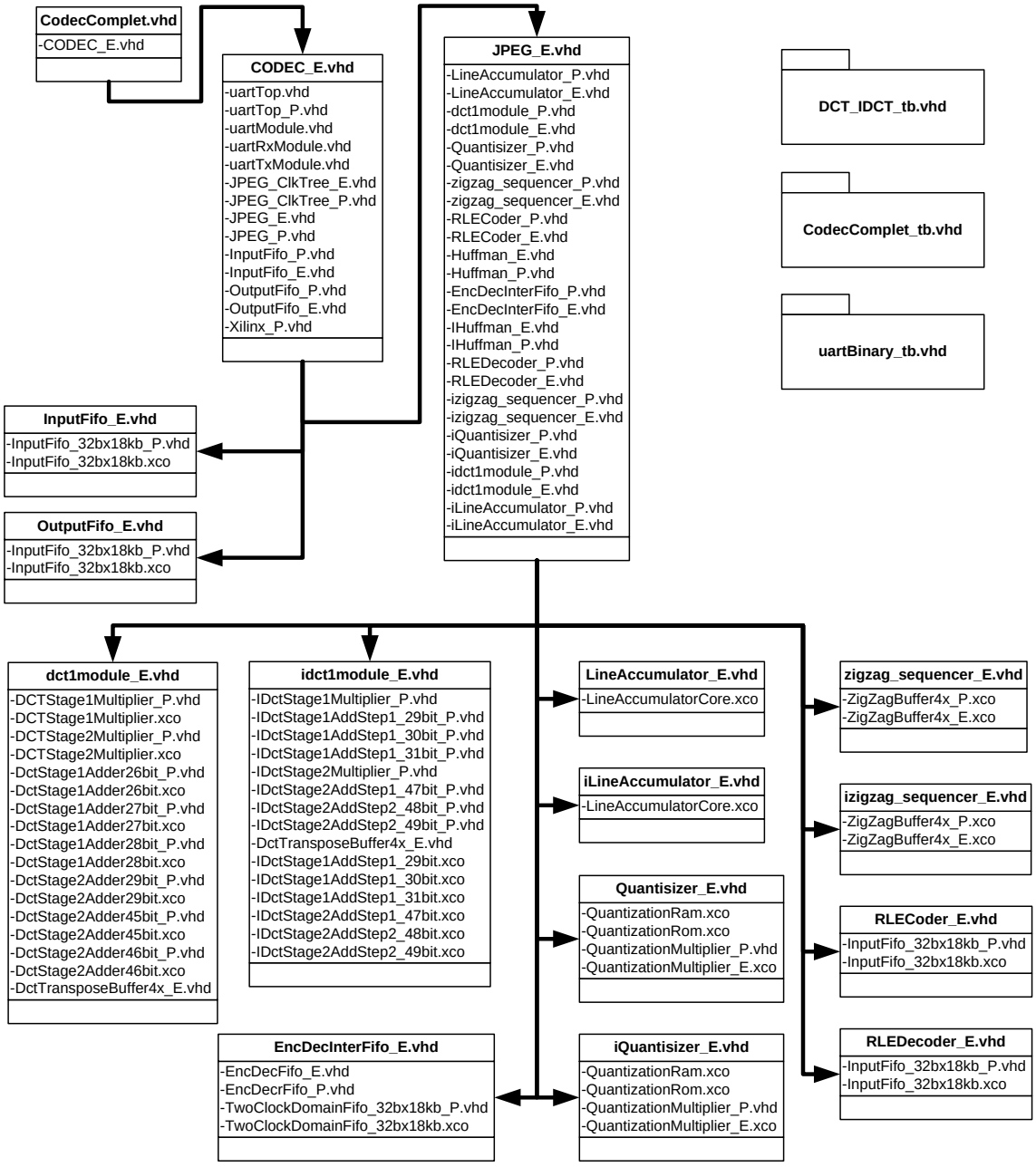

Figure B.1 - Hiérarchie

Le banc d’essai prend donc la forme d'un arbre renversé. Le fichier `JPEG_E.vhd` est le plus volumineux de tout l'environnement d'essai. Les fichiers non segmentés signifient qu'ils comportent uniquement des fonctions et qu'ils ne sont pas composés d'autres fichiers. Les trois fichiers indépendants sont des bancs d'essai différents. Le diagramme ne couvre pas les processus et fonction inclus dans chaque fichier. Les instances peuvent être composées de plusieurs modules du même type et interfacer ces mêmes modules de plusieurs façons.

ANNEXE C

SIMULATION DU CODEC MATÉRIEL

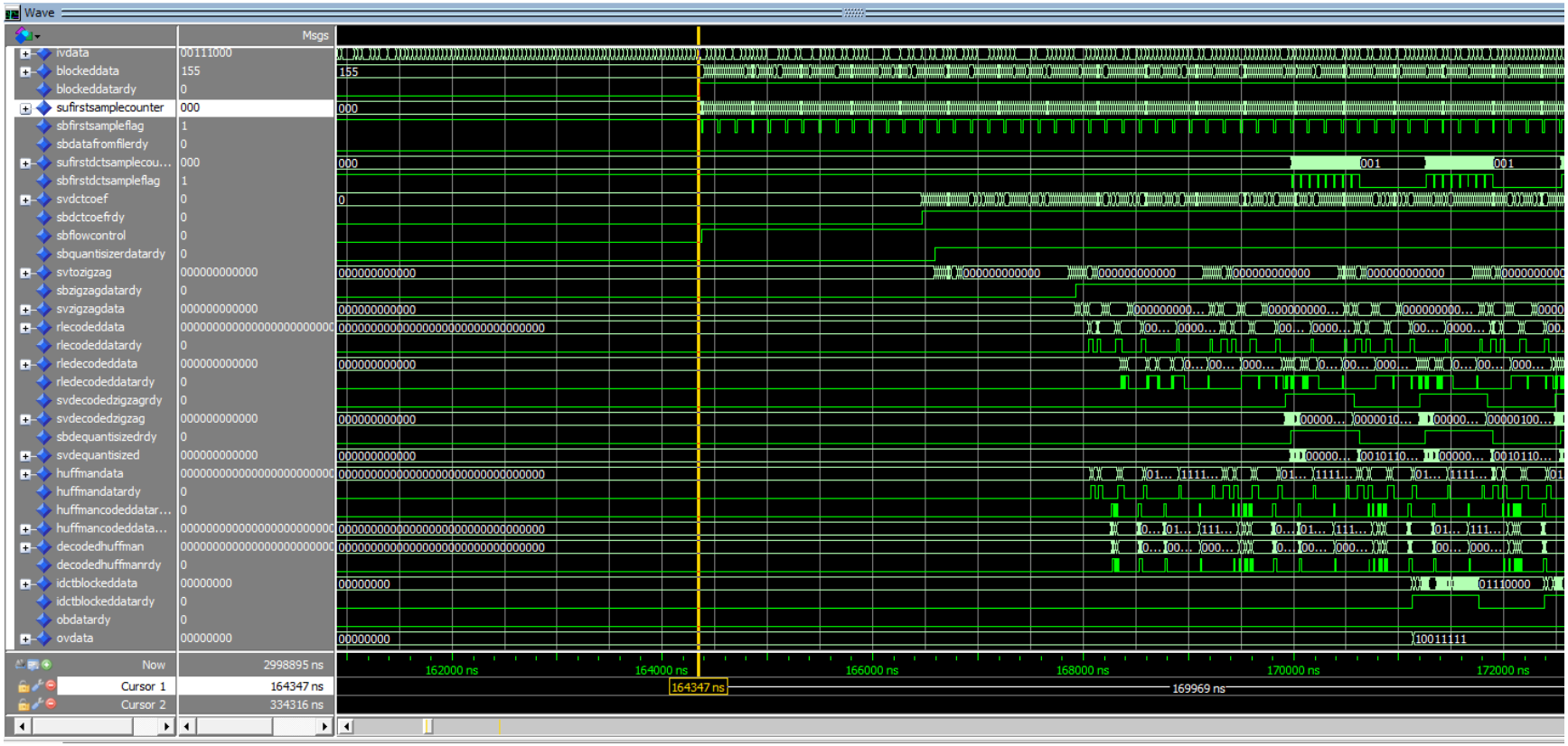

Figure C.1 - Simulation du CODEC matériel sous MODELSIM ®

La figure C.1 illustre les signaux principaux du CODEC matériel. Certains servent à la signalisation entre les divers modules décrits à la figure 3.12. Les plus importants sont décrits ci-dessous. Les signaux finissant par les lettre « rdy » sont des signaux de signalisation de type *data ready, data enable* ou *chip enable*.

- **`ivdata`**: Il s'agit du train de pixels entrant dans le CODEC. À ce point, l'image est transmise ligne par ligne.
- **`blockeddata`**: Le train de pixels entrant est converti en bloc de 64 pixels qui conservent la même position que dans l'image. Pour ce faire, il est nécessaire d'employer un accumulateur de ligne de dimension de huit lignes au minimum.
- **`svdctcoef`**: Il s'agit des coefficients sortant de la DCT2-2D en test. Ces coefficients se dirigent vers le module de quantification.
- **`svtozigzag`**: Ce signal véhicule les coefficients quantifiés vers le module de réorganisation zigzag.
- **`svzigzagdata`**: Les coefficients quantifiés et réorganisés sont acheminés par ce signal vers le module d'encodage RLE.
- **`rlecodeddata`**: Le module d'encodage RLE fonctionne à l'aide d'une machine à état car plusieurs échantillons sont considérés pour générer un symbole. Le débit des symboles sortant est donc irrégulier tel qu'il est possible de le constater.
- **`huffmandata`**: Les symboles en provenance du module RLE sont codés via un générateur de code Huffman. Comme le module d'encodage Huffman est codé via un multiplexeur, les symboles sont codés instantanément.
- **`decodedhuffman`** : Les symboles sont ici en format RLE. Ils proviennent du module de décodage Huffman.
- **`rledecodeddata`** : Les données de ce signal sont acheminées vers le module de réorganisation zigzag inverse.
- **`svdecodedzigzag`**: Les données de ce signal seront déquantifiées par le module de déquantification.
- **`svdequantisized`**: Les données déquantifiées sont envoyées vers la IDCT2-2D (DCT2-2D inverse).
- **`idctblockeddata`**: Les données sortant de la IDCT2-2D sont en blocs de 64 pixels. Il faut les réorganiser en ligne.
- **`ovdata`**: Ce signal véhicule le train de pixels de l'image décodée.

RÉFÉRENCES

Ali Khayam S., (2003). ECE 802-602 : Information Theory and Coding. *The Discrete Cosine Transform Theory and Application*, Department of Electrical & Computer Engineering, Michigan State University.

Auclair Beaudry J.S., (2009) *Modelage de contexte simplifié pour la compression basée sur la transformée en cosinus discrète*. Mémoire de maîtrise, Université de Sherbrooke, Département de génie électrique et de génie informatique, Sherbrooke, Canada, 108 p.

Ashenden P.J., (2001). *The Designer's Guide To VHDL*, 2e édition. Morgan Kaufmann, San Francisco, USA, 759 p.

Ashenden P.J., (2008). *VHDL-2008 Just the New Stuff*, 1e édition. Morgan Kaufmann, San Francisco, USA, 256p.

Bi G., Zeng Y., (2003). *Transfor*ms *and Fast Algorith*ms *for Signal Analysis and Representations*, 1ère édition. Birkhäuser Boston, New York, USA, 440 p.

Brice R., (2003). *Newnes Guide to Digital TV*, 2e édition. Newnes, San Francisco, USA, p. 304.

Britanak V., Yip P, Kamisetty R.R., (2007). *Discrete Cosine and Sine Transfor*ms, 1ère édition. Academic Press, Oxford, UK, 368 p.

CCITT, Recommendation T.81 (1993). *Digital Compression and Coding of Continuous-Tone Still Images - Part 1 : Requirements and Guidelines*, Terminal equipment and protocols for telematic services, International Telecommunication Union, Genève, Suisse, 186 p.

Chang K., (2000) *RF and Microwave Wireless Syste*ms, 1ère édition. Wiley-Interscience Publication, Hoboken, USA, 360 p.

Chen, W.H., Smith, C.H., Fralick, S.C., (1977) A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, volume 25, no 9, p. 1004–1009, septembre.

Gharge S., Shoba K., (2007) Simulation and Implementation of Discrete Cosine Transform for MPEG-4, *International Conference on Computational Intelligence and Multimedia Applications*, IEEE, Mumbai, Inde, p. 137-141, décembre.

Gukhool H.D., (2009) *Compression intra-images en temps réel et à bas débit de qualité supérieure à 30db*. Mémoire de maîtrise, Université de Sherbrooke, Département de génie électrique et de génie informatique, Sherbrooke, Canada, 112 p.

Gustavo A. H.; Eric E. F.; Diogo Z.; Sergio B., (2001). *The BinDCT Processor*, UFRGS Federal University, Microelectronics Group, Porto Alegre, Brazil, p. 8.

Hamilton E. (2009). *JPEG Ffile Interchange Format (JFIF)*, Technical Report : ECMA TR/98, ECMA Internation*Al.*, Genève, Suisse, 16 p.

Intopix, (2011) Xilinx JPEG2000 Encoder & Decoder IP cores - intoPIX, Technical Specifications, Intopix S.A., Louvain-la-Neuve, Belgique, 4 p.

Jie, L.; Tran, T.D., (2001) Fast multiplierless approximations of the DCT with the lifting scheme, *IEEE Transactions on Signal Processing*, volume 49, no 12, p. 3032-3044, décembre.

Keith J., (2005). *Video Demystified*, Newnes, 4e édition. Pearson, San Francisco, USA, 960 p.

Le Dinh C.T., (2008). GEI 754 : Traitement d'images. *Notes de cours*, Université de Sherbrooke.

Le Dinh C.T., (2009). GEI 759 : Ingénierie des systèmes numériques. *Notes de cours*, Université de Sherbrooke.

Lemieux R., Masson P., Bellemarre C., Martin M., Bisson D., Bernard M., Fauteux P., Julien C., Lalonde-Filion M., Morier C., Morin-Guimond A., Patry M.A., Saint-Pierre A., Moisan P., Michaud F. ,(2007), Telecoaching in traumatology , *Canadian Medical and Biological Engineering Conference*, Toronto, Canada, p. 4.

Marcellin M.W., Gormish M.J., Bilgin A., Boliek M.P., (2000) An overview of JPEG-2000. *Proceedings IEEE Data Compression Conference*, p. 523–541.

Pedroni V.A., (2004). *Circuit Design with VHDL*, 1ère édition. The MIT Press, Cambridge, USA, 375 p.

Pillai L., (2002). XAPP610 : Application Note. *Video Compression Using DCT*, Xilinx, San Jose, USA, 8p.

Pillai L., (2002). XAPP611 : Application Note. *Video Compression Using IDCT*, Xilinx, San Jose, USA, 8p.

Ponomarenko N.N., Egiazarian K.O., Lukin V.V., Astola J.T., (2005) DCT based high quality image compression, *Proceedings of the 14th Scandinavian Conference on Image Analysis*, Springer-Verlag Berlin, Heidelberg , Allemagne, p. 1177-1185, juin.

Ponomarenko N.N., Egiazarian K.O., Lukin V.V., Astola J.T., (2007) High quality DCT-based image compression using partition schemes, *IEEE Signal Processing Letters*, volume 14, no. 2, p.105-108, février.

Raymond K.W., Moon-Chuen L.C., (2006) Multiplierless approximation of fast DCT algorithms, *Proceedings IEEE International Conference on Multimedia and Expo*, p. 1925-1928, juillet.

Roman C.K., Shirani S., (2005) ASIC and FPGA implementations of H.264 DCT and quantization blocks, *Proceedings IEEE International Conference on Image Processing*, Genoa , Italie, p. 1020-1023, septembre.

Ruiz G.A., Michell J.A., Buron A.M., (2005) Multiplierless approximation of fast DCT algorithms, *Proceedings IEEE International Conference on Image Processing*, Genoa, Italie, p. 1036-1039, septembre.

Sayood K., (2005). *Introduction to Data Compression*, Series in Multimedia and Information Systems, 3e édition. Morgan Kaufmann, San Francisco, USA, 704 p.

Skodras A., (2000) The JPEG 2000 still image coding system: An overview. *IEEE Transactions on Consumer Electronics*, volume 46, no 4, p. 1103–1127, novembre.

Skodras A., Christopoulos C. et Touradj E., (2001) The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, volume 18, no 5, p. 36–58, septembre.

Stallings W., (2006). *Computer Organization and Architecture*, The William Stallings Books on Computer and Data Communications Technology, 7e édition. Pearson, Saddle River, USA, 778 p.

Tran T.D., (2000) The BinDCT: Fast multiplierless approximation of the DCT, *IEEE Signal Processing Letters*, volume 7, no. 6, p.141-144, juin.

Vassil D., Khan W.,(2004) Multiplierless DCT algorithm for image compression applications. *International Journal Information Theories & Applications*, volume 11, no 2, p. 162–169, mai.

Xilinx Inc., (2010) Xilinx DS202 Virtex-5 FPGA Data Sheet: DC and Switching Characteristics, Data Sheet, Xilinx Inc., San Jose, California, USA, 91 p.

Yishu W., (2005) *Implementation of Digital Filter by Using FPGA*, Mémoire de maîtrise, Curtin University of Technology, Engineering department, Sydney, Australia, 81 p.