Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Faculté des Sciences de l'Ingéniorat **Année : 2013/2014**
Département d'Electronique

THESE

Présentée en vue de l'obtention du diplôme de ***DOCTORAT***

Etude et implémentation d'un codeur vidéo sur un système embarqué de type FPGA :
Applications aux transformations orthogonales du codeur H264/AVC

Option
Signaux et Images

Par
HARIZE Saliha

Directeur de Thèse: Noureddine DOGHMANE Prof. Univ. Badji Mokhtar-Annaba

Devant le Jury

Président	:	**M. ABBASSI Hadj Ahmed**	**Pr. Univ. Annaba**
Examinateurs:		**M. FAROUKI Atef**	**Pr. Univ. Constantine**
		M. BEDDA Mouldi	**Pr. Univ. Aljouf. Arabie Saoudite**
		M. TALEB-AHMED Abdelmalik	**Pr. Univ. Valenciennes. France**
		M. CHEMALI Hamimi	**MCA . Univ. Sétif**

Dédicaces

A la mémoire de mon très cher et regretté père…

Remerciements

Mes remerciements les plus sincères vont à mon encadreur Professeur Doghmane Noureddine, pour ses conseils efficaces, ses orientations, sa patience et ses suggestions avisées qui ont permis de mener à bien ce travail, sans oublier ses qualités humaines.

Je tiens aussi à exprimer ma profonde gratitude à Dr. Benouaret Mohamed pour son aide précieuse, ses encouragements, ses discussions fructueuses ainsi qu'à la mise à ma disposition de la carte de développement DE2 d'Altera.

Mes remerciements les plus profonds et mes respects les plus sincères vont à Pr. Abassi pour l'honneur qu'il m'a fait en acceptant de présider le jury, ainsi qu'à Pr. Farouki, Pr. Bedda, Pr. Taleb Ahmed et Pr. Chemali pour avoir accepté d'examiner ce travail et de l'enrichir par leurs grands savoirs et leurs grandes expériences. J'adresse un salut particulier à Pr. Bedda et Pr. Taleb Ahmed pour avoir accepté d'être membres de ce jury malgré l'éloignement et leurs préoccupations. Qu'ils trouvent dans ces mots l'expression de mon estime la plus profonde.

Résumé :

H.264/AVC est la norme de compression vidéo la plus récente. Elle est basée sur une transformation en cosinus discrète (TCD) entière sur des blocs 4x4, et aussi sur des blocs de taille 8x8 dans le profil FRExt. Les filtres anti-blocs, filtres de type RIF sont alors introduits pour corriger le fameux problème des artefacts. Ce travail présente, d'abord, une analyse des performances du standard H.264/AVC en menant des tests sur des paramètres, dont les plus importants sont ceux relatifs aux aspects du standard H.264/AVC mentionnés ci-dessus. Les simulations sont effectuées avec la plateforme JM 18.2. Ensuite, une méthodologie d'implémentation de filtres RIF sur un circuit de type FPGA est décrite. Une implémentation de la transformée en ondelettes discrète (TOD) à une dimension et à un niveau a été également réalisée. Enfin, des implémentations de la transformée en cosinus discrète (TCD) entière 4x4, entière 8x8, et classique 8x8 sont exposées. Leurs performances sont évaluées et comparées. La description est faite en VHDL et les tests réalisés à l'aide du logiciel ModelSim-Altera 6.3g_p1. Les résultats, confrontés avec ceux obtenus avec Matlab, ont prouvé le bon fonctionnement de l'implémentation. Les performances de la réalisation sur le circuit Cyclone II EP2C35F672C6 estimées avec Quartus II 8.1 Web Edition d'Altera, ont montré un bon compromis entre la surface occupée et la consommation électrique, et surtout une grande vitesse qui répond aux exigences du traitement en temps réel.

Mots clés : Compression vidéo, H.264/AVC, filtres RIF**,** JM 18.2, TCD, TOD, VHDL, FPGA.

Abstract:

H.264/AVC is the most recent video compression standard. It is based on a discrete cosine transform (DCT) on 4x4 blocks, and also on blocks of dimension 8x8 in the FRExt profile. Therefore, deblocking filters, of FIR type, are introduced to correct the artifacts famous problem. First, this work presents a performance analysis of the H.264/AVC standard by running tests on parameters, the most important of them being those cited previously. The simulations are conducted using the JM 18.2 plateform. Then, a methodology for implementing FIR filters on FPGA is described. A 1D, one level discrete wavelet transform (DWT) implementation has also been performed. Lastly, implementations of the integer DCT 4x4, the integer DCT 8x8 and the real DCT 8x8 are explained. Their performances are evaluated and compared. The description is made in VHDL and the tests realized using ModelSim-Altera 6.3g_p1 software. The results, compared to those obtained with Matlab, have proved the correct functioning of the implementation. The realization performance on the Cyclone II EP2C35F672C6 circuit, estimated with Altera Quartus II 8.1 Web Edition, have shown a good tradeoff between the area used and the power consumption, and especially a high speed which meets the real time processing requirements.

Keywords: Video compression, H.264/AVC, FIR filters, DCT, DWT, JM 18.2, VHDL, FPGA.

ملخـص :

يمثل H.264/AVC احدث معايير ضغط الصور و الفيديو ويعتمد على تحويل جيب التمام المتقطع المطبق على كتل بقياس (4x4) و (8x8) في الجزء FREXt لهذا تستعمل المرشحات من نوع FIR لمنع ظهور هذه الكتل. في البداية يقدم هذا العمل تحليلا للنتائج القياسية للمعيار H.264/AVC من خلال القيام بتجارب على بعض المتغيرات, من أهمها المذكورة أعلاه وقد أجريت التحاليل باستعمال القاعدة JM18.2 .يتبع هذا بتقديم و شرح منهجية لغرز هذا النوع من المرشحات على مجموعات البوابات المبرمجة من نوع FPGA والتي تشكل كذلك غرزا للتحويل المويجي المتقطع ذات بعد واحد وعلى مستوى واحد وأخيرا يقدم غرز التحويل الجيبي المتقطع الصحيح بقياس (4x4) و (8x8) وكذالك التحويل الجيبي المتقطع الحقيقي بقياس (8x8) مع تقييم ومقارنة النتائج القياسية.

يتم الوصف بلغة VHDL وأجزت تجارب الغرز باستعمال البرنامج ModelSim-Altera 6.3g-p1 ,وقد أظهرت النتائج المحصل عليها والتي قورنت بنتائج البرنامج Matlab صحة وفعالية الغرز الذي جسد على الدارة Cyclone II EP2C35F672C6 قيست المعلومات القياسية بالبرنامج Altera Quartus II8.Web Edition وأكدت توازن جيد بين المساحة المستعملة ,الطاقة المستهلكة وخاصة سرعة تنفيذ تفي بمتطلبات المعالجة في الوقت الحقيقي.

الكلمات المفاتيح:

ضغط الصور و الفيديو H.264/AVC , مرشحات FIR, JM18.2, التحويل الجيبي المتقطع, التحويل المويجي المتقطع,VHDL , على مجموعات البوابات المبرمجة FPGA

LISTE DES ACRONYMES

A

ASIC: Application Specific Integrated Circuit

AVC : Advanced Video Coding

AVI : Audio Video Interlaced

ASO : Arbitrary Slice Ordering

B

B : Bi-predictive

BMA : Block Matching Algorithm

Bps : bits per second

C

CABAC : Context-Adaptive Binary Arithmetic Coding

CCIR : Comité Consultatif International des Radio Communications

CAVLC : Context-Adaptive Variable Length Coding

CD : Compact Disc

CDA: Classic Distributed Arithmetic

CIF : Common Intermediate Format

CLB: Configurable Logic Bloc

CLC=Codage à longueur Variable

CODEC : Compression/DECompression

CPLD: Complex Programmable Logic Device

CR : Compression Rate

D

DA: Distributed Arithmetic

Db2: ondelette Daubechies d'ordre 2

DCT : Discrete Cosine Transform

DHT: Discrete Hadamard Transform

DP : Data Partitioning

DVD :

DSP: Digital Signal Processor

DWT: Discrete Wavelet Transform

E

Exp-Golomb : Exponential-Golomb.

EPZS : Enhanced Predictive Zonal Search.

F

FIR : Finite Impulse Response

Flv: flash Video

FMO **:** Flexible macroblock ordering

FPGA; Field Programmable Gate Array

FRExt : Fidelity Range Extensions

G

GOB : Group Of Blocks

GOP : Group Of Pictures

H

HD : High Definition

HEVC : High Efficiency Video Coding

I

I : Intra

IEC: International Electrotechnical Commission

ITU-T: International Telecommunications Union – Telecommuniction

ISO : International Organization for Standardization

I2C: Inter Integrated Circuit

J

JPEG : Jointed Picture Expert Group

JVT : Joint Video Team

L

LUT : Look-Up Table

M

MAE : Mean Absolute Error

MB : Macro Block

MBAFF: MacroBlock-Adaptive Frame-Field

MBPS: Mega Bits Per Second

MDA: Modified Distributed Arithmetic

MISB : Motion Imagery Standards Board

MPEG : Moving Picture Expert Group

MSE : Mean Squared Error

MSPS: Mega Samples Per Second

N

NAL :Network Abstraction Layer

NTSC : National Television Standards Committee

P

P : Predictive

PAL : Phase Alternating Line

PC : Personal Computer

PicAFF/PAFF : Picture Adaptive Frame-Field

PLL: Phase Locked Loop

PPS : Picture Parameter Set

PSNR : Peak Signal to Noise Ratio

Q

QCIF : Quarter CIF

QP : Quantification Parameter

R

RAM: Random Access Memory

RD : Rate Distorsion.

RGB : Red Green Blue

RIF : Réponse Impulsionnelle Finie

RNIS : Réseau Numérique à Intégration de Services

ROM: Read Only Memory

RTL : Register Transfer Level

RTP : Rapid Transport Protocol

RS : Redondant Slices

RVB : Rouge Vert Bleu

S

SAE/SAD : Sum of Absolute Errors / Sum of Absolute Differences

SECAM : SEquentiel Couleur Avec Mémoire

SI: Switching Intra

SQCIF : Sub QCIF

SP : Switching Predictive

SPS : Sequence Parameter Set

SRAM: Static RAM

SSIM: Structural SIMilarity

SVH : Système Visuel Humain

SI : Switching Intra

SP : Switching Predictive

T

TCD : Transformée en Cosinus discrète

TCG : Temps de Codage Global

TEM : Temps d'Estimation du Mouvement

TVHD : TéléVision Haute Définition

V

VCEG: Video Coding Expert Group

VLC : Variable Length Coding

VLSI: Very Large Scale Integration

VOD : Video On Demand

W

Wmv : Windows media video

Y

YCbCr : Luminance, Chrominance blue, Chrominance rouge

YUV : Espace de couleur (idem que YCbCr)

LISTE DES FIGURES

LISTE DES TABLES

SOMMAIRE

Introduction générale:

Les domaines des télécommunications, du multimédia et de la transmission vidéo ont connu une évolution fulgurante. Ainsi, leurs performances s'améliorent chaque jour pour répondre aux exigences de plus en plus élevées en termes de qualité et temps de traitement. En effet, la quantité des informations échangées sur internet et d'autres réseaux est tellement élevée, la vidéo en représentant plus de 75%, que de nouvelles unités de mesure ont été créées. Ainsi, les données sont exprimées en "petabyte" où le préfixe "peta" est égal à 10^{15} dans le système international des unités, et de symbole PB. Une unité relative est le "pebibyte" (PiB), qui utilise un préfixe binaire et qui vaut 1024^5bytes sachant que 2^{50} bytes= 1125899906842624bytes !. Un exemple de domaine où les données sont exprimées en petabyte est le compte Hotmail qui, pour le transférer vers le nouveau compte e-mail, a nécessité le déplacement de 150 Petabytes de données d'utilisateurs de six semaines. Ceci a été annoncé par Microsoft en Mai 2013 [1] . Cette même source a publié en 2013, qu'un petabyte suffit pour le stockage de l'ADN de toute la population des états unis. La compression vidéo s'impose donc comme outil indispensable pour répondre à ces développements et les systèmes de compression sont utilisés partout dans la vie courante. On trouve les vidéos codecs numériques dans les systèmes DVD (lecteurs, enregistreurs), les CD Vidéo, les systèmes de diffusion terrestre et satellitaires, … etc.

Cependant, si l'on se réfère aux courbes (figure i.1) des lois empiriques prédictives de l'évolution des densités d'intégration (Loi de Moore [2]) et de la complexité algorithmique (Loi de Shannon) pour l'évolution des applications multimédia, on constate que l'écart entre les deux courbes croit de plus en plus au fil des années [3].

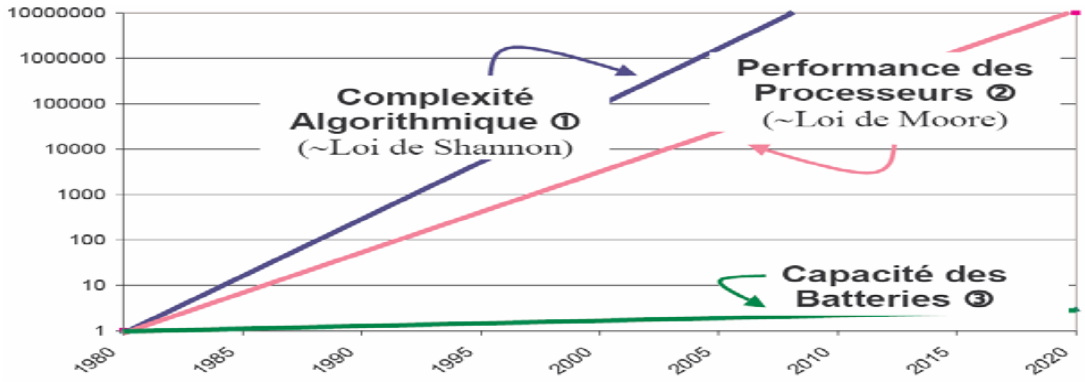


Figure i.1 Lois empiriques prédictives.

Pour faire face à toutes ces exigences, les groupes internationaux de développement et de standardisation tels que l'UIT-T, l'ISO/IEC, le MPEG et le JVT travaillent continuellement pour améliorer les performances des normes de compression, en particulier celles relatives à la compression vidéo.

Dès le début des années 1990, les standards H.261 [4], MPEG-1 [5], MPEG-2 / H.262 [6], H.263 [7], et MPEG-4 (Part 2) [8] ont commencé à apparaitre avec toujours la recherche du meilleur compromis cout-performances. Le standard ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding, beaucoup plus connu sous la désignation H.264/AVC, [9] a été jusqu'au début de l'année courante 2013, le dernier de la série et qui à cette date , reste le plus puissant. En Janvier 2013, un brouillon du High Efficiency Video Coding (HEVC) standard nommé H.265 a été annoncé. On dit que Le H.265 peut faire mieux que le H.264 en compression pour une même qualité d'images (de 35 à 67 %). La norme H.264 a été développée par le groupe JVT(Joint Video Team) qui est composé d'experts des groupes VCEG (Video Coding Experts Group) du ITU-T et MPEG (Moving Picture Experts Group) d' ISO/IEC. Le standard permet d'améliorer l'efficacité de codage par un facteur de deux par rapport au standard MPEG-2, un standard très répandu aussi. En Juillet 2004, une extension, appelée FREXt (Fidelity Range Extensions) a été rajoutée au standard . Elle a montré encore plus d'efficacité, atteignant une amélioration par un facteur de trois par rapport à MPEG-2. En Europe, l'organisation de standardisation DVB (Digital Video Broadcast) a approuvé le H.264/AVC pour la diffusion de la télévision en Europe dès la fin de 2004. Le MISB (Motion Imagery Standards Board) du département de la défense des Etats-Unis a aussi adopté H.264/AVC comme codec vidéo préféré pour toutes les applications. La norme H.264 est actuellement utilisée dans les services de vidéo à la demande (VOD) sur Internet afin de fournir des films et des émissions de télévision sur ordinateur.

Néanmoins, la mise en œuvre et l'utilisation en temps réel de ces techniques de compression vidéo exige du hardware à adopter certaines performances, à savoir :

- ✓ Une dimension (area) aussi petite que possible définie comme la quantité de ressources utilisées pour implémenter la fonction.
- ✓ Un haut débit (throughput), représentant le nombre d'échantillons traités par unité de temps.
- ✓ Une latence (latency) définie comme étant le temps nécessaire pour générer un échantillon en sortie après acquisition de l'échantillon à traiter. Celle-ci doit être aussi faible que possible.

- ✓ Une faible consommation d'énergie.

Les processeurs classiques ne peuvent plus répondre à de telles exigences. Il est donc nécessaire de trouver d'autres méthodes d'implémentation capables de supporter de telles applications complexes et de satisfaire les contraintes posées. Les circuits programmables se sont alors imposés comme substituts. Plus particulièrement les FPGAs (Fied Programmable Gate Arrays) qui commencent à dépasser les CPLDs (Complex Programmable Logic Devices), les ASICs (Application Specific Integrated Circuits) et les DSPs (Digital Signal Processors). Les FPGAs dont la technologie a été introduite par Xilinx ^{TM} en 1984 [10], présentent les avantages de distribution des ressources : flexibilité, prototypage rapide, reconfiguration, support de plusieurs niveaux de parallélisme, économie des tests de fabrication. Le prototype peut être éventuellement implémenté sur un circuit ASIC. Les PDSPs, par contre, ont une architecture fixe, nombre limité d'accumulateurs et multiplieurs, largeur des bus de données fixe et il est souvent nécessaire d'utiliser plusieurs DSP pour répondre au besoin des bandes passantes. A l'heure actuelle, on compte une dizaine de fabricants de circuits FPGA, le marché étant nettement dominé par les sociétés Altera [11] et Xilinx [12] et selon les prévisions, le marché mondial du FPGA devrait passer de 1.9 Milliard de dollars en 2005 à 5.6 milliards en 2014 [13].

A noter qu'une très bonne description de tous les circuits programmables est donnée dans [14] et elle est suivie par une comparaison.

Depuis quelque temps, les FPGAs commencent à dominer dans les applications telles que le filtrage, les transformées (FFT, DCT, DWT), la détection et correction d'erreurs, … etc. Les travaux dans le domaine de la compression d'images et de vidéos sont basés essentiellement sur l'implémentation des transformées en ondelettes discrète (DWT) [[15], [16], [17], [18], [19], [20], [21]] et en cosinus discrète (DCT) [[22], [23], [24]]. Etant donné que ces transformations consomment une bonne part du temps de traitement, estimée à 21%, alors la solution hardware s'impose, et c'est dans ce sens qu'a été orientée cette thèse visant deux objectifs :

- Le premier est de développer une méthode quasi générale pour l'implémentation des filtres à réponse impulsionnelle finie (RIF ou FIR) sur FPGA (Cyclone II EP2C35F672C6 d'Altera). L'intérêt est que le procédé peut être exploité pour l'implémentation de la DWT selon le schéma en bancs de filtres et aussi pour l'implémentation des filtres anti-blocs (deblocking filters). Ce type de traitement est un des nouveaux aspects du H.264, qui lorsqu'il est

effectué dans la boucle de codage sur les blocs 4×4, permet de réduire les artefacts caractéristiques du codage avec transformation en bloc. Les filtres RIF sont aussi mis en œuvre dans l'interpolation des échantillons utilisés en prédiction sous-pixélique.

- Le deuxième but est d'implémenter, sur FPGA, la transformée en DCT entière 4×4 préconisée dans la norme H.264/AVC et la transformée entière 8x8 permise aussi dans le profil FRExt. Les performances de ces deux modes seront comparées entre elles et avec la DCT classique 8x8.

Dans les deux cas, les performances en termes de ressources matérielles, débit, latence et consommation électrique seront évaluées et analysées et les compromis discutés.

Cette thèse est organisée en cinq chapitres :

Dans le premier chapitre, les notions de base relatives à l'image et la vidéo sont présentées. Le principe général de fonctionnement des codeurs vidéo est traité. Il est suivi par les différents organismes de standardisation et les principales normes de compression (les MPEGx et les H.26x) .

Le deuxième chapitre présente les fondements de base de la norme H.264/AVC où l'ensemble de ses aspects sont détaillés. Il inclut les différents profils et niveaux, les modes de codage (I, P et B), les algorithmes de codage CABAC et CAVLC et le filtrage anti blocs. Par contre, les transformées en cosinus discrète, classique et entière sur des blocs 4x4 sont traitées au troisième chapitre. Dans ce dernier, la DWT est aussi développée.

Le chapitre IV est consacré aux simulations menées sur la plateforme JM 18.2 afin d'analyser les performances du standard H.264. Différents paramètres tels que le format (CIF, QCIF), les dimensions du frame, le nombre de frames de la séquence à coder, le pas de quantification, … etc., sont étudiés et leurs effets sur la qualité, le débit et le temps de codage sont mesurés. Les paramètres relatifs aux deux aspects qui nous intéressent, à savoir les différents modes de transformation spatiale et le filtrage anti-blocs, sont intensément testés. Les résultats sont analysés et discutés.

Le dernier chapitre est un rapport détaillé sur la méthodologie développée pour implémenter les filtres de type RIF et les différentes transformées en cosinus discrète sur un circuit de type FPGA. Les techniques appelées arithmétique distribuée et coefficients distribués sont expliquées. Enfin, les résultats et les performances obtenues en utilisant Modelsim et Quartus II sont examinés et critiqués.

Chapitre I : La compression vidéo

I.1 Introduction :

L'explosion actuelle du multimédia a dépassé toutes les prévisions. Personne ne peut se passer, au quotidien, du téléphone, de la télévision et d'internet. Ceci a entrainé une consommation gigantesque d'informations de tout type (documents, musique, images et vidéos). Cette dernière est la plus gourmande en ressources aussi bien en transmission qu'en stockage à un point où la compression est devenue une activité grand publique. Les vidéos on-line sont codées avec une variété de codecs, ce qui a conduit à la disponibilité de codec packs. Il s'agit d'un assortiment pré-assemblé de codecs communément utilisés combinés en un et utilisés dans les PC. Les codeurs vidéo utilisent, en général, des formats de compression standards. Par exemple, les vidéos créées avec le codeur standard MPEG-4 Part 2 tel que Xvid peuvent être décodées (lues) en utilisant n'importe quel autre codeur standard MPEG-4 Part 2 tel que FFmpeg MPEG-4 ou DivX Pro. Chaque codeur possède des qualités mais aussi des défauts. Des comparaisons sont fréquemment publiées. La balance entre la puissance de compression, la vitesse ou la rapidité, et la fidélité ou la qualité (y compris les artefacts) est souvent considérée comme le critère le plus important du mérite technique. Ainsi, il existe plusieurs normes de compression, en évolution continue pour répondre aux exigences des progrès technologiques fulgurants. Dans ce chapitre, les notions de base relatives à l'image et la vidéo sont présentées. Le principe général de fonctionnement des codeurs vidéo est donc traité, suivi par une présentation des différents organismes de standardisation et les principales normes de compression (les MPEGx et les H.26x).

I.2 Notions de base :

I.2.1 L'image numérique et sa représentation :[25]

L'image peut être définie comme une fonction à 2 dimensions f(x, y) où x et y sont les coordonnées spatiales. L'amplitude de f en un point (x, y) représente l'intensité de l'image en ce point. Une image peut être continue par rapport aux coordonnées x, y et aussi en amplitude. La conversion d'une telle image sous forme numérique passe par la discrétisation simultanée des coordonnées et de l'amplitude. Il s'agit des opérations d'échantillonnage et de quantification. Ainsi, on obtient une matrice de données

repérables par leurs positions (x, y). Chaque donnée (nombre) représente la caractéristique du point élémentaire de l'image appelé pixel.

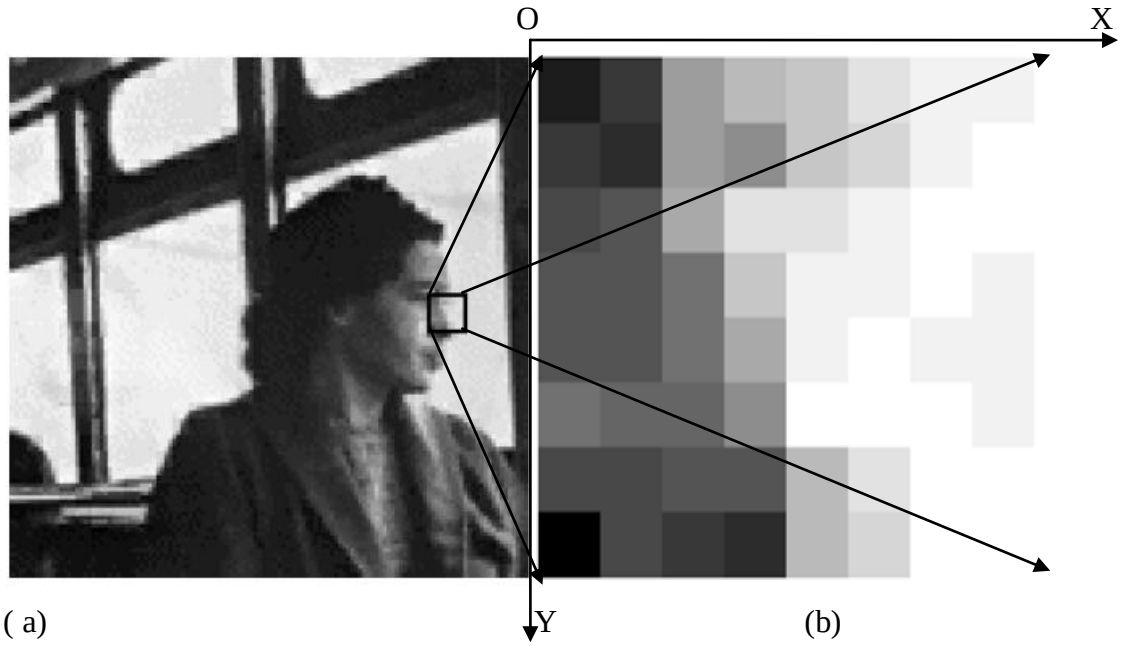


Figure I.1 L'image (a) Rosa Park, (b) un bloc 8x8 de pixels

Le codage d'un pixel dépend du type d'image et il en existe trois :

- ✓ Les images à deux niveaux : une image en noir et blanc est l'exemple le plus courant et un bit suffira pour coder la valeur d'un pixel.
- ✓ Le terme niveaux de gris est utilisé pour désigner des images monochromes telles que les images des téléviseurs en niveaux de gris. La valeur d'un pixel est codée sur 8 bits et donc 256 niveaux de gris peuvent être définis, mais seuls 128 niveaux de gris sont détectables par l'œil humain.
- ✓ Les images couleurs : formées par combinaison de composantes individuelles et où la couleur peut être codée, soit par composition de couleurs primaires RVB (Rouge-Vert-Bleu ou ″RGB″ en anglais), soit par composition d'informations de luminance et de chrominance. La valeur du pixel doit représenter les composantes trichromatiques de la couleur. 24 bits sont alors nécessaires pour coder la valeur d'un pixel (8 bits par composante) et par conséquent plus de 16 millions de couleurs peuvent être définis. Seulement, des résultats expérimentaux ont montré que l'œil est beaucoup plus sensible aux variations de l'intensité lumineuse (luminance) qu'à celles de la couleur (chrominance). Le résultat est qu'il est possible de transmettre l'information de couleur avec moins de détails que l'information de luminance.

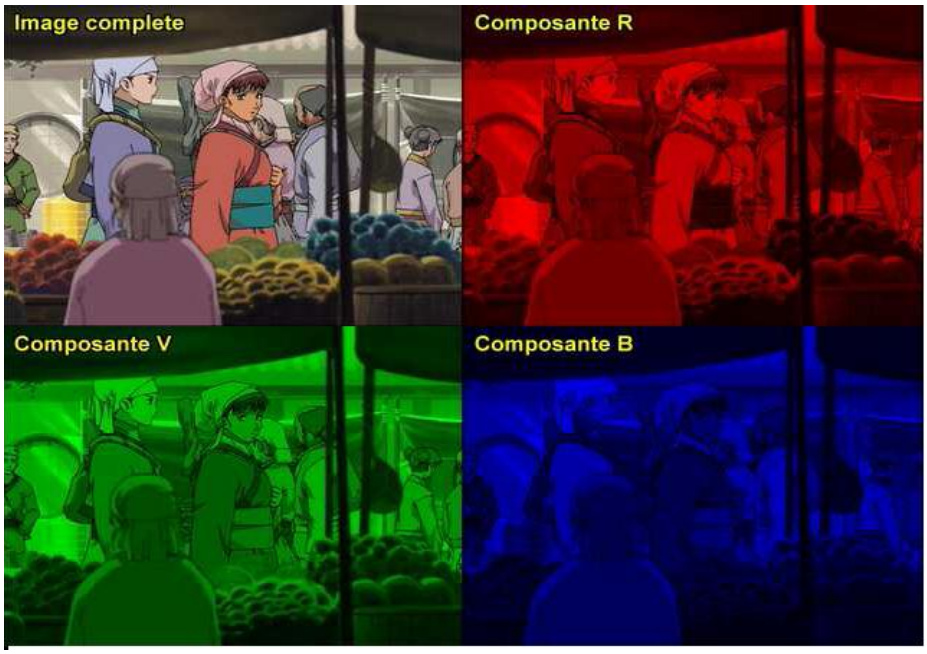


Figure I.2. Décomposition de l'image en composantes: rouge, vert et bleu

I.2.2 Les espaces de couleur :[26]

Toute longueur d'onde visible peut être visuellement simulée en convoluant le signal avec les fonctions de sensibilité des trois différents capteurs rétiniens du système visuel humain dit LMS (Large dit rouge, Medium dit vert, Short dit bleu). Le tableau I.1 présente le spectre des couleurs et les longueurs d'ondes correspondantes.

Aperçu	Couleur	Intervalle de longueur d'onde(nm)
	UV	10-380
	Violet	380-450
	Bleu	450-495
	Vert	495-570
	Jaune	570-590
	Orange	590-620
	Rouge	620-750
	IR proche	750-3000

Table I.1 Spectre des couleurs et longueurs d'onde correspondantes

Dans le cas d'une compression avec pertes, la reconstruction de chaque bande (RVB) risque de ne pas appréhender les structures de l'image de la même façon, engendrant différentes erreurs de reconstruction et par la même, de fausses couleurs visuellement choquantes. On préféra donc un espace couleur composé de luminance et chrominance rouge et bleu YCbCr (ou YUV) où les primaires sont décorrélées, ce qui offre l'avantage de séparer les informations d'intensité lumineuse et de couleur. Il existe deux

matrices de passage de l'espace RVB à l'espace YUV spécifiées par les normes CCIR 656 et CCIR 601. La matrice suivante est celle définie par CCIR 601:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.229 & 0.587 & 0.114 \\ 0.500 & -0.419 & -0.081 \\ -0.169 & -0.331 & 0.500 \end{bmatrix} \text{x} \begin{bmatrix} R \\ V \\ B \end{bmatrix} \quad (1)$$

La figure suivante est le résultat de la transformation de l'image complète de la figure I.2 en une composante de luminance (Y) et deux composantes de chrominance (U et V).

Figure I.3. Décomposition de l'image I.2 en composantes: luminance(Y) et chrominance (U et V)

I.2.3 Définition de la vidéo :

La vidéo est une succession d'images animée défilant à une certaine vitesse. La vidéo est basée sur le principe fondamental de la rétention de l'œil humain, pendant un certain temps (de l'ordre du dixième de seconde), de toute image imprimée sur la rétine. On distingue deux grandes familles de systèmes vidéo : les systèmes vidéo analogiques et les systèmes vidéo numériques.

I.2.3.1 La vidéo analogique :

La caméra balaye l'image bidimensionnelle qu'elle a devant elle par un faisceau d'électrons qui se déplace très rapidement de gauche à droite et plus lentement de haut en bas et produit une tension en fonction du temps. Elle enregistre ainsi l'intensité lumineuse, et à la fin du balayage, on a alors une trame. Le faisceau revient à l'origine pour recommencer. Le récepteur va recevoir cette intensité en fonction du temps, et pour reconstruire l'image, va répéter le processus de balayage. Il existe deux systèmes selon les paramètres du balayage:

- ✓ Le système PAL/SECAM (*Phase Alternating Line / SEquentiel Couleur Avec Mémoire)* : utilisé en Europe, ce système utilise 625 lignes (seulement 576 sont

affichées), un rapport vertical/horizontal de 4/3 et un débit de 25 images par seconde.

- ✓ Le système NTSC (*National Television Standards Committee*) : Utilisé au Japon et en Amérique, ce système utilise seulement 525 lignes (483 affichées) et un débit de 30 images par seconde (figure I.4).

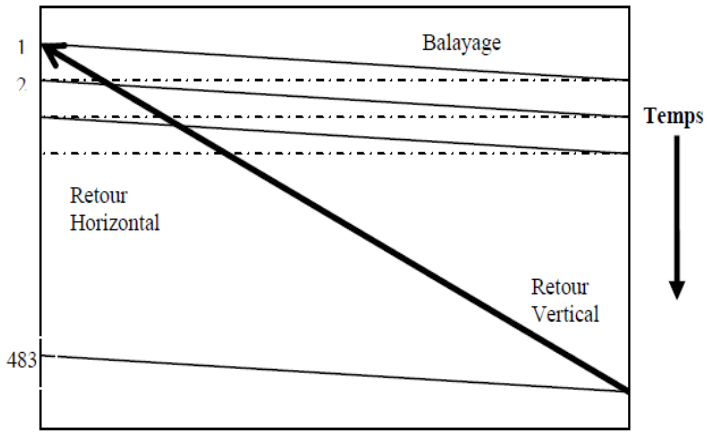


Figure I.4 Principe de balayage utilisé pour la vidéo et la télévision

La télévision couleur exploite ce même principe de balayage mais en utilisant trois faisceaux au lieu d'un, un par couleur primaire : rouge, vert et bleu (RVB). Les signaux RVB sont ensuite transformés en un signal de luminance et deux signaux de chrominance. La télévision haute définition (TVHD) utilise le même principe mais double le nombre de lignes, pour obtenir une meilleure qualité. En outre, elle utilise un format 16/9 au lieu de 4/3 et ceci pour mieux s'adapter au format des films de cinéma [27]. La figure I.5 représente le signal vidéo monochrome alors que la figure I.6 montre le signal vidéo couleur.

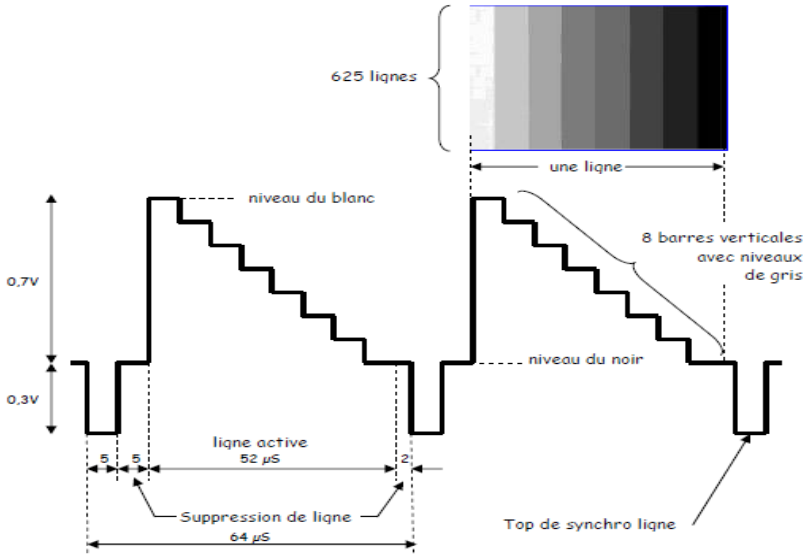


Figure I.5 Le signal vidéo monochrome

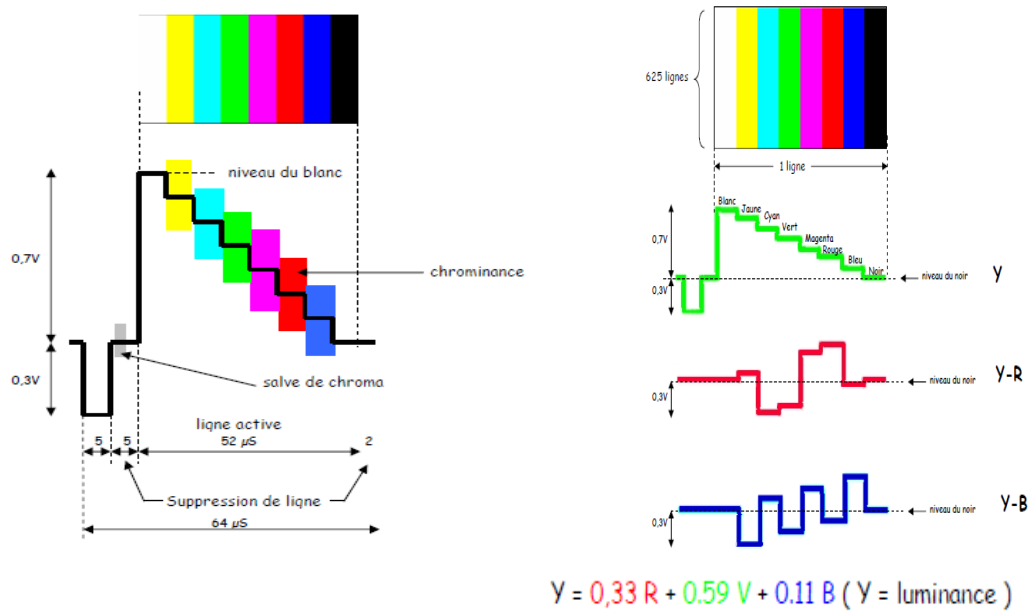


Figure I.6 Le signal vidéo couleur

I.2.3.2 La vidéo numérique :[26]

Le principe de balayage utilisé est similaire à celui de la vidéo analogique. La vidéo numérique est tout simplement une suite d'images, chacune formée d'une matrice de pixels. Pour la vidéo numérique couleur, on utilise 8 bits pour chaque couleur RVB, soit donc 24 bits par pixel, ce qui correspond à environ 16,8 millions de couleurs.

I.2.3.3 La vidéo progressive et la vidéo entrelacée :

Les vidéos entrelacées permettent d'atteindre un taux de rafraichissement élevé, sans augmenter la bande passante. Plutôt que de rafraîchir toute l'image, on rafraîchit les lignes paires puis on rafraîchit les lignes impaires à la prochaine image. Cette technique permet de capturer des mouvements plus rapides. Par contre, cette technique entraîne un artefact appelé peigne d'entrelacement. Cet artefact peut être corrigé par les logiciels de traitement et de montage vidéo (dés-entrelacement).

Les vidéos progressives rafraîchissent l'image entière. Ceci permet d'éviter les artefacts d'entrelacement, mais nécessite deux fois plus d'information qu'une vidéo entrelacée.

Il existe plusieurs standards de vidéo haute-définition (HD) entrelacée et progressive désignées respectivement par la lettre i (interlaced) ou p:

720p, 720i, 1080p ou 1080i

Le chiffre correspond au nombre de lignes verticales. Le nombre de lignes horizontales peut être calculé en sachant que le ratio d'aspect d'une vidéo HD est de 16:9.

• 720 / 9 * 16 = 1280 d'où un format 1280 x 720

• 1080 / 9 * 16 = 1920 et on a alors 1920 x 1080

Vidéo entrelacée Vidéo progressive

Figure I.7 vidéo entrelacée et vidéo progressive

I.2.4Définition du codeur :

Un codeur ou CODEC (COmpression/DECompression) vidéo est un outil pouvant être logiciel (software), matériel ou une combinaison des deux permettant la compression et/ou la décompression des vidéos numériques. La compression est souvent avec pertes. Les codeurs vidéo utilisent, en général, des formats de compression standards. Une large variété de formats de compression vidéo peut être implémentée sur les PCs ou dans des équipements électroniques grands publics. Il est alors possible de trouver plusieurs codeurs dans un même produit, ce qui permet d'éviter d'avoir à choisir un seul format de compression pour des raisons de compatibilité.

Il existe plusieurs codecs vidéo:

- H.264
- x264
- MPEG-2
- MPEG-4

codecs dérivés des standards MPEGs

- DivX
- Xvid
- DV
- WMV

• Apple Pro Res

• MJPEG

• JPEG 2000

Il existe plusieurs formats vidéo reconnaissables par l'extension du fichier. Parmi les plus courants, on trouve:

• Quicktime (.mov)

• Windows Media Video (.wmv)

• AVI (Audio Video Interlaced) (.avi)

• MP4

• Flash (.flv)

Un fichier vidéo est donc défini par son format vidéo et son codec vidéo. Pour qu'un logiciel soit capable de lire ou écrire un fichier vidéo, il doit supporter le format vidéo, le codec vidéo et le codec audio. Par exemple, Windows Media Player peut lire un fichier *Quicktime* qui utilise un codec *h.264*, mais ne peut pas lire un fichier *Quicktime* qui utilise un codec *Apple Intermediate codec*.

I.3 La compression vidéo :

La compression vidéo est un processus qui vise à réduire la quantité d'information représentant une vidéo afin d'atteindre un taux binaire mesuré en bits par seconde (bits/s ou bps). Ce taux est déterminé par la largeur de la bande passante (dans le cas d'une transmission), et par la capacité de stockage (dans le cas d'un archivage). La contrainte majeure à cette réduction est la qualité de la vidéo qui doit satisfaire certaines exigences et la complexité du traitement impliqué dans l'opération.

Considérons, par exemple, que le modem d'un réseau téléphonique peut fonctionner au taux maximal de 56.600 bps. Si le taux de transmission d'une vidéo est de 30 images par seconde, que l'image a une résolution de 288x352 (288 lignes et 352 pixels par ligne), et que 8 bits sont utilisés pour représenter un pixel de chacune des couleurs primaires (RVB), alors le taux binaire nécessaire est : 288 x 352 x 8 x 3 x 30 = 72.990.720 bps. Ceci implique que les données vidéo en question doivent être compressées, au moins 1289 fois pour accomplir la transmission.

La qualité exigée de la vidéo reconstruite dépend de l'application. Dans le cas de diagnostics médicaux et certains domaines scientifiques basés sur des mesures et des programmes, la vidéo reconstruite doit refléter la vidéo originale. Dans ce cas, seuls les schémas réversibles sont permis et on parle de compression sans pertes (lossless). Les

taux de compression, dans ce cas, restent très faibles, de l'ordre de 2 :1. Dans des applications telles que la télévision, la visiophonie ou la téléconférence, une perte relative d'information est tolérée et ce type de compression est dit avec pertes (lossy).

La compression des données vidéo numériques n'est possible que si les redondances qui existent dans une vidéo sont éliminées et il en existe essentiellement trois types :

• La redondance spatiale : corrélation entre les pixels voisins.

• La redondance temporelle : corrélation entre les images successives de la vidéo.

• La redondance psycho-visuelle : propriétés du système visuel humain.

I.3.1 Redondance spatiale :

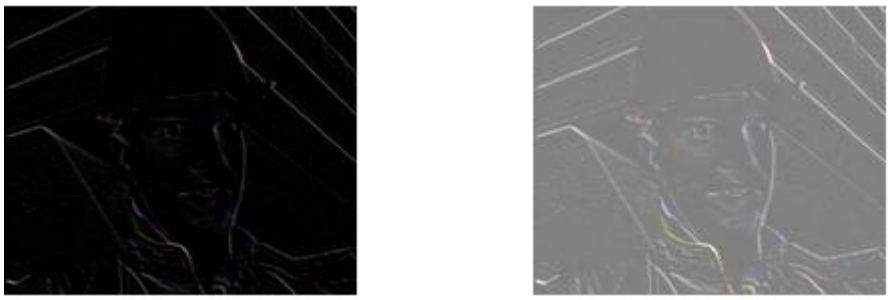
C'est la corrélation statistique entre les pixels d'une image appelée redondance intra frame. Chaque image, prise indépendamment des autres, présente des zones uniformes plus ou moins grandes dans lesquelles les valeurs des intensités des pixels sont très voisines. On peut diminuer cette redondance en codant chaque image séparément. Ce type de codage est appelé codage INTRA. L'utilité de ce codage est de pouvoir accéder de façon aléatoire à chaque image individuellement.

I.3.2 Redondance temporelle :

Cette redondance est une corrélation statistique entre les pixels d'images successives dans une séquence vidéo. L'intervalle de temps entre la prise d'images est très court et donc la similarité sera forte. Deux images qui se suivent dans une séquence vidéo sont quasiment identiques sauf dans le cas d'un changement de plan. Le but est alors de ne stocker que la différence, c.à.d. ce qui a été modifié entre deux images successives. Ce type de codage est appelé codage INTER. La redondance des données, est illustrée dans la figure I.8. Dans les trois images I.8(a), I.8(b) et I.8(c), on remarque qu'il n'y a pas une grande différence puisque la camera n'a changé ni de plan ni de position. Donc, la seule différence se trouve dans les parties qui ont subi un mouvement. Ceci peut être démontré par la figure I.8(g) qui représente la différence entre les deux images successives (figure I.8(a) et figure I.8(b)) et la figure I.8(h) qui n'est autre que l'image de la figure I.8(g) à laquelle on a ajouté 128 afin de mieux visualiser les redondances temporelles. La partie grise de cette image correspond à la similitude des deux images (a) et (b). Par contre, les trois images I.8(d), I.8(e) et I.8(f), correspondent à des frames capturés après changement de plan.

(a) Frame 51 (b) frame 52 (c) frame 53

(d) Frame 139 (e) frame 140 (f) frame 141



(g) (h)

Figure I.8 La redondance temporelle dans la séquence vidéo "Foreman".

I.3.3 Redondance psycho-visuelle :

En plus d'éliminer les redondances temporelles et spatiales, la redondance psycho-visuelle est généralement réduite elle aussi. Le signal luminance (luma Y) est beaucoup plus important perceptuellement que le signal couleur ou chrominance (chroma) qui peut donc être représenté à une résolution plus faible. C'est dans ce but que la première étape d'une compression vidéo est le changement du plan couleur (RVB vers YUV ou YCbCr). Les codeurs vidéo utilisent différents taux de sous-échantillonnage de la couleur selon les besoins de la compression. Les algorithmes de compression vidéo pour le Web et les DVD utilisent le schéma 4:2:0. Les codecs vidéo professionnels conçus pour fonctionner à des débits très élevés et enregistrer une grande quantité d'information sur les couleurs pour des manipulations postproduction échantillonnent à

3:1:1 (rarement), 4:2:2 and 4:4:4. Exemples de ces codecs sont Panasonic DVCPRO50 et DVCPROHD codecs (4:2:2), aussi HDCAM-SR de Sony (4:4:4) ou le Panasonic HDD5 (4:2:2). Le codec Prores HQ 422 d'Apple échantillonne aussi selon le format 4:2:2.

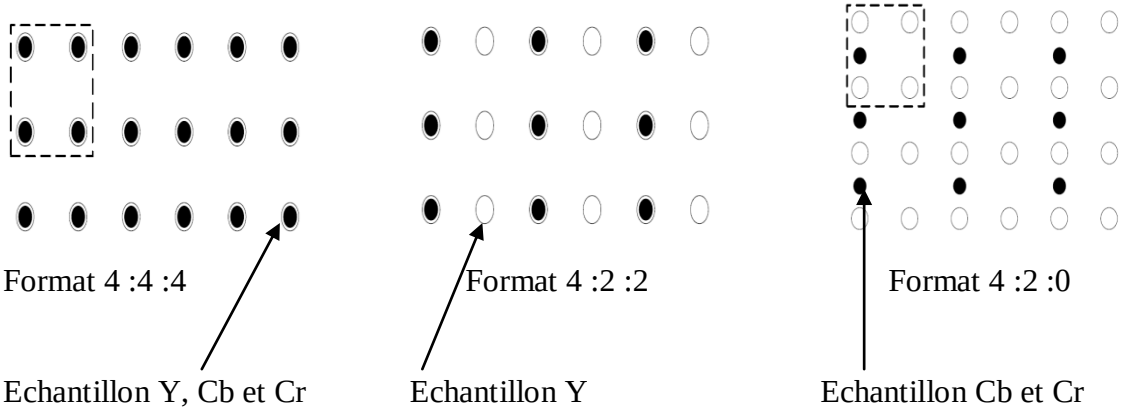


Figure I.9 Les formats d'échantillonnage.

Avec l'échantillonnage 4:4:4 les trois composants (Y, Cb and Cr) ont exactement la même résolution et donc un échantillon de chaque composant existe à chaque position des pixels. Pour quatre échantillons luminance, il y a quatre échantillons Cb et quatre échantillons Cr.

Dans l'échantillonnage 4:2:2, les composants chrominance ont la même résolution verticale que la luminance, mais la moitié de la résolution horizontale. Le format 4:2:2 est utilisé pour la reproduction haute qualité des couleurs.

Dans le très connu format d'échantillonnage 4:2:0,Cb et Cr ont la moitié de la résolution verticale et la moitié de la résolution horizontale de Y.

La table suivante résume ces formats sous forme de pourcentage des composantes de chrominance par rapport à la composante luminance.

Format	Résolution verticale (%)	Résolution horizontale (%)
4 :4 :4	100	100
4 :2 :2	100	50
4 :2 :0	50	50

Table I.2 Pourcentage des composantes de la chrominance par rapport à la luminance.

Avec le format 4:4:4, un total de 12 échantillons est nécessaire (quatre échantillons par composant) résultant en 12 x 8 = 96 bits et une moyenne de 96/4 = 24 bits par pixel.

En utilisant l'échantillonnage 4:2:0, seuls six échantillons sont nécessaires (quatre pour Y, un pour Cb et un pour Cr) résultant en un total de 6 x 8= 48 bits et une moyenne de 48/4= 12 bits par pixel.

I.3.4 Les formats vidéo: [28]

CIF (Common Intermediate Format) est le format de base pour tout un ensemble de format vidéo. Ces formats correspondent aux tailles des images composant la vidéo et sont résumés dans la table I.3. Le choix du format dépend de l'application et de la capacité de stockage ou de la capacité de transmission. Par exemple, le format 4CIF convient pour la télévision standard et la vidéo sur DVD ; les formats CIF et QCIF sont très exploités en vidéoconférences; QCIF et SQCIF (sub-QCIF) sont très utilisés dans les applications multimédia mobiles où la résolution de l'affichage et le taux binaire sont limités.

La table I.3 donne aussi le nombre de bits requis pour représenter une image (frame) non compressée pour chaque format en supposant que l'échantillonnage est le 4 :2 :0 et que 8 bits sont utilisés pour la luminance et la chrominance.

Format de l'image	Résolution de la luminance (horiz x vert)	Résolution de la Chrominance (horiz x vert)	Bits par frame (4:2:0, 8 bits par échantillon)
sub-QCIF	128 x 96	64 x 48	147.456
QCIF	176 x 144	88 x 72	304.128
CIF	352 x 288	176 x 144	1.216.512
4CIF	704 x 576	352 x 288	4.866.048
16CIF	1408 x 1152	704 x 576	19.464.192

Table I.3 Les formats vidéo.

I.3.5 Schéma de base d'une compression vidéo :

Quel que soit le type de codeur (voix, musique, image fixe, vidéo), un certain nombre d'outils communs permettent de réaliser la compression du signal. Ces outils, brièvement décrits ci-dessous, sont largement détaillés dans [29].

La prédiction : ce mécanisme permet d'établir un ensemble de valeurs prédites sur les échantillons du signal d'entrée. Généralement, cette estimation est basée sur les informations des échantillons déjà encodés du signal. La différence entre les

échantillons du signal original et leur prédiction permet d'obtenir les résidus de prédiction qui sont plus efficaces à compresser.

La transformation : Ce procédé consiste à projeter les résidus de prédiction dans une base permettant de réduire la corrélation statistique entre les échantillons des résidus de prédiction. Cette opération permet de rassembler l'énergie du signal sur un faible nombre d'échantillons. Dans cette catégorie, la transformée en cosinus discrète TCD (*Discrete Cosine Transform* ou DCT) est de loin la plus utilisée à la fois pour sa simplicité algorithmique et ses performances.

La quantification : Ce procédé permet d'approximer les échantillons du signal transformé afin de réduire la quantité d'information à transmettre. Souvent, la précision de cette approximation est contrôlée à l'aide d'un pas représentant la plus petite valeur absolue représentable, mais également l'incrément entre deux valeurs successives à la sortie du quantificateur. Parmi tous les outils de compression, la quantification est la seule opération irréversible, induisant des pertes d'information. L'objectif de tout codeur est de minimiser la perte d'information résultante sous une contrainte de débit en sortie du codeur.

Le codage entropique : Ce mécanisme permet de représenter efficacement les symboles issus du quantificateur en prenant en compte leur fréquence d'apparition. Ainsi, un mot de code de longueur variable CLV (*Variable Length Coding* ou VLC) est associé à chaque symbole en fonction de la statistique de la source. Les mots de code les plus courts sont attribués aux symboles fréquents, et inversement, des mots de code plus longs sont réservés aux symboles les moins probables. Les codeurs utilisent souvent ce procédé pour encoder les paramètres de compression, les plus connus étant le codage de Huffman et le codage arithmétique.

Trois modes de codage sont possibles et conduisent à distinguer trois types d'image :

• Les images INTRA (key frames), dites de type I : Ces images sont codées intégralement, sans aucune référence aux images voisines de la séquence vidéo. C'est la redondance spatiale qui est exploitée et éliminée à l'aide de la TCD. Cette opération est réalisée après changement du plan de couleur de l'image du RVB vers YCbCr ou YUV. Chacun de ces plans est décomposé en blocs de 8x8 pixels et transformés en matrices de coefficients fréquentiels classés selon l'amplitude des signaux. Ces coefficients sont ensuite quantifiés et codés en utilisant un codage entropique comme montre la figure I.10. Les images I sont les plus coûteuses en débit mais servent de

point de référence dans une séquence vidéo. Chaque changement de plan dans une séquence vidéo commence obligatoirement par une image de type I.

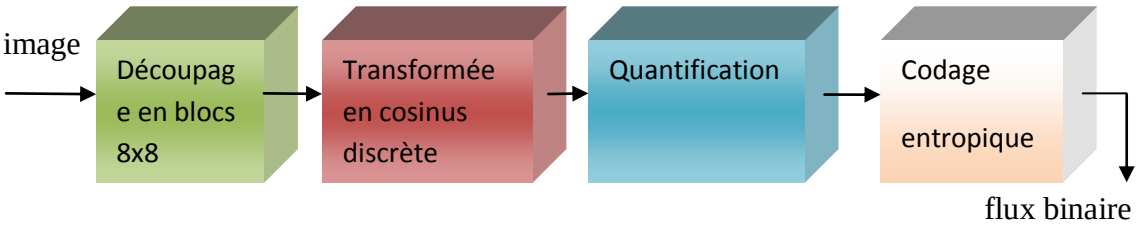


Figure I.10 Schéma du codage INTRA

• Les images INTER prédites ou à compensation de mouvement, dites de type P : Les images « Prédictives » exploitent à la fois la redondance spatiale et la redondance temporelle des images d'une séquence vidéo. Elles sont codées à partir de l'image « I » ou « P » précédente à l'aide de vecteurs de mouvement obtenus par estimation de mouvement. Les images sont découpées en blocs de 16 x 16 pixels. Les vecteurs de mouvement sont ensuite calculés en fonction du déplacement de chacun de ces blocs de pixels d'une image à la suivante puis codés avec un Codage à Longueur Variable (CLV).

• Les images INTER prédites bidirectionnelles, dites de type B : elles ne sont pas utilisées par tous les codeurs. Elles sont codées avec une estimation du mouvement par rapport à une image précédente et une image suivante. Elles ne servent jamais de référence. Les images de type I et P servent de référence aux images P et B, les images de type B ne servent jamais de référence. Le codage d'une image P et B est illustré par la figure I.11.

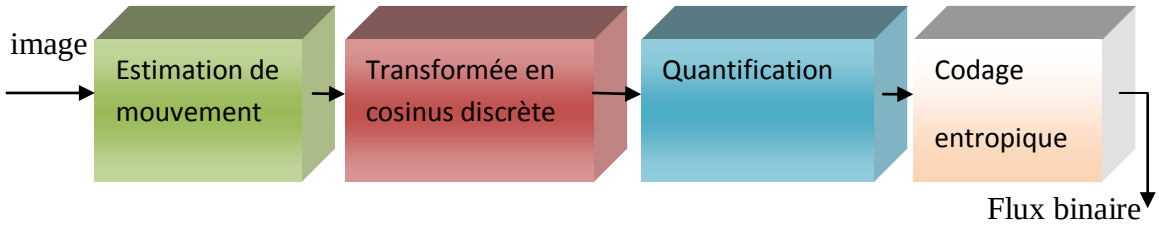


Figure I.11 Schéma du codage INTER

I.3.6 Décomposition d'une séquence vidéo :

Les données d'une vidéo sont généralement structurées en couches selon le schéma suivant :

Séquence **{**Groupe d'images (GOP)**{**Image (frame)**{**Groupe de blocs (GOB)**{**Macrobloc (MB)**{**Bloc**{**sous-bloc**}}}}}**.

Un GOP (Group Of Pictures) est une suite d'images codées suivant trois modes : codage intra image (I-frame), prédictif (P-frame) et bidirectionnel (B-frame). L'ordre de ces codages est fixe, avec une image I en début de GOP, puis un motif répétitif d'images B et d'images P jusqu'à la fin du GOP tel qu'indiqué par la figure I.12. La fréquence des images I est variable, elle donne la taille du GOP. Les images I sont indispensables pour que le décodeur puisse commencer la chaîne de prédiction.

Ordre de codage/décodage : 1423756

Ordre d'affichage (lecture) : 1234567

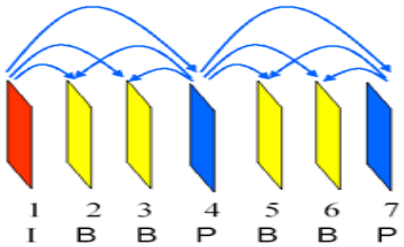


Figure I.12 Ensemble d'images (Groupe of Pictures)

I.4 Les organismes de standardisation des normes de compression vidéo :

Dans le domaine de la compression vidéo, il existe plusieurs organismes internationaux composés de chercheurs universitaires et d'industriels et dont le but principal est de développer des standards de compression vidéo aussi performants que possible pour répondre aux besoins croissants du multimédia. Ce sont :

- ITU-T VCEG (*International Telecommunications Union – Telecommuniction Video Coding Expert Group)* : organisme qui a développé la série de normes de compression vidéo telles que H261, H263.
- ISO/IEC (*International Organization for Standardization/International Electrotechnical Commission)*: organisme international dont le groupe le plus connu est le MPEG (Moving Experts Group). Fondé en 1988 pour développer des standards de compression vidéo, ce groupe a élaboré les standards MPEG1, MPEG2 et MPEG4.

- JVT (Joint Video Team) qui n'est autre que l'association des deux premiers groupes qui a crée le fameux H264/AVC. A noter qu'il existe six appellations différentes de cette norme : H.264, H.26L, ISO/IEC 14496-10, JVT, MPEG-4 AVC and MPEG-4 Part 10.

I.5 Les normes de compression vidéo : [30, 31, 32, 33]

I.5.1 MPEG1: [34]

Finalisé en 1992, MPEG1 est le premier standard du groupe MPEG permettant compresser une vidéo numérique. Le résultat est plus de 5 publications, qui forment le MPEG1. En 1993, les trois premières parties furent acceptées par l'ISO, et traitent des flux binaires de la vidéo (Part1), de l'audio (Part2) et de leur multiplexage (Part3). La partie 4 (1995) décrit une plateforme de tests pour vérifier la compatibilité sur tous les supports, et la partie 5 (1998) est une implémentation de référence des algorithmes.
Le MPEG1 utilise une résolution de 352 pixels par 240 pixels, soit 84.480 pixels. Son débit binaire typique est de l'ordre de 1.5Mbps.

I.5.2 MPEG2: [35]

Les limitations du MPEG1 furent vite apparentes. Sa résolution limitée et sa qualité assez basse n'en faisait pas un standard capable de durer longtemps face à l'évolution rapide des moyens informatiques et numériques. Le MPEG2 apparut alors et a été finalisé en novembre 1994 en introduisant un large éventail de choix en ce qui concerne la résolution et le contrôle du débit binaire. Initialement élaboré pour la transmission de vidéo de qualité TV ayant un débit compris entre 4 et 9Mb/s. Le standard permet donc désormais la compression progressive ou entrelacée de vidéo selon des débits allant de 1.5Mb/s à 100Mb/s.

I.5.3 MPEG3 :

Le MPEG3 était à l' origine destinée aux très hauts débits mais ne vit pas le jour en tant qu'entité puisqu'il fut assimilé au standard MPEG2.

I.5.4 MPEG4 : [36]

Dès 1995, le standard MPEG4 commença à émerger du point de vue théorique avec pour but la réalisation d'un débit inférieur à 64 kbits/s. Le premier document officiel le concernant fût écrit au début de 1998, c'est l'ISO 14496. Le but duMPEG4 est

d'englober les codecs MPEG existants et de leur rajouter une nouvelle dimension permettant un standard beaucoup plus flexible et beaucoup plus performant. La norme permet le codage d'une grande variété de format vidéo (taille, résolution, fréquence image) mais aussi le codage d'objets vidéo de forme arbitraire, d'images fixes ainsi que d'objets synthétiques 3D. De ce fait, cette norme adresse une large gamme d'applications audiovisuelles allant de la visioconférence à la production audiovisuelle en passant par le streaming sur Internet. MPEG-4 Visual (Part 2) a été finalisé en 1999. MPEG-4 AVC (Advanced Video Coding ou Part 10) a été développé en collaboration avec ITU-T en tant que recommandations H.264.

I.5.5 MPEG7 et MPEG21 :

Le MPEG7 est le dernier né de la famille des codecs MPEG, et a reçu sa première esquisse officielle en Septembre 2000. Le MPEG7 ne concerne plus vraiment la compression vidéo mais traite essentiellement de contenu multimédia et d'interactivité. Le MPEG21 est une extension du MPEG7.

I.5.6 H.261 :

La norme H.261 [37] est la première norme vidéo (réalisée en 1990). Elle a été développée pour les applications de visiophonie à des débits px64 kb/s ou p est compris entre 1 et 30 pour le réseau RNIS (Réseau Numérique à Intégration de Services).

Le format d'image traité est le QCIF (144x176 pixels) et optionnellement le CIF (288x352pixels) et le Sub-QCIF (96x128). Chaque MB subit une TCD de taille 8x8. Les coefficients ainsi obtenus sont alors quantifiés puis codés par un codage à longueur variable. La prédiction se fait INTER image et peut être améliorée par une compensation de mouvement et un filtrage. Les images peuvent être codées I ou P mais pas B.

I.5.7 H.263 :

H.263 [38] est une norme de codage vidéo pour la communication vidéo à très bas débit dont la première version a été adoptée en 1995. Elle vise les applications de visiophonie et visioconférence. Cette norme repose sur les principes mis en place par les recommandations H.261.

Les formats supportés par cette norme sont le Sub-QCIF (96×128 pixels) et le QCIF(144x176 pixels) et optionnellement le CIF (288x352 pixels), le 4CIF (576x704 pixels) et le 16CIF (1152x1408 pixels). L'utilisation d'image B est possible. En plus de

cette configuration de base, six options de codage ont été ajoutées afin d'augmenter les performances du codage et ses fonctionnalités. La version 2 de la recommandation H.263 (1998), souvent appelée H.263+ [39] met en œuvre douze options supplémentaires et permet désormais de définir des formats et des fréquences d'image personnalisées. Les options ajoutées améliorent la qualité et la robustesse aux erreurs. La dernière version de H.263 (2000) appelé H.263++ [40] ajoute trois options. Outre l'amélioration en termes de qualité et de taux de compression, elle prend mieux en compte la transmission vidéo en temps réel sur un réseau à qualité de service non garantie (perte de paquets…).

I.5.8 H.264 :

Développé par le groupe JVT, le H.264**[41]** appelé aussi MPEG-4 Part 10 ou AVC (Advanced Video Coding) est un standard très performant pour la compression vidéo. En comparaison avec le MPEG-2,le H.264 délivre des vidéos de haute définition avec seulement la moitié du taux binaire et même le quart. La norme H.264 est utilisée dans des applications telles que les lecteurs de disques Blu-ray, les vidéos de You Tube et iTunes, les logiciels du web tels qu'Adobe Flash Player et Microsoft Silverlight. H.264 est aussi deux fois plus efficace que le standard MPEG-4 Part 2 pour la compression des vidéos naturelles. On se limitera à cette brève introduction du H.264 puisque le chapitre deux lui est entièrement consacré.

I.5.9 HEVC (H.265) :

Successeur désigné du H.264, le format de codage vidéo H.265 est bien parti pour être lancé dans des produits dès 2013. En Janvier 2013, de nombreux représentants des télécoms, de l'informatique, de l'industrie de l'électronique et du monde de la télévision ont approuvé la publication d'un brouillon pour la norme High Efficiency Video Coding (HEVC). C'est le format de codage vidéo H.265 qui doit prendre la suite de la norme actuelle H.264 / AVC. Si H.264 est utilisé pour l'encodage et le décodage vidéo HD et Full HD, H.265 a été pensé pour accompagner du contenu 4K (4 096 x 2 160 pixels) et Ultra HDTV ou 8K (7 680 x 4 320 pixels). Le tour de force est que le H.265 promet d'améliorer les performances du H.264 et qu'il doublerait le niveau de compression sans pour autant compromettre la qualité de l'image. On dit que le H.265 peut faire de 35 à 67 % de mieux que le H.264 en compression.

H.265 est mis au point dans le cadre d'une collaboration entre le groupe MPEG et l'ITU-T. Sa finalisation est prévue pour le début de l'année 2013 avec d'abord une adoption

dans le domaine de la mobilité. Pour les services de télévision, cela devrait prendre plus de temps. Le groupe MPEG travaille par ailleurs sur un nouveau format de compression pour la vidéo 3D permettant de se passer de lunettes. Une technologie qui pourrait être normalisée en 2014. Cette solution peut fournir plus que les vues gauche et droite actuellement proposées par le format Blu-ray 3D. Les utilisateurs vont pouvoir déplacer leurs têtes vers la gauche ou vers la droite afin d'obtenir des effets 3D.

I.6 Paramètres d'évaluation de la qualité:

Quelles que soient les performances de la technique de compression, il existera toujours des distorsions dans l'image et la vidéo reconstruites dont la qualité peut être évaluée objectivement et subjectivement. La mesure objective de la qualité est faite en termes de rapport signal à bruit ou PSNR (Peak Signal to Noise Ratio), taux de compression (CR) et similarité structurelle ou SSIM (Structural SIMilarity). Plus la valeur du PSNR est élevée, plus l'image compressée est proche de l'image source. Des images/vidéos ayant les mêmes valeurs du PSNR peuvent avoir des qualités perceptuelles différentes [42]. L'évaluation par la SSIM a prouvé sa correspondance à la qualité perçue visuellement. En fait, cette méthode consiste à mesurer l'erreur de visibilité entre une image reconstruite et une image de référence en utilisant une variété de propriété du système visuel humain (SVH). La valeur de SSIM est comprise entre 0 et 1. Plus la valeur de SSIM proche de 1, plus les images se ressemblent [43].

Avec l'évaluation subjective, des observateurs jugent directement et visuellement la qualité de l'image /vidéo [44].

La valeur du PSNR est exprimée en décibel (dB) et est donnée par l'équation suivante :

$$PSNR = 10log10\frac{S^2}{[\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[\,I(m,n)-\hat{I}(m,n)]^2\;\;]} \qquad (2)$$

Avec:

- *S*: Intensité maximale des pixels (2^8-1= 255).
- *I(x,y)* : Représente l'image initiale.
- *Î(x,y)* : Représente l'image reconstruite.
- $[\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[\,I(m,n)-\hat{I}(m,n)]^2\;\;]$ représente l'erreur quadratique moyenne ou MSE (Mean Square Error).

Pour deux images x et y de taille N x N, le SSIM est défini comme suit :

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (3)$$

Avec :

σ_x étant l'écart type de x, σ_x l'écart type de y et σ_{xy} l'écart type de xy définis comme suit :

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{1/2}$$

$$\sigma_y = \left(\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu_y)^2\right)^{1/2}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

μ_x la moyenne de x et μ_y la moyenne de y données comme suit :

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i$$

C1 et C2 sont des constantes égales à *0.01 et 0.03* respectivement [43].

I.7 Conclusion :

Dans ce chapitre, les notions de base relatives à la vidéo numérique et sa compression ont été introduites. Les normes de compression vidéo MPEGs et H.26x ont été présentées succinctement. Elles se distinguent principalement par leurs débits et par la qualité qu'elles offrent en rapport direct avec la complexité du traitement. Néanmoins, le schéma de base reste le même, basé sur le découpage en blocs, la TCD, le codage INTRA et le codage INTER ou à compensation de mouvement.

Une étude détaillée de la norme H.264 fera l'objet du chapitre deux.

Chapitre II : La norme de compression H.264/AVC

II.1 Introduction :

H.264/AVC, ou MPEG-4 AVC (Advanced Video Coding), est une norme de codage vidéo développée conjointement par l'UIT-T Q.6/SG16 Video Coding Experts Group (VCEG) ainsi que l'ISO/CEI Moving Picture Experts Group (MPEG). Elle est donc le produit d'un effort de partenariat connu sous le nom Joint Video Team (JVT). La norme UIT-T H.264 et la norme ISO/CEI MPEG-4 Part 10 (ISO/CEI 14496-10) sont techniquement identiques, et la technologie employée est aussi connue sous le nom AVC, pour "Advanced Video Coding". La première version de la norme a été approuvée en Mai 2003 et la plus récente date d'Avril 2012. Des extensions connues sous le nom de FREXt (Fidelity Range Extensions) ont été rajoutées au standard, lui fournissant ainsi plus de capacités.

Malgré l'apparition du H.265, le H.264 reste le standard de fait de la compression vidéo et l'état de l'art en la matière. On ne compte plus les appareils compatibles : de la platine blu-ray au téléphone portable, il est omniprésent.

Ce deuxième chapitre est entièrement consacré au développement des principaux aspects du standard H.264. Les différents profils et niveaux de la norme sont présentés et les trois modes de codage (I, P et B) sont détaillés. le H.264 permet d'avoir des blocs de tailles variables, de faire référence aux images précédentes pour encoder les vecteurs, d'estimer les mouvements au quart de pixel et de pondérer les vecteurs, de compresser efficacement les coefficients résiduels (CAVLC ou CABAC), de mieux repérer les redondances spatiales en se basant sur les blocs voisins, d'utiliser des filtres de lissage de blocs (deblocking filters) etc. Toutes ces fonctions sont aussi examinées.

II.2 Structure de la norme H.264/AVC :

II.2.1 Les profils : [45,46]

Le standard inclut les 6 ensembles de caractéristiques suivants, qui sont appelés des *profils*, chacun ciblant une classe d'applications précises et utilisant un ensemble particulier d'algorithmes de codage.

- **Baseline Profile (BP)** : Principalement pour les applications à bas-coût utilisant peu de ressources, ce profil est très sollicité dans les applications mobiles et de visioconférence.
- **Main Profile (MP)** : À l'origine, prévu pour les applications grand public de diffusion et de stockage, ce profil a perdu de l'importance quand le profil *High* a été ajouté avec le même objectif.
- **Extended Profile (XP)** : Prévu pour la diffusion en flux (*streaming*) des vidéos, ce profil a des capacités de robustesse à la perte de données et de changement de flux.
- **High Profile (HiP)** : Le profil principal pour la diffusion et le stockage sur disque, en particulier pour la télévision haute définition (ce profil a été adopté pour les disques HD DVD et Blu-ray ainsi que pour la télévision numérique française haute définition).
- **High 10 Profile (Hi10P)** : Ce profil va au-delà des applications grand public et s'appuie sur le profil High, ajoutant jusqu'à 10 bits de précision par pixel.
- **High 4:2:2 Profile (Hi422P)** : Le profil principal pour les applications professionnelles, il s'appuie sur le profil High 10, ajoutant le support pour la quantification 4:2:2 jusqu'à 10 bits par pixel.
- **High 4 :4 :4 Profile (Hi444P)** : Ce profil s'appuie sur le profil High 4:2:2, ajoutant le support pour la quantification 4:4:4, jusqu'à 12 bits par pixel et en plus le support pour un mode sans perte efficace. **Note :** Le profil High 4:4:4 est en cours de suppression du standard en faveur d'un nouveau profil 4:4:4 en cours de développement.

La table II.1 récapitule les caractéristiques des différents profils.

	Baseline	Extended	Main	High	High 10	High 4:2:2	High 4:4:4
tranches I et P	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
tranches B	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
tranches SI et SP	✗ Non	✓ Oui	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non
Image de Références Multiples	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
Filtre anti-blocs	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
codage CAVLC	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
codage CABAC	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
ordonnancement flexible des macroblocs (FMO)	✓ Oui	✓ Oui	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non
ordonnancement arbitraire des tranches (ASO)	✓ Oui	✓ Oui	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non
tranches redondantes (RS)	✓ Oui	✓ Oui	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non
partitionnement des données (DP)	✗ Non	✓ Oui	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non
codage entrelacé (PicAFF, MBAFF)	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
format 4:2:0	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
format monochrome (4:0:0)	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui
format 4:2:2	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui
format 4:4:4	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✓ Oui
pixel 8 Bit	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui	✓ Oui
pixel 9 et 10 Bit	✗ Non	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui
pixel 11 et 12 Bit	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✓ Oui
transformée 8×8	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui
matrices de quantification	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui
quantification Cb et Cr séparée	✗ Non	✗ Non	✗ Non	✓ Oui	✓ Oui	✓ Oui	✓ Oui
codage sans-perte	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✗ Non	✓ Oui

Table II.1 Les profils du H.264/AVC

Les profils Base (BP), Main (MP) et Extended (XP) sont ceux définis dans la version originale du standard, alors que les autres profils (High (HP), High 10 (Hi10P), High 4:2:2 (Hi422P), High 4:4:4 (Hi444P)) ont été définis dans l'amendement FRExt.

II.2.2 les niveaux :

Il est d'une extrême importance la considération des ressources calculatoires et mémoires nécessaires à toute implémentation. La taille ou le format de l'image et la fréquence de la vidéo (frame rate) ont une influence notable sur ces paramètres. H.264/AVC définit 16 niveaux différents liés aux paramètres précédemment cités et aussi au nombre d'images références et au débit. La table suivante présente les caractéristiques des différents niveaux.

Niveau	**Taille maximum de l'image**	**Nombre d'images par seconde**	**Débit maximum (pour VCL) dans les profils Non-FRExt**	**Nombre maximum d'images références**
1	QCIF	15	64 kbps	4
1.b	QCIF	15	128 kbps	4
1.1	CIF ou QCIF	7.5 / 30	192 kbps	2 / 9
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR (480i or 576i)	30 / 25	4 Mbps	6
2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD Formats (720p ou 1080i)	60p / 30i	20 Mbps	4
4.1	HD Formats (720p ou 1080i)	60p / 30i	50 Mbps	4
4.2	1920x1080p	60p	50 Mbps	4
5	2kx1k	72	135 Mbps	5
5.1	2kx1k ou 4kx2k	120 / 30	240 Mbps	5

Table II.2 Les niveaux du H.264/AVC

Les débits sont plus importants dans le cas des profils FRExt puisqu'ils sont spécifiques à des applications exigeant de la haute fidélité. La table II.3 donne les coefficients de pondération des débits présentés dans la 4ème colonne de la table II.2.

Profil FRExt	Coefficient de pondération
High	1.25
High 10	3
High 4:2:2	4
High 4:4:4	4

Table II.3 Les coefficients de pondération applicable au profil FRExt.

Il est à noter que les niveaux spécifient la taille maximale du frame (nombre de pixels par frame) et non pas les dimensions horizontale et verticale. Cependant, des contraintes sur ces tailles sont indiquées et précisent qu'elles doivent respecter l'équation 4.

$$\text{Taille (Horiz ou vert)} \leq \sqrt{(tailleduframe * 8)} \quad (4)$$

Si, à un niveau donné, la taille de l'image est plus petite que ce qui est spécifié dans la table II.2, alors un plus grand nombre d'images références peut être utilisé (jusqu'à 16 images) pour l'estimation et la compensation de mouvement.

Similairement, au lieu de spécifier une fréquence de frame à un niveau particulier, c'est un débit d'échantillons (de pixels) qui est donné et exprimé en termes de macro blocs par seconde. Ainsi, si la taille de l'image est plus petite que ce qui est spécifié dans la table II.2, le débit des frames peut être augmenté jusqu'à un maximum de 172 frames/sec.

II.2.3 Format des données codées:

Le H.264/AVC spécifie deux couches: la couche de codage vidéo ou VCL (Video Coding Layer), et la couche d'abstraction réseaux ou NAL (Network Abstraction Layer).

La NAL est définie pour permettre l'usage de la même syntaxe vidéo dans de nombreux environnements réseau. Ceci, inclut des possibilités telles que des paramètres de séquence (SPS : *Sequence parameter set*) et d'image (PPS : *picture parameter set*) qui offrent plus de robustesse et de flexibilité que les conceptions antérieures.

La VCL, décrite dans le paragraphe suivant, est spécifiée pour représenter efficacement le contenu des données vidéo. C'est la sortie du processus de codage, séquence de bits représentant les données vidéo codées, qui sont encapsulées dans des unités NAL avant d'être transmises ou stockées.

II.3 Les principaux outils de codage du H.264/AVC :

L'architecture du codeur et les principales phases de compression sont présentées dans la figure II.1. A noter que la compression est basée sur une transformée en DCT compensée en mouvement, et que le codeur renferme un décodeur dont les principaux

blocs fonctionnels peuvent être repérés en suivant le sens de la droite vers la gauche dans la figure II.1.

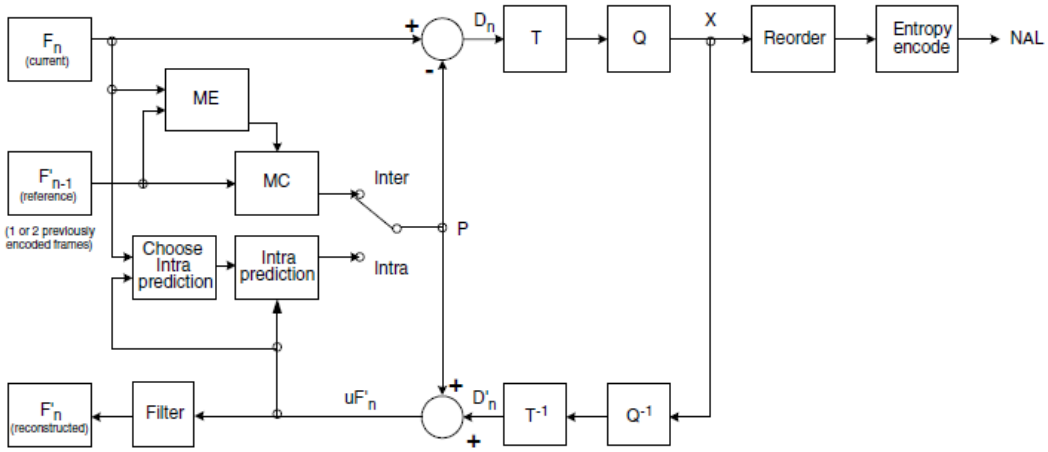


Figure II.1: Schéma fonctionnel d'un codeur vidéo H.264 [47]

Chaque image, qui peut être un frame ou un champ (cas des vidéos entrelacées) est partitionnée en une ou plusieurs tranches (slices). Chaque slice est composé de macroblocs, qui sont des blocs de dimension fixe couvrant une zone rectangulaire de l'image. Un macrobloc se compose de 16x16 échantillons de la composante de luminance et de 8x8 échantillons des deux composantes de chrominance. Cependant, chaque macrobloc est aussi divisé en des partitions de sous-macroblocs utilisés en prédiction à compensation de mouvement. Ces partitions de prédiction peuvent avoir des dimensions différentes: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 et 4x4.

La transformée spatiale des données résiduelles peut être 8x8 (cas du FREXt) ou 4x4. Dans les principaux précédents standards, la transformée s'opérait toujours sur des blocs 8x8. L'option 4x4 fournit une spécificité améliorée dans la localisation des signaux différentiels. La taille du bloc utilisé pour la transformée spatiale est toujours soit identique soit plus petite que la taille du bloc utilisé pour la prédiction. La hiérarchie d'une séquence vidéo est comme suit :

Séquence vidéo{images {tranche{macroblocs {partitions de macrobloc{partitions de sous-macrobloc{blocs {échantillons}}}}}}.

Une séquence vidéo est un groupe d'images désigné dans la littérature par GOP (Group Of Pictures).

II.3.1 Les tranches (Slices) :

Les macroblocs sont organisés en tranches (slices) qui représentent des sous ensembles d'une image donnée pouvant être décodées indépendamment. L'ordre de transmission

des macroblocs dépend de ce qui est appelé la carte d'allocation des macroblocs (*Macroblock Allocation Map)* et ne suit pas nécessairement l'ordre "raster-scan". La norme H.264/AVC définit cinq types de tranches dont la plus simple est la tranche I (Intra).

- Slice I (Intra) : tous les macroblocs sont codés sans aucune référence aux autres images de la séquence vidéo.
- Slice P (prédictive) : Contient des macroblocs P utilisant les macroblocs des slices I.
- Slice B (bi-prédictive): utilise pour la prédiction des macroblocs des tranches I et/ou P
- Sice SP (switching Predictive) et SI (switching Intra) : Les tranches SP et SI sont des tranches à codage spécial permettant des aiguillages efficaces entre des flux vidéo et l'accès aléatoire pour les décodeurs [48,49]. Une exigence ordinaire dans une application de streaming, est la possibilité de basculer entre plusieurs flux codes. C'est le cas où une même vidéo est codée à différents débits binaires pour une transmission sur internet. Dans ce cas un décodeur tente de décoder le flux le plus important qu'il peut recevoir mais change automatiquement à un flux plus bas si le débit des données chute.

L'ordonnancement flexible des macroblocs (FMO : Flexible macroblock ordering) et l'ordonnancement arbitraire des tranches (ASO : Arbitrary slice ordering) sont des techniques de restructuration de l'ordonnancement des régions fondamentales de l'image (macroblocs). Typiquement utilisées pour améliorer la résistance aux erreurs et aux pertes, ces techniques peuvent également être utilisées à d'autres fins.

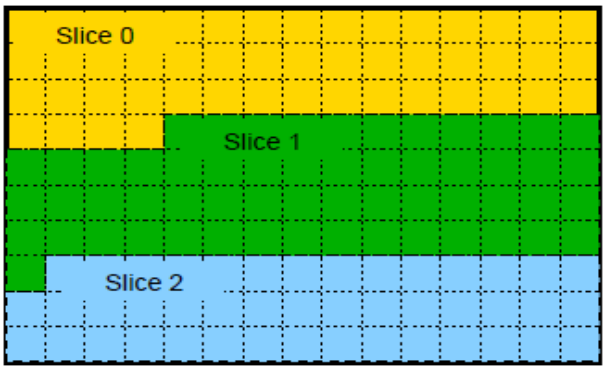


Figure II.2 Découpage d'une image en slices.

La Figure II.3 montre une illustration simplifiée de la syntaxe d'un slice codé. L'entête du slice définit, entre autres, le type du slice et l'image codée à laquelle appartient le

slice. Elle aussi peut contenir des instructions relatives à la gestion des images de référence. Les données du slice consistent en une série de macroblocs codés et/ou de macroblocs omis (non codés).

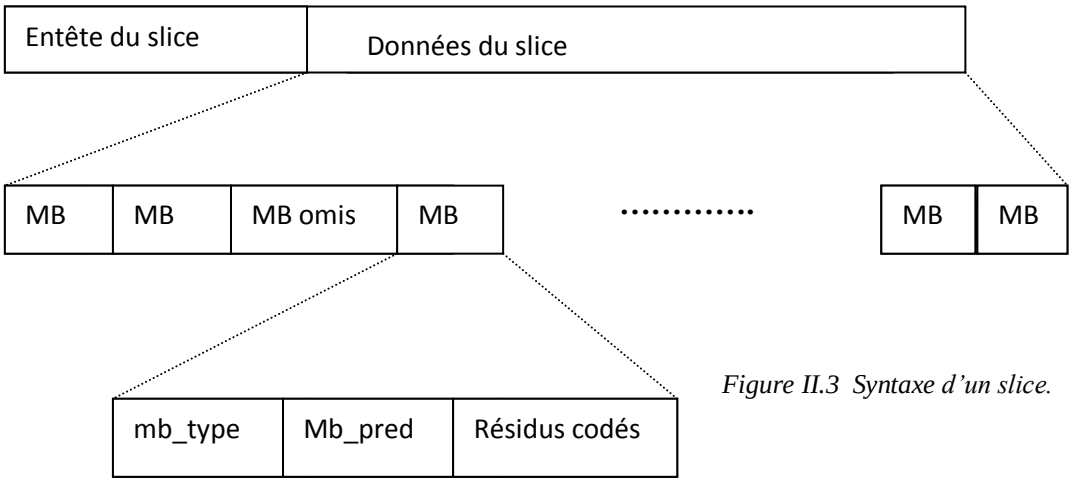


Figure II.3 Syntaxe d'un slice.

Chaque macrobloc contient une série d'entêtes et des données résiduelles codées comme le montre la table II.4.

mb type	Détermine le type de codage du macrobloc (I, P ou B) et la taille de la partition du macrobloc.
mb pred	Détermine les modes de l'intra-prédiction, les listes de référence (liste 0 Et/ou liste 1), et les vecteurs de mouvement codés différentiellement pour chaque partition du macrobloc (macroblocs inter, excepté pour la taille de la partition 8 × 8).
sub mb pred	(uniquement pour les partitions des macroblocs 8 × 8). Détermine la taille de la partition pour chaque sous-bloc ; les listes de référence et les vecteurs de mouvement pour chaque sous-partition.
coded block pattern	Identifie les blocs 8 × 8 (luma et chroma) qui contiennent des coefficients transformés codés.
mb qp delta	Change le paramètre de quantification.
residual	Coefficients transformés codés correspondant aux échantillons de l'image différentielle après prédiction.

Table II.4 Eléments d'un macrobloc.

II.3.2 Les macroblocs :

Dans un flux vidéo, chaque image, pouvant être un frame ou un champ (cas des vidéos entrelacées) est partitionnée en macroblocs de dimension fixe. Ces macroblocs couvrent

une zone rectangulaire de l'image de 16x16 échantillons de la composante de luminance et de 8x8 échantillons de chacune des deux composantes de chrominance. Tous les échantillons de la composante luminance et chrominance d'un macrobloc sont prédits soit spatialement soit temporellement. Les résidus résultant de la prédiction sont transmis après codage des transformées.

II.4 Les modes de codage :

II.4.1 Codage des tranches de type I (Slice I):

Dans un slice I, ainsi que pour les macrocblocs codés en mode intra et faisant partie de slices non I, les pixels sont d'abord prédits spatialement à partir des valeurs de leurs pixels voisins. L'information différentielle subit alors une transformée 4x4 ou 8x8 (cas du FRExt) puis une quantification.

Les coefficients quantifiés sont réordonnés en "zig-zag" ou en "raster scan ordre" et compressés par l'une des méthodes de codage, à savoir CAVLC (context-adaptive variable length coding) ou CABAC (context-adaptive binary arithmetic coding). En mode de codage PAF ou PicAFF (Picture-adaptive frame-field, chaque champ est compressé de la même manière qu'une image entière. En mode MBAFF (Macroblock-adaptive frame-field), si une paire de macroblocs est en mode champ, alors les champs voisins sont utilisés pour la prédiction spatiale ; et si une paire de macroblocs est en mode frame, les frames voisins seront exploités pour la prédiction. La décision champ ou frame est prise avant l'application du reste des outils de codage décrits ci-après.

La prédiction temporelle n'est pas utilisée pour les macroblocs codés intra, mais l'est pour les macroblocs de type P et B.

II.4.1.1 Prédiction Spatiale Intra :

La norme définit trois types de prédiction spatiale en mode intra:

- ✓ prédiction de bloc entier pour la composante de luminance 16x16 ou la taille du bloc correspondant de la composante de chrominance ,
- ✓ prédiction luminance 8x8 (uniquement pour FRExt),
- ✓ prédiction luminance 4x4.

II.4.1.1.1 Prédiction 16x16 de luminance :

Dans le cas de la prédiction d'un bloc entier, les valeurs des pixels de la composante de luminance ou de chrominance sont prédites des pixels situés sur les bords des macroblocs voisins décodés précédemment. Cette opération peut être conduite selon

quatre modes sélectionnés par le codeur pour la prédiction de chaque macrobloc, et schématisés dans la figure II.4:

a. Mode 0 (vertical) : Interpolation verticale à partir des pixels situés au dessus du macrobloc considéré,
b. Mode 1 (horizontal) : Interpolation horizontale à partir des pixels situés à gauche du macrobloc considéré,
c. Mode 2 (DC) : les échantillons sont obtenus en faisant la moyenne entre l'horizontale et la verticale,
d. Mode 3 (plane) : les échantillons situés dans la partie supérieure par rapport à la diagonale sont interpolés à un angle de 45° entre haut-droit et bas-gauche. Les autres sont interpolés à un angle de 45° de bas-gauche à haut-droit.

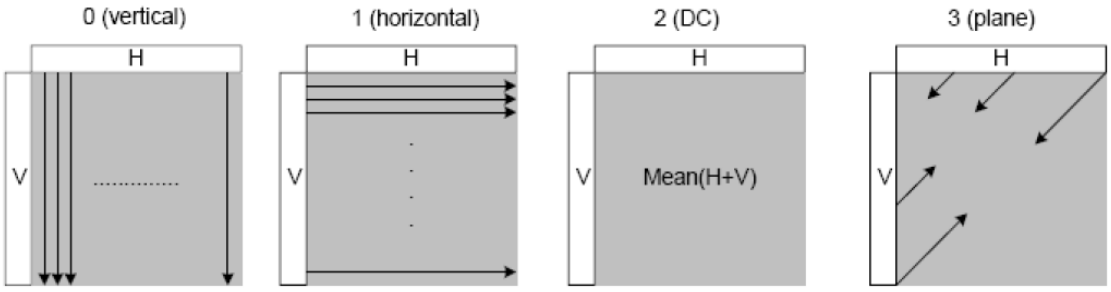


Figure II. 4 Modes d'intra-prédiction d'un bloc de luminance16x16.

La prédiction de la chrominance est définie pour trois tailles du bloc qui dépendent du format choisi (8x8 chroma pour le format 4:2:0, 8x16 chroma pour le format 4:2:2, et 16x16 chroma pour le format 4:4:4). Le type de l'intra-prédiction de la chrominance est sélectionné indépendamment du type de l'intra-prédiction de la luminance.

II.4.1.1.2 Prédiction d'un bloc de luminance 4x4:

La prédiction d'un bloc de luminance 4x4 se réfère aux échantillons au-dessus et à gauche qui ont été précédemment codés et reconstruits. Ils sont donc disponibles dans le codeur et le décodeur pour former une référence de prédiction.

Les neuf modes de prédiction sont décrits par la figure II.5 et récapitulés dans la figure II.6 :

- Mode 0 (Vertical): Les échantillons sont prédits à partir des échantillons du haut par une simple interpolation verticale.
- Mode 1 (horizontal): Les échantillons de gauche I, J, K et L sont extrapolés horizontalement.

- Mode 2 (DC): Les échantillons sont obtenus en faisant la moyenne entre la verticale et l'horizontale.
- Mode 3 (Down-Left Diagonal): Les échantillons sont interpolés à un angle de 45° entre haut droite et bas-gauche.
- Mode 4 (Direct Diagonal): Les échantillons sont extrapolés à un angle de 45° vers le bas et vers la droite.
- Mode 5 (Vertical-Left): Extrapolation sous un angle, approximativement, de 26.6° du haut gauche au bas droit.
- Mode 6 (Horizontal-Down): Extrapolation sous un angle, approximativement de 26.6° au dessous de l'horizontale.
- Mode 7 (Vertical-Right): Interpolation sous un angle approximativement de 26.6° à la droite de la verticale.
- Mode 8 (Horizontal-Up): Interpolation sous un angle approximativement de 26.6° au-dessus de l'horizontale.

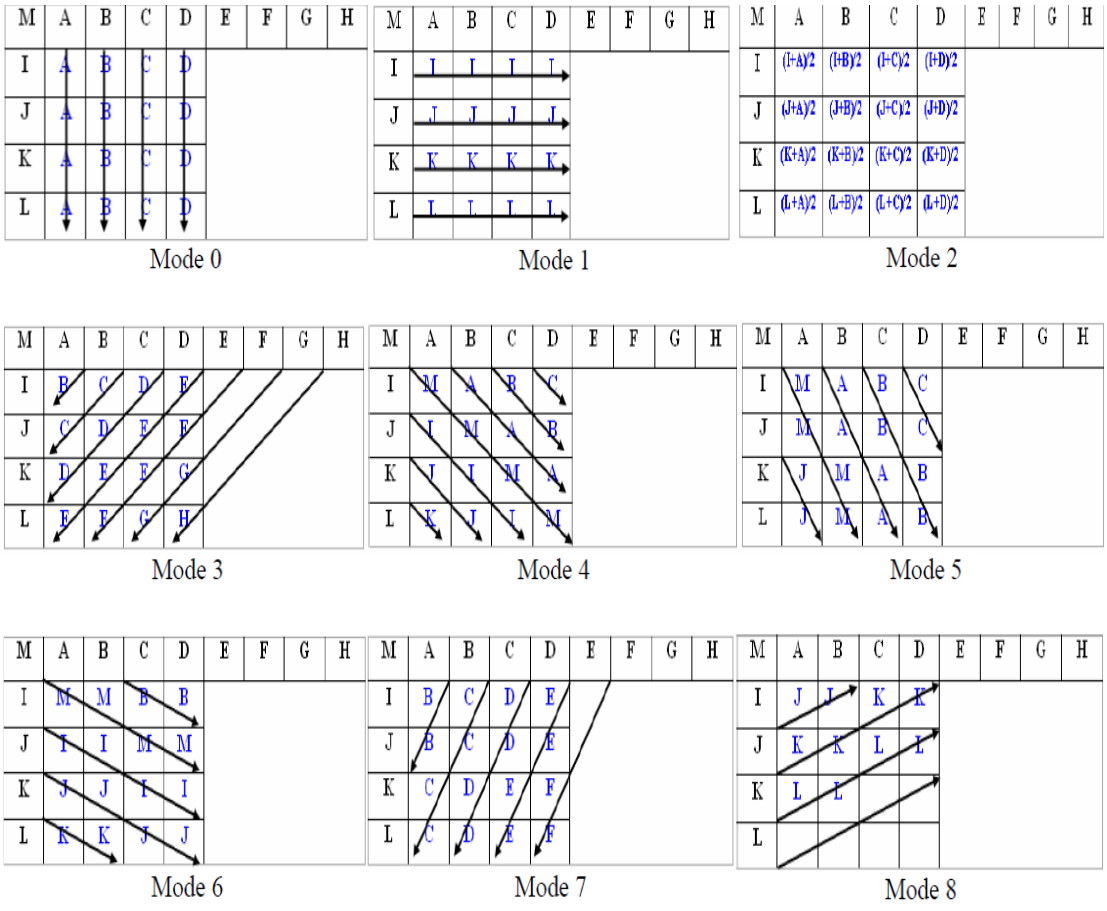


Figure II.5 : Prédiction d'un bloc 4x4 en mode intra

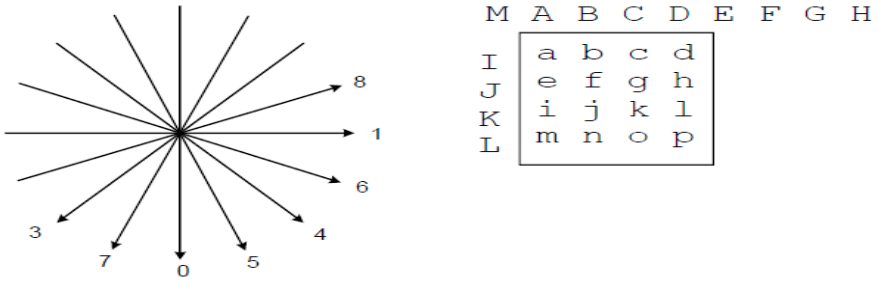


Figure II.6 Récapitulation des modes de prédiction d'un bloc 4x4.

II.4.1.1.3 Prédiction d'un bloc 8x8 :

Dans le cas du profil FREXt, l'intra-prédiction de blocs 8x8 utilise le même principe que l'intra-prédiction 4x4 avec une taille de bloc 8x8 au lieu de 4x4.

II.4.1.2 La transformée :

La transformation entière opère sur des macroblocs d'erreur provenant de l'intra-prédiction ou de l'inter-prédiction. Ces macroblocs ont une taille fixe 4x4 contrairement aux normes antérieures qui traitent des blocs de taille 8x8. La transformation entière a pour but de décorréler les blocs originaux ou différentiels, de concentrer l'énergie aux fréquences basses et de permettre d'éliminer les redondances spatiales. Parmi les aspects les plus importants apportés par la norme H.264/AVC, on trouve :

- ✓ La norme est basée sur une transformée directe et une transformée inverse, toutes deux entières. Ceci, contrairement aux normes MPEG-2 et MPEG-4 part 2, garantit une complète convergence entre le codeur et le décodeur.
- ✓ La transformée est définie de façon à ce que pour des pixels codés sur 8 bits, elle peut être implémentée facilement à l'aide de l'arithmétique 16 bits. En effet, les transformées spécifiées dans les standards précédents utilisaient 32 bits voire même plus.
- ✓ La transformée (du moins la 4x4) peut être implémentée moyennant juste des additions, des soustractions et des décalages. La taille du bloc (4x4) convient aussi à des implémentations hardwares.

Etant donné que la taille du macrobloc est 16x16, il doit alors être divisé en blocs de tailles 8x8 ou 4x4 pour pouvoir appliquer des transformées 8x8 ou 4x4.

De plus amples détails seront présentés au chapitre trois consacré aux transformées.

II.4.1.3 La quantification :

La quantification a pour but de discrétiser les amplitudes des coefficients issus de la transformée entière. La quantification la plus simple est la quantification uniforme qui consiste à diviser tous les coefficients par la même valeur. Toutefois, il est judicieux d'adapter ce quantificateur aux valeurs obtenues après la transformée. On utilise alors des tables de seuillage de quantification qui préservent l'importance des coefficients de la transformée. Il suffit de diviser les coefficients les plus énergétiques (basses fréquences) par des seuils peu élevés et les coefficients les moins énergétiques (hautes fréquences) par des seuils élevés. La quantification directe est définie par l'équation suivante :

$$Z_{ij} = round(Y_{ij} / Qstep) \quad (5)$$

Où Yi j est le coefficient transformé, Qstep est la valeur du pas de quantification et Zij est le coefficient transformé.

Chaque table de seuillage est caractérisée par la donnée d'un paramètre QP (Quantification Parameter). Dans la norme H.264, cinquante deux pas de quantificateurs sont prévus [50] indexés par un paramètre de quantification comme l'illustre la table II.5.

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	…
QPstep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	…

QP	18	…	24	…	30	…	36	…	42	…	48	…	51
QPstep	5	…	10	…	20	…	40	…	80	…	160	…	224

Table II.5 Les pas de quantification définis dans le H.264/AVC

(A noter que la valeur de Qstep double pour chaque incrément de 6 de QP).

Disposer d'un large intervalle de valeurs de quantificateurs permet au codeur de contrôler la balance entre le débit et la qualité. Les valeurs de QP peuvent être différentes pour la composante de luminance et la composante de chrominance. Les deux paramètres varient de 0 à 51 et, par défaut, QP chrominance (noté QPc) peut être déterminé à partir de QP luminance(noté QPy) de manière à ce que QP_C soit inférieur à QPy pour QPy > 30. Cependant, la relation entre QPc et QPy peut être définie par l'utilisateur et signalée dans les paramètres de l'image.

II.4.1.4 Le balayage des données (scanning) :

Le balayage consiste à mettre les coefficients les plus énergétiques au début du flux des données et placer les données nulles à la fin pour minimiser la taille des informations à coder.

Lorsqu'un macrobloc est compressé en utilisant la transformée 4x4 en mode frame, les coefficients quantifiés sont balayés en "zig-zag" comme indiqué par la figure II.7 (a). En mode "field", le balayage est modifié et suit l'ordre indiqué par la figure II.7(b), reflétant une corrélation moindre entre les données dans la direction verticale.

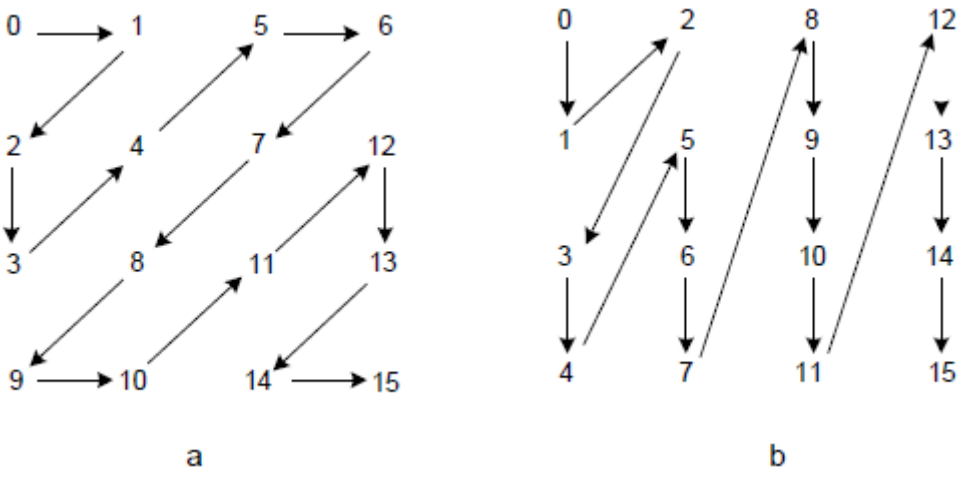


Figure II.7 Ordre de balayage des coefficients (a) en mode frame (b) en mode champ (field)

II.4.1.5 Le codage entropique :

Le codage entropique peut être réalisé de trois manières différentes dans la norme H.264. Une première méthode utilise une table universelle de mots de code (UVLC - Universal Variable Length Coding). Cette table est utilisée pour coder la plupart des éléments de synchronisation comme les entêtes. Les deux autres méthodes sont utilisées pour coder presque tous les autres éléments syntaxiques (coefficients, vecteurs mouvements). Il s'agit d'une part, d'un codage VLC adaptatif au contexte (CAVLC Context Adaptative Variable Length Coding) et d'un codage arithmétique adaptatif contextuel (CABAC - Context Adaptative Binary Arithmetic Coding) d'autre part.

Par rapport au CAVLC, le codage CABAC garantit en général une réduction du débit binaire de 10 à 15% [51] pour une même qualité d'images.

II.4.1.5.1 Codage VLC, UVLC et CAVLC:

Lorsque le mode de codage entropique est 0, les données résiduelles sont codées selon l'algorithme CAVLC et le reste des unités par le code Exponential-Golomb (Exp-Golomb).

a) Codage entropique Exp-Golomb :

Les codes Exp-Golomb [52] sont des codes à longueur variable avec une construction régulière. L'examen de la table II.6 permet de déduire qu'ils ont la forme :

[M zéros][1][INFO]

INFO est un champ d'information binaire non signée. La longueur du code est $(2M + 1)$ bits et chaque code peut être construit par le codeur en se basant sur l'indice CodeNum, correspondant à la valeur réelle (signée dans le cas des vecteurs de mouvement) du coefficient :

$$M = floor(log_2[CodeNum + 1]) \quad (6)$$

$$INFO = CodeNum + 1 - 2^M \quad (7)$$

Le décodage suit la procédure suivante:

1. Lire les premiers M zéros précédents le 1.
2. Lire les M bits du champ INFO.
3. $CodeNum = 2^M + INFO - 1$ (8)

A noter que pour le code 0, INFO et M sont égaux à zéro.

CodeNum	Code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
…	…

Table II.6 Code Exponential Golomb (appelé aussi Universal Variable Length Code)

b) Le codage CAVLC :

Il s'agit de la méthode utilisée pour le codage des coefficients transformés et scannés en "zig-zag" de blocs 4 × 4 (et 2 × 2). Le codage CAVLC [53] exploite les caractéristiques des blocs 4x4:

1. Les blocs obtenus après transformation et quantification sont creux (sparse). Le codage CAVLC utilise alors "run-level coding" pour représenter les séries de zéros consécutifs d'une manière compacte.

2. Les coefficients non nuls correspondants aux hautes fréquences après lecture en "zig-zag" sont souvent des séquences de ±1 et CAVLC signale leur nombre d'une façon compacte.

3. Le nombre de coefficients non nuls dans les blocs voisins est corrélé. Ce nombre est codé par l'utilisation de tables de consultation (LUT pour Look-Up Table).

4. L'amplitude des coefficients non nuls est beaucoup plus importante au début du vecteur des coefficients réordonnés et diminue en se déplaçant vers les hautes fréquences. Le code CAVLC adapte le choix des LUTs des VLC pour le niveau des paramètres en fonction des amplitudes récemment codées.

Le standard H.264 spécifie alors douze LUTs de codes supplémentaires:

- six pour caractériser le contenu des blocs transformés,
- quatre pour indiquer le nombre de coefficients,
- une table pour signaler l'amplitude du coefficient quantifié (code suivi d'un code à longueur fixe adapté au contexte pour identifier la valeur particulière) et
- une pour représenter les séries de coefficients quantifiés consécutivement nuls.

La sélection des tables et la longueur du code à longueur fixe est basée sur des statistiques locales du flux courant, d'où la dénomination CAVLC.

II.4.1.5.2 Codage CABAC :

Le codage CABAC a montré sa capacité à améliorer l'efficacité de la compression de 10à 15% par rapport au codage CAVLC mais aux dépends d'un traitement beaucoup plus complexe. Le codage arithmétique permet d'affecter un nombre non entier de bits aux symboles.

Le codage CABAC atteint les meilleurs taux de compression [54] par :

(a) la sélection de modèles de probabilité pour chaque élément syntaxique selon son contexte,

(b) l'adaptation des estimations de probabilités basées sur les statistiques locales, et

(c) l'utilisation du codage arithmétique à la place du codage à longueur variable.

Le codage d'un symbole passe par les étapes suivantes:

1. "Binarisation": CABAC utilise le codage arithmétique binaire où uniquement des décisions binaires (0 ou 1) sont codés. Tout symbole de valeur non binaire tel qu'un coefficient transformé ou un vecteur de mouvement, doit alors être converti en un code binaire avant le codage arithmétique, c'est la binarisation.

Les étapes suivantes (2, 3 et 4) sont répétées pour chaque bit (ou 'bin') du symbole binarisé :

2. Sélection du modèle du contexte. Un modèle de contexte est un modèle de probabilité pour un ou plusieurs bins du symbole binarisé et est choisi parmi une sélection de modèles disponibles en fonction des statistiques des symboles de données récemment codés. Le modèle contextuel stocke les probabilités de la valeur (0 ou 1) de chaque bin.

3. Codage arithmétique : le codeur arithmétique code chaque bin selon le modèle de probabilité sélectionné. Il n'existe que deux sous ensembles pour chaque bin correspondant à 0 ou à 1.

4. Mise à jour des probabilités: le modèle contextuel sélectionné est mis à jour en fonction de la valeur codée actuelle. Par exemple, si la valeur de bin est 1, le compteur des fréquences de 1 augmente.

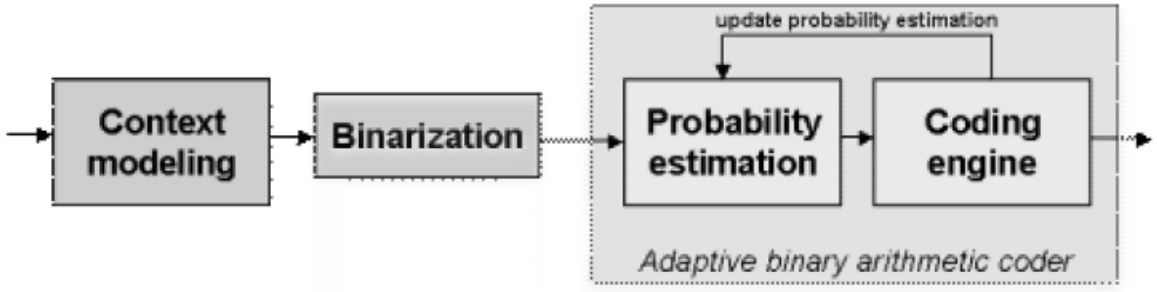


Figure II.8 Schéma synoptique du codage CABAC [55]

II.4.2 Codage des tranches de type P (Slice P):

Avec ce type de tranches, une prédiction temporelle est utilisée en estimant le mouvement entre les images.

II.4.2.1 L'estimation et la Compensation de mouvement :

L'inter-prédiction est basée sur l'estimation et la compensation du mouvement en profitant des redondances temporelles qui existent entre les images consécutives. L'efficacité de H.264 dérive de l'estimation du mouvement. Le standard a gardé la plupart des méthodes utilisées dans les normes antérieures et il a ajouté la flexibilité et d'autres fonctionnalités. Il existe plusieurs techniques d'estimation et de compensation de mouvement dont les plus connues sont celles par correspondance de blocs (BMA : *Block Matching Algorithm*). Ces techniques sont largement utilisées dans les codeurs vidéo normalisés. En effet, elles garantissent le meilleur compromis entre complexité et efficacité de codage, et une plus grande facilité d'implantation [56]. C'est pourquoi, nous considérons par la suite uniquement ces méthodes.

a- L'estimation de mouvement :

A chaque macrobloc de l'image, on associe une information de mouvement. L'opération d'estimation de mouvement permet de déterminer dans l'image de référence le macrobloc qui rassemble le plus au macrobloc à coder. Cet algorithme de recherche n'est pas normalisé et son efficacité a une influence fondamentale sur la performance du codeur. La méthode la plus utilisée est le *block-matching* comme indique la figure II. 9 : le macrobloc est comparé avec les macroblocs pointés par les vecteurs testés dans la zone de recherche de l'image de référence. Le vecteur est en général déterminé avec une précision d'un demi-pixel jusqu'au ¼ pixel. La sélection est faite sur le macrobloc qui minimise la somme des valeurs absolues des différences entre les pixels du bloc courant et le bloc de référence.

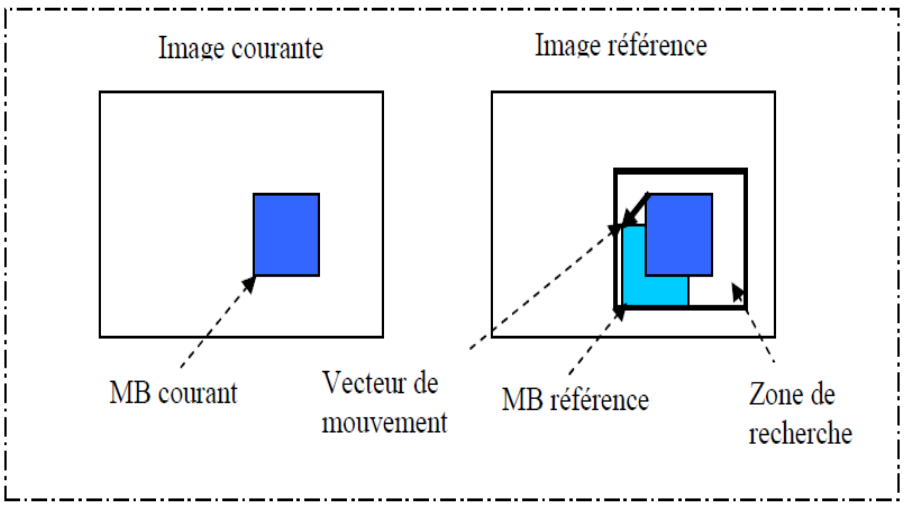


Figure II. 9 Principe de l'estimation de mouvement.

b- La compensation de mouvement : C'est la formation du nouveau macrobloc à partir de l'image référence de façon à permettre la même opération de compression dans le décodeur où seules les images décodées sont disponibles. Il diffère des normes précédentes en proposant une grande variété de formes et de tailles de blocs.

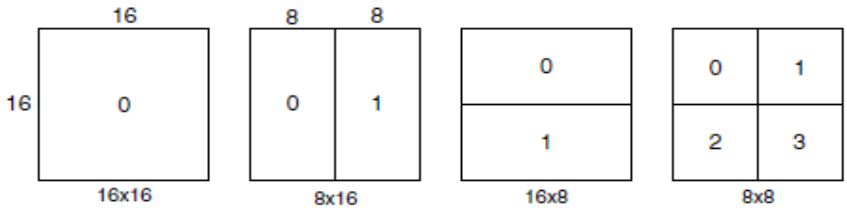
La composante luminance de chaque macrobloc (16x16 pixels) peut être divisée en sous partitions de quatre manières différentes comme illustré par les figures II.10 et II.11, et compensée en mouvement :

- une partition 16×16,
- deux partitions 16×8,
- deux partitions 8x16, ou
- quatre partitions 8x8

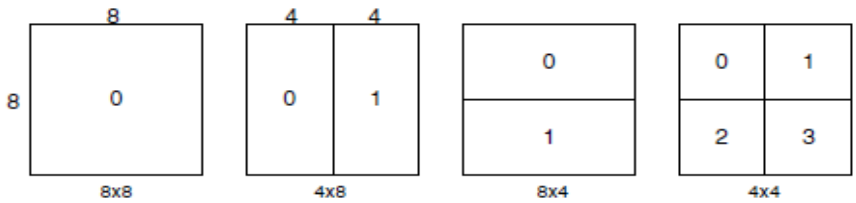
Si le mode 8×8 est choisi, chaque sous bloc peut encore être partagé de quatre manières différentes :

- un sous bloc 8 × 8,
- Deux sous blocs 8 × 4,
- Deux sous blocs 4×8, ou
- Quatre sous blocs 4×4.

Cette caractéristique permet de s'adapter plus finement au contenu spatial et au mouvement des images, elle permet de diminuer plus que 16% le débit binaire comparé à celui utilisant seulement les blocs 16x16.



(a) Partitions



(b) Sous partitions

Figure II.10 Exemple de partition avec blocs à taille variable.

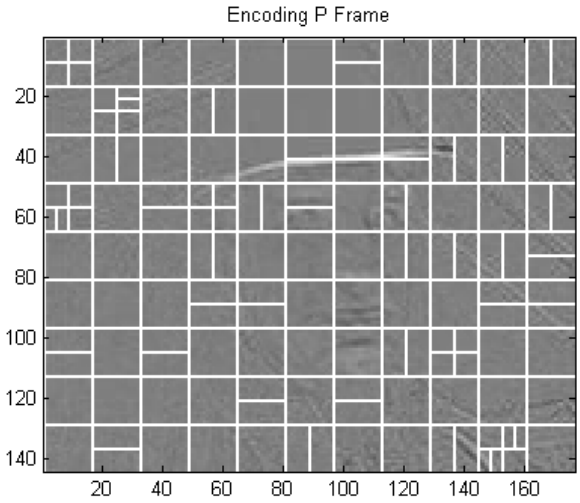


Figure II.11 Codage d'une image (frame) en utilisant différentes partitions.

Un vecteur de mouvement est nécessaire pour chaque partition ou sous partition. Le choix de la partition et le vecteur de mouvement sont codés et transmis dans le flux binaire compressé.

Choisir une partition de taille 16×16, 16×8, ou 8×16 nécessite un faible nombre de bits pour indiquer le type de partition et le vecteur de mouvement mais les résidus compensés en mouvement peuvent contenir une quantité importante d'énergie dans les zones d'images renfermant beaucoup de détails.

Une partition de petite dimension (8×4, 4×4, etc.) peut, par contre, donner lieu à des résidus de faible énergie mais requiert un grand nombre de bits pour signaler les vecteurs de mouvement et le choix de(s) partition(s).

Le choix de la taille des partitions a donc un impact significatif sur la performance de la compression.

En général, une grande taille de la partition convient pour coder les régions homogènes de l'image et les petites partitions sont bénéfiques pour les zones à détails.

Les composantes de chrominance (Cb et Cr) sont partitionnées de la même façon que la composante de luminance, à l'exception que les partitions ont la moitié de la résolution horizontale et verticale. Par exemple, une partition 8x16 de la luminance correspond à une partition 4x8 de la chrominance.

Le mouvement est estimé à partir de plusieurs images références et la sélection de l'image de référence se fait au niveau de la partition. A cet effet, différentes partitions d'un même macrobloc utiliseront la même image de référence. La limite sur le nombre d'images de référence permises pour l'estimation du mouvement est spécifiée pour chaque niveau du standard.

c) **Mesure de l'énergie:**

La méthode de correspondance de blocs vise à minimiser un certain critère de distorsion (erreur quadratique, erreur absolue…). Le meilleur choix correspond au bloc qui minimise l'énergie des résidus compensés en mouvement. Le choix de l'opérateur de distorsion n'est pas imposé par les normes des comités de standardisation (MPEG et UIT-T). Trois méthodes de mesure de l'énergie sont utilisées (l'Erreur Quadratique Moyenne ou MSE, l'Erreur Absolue Moyenne ou MAE et la Somme des Valeurs Absolues des Différences (SAD : *Sum of Absolute Differences ou SAE)*, mais la SAE sur la luminance des pixels est exploitée le plus souvent [57] pour sa simplicité.

Dans les expressions suivantes, N représente la taille du bloc, C_{ij} et R_{ij} sont respectivement les échantillons actuels et les échantillons de référence :

$$\text{MSE} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(C_{ij} - R_{ij})^2 \quad (9)$$

$$\text{MAE} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right| \quad (10)$$

$$\text{SAE} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right| \quad (11)$$

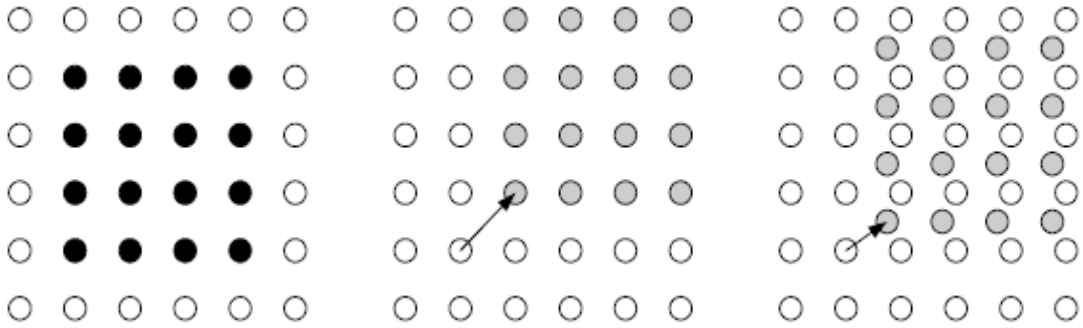
d) **Les vecteurs de mouvement** : [47]

Chaque partition ou sous partition d'un macrobloc codé en mode inter, est prédite à partir d'une zone de la même taille d'une image référence.

Pour estimer le vecteur de mouvement représentant l'offset entre les deux zones, les valeurs des pixels sont d'abord interpolées pour atteindre une précision au quart de pixel pour la composante de luminance et jusqu'au huitième du pixel pour la composante de chrominance. Les échantillons luminance et chrominance n'existent pas aux positions sous-pixéliques dans l'image référence et il est alors nécessaire de les créer par interpolation des échantillons voisins codés.

Dans la figure II.12, un bloc 4×4 de l'image courante (a) est prédit à partir d'une zone de l'image de référence au voisinage de la position courante du bloc.

Si les composants horizontaux et verticaux du vecteur de mouvement sont entiers (b), les échantillons importants du bloc de référence existent (points gris). Mais, si l'un ou les deux composants du vecteur de mouvement sont des valeurs réelles (c), les échantillons prédits sont générés par interpolation des échantillons adjacents de l'image de référence (points blancs).



(a) bloc 4x4 de l'image courante *(b) Bloc de référence: vecteur (1, -1)* *(c) Bloc de référence: vecteur (0.75, -0.5)*

Figure II.12 Exemple de prédiction entière et sous-pixélique.

- Génération des échantillons interpolés:

Les échantillons demi-pixels de la composante de luminance de l'image de référence sont les premiers à être générés. Chaque échantillon demi-pixel adjacent à deux échantillons entiers est interpolé en utilisant un filtre à réponse impulsionnelle finie RIF (ou FIR en anglais) à six coefficients : [1, −5, 20, 20, −5, 1]/32.

Par exemple, l'échantillon b dans la figure II.13 est calculé à partir des six échantillons entiers horizontaux E, F, G, H, I et J : $b = round(\frac{E-5f+20G+20H-5I+J}{32})$

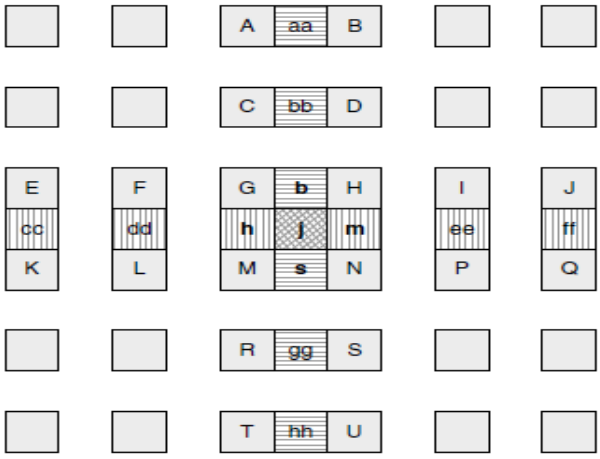


Figure II.13 Interpolation des échantillons demi-pixel

Similairement, h est interpolé par filtrage des échantillons A, C, G, M, R et T. Une fois tous les échantillons adjacents à des échantillons entiers horizontalement et verticalement ont été calculés, le reste des positions demi-pixels sont déterminées par interpolation des six échantillons demi-pixels horizontaux ou verticaux issus de la première opération. L'échantillon j, par exemple, est généré par filtrage de cc, dd, h, m, ee et ff.

Les échantillons aux positions quart de pixel sont déterminés par interpolation linéaire (figure II.14).

Les positions quart de pixel ayant deux échantillons demi-pixel adjacents horizontalement ou verticalement (a, c, i, k et d, f, n, q dans la figure II.14) sont interpolés linéairement entre ces échantillons adjacents. Par exemple, $a = round(\frac{G+b}{2})$

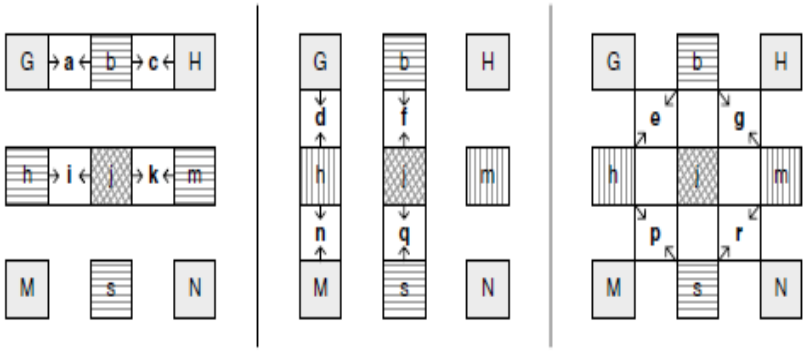


Figure II.14 Interpolation des échantillons quart de pixel.

Les autres positions quart de pixel (par exemple, e, g, p et r dans la figure II.14) sont linéairement interpolées entre une paire d'échantillons demi-pixel diagonalement opposés. Par exemple, e est interpolé entre b et h.

Les vecteurs de mouvement à résolution au quart de pixel exigent des vecteurs avec une résolution au huitième de pixel pour la composante chrominance, si on considère le format 4:2:0. Les échantillons interpolés sont générés à des intervalles au huitième d'échantillon entre les échantillons entiers dans chaque composante de chrominance par une interpolation linéaire (voir figure II.15).

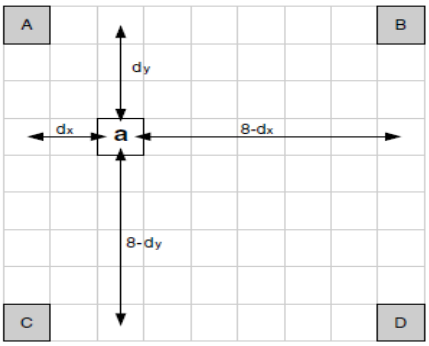


Figure II.15 Interpolation des positions huitième de pixel (composante de chrominance)

Chaque position d'un sous échantillon a est une combinaison linéaire des positions des échantillons entiers voisins A, B, C et D:

$$a = round(\frac{[(8-dx).(8-dy)A + dx.(8-dy)B + (8-dx).dyC + dx.dyD]}{64})$$

Dans la figure II.15, d*x* = 2 et d*y* = 3, on obtient alors:

$$a = round[\frac{30A + 10B + 18C + 6D}{64}]$$

Après la prédiction temporelle, les données résiduelles (originales – valeurs des pixels prédites) subissent les mêmes étapes que les tranches I, à savoir transformée, quantification et codage entropique. Les vecteurs de mouvement sont aussi compressés.

II.4.3 Codage des tranches de type B (Slice B) :

Chaque partition d'un macrobloc code en mode inter dans une tranche de type B peut être prédite d'une ou de deux images références, située (s) temporellement avant ou après l'image actuelle.

La figure suivante montre trois exemples :

a) Une image référence passée et une future (technique utilisée dans les normes antérieures),

b) Deux images références passées,

c) Deux images références futures.

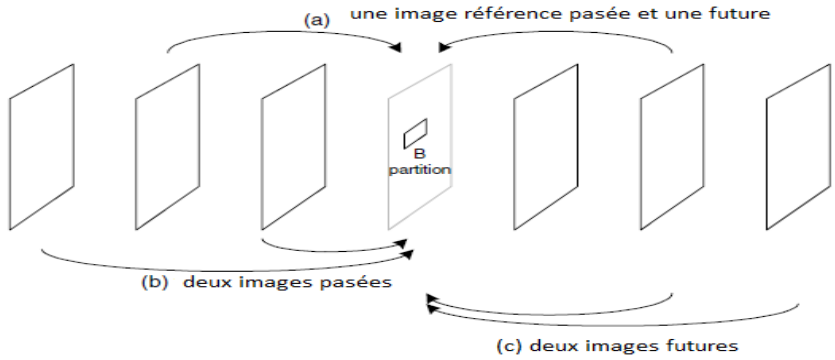


Figure II. 16 Exemple de prédiction pour les macroblocs de type B

II.4.3.1 Les images références :

Les tranches B utilisent deux listes d'images références codées précédemment. La liste 0 qui contient les images à court terme et la liste 1 qui contient les images à long terme.

Les deux listes peuvent contenir des images codées passées ou futures. L'ordre d'indexation, par défaut, est comme suit:

Liste 0: L'indice 0 est affecté à l'image passée la plus proche, en se basant sur la numérotation des images. Elle est suivie par toute autre image passée selon l'ordre croissant des images, puis par toute image future selon l'ordre croissant à partir de l'image courante.

Liste 1: L'indice 0 est affecté à l'image future la plus proche, suivie par toute image future selon l'ordre croissant des images, puis par toute image passée selon, toujours l'ordre croissant.

Exemple: Un décodeur H.264 stocke six images références à court terme dans l'ordre : 123, 125,126, 128, 129, 130. L'image actuelle est 127. Les six images références sont utilisées comme références dans les listes 0 et la liste 1. La table II.7 montre leurs indices dans le buffer.

Indice	Liste 0	Liste 1
0	126	128
1	125	129
2	123	130
3	128	126
4	129	125
5	130	123

Table II.7 Indices du buffer pour l'image courante 127

II.4.3.2 Options de prédiction :

Il existe plusieurs façons de prédire les macroblocs dans les slices de type B :

- ✓ Mode direct
- ✓ Prédiction compensée en mouvement utilisant pour référence les images de la liste 0,
- ✓ Prédiction compensée en mouvement utilisant pour référence les images de la liste 1,
- ✓ Bi-prédiction compensée en mouvement utilisant les images références de la liste 0 et la liste 1.

Différents modes de prédiction peuvent être sélectionnés pour chaque partition tel qu'il est indiqué dans la table II.8. Mais, si la taille de la partition est 8x8, alors le mode choisi doit être appliqué à toutes ses sous-partitions.

Taille de la partition	Options
16x16	Direct, liste 0, liste1 ou bi-prédictive
16x8 ou 8x16	Liste 0, liste 1 ou bi-prédictive
8x8	Direct, liste 0, liste1 ou bi-prédictive

Table II.8 Options de prédiction des macroblocs de type B.

II.5 Le filtre anti-blocs :

Il est connu que les traitements par bloc tels que la prédiction, la transformée et la quantification, introduisent des distorsions au niveau des contours des blocs. Ceci dégrade alors la qualité objective et subjective du flux vidéo. Pour éliminer ce type d'artefact, deux schémas de filtrage anti-blocs ont été proposés : le post filtrage et le filtrage effectué dans la boucle de codage (in-loop filtering). Ce dernier est utilisé dans la norme H.264/AVC comme l'illustre la figure II.17

Le filtre anti-blocs est appliqué après la transformée inverse dans le codeur (avant la reconstruction du macrobloc pour les prédictions futures), et avant la reconstruction du macrobloc dans le décodeur. Le codeur peut, cependant, désactiver l'opération de filtrage.

Les avantages du filtrage dans la boucle sur le post-filtrage sont discutés dans [56].

Les résultats expérimentaux ont montré que le filtrage dans la boule réduit le débit binaire de 5 à 10% tout en produisant la même qualité objective que la vidéo non filtrée [59]. Cependant, la complexité calculatoire du filtre dans la norme H.264/AVC est très grande à cause, principalement, des schémas d'accès aux données et la haute adaptabilité. Il consomme facilement le tiers de la complexité calculatoire du décodeur [[60],[61]].

Plusieurs paramètres ainsi que les caractéristiques locales de l'image contrôlent le processus de filtrage. Ces paramètres incluent les éléments syntaxiques tels que le paramètre de quantification, le "boundary strength", représentant la différence de deux blocs de chaque coté du bord, et le gradient des échantillons à travers les bords. Tous les seuils de filtrage dépendent du quantificateur puisque les artefacts sont toujours plus sévères lorsqu'une quantification grossière est menée.

L'implémentation hardware du filtre adaptatif peut être réalisée par l'implémentation de plusieurs filtres indépendants et d'un circuit de sélection. Cette sélection dépend des paramètres basés sur des tests sur les contenus.

L'accès aux données est une autre source de la complexité du filtre anti-blocs. Le filtrage est appliqué aux bords horizontaux et verticaux de chaque bloc 4x4 de l'image, excepté pour les bords limites des tranches, en plus de l'accès à chaque échantillon de l'image comme indiqué par la figure II. 17.

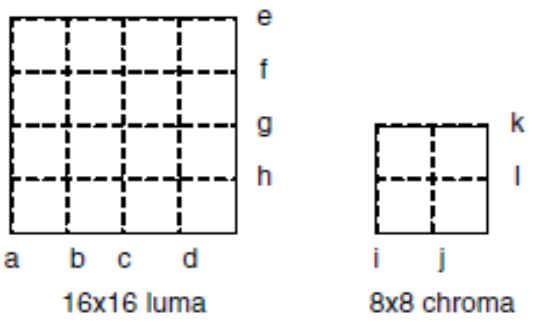


Figure II.17 Les bords à filtrer dans un macrobloc.

Le filtrage est appliqué selon l'ordre suivant :

1. Filtrer 4 bords verticaux de la composante de luminance (a, b, c, d)
2. Filtrer 4 bords horizontaux de la composante de luminance (e, f, g, h)
3. Filtrer 2 bords verticaux de chaque composante de chrominance (i, j).
4. Filtrer 2 bords horizontaux de chaque composante de chrominance (k, l).

L'opération de filtrage affecte jusqu'à trois échantillons de chaque coté des bords. La figure II.18 Montre quatre échantillons de chaque coté des bords verticaux ou horizontaux dans des blocs adjacents.

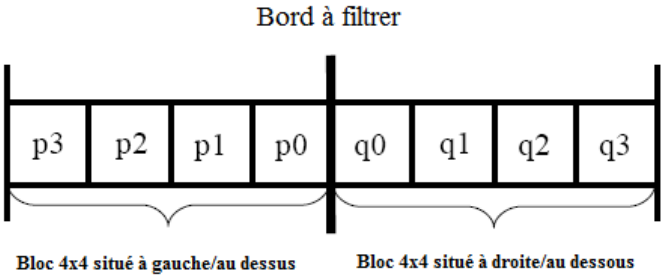


Figure II.18 Convention de nomination des pixels des bords des blocs impliqués dans l'opération de filtrage.

II.5.1 "Boundary strength" :

Le choix du filtrage dépend du "boundary strength" ou bS et du gradient des échantillons de l'image autour du bord. Le paramètre bS est choisi selon les règles suivantes :

- Si p et/ou q est codé en mode intra et qu'il s'agit du bord d'un macrobloc, alors bS = 4 correspondant au filtrage le plus fort.
- Si p et q sont codés en mode intra et qu'il ne s'agit pas du bord d'un macrobloc, alors bS = 3.
- Si ni p ni q n'est codé intra et que p et q contiennent des coefficients codés, alors *b*S = 2.

- *b*S = 1 si ni p ni q n'est codé intra et que les blocs utilisent des images de référence différentes ou un nombre différent frames de référence.
- Autrement, *b*S = 0 (pas de filtrage).

II.5.2 Décision de filtrage : [47]

Un groupe d'échantillons de l'ensemble (p2, p1, p0, q0, q1, q2) est filtré si et seulement si :

(a) BS >0 et

(b) $|p0-q0| < \alpha$ *et* $|p1-p0| < \beta$ *et* $|q1-q0| \leq \beta$.

α et β sont des seuils définis dans le standard qui croient proportionnellement au paramètre de quantification moyen des deux blocs p et q.

Le filtre est désactivé dans le cas d'une variation importante à travers le bord du bloc. Avec un faible QP, tout aspect autre qu'un faible gradient à travers le bord est probablement dû aux caractéristiques de l'image plutôt qu'un effet de bord et doit donc être préservé. Dans ce cas, les seuils α et β sont bas. Les distorsions de blocs sont plus accentuées avec un fort QP et par conséquent les valeurs de α et β sont plus grandes pour permettre le filtrage d'un nombre plus important d'échantillons des bords.

II.5.3 Implémentation du filtre: [47]

a) bS ∈{1, 2,3}

Un filtre à 4 coefficients est appliqué aux entrées p1, p0, q0 et q1, produisant les sorties filtrées p'0 et q'0.

Si $|p2 - p0| < \beta$, un filtre à 4 coefficients est appliqué aux entrés p2, p1, p0 et q0, produisant la sortie filtrée p'1 (uniquement la luminance).

Si $|q2-q0| < \beta$, un filtre à 4 coefficients est appliqué aux entrées q2, q1, q0 et p0, produisant la sortie filtrée q'1 (uniquement la luminance).

b) bS = 4

SI $|p2 - p0| < \beta$ et $|p0 - q0| <$round $(\alpha/4)$ et qu'il s'agit d'un bloc de la composante de luminance :

La sortie filtrée p'0 est produite par application d'un filtre à 5 coefficients aux entrées p2, p1, p0, q0 et q1,

p'1 est produit par application d'un filtre à 4 coefficients aux entrées p2, p1, p0 et q0,

p'2 est produit par application d'un filtre à 5 coefficients aux entrées p3, p2, p1, p0 et q0,

SINON:

p'0 est produit par application d'un filtre à 3 coefficients aux entrées p1, p0 et q1.

si|q2 − q0| < β et |p0 − q0| <round (α/4) et que c'est un bloc de luminance :

q'0 est produite par application d'un filtre à 5 coefficients aux entrées q2, q1, q0, p0 et p1,

q'1 est produite par application d'un filtre à 4 coefficients aux entrées q2, q1, q0 et p0,

q'2 est produite par application d'un filtre à 5 coefficients aux entrées q3, q2, q1, q0 et p0,

Sinon:

q'0 est produite par application d'un filtre à 3 coefficients aux entrées q1, q0 et p1.

II.6 Conclusion :

Dans ce chapitre, les aspects les plus importants du standard de compression Vidéo H.264/AVC ont été présentés et détaillés. Les différents profils et niveaux ont été montrées. La compensation de mouvement, effectuée avec une précision au quart de pixel et même au huitième de pixel pour la chrominance, et pouvant utiliser plusieurs images de référence déjà codées ainsi que sept tailles de blocs différentes a été exposée. L'algorithme du filtrage anti-blocs, effectué dans la boucle de codage sur des blocs de taille 4x4 et permettant de réduire les artefacts a été développé. Les codages CAVLC et CABAC ont aussi été présentés. Enfin, les transformées ont été brièvement discutées puisqu'elles seront abordées en détails dans le prochain chapitre.

Néanmoins, il faut souligner que ce chapitre n'est pas exhaustif et que d'autres fonctionnalités existent. Toutes ces caractéristiques sont à l'origine de la popularité et aussi de la complexité de cette norme.

Chapitre III : les transformées

III.1 Introduction :

L'objectif de l'étape de transformation dans un codeur image ou vidéo est de convertir l'image d'un domaine vers un autre. Le choix d'une transformée dépend d'un certain nombre de critères :

1. Les données dans le domaine transformé doivent être décorrélées (séparées en composants ayant un minimum d'inter dépendance), et compactes (la majeure partie de l'énergie des données transformées doit être concentrée dans un petit nombre de coefficients).
2. La transformée doit être réversible.
3. La transformée doit être sans complexité calculatoire (nécessite une petite mémoire, faisable en utilisant une arithmétique à précision limitée, requière un nombre réduit d'opérations arithmétique, etc.)

De nombreuses transformées ont été proposées pour la compression d'images et de vidéos. Les plus connues peuvent être classées dans deux catégories : basées sur des blocs et basées sur des images.

Dans la première classe, se trouve la très populaire DCT qui opère sur des blocs de taille NxN. Les transformées basées sur des blocs n'exigent pas de grande capacité mémoire et conviennent parfaitement à la compression des résidus compensés en mouvement à base de blocs, mais leur inconvénient majeur est l'apparition d'artefacts au niveau des contours des blocs.

Les transformées à base d'images traitent l'image entière ou le frame entier (cas des vidéos). La transformée la plus connue de cette catégorie est la DWT ou simplement la transformée en ondelettes qui a montré, grâce à ses capacités d'analyse multi-résolution, une meilleure efficacité que les transformées à base de blocs mais présente l'inconvénient d'être très gourmande en mémoire.

La DCT et la DWT sont utilisées dans la norme MPEG-4 Visual et une variante de la DCT est incorporée dans le standard H.264 où elle est associée à la transformée de Hadamard. Ces techniques sont largement discutées dans les sections suivantes.

III.2 La transformée de Hadamard discrète (DHT) :

La transformée de Hadamard discrète [62] est définie par les équations suivantes:

$$DHT(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{n}(-1)^{p_i(x)p_i(u)}\,(-1)^{\,p_i(y)p_i(v)} \qquad (12)$$

et $$iDHT(x,y,u,v) = DHT(x,y,u,v) \qquad (13)$$

DHT étant la transformée directe et iDHT la transformée inverse.

$n=\log_2(N)$ et p_i (arg) représente le $i^{ème}$ bit de la représentation binaire naturelle de l'argument "arg", le bit à la position 0 étant le moins significatif et le $(n-1)^{ème}$ correspond au bit le plus significatif. Si on considère, par exemple, N=16, on a n=4. Le code binaire de 8 étant égal à 1000, alors $p_0(8)= p_1(8)= p_2(8)=0$ et $p_3(8)=1$.

Il est évident que si on ne considère pas le facteur 1/N, le noyau de la transformée est toujours un entier égal à ±1. En plus, les transformées directe et inverse sont identiques, d'où la simplicité de cette transformée.

Une propriété particulière et utile de la transformée de Hadamard est que les matrices d'ordre élevé peuvent être générées récursivement des matrices d'ordre inférieur. Par exemple, la matrice de Hadamard 2Nx2N peut être obtenue à partir de la matrice NxN :

$$\mathrm{DHT}_{2N} = \begin{bmatrix} DHT_N & DHT_N \\ DHT_N & -DHT_N \end{bmatrix} \qquad (14)$$

La figure suivante montre un ensemble de 16 images de base de la transformée de Hadamard obtenues pour N=4. Ces fonctions de base sont rectangulaires et non pas sinusoïdales comme c'est le cas de la DCT par exemple.

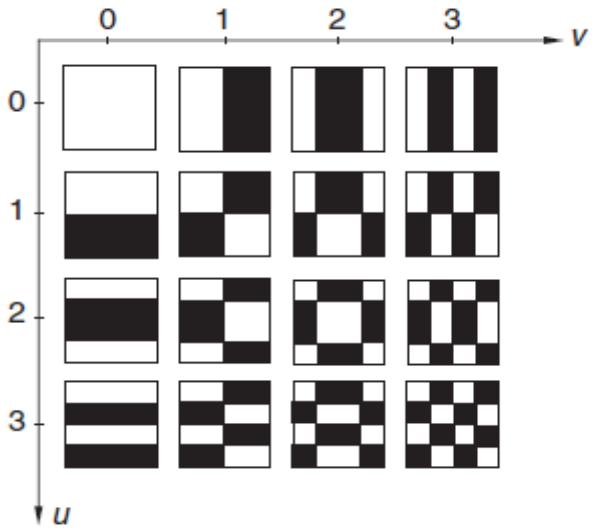


Figure III.1 Ensemble des 16 images de base de la DHT (N=4)

III.3 La transformée en cosinus discrète (DCT) :

La DCT qui joue un rôle d'une importance extrême dans la compression d'images et de vidéos a été établis par N.Ahmed et al. en 1974 [63].

La transformée en cosinus discrète opère sur un bloc X, de taille NxN échantillons qui sont soit des échantillons de l'image ou des valeurs différentielles obtenues après prédiction, et produit un bloc Y de coefficients, de même taille NxN. La DCT directe et la DCT inverse (note IDCT) peuvent être décrites en fonction de la matrice de transformation A de taille NxN.

La DCT directe est décrite par : $Y = AXA^T$ (15)

La DCT inverse est donnée par : $X = A^T YA$ (16)

Les éléments de la matrice A sont donnés par l'équation suivante:

$$A_{ij} = C_i \, Cos(2j+1)i\pi/2N$$

avec
$$\begin{cases} C_i = \sqrt{1/N} \ (i=0) \\ C_i = \sqrt{2/N} \ \ (i>0) \end{cases} \quad (17)$$

Les équations 15 et 16 peuvent être réécrites sous forme de sommation:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cdot \cos\left(\frac{(2j+1)y\pi}{2N}\right) \cos\left(\frac{(2i+1)x\pi}{2N}\right) \quad (18)$$

$$X_{ij} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_x C_y Y_{xy} \cdot \cos\left(\frac{(2j+1)y\pi}{2N}\right) \cos\left(\frac{(2i+1)x\pi}{2N}\right) \quad (19)$$

Le résultat d'une DCT directe à deux dimensions est une matrice NxN de coefficients représentant l'image dans le domaine fréquentiel. Ces coefficients peuvent être considérés comme des "poids" d'un ensemble de bases standards et les figures III.2 et III.3 en représentent les cas 4x4 et 8x8 respectivement. Il s'agit de combinaisons de fonctions cosinus horizontales et verticales. Le bloc de l'image en question peut alors être reconstruit par combinaison de toutes les bases pondérées par le correspondant facteur (le coefficient).

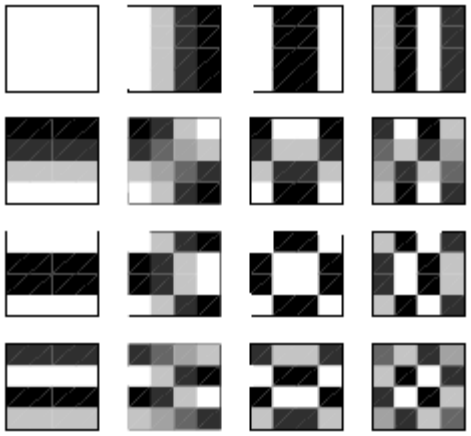


Figure III.2 Les images de base de la DCT 4x4.

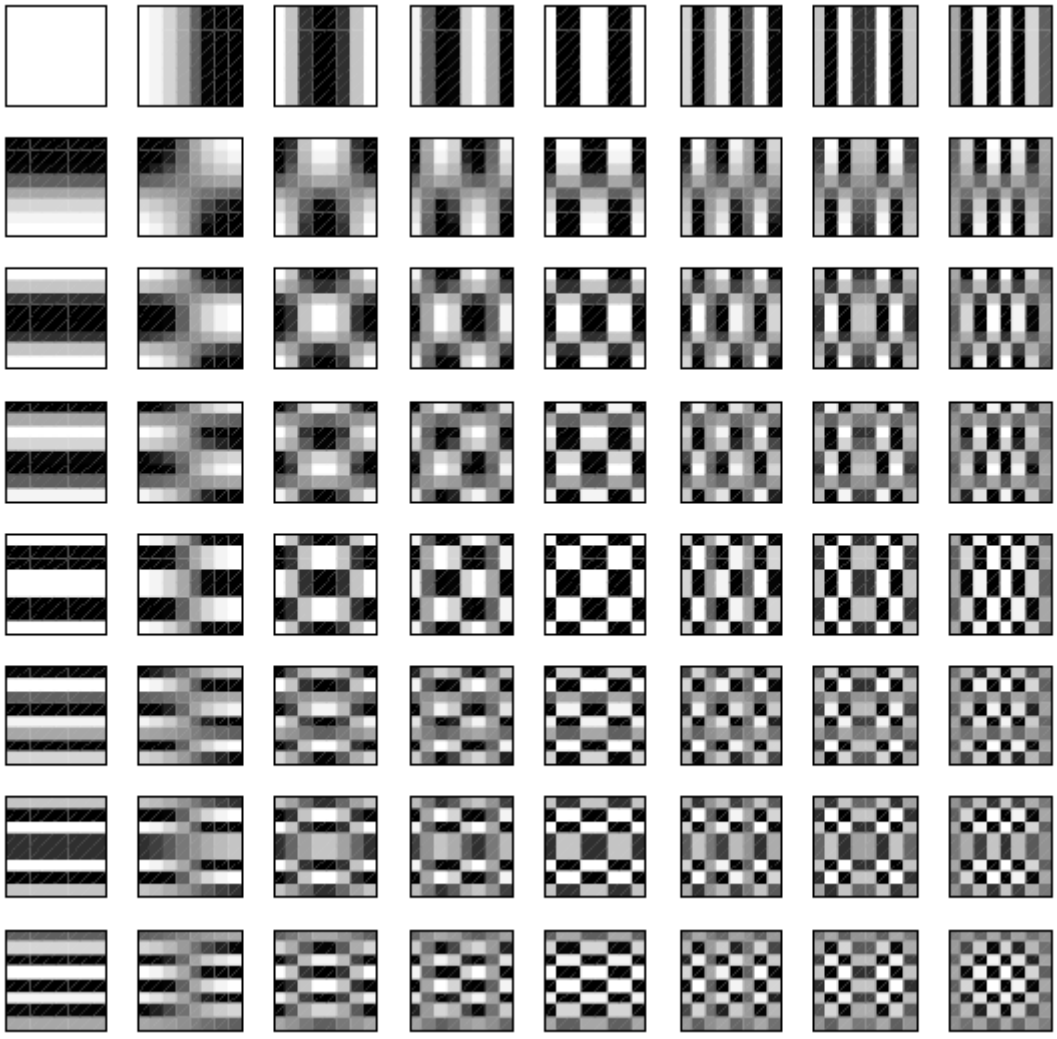


Figure III.3 Les images de base de la DCT 8x8.

III.4La transformée DCT dans le contexte du H.264:

Tous les standards vidéos précédents tels que MPEG1, MPEG2, MPEG4 Part2, JPEG, H.261, H.263 utilisent la DCT sur des blocs de taille 8x8. Dans la norme H.264, la taille du bloc utilisé est 4x4. Une des raisons de ce choix est la réduction de la complexité de l'implémentation.

La norme H.264 utilise trois transformées qui dépendent du type des données résiduelles à coder:

- Une transformée de Hadamard pour les matrices 4×4 des coefficients DC de la composante de luminance dans les macroblocs codés intra en mode 16×16,
- Une transformée de Hadamard pour les matrices 2 × 2 des coefficients DC de la composante de chrominance quelque soit le macrobloc, et
- Une transformée en cosinus discrète pour tous les autres blocs des données résiduelles.

Les données d'un macrobloc sont transmises selon l'ordre indiqué dans la figure III.4.

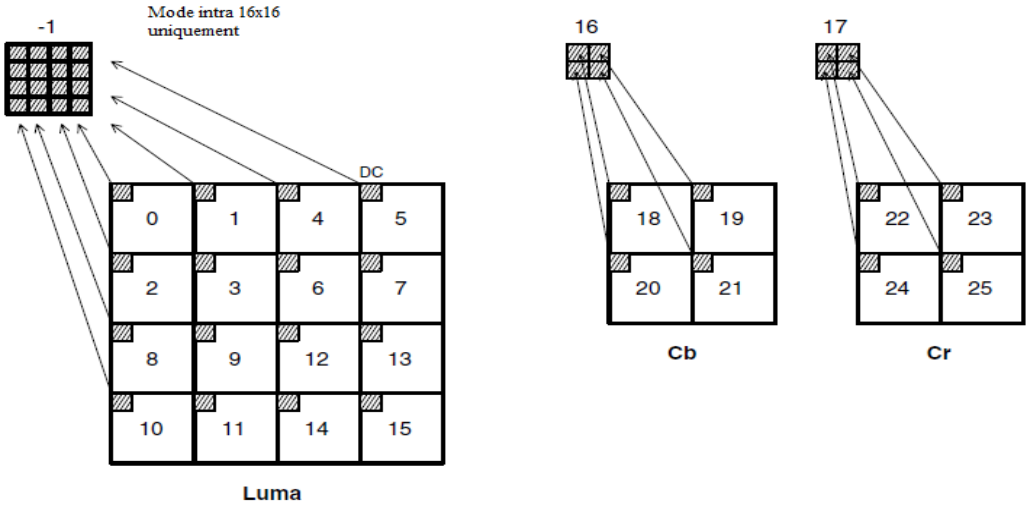


Figure III.4. Ordre de parcours des blocs résiduels d'un macrobloc.

Si le macrobloc est codé en mode intra 16x16, alors le bloc marqué "-1", contenant les coefficients DC de chaque bloc 4x4 de la composante de luminance, est transmis en premier.

Ensuite, les blocs de luminance résiduels 0-15 sont transmis dans l'ordre montré. Les blocs 16 et 17 formés des coefficients DC (matrice 2x2) des composantes de chrominance Cb et Cr sont envoyés. Enfin, les blocs 18-25 (sans les coefficients DC) sont transmis.

III.4.1Transformée des résidus 4 × 4 (blocs 0–15, 18–25) :

Cette transformée opère sur des blocs de taille 4x4 composés des données résiduelles notés 0-15 et 18-25 dans la figure III.4, après une prédiction compensée en mouvement ou une prédiction intra. La transformée est basée sur la DCT mais avec quelques différences fondamentales [64]:

1. C'est une transformée entière où toutes les opérations sont menées en utilisant l'arithmétique entière sans perte de précision lors de l'opération de décodage.
2. Il est possible d'assurer une divergence nulle entre le codeur et le décodeur qui réalise les transformées inverses en utilisant une arithmétique entière.
3. La partie principale de la transformée peut être implémentée moyennant des opérations d'additions et de décalage uniquement.
4. Une multiplication de mise à l'échelle (partie de la transformée) est intégrée au quantificateur, réduisant ainsi le nombre total de multiplications.

La DCT 4×4 DCT est donnée par:

$$Y = AXA^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} [X] \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \quad (20)$$

Où : $a = \frac{1}{2}$, $b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8})$, $c = \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8})$

Le produit matriciel (17) peut être reformulé pour donner la forme suivante:

$$Y = (CXC^T) \otimes E = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} [X] \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \quad (21)$$

CXC^T est un noyau de la transformée 2D. E est une matrice de facteur de pondération et le symbole $\otimes$ indique que chaque élément de (CXC^T) est multiplié par le facteur correspondant de la matrice E, c.à.d. qu'il s'agit d'une multiplication scalaire et non pas d'une multiplication matricielle. Les constantes a et b ont les mêmes valeurs que précédemment et d est égal à c/b, soit approximativement 0.414.

Pour simplifier l'implémentation de la transformée, d est pris égal à 0.5 et b doit être modifié en conséquence afin d'assurer l'orthogonalité de la transformée, on obtient alors:

$a = \frac{1}{2}$, $b = \sqrt{\frac{2}{5}}$, $d = \frac{1}{2}$

La 2ème et la 4ème ligne de la matrice C et la 2ème et 4ème colonne de la matrice C^T sont pondérées par un facteur de deux et la matrice E subit une pondération inverse de façon à compenser et éviter des multiplications par ½ dans le noyau de la transformée CXC^T qui pourrait résulter en une perte de précision en utilisant l'arithmétique entière. Finalement, la transformée directe devient:

$$Y=(C_fXC_f^T)\otimes E_f=\left(\begin{bmatrix}1&1&1&1\\2&1&-1&-2\\1&-1&-1&1\\1&-2&2&-1\end{bmatrix}[X]\begin{bmatrix}1&2&1&1\\1&1&-1&-2\\1&-1&-1&2\\1&-2&1&-1\end{bmatrix}\right)\begin{bmatrix}a^2&ab/2&a^2&ab/2\\ab/2&b^2/4&ab/2&b^2/4\\a^2&ab/2&a^2&ab/2\\ab/2&b^2/4&ab/2&b^2/4\end{bmatrix} \quad (22)$$

Cette transformée est une approximation de la DCT 4x4 mais avec les modifications apportées aux facteurs d et b, le résultat de la nouvelle transformée sera identique à celui de la DCT 4x4.

L'exemple suivant [47] permet de juger l'ordre de grandeur de l'erreur commise en utilisant la transformée DCT approximative.

Le bloc 4x4 à traiter :

$$X=\begin{bmatrix}5&11&8&10\\9&8&4&12\\1&10&11&4\\19&6&15&7\end{bmatrix}$$

Résultat de l'application de la transformée DCT :

$$Y=AXA^T=\begin{bmatrix}35.0&-0.079&-1.5&1.115\\-3.299&-4.768&0.443&-9.010\\5.5&3.029&2.0&4.699\\-4.045&-3.010&-9.384&-1.232\end{bmatrix}$$

Résultat de l'application de la transformée DCT approximative :

$$Y'=(C_fXC_f^T)\otimes E_f=\begin{bmatrix}35.0&-0.158&-1.5&1.107\\-3.004&-3.900&1.107&-9.200\\5.5&2.688&2.0&4.901\\-4.269&-3.200&-9.329&-2.100\end{bmatrix}$$

La différence entre les deux transformées est :

$$Y-Y'=\begin{bmatrix}0&0.079&0&0.008\\-0.295&-0.868&-0.664&0.190\\0&0.341&0&-0.203\\0.224&0.190&-0.055&0.868\end{bmatrix}$$

Il est évident qu'il y a une différence entre les résultats des deux transformées. Cette erreur dépend de b ou de d. Mais, dans le contexte du codec H.264, la transformée approximative conduit aux mêmes performances de compression que la DCT et possède, en plus, des avantages très importants. Le noyau de la transformée CXC^T peut

être calculé en utilisant seulement des additions/soustractions et des décalages. Une arithmétique à 16 bits peut être utilisée dans tous les calculs puisque les entrées sont toujours dans l'intervalle [-255, 255].

L'opération de mise à l'échelle$\otimes$ E_f nécessite une multiplication pour chaque coefficient. Cette opération peut être absorbée par le processus de quantification.

La transformée inverse est donnée par l'équation 23. La norme H.264 [65] définit explicitement cette transformée comme une séquence d'opérations arithmétiques.

$$X= C_i^T(Y\otimes E_i)C_i = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left([Y] \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \quad (23)$$

Dans les matrices C et C^T, les facteurs ±1/2 peuvent être facilement implémentés par des décalages à droite sans perte significative en précision puisque les coefficients Y sont pré-pondérés.

Les transformées directe et inverses sont orthogonales et on peut vérifier facilement que $T^{-1}\big(T(X)\big) = X$.

Lors de l'étape de quantification directe (voir chapitre II, section II.4.1.3, équation 5), les facteurs dits de post-pondération a^2, ab/2 ou $b^2/4$ (équation 22) sont incorporés. D'abord, le bloc d'entrée X est transformé et donne un bloc de coefficients non échelonnés $W = CXC^T$, puis, chaque coefficient W_{ij} est quantifié et pondéré en une seule opération :

$$Z_{ij} = round(\frac{W_{ij}\, PF}{Qstep}) \quad (24)$$

La valeur de PF qui peut être a^2, ab/2 ou $b^2/4$ dépend de la position (i,j):

Position (i,j)	PF
(0,0), (2,0), (0,2) ou (2,2)	a^2
(1,1), (1,3), (3,1) ou (3,3)	$b^2/4$
autre	ab/2

Table III.1 Valeurs de PF en fonction de la position (i,j)

Dans le modèle de référence [66], le facteur *PF*/*Qstep* est implémenté sous forme de multiplication par un facteur MF et un décalage à droite, évitant ainsi toute opération de division et simplifiant en conséquence l'arithmétique.

$Z_{ij} = round(W_{ij}\, MF/2^{qbits})$

Avec $MF/2^{qbits} = PF/Qstep$

et $qbits = 15 + floor(\frac{QP}{6})$ (25)

Avec l'arithmétique entière, l'équation 25 peut être implémentée ainsi:

$$|Z_{ij}| = (|W_{ij}|.MF + f) \gg qbits$$

$$signe(Z_{ij}) = signe(W_{ij}) \quad (26)$$

>>est l'opérateur de décalage à droite implémentant la division. Selon le modèle de référence, f est $2^{qbits}/3$ pour les blocs codés intra et $2^{qbits}/6$ pour les blocs codés inter.

Dans le codeur de référence du standard H.264, les six premières valeurs utilisées du facteur multiplicateur MF et pour chaque position du coefficient sont données dans la table III.2.

QP	Positions (0,0),(2,0),(2,2),(0,2)	Positions (1,1),(1,3),(3,1),(3,3)	Autres positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

Table III.2 Valeurs du facteur multiplicateur MF.

La 2ème et 3ème colonne de la table correspondant aux positions des facteurs $b^2/4$ et ab/2 différent légèrement des résultats obtenus avec l'équation 22. Pour QP>5, les facteurs MF restent constants mais le diviseur 2^{qbits} croit par un facteur de deux pour chaque augmentation de 6 pour QP.

Par exemple, pour 6≤ QP ≤ 11, *qbits* = 16 et pour 12 ≤QP≤ 17, qbits=17 et ainsi de suite.

Exemple de détermination du facteur MF:

QP = 4 and (i , j) = (0,0).

Qstep = 1.0 (voir table II.5), PF = a^2= 0.25 et qbits = 15 d'où 2^{qbits} = 32768.

$MF/2^{qbits}$= PF/ Qstep, ce qui donne MF = (32768 × 0.25)/1 = 8192.

L'opération de quantification inverse est donnée par l'équation:

$$Y_{ij} = Z_{ij} Qstep \quad (27)$$

Le facteur de pré-pondération pour la transformée inverse est incorporé dans cette opération avec un facteur d'échelle de 64 afin d'éviter les erreurs d'arrondis.

$$W_{ij}' = Z_{ij} Qstep.PF.64 \quad (28)$$

$W_{i\ j}$' est un coefficient échelonné transformée par le noyau de la transformée inverse $C_i^T W C_i$ (équation 23). Les résultats de la transformée inverse sont divisés par 64 pour éliminer le facteur d'échelle. Ceci peut être implémenté uniquement par une addition et un décalage à droite.

Le standard H.264 ne spécifie pas Qstep ou PF directement. A la place, le paramètre *V=Qstep.PF*.64 est défini pour $0 \leq QP \leq 5$ (voir table III.3) et pour chaque position du coefficient de façon à ce que l'opération de pondération devient :

$$W_{ij}' = Z_{ij} V_{ij} 2^{floor\ (\frac{QP}{6})} \quad (29)$$

Le facteur $2^{floor(QP/6)}$ réalise une augmentation de la sortie par 2 à chaque accroissement de QP de 6 .

QP	Positions (0,0),(2,0),(2,2),(0,2)	Positions (1,1),(1,3),(3,1),(3,3)	Autres positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

Table III.3 Valeurs du facteur V

III.4.2 *Transformée des coefficients DC* 4 × 4 de la composante de luminance (mode intra 16 × 16 uniquement) :

Dans le cas de codage d'un macrobloc 16x16 en mode d'intra prédiction, chaque bloc résiduel 4x4 est d'abord transformé en utilisant le noyau de la transformée décrite dans le paragraphe précédent ($C_f X C_f^T$). Le coefficient DC de chaque bloc 4x4 est ensuite transformée par la transformée de Hadamard 4x4 :

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} [W_D] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} /2 \right) \quad (30)$$

W_D représente le bloc 4x4 des coefficients DC et Y_D est le bloc après transformation. Les coefficients $Y_{D(i,j)}$ sont alors quantifiés :

$$|Z_{D(i,j)}| = \left(|Y_{D(i,j)}|.MF_{(0,0)} + 2f\right) \gg (qbits + 1)$$

$$signe\left(Z_{D(i,j)}\right) = signe(Y_{D(i,j)}) \quad (31)$$

MF(0,0) est le facteur multiplicateur correspondant à la position (0,0) dans la table III.2. f et qbits ont été définis précédemment.

Au niveau du décodeur, la transformée de Hadamard inverse est appliqué et elle est suivie par le ré-échelonnage. A noter que l'ordre n'est pas inversé comme on peut s'y attendre.

$$W_{QD} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} [Z_D] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (32)$$

La remise à l'échelle appliquée au niveau du décodeur est décrite par l'équation suivante:

$$W'_{D(i,j)} = W_{QD(i,j)} V_{(0,0)} 2^{floor\left(\frac{QP}{6}\right)-2} \quad \text{pour } (QP \geq 12)$$

$$W'_{D(i,j)} = \left[W_{QD(i,j)} V_{(0,0)} + 2^{l-floor\left(\frac{QP}{6}\right)}\right] \gg \left(2 - floor\left(\frac{QP}{6}\right)\right) \quad \text{pour } (QP < 12) \quad (33)$$

Les coefficients DC pondérés W'_D sont insérés dans leurs blocs 4x4 respectifs et puis chaque bloc 4x4 subit la transformation inverse par utilisation du noyau de la transformée DCT inverse $C_i^T W' C_i$.

Dans un macrobloc 16x16 codé en mode intra, une grand partie de l'énergie est concentrée dans les coefficients DC de chaque bloc 4x4 qui ont tendance à être corrélés fortement. Après application de cette transformée supplémentaire, l'énergie est concentrée d'avantage dans un petit nombre de coefficients significatifs.

III.4.3 Transformée des coefficients DC de la composante de chrominance (2 × 2) :

Les coefficients DC de chaque bloc 4 × 4 block de la composante de chrominance sont regroupés dans un bloc 2 × 2 W_D et subissent une extra transformation avant d'être quantifiés :

$$W_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} [W_D] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (34)$$

La quantification du bloc résultant de la l'application de la transformée se fait selon l'équation 28.

Au cours du décodage, la transformée inverse est appliquée :

$$W_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} [Z_D] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (35)$$

La mise à l'échelle suit l'équation :

$$W'_{D(i,j)} = W_{QD(i,j)} V_{(0,0)} 2^{floor\left(\frac{QP}{6}\right)-1} \quad \text{pour } (QP \geq 6)$$

$$W'_{D(i,j)} = \left[W_{QD(i,j)} V_{(0,0)}\right] \gg 1 \quad \text{pour } (QP < 6) \quad (36)$$

Les coefficients remis à la bonne échelle sont repositionnés dans leurs blocs 4x4 respectifs. Ces derniers sont alors transformés ($C_i^T W' C_i$).

Cette transformation supplémentaire contribue à une décorrélation des coefficients DC 2x2 de la composante de chrominance et permet ainsi d'améliorer les performances de la compression.

III.5 La transformée en ondelettes discrète (DWT) :

III.5.1 Historique :

D'un point de vue historique, l'analyse par les ondelettes repose sur les fondements des travaux de Joseph Fourier au dix-neuvième siècle. La différence est que la théorie de Fourier est basée sur une analyse fréquentielle alors que les ondelettes sont basées sur une analyse échelle ou résolution. La première mention du terme "wavelet" ou ondelette a été en 1909, dans une thèse d'Alfred Haar [67]. Le concept d'ondelettes sous sa forme actuelle a été introduit par Jean Morlet et son équipe au centre de physique théorique à Marseille (France) sous la direction d'Alex Grossmann. Les méthodes d'analyse et transformée par ondelettes ont été développées par Y. Meyer. L'algorithme principal a été suggéré par Stephane Mallat en 1988. Les chercheurs aux états unis tels qu'Ingrid Daubechies, Ronald Coifman, et Victor Wickerhauser ont fortement contribué au lancement de la méthode [67]. Les ondelettes sont devenues un outil incontournable dans le traitement du signal, de l'image et de la vidéo depuis leur recommandation par l'ITU-T pour le standard JPEG2000 [68]. La transformée en ondelettes possède, en plus de la propriété de compacité de l'énergie, la possibilité de reconstruction progressive des images. Il existe plusieurs familles d'ondelettes dont les plus connues sont :

Famille	**Abréviation utilisée**	**Ondelettes de la famille**
Haar	Haar	Haar
Daubechies	db	db1 db2 db3 db4 db5 db6 db7 db8 db9 db10
Symlets	sym	sym2 sym3 sym4 sym5 sym6 sym7 sym8
Coiflets	coif	coif1 coif2 coif3 coif4 coif5
BiorSplines	bior	bior1.1 bior1.3 bior1.5 bior2.2 bior2.4 bior2.6 bior2.8 bior3.1 bior3.3 bior3.5 bior3.7 bior3.9 bior4.4 bior5.5 bior6.8
ReverseBior	rbio	rbio1.1 rbio1.3 rbio1.5 rbio2.2 rbio2.4 rbio2.6 rbio2.8 rbio3.1 rbio3.3 rbio3.5 rbio3.7 rbio3.9 rbio4.4 rbio5.5 rbio6.8

Table III.4 Quelques familles d'ondelettes.

III.5.2 Implémentation :

La transformée en ondelettes discrète (DWT) d'une séquence consiste à développer deux séries, l'une correspondant à l'approximation et l'autre aux détails de la séquence. La définition de la DWT d'une séquence composée de N points $x\,[n]$,$0 \le n \le N-1$ est donnée par [69]:

$$DWT\{f(t)\} = W_{\varphi}(j_0,k) + W_{\psi}(j,k) \tag{37}$$

$$W_{\varphi}(j_0,k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x[n].\varphi_{j_0,k}[n] \tag{38.a}$$

$$W_{\psi}(j,k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x[n].\psi_{j,k}[n] \tag{38.b}$$

La séquence $x\,[n]$,$0 \le n \le N-1$ peut être reconstruite à partir des coefficients de la DWT $W\varphi$ et$W\psi$ de la façon suivante :

$$x[n] = \frac{1}{\sqrt{N}}\sum_k W_{\varphi}(j_0,k).\varphi_{j_0,k}[n] + \frac{1}{\sqrt{N}}\sum_{j=j_0}^{\infty}\sum_k W_{\psi}(j,k).\psi_{j,k}[n] \tag{39}$$

Le paramètre d'échelle dans le second terme de l'addition de l'équation (39) a un nombre infini de termes. Mais, en pratique, la limite supérieure est fixée à une certaine valeur J. La valeur initiale de l'échelle j_0 est d'habitude fixée à zéro et correspond au signal original. Ainsi, les coefficients de la DWT pour x[n], $0 \le n \le N-1$ sont calculés pour $j = 0, 1, \ldots, J-1$ et $k = 0, 1, \ldots, 2j-1$. De même que N est typiquement une puissance de 2, de la forme $N = 2^J$.

La transformée en ondelettes peut être implémentée de différentes manières :

III.5.2.1 Par bancs de filtres :

La transformée en ondelettes discrètes est une représentation multi-résolutions/ multifréquences qui se base sur des bancs de filtres avec des coefficients équivalents à des fonctions d'ondelettes discrètes [70]. L'opération de base d'une transformée en

ondelettes discrète appliqué à un signal composé de N échantillons peut se résumer ainsi:

Une paire de filtres (banc de filtres) à réponse impulsionnelle finie (RIF) est appliquée au signal et le décompose en une bande de basse fréquence, et une bande de haute fréquence. Chaque bande est sous-échantillonnée par un facteur de deux de façon à ce que chaque bande constitue un signal avec N/2 échantillons. Ce type de codage en sous bandes est appelé codage en sous bandes en bandes d'octave car la zone passe bas est à chaque fois divisée en deux moitiés et c'est la phase d'analyse ou décomposition (figure III.5 (a)). Dans la phase de synthèse ou reconstruction, les procédures sont répétées dans la direction inverse en utilisant une paire, parfois différente, de filtres passe bas et passe haut (figure III.5(b)).

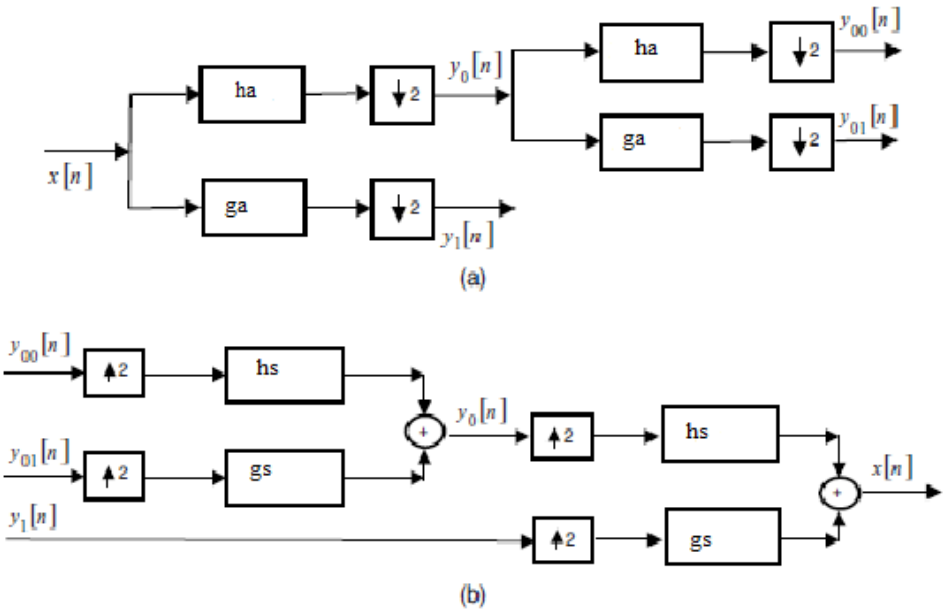


Figure III.5 Codage en sous bandes à 2 niveaux: (a) analyse, (b) synthèse.

Dans la figure III.5, (ha, ga) représente le banc de filtres d'analyse, passe bas et passe haut alors que (hs, gs) représente la paire de filtres de synthèse, passe bas et passe haut. ↓2 est l'opérateur de sous-échantillonnage par 2 et ↑2, l'opérateur de sur-échantillonnage ou interpolation par 2.

Les filtres RIF doivent assurer la réversibilité de l'opération et la complémentarité des puissances appelée aussi la propriété de Smith–Barnwell permettant d'atteindre cet objectif [[71], [72], [73], [74], [75], [76]] :

$$\left|H_s\left(e^{j\omega}\right)\right|^2 + \left|G_s\left(e^{j\omega}\right)\right|^2 = 2 \quad (40)$$

$H_s\left(e^{j\omega}\right) et G_s\left(e^{j\omega}\right)$ sont les transformées de Fourier des filtres de synthèse hs et gs respectivement. Selon le type d'ondelettes, orthogonales ou bi-orthogonales, le banc de filtres doit satisfaire un ensemble de conditions.

Cas d'ondelettes orthogonales :

La reconstruction parfaite est assurée par les conditions suivantes :

1. Le filtre est de longueur paire.
2. Les paires de filtres (hs, gs), (ha, ga), (ha, hs) et (hs, ga) doivent satisfaire la condition de complémentarité de la puissance (eq. 40).
3. Les filtres hs et ha sont inverses l'un de l'autre dans le temps : ha[n]= hs[-n].
4. Les filtres *gs* [*n*] et *ga* [*n*] sont inverses l'un de l'autre dans le temps.
5. Les filtres *ga* [*n*] et *ha* [*n*] satisfont la condition : $ga\,[n] = (-1)^{n+1} ha[L-1-n]$.
6. Les filtres *gs* [*n*] et *hs* [*n*] sont inverses dans le temps et représentent des versions modulées l'une de l'autre : $gs\,[n] = (-1)^n\, hs\,[L-1-n]$.
7. Enfin, $\sum_n h_a\,[n] = \sum_n h_s\,[n] = \sqrt{2}$.

Malgré que les filtres satisfassent les conditions précédentes, ils ne possèdent pas une réponse impulsionnelle à phase linéaire.

Cas d'ondelettes bi-orthogonales :

La reconstruction parfaite est assurée grâce aux conditions suivantes :

1. Le filtre est de longueur paire.
2. La linéarité de la phase est assurée.
3. Les filtres hs et ha ne sont pas inverses l'un de l'autre dans le temps.
4. Les filtres *gs* [*n*] et *ga* [*n*] ne sont pas inverses l'un de l'autre dans le temps.
5. Les filtres *gs* [*n*] et *ha* [*n*] satisfont la condition : $gs\,[n] = (-1)^n ha[n]$.
6. Les filtres *ga* [*n*] et *hs* [*n*] sont des versions modulées l'une de l'autre avec un changement de signe : $ga\,[n] = (-1)^{n+1}\, hs[n]$.
7. Enfin, $\sum_n h_a\,[n] = \sum_n h_s\,[n] = \sqrt{2}$.

Un exemple de filtres de transformation en ondelettes biorthogonales sont le CDF9/7 (ou bior4.4) et le CDF5/3 utilisées dans le standard JPEG 2000, respectivement pour les modes de compression avec et sans pertes.

Cette approche peut être étendue aux signaux à deux dimensions tels que les images. Dans ce cas, chaque ligne de l'image est filtrée par un filtre passe-bas et un filtre passe-

haut. La sortie de chaque filtre est sous-échantillonnée par un facteur de deux pour produire les images intermédiaires L (image originale filtrée passe-bas et sous-échantillonnée dans la direction de x) et H (image originale filtrée passe-haut et sous-échantillonnée dans la direction de x).

Ensuite, chaque colonne des nouvelles images est filtrée avec les filtres passe-bas et passe-haut et sous-échantillonnée par un facteur de deux pour produire les quatre sous-images LL, LH, HL et HH (figure III.6 (a)) .Dans la phase de reconstruction ou synthèse, ces quatre sous-images sont combinées pour recréer une image avec le même nombre d'échantillons que l'image originale. (Figure III.6 (b)).

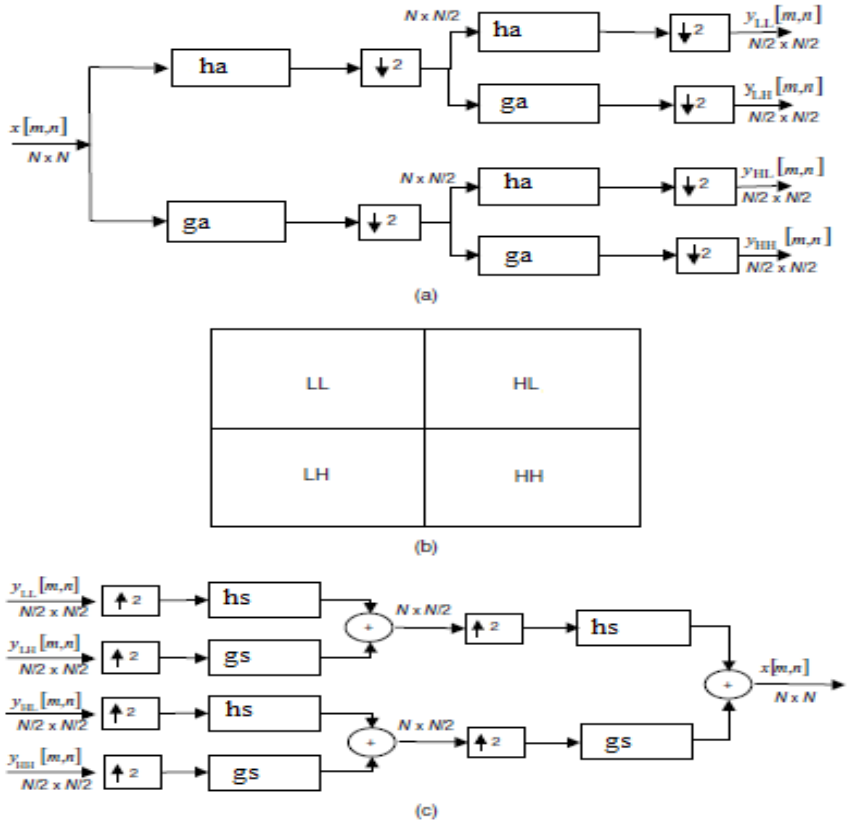
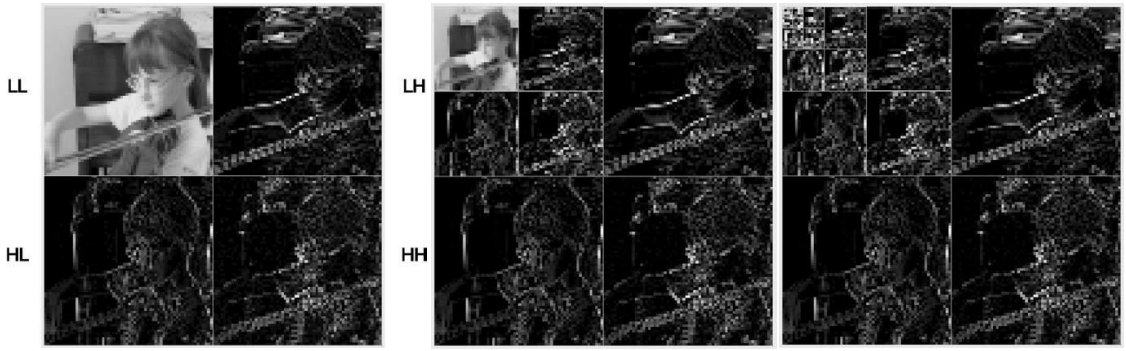


Figure III.6 Un niveau de DWT 2D. (a) 2D DWT décomposition, (b) emplacement des approximations et des détails, (c) 2D DWT reconstruction.

Dans une application de compression d'images, la décomposition en ondelettes à deux dimensions décrite ci-dessus est appliquée à nouveau à la sous-bande LL formant ainsi quatre nouvelles sous-images. L'image passe-bas résultante peut être filtrée itérativement pour créer un arbre d'images en sous-bandes. Les figures III.7 (a), (b) et (c) montrent le résultat de l'application d'une, de deux et de cinq étapes niveaux de décomposition respectivement [47]. Un grand nombre de coefficients dans l'image

haute fréquence est proche de zéro (presque noir) et ceci permet d'atteindre des taux de compression variables en fonction du nombre d'échantillons jugés insignifiants et enlevés avant la transmission de l'image.

Au niveau du décodeur, l'image originale est reconstruite par des opérations répétées de sur-échantillonnage, filtrages et addition.



(a) (b) (c)

Figure III.7 Différents niveaux de décomposition d'une image. (a) : Un niveau, (b) : 2 niveaux et (c) : 5 niveaux [47]

III.5.2.2 Par schéma d'élévation "lifting scheme" :

La recherche de nouvelles ondelettes et l'étude de leurs performances dans diverses applications ont fait l'objet de recherches intensives pendant les années 90. En 1995, Sweldens a proposé le lifting scheme [77]. C'est une technique de construction d'ondelettes ayant des moments nuls pour des ordres de plus en plus élevés. Le lifting scheme permet une implémentation très simple des décompositions en ondelettes et de leurs opérations inverses. Sweldens a publié plusieurs articles sur le sujet ([78] [79] [80]).

En 1996, Calderbank et al. [79] ont proposé une technique à base de lifting scheme pour rendre réversibles, c'est-à-dire d'entiers en entiers, les décompositions en ondelettes dyadiques associées à des filtres RIF. Cette technique est utilisée aujourd'hui dans la norme JPEG 2000 pour la compression des images.

Le lifting scheme est une alternative intéressante au schéma de filtrage convolutif classique de la transformée, car beaucoup moins complexe. En effet, le nombre d'opérations est divisé par un rapport allant jusqu'à deux en comparaison avec un schéma classique. De plus, une implémentation par lifting scheme est très économique en mémoire, qui ne dépend pas du nombre d'échantillons du signal à décomposer.

Enfin, Sweldens et Daubechies [78] ont montré que toute décomposition en ondelettes bi-orthogonales associée à des filtres RIF admet une représentation en lifting scheme.

Le lifting scheme consiste en trois étapes (figure III.8) :

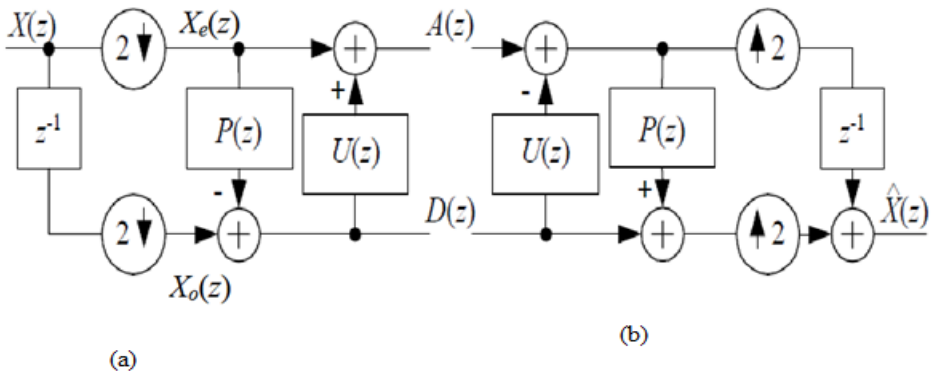


Figure III.8 Lifting scheme. (a) Décomposition, (b) Reconstruction.

En premier lieu, le signal d'entrée X est partitionné en plusieurs composantes par une décomposition polyphase. Généralement, cette dernière est dyadique et les échantillons d'indices pairs X_e et d'indices impairs X_o sont séparés. Puis, un opérateur de prédiction est appliqué aux échantillons pairs et le résultat est soustrait aux échantillons impairs. Cette opération élémentaire, appelée étape de prédiction (predict) ou pas primal, donne une erreur de prédiction :

$$D(Z) = X_o(Z) - P(X_e(Z)) \quad \text{(D pour Détails)} \tag{41}$$

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

$$A(Z) = X_e(Z) + U((D(Z)) \quad \text{(A pour Approximations)} \tag{42}$$

Dans le cas de filtres d'analyse à supports longs, plusieurs enchaînements de pas primaux et duaux Pi et Ui sont nécessaires à leur représentation en lifting scheme. Apparaissent alors des signaux intermédiaires Di et Ai, i = 1, ...,L. Au final, et après une éventuelle normalisation, un signal filtré passe-bas A_L et un signal filtré passe-haut D_L sont obtenus. Un grand intérêt pratique de cette représentation réside dans le fait que l'inversion de la transformée consiste simplement à changer les additions en soustractions et les soustractions en additions sans effectuer le calcul d'opérateurs inverses P_i^{-1}ou U_i^{-1} :

$$X_e(Z) = A(Z) - U(D(Z)) \tag{43}$$

$$X_o(Z) = D(Z) + P(X_e(Z)) \tag{44}$$

Les éventuelles erreurs d'arrondis commises à l'analyse dans les opérations U(D) et $P(X_e)$ seront commises de la même manière à la synthèse. Toutefois, les erreurs d'arrondis commises lors d'une somme ne sont pas systématiquement compensées lors de la soustraction et vice versa, c'est pourquoi la représentation en lifting scheme de la figure III.8 n'est pas systématiquement réversible [81].

Toutes les structures que nous venons de présenter se généralisent pour s'appliquer aux signaux bidimensionnels (2-D). La figure suivante montre l'efficacité de la transformée en ondelettes lorsqu'elle est implémentée en schéma lifting par rapport à l'implémentation par bancs de filtres et ceci pour différentes ondelettes.

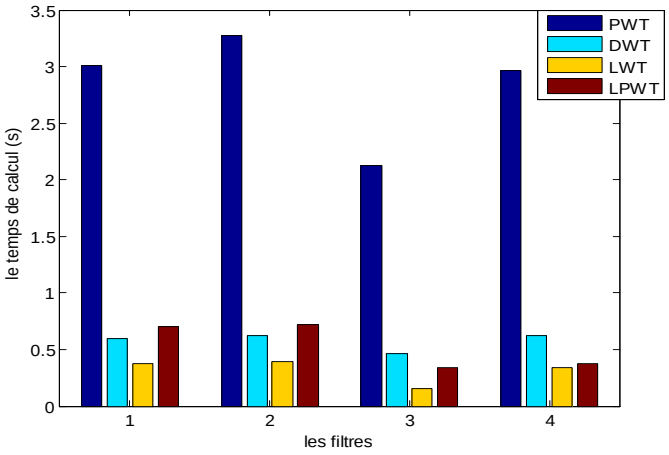


Figure III.9 Comparaison en termes de temps de calcul (image :barbara.png) (1:db5, 2:coif2, 3:sym2, 4:bior2.4) PWT : Packet Wavelet Transform DWT : Discrete Wavelet Transform LWT: Lifting Wavelet Transform LPWT: Lifting Packet Wavelet Transform.

Il existe cependant d'autres techniques de séparation des images telles que la transformée bidimensionnelle non séparable développée pour agir sur une grille d'échantillonnage quinconce, mais qui a du mal à trouver sa place au sein des autres méthodes. Sa complexité d'implémentation en fait un facteur discriminant pour des codeurs requérant un minimum d'opérations. Une autre difficulté est de définir des bancs de filtres non-séparables efficaces et permettant une bonne décorrélation [[82], [83], [84]].

D'autres chercheurs se tournent actuellement vers les transformées hybrides basées sur la DCT et la DWT et qui semblent prometteuses, surtout pour l'amélioration de la qualité de l'image mais, bien entendu, au prix d'une complexité croissante de l'implémentation [85].

III.6 Conclusion :

L'objectif fondamental d'une transformée est de décorréler les données et de compacter le maximum d'énergie dans un nombre minimum de coefficients. Les principales transformées utilisées dans le domaine de la compression d'images et de vidéos ont été

exposées dans ce chapitre. La DCT et la DWT ont été largement développées vue leur importance dans ce contexte. Les différentes implémentations ont aussi été présentées. La DCT approximative, recommandée par le standard H.264/AVC, a été largement exposée et discutée. Ce développement est nécessaire pour son implémentation hardware sur FPGA, objet du chapitre V de cette thèse.

Chapitre IV : Analyse des performances du H.264/AVC

IV.1 Introduction :

Depuis sa finalisation en Mai 2003,le standard H.264/AVC n'a cessé de susciter l'intérêt de la communauté scientifique. En effet, la norme ne propose pas seulement d'augmenter l'efficacité du codage et de la compression (au moins doubler le taux de compression de MPEG4), mais rajoute aussi plusieurs mécanismes visant à améliorer les performances de compression. Ceci, bien entendu, s'obtient au prix d'une complexité croissante.

Dans ce chapitre, l'impact de plusieurs paramètres est évalué avec pour objectif principal l'analyse de l'apport réel de ces paramètres sur la qualité de la vidéo (mesure du PSNR) et le surcoût induit sur le débit binaire et le temps de traitement. Ces deux derniers éléments sont d'une importance extrême en raison des limitations des bandes passantes dans les réseaux et du traitement en temps réel. L'étude est menée sur le codeur de référence H.264/AVC. La méthodologie suivie est détaillée dans la section suivante.

IV.2 Méthodologie :

Des séquences vidéo ont été choisies pour être des séquences de référence pour ce type d'étude. Notre sélection s'est portée sur quatre séquences : Foreman, Bus, City et Football. Le critère de choix est la mobilité (activité) des séquences.

Séquence Foreman :

Frame1 *frame 4* *frame 12*

Frame 90 *frame 100* *frame 140*

Figure IV.1 Quelques frames de la séquence vidéo "FOREMAN".

Séquence BUS :

Frame 1 *frame 10* *frame 20*

Frame 30 *frame 40* *frame 50*

Figure IV.2 Quelques frames de la séquence vidéo "BUS".

Séquence CITY :

Frame 1 *frame 20* *frame 50*

Frame 70 *frame 100* *frame 130*

Figure IV.3 Quelques frames de la séquence vidéo "CITY".

Séquence FOOTBALL :

Frame 1 *frame 15* *frame 25*

Frame 42 *frame 54* *frame 115*

Figure IV.4 Quelques frames de la séquence vidéo "FOOTBALL".

Pour chaque séquence, les paramètres suivants ont été testés :

- ✓ deux formats d'images QCIF (176 × 144) et CIF (352x 288),
- ✓ deux algorithmes de codage entropique (CABAC et CAVLC),
- ✓ différents nombres de frames à coder (3, 30, 60, 120),
- ✓ différents pas de quantification ([5, 45] par pas de 5),
- ✓ différentes plages de recherche "search range",
- ✓ différentes séquences caractérisées par différentes mobilités.
- ✓ les trois modes de transformée spatiale (entière 4x4, entière 8x8 et adaptative),
- ✓ le filtre de lissage (Deblocking filter).

Les séquences vidéo sélectionnées ont été codées en utilisant l'encodeur H.264 de référence, à savoir la plateforme JM 18.2. Dans chaque test, le PSNR est mesuré ainsi que le débit binaire et le temps de codage. Pour le PSNR, on peut quantifier séparément le PSNR de la composante de luminance (PSNR Y) et ceux des composantes de chrominance (PSNR U et PSNR V). Il est aussi possible de calculer la moyenne pondérée des PSNR des trois composantes ou de juger selon la composante de luminance uniquement. Il faut noter que les résultats obtenus, en particulier les temps de codage, dépendent des caractéristiques de l'ordinateur utilisé. Les résultats des simulations exposées dans ce chapitre ont été acquis en travaillant avec un micro-ordinateur dont les paramètres du système sont les suivants :

- Processeur : Intel ® Core ™2 Duo CPU T6670 @ 2.20 GHz.
- Mémoire RAM : 2 Go.
- Type du système : système d'exploitation : windows 7 32 bits.

IV.3 Présentation de la plateforme JM 18.2 :

IV.3.1 Installation :

JM 18.2 est un logiciel de référence permettant d'analyser les performances du standard H.264/AVC en faisant varier plusieurs paramètres. Le logiciel renferme un espace de travail (workspace) " Visual Studio.NET" dont le nom dépend de la version .NET que l'utilisateur possède :

- "jm_vc7.sln" pour .NET 2003 (v7),
- "jm_vc8.sln" pour .NET 2005 (v8),
- "jm_vc9.sln" pour .NET 2008 (v9).

Chaque "workspace" renferme les projets suivants:

lencod H.264/AVC : codeur de référence.

ldecod H.264/AVC : décodeur de référence.

Rtpdump : un outil pour analyser les contenus des paquets RTP (Rapid Transport Protocol).

rtp_loss : un outil pour simuler la perte des paquets RTP.

Après sélection et compilation du projet désiré, les fichiers exécutables "lencod.exe" ou "ldecod.exe" seront créés dans le répertoire "bin". Concernant les fichiers "rtpdump.exe" et "rtp_loss.exe", ils seront créés dans les répertoires "rtpdump" et "rtp_loss" respectivement. Ceci est valable dans le cas d'une installation sous windows. Pour l'installer sous unix, il faut se reporter au manuel "H.264/14496-10 AVC REFERENCE SOFTWARE MANUAL", plus connu sous l'appellation "JVT-A10" [86]

IV.3.2 Paramètres :

Le codeur renferme plus de 300 paramètres classés en plusieurs catégories:

1. "File input/output related parameters": cette catégorie inclut les paramètres relatifs au fichier vidéo à traiter tels que le nom du fichier, le format, la fréquence, … etc.
2. "Primary control parameters": il s'agit des paramètres de contrôle primaires du codage tels que le profile, le niveau, le pas de quantification des tranches

I et P, le "search range", le nombre de frames de référence utilisés pour l'estimation de mouvement en mode inter, … etc.

3. "Error resiliency and slice control": dans cette classe, on trouve les modes de prédiction (luminance, chrominance), le pas de quantification des tranches codées B, leur nombre, … etc.
4. "Sp coding support": englobe les paramètres de contrôle des tranches SP et SI tels que le pas de quantification et la périodicité.
5. "Output control/entropy coding, nals" : cette classe réunit le mode de codage entropique (CABAC ou CAVLC), le mode de partition et le mode du fichier de sortie.
6. "Interlace format handling" : permet la sélection du mode de codage PAF ou MBAFF.
7. Frext profile parameters: cette catégorie regroupe les paramètres tels que le format YUV, la résolution des composantes de luminance et de chrominance, les modes de transformées, … etc.
8. Deblocking filter parameters : où sont réunis les paramètres d'activation/désactivation du filtrage et les paramètres de contrôle du filtrage (α_offset, β_offset) relatifs à chaque type de slice (I, P ou B).

Les tests ont été menés avec le profile high (code 100) .

IV.4 Simulations, résultats et discussions :

IV.4.1Variation du PSNR et du débit en fonction du pas de quantification :

La séquence vidéo testée est "Foreman", échantillonnée en YUV : 4 :2 :0 et utilisant le codage entropique "CABAC". trois frames, en format QCIF (176x144) sont codés.

Les valeurs des PSNR de Y , U et V sont les moyennes des valeurs des PSNR des images codées I, P et B. Le débit donné par le codeur de la plateforme JM 18.2 est calculé selon l'expression suivante :

$$Débit = \left(\frac{N_{bt}}{N_{fc}}.f_v\right) / 1000 \qquad (45)$$

Avec : N_{bt} = nombre total de bits de codage des frames,

N_{fc} = nombre de frames à coder,

f_v : fréquence de la séquence vidéo à coder (25 ou 30),

1000 : pour la conversion en Kbits et qui est une valeur approximée, la valeur

exacte étant 1024.

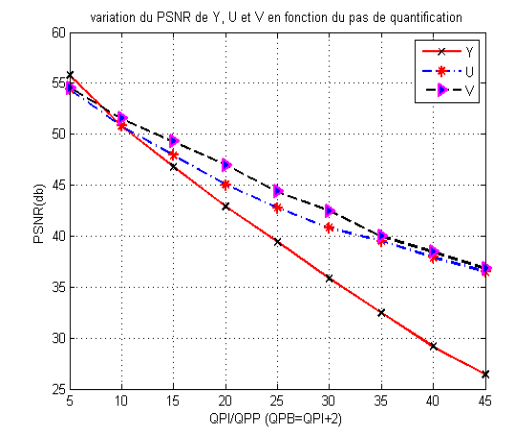


Figure IV.5(a)PSNR de Y, U et V en fonction de QP.

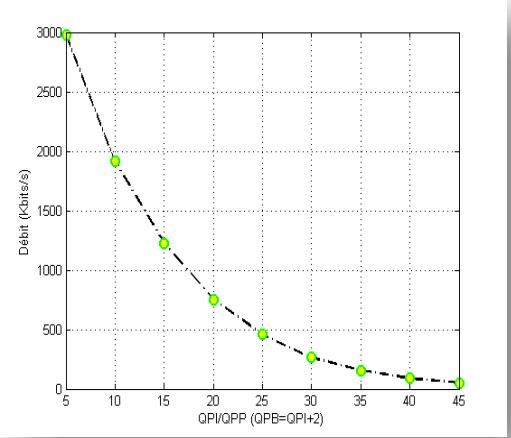


Figure IV.5 (b) Débit en fonction de QP.

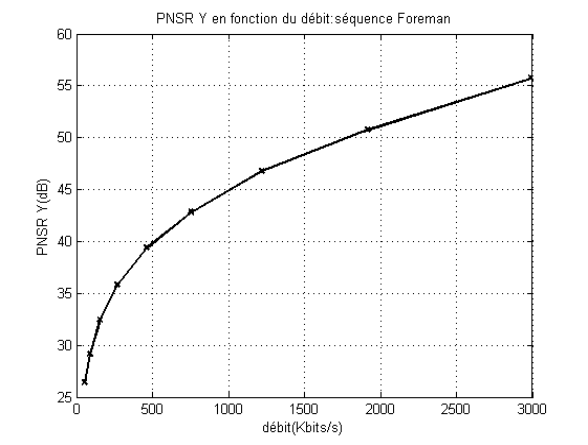


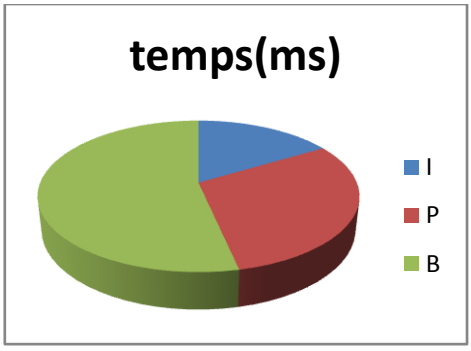
Figure IV.5(c) PSNR de Y en fonction du débit

Comme prévu, les courbes précédentes montrent que le PSNR et le débit décroissent en fonction du pas de quantification. Mais, si le PSNR présente une décroissance régulière, le débit dévoile une forte décroissance dans une première région (QP = [0, 15]), moyenne dans la zone (QP=[15, 25]), puis faible dans la dernière plage.

Le résultat suivant, mené avec QPI = QPP =15 et QPB = 17, montre que les images I sont les plus gourmandes (nombre de bits le plus élevé). Cependant, les images I sont les plus rapides à compresser (voir Tableau IV.1). Viennent ensuite les images codées P et enfin les images codées B, les moins gourmandes mais dont le temps de traitement est le plus long (voir figures IV.6(a) et IV.6(b)).

Type de codage de l'image	Temps (ms)	Nombre de bits
I	653	69960
P	1199	37272
B	2122	14856

Tableau IV.1 Temps de codage et nombre de bits des frames codés I, P et B.



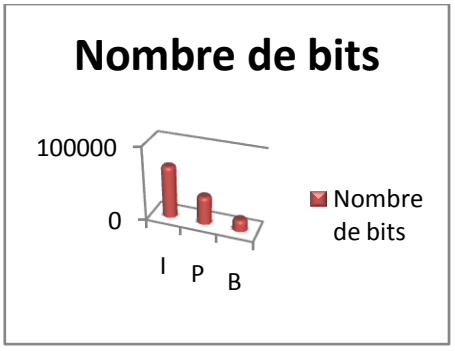


Figure IV.6(a) Temps de codage des frames I, P et B. Figure IV.6(b) Nombre de bits des frames I, P et B.

Il est d'usage d'utiliser le même pas de quantification pour les frames codés I et P (QPI = QPP) et d'accroitre ce pas de deux pour les frames codés B (QPB = QPI + 2) pour améliorer la compression. Dans ce deuxième test, les performances d'une compression utilisant ce processus et une compression utilisant un pas de quantification identique pour les trois types de frames seront comparées.

IV.4.2 Comparaison des performances de compression utilisant un pas de quantification unique et un pas de quantification différent :

Dans un premier temps, un test est mené avec les mêmes paramètres que précédemment en utilisant une fois, un pas de quantification des frames B différent de celui de I et P, puis en prenant un pas identique pour les trois types. Dans tous les tableaux suivants, les abréviations TCG et TEM sont utilisées pour temps de codage global et temps d'estimation de mouvement respectivement.

QPI=QPP (QPB=QPI+2)	5	10	15	20	25	30	35	40	45
PSNRY (dB)	55.752	50.753	46.758	42.840	39.402	35.832	32.433	29.136	26.440
Débit (kbits/s)	2986.88	1915.76	1219.28	753.28	461.44	265.60	154.40	88.00	52.00
TCG (s)	5.029	4.423	3.964	3.601	3.285	3.039	2.838	2.657	2.476

Tableau IV.2 Résultats des performances pour des paramètres de quantification différents pour les 3 types (I, P et B)(cas de 3 frames)

QPI=QPP =QPB	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**
PSNRY (dB)	56.359	51.239	47.158	43.097	39.688	36.026	32.619	29.338	26.522
Débit (kbits/s)	3134.64	2027.04	1288.48	779.20	479.52	273.92	156.72	90.00	52.72
TCG (s)	5.089	4.579	4.081	3.686	3.368	3.096	2.931	2.703	2.561

Tableau IV.3 Résultats des performances pour des paramètres de quantification égaux pour les 3 types (I, P et B)(cas de 3 frames)

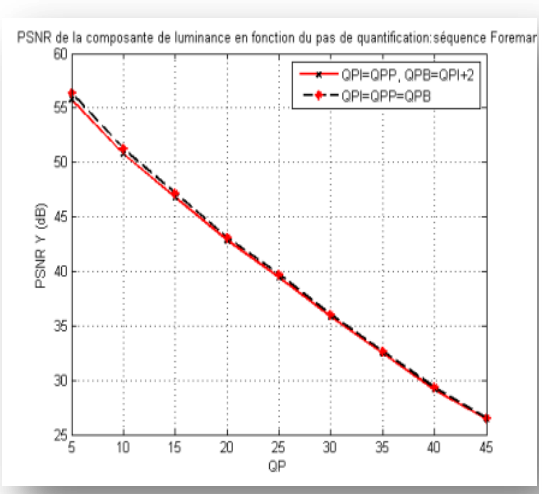


Figure IV.7(a) Comparaison PSNR(QPI=QPP=QPB) et PSNR Y QPI=QPP=QPB-2)(cas de 3 frames)

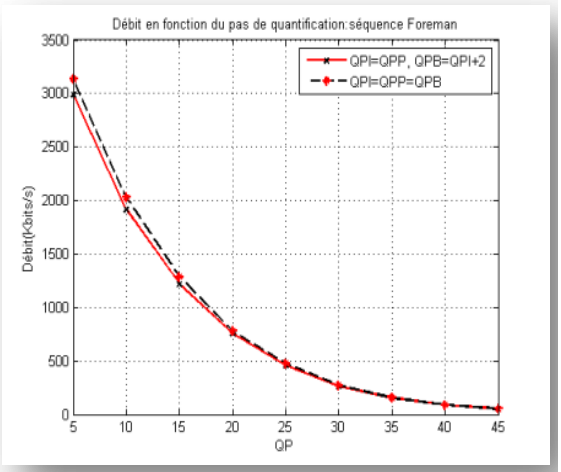


Figure IV.7(b) Comparaison débit (QPI=QPP=QPB) et débit ((QPI=QPP=QPB-2)(cas de 3 frames)

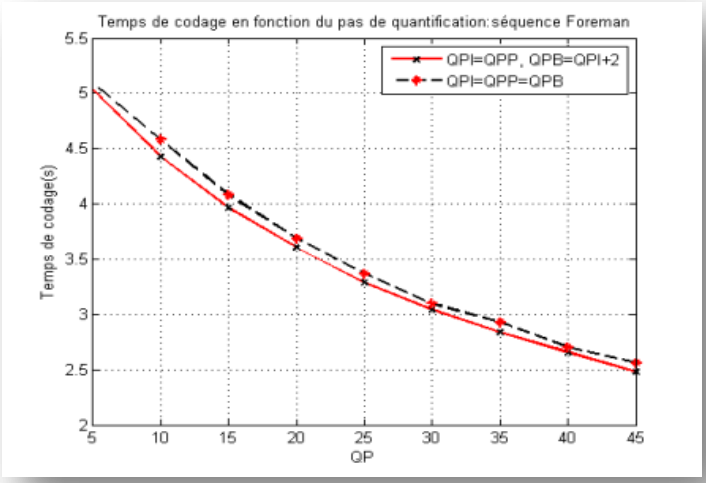


Figure IV.7(c) Temps de codage (QPI=QPP=QPB) et Temps de codage (QPI=QPP=QPB-2)(cas de 3 frames)

Les résultats précédents obtenus pour le codage de 3 frames montrent qu'il n'y a pratiquement aucune différence entre les deux cas. Ceci est prévisible puisqu'il n'y a qu'un seul frame B à coder. Nous verrons dans ce qui suit que l'écart s'élargit au fur et à mesure que le nombre de frames B augmente.

Les résultats suivants sont obtenus avec le codage de 30 frames et 7 frames de type B :

QPI=QPP (QPB=QPI+2)	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**
PSNRY (dB)	53.427	48.936	45.133	41.488	38.027	34.440	31.020	27.576	24.630
Débit (kbits/s)	2009.59	1155.76	645.84	354.98	196.38	107.58	59.11	30.05	16.63
TCG (s)	85.207	79.030	75.447	73.249	71.444	70.774	68.752	65.560	60.056

Tableau IV.4 Résultats des performances pour des paramètres de quantification différents pour les 3 types (I, P et B)(cas de 30 frames)

QPI=QPP =QPB	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**
PSNRY (dB)	55.309	50.259	46.223	42.357	38.852	35.248	31.663	28.160	25.185
Débit (kbits/s)	2369.14	1396.79	783.30	429.77	234.31	127.06	69.08	35.34	18.89
TCG (s)	87.943	81.460	77.163	74.887	72.585	71.793	70.474	68.160	62.873

Tableau IV.5 Résultats des performances pour des paramètres de quantification égaux pour les 3 types (I, P et B)(cas de 30 frames)

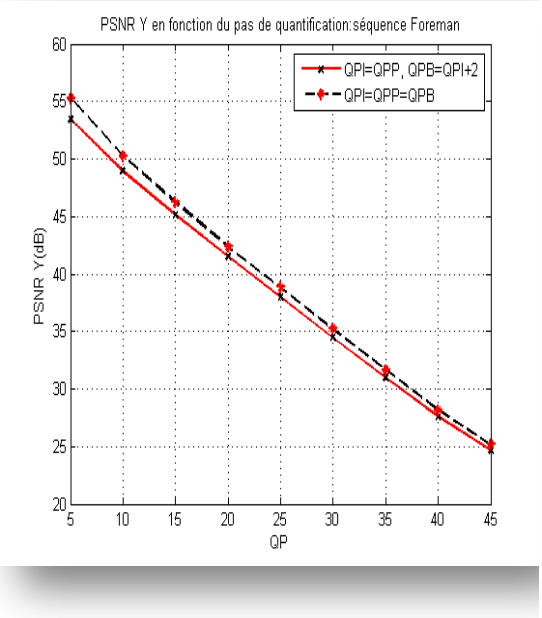


Figure IV.8(a) PSNR Y pour QP=QPP≠QPB et QPI=QPP=QPB (cas de 30 frames)

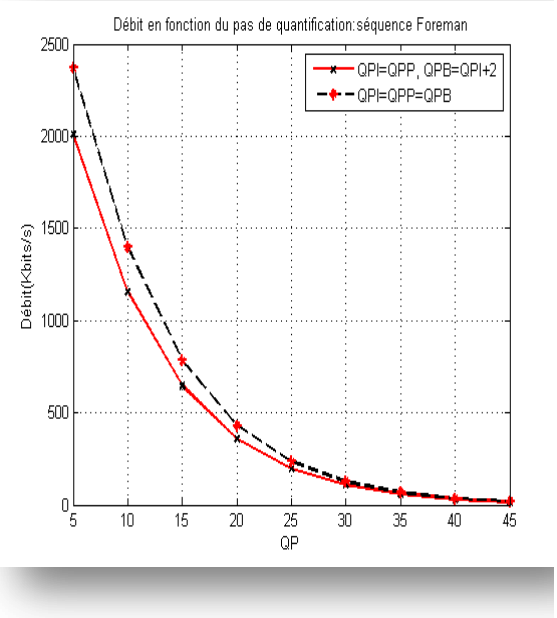


Figure IV.8(b) Débit pour QPI=QPP≠QPB et QPI=QPP=QPB (cas de 30 frames)

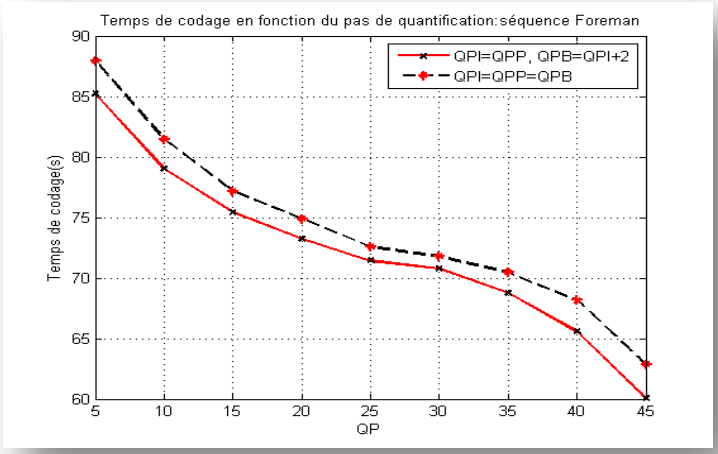


Figure IV.8(c) Temps de codage pour QPI=QPP=QPB et QPI=QPP≠QPB (30 frames).

Les courbes IV.8 (a), IV.8 (b),et IV.8 (c) montrent un gain en qualité variant de 0.5550 dB pour un pas de quantification de 45 à 1.8820 dB pour un pas de quantification de 5, soit une moyenne de 0.9532 dB dans le cas d'un pas identique. Ce gain s'obtient au prix d'une augmentation du débit allant de 359.55 Kbits/s à 2.26 Kbits/s correspondant à QP variant de 5 à 45 ; et un alourdissement du temps de codage avoisinant les 3 secondes.

IV.4.3 Comparaison des codages entropiques (CABAC vs CAVLC) :

Cette simulation est conduite avec 3 frames de la séquence Foreman et les paramètres du premier test avec un pas de quantification QPI=QPP=15, QPB = 17 et le codage entropique CAVLC.

Type de Codage	Temps de codage (I) (ms)	Temps de codage (P) (ms)	Temps de codage (B)(ms)	Temps de codage moyen (s)	Bits(I)	Bits(P)	Bits(B)	Nombre de bits moyen
CAVLC	518	1035	1946	1.116	71072	38616	15512	41733
CABAC	653	1199	2122	1.325	69960	37272	14856	40696

Tableau IV.6 Comparaison des performances des codages entropiques CABAC et CAVLC.

Cette première évaluation (tableau IV.2) indique que le codage CABAC permet une amélioration du débit par rapport au codage CAVLC contre une augmentation du temps de codage. Le prochain test comparera les performances des deux types de codage en termes de débit, qualité (PSNR) et temps de codage pour une séquence composée de 60 frames de la même séquence vidéo Foreman.

Codage CAVLC (60 frames) :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.303	48.922	45.164	41.520	38.008	34.423	30.937	27.552	24.584
Débit (kbits/s)	2026.77	1157.66	636.24	345.32	187.33	101.74	55.28	29.18	15.67
TCG (s)	160.870	152.681	148.308	147.821	145.845	144.348	140.269	132.654	120.645
TEM (s)	91.804	93.517	98.235	105.777	107.817	110.027	108.722	103.150	92.756

Tableau IV.7 Résultats des performances du codage CAVLC (pour 60 frames)

Codage CABAC (60 frames) :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.356	48.933	45.176	41.533	38.030	34.499	31.018	27.661	24.563
Débit (kbits/s)	1937.22	1100.67	601.39	322.03	173.35	93.03	50.17	26.06	13.94
TCG (s)	177.750	164.993	157.276	153.231	150.113	147.517	142.887	134.848	123.279
TEM (s)	92.225	94.011	98.730	104.280	107.494	109.311	107.704	102.029	92.251

Tableau IV.8 Résultats des performances du codage CABAC (pour 60 frames)

On note une réduction moyenne du débit de 5.83% (figure IV.9(b)) lorsque le codage CABAC est utilisé par rapport au codage CAVLC pour une même qualité d'images (figure IV.9(a)). Les écarts entre les débits produits en utilisant le codage CAVLC puis CABAC sont indiqués dans le tableau IV.9.

QP	5	10	15	20	25	30	35	40	45
Différence (Kbits/s)	89.55	56.99	34.85	23.29	13.98	8.71	5.11	3.12	1.73

Tableau IV.9 Réduction du débit avec le codage CABAC par rapport au codage CAVLC.

La figure IV.9(c) donne l'avantage au codage CAVLC pour ce qui est du temps de codage. Dans ce cas particulier, l'augmentation du temps de codage est de 4.70 % en moyenne avec le codage CABAC. Cet écart tend à rétrécir au fur et à mesure que le débit s'affaiblit et la figure IV.9(d) permet de trancher en faveur du codage CABAC . L'ensemble des résultats consolide l'idée de l'éternel compromis à faire entre les trois variables : qualité, débit et temps de codage.

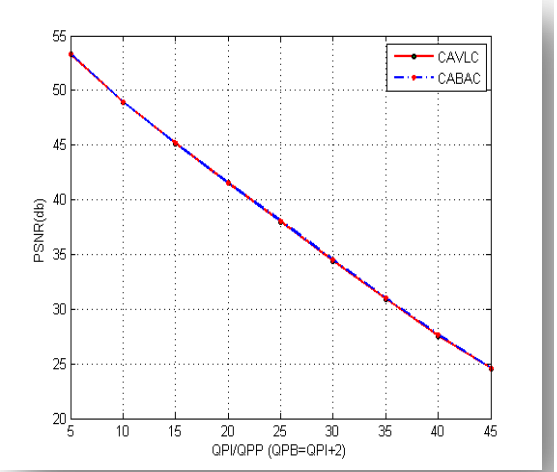


Figure IV.9(a) PSNR Y (CAVLC vs CABAC)

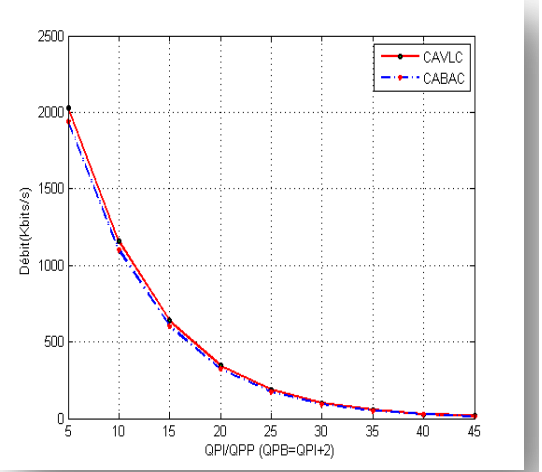


Figure IV.9(b) Débit (CAVLC vs CABAC)

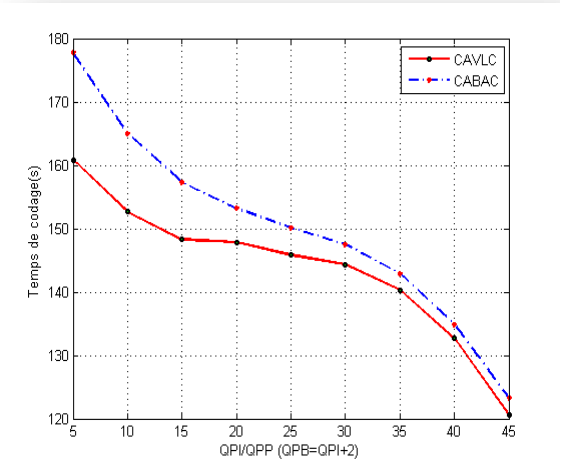


Figure IV.9(c) Temps de codage (CABAC vs CAVLC)

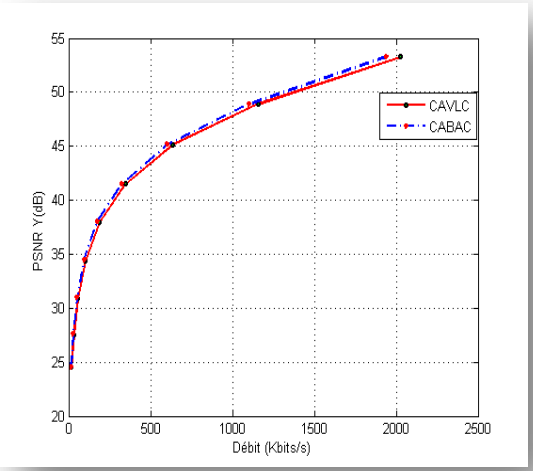


Figure IV.9(d) PSNR en fonction du débit (CABAC vs CAVLC

IV.4.4 Etude de l'effet du paramètre "search range" :

Le paramètre "search range" représente la taille de la fenêtre dans laquelle la recherche de correspondances entre les blocs doit se faire. Plus cette fenêtre est large, plus la probabilité de retrouver les blocs à forte ressemblance est forte. Cette possibilité a toute son importance dans les scènes vidéo rapides. Dans les simulations suivantes, nous allons tester différentes valeurs de ce paramètre : 16, 20, 24, 28 et 32. La séquence à coder est composée de 9 frames dont 7 sont de type B, le codage entropique étant CAVLC. Afin de comparer les performances, deux séquences vidéo sont utilisées :

Foreman et Football. Le choix de ces deux dernières est basé sur leurs activités bien distinctes. La première étant assez lente alors que la deuxième est rapide.

1. Séquence vidéo "Foreman":

a. search range=16:

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.466	46.821	42.469	38.230	34.090	29.970	26.107
Débit (kbits/s)	1053.88	593.32	325.33	173.24	88.77	42.88	20.21
TCG (s)	12.672	11.713	11.360	11.110	10.856	10.534	9.426
TEM (s)	7.361	7.322	7.603	7.840	7.943	7.846	6.964

Tableau IV.10 Résultats des performances pour search range=16(séquence Foreman)

b. Search range=20:

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.497	46.806	42.472	38.216	34.127	29.949	26.118
Débit (kbits/s)	1056.13	595.95	324.61	173.64	89.44	42.92	20.37
TCG (s)	12.333	11.778	11.379	11.093	10.879	10.407	9.391
TEM (s)	7.192	7.360	7.610	7.818	7.960	7.750	6.933

Tableau IV.11 Résultats des performances pour search range=20(séquence Foreman)

c. Search range=24:

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.474	46.832	42.483	38.235	34.046	29.920	26.046
Débit (kbits/s)	1056.83	593.24	324.07	172.88	87.75	42.75	19.95
TCG (s)	12.506	11.940	11.576	11.344	11.081	10.565	9.524
TEM (s)	7.367	7.528	7.834	8.072	8.169	7.912	7.079

Tableau IV.12 Résultats des performances pour search range=24(séquence Foreman)

d. Search range=28

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.475	46.822	42.460	38.234	34.049	29.945	25.979
Débit (kbits/s)	1053.59	593.00	323.87	172.43	88.20	42.52	19.84
TCG (s)	12.436	11.994	11.605	11.324	11.117	10.582	9.622
TEM (s)	7.309	7.561	7.847	8.061	8.193	7.929	7.164

Tableau IV.13 Résultats des performances pour search range=28(séquence Foreman)

e. Search range=32:

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.463	46.806	42.491	38.248	34.082	29.950	25.991
Débit (kbits/s)	1053.61	593.89	325.51	174.20	88.56	42.59	19.97
TCG (s)	12.556	11.947	11.593	11.342	11.099	10.579	9.567
TEM (s)	7.402	7.548	7.852	8.068	8.181	7.925	7.113

Tableau IV.14 Résultats des performances pour search range=32(séquence Foreman)

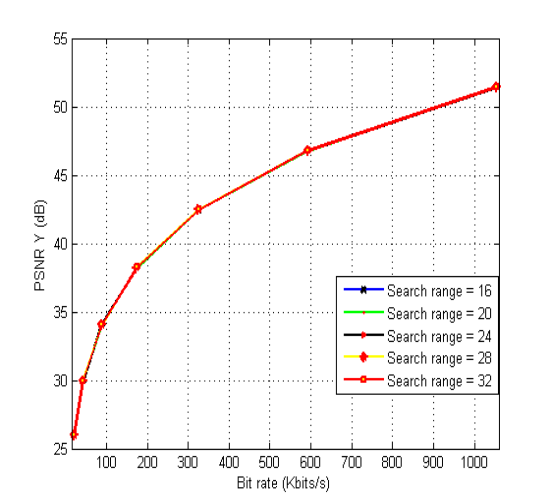


Figure IV.10(a) Courbes R-D pour différents search range (séquence Foreman)

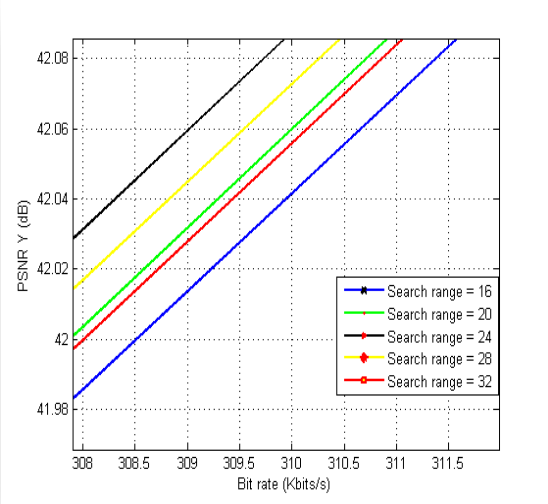


Figure IV.10(b) Figure IV.10.a zoomée

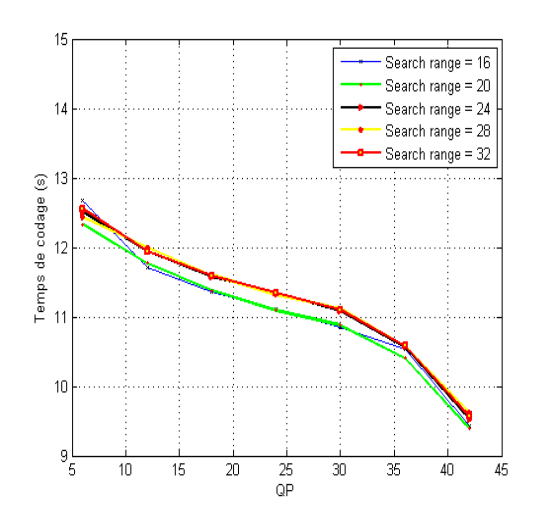


Figure IV.10(c) Temps de codage pour différents seach range (séquence Foreman)

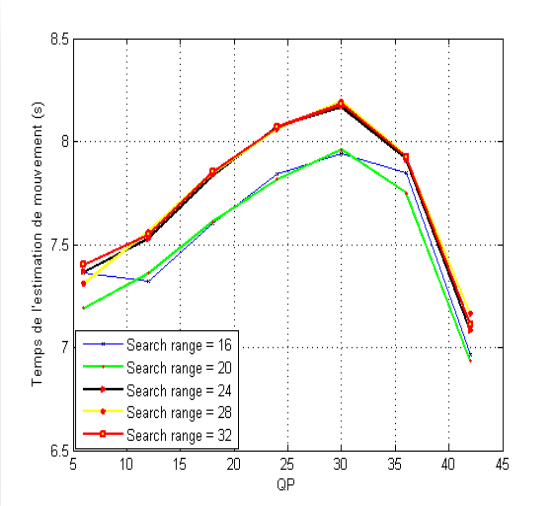


Figure IV.10(d) Temps d'estimation du mouvement pour différents seach range (séquence Foreman)

2. Séquence vidéo Football:

a. Search range=16

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.679	46.263	40.705	35.400	30.592	26.349	22.411
Débit (kbits/s)	2034.56	1398.60	920.25	549.23	294.23	136.16	52.88
TCG (s)	16.470	16.010	15.515	14.895	14.240	13.072	11.521
TEM (s)	10.399	10.506	10.710	10.793	10.732	10.062	8.836

Tableau IV.15 Résultats des performances pour search range=16(séquence Football)

b. Search range=20

QP	6	12	18	24	30	36	42
PSNR Y (dB)	51.680	46.268	40.729	35.376	30.573	26.273	22.438
Débit (kbits/s)	2037.49	1400.48	920.45	547.61	294.07	135.13	53.05
TCG (s)	16.669	16.077	15.555	14.946	14.249	13.148	11.505
TEM (s)	10.528	10.556	10.732	10.839	10.747	10.146	8.824

Tableau IV.16 Résultats des performances pour search range=20 (séquence Football)

c. Search range=24

QP	6	12	18	24	30	36	42
PSNR Y	51.670	46.273	40.711	35.406	30.561	26.343	22.529
Débit (kbits/s)	2036.99	1402.92	919.07	548.72	292.64	136.23	53.56
TCG (s)	16.932	16.464	15.981	15.470	14.663	13.559	11.866
TEM (s)	10.856	10.967	11.146	11.350	11.162	10.530	9.163

Tableau IV.17 Résultats des performances pour search range=24(séquence Football)

d. Search range=28

QP	6	12	18	24	30	36	42
PSNR Y	51.690	46.261	40.701	35.381	30.566	26.287	22.365
Débit (kbits/s)	2039.25	1401.32	919.20	548.11	294.15	135.35	52.76
TCG (s)	17.062	16.551	16.141	15.461	14.738	13.606	11.941
TEM (s)	10.943	11.038	11.268	11.334	11.210	10.583	9.253

Tableau IV.18 Résultats des performances pour search range=28(séquence Football)

e. Search range=32

QP	6	12	18	24	30	36	42
PSNR Y	51.682	46.283	40.701	35.395	30.602	26.350	22.545
Débit (kbits/s)	2035.57	1402.84	919.25	549.40	293.95	136.41	54.04
TCG (s)	17.076	16.573	16.056	15.445	14.781	13.589	11.927
TEM (s)	10.954	11.044	11.203	11.322	11.246	10.551	9.195

Tableau IV.19 Résultats des performances pour search range=32 (séquence Football)

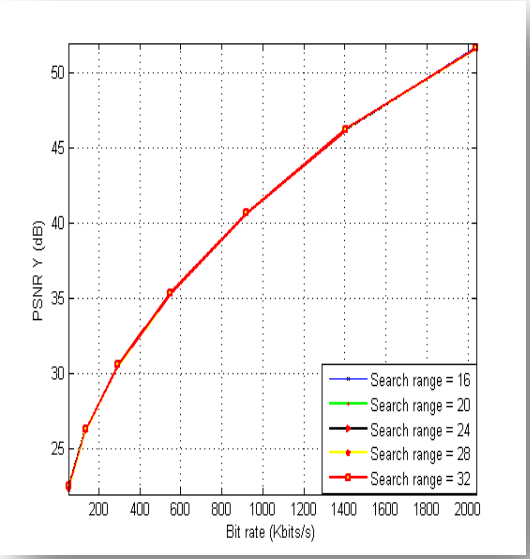


Figure IV.11(a) Courbes R-D pour différents search range (séquence Football)

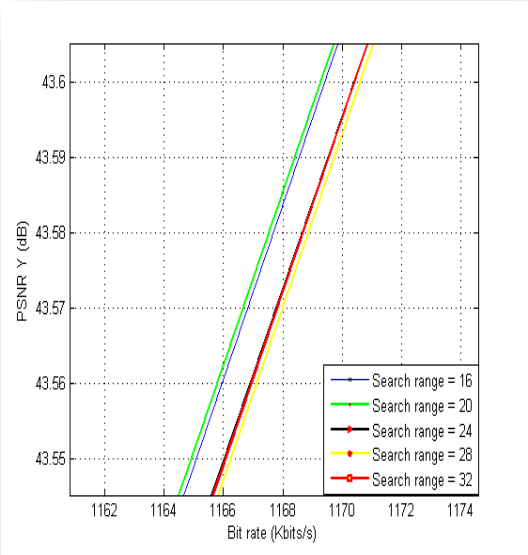


Figure IV.11(b) Figure IV.11.a zoomée

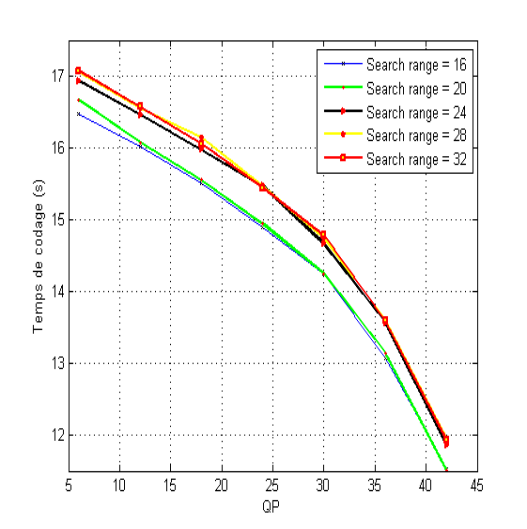


Figure IV.11(c) Temps de codage pour différents search range (séquence Football)

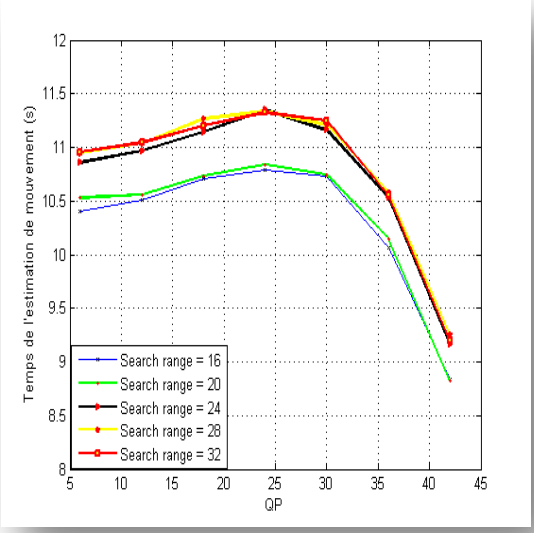


Figure IV.11(d) Temps d'estimation du mouvement pour différents search range (séquence Football)

Les résultats montrent que la différence en termes de qualité est insignifiante. En effet, elle n'est que de l'ordre du 100^{ième} du dB (figures IV.10 (b) et IV.11(b)). Il faut tout de même mentionner qu'une grande fenêtre de recherche prend toute son importance dans le cas de séquences vidéo à très grande activité. La figure IV.11 (b) montre qu'un search range de valeur 32 a une meilleure performance que celle exposée dans la figure IV.10 (b). Quand à la variation du temps de codage, elle ne dépasse pas 0.5 s. Le temps de codage est d'autant plus important que la fenêtre de recherche est grande. Ceci est la conséquence, tout à fait prévisible du temps de l'estimation de mouvement qui augmente avec la taille de la fenêtre de recherche. Les figures IV.10 (c) et IV.11 (c) indiquent que le temps de codage global décroit au fur et à mesure que QP croit. Ce résultat est tout à fait logique puisque l'augmentation de QP résulte en l'élimination de plus de détails et par conséquent il y'a moins d'information à coder. Par contre le temps imparti à l'estimation du mouvement augmente légèrement jusqu'au point où il n'y a plus ou il y'a très peu de détails, puis diminue rapidement (voir figures IV.10 (d) et IV.11 (d)).

IV.4.5 Effet de la variation du nombre de frames à coder :

La séquence vidéo utilisée est Foreman (176x144), le codage est CABAC, le paramètre sous test est le nombre de frames à coder, et 7 frames sont du type B. La séquence est alors :

30 frames codés : I P B B B B B B B P B B B B B B B P B B B B B B B P B B B B

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.427	48.936	45.133	41.488	38.027	34.440	31.020	27.576	24.630
Débit (kbits/s)	2009.59	1155.76	645.84	354.98	196.38	107.58	59.11	30.05	16.63
TCG (s)	85.633	79.984	76.148	74.164	72.025	71.253	69.456	66.015	60.737
TEM (s)	43.135	44.315	46.701	49.436	50.673	52.097	51.798	49.569	45.077

Tableau IV.20 Résultats des performances du codeur en fonction du nombre de frames à coder (cas de 30 frames).

60 frames:

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.356	48.933	45.176	41.533	38.030	34.499	31.018	27.661	24.563
Débit (kbits/s)	1937.22	1100.67	601.39	322.03	173.35	93.03	50.17	26.06	13.94
TCG (s)	177.750	164.993	157.276	153.231	150.113	147.517	142.887	134.848	123.279
TEM (s)	92.225	94.011	98.730	104.280	107.494	109.311	107.704	102.029	92.251

Tableau IV.21 Résultats des performances du codeur en fonction du nombre de frames à coder (cas de 60 frames).

120 frames :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.571	49.114	45.309	41.575	37.970	34.282	30.731	27.415	24.176
Débit (kbits/s)	2087.05	1224.96	694.15	380.54	207.88	113.23	60.40	31.61	16.38
TCG (s)	370.395	345.700	328.713	318.524	311.614	308.841	300.354	285.529	261.588
TEM (s)	197.074	200.442	207.768	217.266	223.855	230.084	228.457	218.551	198.072

Tableau IV.22 Résultats des performances du codeur en fonction du nombre de frames à coder (cas de 120 frames).

Les résultats de cette simulation montrent que le temps de codage est proportionnel au nombre de frames à coder et qu'il suit pratiquement le même rapport (figure IV.12(c)). En effet, lorsque le nombre d'images à coder a doublé, le temps l'a aussi et c'est la même constatation lorsque ce nombre a quadruplé. La qualité n'est pas liée à ce paramètre. Le cas de 3 frames n'est pas à prendre en considération vu que les frames de type B ne sont pas réellement exploités et représentent 33.33% du nombre total de frames à coder. Dans les trois autres cas, c'est la proportion de frames codés B qui influe sur le débit. Plus elle est élevée, plus faible est le débit.

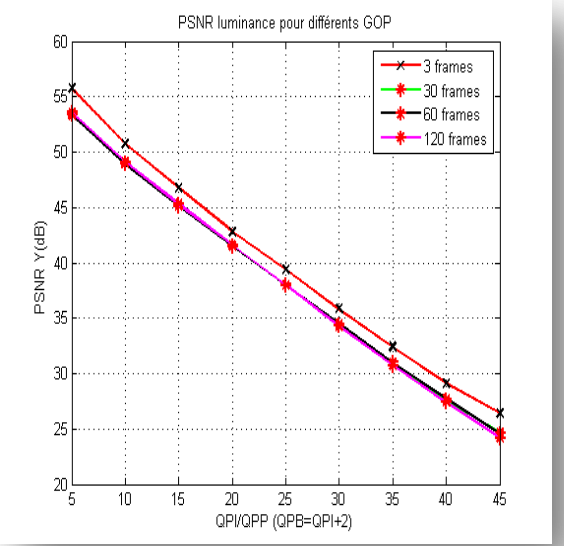


Figure IV.12(a) PSNR Y en fonction du pas de quantification pour différents GOP.

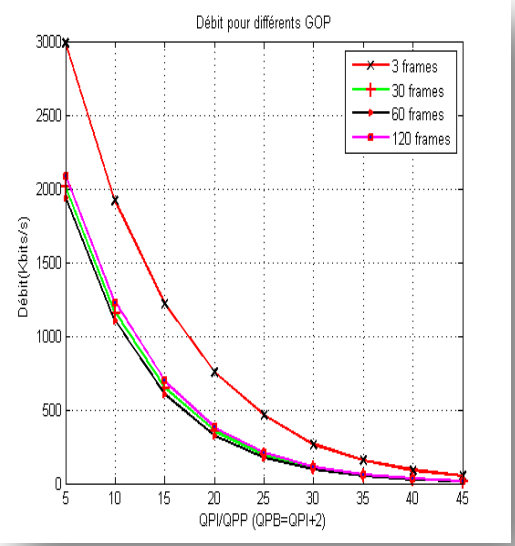


Figure IV.12(b) Débit en fonction du pas de quantification pour différents GOP.

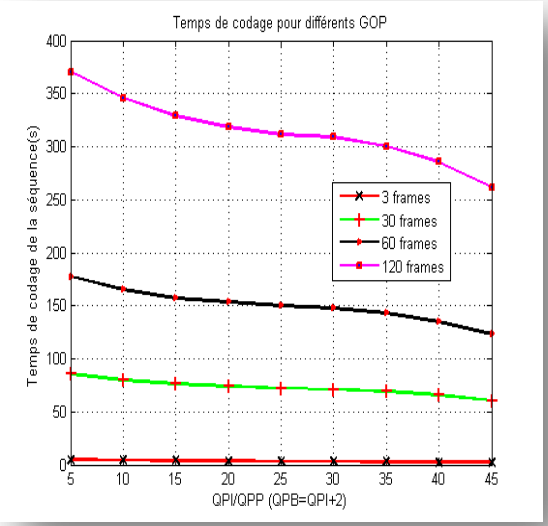


Figure IV.12(c) Temps de codage en fonction du pas de quantification pour différents GOP.

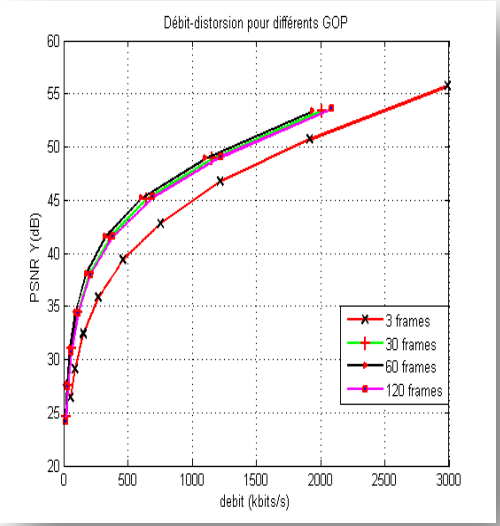


Figure IV.12(d) Courbes R-D pour différentes GOP.

IV.4.6 Influence de la taille du frame sur les performances de codage:

Simulation Foreman (CABAC,QCIF :176x144) vs Foreman (CABAC, CIF:352x288) :

Foreman (CABAC,QCIF):

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	55.752	50.753	46.758	42.840	39.402	35.832	32.433	29.136	26.440
Débit (kbits/s)	2986.88	1915.76	1219.28	753.28	461.44	265.60	154.40	88.00	52.00
TCG (s)	5.001	4.446	4.003	3.617	3.268	3.038	2.848	2.657	2.487
TEM (s)	1.466	1.474	1.505	1.510	1.466	1.455	1.402	1.322	1.235

Tableau IV.23 Résultats des performances du codeur en fonction de la taille du frame (format QCIF)

Foreman (CABAC, CIF) :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	56.820	51.104	46.716	42.842	39.688	36.470	33.510	30.710	28.014
Débit (kbits/s)	11957.60	7735.12	4574.56	2507.92	1378.16	722.80	392.56	221.36	124.72
TCG (s)	20.204	17.923	15.775	14.017	12.622	11.607	10.902	10.163	9.473
TEM (s)	6.232	6.201	6.122	6.061	5.828	5.629	5.425	5.032	4.590

Tableau IV.24 Résultats des performances du codeur en fonction de la taille du frame (format CIF)

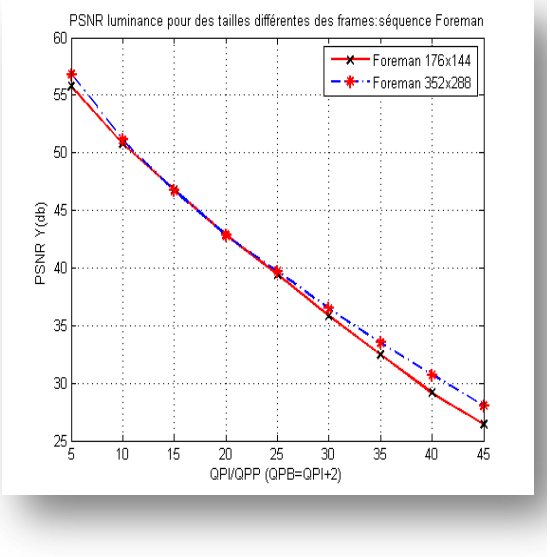


Figure IV.13(a) Influence du format sur le PSNR

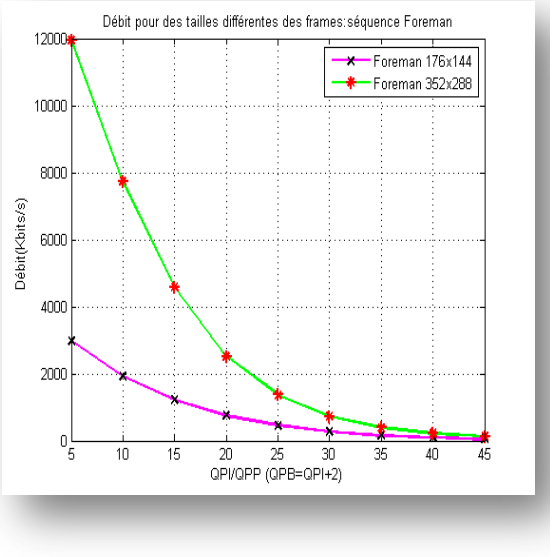


Figure IV.13(b) Influence du format sur le débit.

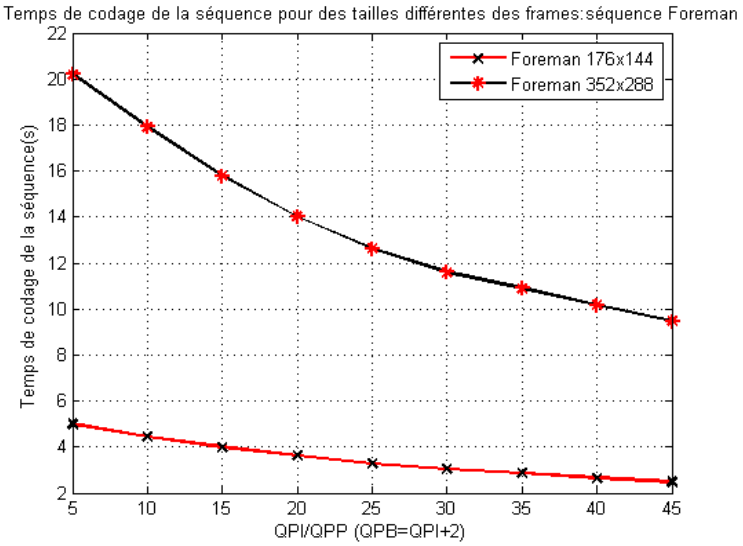


Figure IV.13(c) Influence du format sur le temps de codage.

Ce test permet de conclure que même si la qualité n'est pas altérée par le format de l'image à coder, il existe une énorme différence dans le débit et le temps de codage (figures IV.13(b) et IV.13(c)) en dépit d'un déclin vers les taux de compression élevés. Ce résultat avantage donc le format QCIF sur le format CIF. Il faut, par contre, vérifier ce résultat en menant des simulations avec d'autres profils.

IV.4.7 Effet de l'activité de la séquence vidéo sur les performances de codage :

Les séquences utilisées (Foreman , Bus, City et Football) ont le même format YUV 4 :2 :0, QCIF (176x144), 30 frames/seconde, algorithme de codage CABAC, search range de 32, 3 frames à coder.

Résultat de la simulation de codage de la séquence BUS:

QP	5	10	15	20	25	30	35	40	45
PSNRY	56.216	50.791	46.126	41.673	37.541	33.286	29.658	26.305	23.754
Débit (kbits/s)	4349.44	3027.12	2028.80	1366.16	886.88	519.92	297.52	146.64	74.16
TCG (s)	5.820	5.324	4.731	4.221	3.845	3.564	3.336	3.061	2.783
TEM (s)	1.617	1.660	1.633	1.654	1.668	1.690	1.728	1.670	1.497

Tableau IV.25 Résultats de performance du codage de la séquence Bus.

Résultat de la simulation du codage de la séquence Football:

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	56.139	50.801	46.255	41.553	37.092	32.822	29.138	25.908	23.027
débit (kbits/s)	4981.68	3537.68	2435.44	1733.28	1169.60	717.28	408.56	217.20	114.32
TCG (s)	6.238	5.532	5.005	4.579	4.166	3.812	3.470	3.203	2.911
TEM (s)	1.752	1.748	1.776	1.842	1.833	1.838	1.796	1.733	1.569

Tableau IV.26 Résultats de performance du codage de la séquence Football.

Résultat de la simulation du codage de la séquence city:

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	56.119	50.650	46.213	42.068	38.262	34.418	30.769	27.559	25.446
Débit (kbits/s)	3478.88	2237.12	1433.36	890.88	547.20	312.56	162.56	81.36	37.36
TCG (s)	5.089	4.531	4.024	3.561	3.274	3.071	2.909	2.787	2.494
TEM (s)	1.271	1.271	1.300	1.316	1.375	1.438	1.444	1.457	1.245

Tableau IV.27 Résultats de performance du codage de la séquence City.

La figure IV.14(a) montre que la qualité de la vidéo compressée est pratiquement indépendante de la mobilité de la séquence pour des taux de compression assez faibles. La différence de qualité apparait surtout pour de forts taux où on observe, par exemple, un écart de plus de 3 dB entre la séquence Foreman et la séquence Football pour un pas de quantification de 45. Par contre, lorsqu'il s'agit de débit, c'est la tendance inverse où la distinction est plus que nette pour des pas de quantification faibles (taux de compression faibles) et va en s'atténuant au fur et à mesure que la compression s'accentue. Le débit est d'autant plus important que la mobilité de la séquence est plus forte (figure IV.14(b)). Les résultats confirment que les séquences utilisées dans les simulations se classent selon la mobilité de la façon suivante :

Football > Bus > City > Foreman.

Il faut peut être signaler que les deux séquences City et Foreman sont d'une mobilité similaire mais la vidéo City est plus riche en détails.

Ce résultat est tout à fait logique. En effet, plus une séquence est mobile, plus les frames consécutifs diffèrent et plus importantes sont les données résiduelles et les vecteurs de mouvement d'où un débit plus grand.

Les temps de codage suivent la même tendance que les débits (figure IV.14(c)). Autrement dit, plus la séquence est mobile, plus le temps de codage est long et ceci pour les mêmes raisons données précédemment.

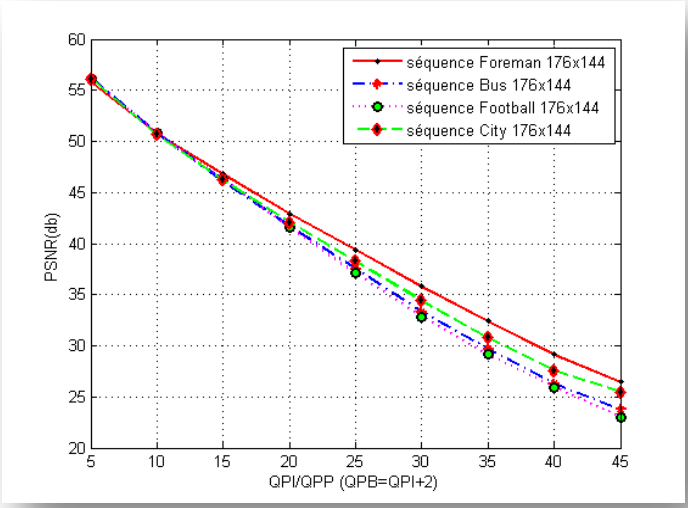


Figure IV.14(a) PSNR de différentes séquences vidéo.

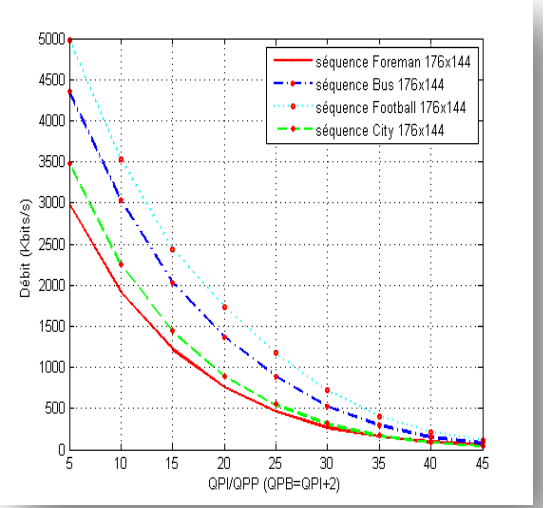


Figure IV.14(b) Débits de différentes séquences vidéo

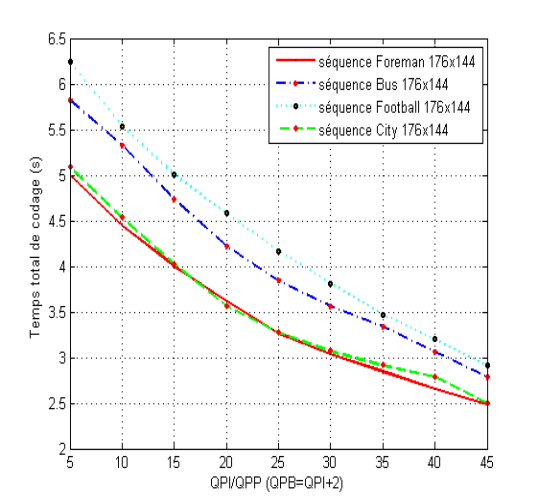


Figure IV.14(c) Temps de codage de différentes séquences vidéo

IV.4.8 Etude des différents modes de transformation spatiale :

Les différents modes de transformations sont parmi les nouveaux aspects introduits dans le standard H.264/AVC. On trouve :

- ✓ Une transformée spatiale entière 4x4, conceptuellement similaire à la DCT mais beaucoup plus simple et permet un décodage exact. Dans les simulations, cette option est notée "mode 0".
- ✓ Une transformée spatiale entière 8x8 permettant une compression plus efficace des régions fortement corrélées. Elle est notée "mode 2".
- ✓ Une sélection adaptative entre les deux transformées citées précédemment. Ce type de transformée est noté "mode 1".

Les simulations menées permettront de comparer les performances de ces trois modes en termes de qualité, débit et temps de codage. Elles sont menées sur les quatre séquences vidéo présentées au début de ce chapitre (Foreman, Football, City et Bus). Les principaux paramètres utilisés sont les suivants :

YUV Format : YUV 4:2:0

Frames to be encoded : 25

Freq. for encoded bitstream : 30.00

Image format : 176x144

Search range : 32

Entropy coding method : CAVLC

Motion Estimation Scheme : EPZS

Ordre de codage : 0 **8** 4 2 1 3 6 5 7 **16** 12 10 9 11 14 13 15 **24** 20 18 17 19 22 21 23

I **P** B B B B B B B **P** B B B B B B B **P** B B B B B B B

Ordre d'affichage : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Séquence Foreman :

Mode 0 :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.510	49.074	45.150	41.516	38.049	34.454	30.964	27.441	24.406
Débit (kbits/s)	2102.45	1216.44	677.11	379.28	215.33	122.12	68.15	36.05	19.96
TCG (s)	48.946	47.367	46.717	46.788	46.705	46.917	46.468	44.448	42.684
TEM (s)	31.346	32.126	33.658	35.490	36.655	37.738	37.961	36.356	34.718

Tableau IV.28 Résultats du codage de la séquence vidéo Foreman en mode 0

Mode 1:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	53.159	48.726	44.984	41.409	37.954	34.416	30.873	27.518	24.700
Débit (kbits/s)	2043.78	1165.41	657.85	370.83	209.83	118.31	66.12	35.72	20.03
TCG (s)	62.565	59.588	57.703	56.862	56.098	55.816	54.638	52.608	47.533
TEM (s)	34.539	35.446	37.215	39.376	40.716	41.872	41.801	40.374	36.154

Tableau IV.29 Résultats du codage de la séquence vidéo Foreman en mode 1

Mode 2:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	52.560	48.393	44.645	41.005	37.659	34.166	30.666	27.396	24.431
Débit (kbits/s)	2170.86	1233.60	703.28	393.84	223.46	123.83	66.54	35.44	19.52
TCG (s)	37.730	36.335	35.765	35.377	35.011	35.183	34.849	32.932	29.958
TEM (s)	22.519	23.066	24.211	25.293	26.055	27.005	27.231	25.829	23.278

Tableau IV.30 Résultats du codage de la séquence vidéo Foreman en mode 2

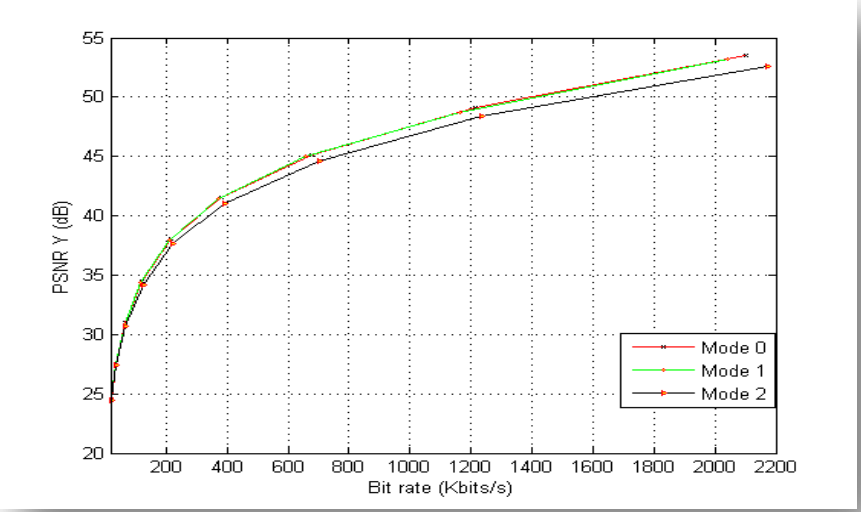


Figure IV.15.a Courbe débit-distorsion

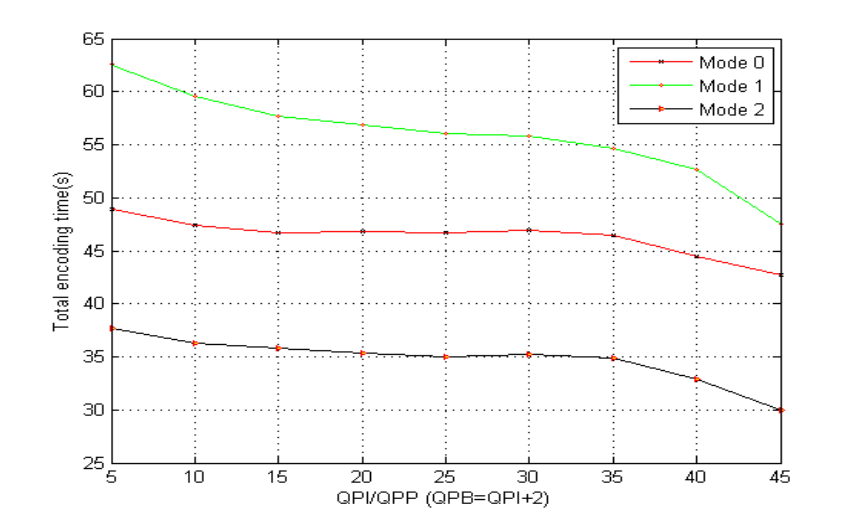


Figure IV.15.b Temps de codage global

Figure IV.15 Analyse de la séquence vidéo Foreman selon les 3 modes de transformation spatiale.

Les résultats des simulations de cette première séquence vidéo indiquent que les trois modes de transformations se valent en termes de qualité et de débit surtout dans les cas

de fortes compressions. La courbe débit-distorsion , plus connue sous l'appellation" R-D curve" dévoile un léger avantage du mode 0 et du mode 1. En effet, cette courbe montre que ces deux modes ont les mêmes performances pour l'ensemble des taux de compression. L'écart entre eux et le mode 2 est d'autant plus large que les compressions sont plus faibles. Néanmoins, la différence n'est jamais significative puisque la même qualité est obtenue avec les trois modes. Par contre, les courbes de la figure IV.15.c présentant les temps de codage relatifs aux trois modes, exhibent d'importantes différences. Elles indiquent que le mode 2 est le plus rapide. Il est suivi du mode 0 puis le mode 1. Ce dernier est en moyenne 1.61x et 1.21x plus long que le mode 2 et que le mode 0 respectivement. Le mode 0 est en moyenne 1.33x plus long que le mode 2. Cette tendance doit être vérifiée avec d'autres séquences avant de pouvoir confirmer sa généralité.

Séquence Football :

Mode 0 :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	54.337	49.435	44.839	40.075	35.581	31.325	27.611	24.100	20.828
Débit (kbits/s)	4204.27	3110.12	2233.87	1503.04	946.76	554.44	297.15	147.34	59.58
TCG (s)	80.049	75.417	73.838	72.844	70.969	67.425	62.346	56.776	48.714
TEM (s)	58.355	56.226	56.545	57.526	57.622	55.874	52.294	47.715	40.461

Tableau IV.31 Résultats du codage de la séquence vidéo Football en mode 0

Mode 1 :

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	54.128	49.094	44.519	39.808	35.296	31.053	27.419	23.994	20.833
Débit (kbits/s)	4172.72	3066.27	2198.74	1479.15	923.43	536.70	285.04	139.14	57.80
TCG (s)	94.790	92.640	90.215	87.500	83.878	79.326	72.941	65.561	56.554
TEM (s)	61.563	62.080	62.655	63.487	63.183	61.445	57.495	51.811	44.061

Tableau IV.32 Résultats du codage de la séquence vidéo Football en mode 1

Mode 2:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	53.449	48.468	44.010	39.232	34.879	30.775	27.180	23.786	20.749
Débit (kbits/s)	4565.26	3199.61	2312.38	1543.36	971.49	563.63	293.89	139.41	56.76
TCG (s)	59.442	58.333	56.984	55.462	53.538	50.949	47.307	43.006	36.577
TEM (s)	41.688	41.896	42.083	42.365	42.059	40.863	38.415	34.950	29.216

Tableau IV.33 Résultats du codage de la séquence vidéo Football en mode 2

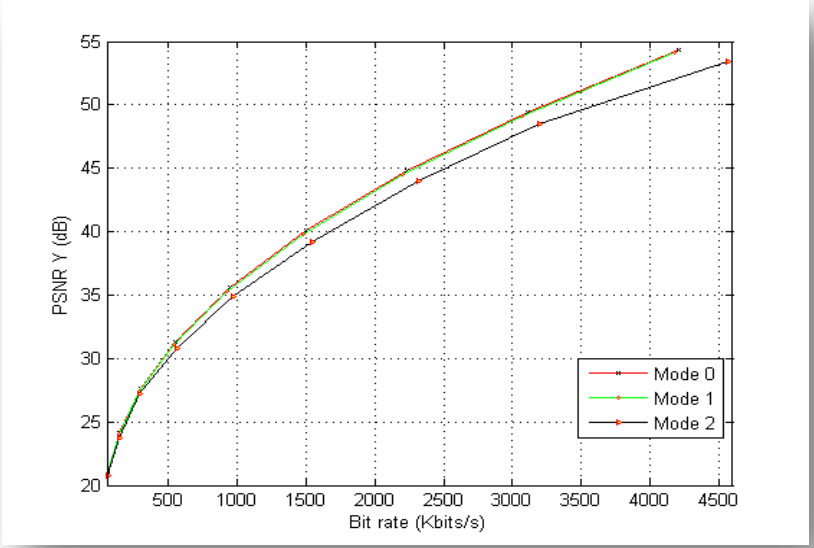


Figure IV.16.a Courbe débit-distorsion

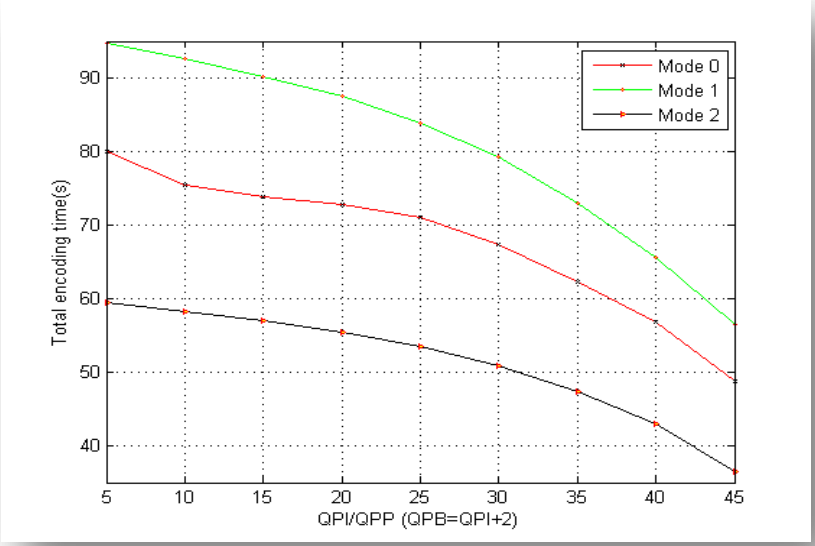


Figure IV.16.b Temps de codage

Figure IV.16 Analyse de la séquence vidéo Football selon les 3 modes de transformation spatiale.

Les simulations menées sur la séquence "FOOTBALL" confirment la même tendance observée avec la séquence "FOREMAN". La différence la plus importante étant toujours le temps de codage qui indique que celui du mode 1 est le plus long. Il est en moyenne 1.19x et 1.56x plus lent que le mode 0 et le mode 2 respectivement. Le temps de codage en mode 0 étant 1.32x plus long que celui du mode 2.

Séquence City :

Mode 0 :

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	53.614	48.832	44.384	40.334	36.798	33.271	29.865	26.469	23.905
Débit (kbits/s)	2343.28	1421.44	776.37	388.77	205.38	111.70	59.88	32.27	14.68
TCG (s)	48.814	46.394	45.448	46.191	48.355	49.734	49.281	46.516	39.951
TEM (s)	29.884	30.071	31.504	34.431	38.090	40.564	40.900	38.724	32.589

Tableau IV.34 Résultats du codage de la séquence vidéo City en mode 0

Mode 1:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	53.239	48.356	44.158	40.271	36.803	33.251	29.910	26.586	24.029
Débit (kbits/s)	2287.05	1356.71	745.38	381.40	203.97	109.69	58.70	31.39	14.85
TCG (s)	62.689	59.313	57.235	56.822	58.177	59.055	57.617	53.660	45.974
TEM (s)	33.038	33.500	35.301	38.550	42.516	45.103	45.012	42.063	35.119

Tableau IV.35 Résultats du codage de la séquence vidéo City en mode 1

Mode 2 :

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	52.641	48.065	43.829	39.737	36.319	32.747	29.520	26.421	24.019
Débit (kbits/s)	2432.75	1423.11	797.76	401.94	211.83	109.59	57.94	30.19	14.78
TCG (s)	37.627	36.272	35.527	35.926	36.415	36.819	36.482	34.189	28.812
TEM (s)	21.679	22.147	23.289	25.472	27.353	28.618	28.987	27.303	22.460

Tableau IV.36 Résultats du codage de la séquence vidéo City en mode 2

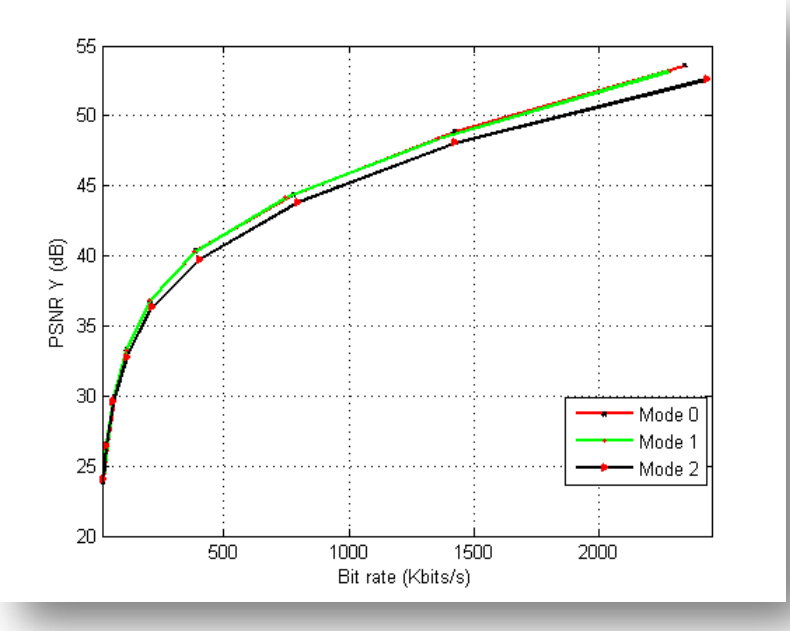


Figure IV.17.a Courbe débit-distorsion

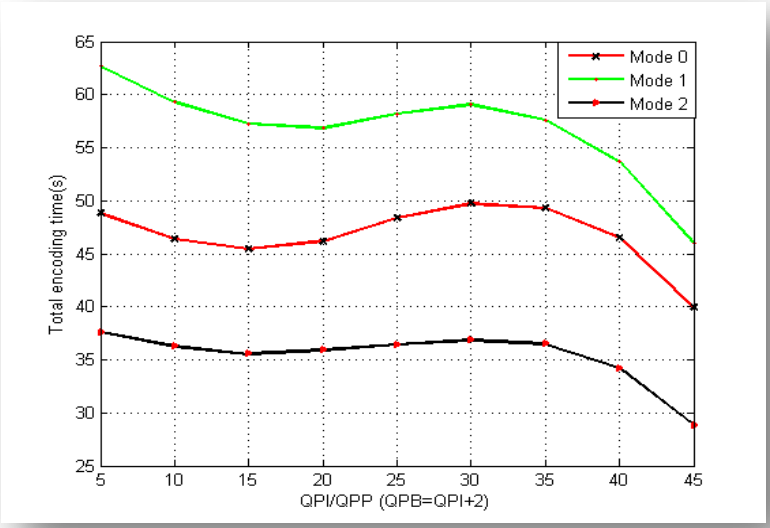


Figure IV.17.b Temps de codage

Figure IV.17 Analyse de la séquence vidéo City selon les 3 modes de transformation spatiale.

Séquence Bus:

Mode 0:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	54.186	49.264	44.639	40.010	35.751	31.559	27.874	24.615	21.346
Débit (kbits/s)	3369.19	2380.68	1600.63	992.34	594.22	333.10	176.35	91.65	43.52
TCG (s)	58.517	57.174	55.798	55.010	54.813	54.843	54.579	52.369	47.111
TEM (s)	38.625	39.038	39.623	41.015	42.764	44.313	45.294	43.901	39.300

Tableau IV.37 Résultats du codage de la séquence vidéo Bus en mode 0

Mode 1:

QP	5	10	15	20	25	30	35	40	45
PSNRY(dB)	53.900	48.747	44.200	39.718	35.510	31.418	27.809	24.595	21.499
Débit (kbits/s)	3331.56	2310.04	1539.80	958.18	570.12	322.15	171.74	86.87	42.34
TCG (s)	74.733	72.259	69.596	67.306	66.244	65.331	64.199	60.915	54.342
TEM (s)	42.890	43.296	44.006	45.427	47.554	49.163	50.027	48.193	42.658

Tableau IV.38 Résultats du codage de la séquence vidéo Bus en mode 1

Mode 2 :

QP	5	10	15	20	25	30	35	40	45
PSNRY	53.442	48.408	43.849	39.100	34.996	30.924	27.379	24.305	21.349
Débit (kbits/s)	3713.35	2432.86	1649.96	1018.92	606.99	333.54	171.73	86.04	41.91
TCG (s)	45.823	44.415	43.195	42.161	41.391	41.257	41.133	39.528	35.151
TEM (s)	28.670	28.776	29.194	29.970	30.814	31.943	32.869	31.976	28.223

Tableau IV.39 Résultats du codage de la séquence vidéo Bus en mode 2

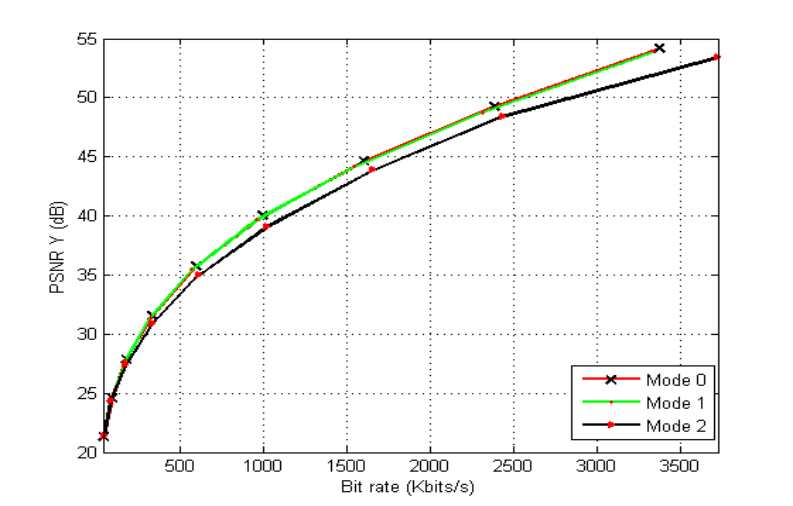


Figure IV.18.a Courbe débit-distorsion

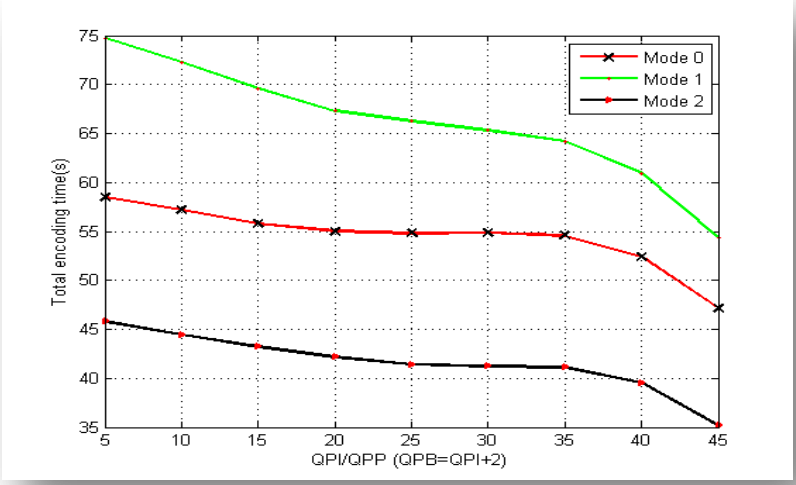


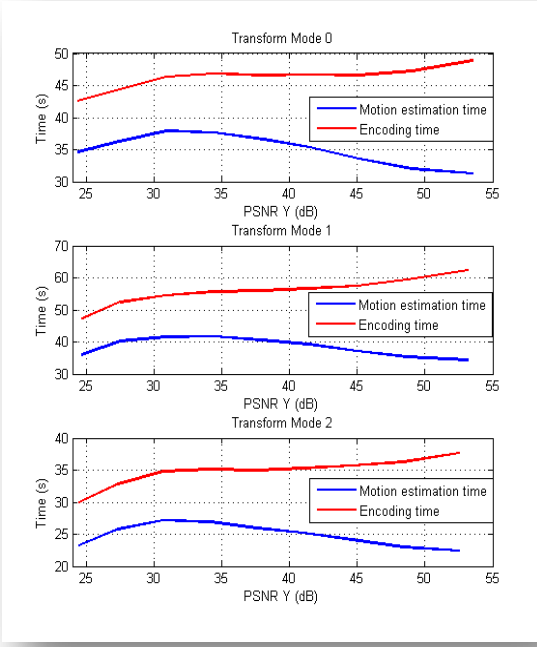
Figure IV.18.b Temps de codage

Figure IV.18 Analyse de la séquence vidéo Bus selon les 3 modes de transformation spatiale.

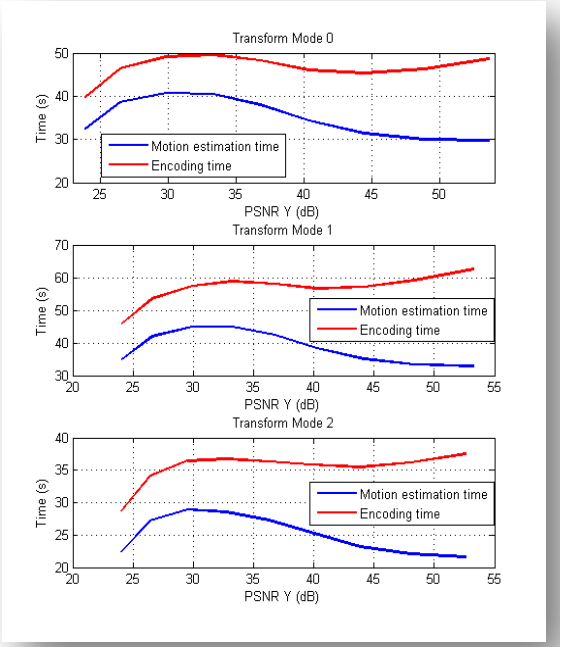
Les résultats précédents confirment les conclusions tirées des tests des séquences Foreman et Football malgré leurs différences.

La figure IV.19 montre la variation du temps de codage global et du temps d'estimation du mouvement en fonction de la séquence vidéo à coder et aussi en fonction du mode de transformation adopté. Pour chaque séquence, cette variation est la même indépendamment du mode de transformation, excepté, bien sur, pour les temps de codage globaux. En effet, les temps les plus longs restent ceux obtenus avec le mode 1. Le temps de codage est d'autant plus petit que le débit est faible. Cependant, il

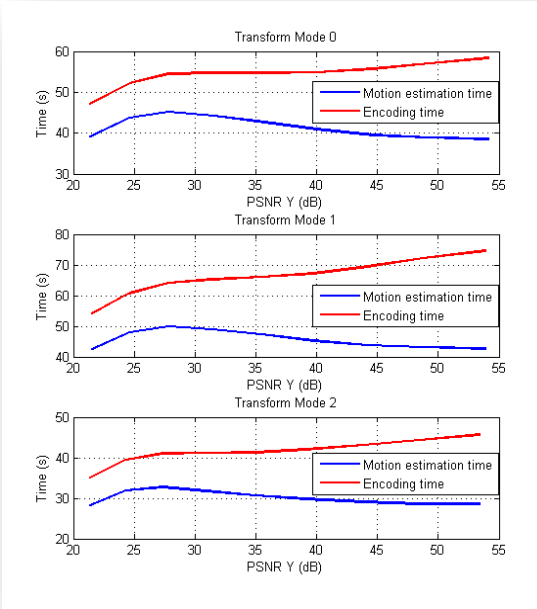
commence à augmenter au fur et à mesure que les détails se perdent. Dès qu'il n'y en a plus, il recommence à diminuer rapidement. Cette augmentation/diminution est un enchaînement de l'augmentation/diminution du temps d'estimation du mouvement représentant, en moyenne, 70% du temps de codage global.



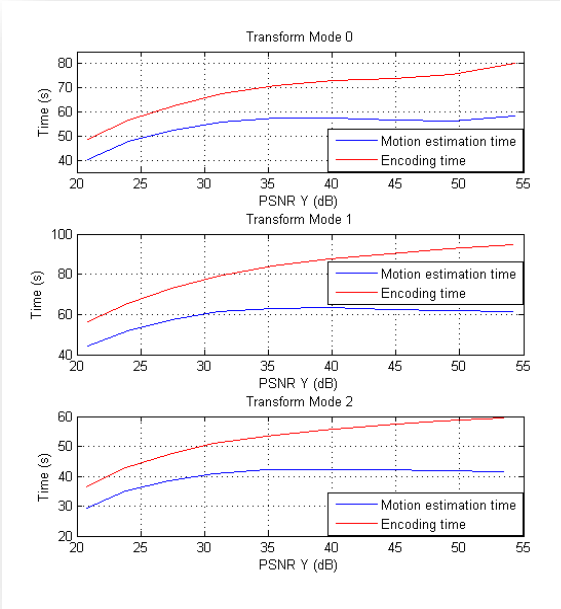
Séquence Foreman



Séquence City



Séquence Bus



Séquence Football

Figure IV.19 Variation du temps de codage et du temps d'estimation du mouvement en fonction de la séquence vidéo et du mode de transformation.

La figure IV.20 permet une comparaison perceptible de la qualité des frames obtenus par application des trois modes de transformation spatiale et à des débits différents.

Frame original de la séquence vidéo Bus.

Mode 0	Mode 1	Mode 2
		
QP=15 (psnr=44.639)	QP=15 (psnr=44.200)	QP=15 (psnr=43.849)
		
QP=30 (psnr=31.559)	QP=30 (psnr=31.418)	QP=30 (psnr= 30.924)
		
QP=35 (psnr=27.874)	QP=35 (psnr=27.809)	QP=35 (psnr=27.379)
		
QP=40 (psnr=24.615)	QP=40 (psnr=24.595)	QP=40 (psnr= 24.305)

QP=45 (psnr=21.346) QP=45 (psnr=21.499) QP=45 (psnr=21.35)

Figure IV.20 Evaluation subjective de la qualité de compression par application des trois modes de transformation spatiale et à des débits différents.

A la fin de ces tests, il est possible d'affirmer que le mode 0 (uniquement la transformée entière 4x4) est le seul concurrent du mode 2 (uniquement la transformée entière 8x8). En effet, d'après les courbes R-D, ce mode permet d'obtenir les meilleures performances du point de vue qualité et débit. Il est à égalité avec le mode 1et peut même le dépasser légèrement en qualité. Cependant, l'ensemble des résultats des simulations menées sur les quatre séquences vidéo prouvent que le mode 2 est le plus rapide. Le mode 0 le suit, alors que le mode 1 est le plus lent. La lenteur du mode 1 est due à l'algorithme de sélection du mode de la transformée. Ce type d'algorithmes est encore le sujet de beaucoup de travaux cherchant à les améliorer afin de réduire le temps de codage lorsque le mode 1 est utilisé [87, 88, 89]. Le tableau IV.40 récapitule les rapports des temps de codage entre les trois modes et pour les quatre séquences vidéo. Par exemple, pour la séquence City, il faut lire que le temps de codage lorsque le mode 1 est utilisé, est 1.60x plus long que lorsque le mode 2 est appliqué. Les rapports sont pratiquement constants malgré la diversité des séquences qui sont de nature et de mobilité complètement différentes.

Séquence vidéo	**Foreman**	**Football**	**City**	**Bus**
Mode 1/Mode 0	1.21x	1.19x	1.21x	1.21x
Mode 1/Mode 2	1.61x	1.56x	1.60x	1.59x
Mode 0/Mode 2	1.33x	1.32x	1.32x	1.31x

Tableau IV. 40 Comparaison des temps de codage pour les 3 modes de transformées et les 4 séquences vidéo testées.

Afin de consolider les résultats trouvés, un dernier test est mené sur la séquence vidéo Foreman en format CIF (352x288).

Séquence Foreman en format CIF :

Mode 0:

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	54.001	49.050	44.830	41.353	38.343	35.216	32.044	28.922	25.905
Débit (kbits/s)	9774.93	5622.61	2752.30	1281.88	651.52	346.90	188.07	105.42	55.70
TCG (s)	214.320	202.854	192.466	185.098	180.051	176.690	175.870	167.491	155.308
TEM (s)	139.824	140.306	140.646	141.737	141.946	141.957	142.476	136.399	125.390

Tableau IV.41 Résultats du codage de la séquence vidéo Foreman (format CIF) en mode 0

Mode 1:

QP	5	10	15	20	25	30	35	40	45
PSNRY (dB)	53.773	48.810	44.644	41.269	38.290	35.168	32.170	29.017	25.926
Débit (kbits/s)	9597.93	5436.81	2645.23	1234.49	625.44	331.69	181.89	100.48	55.34
TCG (s)	270.286	253.237	235.996	222.931	214.740	208.211	203.193	198.847	180.260
TEM (s)	152.700	154.030	154.486	155.784	156.422	155.648	154.141	151.347	135.717

Tableau IV.42 Résultats du codage de la séquence vidéo Foreman (format CIF) en mode 1

Mode 2:

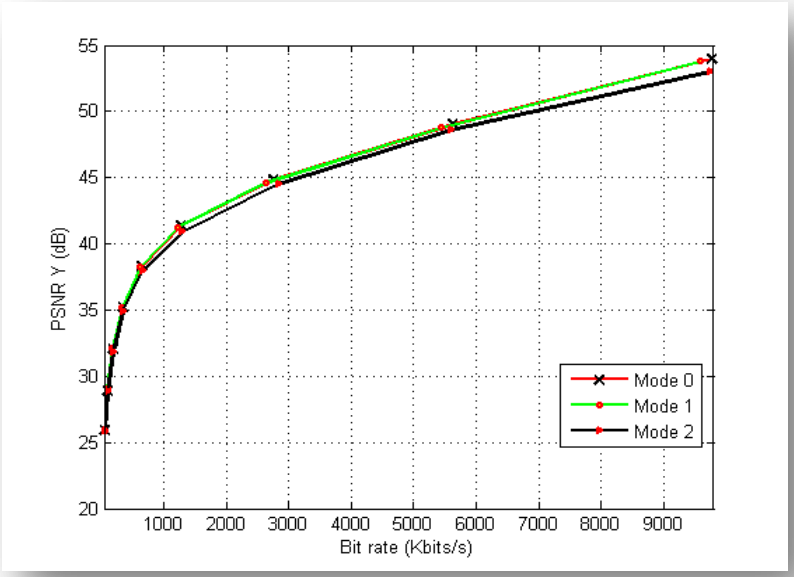
QP	5	10	15	20	25	30	35	40	45
PSNRY	53.021	48.633	44.531	40.918	37.992	34.891	31.870	28.820	25.888
Débit (kbits/s)	9742.14	5593.09	2829.99	1291.86	662.64	347.70	186.89	103.07	54.89
TCG (s)	160.181	151.009	142.834	137.065	133.293	130.722	128.396	122.154	111.627
TEM (s)	97.063	97.236	97.289	98.426	99.441	99.929	99.638	95.045	85.748

Tableau IV.43 Résultats du codage de la séquence vidéo Foreman (format CIF) en mode 2

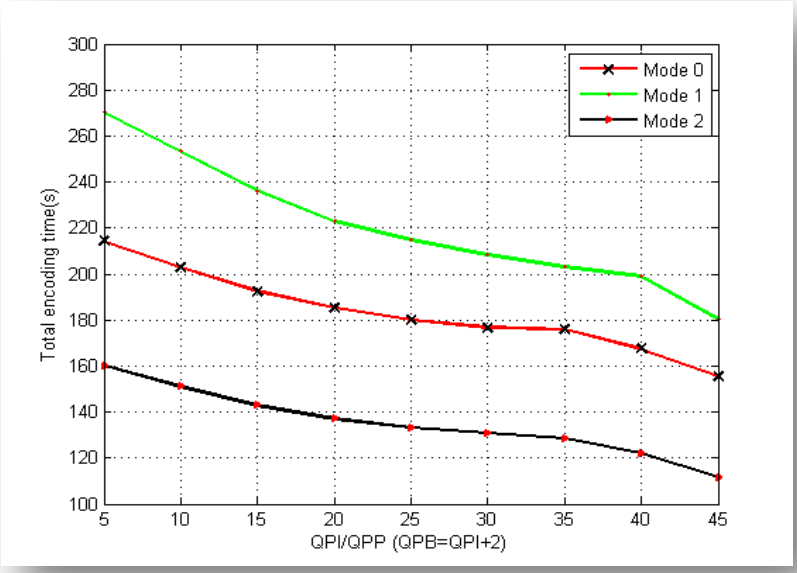
Ce dernier résultat prouve que la relation entre les temps de codage des différents modes de transformée reste valide avec le format CIF (voir figure IV.21). Le mode 1 est toujours 1.21x et 1.63x plus lent que le mode 0 et le mode 2 respectivement. Quant au mode 2, il permet encore d'obtenir le temps de codage le plus rapide. Une comparaison des autres mesures de la performance ont montré qu'en mode 0 , le bit rate pour le format CIF varie de 4,65x à approximativement 3x celui du format QCIF, avec une valeur moyenne de 3,45. Il est d'autant plus élevé que le pas de quantification est faible. Le tableau IV.44 résume cette comparaison.

Transformation	Mode 0	Mode 1	Mode 2
Débit (Kbits/s) (valeur moyenne)	4.65x – 3x (3.45x)	4.82X – 2.75x (3.54x)	4.49x – 2.80x (3.40x)
TCG (s) (valeur moyenne)	4.38x – 3.63x (3.95x)	4.32x – 3.71x (3.94x)	4.25x – 3.68x (3.88x)

Tableau IV44. comparaison des performances des 3 modes pour les formats QCIF et CIF.



Courbe débit-distorsion



Temps de codage

Figure IV.21 Analyse de la séquence vidéo Foreman en format CIF selon les 3 modes de transformation spatiale.

Pour le PSNR, les mêmes performances ont été obtenues avec les deux formats avec un très léger avantage pour le format CIF pour des pas de quantification élevés comme le montre la figure IV.22. La différence étant de 1.5 dB pour le mode 0, 1.23 dB pour le

mode 1 et 1.45 dB pour le mode 2. Ces différences sont insignifiantes pour l'estimation subjective de la qualité.

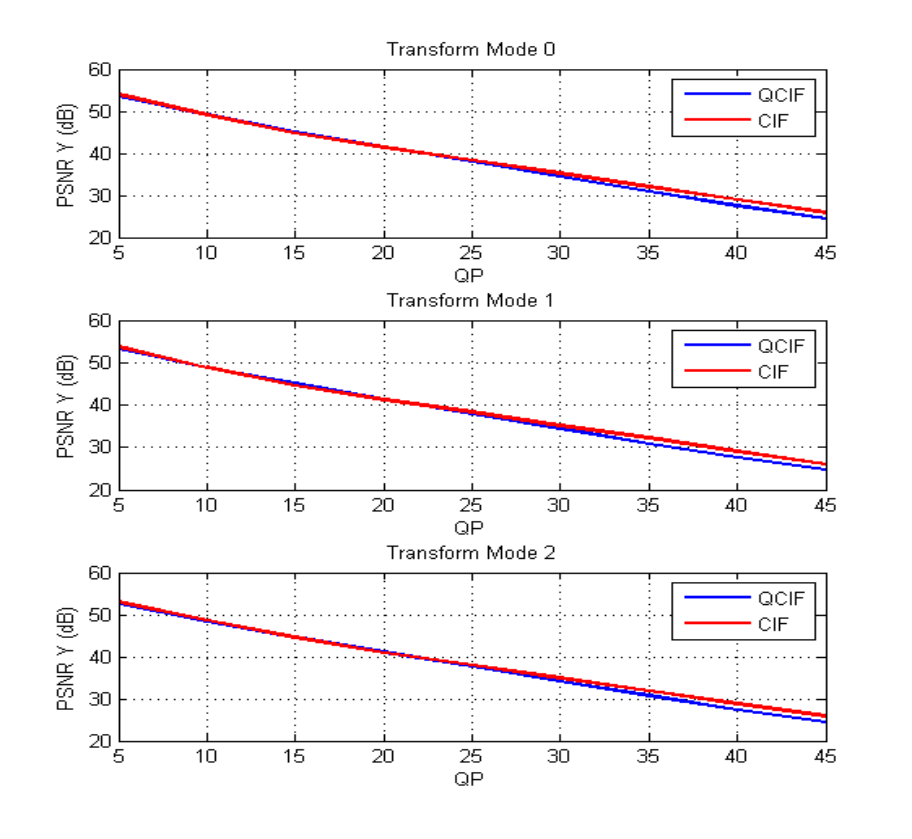


Figure IV.22 Comparaison de la qualité obtenue avec le fomat QCIF et CIF selon les 3 modes de transformation spatiale

IV.4.9 Etude du filtre anti-blocs:

Pour corriger l'effet de blocs et assurer une meilleure qualité vidéo à des débits faibles, le standard H.264/AVC fournit l'option de filtrage dans la boucle de codage. Deux seuils α et β sont définis dans la norme, permettant ainsi de prendre la décision de filtrage ou non du bord du bloc courant. Ces seuils α et β dépendent du paramètre de quantification QP et sont choisis en fonction des résultats des performances débit-qualité. Ils augmentent avec l'augmentation de QP et vice-versa. L'encodeur de la plateforme JM 18.2 fournit deux paramètres contrôlables α_offset et β_offset consentant ainsi d'élargir les possibilités de contrôle du filtrage. Les seuils α et β sont alors des fonctions de QP et de α_offset et β_offset respectivement. Les valeurs de α_offset et β_offset varient de -6 à +6 par pas de 1. Les valeurs négatives correspondent à un

filtrage faible alors que les valeurs positives permettent un filtrage fort. Ces valeurs s'appliquent indépendamment aux frames codés I, P et B.

Le nombre de simulations à mener pour déterminer le meilleur filtrage est alors très élevé. Dans notre cas, pour chaque séquence, puis pour chaque valeur du débit, correspondant à une valeur de QP, le nombre de tests à conduire est limité à 13 correspondants à 13 filtrages différents. La figure IV.23 montre un frame de la séquence vidéo "Foreman" codé avec QP=36 et non filtré (Figure IV.23.b). Le même frame filtré avec deux valeurs différentes de α_offset et β_offset est représenté sur les figures IV.23.c et IV.23.d.

Figure IV.23.a frame original

Figure IV.23.b frame non filtré (psnr=30.054)

Figure IV.23.c frame filtré (0,0) (psnr=30.087)

Figure IV.23.d frame filtré (+6,+6) (psnr=30.757)

Figure IV.23 Effet du filtre anti-blocs.

Cette amélioration de la qualité est atteinte au prix d'un temps de codage plus long et d'un débit légèrement plus élevé. Le tableau IV.45 résume le coût d'une telle performance sachant que ce test a été mené sur 60 frames seulement.

Séquence	Non filtrée	Filtrée (0,0)	Filtrée (+6,+6)
TCG (s)	84.836	85.188	85.996
Débit (kbits/s)	50.94	59.56	59.73

Tableau IV.45 Comparaison du temps de codage et du débit avec et sans filtrage.

L'augmentation du débit est due au fait que les informations concernant le filtre utilisé doivent être codées et transmises.

Il est important de signaler que la conduite de ces tests est parfois très difficile. Les paramètres de contrôle α_offset et β_offset permettant d'atteindre la meilleure qualité vidéo dépendent non seulement du facteur de quantification QP, mais aussi du mode de transformation, du nombre de frames de référence et de la séquence à coder. Souvent, les effets escomptés du filtrage ne sont pas perceptibles. A noter aussi que le filtrage le plus fort ne conduit pas forcément au meilleur résultat car il peut effacer un bord inhérent et naturel du frame ou faner les couleurs.

IV.5 Conclusion :

Dans ce chapitre, une étude et une analyse des performances du codeur H.264/AVC ont été conduites sur un certain nombre de paramètres. En conclusion, le jugement des performances dépend de l'application et du type de la vidéo. Ces performances sont fortement influencées par le choix des paramètres. Néanmoins, un compromis doit toujours être établi entre trois variables : qualité, débit et temps de codage. Les images codées B fournissent un bon gain de compression mais au détriment d'un temps de codage plus long. Les performances relatives aux trois modes de transformations spatiales et l'effet du filtre anti-blocs ont été analysées. Les résultats des différentes simulations menées ont indiqué que l'amélioration de la qualité, objectif final de toute compression, nécessite toujours un temps plus long. Les données compressées d'une vidéo sont soit stockées, soit transmises à travers un réseau. Dans beaucoup d'applications telles que les traitements en temps réel (on-line streaming) , le temps est primordial et il est exprimé en nombre de frames par seconde. Il est aussi nécessaire de contrôler le débit du flux codé et de l'adapter au débit disponible du mécanisme de

transmission. La qualité peut être recherchée et obtenue lorsqu'il n'y a plus de contrainte de temps. C'est les cas des traitements off-line.

Pour toutes ces raisons et étant donné qu'une implémentation hardware est toujours plus rapide qu'une implémentation software, nous préconisons la réalisation des parties du codeur sur des circuits programmables de type FPGA (Field Programmable Gate Array). Cette idée est développée dans le chapitre suivant où il sera question d'implémenter, sur FPGA, les filtres de type RIF correspondant à une implémentation de la DWT 1D sur un niveau. Des implémentations de la transformation en DCT entière 4x4, 8x8 et réelle 8x8 seront aussi proposées, analysées et comparées.

Chapitre V : Implémentation hardware

V.1 Introduction :

Les circuits programmables existent depuis longtemps, mais les circuits de type FPGA sont en train de révolutionner le monde, tout comme l'ont fait les DSPs quelques décennies auparavant. Des algorithmes très importants et fondamentaux en traitement numérique, implémentés sur des PDSPs (Programmable DSPS) ou des ASICs, sont maintenant implémentés sur des circuits FPGAs grâce à l'évolution de la technologie et l'augmentation fulgurante de la densité d'intégration [90].

Ce chapitre est partagé en deux grandes parties. La première présente d'abord ces circuits programmables et leurs caractéristiques ainsi que les outils d'aide à la conception de circuits sur FPGAs. Dans la seconde partie, deux implémentations sont exposées en détails, à savoir :

Une méthodologie générale [91] permettant d'implémenter tout filtre RIF sur FPGA. Elle propose principalement deux algorithmes qui dépendent des performances ciblées. Cette méthode peut être exploitée pour:

- ✓ Implémenter la transformée en ondelettes discrètes (DWT).
- ✓ Implémenter les filtres de lissage ou filtres anti-blocs (deblocking filters) utilisés dans le standard H.264/AVC.
- ✓ Implémenter les filtres utilisés dans la prédiction sous-pixélique.

Une implémentation hardware de la transformée en cosinus discrète (DCT) entière à deux dimensions opérant sur des blocs de taille 4x4 et préconisée dans la norme H.264/AVC. Les performances de cette implémentation seront comparées à celles de la DCT entière 8x8 et la DCT classique 8x8.

V.2 Les circuits programmables :

Actuellement, on trouve différentes familles de circuits programmables ou PLDs (Programmable Logic Devices) tels que les CPLDs (*Complex Logic Programmable Devices*), les ASICs, les DSPs et les FPGAs. Les CPLDs sont des composants pour la plupart reprogrammables électriquement ou à fusibles, peu onéreux et très rapides (fréquence de fonctionnement élevée). Cependant, ils disposent d'une capacité fonctionnelle moindre que les FPGAs. Par contre, ceux-ci sont des composants VLSI constitués de blocs mémoires vives, entièrement reconfigurables [92]. Ces blocs sont

structurés en LUT (*Look Up Table*), flip-flop, RAM et l'ensemble dispose d'un vaste système d'interconnexions. Le DSP a longtemps régné en maître absolu dans les applications de traitement numérique du signal. Mais, depuis quelques années, il doit faire face au nouveau concurrent : le FPGA. Celui-ci a déjà quelques décennies de carrière mais l'intégration de nouvelles fonctions (mémoires, processeurs, etc.) lui ont permis de sortir de ses applications classiques de traitement d'opérations de logique booléenne et d'accéder aux applications complexes de traitement du signal, d'images et de vidéos. On le trouve de plus en plus sur les cartes standards. La technologie a beaucoup évolué ces derniers temps et les FPGA sont devenus aujourd'hui de véritables processeurs numériques des signaux, qui viennent concurrencer les composants DSP et ASIC. Malgré tout, les FPGA ne s'imposent pas si facilement dans les applications de traitement du signal. Il y a deux raisons à cela. La première, c'est que pour programmer efficacement un FPGA, il faut repenser les algorithmes fondamentaux du traitement du signal. De plus, il ne suffit pas d'avoir des compétences en logiciel, il faut également des compétences en électronique numérique (au niveau "matériel"). La deuxième raison qui freine l'utilisation des FPGA, c'est que les outils de conception standard pour FPGA ne sont pas adaptés pour réaliser des applications de traitement du signal relativement complexes [93].

V.3 Les différents éléments d'un circuit FPGA [26]:

Les éléments constitutifs d'un FPGA sont toujours approximativement les mêmes quelle que soit l'architecture choisie et chaque fabricant apporte ses variantes. Les FPGAs se composent d'une matrice de blocs logiques élémentaires (CLB) permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées/sorties (IOB) dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs et des ressources d'interconnexion (programmable interconnect) totalement flexibles (figure V.1) [94].

Le programme, appelé " bitstream ", n'est pas destiné à être exécuté par un microprocesseur, mais à configurer des portes logiques et une logique d'interconnexions permettent de relier ces portes logiques entre elles.

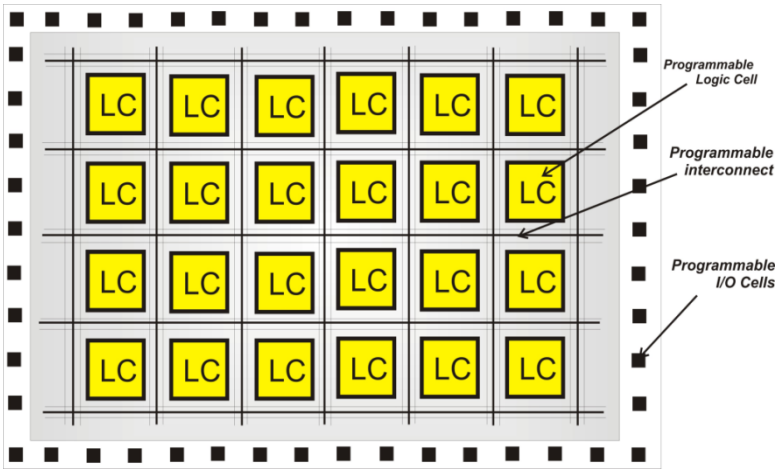


Figure V.1 Structure de base d'un circuit FPGA.

Généralement, ces éléments sont :

a) Les éléments logiques :

Il s'agit des blocs de base de tout circuit FPGA. On peut réaliser dans ces blocs toutes les opérations de logique combinatoire (dans la limite d'un nombre d'entrées). Ces blocs ont souvent la même constitution et cela malgré la différence de fabricants et d'architectures. La structure de base est celle de la figure V.2. Ces structures sont généralement constitués d'une ou plusieurs tables LUT (*Look Up Table*) qui contiennent, après configuration, la table de vérité de la fonction logique qu'elles doivent réaliser ou alors un ensemble de valeurs qui sont mémorisées comme dans une mémoire ROM. La taille des tables LUT est généralement de 16x1 (c'est à dire qu'elles disposent de 4 entrées). Les tables LUT sont suivies d'un registre de sortie, ce qui permet de synchroniser, si nécessaire, la sortie sur une horloge. La plupart des blocs logiques de bases sont munis d'une chaîne de propagation rapide de retenue afin de former de petits additionneurs rapides.

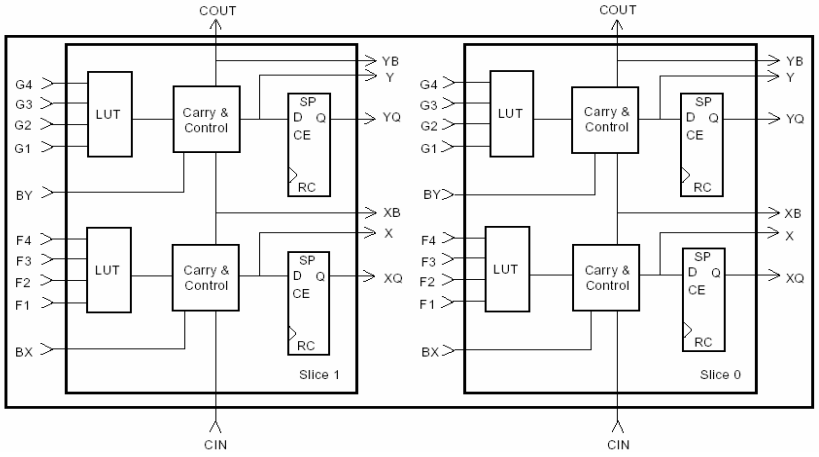


Figure V.2 Structure d'un élément logique.

b) Les éléments de mémorisation :

Actuellement, Les FPGAs sont utilisés pour des applications plus importantes qui demandent souvent des capacités de stockage (par exemple les applications du traitement vidéo). La nécessité d'intégrer des blocs de mémoire directement dans l'architecture des FPGAs est vite devenue capitale. De cette façon, les temps d'accès à la mémoire sont réduits. En effet, il n'est plus nécessaire de communiquer avec des éléments extérieurs au circuit.

c) Les éléments de routage :

Les éléments de routage sont les composants les plus importants dans les FPGAs. En fait, ces éléments représentent la plus grosse partie du silicium consommée sur la puce du circuit. Ces ressources sont composées de segments (de longueurs différentes) permettant de relier entre eux les autres éléments via des matrices de connexions. Le routage de ces ressources est un point critique du développement d'une application sur un FPGA. Ces éléments sont très importants puisqu'ils vont déterminer la vitesse et la densité logique du système. Par exemple, les matrices de routage sont physiquement réalisées grâce à des transistors de cellules SRAMs, qui ont une résistance et une capacité, ce qui entraîne l'existence de constantes de temps.

d) Les éléments d'entrées/sorties :

Le but des éléments d'entrées/sorties est de relier un circuit avec son environnement extérieur. Ceux-ci peuvent bénéficier de buffer ou d'autres éléments permettant la gestion des entrées et des sorties.

e) Les éléments de contrôle et d'acheminement des horloges :

L'horloge est un élément essentiel pour le bon fonctionnement d'un système électronique. Les circuits FPGA sont prévus pour recevoir une ou plusieurs horloges. Des entrées peuvent être spécialement réservées à ce type de signaux. Les circuits FPGA disposent des éléments d'asservissement des horloges (des PLLs ou des DLLs) afin d'avoir la même horloge dans tout le circuit (synchronisation des signaux). Ces éléments permettent de créer à partir d'une horloge d'autres horloges à des fréquences multiples de la fréquence de l'horloge incidente.

V.4 Etapes de l'implémentation d'un circuit sur FPGA :

Une implémentation sur FPGA suit, généralement, les étapes décrites dans l'organigramme ci-dessous :

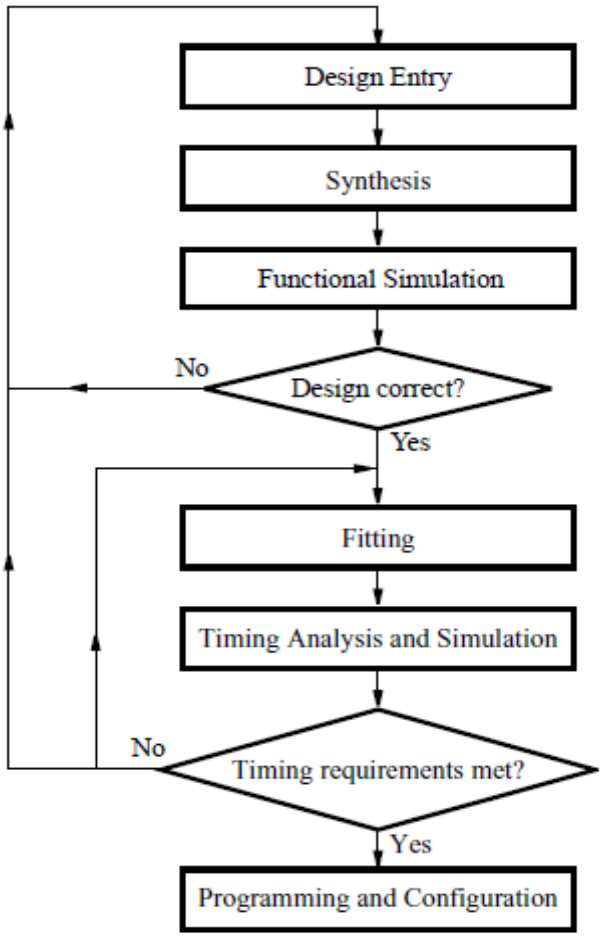


Figure V.3 Organigramme de conception et d'implémentation d'un circuit sur FPGA.

- Design Entry : C'est l'étape de conception où le circuit désiré est spécifié en utilisant soit un langage de description matérielle; tel que Verilog ou VHDL, où à l'aide d'un schéma.
- Synthesis : Cette étape de synthèse est assurée par un outil d'aide à la conception qui permet de synthétiser le circuit en un ensemble d'éléments logiques (LEs) nécessaires à la réalisation du circuit ainsi que les connections entre ces éléments. C'est ce qu'on appelle la "netlist".
- Functional Simulation : Le circuit est testé pour vérifier son fonctionnement. La simulation ne prend pas en compte l'analyse temporelle.
- Fitting : Il s'agit de placer les éléments logiques définis dans la netlist et les faire correspondre aux éléments logiques du circuit FPGA réel. Un choix des connections entre les éléments est aussi réalisé pendant cette étape.

- Timing Analysis : c'est l'analyse temporelle où les temps de propagation dans les différents chemins du circuit sont analysés pour fournir des indications sur les performances du circuit conçu.
- Timing Simulation : c'est l'étape de la simulation comportementale et temporelle.
- Programming and Configuration : Le circuit conçu est implémenté dans un circuit physique FPGA par programmation.

V.5 Les outils de conception :

V.5.1 Le langage de description matérielle :

La densité actuelle des fonctions logiques (portes et bascules) intégrées dans les PLDs est telle (plusieurs milliers, voire millions, de portes) qu'il n'est plus possible d'utiliser les outils de saisie d'un schéma pour développer les circuits. Les sociétés de développement et les ingénieurs ont voulu s'affranchir des contraintes technologiques des circuits et ont créé des langages dits de description matérielle de haut niveau (les HDL, Hardware Description Language). Deux d'entre eux ont émergé et sont couramment utilisés : VHDL (VHSIC Hardware Description Language) et Verilog. Ces deux langages bénéficient du support de la quasi-totalité des logiciels [95].

Dans les années 80, le département de la défense aux Etats-Unis fait un appel d'offre pour développer un langage de description matérielle numérique unique. Le langage VHDL (VHSIC Hardware Description Language) est inventé pour répondre à ces critères. Ce langage se base sur le VHSIC (Very High Speed Integrated Circuit) qui est un projet de recherche national mené par le groupe IBM/Texas Instruments / Intermetrics. Ce langage, ouvert au domaine public en 1985, devient une norme en 1987 sous la dénomination d'IEEE 1076-1987. Des changements minimes ont été apportés pour la seconde normalisation en 1993 qui porte le nom VHDL'93. La dernière évolution de la norme est la norme IEEE 1076-2001. Cependant, la norme IEEE 1076-2001 ne permet pas seule la description mixte (numérique et analogique). Le Verilog et le VHDL ont des capacités techniques équivalentes [96]. Le choix du langage est souvent dicté par la " culture " de l'équipe recherche. De plus, ce choix est parfois imposé par les outils disponibles, les logiciels de simulation et de synthèse, dont les langages associés sont fixés par des aspects économiques. VHDL a continué son évolution et la description analogique s'y est intégrée sous la référence IEEE 1076.6 (VHDL-AMS).

Le langage VHDL autorise trois niveaux de description [97] : le niveau structurel qui décrit le câblage des composants élémentaires,

- le niveau flot de données qui décrit les transformations d'un flot de données de l'entrée à la sortie,
- le niveau comportemental qui décrit le fonctionnement par des blocs programmes appelés Processus qui échangent des données au moyen de signaux comprenant des instructions séquentielles.

En VHDL [98], une structure logique est décrite sous la forme d'une paire de fonctions, d'une part une entité (ENTITY) et d'autre part, l'architecture (ARCHITECTURE) de la façon suivante :

```
LIBRARY LIBNAME;
USE LIBENAME.PACKAGENAME.ALL;
ENTITY ENTITY_NAME IS
PORT (
      SIGNAL_NAME :MODE SIGNAL_TYPE;
      .
      .
      .
      SIGNAL_NAME :  MODE SIGNAL_TYPE);
END ENTITY_NAME;

ARCHITECTURE ARCHITECTURE_NAME OF ENTITY_NAME IS
DECLARATION DE COMPOSANTS
   +  DECLARATION DES SIGANUX INTERNES
   +  AUTRE DECALARATION ...
BEGIN
   INSTRUCTIONS CONCURENTES;
   PROCESSUS;
   INSTANCES DE COMPOSANT;
END  ARCHITECTURE_NAME;
```

L'entité décrit les signaux d'entrées et de sorties de la structure ainsi que leurs noms et leurs types. Par contre l'architecture décrit le comportement de l'entité. Il est possible de créer plusieurs architectures pour une même entité où chacune décrit l'entité de façon différente.

V.5.2 Modelsim et Quartus II :

Modelsim est un logiciel de Mentor Graphics qui permet principalement, en utilisant le langage VHDL ou Verilog, de concevoir des systèmes numériques sous forme de projets, les décrire sur plusieurs niveaux (fichiers VHDL) et les simuler à l'aide de fichiers testbench. Modelsim ne permet que l'analyse fonctionnelle.

Quartus II est un environnement de conception de systèmes en vue de les implémenter sur FPGA. Tout comme Modelsim, Quartus II permet de décrire les circuits en VHDL mais aussi sous forme graphique ou schématique. Il permet de les simuler, de les analyser aussi bien fonctionnellement que temporellement afin de juger leurs performances et enfin de programmer la cible (FPGA).

V.6 Implémentation de la DWT à base de bancs de filtres :

La DWT est intensément utilisée dans le débruitage et la compression des signaux, des images et des vidéos. La DWT est basée sur des bancs de filtres RIF. Cette catégorie de filtres est aussi largement appliquée dans d'autres variétés de traitement numérique des signaux. C'est le cas des filtres de lissage ou filtres anti-blocs et des filtres de prédiction sous-pixélique introduits dans la norme H.264/AVC.

Les principales qualités des ces filtres sont leur stabilité et la linéarité de leur phase. Cette dernière est une propriété où la réponse de phase du filtre est une fonction linéaire de la fréquence, excluant ainsi une possibilité de distorsion dite de phase. Un filtre à phase linéaire a un retard de groupe constant, et par conséquent toutes les composantes fréquentielles auront des retards égaux et il n'y aura aucune distorsion due aux retards entre les composantes fréquentielles du signal filtré. Un filtre à phase non linéaire possède un retard de groupe variant en fonction de la fréquence et introduisant ainsi une distorsion de phase sur le signal filtré. La symétrie/l'antisymétrie est une propriété très importante des ondelettes bi-orthogonales de la famille "biorx.y" très appréciées en compression d'images et de vidéos.

L'opération d'un filtre est représentée mathématiquement par le produit de convolution. Il s'agit sous sa forme discrète d'une somme de produits . Les performances du filtre dépendent largement de l'architecture du multiplieur. Ainsi, différentes techniques sont utilisées pour implémenter des filtres RIF sur des circuits programmable et sont alors évaluées en termes de surface, vitesse et consommation d'énergie.

Dans la littérature, un grand nombre d'articles publiés au sujet d'implémentation sont basées sur la technique dite arithmétique distribuée (AD), une technique introduite par Croisier et al. [99], puis développée par Peled et Lui [100]. Dans [101], les auteurs ont présenté une décomposition systolique pour implémenter un filtre RIF à base de la technique d'arithmétique distribuée et ont trouvé qu'une adresse de taille 4 conduit à la meilleure réalisation en termes de surface et puissance. *Yajun Zhou et Ping zheng Shi* [102] ont implémenté un filtre RIF ayant 31 coefficients présentant une symétrie paire.

Ils ont alors comparé le cout d'une implémentation à arithmétique traditionnelle et celle à base de l'arithmétique distribuée. Ils ont reporté que l'AD permet d'économiser 50% en ressources matérielles. Dans [103], un multiplieur à coefficients constants a été conçu pour implémenter un algorithme de débruitage utilisant l'ondelette db2. Les données, utilisées comme adresses d'une LUT ont été divisées en deux parties: haute et basse. Un offset est alors calculé pour résoudre le problème des adresses négatives correspondantes à la moitié la plus significative. Cette idée convient très bien pour des filtres de faible support (nombre de coefficients, ou durée, limité) mais devient trop compliquée dans le cas de filtres de durée élevée.

La section suivante présente une méthodologie d'implémentation de filtres RIF quel que soit le type. Basée sur l'arithmétique distribuée, elle fournit au concepteur différentes techniques qui dépendent des performances désirées du filtre. A souligner le compromis à faire entre la surface et la puissance pour gagner en latence. Cette dernière est d'une importance extrême dans les applications temps réel telles que la diffusion live , la téléphonie, la vidéo, … etc.

V.6.1Concepts de base:

V.6.1.1Linéarité de la phase:

La fonction de transfert en z d'un filtre de longueur (p+1) ou d'ordre p est :

$$H(z)=\sum_{n=0}^{P} h_n z^{-n} \qquad (46)$$

Un tel filtre a une phase linéaire si sa réponse impulsionnelle possède un axe de symétrie : $h_n = h_{(p-n)}$ pour $0 \leq n \leq p$ ou un axe d'antisymétrie : $h_n = -h_{(p-n)}$ pour$0 \leq n \leq p$. Quatre classes de filtres RIF à phase linéaire peuvent être définies :

- Filtres RIF symétriques de longueur paire/impaire.
- Filtres RIF antisymétriques de longueur paire/impaire.

La méthodologie proposée sera appliquée pour l'implémentation du banc de filtres CDF9/7 (bior4.4). Les retards de groupe sont calculés afin de les comparer avec les résultats de l'implémentation hardware. Pour le filtre passe bas à 9 coefficients, en utilisant l'équation 46 avec p=8 et en exploitant sa symétrie, nous pouvons réécrire H(Z) comme suit:

$$H(z) = z^{-4}\left[h_0\left(z^4+z^{-4}\right) + h_1\left(z^3+z^{-3}\right) + h_2\left(z^2+z^{-2}\right) + h_3\left(z+z^{-1}\right) + h_4\right] \quad (47)$$

La réponse fréquentielle correspondante, en utilisant les fréquences normalisées (pas d'échantillonnage supposé égal à 1 et $0 \leq f \leq 1$ pour une période de la réponse fréquentielle), est donnée par:

$$H(e^{j\omega}) = e^{-j4\omega}\left[2h_0\cos(4\omega) + 2h_1\cos(3\omega) + 2h_2\cos(2\omega) + 2h_3\cos(\omega) + h_4\right] \quad (48)$$

La mesure standard de la linéarité de la phase d'un système est le retard de groupe défini par :

$$\tau(\omega) = -d\phi(\omega)/d\omega \quad (49)$$

ω étant la pulsation normalisée ($\omega=2\pi f$, ou $0 \leq f \leq 1$) , $\phi(\omega)$ étant la phase.

La fonction de phase du filtre à 9 coefficients est $\phi(\omega) = -4\omega + \theta$ (θ est 0 ou $\pi/2$) et c'est une fonction linéaire de ω. Le retard de groupe est $\tau(\omega) = -4$ qui indique un retard de groupe de 4 échantillons.

Le même calcul donne, pour le filtre passe haut à 7 coefficients, un retard de groupe de 3 échantillons.

V.6.1.2 Structure polyphase :

La structure générale d'un filtre RIF symétrique est décrite par l'équation 50, appelée produit de convolution :

$$y[n]=\begin{cases}\sum_{i=0}^{n} h_i x[n-i] & n \leq P \\ \sum_{i=0}^{P} h_i x[n-i] & n > P\end{cases} \quad (50)$$

Où h_0, $h_{1,\ldots,}h_P$ représentent les (P+1) coefficients de la réponse impulsionnelle du filtre et x[n] représente la séquence de données à filtrer. P caractérise la longueur de la réponse impulsionnelle du filtre.

Une approche pour implémenter les filtres RIF décimateurs ou implémenter l'opération de convolution peut être réalisée avec soit une structure non polyphasée correspondant à l'équation 50, soit avec une structure polyphasée [[104], [105]]. Il est à rappeler que l'invention de la représentation polyphase d'un filtre par Bellanger et al. [106] a permis la simplification de plusieurs résultats théoriques en traitement du signal et a conduit à des implantations plus simples des bancs de filtres. Une implémentation polyphase d'un

filtre RIF décimateur sépare sa réponse impulsionnelle en M sous filtres différents, où M représente le facteur de décimation ou de sous-échantillonnage. L'efficacité du filtrage polyphase réside dans le fait que des valeurs spécifiques des données d'entrée sont multipliées uniquement par des valeurs spécifiques de la réponse impulsionnelle. Pour *M=2*, les valeurs x[0],x[2], x[4], ... sont combinées uniquement avec les coefficients du filtre $h_0,h_2,h_4,...$, et les valeurs x[1],x[3], x[5],... sont combinées avec les coefficients $h_1,h_3,h_5,...$

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + h_5 z^{-5} + \dots h_p z^{-p} \quad (51)$$

Pour un filtre de longueur impaire (p pair):

$$H(z) = \left(h_0 z^{-2} + h_2 z^{-2} + h_4 z^{-4} + \dots + h_p z^{-p}\right) + \left(h_1 z^{-1} + h_3 z^{-3} + h_5 z^{-5} + \dots + h_{(p-1)} z^{-(p-1)}\right) \quad (52)$$

$$H(z) = \left(h_0 z^{-2} + h_2 z^{-2} + h_4 z^{-4} + \dots + h_p z^{-p}\right) + z^{-1}\left(h_1 + h_3 z^{-2} + h_5 z^{-4} + \dots + h_{(p-1)} z^{-(p-2)}\right) \quad (53)$$

Pour un filtre de longueur paire (p impair):

$$H(z) = \left(h_0 z^{-2} + h_2 z^{-2} + h_4 z^{-4} + \dots + h_{(p-1)} z^{-(p-1)}\right) + \left(h_1 z^{-1} + h_3 z^{-3} + h_5 z^{-5} + \dots + h_p z^{-p}\right) \quad (54)$$

$$H(z) = \left(h_0 z^{-2} + h_2 z^{-2} + h_4 z^{-4} + \dots + h_{(p-1)} z^{-(p-1)}\right) + z^{-1}\left(h_1 + h_3 z^{-2} + h_5 z^{-4} + \dots + h_p z^{-(p-1)}\right) \quad (55)$$

Dans les deux cas, l'équation (46) peut être décomposée et exprimée sous forme de filtres opérant à la moitié du taux initial sur les échantillons pairs er impairs. Elle peut alors être réécrite comme suit :

$$H(z) = H_e(z^2) + z^{-1} H_o(z^2) \quad (56)$$

H_e est la partie paire et H_o est la partie impaire.

Pour un filtre de longueur impaire (p pair):

$$\left.\begin{aligned} H_e(z^2) &= \sum_{n=0}^{(\frac{P}{2})} h_{2n} z^{-2n} \\ H_o(z^2) &= \sum_{n=0}^{(\frac{P}{2})-1} h_{(2n+1)} z^{-2n} \end{aligned}\right\} \quad (57)$$

Pour un filtre de longueur paire (impair):

$$\left.\begin{aligned} H_e(z^2) &= \sum_{n=0}^{(p-1)/2} h_{2n} z^{-2n} \\ H_o(z^2) &= \sum_{n=0}^{(p-1)/2} h_{(2n+1)} z^{-2n} \end{aligned}\right\} \quad (58)$$

La figure suivante schématise la structure transversale et la structure polyphase d'un filtre

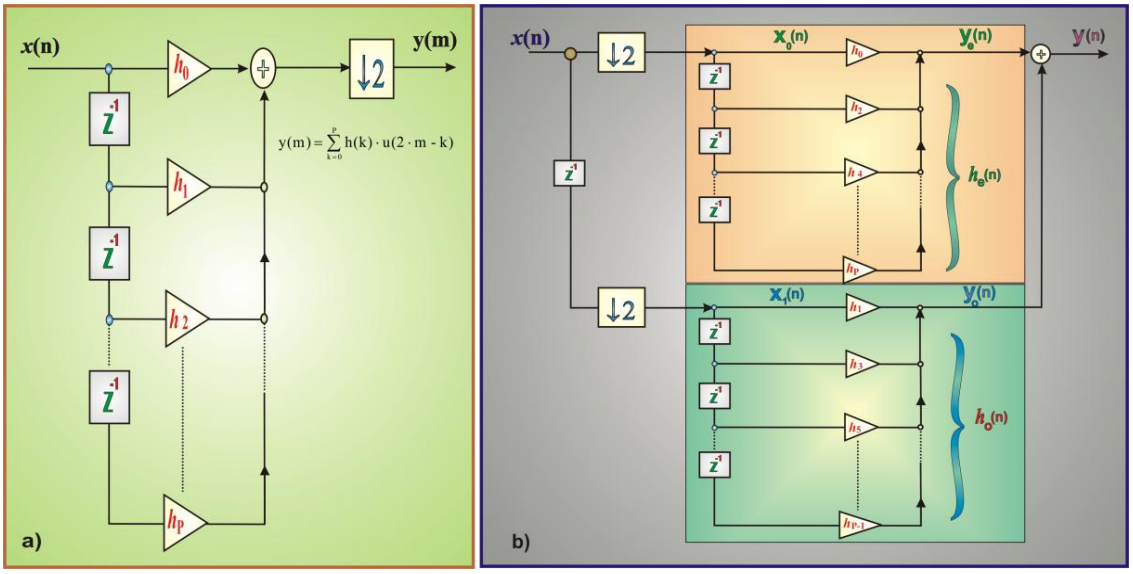


Figure V.4 (a) structure transversale suivie d'un décimateur , (b) structure polyphase.

Dans la structure non polyphase (figure V.4 (a)), le filtrage est suivi par une opération de décimation. Dans ce cas, la moitié des échantillons traités par le filtre sont tout simplement ignorés. Ceci est donc une perte de temps et de mémoire et montre l'inefficacité de cette structure. En contrepartie, la structure polyphase inverse les procédés (figure V.4 (b)). Le sous échantillonnage commence et est suivi par une séparation des échantillons pairs et impairs. Ceci permet un traitement en parallèle des deux ensembles d'échantillons simultanément par les deux parties paire et impaire du filtre.

Mathématiquement, la convolution est un produit scalaire et dans le cas d'un filtre RIF, il peut être écrit de la façon suivante (pour un filtre avec (p+1) coefficients):

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} x_0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & x_0 & 0 & 0 & \dots & 0 \\ x_2 & x_1 & x_0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_P & x_{P-1} & x_{P-2} & x_{P-3} & \dots & x_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1} & x_{N-2} & x_{N-3} & x_{N-4} & \dots & x_{N-P-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{P-1} \\ h_P \end{bmatrix} \quad (59)$$

Après décimation par un facteur de 2 et en supposant qu'on garde les résultats d'ordre pair, le système (59) se réduit au système (60) :

$$\underline{\boldsymbol{y}}(\downarrow 2) = \begin{bmatrix} y_0 \\ y_2 \\ \vdots \\ y_{N-3} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} x_0 & 0 & 0 & 0 & \dots & 0 \\ x_2 & x_1 & x_0 & 0 & \dots & 0 \\ x_4 & x_3 & x_2 & x_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_P & x_{P-1} & x_{P-2} & x_{P-3} & \dots & x_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1} & x_{N-2} & x_{N-3} & x_{N-4} & \dots & x_{N-P-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{P-1} \\ h_P \end{bmatrix} \quad (60)$$

La structure polyphase permet alors la décomposition du système (60) en phase paire et phase impaire. Le système précédent peut s'écrire comme suit :

$$\underline{\boldsymbol{y}}(\downarrow 2) = \begin{bmatrix} x_0 & 0 & 0 & 0 & \dots & 0 \\ x_2 & x_0 & 0 & 0 & \dots & 0 \\ x_4 & x_2 & x_0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_P & x_{P-2} & x_{P-4} & x_{P-6} & \dots & x_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1} & x_{N-3} & x_{N-5} & x_{N-7} & \dots & x_{N-P-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_2 \\ \vdots \\ h_{P-2} \\ h_P \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & 0 & 0 & 0 & \dots & 0 \\ x_3 & x_1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{P-1} & x_{P-3} & x_{P-5} & x_{P-7} & \dots & x_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-2} & x_{N-4} & x_{N-6} & x_{N-8} & \dots & x_{N-P} \end{bmatrix} \begin{bmatrix} h_1 \\ h_3 \\ \vdots \\ h_{P-3} \\ h_{P-1} \end{bmatrix} \quad (61)$$

A cette étape, la symétrie du filtre permet encore une simplification du système qui s'écrit alors:

$$\underline{\boldsymbol{y}}(\downarrow 2) = \begin{bmatrix} x_0 & 0 & \cdots & 0 \\ x_2 & x_0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ x_{(\frac{p}{2})} & x_{(\frac{p}{2}-2)} & \cdots & x_0 \\ \vdots & \vdots & \cdots & \vdots \\ (x_{N-1} + x_{N-P-1}) & (x_{N-3} + x_{N-P+1}) & \cdots & x_{N-(\frac{P}{2}+1)} \end{bmatrix} \begin{bmatrix} h_0 \\ h_2 \\ \vdots \\ h_{P/2} \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ x_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ x_{(\frac{p}{2}-1)} & x_{(\frac{p}{2}-3)} & \cdots & x_1 \\ \vdots & \vdots & \cdots & \vdots \\ (x_{N-2} + x_{N-P}) & (x_{N-4} + x_{N-P+2}) & \cdots & \left(x_{N-\frac{P}{2}} + x_{N-\frac{P}{2}+2}\right) \end{bmatrix} \begin{bmatrix} h_1 \\ h_3 \\ \vdots \\ h_{(\frac{P}{2})-1} \end{bmatrix} \quad (62)$$

V.6.1.3 L'arithmétique distribuée :

Le multiplieur à base d'arithmétique distribuée et le multiplieur parallèle sont d'importantes techniques utilisées dans le domaine des implémentations sur FPGAs. Ils sont intensément exploités dans le calcul des sommes de produits [[101], [102]]. Le multiplieur parallèle utilisant le concept de LUT traite tous les bits de la donnée en entrée simultanément au prix de larges ressources matérielles [103]. A l'opposé, une structure à base d'arithmétique distribuée, traite le flux de données en entrée bit par bit, profitant généralement des résultats des opérations menées sur les bits précédents. L'avantage d'une conception utilisant l'arithmétique distribuée est que tous les bits passent par la même architecture logique, résultant en une réduction significative du hardware. L'approche série requière pratiquement $(1/M)^{ième}$ des ressources hardwares nécessaires pour le schéma équivalent parallèle à M bits. En contre partie, une conception série prend environ M cycles d'horloge, alors que l'équivalent parallèle s'exécute en un cycle. Cependant, le produit temps-hardware d'une structure série est souvent inférieur à celui de la structure parallèle équivalente. La raison est que les retards entre les registres du premier schéma sont beaucoup plus faibles que ceux du deuxième. Ceci fait qu'une machine série peut opérer à une plus grande cadence puisqu'elle tend à avoir des routages très localisés, souvent vers une seule destination. La machine parallèle, par contre, a recourt à plusieurs signaux étendus à travers la taille des données fournies par chaque composant séparément. Les ressources limitées et souvent lentes des FPGAs justifie d'avantage le traitement série et il y a des cas où le débit global d'une architecture série dépasse celui de l'architecture parallèle.

Pour revenir à l'arithmétique distribuée, et dans le cas de données non signées en entrée du filtre, elles sont représentées en binaire sur B bits et exprimées sous la forme:

$$x[n]=\sum_{b=0}^{B-1} x_b[n]\, 2^b \qquad \text{avec } x_b[n]\in[0,1] \tag{63}$$

$x_b[n]$ étant le $b^{ième}$ bit de l'échantillon $x[n]$.

En remplaçant l'équation (63) dans l'équation (50), on trouve:

$$y[k]=\sum_{n=0}^{P}[h(n)\sum_{b=0}^{B-1} x_b\,(k-n)2^b] \tag{64}$$

L'équation précédente peut être réécrite d'une façon différente:

$$y[k]=\sum_{b=0}^{B-1} 2^b\ [(\sum_{n=0}^{P} h[n]\, x_b[k-n])] \tag{65}$$

Par exemple, dans le cas de 3 coefficients, 3 échantillons et une représentation sur 3 bits, l'équation (65) donne:

$$y[k]=\begin{cases} 2^0\left[h(0)\cdot x_0(k)+h(1)\cdot x_0(k-1)+h(2)\cdot x_0(k-2)\right]+ \\ 2^1\left[h(0)\cdot x_1(k)+h(1)\cdot x_1(k-1)+h(2)\cdot x_1(k-2)\right]+ \\ 2^2\left[h(0)\cdot x_2(k)+h(1)\cdot x_2(k-1)+h(2)\cdot x_2(k-2)\right] \end{cases} \quad (66)$$

Une implémentation possible pourrait être celle de la figure V.5.

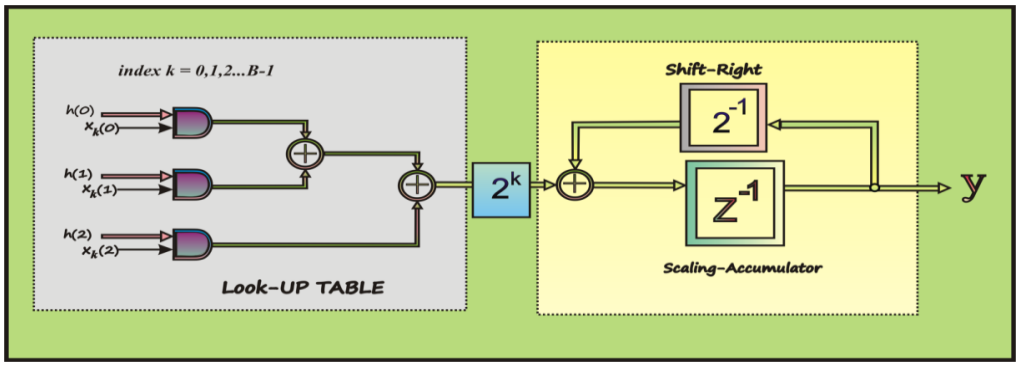


Figure V.5 Structure du noyau d'un multiplieur à base d'arithmétique distribuée.

La structure du filtre RIF telle que décrite par l'équation (50) consisterait, pour un seul échantillon filtré, en (P+1) multiplieurs et (P) additionneurs ou (P+1) MAC blocs (MAC=Multiply And Accumulate). En général, s'il y a (P+1) multiplieurs dans un circuit et si le retard du multiplieur est T secondes, alors un filtre avec (P+1) coefficients ne peut fonctionner à une fréquence d'échantillonnage supérieure à $1/T(P+1)$. L'arithmétique distribuée permet cependant un grand gain en ressources hardware par l'utilisation des tables de consultation (LUT). En fonction du flux d'entrée, qui n'est que de quelques possibilités, la LUT stocke les coefficients du filtre et leurs combinaisons arithmétiques linéaires, alors que les bits correspondant au même poids fournissent l'adresse. La somme pondérée est réalisée par une accumulation suivie d'une mise à l'échelle. Ce procédé est illustré par la figure V.6.

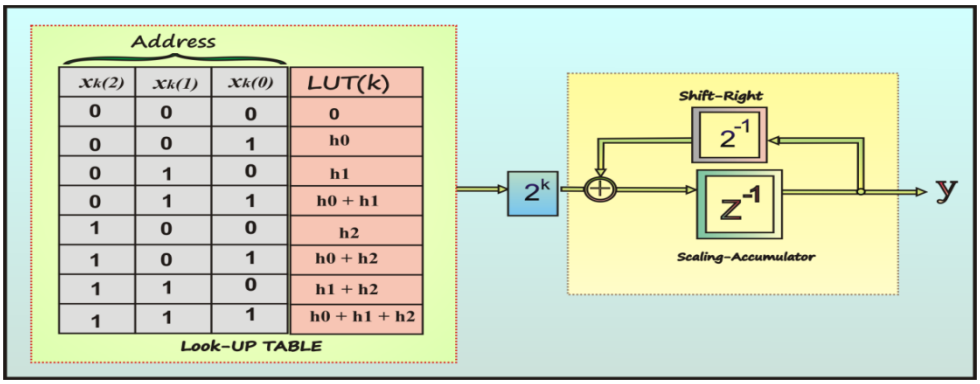


Figure V.6 Implémentation de la technique d'AD à base de LUT.

Pour implémenter cette architecture, il est nécessaire de trouver l'équation de récurrence permettant de gérer correctement la pondération imposée par le choix de la taille des données. Ainsi, le produit interne peut être réécrit comme suit :

$$Acc(k) = 2^{-1}Acc(k-1) + 2^{M}LUT(k) k = 1,2,\dots M \quad (67)$$

M est au moins égal à la résolution choisie +1.

D'après la figure V.6, l'équation (67) peut être reformulée sous la forme:

$$y(k) = 2^{-1}y(k-1) + 2^{M}LUT(k) \quad (68)$$

V.6.2 Méthodologie à base d'arithmétique distribuée classique (CDA):

L'implémentation sur FPGA proposée est modulaire et peut ainsi être réutilisée partiellement ou entièrement comme modèle ("template"). Les quatre blocs sont :

- Un séparateur (splitter)
- Un registre de données (data register)
- Une mémoire (ROM ou look-up table)
- Un contrôleur (controller)

V.6.2.1 Le séparateur:

Ce composant implémente la structure polyphase du filtre. C'est la première étape du processus de conception et son rôle est de séparer les données à filtrer en échantillons d'ordre pair et échantillons d'ordre impair qui seront traités en parallèle. Le séparateur est implémenté en tant qu'une machine à deux états qui lit les données en entrée sous le contrôle d'un signal de chargement "*load*" et les transfère à une sortie paire et une sortie impaire d'une façon alternée. La fréquence du signal *load* est liée au temps de traitement d'un échantillon.

$F_{load} = {}^{1}/_{T_{ps}}$ $avec T_{ps}=$ temps de traitement d'un échantillon (sample processing time).

Un signal sortie active (output enable signal ou *"OEN")* est alors généré et sa fréquence est la moitié de celle du signal *"load"*.

V.6.2.2 Le composant registre de données :

Dans les sections suivantes, les valeurs données sont applicables à tout filtre RIF à phase linéaire aussi bien symétrique qu'antisymétrique présentant un nombre de coefficients pair, alors que les valeurs correspondantes à un nombre de coefficients impair sont précisées entre crochets.

Les échantillons d'ordre pair sont chargés dans des registres séquentiellement par une opération de décalage en parallèle. Dès que les $(P/2+1)$ $[(P+1)/2]$ échantillons pairs sont lus, le premier d'entre eux est rajouté[1] au dernier, le second est rajouté à l'avant dernier et ainsi de suite. Si le filtre est d'ordre pair, l'échantillon central est gardé seul car il représente la médiane des données et correspond au point de symétrie du filtre. Le résultat est $round((P+2)/4)$ $[round((P+1)/4)]$ données parallèles. Ces données sont sérialisées pour fournir une adresse paire convoyée par $round$ $((P+2)/4)$ $[round((P+1)/4)]$ bits, et permettant l'accès à la LUT paire. Round étant toute fonction qui donne l'entier le plus proche vers l'infini. La même procédure de chargement, décalage et d'additions est appliqué $(P/2)$ $[(P+1)/2]$ échantillons impairs et conduit à $round(P/4)$ $[round((P+1)/4)]$ bits qui constituent l'adresse impaire de la LUT impaire.

La figure V.7 aide dans l'établissement des correspondances entre les échantillons initiaux , les adresses paires et les adresses impaires dans le cas du filtre passe-bas à 9 coefficients du banc de filtres CDF9/7. Toutes les opérations du registre de données sont menées sous le contrôle de plusieurs signaux y compris l'horloge principale (master clock).

[1] *dans le cas d'un filtre antisymétrique, l'addition est remplacée par une soustraction.*

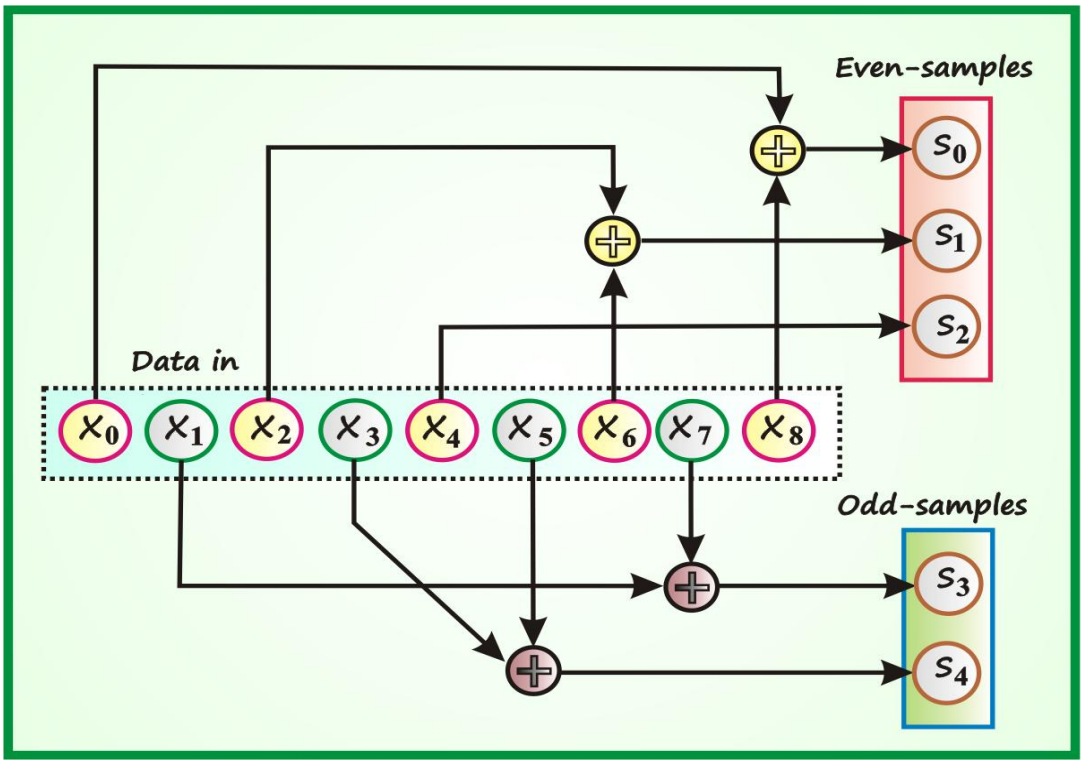


Figure V.7 Génération des adresses paires et impaires. (cas du filtre passe bas à 9 coefficients)

V.6.2.3 La ROM ou LUT :

Dans toute réalisation sur FPGA, une métrique importante de la conception est la surface utilisée conditionnée par la dimension de la LUT qui est à son tour imposée par la longueur du filtre. Les avantages de l'arithmétique distribuée, la structure polyphase et la symétrie du filtre contribuent à la réduction de la taille de la LUT utilisée. En effet, la LUT paire aurait $2^{(P/2+1)}$ $[2^{(P+1)/2}]$ entrées et la LUT impaire $2^{(p/2)}$ $[2^{(P+1)/2}]$ entrées. Cette stratégie réduit la LUT de $2^{(P/2+1)}$ $[2^{(P+1)/2}]$ à $2^{round\ ((P+2)/4)}[2^{round\ ((P+1)/4)}]$ adresses pour la section paire et de $2^{P/2}[2^{(P+1)/2}]$ à$2^{round\ (P/4)}[2^{round\ ((P+1)/4)}]$ pour l'impaire. Ainsi, on obtient une diminution (ou gain) par un facteur de 4. Plus la ROM est petite, plus faible est la consommation d'énergie.

Les contenus des ROM paire et impaire sont présentés dans la table V.1. L'utilisation des valeurs réelles des coefficients du filtre conduit à une implémentation complexe qui consomme lourdement les ressources matérielles, de l'énergie et produit un circuit qui ne peut répondre au critère de la rapidité exigé dans une majorité d'applications. A cet effet, les coefficients du filtre sont échelonnés (augmentés) par une multiplication par $2^{R,}$ R étant la résolution des échantillons à traiter, et stockés sous forme d'entiers représentés en complément à 2 (fixed point). Dans le tableau V.1, k est le pointeur

d'adresses de la LUT et est composé de 3 bits et 2 bits pour la LUT paire et la LUT impaire respectivement. Les 3 bits sont extraits de S_0, S_1 et S_2 et ont les poids 2^0, 2^1 …2^R. Le même principe est appliqué aux 2 bits composant l'adresse impaire qui sont dérivés de S_3 et S_4 (comme représenté dans la figure V.7).

LUT paire	
k	LUT(k)
0	0
1	h0
2	h2
3	h2 + h0
4	h4
5	h4 + h0
6	h4 + h2
7	h0 + h2 + h4

LUT impaire	
k	LUT(k)
0	0
1	h1
2	h3
3	h1+h3

Tableau V.1 La LUT paire et la LUT impaire (cas du filtre passé bas à 9 coefficients).

Les sorties des deux sections de LUT sont additionnées pour subir un dernier traitement, décrit dans la prochaine section. Les valeurs des coefficients des filtres à implémenter (CDF9/7) sont listées, en virgule flottante, dans le tableau V.2.

Des précautions doivent être prises afin d'éviter tout risqué de dépassement (overflow) qui pourrait se produire dans l'additionneur. C'est dans ce but que les coefficients sont représentés en complément à 2 en utilisant au moins un extra bit indispensable pour représenter la plus grande somme de coefficients.

Filtre passe bas	coefficients	Filtre passe haut	coefficients
h_0,h_8	0.026748757411	g_0,g_6	0.091271763114
h_1,h_7	-0.016864118443	g_1,g_5	-0.057543526229
h_2,h_6	-0.078223266529	g_2,g_4	-0.591271763114
h_3,h_5	0.2668641184430	g_3	1.11508705
h_4	0.6029490182360		

Tableau V.2 Coefficients des filtres du banc CDF9/7 (filtres d'analyse).

V.6.2.4 Le contrôleur :

Le contrôleur se compose d'un accumulateur de mise à l'échelle et d'un contrôleur de temps. L'accumulateur implémente l'équation de récurrence (67). Ce composant est aussi responsable de la génération du signal d'échantillonnage "*load*" qui contrôle le séparateur et de l'assertion du signal qui active le dernier circuit du filtre pour fournir le

résultat final. A la fin de l'opération, le résultat est décalé à droite R fois pour compenser l'échelonnage et obtenir la donnée finale correctement.

La figure V.8 présente l'architecture de l'ensemble des composants décrivant le schéma global de l'implémentation du filtre RIF passe-bas à base d'arithmétique distribuée et une résolution de 8 bits. Le chronogramme consiste en deux cycles réservés pour le chargement des données à filtrer et neuf cycles pour accomplir la tache de multiplication suivis par un cycle pour fournir le résultat sur le front descendant de l'horloge.

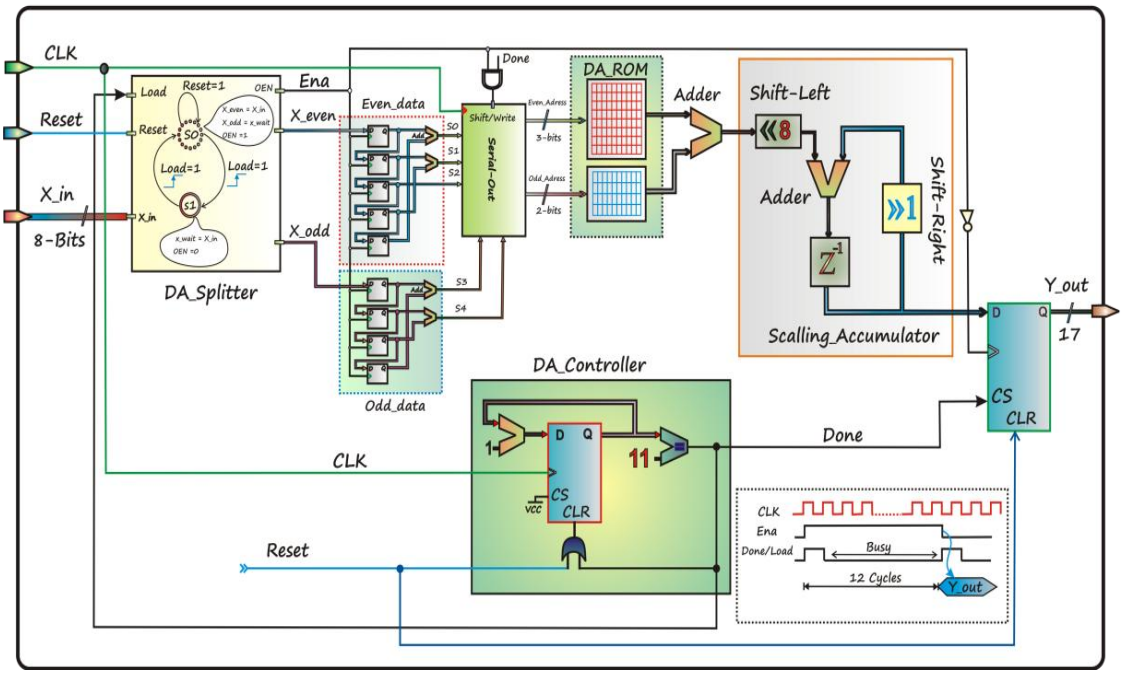


Figure V.8 Schéma synoptique de l'implémentation du filtre passé bas à 9 coefficients et une résolution de 8 bits.

V.6.3 Méthodologie à base d'arithmétique distribuée modifiée (MDA):

Cette architecture est aussi basée sur quatre composants, avec un séparateur identique à celui décrit précédemment dans la section V.6.2.1.

Le bloc appelé registre de données MDA (*MDA data register)* se charge des taches de chargement des données à traiter dans des registres, de l'addition des échantillons correspondants aux coefficients symétriques/antisymétriques et du décalage de la même manière que le registre de données CDA décrit dans la section V.6.2.2, mais la génération des adresses est différente. Le groupe constitué de $round\,((P+2)/4)$ $[round((P+1)/4)]$ données paires parallèles et le groupe des $round(P/4)$ $[round((P+1)/4)]$ données impaires sont séparés en ensembles de deux

registres. Un bus d'adresses, de 4 bits est généré de chaque ensemble pour adresser une LUT à 16 entrées (voir figure V.9).

Il faut noter que les ensembles de registres ont une taille dépassant la résolution des données à traiter de un bit. Ainsi, une extension par zéro est effectuée pour former les deux derniers bits. Si le nombre de registres est impair, le dernier génère les 2 bits qui seront utilisés comme adresse à une LUT à 4 entrées.

Ce processus peut être effectué pour des longueurs d'adresses différentes et a un impact direct sur la métrique des performances de l'implémentation. Des adresses plus grandes conduisent à des tailles plus importantes des LUTs, pouvant devenir irréalisables avec une consommation d'énergie plus importante mais le débit augmentera aussi. D'un autre coté, travailler avec des ensembles de 4 bits augmentera le nombre de LUTs et d'additionneurs. Le résultat est un chemin critique plus long conduisant à un débit plus faible.

Pour le composant nommé LUT MDA (*MDA LUT)*, les contenus des différentes LUTs sont des combinaisons linéaires des coefficients du filtre à implémenter. La figure V.9 montre une partie de la LUT paire dans le cas du filtre passé bas à 9 coefficients. La LUT impaire est obtenue simplement en remplaçant h_0 par h_1 et h_2 par h_3 ou h_5.

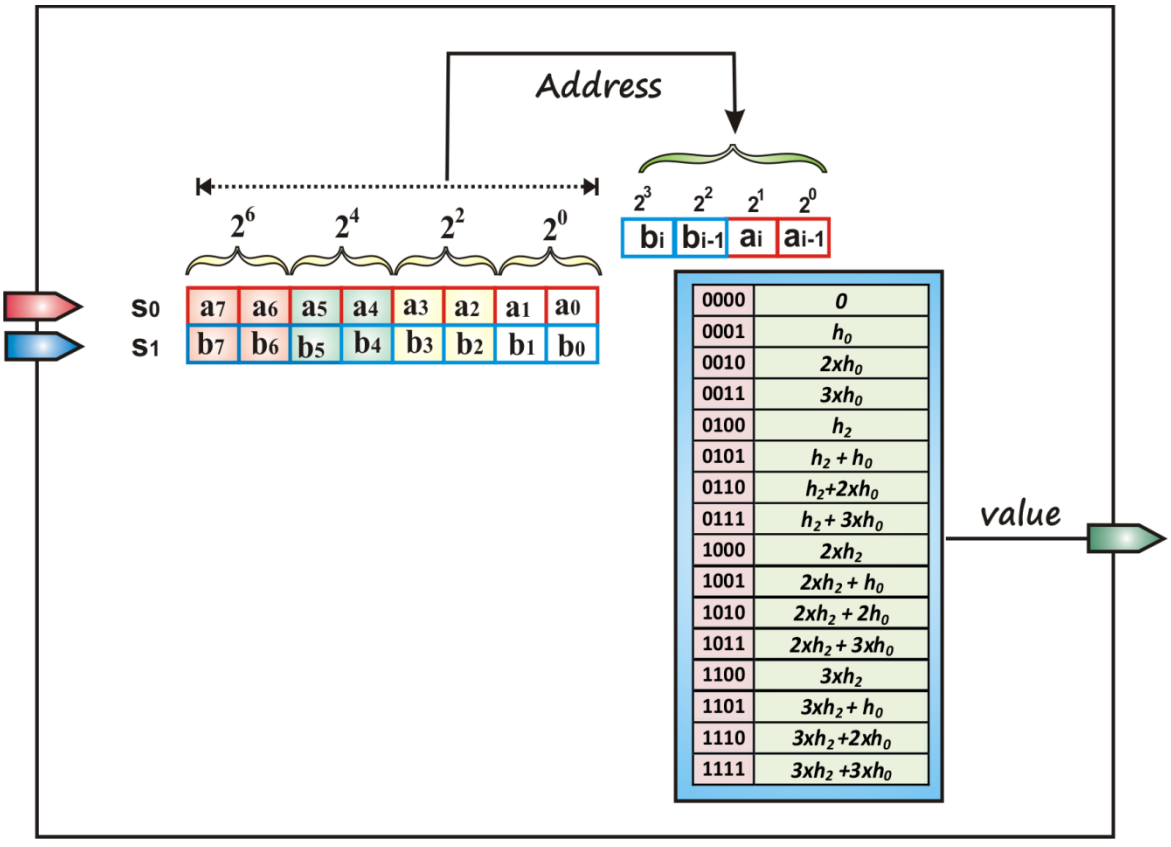


Figure V.9 La partie MDA LUT paire.

Finalement, le contrôleur MDA fonctionne d'une façon similaire à celui décrit dans la section V.6.2.4, à la différence que l'équation de récurrence (68) s'écrit maintenant:

$$y(k) = 2^{-2}y(k-1) + 2^{M'}LUT(k) \quad (69)$$

$M' = (R/2 + 2)$ R= résolution des données

La figure V.10 est un diagramme bloc de l'architecture à base de MDA.

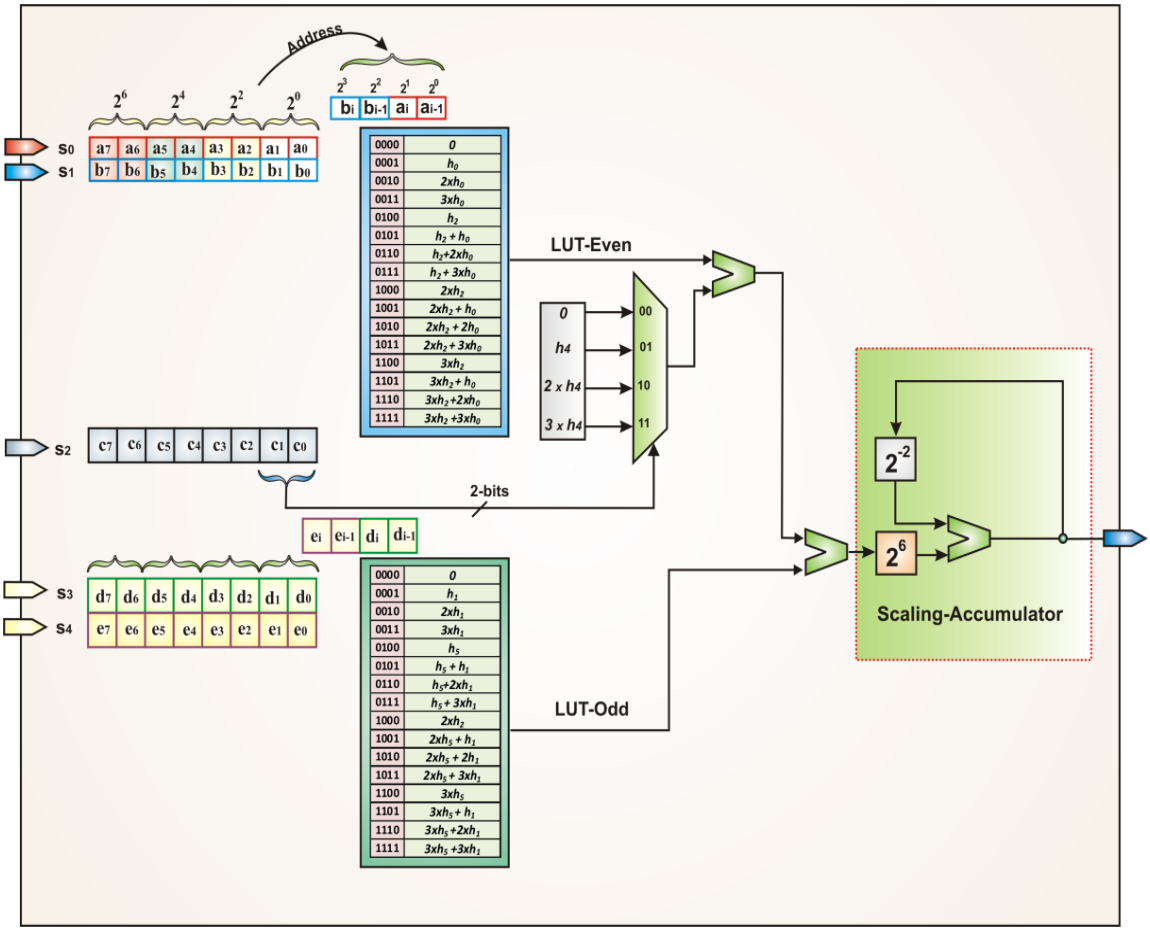


Figure V.10 Diagramme bloc de l'architecture à base de MDA.

V.6.4 Résultats et discussion :

V.6.4.1 Résultats des simulations:

Pour vérifier la conception à base de CDA, le banc de filtres CDF9/7 a été implémenté sur le circuit FPGA " Cyclone II EP2C35F672C6 d'Altera" et une résolution de 12 bits. Les descriptions de haut niveau des implémentations ont été écrites en langage VHDL et les simulations exécutées avec les logiciels Modelsim et Quartus II.

D'abord, un vecteur test composé de différentes valeurs de l'intervalle [0, 4095] (résolution de 12 bits) a été appliqué aux deux types de filtres. Les données sont :

{23, 345, 876, 1567, 34, 562, 3, 12, 34, 23} et la sortie filtrée avec la partie passe haut à 7 coefficients est montrée dans la figure V.11.

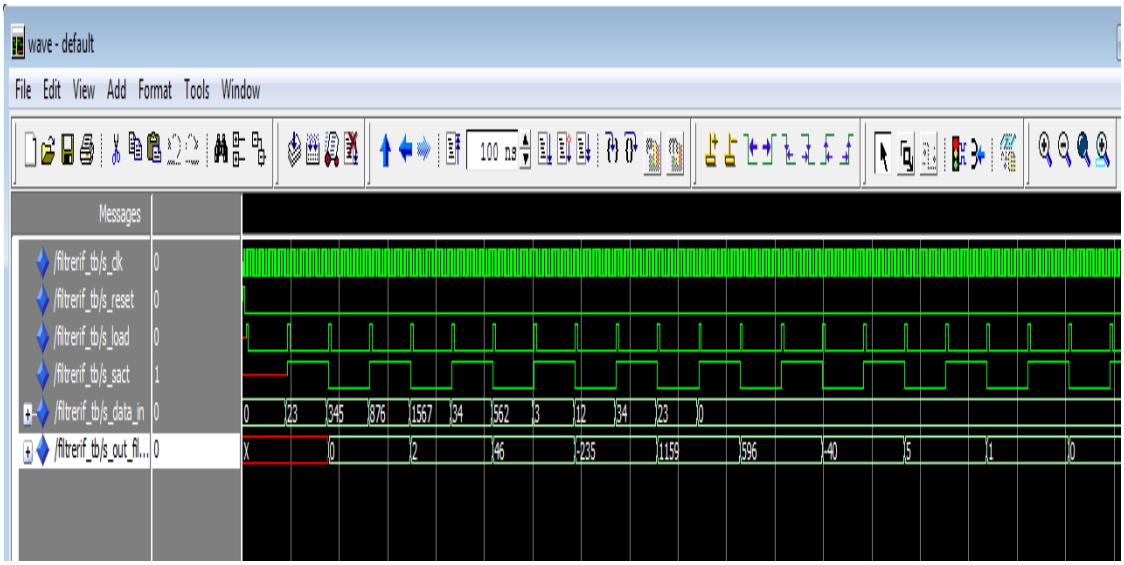


Figure V.11 Données filtrées par le filtre passe haut à 7 coefficients et 12 bits de résolution.

La table V.3 illustre la similarité entre les résultats obtenus en utilisant Matlab et ceux obtenus avec l'outil Modelsim. Les résultats de Matlab corroborent les résultats obtenus en adoptant la conception proposée et prouvent que les filtres implémentés fonctionnent correctement. L'erreur absolue entre les sorties filtrées données par Matlab et Modelsim provient des opérations de troncation puisque les opérations de division sont menées par des opérations de décalage à droite.

Dans le tableau V.3, les données à filtrer, listées précédemment, sont présentées dans la première colonne. La deuxième colonne donne les résultats du filtrage des ces données obtenus avec Matlab. La troisième colonne montre les résultats obtenus avec Modelsim dans le cas du filtre passe haut à 7 coefficients. Enfin, la quatrième et cinquième colonne est similaire à la deuxième et troisième colonne mais dans le cas du filtre passe bas à 9 coefficients.

Un deuxième test a été mené sur un signal périodique multi-tons entaché de bruit blanc gaussien. Il a été filtré avec le banc CDF9/7 et les signaux obtenus sont exposés dans les figures V.12 et V.13 respectivement.

Données à filtrer	Résultat (FPH^2) Matlab	Résultat (FPH^2) Modelsim	Résultat (FPB^2) Matlab	Résultat (FPB^2) Modelsim
23	2	2	1	0
345	47	46	16	15
876	-234	-235	12	12
1567	1159	1159	1025	1024
34	596	596	515	515
562	-39	-40	146	146
3	5	5	21	21
12	2	1	3	3
34	0	0	1	0
23	0	0	0	0

Tableau V.3 Comparaison des résultats obtenus avec Matlab et ceux obtenus avec Modelsim.(^{2}FPH=Filtre Passe-Haut, FPB= Filtre Passe-Bas)

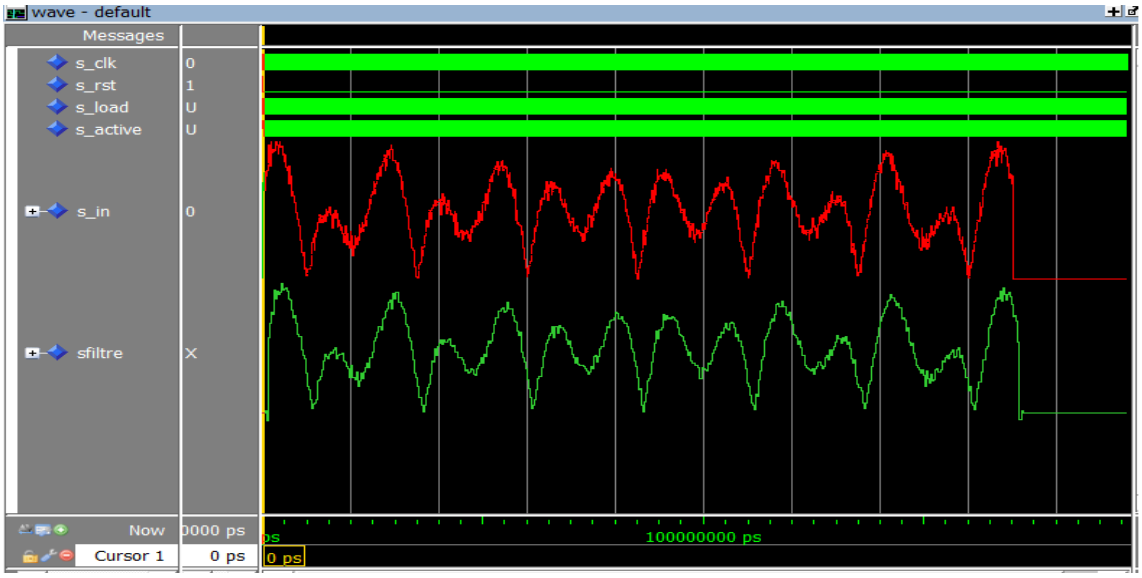


Figure V.12 Résultat de simulation avec Modelsim. Signal bruité (en haut) , signal filtré ou approximation (en bas).

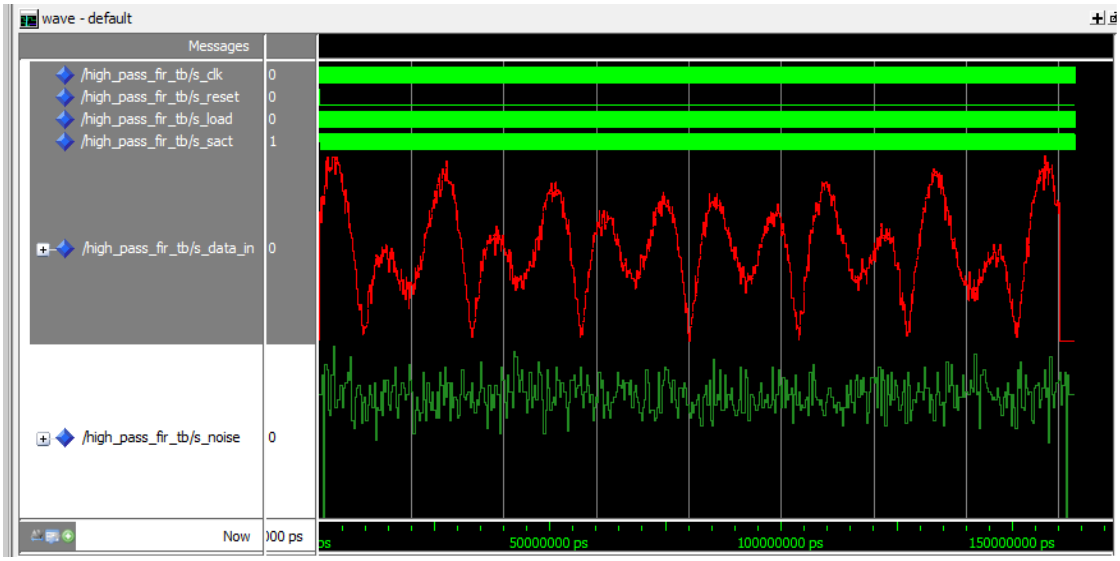


Figure V.13 Résultat de simulation avec Modelsim. Signal bruité (en haut), bruit ou détails (en bas).

En termes de surface, le rapport de synthèse généré par Quartus II et présenté dans le tableau V.4, prouve l'efficacité de la méthode appliquée. En effet, moins de 1% des éléments logiques disponibles dans le circuit ciblé ont été utilisés, conduisant à un circuit de très petite dimension. Le rapport a aussi montré qu'aucun multiplieur intégré du circuit cible n'a été utilisé. Il est aussi important d'insister sur le fait que la technique d'arithmétique distribuée permet de découpler la latence de la longueur du filtre. Mais ce rapport dévoile l'inconvénient majeur de cet algorithme qui est la latence, conséquence de l'architecture choisie du circuit faisant tâche de multiplieur.

Nom de l'entité de haut niveau	RIF passe-bas	RIF passe-haut
Famille	Cyclone II	Cyclone II
Circuit	EP2C35F672C6	EP2C35F672C6
Nombre d'éléments logiques	295/33216 (<1%)	252/ 33216 (<1%)
Fmax (MHz)	160.18	190.04
Latence (cycles)	16	16
Puissance dissipée (mW)	147.24	143.15

Tableau V.4 Rapport de synthèse de l'implémentation des filtres CDF9/7 avec une résolution de 12 bits.

Pour approfondir cette analyse, une comparaison des métriques des performances comportementales de quatre multiplieurs (8x8 bits) a été conduite : un multiplieur à

base de CDA, un multiplieur à base de MDA, un multiplieur parallèle et un multiplieur M4K. M4K est un bloc synchrone implémentant une mémoire à double port avec entrées enregistrées et sorties optionnellement enregistrées, disponible dans le circuit FPGA ciblé. Il est utilisé pour des multiplications parallèles entre nombre binaires allant jusqu'à 18 bits.

En se basant sur les performances listées dans le tableau V.5, on peut voir facilement que la méthode CDA exhibe beaucoup d'avantages pour attirer l'attention contre la structure parallèle en termes de surface, et que la latence est sa faiblesse majeure. Cependant, le multiplieur DA reste un substitut essential dans le traitement de données en série en provenance d'un convertisseur série et où le décimateur est d'une importance vitale. Evidemment, le multiplier MDA est un concurrent de choix au multiplieur parallèle et améliore le multiplieur CDA en termes de débit au prix d'une surface et d'une consommation légèrement plus grande. Le multiplieur M4K est notablement en tête de l'ensemble même s'il utilise un nombre relativement important de bits mémoire.

Famille : Cyclone II Circuit : EP2C35F672C6				
Performances	**multiplieur CDA**	**multiplieur MDA**	**multiplieur Parallèle**	**multiplieur M4K**
Nombre d'éléments logiques	**98 / 33,216 (< 1%)**	**112 / 33,216 (<1%)**	**154 /33,216 (< 1 %)**	**68 /33,216 (< 1 %)**
Nombre de fonctions combinatoires	**91 / 33,216 (<1%)**	**106 / 33,216 (< 1 %)**	**153 /33,216 (< 1 %)**	**68 / 33,216 (< 1 %)**
Nombre de registres logiques	**79 / 33,216 (<1%)**	**77 / 33,216 (< 1 %)**	**57 / 33,216 (< 1 %)**	**0 / 33,216 (< 1%)**
Nombre de broches	**59 / 475 (12%)**	**59 / 475 (12 %)**	**59 / 475 (12 %)**	**59 / 475 (12 %)**
Nombre de bits mémoires	**0**	**0**	**0**	**21,760 / 483,840(4%)**
F_{MAX} (MHz)	**217.86**	**182.32**	**110.94**	**260.01**
Latence (cycles)	**8**	**4**	**2**	**3**
Puissance dissipée (mW)	**139.12**	**148.42**	**137.66**	**139.29**

Tableau V. 5. Evaluation des performances de différents multiplieurs (8x8 Bits).

Pour vérifier l'incidence du multiplieur MDA sur les performances des filtres, une architecture à base de CDA et une architecture à base de MDA ont été conçues pour implémenter le filtre passe-bas à 9 coefficients. Le tableau V.6 permet une analyse coût-performance et dévoile l'efficacité de l'architecture MDA qui conduit à une latence très réduite et un débit plus élevé mais définitivement plus d'énergie dissipée en

entrées/sorties. Les résultats montrent que cette nouvelle approche à base de MDA présente une accélération d'un facteur de 1.78 sur la méthode à base de CDA, une augmentation de la dissipation d'énergie des entrées/sorties d'un facteur de 1.07 et un élargissement de la surface occupée par un facteur de 1.13. En conclusion, le schéma MDA peut certainement paver le chemin pour produire des circuits de traitement du signal rapides dans une gamme très large d'applications telles que la compression vidéo.

	architecture à base de CDA	architecture à base de MDA
Surface	199 EL [3]	237 EL [3]
F_{max} (MHz)	180.47	160.51
Puissance dissipée (mW)	144.78	155.55
Latence (cycles)	12	6
Débit(MSPS[4])	15.04	26.75
Débit(MBPS[5])	120.32	214

Tableau V.6 Comparaison des performances des architectures CDA et MDA.

([3]EL= Elément Logique, [4]MSPS= Mega Samples Per Second, [5]MBPS= Mega Bits Per Second)

Enfin, les tableaux V.7 et V.8 présentent des comparaisons des performances des architectures proposes avec des résultats publiés dans [101] et [102] respectivement. Dans le cas de la table V.7, la comparaison est faite sur la base d'un coefficient.

	[101]	RIF CDA	RIF MDA
Famille/Circuit	Xilinx XCV2000E	Cyclone II /EP2C35F672C6	Cyclone II / EP2C35F672C6
Filtre	9 coefficients	9 coefficients	9 coefficients
Surface (EL)	133	199	237
Taille des données (bits)	8	8	8
Fmax (MHz)	74.025	180.47	160.51
Puissance (mW)	59.47	48.84	59.90

Tableau V.7 Comparaison des performances de différentes implémentations (filtre passe bas, résolution de 8 bits).

	[102]	RIF-CDA	RIF-MDA
Famille/Circuit	Non mentionnés	Cyclone II/EP2C35F672C6	Cyclone II / EP2C35F672C6
Filtre	31 coefficients	9 **coefficients**	9 **coefficients**
Surface (EL)	35.81	32.78	36.55
Taille des données (bits)	12	12	12
F_{max} (MHz)	7.42	17.80	16.76

Tableau V.8 Comparaison des performances de différentes implémentations (filtre passe bas, résolution de 12bits).

Les tableaux ci-dessus montrent que les méthodes proposées conduisent à des implémentations plus rapides et relativement meilleures que celles publiées dans [101] et [102]. Ceci est d'une importance vitale dans les traitements en temps réel tels que la

compression vidéo. La puissance dissipée dans le cas de l'algorithme MDA est pratiquement égale à celle donnée dans [101].

V.6.4.2 Implémentation hardware :

Le filtre passe-bas à 9 coefficients et une résolution de 8 bits a été implémenté sur un circuit FPGA cyclone II (EP2C35F672C6) d'Altera. Le logiciel Quartus II a été utilisé pour télécharger le circuit synthétisé dans le circuit FPGA à travers le port" USB Blaster" et en mode "JTAG". 28 broches ont été utilisées et assignées manuellement pour des raisons pratiques.

En premier, la réponse impulsionnelle du filtre a été testée et a prouvé que le filtre fonctionne parfaitement. La latence réelle est 13 cycles d'horloge, qui dépasse donc la latence envisagée et confirmée par les simulations d'un cycle. Mais, le retard de groupe mesuré est exactement de 4 échantillons et confirme le résultat théorique. Le tableau V.9 récapitule les résultats.

Vecteur test	**255**	**0**	**0**	**0**	**0**	**0 …**	**0**
Résultats des simulations (Modelsim)	9	-28	217	-28	9	0 …	0
Résultats des tests de l'implémentation Hardware	9	-28	217	-28	9	0 …	0
Résultats des simulations (Matlab)	10	-28	217	-28	10	0 …	0

Tableau V.9 Comparaison des résultats de l'implémentation hardware avec ceux des simulations.

Pour mener un second test, le même signal bruité (voir section V.6.4.1) a été stocké dans une RAM et a alimenté le filtre en boucle continue. La sortie du filtre est alors lue par le codec audio "Wolfson WM8731" [109] disponible sur la carte DE2 d'Altera et visualisée sur un oscilloscope connecté à travers le port "line out" du codec audio. Une boucle à verrouillage de phase ou PLL (Phase Locked Loop) a été rajoutée et utilisée pour générer la fréquence d'horloge convenable du codec audio (48.43 KHz) et aux autres circuits à partir de l'horloge principale de la carte DE2 (50 MHz). Les différents signaux de contrôle nécessaires pour la communication du circuit FPGA avec le circuit Wolfson WM8731 via le protocole Inter-Integrated Circuit "I2C" sont produits par le bloc d'interface du codec. La figure V.14 présente un schéma synoptique simplifié de l'environnement utilisé pour tester le filtre. La similarité entre les signaux relevés et ceux des figures V.12et V.13 est remarquable.

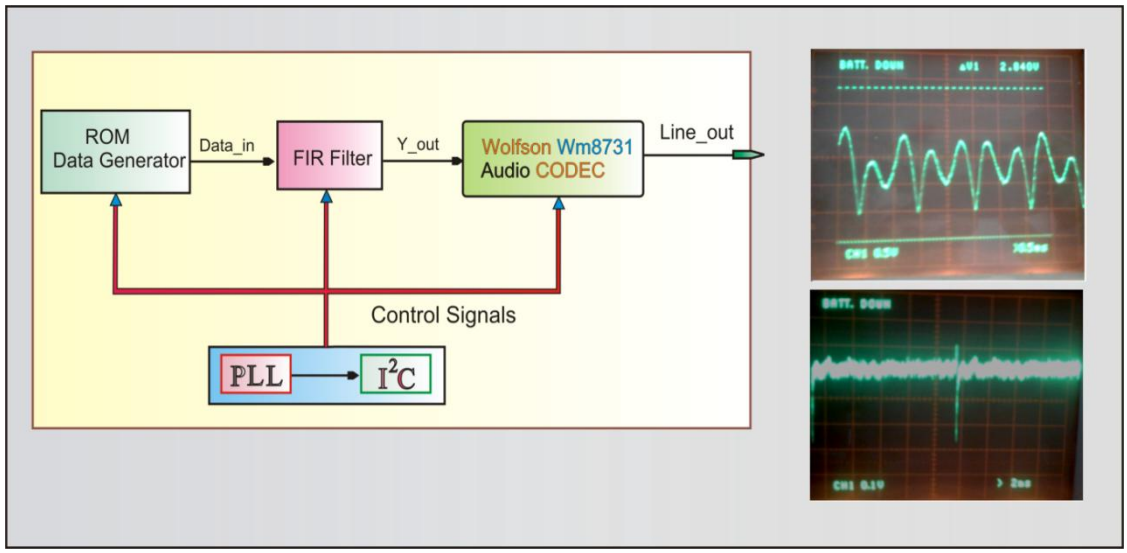


Figure V.14 Schéma synoptique de la configuration de test de l'implémentation hardware.

V.7 Implémentation de la DCT :

V.7.1 Introduction :

La qualité des images et des vidéos sur des outils tels que les téléphones mobiles et les caméras, dépend, en grande partie des codecs (codeurs/décodeurs) qui y sont implémentés pour le codage, le traitement, la compression et la restitution. Ces codeurs doivent répondre aux exigences, de plus en plus croissantes, de rapidité (vitesse de traitement) et de consommation d'énergie. L'ensemble des standards de renommé tel que JPEG pour l'image, MPEG-1, 2, 4, H263 et H.264 pour la vidéo, utilisent la transformée en cosinus discrète et son inverse (DCT et IDCT) comme base de leurs opérations. Ces fonctions sont les plus complexes du point de vue calculatoire et sont la source des limites des performances des algorithmes de compression des images et des vidéos [110]. La complexité calculatoire de la DCT a été estimée à 21% du traitement total [111],[112]. Ceci conduit donc à une lenteur et une grande dissipation d'énergie, alors qu'un codec rapide et efficace énergétiquement est exigé. Les tests menés au chapitre IV ont montré que la DCT entière 4x4 est un concurrent à la DCT entière 8x8 et que le seul handicap reste le temps de traitement. Son implémentation sur un circuit programmable de type FPGA est une solution qui répond aux exigences du traitement en temps réel.

V.7.2 Implémentation de la DCT entière 4x4 :

V.7.2.1 Méthodologie :

La transformée en DCT entière 4x4, présentée au chapitre III, est l'une des caractéristiques introduites dans la norme H.264/AVC. Son noyau est défini par :

$$Y = (C_f X C_f^T) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} [X] \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \quad \text{(voir eq.22)}$$

Y représentant les coefficients de la DCT et X étant les données résiduelles obtenues après prédiction. Les valeurs de ces données varient entre -255 et +255.

Le développement du produit matriciel précédent donne :

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}}_{C_f X} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \quad (70)$$

$$\begin{bmatrix} x_{11}+x_{21}+x_{31}+x_{41} & x_{12}+x_{22}+x_{32}+x_{42} & x_{13}+x_{23}+x_{33}+x_{43} & x_{14}+x_{24}+x_{34}+x_{44} \\ 2x_{11}+x_{21}-x_{31}-2x_{41} & 2x_{12}+x_{22}-x_{32}-2x_{42} & 2x_{13}+x_{23}-x_{33}-2x_{43} & 2x_{14}+x_{24}-x_{34}-2x_{44} \\ x_{11}-x_{21}-x_{31}+x_{41} & x_{12}-x_{22}-x_{32}+x_{42} & x_{13}-x_{23}-x_{33}+x_{43} & x_{14}-x_{24}-x_{34}+x_{44} \\ x_{11}-2x_{21}+2x_{31}-x_{41} & x_{12}-2x_{22}+2x_{32}-x_{42} & x_{13}-2x_{23}+2x_{33}-x_{43} & x_{14}-2x_{24}+2x_{34}-x_{44} \end{bmatrix} \quad (71)$$

Ceci montre que chaque colonne de la matrice résultant du produit $C_f X$ est obtenue en effectuant le même traitement sur les échantillons de chaque colonne du bloc de données 4x4. Ce traitement nécessite des additions/soustractions et des décalages à gauche d'un pas pour réaliser les multiplications par 2. Le diagramme suivant résume les calculs, i étant le numéro de la colonne considérée ($1 \leq i \leq 4$) et S_{1i}, S_{2i}, S_{3i}, S_{4i} ses éléments :

$$S_i = \begin{bmatrix} x_{1i}+x_{2i}+x_{3i}+x_{4i} \\ 2x_{1i}+x_{2i}-x_{3i}-2x_{4i} \\ x_{1i}-x_{2i}-x_{3i}+x_{4i} \\ x_{1i}-2x_{2i}+2x_{3i}-x_{4i} \end{bmatrix} \quad (72)$$

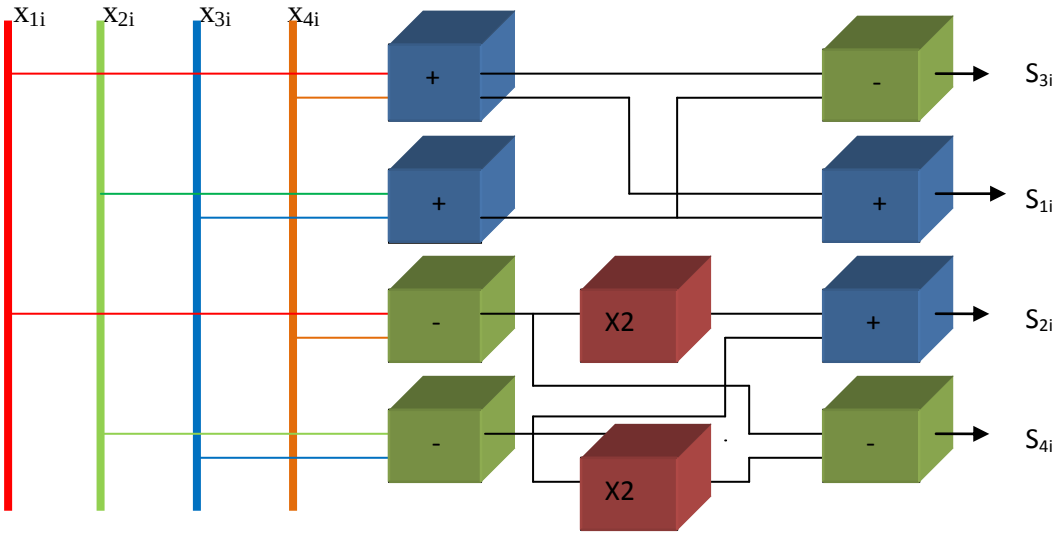


Figure V.15 Diagramme de calcul de la DCT entière 1D 4x4

La deuxième étape de la transformation consiste à effectuer le produit matriciel suivant :

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix} = \underbrace{\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}}_{[Y]} \quad (73)$$

Ce calcul aboutira au même résultat précédent mais transposé. On retrouve que les opérations effectuées sur les données notées S_{ij} sont les mêmes que celles menées sur les x_{ij} à la seule différence d'une transposition :

$$[Y]=\begin{bmatrix} S_{11}+S_{12}+S_{13}+S_{14} & 2S_{11}+S_{12}-S_{13}-2S_{14} & S_{11}-S_{12}-S_{13}+S_{14} & S_{11}-2S_{12}+2S_{13}-S_{14} \\ S_{21}+S_{22}+S_{23}+S_{24} & 2S_{21}+S_{22}-S_{23}-2S_{24} & S_{21}-S_{22}-S_{23}+S_{24} & S_{21}-2S_{22}+2S_{23}-S_{24} \\ S_{31}+S_{32}+S_{33}+S_{34} & 2S_{31}+S_{32}-S_{33}-2S_{34} & S_{31}-S_{32}-S_{33}+S_{34} & S_{31}-2S_{32}+2S_{33}-S_{34} \\ S_{41}+S_{42}+S_{43}+S_{44} & 2S_{41}+S_{42}-S_{43}-2S_{44} & S_{41}-S_{42}-S_{43}+S_{44} & S_{41}-2S_{42}+2S_{43}-S_{44} \end{bmatrix} \quad (74)$$

On peut alors conclure que les traitements sur les deux dimensions (colonnes, lignes) sont identiques et que seules les tailles des données et des résultats changent. Le tableau suivant présente les différentes tailles des données manipulées :

Donnée	Intervalle	Taille en bits
x	[-255, +255]	9 (signé)
S	[-1023, +1023]	11 (signé)
Y	[-8191, +8191]	13 (signé)

Tableau V. 10 Taille des données manipulées

V.7.2.2 Implémentation :

Les résultats mathématiques montrent que les traitements se prêtent naturellement au parallélisme à la seule condition que les seize échantillons (un bloc 4x4) soient disponibles simultanément. Cette dernière condition peut être satisfaite de deux façons :

- Concevoir un composant qui convertirait seize données lues de la mémoire en série en seize données parallèles.
- Concevoir un composant qui lirait quatre données, soit une colonne d'un bloc de taille 4x4 et les délivrerait en parallèle. Dans ce cas, il en faudrait quatre circuits de ce type ainsi qu'un stockage particulier des données en mémoire. En effet, les données résiduelles doivent être arrangées dans quatre mémoires identiques séparées. Le bloc mémoire "i" contiendrait les colonnes "i" de l'ensemble des blocs 4x4 du frame.

Cette deuxième solution permet un gain en temps de traitement par rapport à la première mais au prix de plus de ressources matérielles. En effet, la lecture et la conversion des données en parallèle nécessiterait, dans le premier cas, 16 cycles d'horloge et un seul circuit de conversion série parallèle. Dans le deuxième cas, seulement 4 cycles sont consommés mais quatre circuits de conversion doivent être utilisés.

Les figures V.16.a et V.16.b représentent les schémas synoptiques de la DCT entière 4x4 selon les deux possibilités exposées ci-dessus.

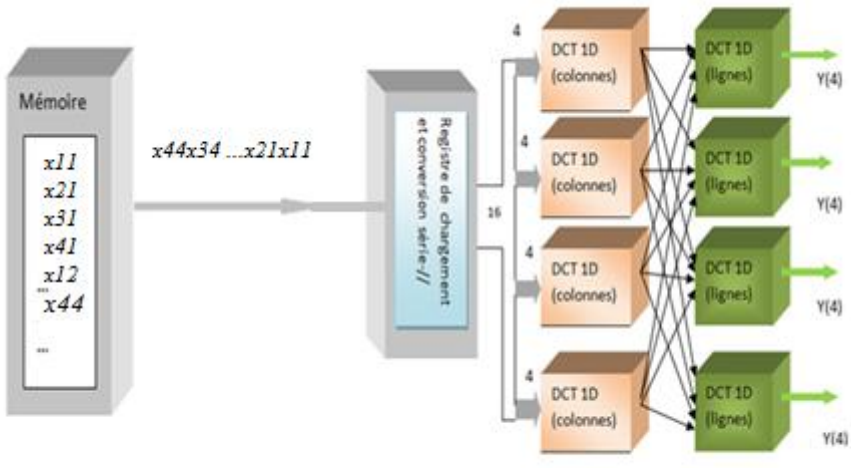


Figure V.16.a Schéma synoptique de la DCT entière 4x4 (version 1)

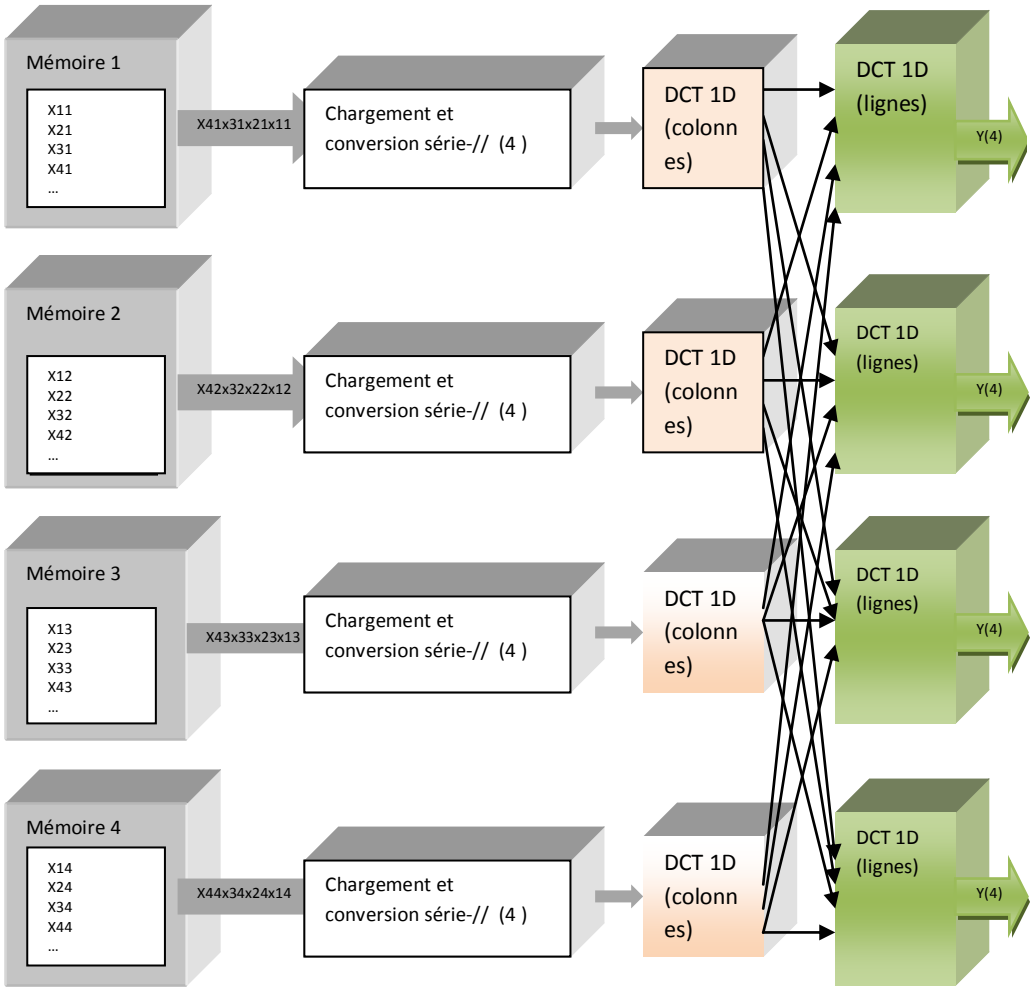


Figure V.16.b Schéma synoptique de la DCT entière 4x4 (version 2)

Les mémoires utilisées sont de type "RAM" et afin de garantir une parfaite synchronisation, il est préférable de n'utiliser qu'une seule. Dans ce cas, les échantillons d'un bloc 4x4 doivent être copiés d'une manière directe (mapped) dans l'espace mémoire. Il devient alors facile d'accéder aux données en parallèle en ligne alors qu'il est impossible d'y accéder en colonne. La figure V.17 montre ce concept, clé de l'optimisation de l'implémentation hardware.

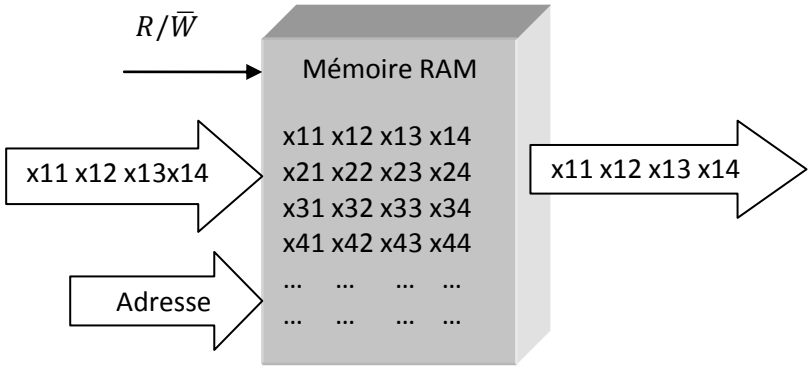


Figure V.17 Organisation de la RAM.

Chaque circuit, excepté la mémoire, fonctionne comme une machine de Mealy à deux états : libre ("idle") ou occupé ("busy") décrite par la figure V.18. La synchronisation des différents circuits est assurée par une technique de scrutation ou "polling" où chaque composant active son successeur dès qu'il a terminé sa part de traitement. Cependant, le résultat n'est envoyé que lorsqu'il reçoit un signal l'informant que le destinataire est prêt à prendre le relais.

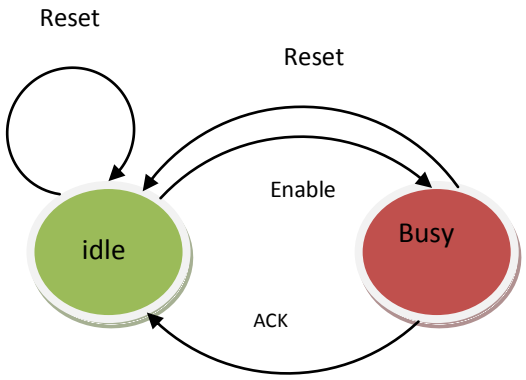


Figure V.18 Diagramme d'état du composant DCT 1D 4x4

1. Le composant de chargement et conversion :

Les figures V. 19.a et V. 19.b représentent les symboles des composants de chargement et conversion selon les deux versions : chargement de 16 données ou chargement de 4 données.

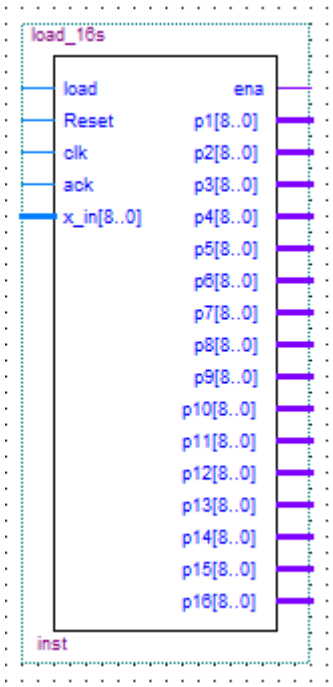


Figure V. 19.a Symbole du circuit de chargement et conversion de 16 données.

load : signal de chargement en entrée . Il détermine la fréquence d'échantillonnage des blocs 4x4.

ack : entrée, qui à l'état haut indique que le circuit successeur (DCT 1D (colonnes)) est prêt à recevoir les données p1,p2, …,P16.

ena : signal d'activation du circuit successeur.

x_in : entrée série des données résiduelles à traiter.

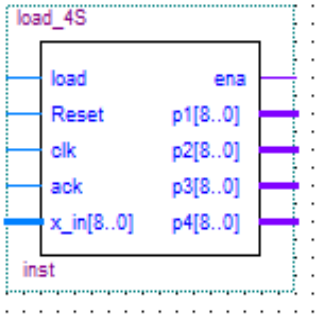


Figure V.19.b Symbole du circuit de chargement et conversion de 4 données.

Les figures V.20.a et V.20.b montrent les résultats des simulations sous ModelSim.

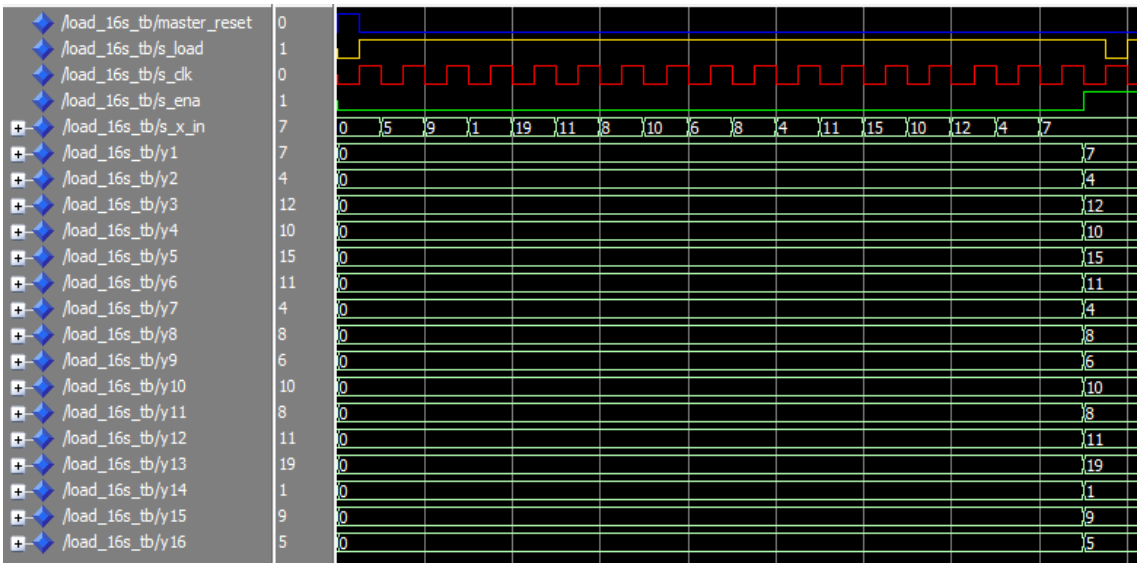


Figure V.20.a Simulation du circuit de chargement et conversion de 16 données.

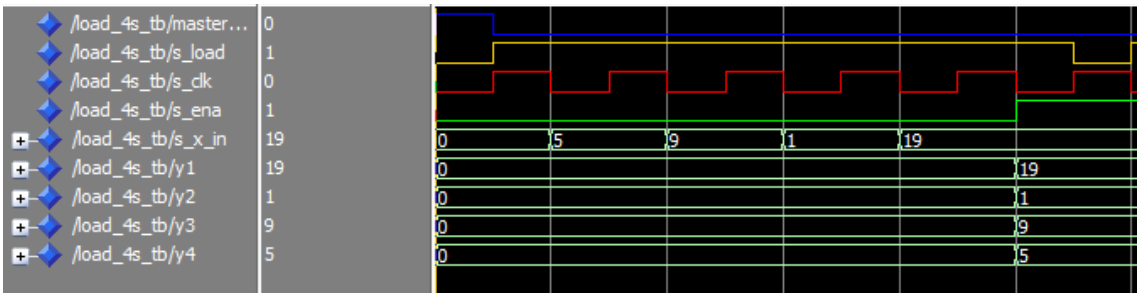
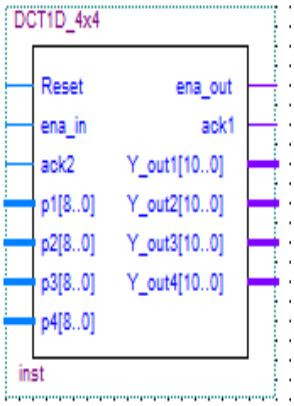


Figure V.20.b Simulation du circuit de chargement et conversion de 4 données.

2. Le composant DCT 1D 4x4 ou DCT 1D (colonnes) :

Ce composant effectue les calculs présentés dans la figure V.15, sur quatre échantillons, notés p1, p2, p3 et p4 pour produire les résultats Y_out1, Y_out2, Y_out3 et Y_out4. Le signal ena_in provient du registre de chargement et conversion. Le circuit utilise le signal ack1 pour informer le composant de chargement et de conversion qu'il est prêt à prendre en charge le traitement des données. Une fois ceci terminé et les sorties prêtes , le composant envoie le signal ena_out , pour activer le composant nommé DCT2D_4x4.

Ack2 est le signal envoyé par l'étage successeur pour informer qu'il est libre et prêt à recevoir des données pour traitement. La figure V.21.a montre le symbole de ce circuit alors que la figure V.21.b présente le résultat de sa simulation.



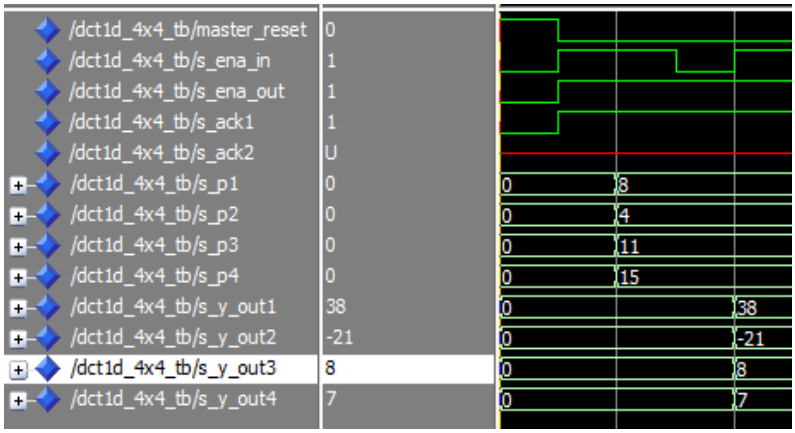
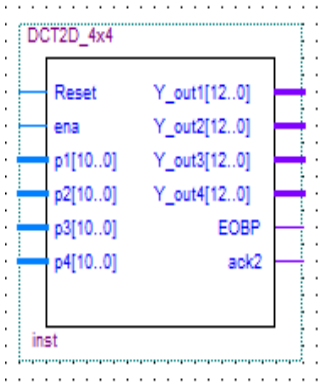


Figure V.21.a Symbole du composant DCT1D_4x4

Figure V.21.b Résultat de la simulation du circuit DCT1D_4x4

3. Le composant DCT 2D 4x4 ou DCT 1D (lignes) :

Le fonctionnement de ce circuit est très similaire à celui du circuit DCT 1D 4x4 à quelques exceptions. D'abord, les données en entrée, p1, p2, p3 et p4 proviennent de quatre circuits différents et par conséquent le signal ack2 doit parvenir à l'ensemble de ces composants. Ensuite, ce circuit est le dernier étage de la chaine et n'active donc aucun circuit. Cependant, le signal EOBP ("End Of Bloc Processing") doit être combiné avec trois autres signaux du même type à travers une porte "AND" pour alimenter le compteur de nombre de blocs à transformer. Les figures V.22.a et V.22.b représentent le symbole du circuit et le résultat de sa simulation sous ModelSim.



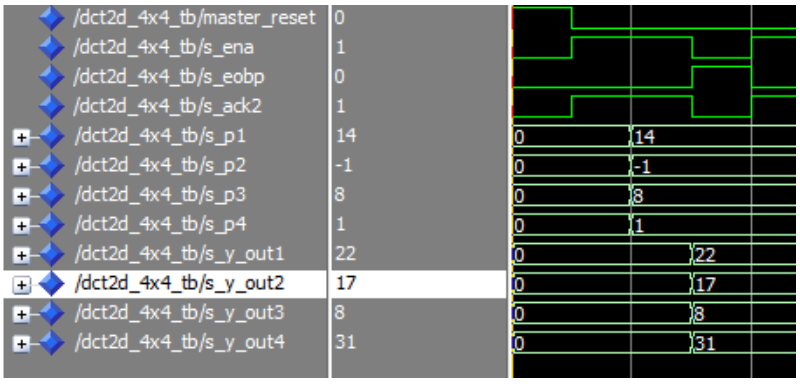


Figure V.22.a Symbole du composant DCT2D_4x4

Figure V.22.b Résultat de la simulation du circuit DCT2D_4x4

Les résultats de la simulation du circuit DCT 2D entière 4x4 sont présentés dans les figures V.23.a (résultat partiel) et V.23.b (résultat final).

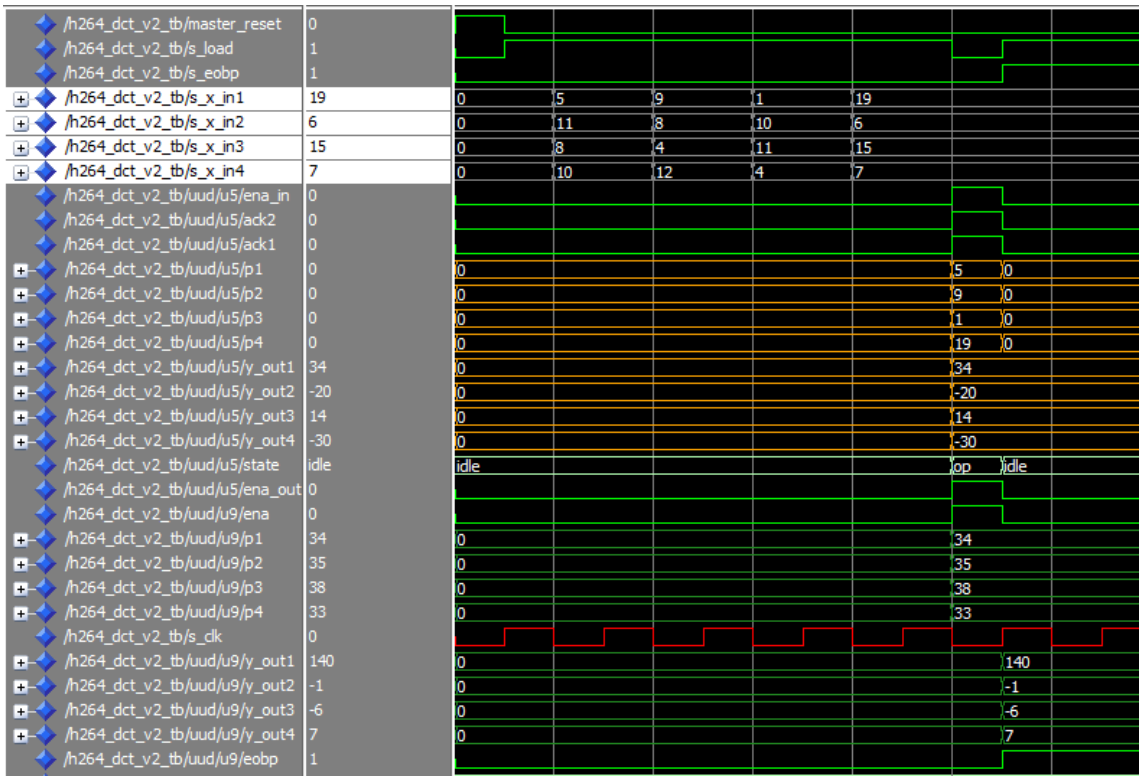


Figure V.23.a Résultat de la simulation de la DCT 2D entière 4x4 (pour 4 échantillons)

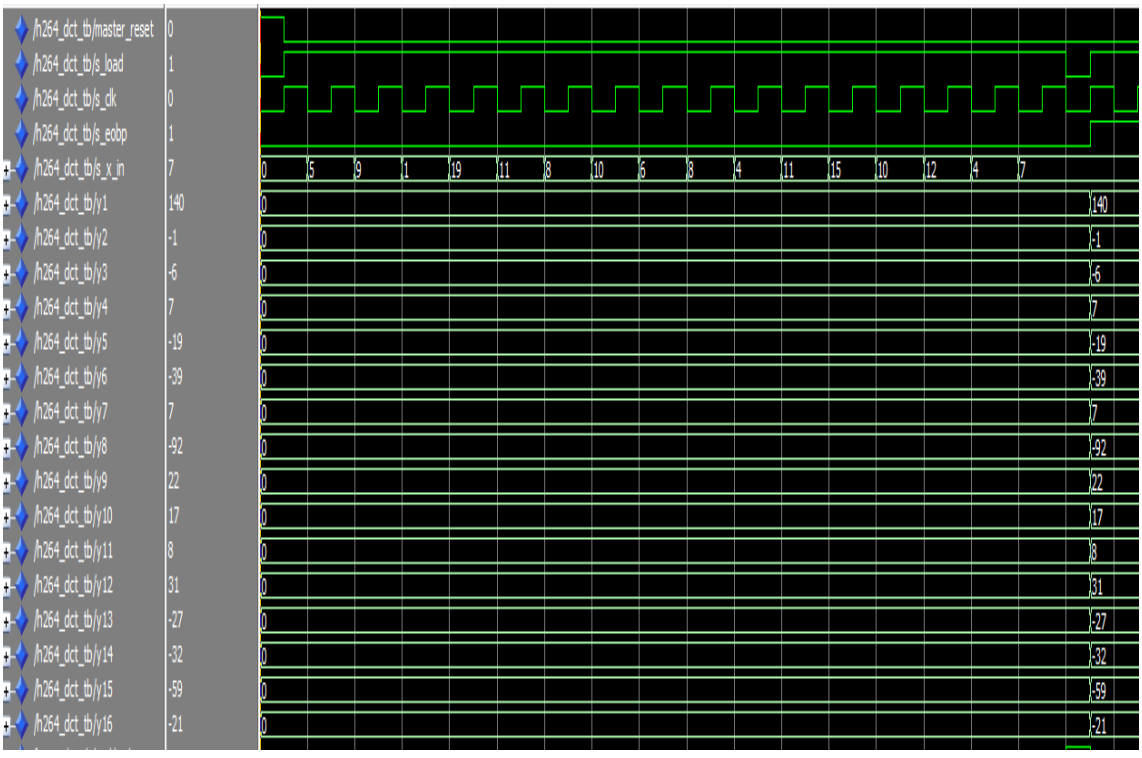


Figure V.23.b Résultat de la simulation de la DCT 2D entière 4x4 (un bloc 4x4)

Il est important de mentionner que cette implémentation combine les techniques de parallélisme et de "pipelining". En effet, la technique de traitement en pipeline permet une économie en cycles de traitement qui se fait en trois phases :

Phase 1 : lecture et conversion série parallèle de 16 ou de 4 données nécessitant 16 ou 4 cycles d'horloges respectivement.

Phase 2 : DCT 1D (colonnes) exigeant un demi cycle d'horloge.

Phase 3 : DCT 1D (lignes) demandant un demi cycle d'horloge.

Le diagramme de la figure V.24 montre les trois phases du traitement et l'économie en cycles d'horloges obtenus en appliquant la technique du pipelining.

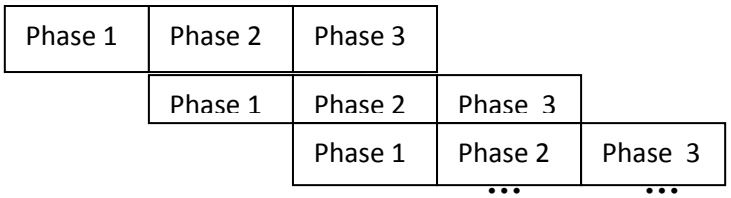


Figure V.24 Les différentes phases du traitement de la DCT 2D entière 4x4

Sans le pipelining, la transformation d'un bloc 4x4 nécessiterait 17 cycles d'horloges pour la version 1 et 5 cycles pour la version 2. Au fur et à mesure que le nombre de blocs à traiter croit, la différence, en cycles, entre les deux techniques croit. Pour n blocs et sans le pipelining, 17n/5n cycles seront consommés. Par contre, (16n +1)/(4n+1) cycles seulement seront utilisés avec le pipelining. Le gain est donc de (n-1) cycles, ce qui est très important si l'on se rappelle qu'un seul frame en format QCIF renferme 1584 blocs de taille 4x4. La différence est alors de 1583 cycles, valeur à multiplier par le nombre de frames de la séquence vidéo en question.

Enfin, les figures V.25 et V.26 donnent les vues RTL des deux versions de l'implémentation de la DCT 2D entière 4x4.

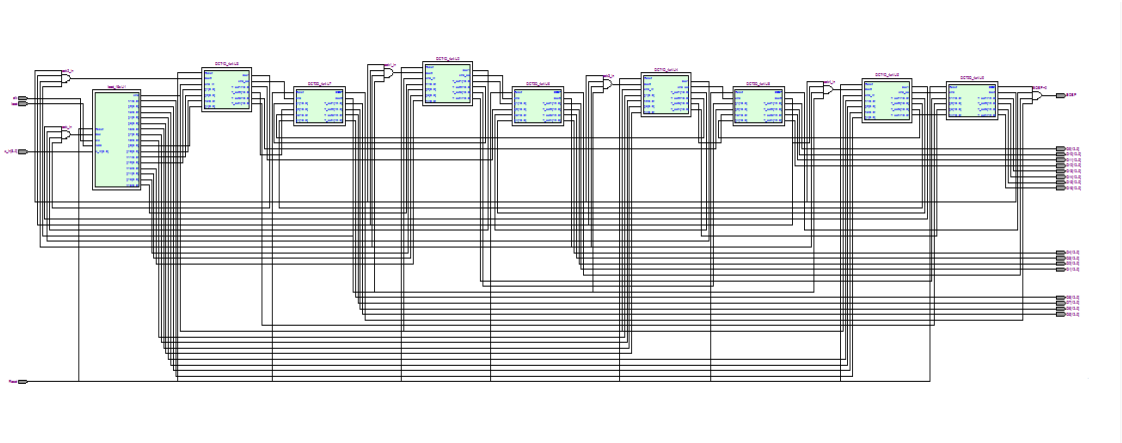


Figure V.25 Vue RTL de la DCT 2D entière 4x4 avec chargement de 16 données (version 1).

Le schéma structurel (fig. V.25) renferme un composant de chargement et de conversion de 16 données, quatre composants DCT 1D (colonnes) et quatre composants DCT 1D (lignes). Par contre, dans celui de la figure V.26, on dénombre quatre

composants de chargement et de conversion de 4 données, quatre composants DCT 1D (colonnes) et quatre composants DCT 1D (lignes). L'instanciation des quatre composants est nécessaire pour un traitement en parallèle. Ce nombre peut être réduit à un lors d'un traitement en série au prix d'une fréquence beaucoup plus faible.

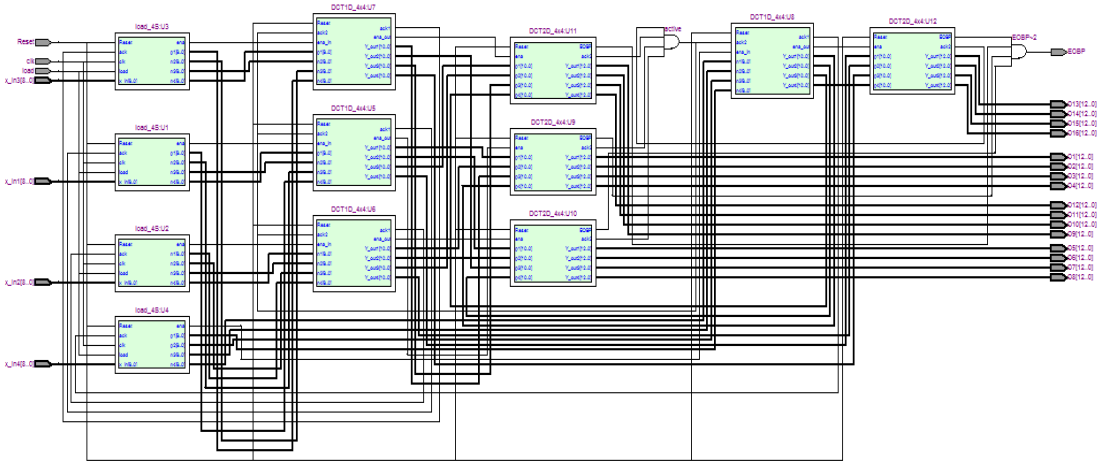


Figure V.26 Vue RTL de la DCT 2D entière 4x4 avec chargement de 4 données (version 2).

V.7.2.3 Mesure des performances des deux versions de la transformée DCT 2D entière 4x4 :

Le tableau V.11 résume les performances de l'implémentation de la DCT 2D entière 4x4 sous ses deux versions.

Implémentation de la DCT 2D entière 4x4	**Version1**	**Version2**
Nombre total d'éléments logiques	2008/33216 (6%)	2062/33216 (6%)
F_{max} (MHz)	160.90	180.86
Latence (cycles)	17	5
Puissance consommée (mW)	136.22	139.39

Tableau V.11 Performance de la DCT 2D entière 4x4(ciblant le circuit Cyclone II EP2C35F672C6 d'Altera)

Les résultats du tableau V.11 avantagent la version 2 de l'implémentation puisqu'à ressources matérielles utilisées égales, la fréquence maximale est 1.12x supérieure pour une puissance consommée 1.02x plus élevée. La mesure des performances montre aussi qu'un frame en format QCIF (176x144) peut être transformé en 65.69 µs , qu'un frame en format CIF (352x288) peut l'être en 262.74 µs et qu'un frame en format HDTV (1280x720) l'est en 2.38 ms.

V.7.2.4 Comparaison des résultats donnés par Matlab et ceux obtenus sous ModelSim/QuartusII :

Données	$\begin{bmatrix} 5 & 11 & 8 & 10 \\ 9 & 8 & 4 & 12 \\ 1 & 10 & 11 & 4 \\ 19 & 6 & 15 & 7 \end{bmatrix}$
DCT 1D (programmée sous Matlab)	34 35 38 33 -20 8 -21 14 14 -1 8 1 -30 9 7 -13
DCT 1D (implémentation hardware)	$\begin{bmatrix} 34 & 35 & 38 & 33 \\ -20 & 8 & -21 & 14 \\ 14 & -1 & 8 & 1 \\ -30 & 9 & 7 & -13 \end{bmatrix}$
DCT 2D (programmée sous Matlab)	140 -1 -6 7 -19 -39 7 -92 22 17 8 31 -27 -32 -59 -21
DCT 2D (implémentation hardware)	$\begin{bmatrix} 140 & -1 & -6 & 7 \\ -19 & -39 & 7 & -92 \\ 22 & 17 & 8 & 31 \\ -27 & -32 & -59 & -21 \end{bmatrix}$

Tableau V.12 Comparaison des résultats de la DCT 2D entière 4x4 software et hardware

Le tableau V.12 montre que l'erreur entre le résultat software et hardware est nulle. Ceci est un des avantages majeurs de la DCT 2D entière 4x4.

V.7.3 Implémentation de la DCT entière 8x8 :

Dans la norme de compression vidéo H.264/AVC et pour les nouveaux profils issus des extensions FREext, une transformée supplémentaire de taille 8×8 a été ajoutée. Les tests menés au chapitre IV ont montré que c'est la transformée la plus rapide, en comparaison à la transformée entière 4x4. La matrice de cette transformée est [59]:

$$T_{8x8} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \quad (75)$$

La méthodologie appliquée pour implémenter cette transformée est la même que celle utilisée pour l'implémentation de la DCT 4x4 à une seule différence. En effet, pour cette dernière, tous les éléments de la matrice sont des puissances de 2. Pour la transformée entière 8x8, certains ne le sont pas (3, 6, 10 et 12). Dans ce cas, ces multiplications doivent être décomposées en sommes de multiplications par des puissances de 2. Ainsi :

$$X * 3 = X * 2 + X$$

$$X * 6 = X * 4 + X * 2$$

$$X * 10 = X * 8 + X * 2 = X * 6 + X * 4$$

$$X * 12 = X * 8 + X * 4 = X * 10 + X * 2$$

L'implémentation hardware nécessite aussi, dans ce cas, un composant pour le chargement de 8 données et de leur conversion série parallèle. Les résultats des simulations menés avec ModelSim (voir figure V.27), confrontés à ceux obtenus avec Matlab, ont prouvé le bon fonctionnement du circuit proposé.

Bloc de données 8x8 (frame 1 de la séquence vidéo "Foreman") :

35	236	255	255	255	255	255	255
42	215	218	239	255	232	243	210
44	214	215	208	238	210	224	167
41	237	255	206	215	229	239	195
29	196	227	194	223	252	247	217
17	132	186	192	237	255	241	231
13	117	202	227	220	222	228	231
8	126	231	237	227	235	237	237

Résultat obtenu avec Matlab :

1832	11784	14312	14064	14960	15120	15312	13944
812	2915	706	468	668	1	240	-444
-240	-624	108	1000	352	28	72	812
-213	-424	-324	-120	211	848	387	1143
-24	936	1176	208	-240	416	336	520
15	140	319	132	-329	-365	-203	-52
-60	-172	-136	-160	176	124	-24	-24
33	70	-70	-2	-20	-174	-110	-196

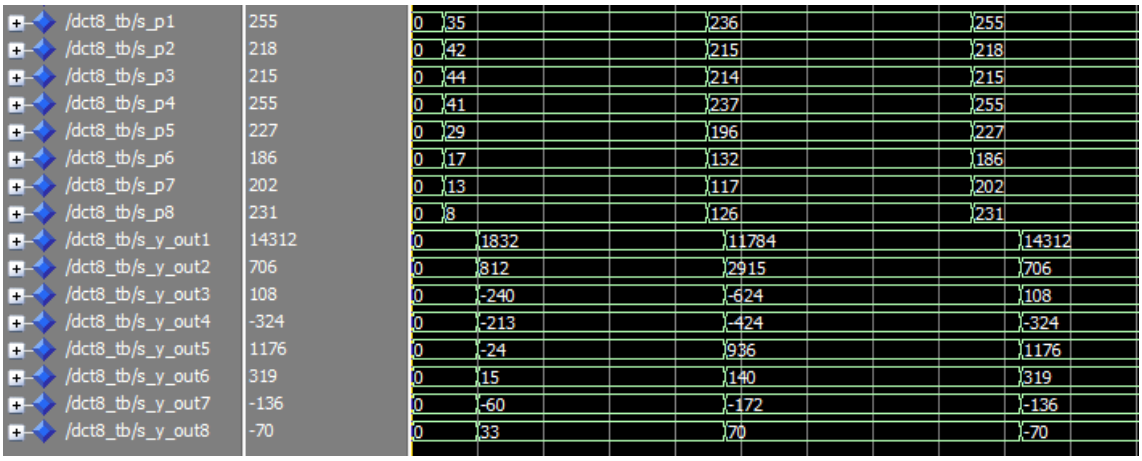


Figure V.27 Résultat de la simulation du circuit DCT 1D 8x8 entière (3 colonnes de données)

L'erreur, dans ce cas aussi est nulle. Les performances en termes de ressources matérielles, vitesse de traitement et puissance consommée seront présentés au paragraphe V.7.5 lorsqu'elles seront comparées à celles de la DCT 4x4 et la DCT 8x8 réelle.

V.7.4 Implémentation de la DCT classique 8x8 :

V.7.4.1 Méthodologie :

La DCT à une dimension est donnée par l'équation suivante :

$$Y(k) = w(k)\sum_{n=1}^{N} x(n).\cos\left(\frac{\pi(2n-1)(k-1)}{2N}\right) \qquad k= 1,2, \ldots,N \qquad (76)$$

$$W(k) = \begin{cases} \frac{1}{\sqrt{N}} sik = 1 \\ \frac{\sqrt{2}}{N} si\ 2 \leq k \leq N \end{cases}$$

Pour N=8, l'équation précédente s'écrit:

$$Y(k) = w(k)\sum_{n=1}^{8} x(n).\cos\left(\frac{\pi(2n-1)(k-1)}{16}\right) \qquad k= 1,2, \ldots,8 \qquad (77)$$

$$W(k) = \begin{cases} \frac{1}{2\sqrt{2}} sik = 1 \\ \frac{1}{2} si\ 2 \leq k \leq 8 \end{cases}$$

Le développement de l'équation précédente donne ce qui suit:

$$Y(1) = 1/2\sqrt{2}\ [x(1).\cos(0) + x(2).\cos(0) + \ldots + x(8).\cos(0)]$$

Y(2)=1/2[x(1).cos(π/16) + x(2).cos(3π/16) + x(3).cos(5π/16) + x(4).cos(7π/16) + x(5).cos(9π/16) + x(6).cos(11π/16) + x(7).cos(13π/16) + x(8).cos(15π/16)]

…

etc.

Notons par A, B, C, D, E, F et G les différents coefficients et réécrivons les équations précédentes sous forme matricielle :

$$\begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \\ y8 \end{bmatrix} = \begin{bmatrix} A & A & A & A & A & A & A & A \\ B & C & D & E & -E & -D & -C & -B \\ F & G & -G & -F & -F & -G & G & F \\ C & -E & -B & -D & D & B & E & -C \\ A & -A & -A & A & A & -A & -A & A \\ D & -B & E & C & -C & -E & B & -D \\ G & -F & F & -G & -G & F & -F & G \\ E & -D & C & -B & B & -C & D & -E \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \end{bmatrix} \quad (78)$$

Cette représentation montre que le calcul d'une DCT à une dimension nécessite 8 multiplications et 7 additions par ligne. L'ensemble nécessitera alors 8 fois (8 multiplications et 7 additions), soit 64 multiplications et 56 additions.

1. <u>Détermination et représentation des coefficients :</u>

Dans le but de minimiser l'erreur sans surconsommer les ressources hardwares, les coefficients sont rééchelonnés, arrondis vers les entiers les plus proches, puis représentés en binaire sur 12 bits selon la formule suivante:

Coefficient en binaire = $dec2bin(\left(round(coefficient * 2^{11})\right), 12)$ (79)

Dec2bin étant la fonction Matlab de conversion du décimal au binaire.

Le tableau suivant récapitule les valeurs des différents coefficients ainsi que leurs représentations en binaire et en complément à 2.

Coefficient	Valeur réelle	Représentation en binaire	Représentation en complément à 2 de (- coefficient)
A	0.3536	001011010100	110100101100
B	0.4904	001111101100	110000010100
C	0.4157	001101010011	110010101101
D	0.2778	001000111001	110111000111
E	0.0975	000011001000	111100111000
F	0.4619	001110110010	110001001110
G	0.1913	000110001000	111001111000

Table V.13 Valeurs des différents coefficients de la DCT classique 8x8

2. Technique des coefficients distribués :

Cette technique est très similaire à l'arithmétique distribuée, sauf qu'au lieu de distribuer les données, on distribue les coefficients.

Précisons, d'abord, qu'un entier signé, peut être retrouvé à partir de sa représentation en complément à 2, sur B bits de la façon suivante :

$C = - k_{B-1}2^{B-1} + \sum_{B-2}^{0}(k_j \quad 2^j)$ (80)

Exemple: Avec une représentation sur 5 bits, $(+10)_{10}$ s'écrit $(01010)_2$ et $(-10)_{10}$ s'écrit $(10110)_{cpl2}$.

$(01010)_2 \rightarrow -0.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 8 + 2 = (10)_{10}$

$(10110)_{cpl2} \rightarrow -1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 = -16 + 4 + 2 = (-10)_{10}$

Dans le cas d'un produit scalaire $<C_i x_i>$ avec i=3, en travaillant toujours sur 5 bits et en considérant le produit scalaire suivant :

$F = C_0x_0 + C_1x_1 + C_2x_2$ avec $C_0= -5$; $C_1 = -10$; $C_2= 6$; $x_0= 2$; $x_1= - 3$ et $x_2= 4$

$F= -10 + 30 + 24 = 44$

Les coefficients en complément à 2, sur 5 bits, s'écrivent :

$-5 \rightarrow (11011)$, $-10 \rightarrow (10110)$ et $6 \rightarrow (00110)$

$F= (-1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0)\ x0 + (-1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0)\ x1$

$+ (-0.24 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0)\ x2$

$$F = [2^0\ 2^1\ 2^2\ 2^3\ 2^4]\begin{matrix} C_0 & C_1 & C_2 \\ \downarrow & \downarrow & \downarrow \end{matrix}\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ -1 & -1 & 0 \end{bmatrix}\begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} = [2^0\ 2^1\ 2^2\ 2^3\ 2^4]\begin{bmatrix} x0 \\ x0+x1+x2 \\ x1+x2 \\ x0 \\ -(x0+x1) \end{bmatrix}$$

$$F = [2^0\ 2^1\ 2^2\ 2^3\ 2^4]\begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix} = 2 + 6 + 4 + 16 + 16 = 44$$

3. Cas d'une matrice :

En utilisant la matrice (76), la DCT 1D d'une matrice 8x8 s'écrit :

$$\begin{bmatrix} A & A & A & A & A & A & A & A \\ B & C & D & E & -E & -D & -C & -B \\ F & G & -G & -F & -F & -G & G & F \\ C & -E & -B & -D & D & B & E & -C \\ A & -A & -A & A & A & -A & -A & A \\ D & -B & E & C & -C & -E & B & -D \\ G & -F & F & -G & -G & F & -F & G \\ E & -D & C & -B & B & -C & D & -E \end{bmatrix} x \begin{bmatrix} x00 & x01 & x02 & x03 & x04 & x05 & x06 & x07 \\ x10 & x11 & x12 & x13 & x14 & x15 & x16 & x17 \\ x20 & x21 & x22 & x23 & x24 & x25 & x26 & x27 \\ x30 & x31 & x32 & x33 & x34 & x35 & x36 & x37 \\ x40 & x41 & x42 & x43 & x44 & x45 & x46 & x47 \\ x50 & x51 & x52 & x53 & x54 & x55 & x56 & x57 \\ x60 & x61 & x62 & x63 & x64 & x65 & x66 & x67 \\ x70 & x71 & x72 & x73 & x74 & x75 & x76 & x77 \end{bmatrix}$$

(81)

Chaque élément de la première ligne du résultat est le produit scalaire du vecteur $[AAAAAAAA]$ et d'une colonne de la matrice des échantillons x_{ij} ($0 \leq i \leq 7\ et\ 0 \leq j \leq 7$). Les éléments de la 2ème ligne résultent des produits scalaires du vecteur [B C D E –E –D –C –B] et de chaque colonne de la matrice [X].

4. Determination de l'élément (1,1) note F_0:

L'élément (1,1) est le résultat du produit scalaire (79) et donne lieu au produit suivant:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x00 \\ x10 \\ x20 \\ x30 \\ x40 \\ x50 \\ x60 \\ x70 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ 0 \end{bmatrix} \quad (82)$$

Le résultat doit être, maintenant, pondéré par les puissances de 2 :

$$\text{Elément (1,1)} = [2^0\ 2^1\ 2^2\ \dots\ 2^{11}] \begin{bmatrix} 0 \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ \sum_0^7 x_{i0} \\ 0 \\ \sum_0^7 x_{i0} \\ 0 \\ 0 \end{bmatrix} \quad (83)$$

Tous les éléments de la 1ère ligne de la matrice résultat seront calculés de la même façon. Il suffit de remplacer les x_{i0} par x_{i1}, x_{i2}, … x_{i7}.

5. <u>Détermination de l'élément (2,1) noté F_1:</u>

Cet élément est donné par:

$$[B\ C\ D\ E\ -E\ -D\ -C\ -B] \begin{bmatrix} x00 \\ x10 \\ x20 \\ x30 \\ x40 \\ x50 \\ x60 \\ x70 \end{bmatrix} = [B\ C\ D\ E] \begin{bmatrix} x00 - x70 \\ x10 - x60 \\ x20 - x50 \\ x30 - x40 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x00 - x70 \\ x10 - x60 \\ x20 - x50 \\ x30 - x40 \end{bmatrix} = \begin{bmatrix} (x10 - x60) + (x20 - x50) \\ (x10 - x60) \\ (x00 - x70) \\ (x00 - x70) + (x20 - x50) + (x30 - x40) \\ (x10 - x60) + (x20 - x50) \\ (x00 - x70) + (x20 - x50) \\ (x00 - x70) + (x10 - x60) + (x30 - x40) \\ (x00 - x70) + (x30 - x40) \\ (x00 - x70) + (x10 - x60) \\ (x00 - x70) + (x10 - x60) + (x20 - x50) \\ 0 \\ 0 \end{bmatrix} \quad (84)$$

Tous les éléments de la 2ème ligne de la matrice seront calculés de cette façon, en remplaçant les xi0 par les xi1 puis xi2 … etc. Bien entendu, il ne faut pas oublier la pondération par les puissances de 2.

6. <u>Détermination de l'élément (3,1) noté F_2 :</u>

$$[F \;\; G \;\; -G \;\; -F \;\; -F \;\; -G \;\; G \;\; F] \begin{bmatrix} x00 \\ x10 \\ x20 \\ x30 \\ x40 \\ x50 \\ x60 \\ x70 \end{bmatrix} = [F \;\; G] \begin{bmatrix} (x00 + x70) - (x30 + x40) \\ (x10 + x60) - (x20 + x50) \end{bmatrix} =$$

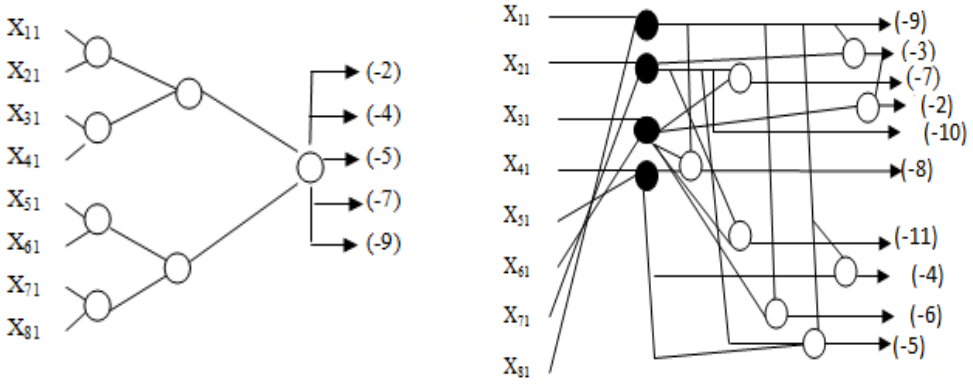
$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} (x00 + x70) - (x30 + x40) \\ (x10 + x60) - (x20 + x50) \end{bmatrix} =$$

$$\begin{bmatrix} 0 \\ (x00 + x70) - (x30 + x40) \\ 0 \\ (x10 + x60) - (x20 + x50) \\ (x00 + x70) - (x30 + x40) \\ (x00 + x70) - (x30 + x40) \\ 0 \\ (x00 + x70) - (x30 + x40) + (x10 + x60) - (x20 + x50) \\ (x00 + x70) - (x30 + x40) + (x10 + x60) - (x20 + x50) \\ (x00 + x70) - (x30 + x40) \\ 0 \\ 0 \end{bmatrix} \quad (85)$$

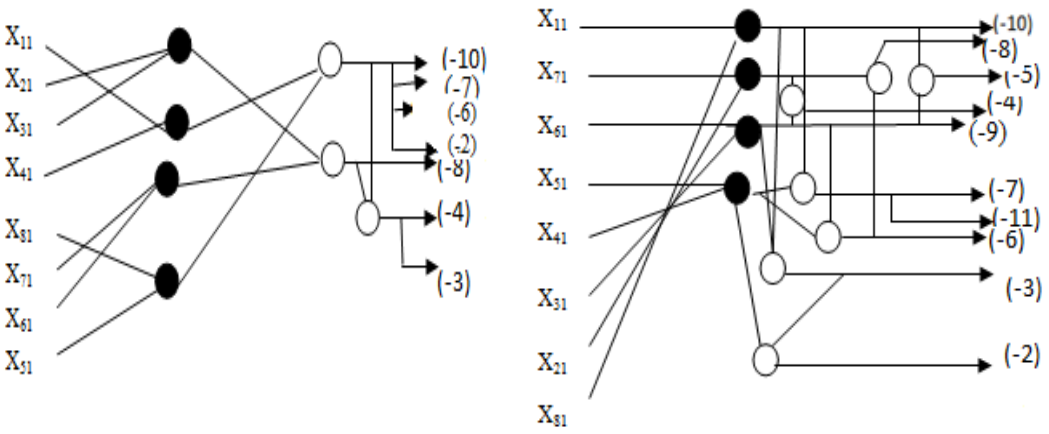
Et c'est ainsi que seront calculés tous les éléments de la 3ème ligne sans oublier la pondération par les puissances de 2.

Le résultat final doit être remis à la bonne échelle en le divisant par 2^{11}.

Les autres coefficients sont calculés d'une manière similaire et la figure V.28 facilite le suivi des opérations à mener pour calculer les quatre premiers coefficients DCT.



(a) *Schéma de calcul du coefficient F_0* (b) *Schéma de calcul du coefficient F_1*



(c) *Schéma de calcul du coefficient F_2* (d) *Schéma de calcul du coefficient F_3*

Figure V.28 Regroupement en schémas papillons des opérations pour le calcul des quatre premiers coefficients de la DCT 8x8 réelle.

● = Soustraction

○ = Addition

Les valeurs entre parenthèses représentent les puissances de 2 par lesquelles chaque résultat intermédiaire doit être pondéré afin d'obtenir le résultat final. Les valeurs absolues des ces valeurs correspondent aux nombres de décalages à droite à effectuer.

V.7.4.2 Implémentation et simulation :

La méthodologie suivie pour l'implémentation est similaire à celle appliquée pour l'implémentation de la DCT entière 4x4. Elle consiste en :

1. Un circuit de chargement en série de 8 données, puis de leur conversion en parallèle.
2. Un circuit de calcul de la DCT 1D : Ce circuit reçoit 8 entrées parallèles (les valeurs des pixels d'une colonne d'un bloc 8x8 du frame) et génère 8 coefficients calculés selon les schémas en papillons. La figure V.29 montre le résultat de la simulation de la DCT 1D sur un bloc 8x8.

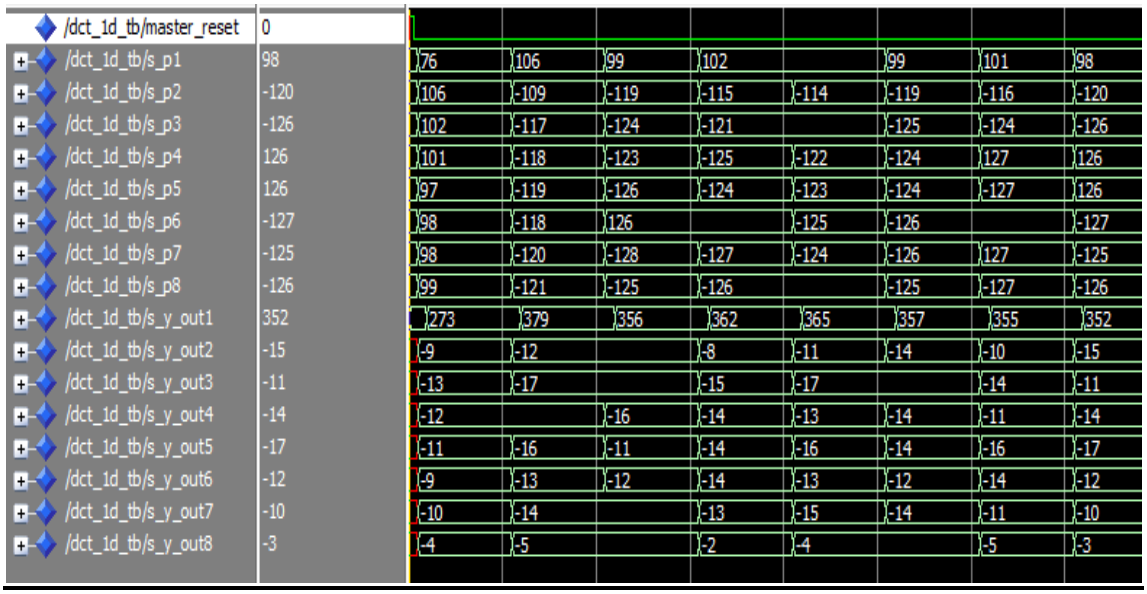


Figure V.29 Résultat de la simulation du circuit DCT1D 8x8 réelle proposé.

V.7.4.3 Comparaison des résultats obtenus sous Matlab avec ceux de la simulation de l'implémentation hardware:

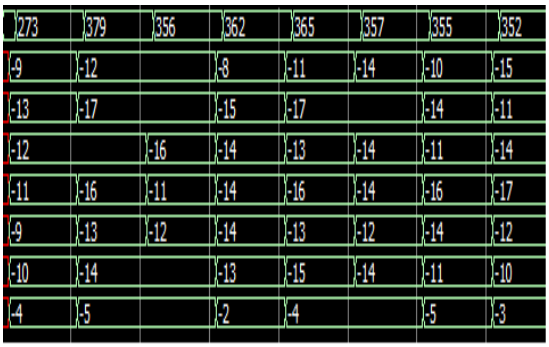
Le bloc test est :

$$\begin{bmatrix} 76 & 106 & 99 & 102 & 102 & 99 & 101 & 98 \\ 106 & 147 & 137 & 141 & 142 & 137 & 140 & 136 \\ 102 & 139 & 132 & 135 & 135 & 131 & 132 & 130 \\ 101 & 138 & 133 & 131 & 134 & 132 & 127 & 126 \\ 97 & 137 & 130 & 132 & 133 & 132 & 129 & 126 \\ 98 & 138 & 126 & 126 & 131 & 130 & 130 & 129 \\ 98 & 136 & 128 & 129 & 132 & 130 & 127 & 131 \\ 99 & 135 & 131 & 130 & 130 & 131 & 129 & 130 \end{bmatrix}$$

Le tableau V.14 montre le résultat de la transformée DCT 1D donné par Matlab (a) et celui obtenu en simulant le circuit implémenté en hardware (b).

277	384	366	361	359	356	357	359
-9	-12	-11	-8	-12	-13	-13	-13
-12	-16	-16	-18	-19	-12	-14	-14
-10	-13	-13	-11	-12	-10	-13	-13
-10	-13	-11	-13	-14	-14	-11	-13
-9	-12	-11	-13	-13	-15	-15	-12
-10	-14	-11	-7	-10	-11	-8	-7
-4	-4	-4	-4	-5	-4	-5	-5

(a)



(b)

Tableau V.14 Comparaison des résultats de Matlab (a) et ceux de Modelsim (b) pour un bloc 8x8 pixels.

La différence est :

4	5	10	-1	-6	-1	2	7
0	0	1	0	-1	1	-3	2
1	1	1	-3	-2	5	0	-3
2	-1	3	3	1	4	-2	1
1	3	0	1	2	0	5	4
0	1	1	1	0	-3	-1	0
0	0	3	6	5	3	3	3
0	1	1	-2	-1	0	0	-2

Il apparait clairement qu'il existe des différences entre les résultats obtenus sous Matlab et ceux obtenus avec Modelsim. Les sources d'erreurs peuvent se résumer aux points suivants:

- ✓ Dans la formule de la DCT, les termes cosinus ont été calculés avec une précision au 10000ème (ils ont été considérés avec 4 chiffres après la virgule). Des essais ont montré que l'erreur diminue en travaillant avec une plus grande précision.

✓ Les coefficients ont été codés sur 12 bits dans le but de réduire les ressources matérielles nécessaires à l'implémentation de la DCT. Une plus grande résolution contribue à réduire l'erreur entre les deux résultats.

✓ Les multiplications/divisions ont été évitées et remplacées par des opérations de décalage à gauche/droite. Les décalages à droite correspondent à des divisions entières et introduisent par conséquent des erreurs à chaque fois que le dividende est impair.

V.7.5 Comparaison des performances de la DCT entière 4x4, la DCT entière 8x8 et la DCT classique 8x8 :

Les figures V.30 et V.31 et V.32 dévoilent la complexité calculatoire et les ressources matérielles utilisées pour implémenter la DCT entière 4x4 1D, la DCT entière 8x8 1D et la DCT classique 8x8 1D respectivement.

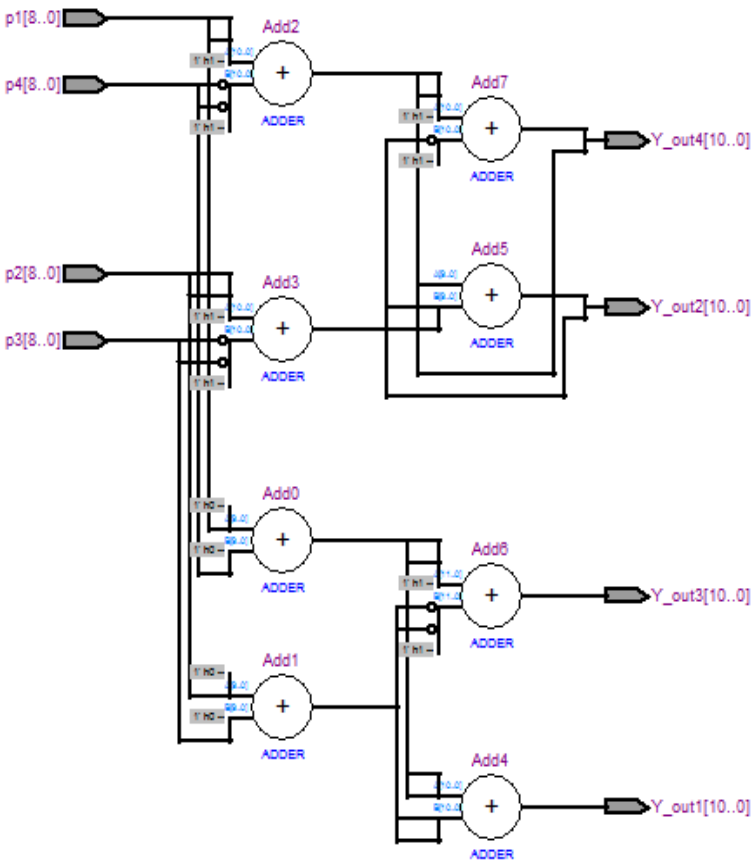


Figure V.30 Vue RTL du circuit DCT 4x4 1D

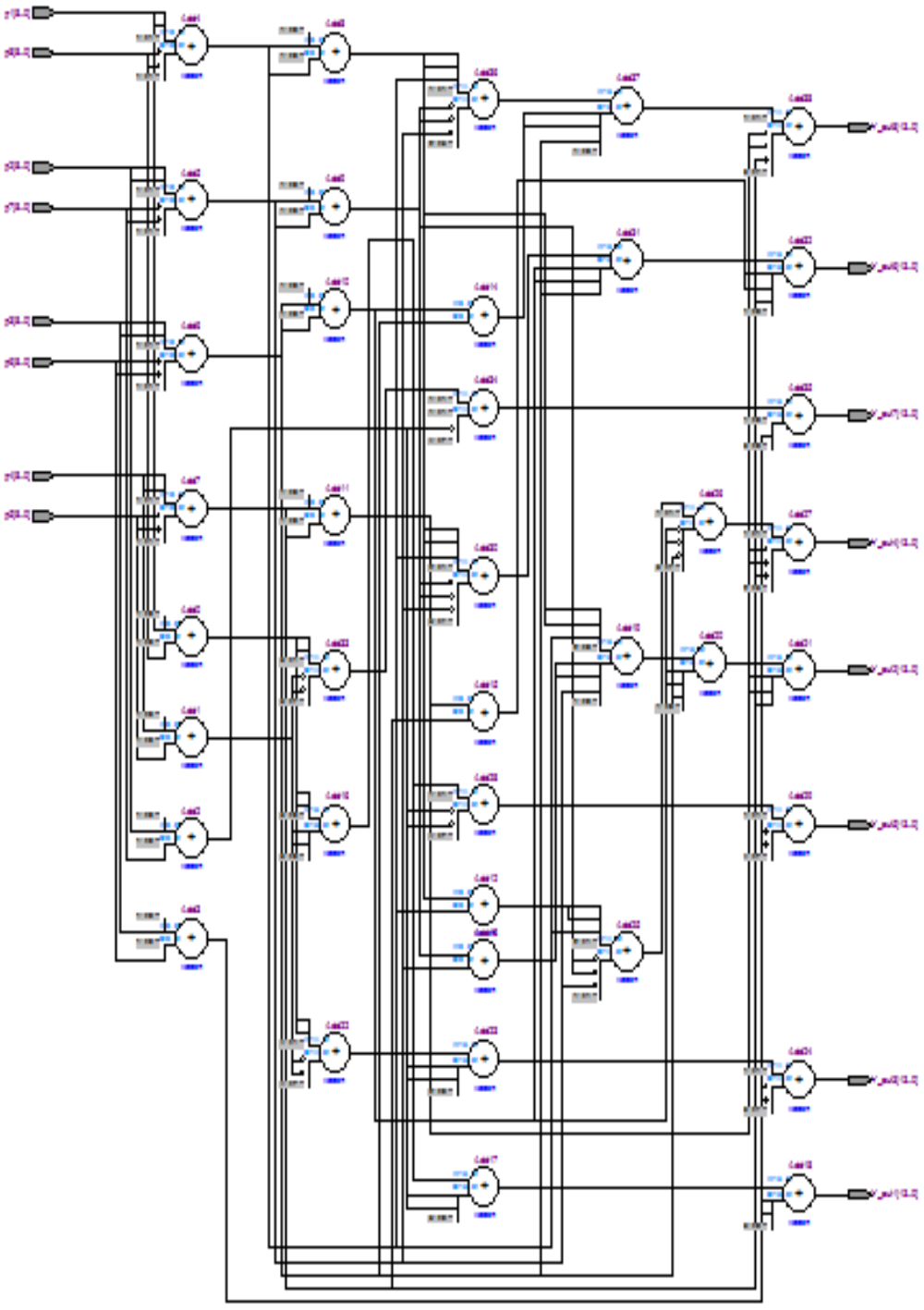


Figure V.31 Vue RTL du circuit DCT entière 8x8 1D

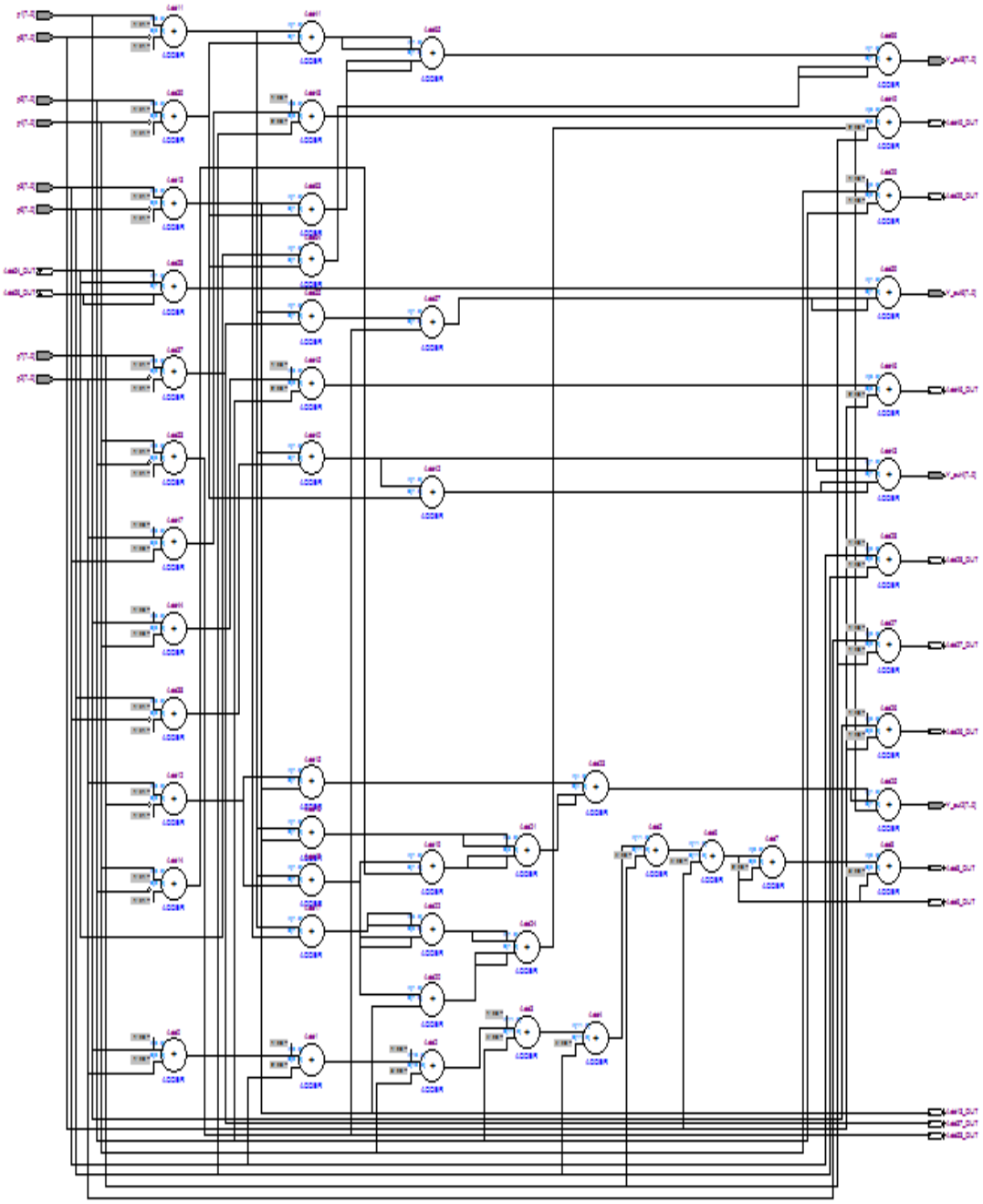


Figure V. 32 Vue RTL du circuit DCT classique 8x8 1D (partie ½)

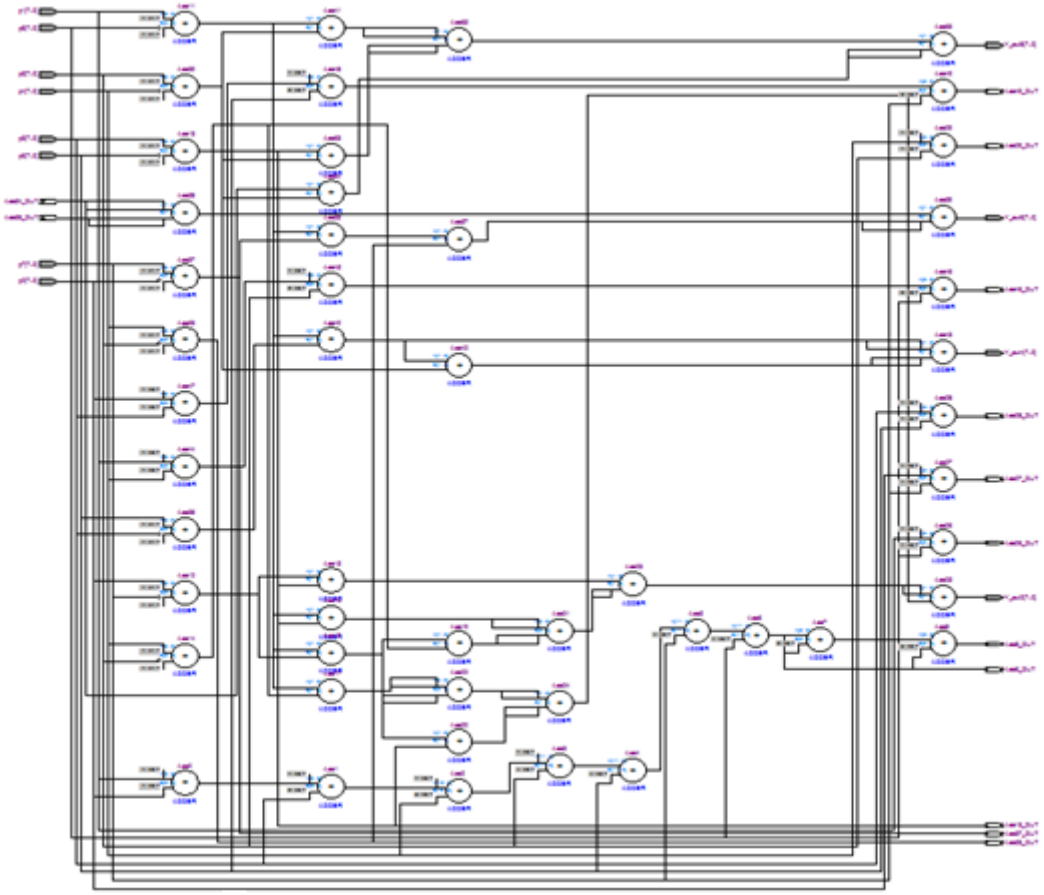


Figure V.33 Vue RTL du circuit DCT 8x8 classique 1D (partie 2/2)

Le tableau V.15 quantifie les ressources matérielles utilisées. Il montre que la DCT classique 1D 8x8 en consomme 6.7x plus que la DCT entière 1D 4x4 et que la DCT entière 1D 8x8 en consomme 5x plus. Ceci étant pour un seul circuit traitant 4 (cas de la DCT 4x4) et 8 (dans les deux autres cas) données. Pour gagner en vitesse de traitement et en supposant (ce qui n'est pas le cas réel) que la deuxième phase de la DCT utilise la même quantité d'éléments logiques, le parallélisme nécessiterait alors 8 de ce circuit (cas de la DCT 4x4) et 16 (cas de la DCT 8x8). Quant au tableau V.15, il récapitule la métrique des performances de l'implémentation de l'ensemble (circuit de chargement et de conversion de 4 données + un composant DCT 4x4 1D) et de l'ensemble (circuit de chargement et de conversion de 8 données + un composant DCT 8x8 1D) dans le cas de la DCT 1D entière 8x8 et la DCT 1D réelle 8x8. Ces rapports de synthèse ont été générés par le logiciel Quartus II8.1, en ciblant le circuit cyclone II/EP2C35F672C6 disponible dans la carte DE2 d'Altera.

	Nombre total d'éléments logiques
Circuit DCT 4x4 1D (entière)	82/32216 (<1%)
Circuit DCT 8x8 1D (entière)	410/32216 (1%)
Circuit DCT 8x8 1D (réelle)	550/32216 (2%)

Tableau V.15 Ressources matérielles consommées par la DCT1D 4x4 entière, la DCT 1D 8x8(entière) et la DCT 1D 8x8 (réelle)

	DCT 1D 4x4 entière	DCT 1D 8x8 entière	DCT 1D 8x8 réelle
Nombre total d'éléments logiques	302/32216 (<1%)	851/32216 (2.64%)	897/32216 (2.78%)
Fmax (MHz)	145.73	81.01	67.68
Latence (cycles)	4.5	8.5	8.5
Puissance consommée (mW)	116.91	124.77	119.08

Tableau V.16 Comparaison des performances de la DCT 1D 4x4, la DCT 8x8entière et la DCT 8x8 réelle.

Les mesures du tableau V.16 permettent de conclure que la DCT entière 4x4 est beaucoup plus simple à implémenter en hardware et qu'elle est plus performante que la DCT 8x8 entière et réelle sur tous les aspects (économie en ressources matérielles, vitesse de traitement plus élevée et puissance consommée plus faible). Pour la DCT entière 8x8, la fréquence peut être améliorée par un traitement en parallèle. Elle atteindra la même vitesse de traitement avec une fréquence égale à la moitié de celle de la DCT entière 4x4. Mais cette performance sera atteinte au prix de larges ressources matérielles et d'une puissance consommée très élevée.

V.8 Conclusion :

Ce dernier chapitre a été entièrement consacré à la présentation de l'implémentation hardware de la transformée en ondelettes et la transformée en DCT. Dans la première partie, une méthodologie basée sur la technique de l'arithmétique distribuée a été détaillée et les résultats de son application sur le banc de filtres CDF9/7 ou bior4.4 ont été présentés et discutés. La méthode proposée permet d'implémenter tout type de filtres RIF, surtout à phase linéaire, et quelle que soit la DWT sur un niveau donné. Le processus peut, bien entendu, être réitéré pour passer à un nombre de niveaux supérieur.

Cette même méthode peut être aussi exploitée dans l'implémentation des filtres anti-blocs introduits dans la norme H.264/AVC. Il suffit, pour cela, d'implémenter plusieurs filtres RIF avec différents coefficients et de rajouter un multiplexeur pour la sélection. En plus, il a été proposé deux schémas de conception qui orientent le concepteur selon les performances visées.

Dans la deuxième partie de ce chapitre, une implémentation hardware de la DCT entière 4x4 sur FPGA a été exposée et les performances de plusieurs propositions ont été discutées. Ces dernières, relatives au circuit FPGA utilisé "Cyclone II EP2C35F672C6", ont été comparées à celles d'une implémentation hardware de la DCT 8x8 entière et de la DCT 8x8 réelle. Les structures de la DCT 4x4 sont beaucoup plus simples, beaucoup plus rapides que celles de la DCT 8x8 et offrent un bon compromis entre performances et coût (surface, latence et dissipation de puissance). L'autre avantage de la DCT entière 4x4/8x8 est qu'il n'y a aucune perte de précision lors de l'opération de décodage.

Il faut, enfin, signaler que l'implémentation des transformées inverses se fait en appliquant les mêmes procédures décrites dans ce chapitre.

Conclusion et perspectives :

Les télécommunications numériques d'aujourd'hui et aussi celles du futur doivent répondre à des contraintes. En effet, il est nécessaire de travailler en temps réel tout en garantissant une très bonne qualité des données transmises (audio, vidéo et autres). D'un autre coté, les canaux de transmission sont de plus en plus limités en bande passante, ce qui a exigé la mise en œuvre d'approches de compression capables à la fois de réduire énormément la quantité de données à transmettre et/ou à stocker pour les adapter aux caractéristiques du canal mais aussi et surtout de préserver la haute résolution des données telles que les images numériques et la vidéo. On compte aujourd'hui, plusieurs normes de compression telles que JPEG, JPEG2000, la série des MPEG-x et la série des H.26X dont le H.264/AVC. Ces techniques, très complexes, exigent du hardware adopté certaines performances comme la flexibilité, l'encombrement, la vitesse de traitement, la consommation électrique …etc.

L'objectif de cette thèse est une contribution au développement et à l'implémentation sur un circuit de type FPGA, de noyaux accélérateurs des traitements de compression vidéo, en l'occurrence les transformées DWT et DCT. D'abord, les fondements de codage vidéo ont été abordés afin d'avoir une vision détaillée des principes employés dans tout type de système de compression vidéo. Nous nous sommes intéressés en particulier à la norme H.264/AVC, un des codeurs les plus récents, de sa structure et de ses éléments principaux. Il a été constaté que le groupe JVT avait déjà prévu une implémentation matérielle de la partie transformée puisqu'il avait développé une transformation entière 4x4 et une transformée entière 8x8 dans la partie FRExt. Par la suite, une étude de l'impact d'un certain nombre de paramètres a été conduite sur la plateforme JM 18.2. Les tests conduits sur les trois modes de transformation spatiale nous ont permis de conclure que le mode 1 n'apporte qu'une modeste amélioration de la qualité. Cependant, le temps de codage global est relativement très long. Néanmoins, il est à préciser que ces tests n'ont concerné que deux formats vidéo: QCIF et CIF et des séquences assez courtes. Les simulations menées sur le filtre anti-blocs ont été les plus difficiles. En effet, non seulement le nombre de tests est très élevé, mais aussi l'analyse et l'interprétation des résultats sont très ardues. L'effet du filtrage reste imprévisible. Il a été, alors conclu que le jugement des performances dépend de l'application et du type de la vidéo. Néanmoins, un compromis doit toujours être fait entre les trois variables : qualité, débit et temps de codage.

Par la suite, une méthodologie sur l'implémentation des filtres RIF a été développée et a permis à la fois d'implémenter la transformée DWT et les filtres anti-blocs recommandés par le standard H.264. Le procédé a été appliqué sur le banc de filtres CDF9/7, recommandé par le standard JPEG2000 pour la compression avec pertes. La transformée en DCT (entière 4x4, entière 8x8 et réelle 8x8) a aussi été implémentée en ciblant toujours une optimisation des performances en termes de vitesse de traitement, surface, débit et consommation d'énergie.

Réalisé sur un circuit FPGA d'Altera, le Cyclone II EP2C35F672C6, les tests menés ont permis de déduire que, dans le cas de l'algorithme à base d'arithmétique distribuée modifiée, un frame en format QCIF peut être traité en 947.44 µs, ce qui équivaut au traitement de 1055 frames QCIF par seconde, 263 frames CIF et 29 frames en format HDTV. Le nombre de frames traités est en réalité plus grand que ces estimations puisque le filtrage ne concerne que les bords des blocs et non pas l'ensemble du frame.

Pour la DCT entière 4x4, la vitesse de transformation est estimée à 15224 frames QCIF par seconde, à 3806 frames CIF par seconde et à 419 frames HDTV par seconde. Les temps de transformation d'un frame évalués à 65.69 µs, 262.74 µs et 2.38 ms pour les formats QCIF, CIF et HDTV sont nettement inférieurs au 33.3 ms nécessaire pour le traitement d'un frame d'une vidéo cadencée à 30 frame/seconde. Ces performances peuvent être optimisées en trouvant d'autres solutions au niveau de l'organisation de la mémoire où sont stockés les échantillons à traiter.

Les noyaux développés peuvent être utilisés ultérieurement sur une plateforme comportant un processeur embarqué et exploités dans des travaux sur la toute dernière norme de compression vidéo HEVC appelée aussi H.265. En effet, les premières informations [114] indiquent qu'un élargissement du nombre de modes de transformation spatiale a lieu. Désormais, les possibilités sont des transformées en DCT entières 4x4, 8x8, 16x16, 32x32 et une transformée en sinus discrète (DST) 4x4. Le filtre anti-blocs a été simplifié et la grille utilisée est de taille 8x8 (elle est de taille 4x4 pour le H.264/AVC). Le paramètre "boundary strength" ou BS ne prend plus que 3 valeurs (0-2) contre 5 valeurs pour le H.264/AVC (0-4). Enfin, le filtrage n'a lieu que si BS=2 alors qu'il est appliqué à chaque fois que BS est différent de zéro dans le H.264/AVC.

Cette thèse et l'ensemble des simulations menées aussi bien avec la plateforme JM 18.2 que les implémentations hardware sur FPGA constituent une bonne base pour les

recherches futures, en particulier l'évaluation des performances de la nouvelle norme de compression vidéo HEVC.

Des simulations sont à mener sur les nouveaux modes de transformation spatiale. Elles doivent porter sur différents formats tels que le QCI, CIF mais aussi sur les formats HDTV (1080x720), le 4K (4096x2160) et l'ultra HDTV ou 8K (7680x4320). Les séquences vidéo doivent aussi être très diversifiées du point de vue activité et de durées assez longues.

L'étude du filtre anti-blocs constitue aussi un champ d'investigation très intéressant. Le choix des paramètres de ce filtre et la décision de l'utiliser ou pas doivent être régis par des conditions un peu plus exactes et clairement établies.

Toutes ces recherches doivent se concentrer sur le facteur qualité-débit. Certes, ce paramètre est lié au temps et les deux sont de nature contradictoire, mais l'implémentation sur des circuits programmables tels que les FPGA constitue une solution très attirante.

Bibliographie

[1] http://www.computerweekly.com/feature/What-does-a-petabyte-look-like

[2] G. E. Moore, Cramming more components onto integrated circuits. Electronics 38(8), Apr. 19(1965).

[3] P. Benoit, « Architectures des Accélérateurs de Traitement Flexibles pour les Systèmes sur Puces », Thèse, Université de Montpellier II ,11 octobre 2004.

[4] ITU-T, "Video codec for audiovisual services at px64 kbits/s," ITU-T Rec. H.261 v1: Nov 1990, v2: Mar.1993.

[5] ISO/IEC JTC 1, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video," ISO/IEC 11172 (MPEG-1), Nov. 1993.

[6] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2:Video," ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994

[7] ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; v1: Nov. 1995, v2: Jan. 1998,v3: Nov. 2000.

[8] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2), Jan.1999

[9] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," document JVT-G050r1, May 2003; documents JVT-K050r1) and JVT-K051r1 March 2004; and Fidelity Range Extensions documents JVT-L047 and JVT-L050 July 2004.

[10] S. E. Wahlstrom, "Programmable logic arrays—cheaper by the millions". Electronics 40(25): 90–95, (Dec. 1967).

[11] http://www.altera.com

[12] http://www.xilinx.com

[13] Global Field-Programmable Gate Array (FPGA) Market 2010-2014 http://www.aarkstore.com/reports/Global-Field-Programmable-Gate-Array-FPGA-Market-2010-2014-108145.html

[14] Jean-Pierre Deschamps, Géry J. A. Bioul, and Gustavo D, "Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems" ,John Wiley & Sons, Inc.2006

[15] M. Nagabushanam et al., "Design and FPGA Implementation of Modified Distributive Arithmetic Based DWT – IDWT Processor for Image Compression", International Conference on Communications and Signal Processing (ICCSP), Bangalore, India, 2011.

[16] A. Ahmad et al., "Efficient Implementation of a 3-D Medical Imaging Compression System Using CAVLC," Proceedings of 2010 IEEE 17th International Conference on Image Processing, Hong Kong, September 26-29, 2010.

[17] Zhe Zhang, Peng Cao, Hu Chen, "A New Hardware & Software Co-Design of JPEG2000 Encoder", Conference on Image and Signal Processing (CISP '09), Nanjing, China, 2009.

[18] Morteza Gholipour, Hossein Ahmadi Noubari, "Hardware Implementation of Lifting Based Wavelet Transform", 2nd International Conference on Signal Processing Systems (ICSPS), Dalian, China, 2010.

[19] Xiaonan Fan et al., "A Pipeline Architecture for 2-D Lifting-based Discrete Wavelet Transform of JPEG2000", International Conference on Multimedia Technology (ICMT), 29-31 Oct., Ningbo, China, 2010.

[20] Gaurav Tewari, Santu Sardar, K. A. Babu, "High-Speed & Memory Efficient 2-D DWT on Xilinx Spartan3A DSP using scalable Polyphase Structure with DA for JPEG2000, Standard", 3rd International Conference on Electronics Computer Technology (ICECT), Kanyakumari, India, 2011.

[21] Xiaodong Xu, Yiqi Zhou, "Efficient FPGA Implementation of 2-D DWT for 9/7 Float Wavelet Filter", Proc. Of ICIECS, 19-20 Dec, Wuhan, China, 2009.

[22] Md. Shabiul Islam et al., "Field programmable gate array (FPGA) realization of a fast 2-D discrete cosine transform algorithm for higher image compression", International Journal of the Physical Sciences Vol. 7(4), pp. 519 - 526, 23 January, 2012.

[23] I. Martisius et al., "A 2-D DCT Hardware Codec based on Loeffler Algorithm", Electronics and Electrical Engineering, 2011. No. 7(113), ISSN 1392 – 1215.

[24] Aree Ali Mohammed, Jamal Ali Hussein, "Hybrid Transform Coding Scheme for Medical Image Application", International Journal of Computer Applications, 27(7):16-20, August 2011.

[25] Rafael C. Gonzalez, Richard E. Woods et Steven L. Eddins, "Digital Image Processing Using MATLAB", Pearson Prentice-Hall, 2004, pp. 12-15.

[26] Ahmed Ben Atitallah, "Etude et Implantation d'Algorithmes de Compression d'Images dans un Environnement Mixte Matériel et Logiciel", thèse de doctorat en électronique, Université Bordeaux 1, Juillet 2007.

[27] Sébastien Crespin « http ://screspin.free.fr/mpeg/rapport.htm».

[28] Yun Q. Shi et Huifang Sun, "Image and Video Compression for Multimedia Engineering, Fundamentals, Algorithms, and Standards ", CRC Press, USA, 2000.

[29] Cédric MARIN, "Vers une solution réaliste de décodage source-canal conjoint de contenus multimédia", thèse de doctorat en physique, Université Paris-sud 11, 2009.

[30] ISO Comittee. ISO/IEC 14496 Part 2 - Coding of audio-visual objects, second edition, 2001/12/01.

[31] Edouard Gomez et Lionel Dufresne. "Mémoire de Multimédia, MPEG4 et XviD", I.S.E.P, 27 Juin 2003.

[32] MPEG Comittee Documentation, http ://www.chiariglione.org/mpeg/

[33] WikiPedia, MPEG-2 and DCT Articles, http ://en.wikipedia.org/wiki/MPEG-2.

[34] Standard MPEG-1 : ISO/IEC 11172-2, Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mb/s.

[35] Standard MPEG-2 : ISO/IEC 13818-2, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information.

[36] Standard MPEG-4 : ISO/IEC 14496-2, Information Technology – Coding of Audio-Visual Objects.

[37] H.261 : Video Codec for Audiovisual Services at p x 64 kb/s. RecommandationH.261 à l'UIT-T. Mars 1993.

[38] H.263 : Video Coding for Low Bit rate Communication. Première Recommandation H.263 à l'UIT-T. Mars 1996.

[39] H.263+ : Video Coding for Low Bit rate Communication. Deuxième recommandation H.263 à l'UIT-T. Février 1998.

[40] H.263++ : H.263 Annexe U, V, W and X. Compléments de la Recommandation H.263 à l'UIT-T. Janvier 2000.

[41] H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) - H.264 learning from WikiPedia.

[42] D. S. Hands, Q. Huynh-Thu, A. W. Rix, A. G. Davis, and R. M. Voelcker, "Objective perceptual quality measurement of 3g video services," in Proc. Fifth IEE Int. Conf. 3GMobile Communication Technologies 3G 2004, 2004, pp. 437–441.

[43] H. S. Z. Wang, A. Bovik and E. Simoncelli, "Image quality assessment: From error measurement to structural similarity," IEEE Trans. Image Processing, vol. 13, no. 4, pp. 600–612, 2004.

[44] S. E. Ghrare, M. A. M. Ali, M. Ismail, and K. Jumari, "Diagnostic quality of compressed medical images: Objective and subjective evaluation," in Proc. Second Asia Int. Conf. Modeling & Simulation AICMS 08, 2008, pp. 923–927.

[45] "ITU-T Home: Study groups : ITU-T Recommendations :ITU-T H.264 (05/2003) ". ITU. 2003-05-30.

[46] "ITU-T Home : Study groups : ITU-T Recommendations : ITU-T H.264 (03/2005) ". ITU.2005-03-01.

[47] Iain E. G. Richardson,"H.264 and MPEG-4 Video Compression. Video Coding for Next-generation Multimedia", Edition Wiley 2003, pp.152-222.

[48]M. Karczewicz and R. Kurceren, "A proposal for SP-frames", ITU-T SG16/6 document VCEG-L27,Eibsee, Germany, January 2001.

[49]M. Karczewicz and R. Kurceren, "The SP and SI Frames Design for H.264/AVC", IEEE Transactions on Circuits and Systems for Video Technology, 2003.

[50] "H.264/MPEG-4 Part 10 Transform & Quantification", H.264/MPEG-4 Part 10 White Paper, 20 Décembre 2002. www.vcodex.com.

[51] White paper, "Emerging H.264 Standard: Overview and TMS320DM642- Based Solutions for Real-Time Video Applications", 2002. www.ubvideo.com/public/h.26l_white-paper.pdf.

[52] S. W. Golomb, "Run-length encoding", *IEEE* Trans. on Inf. Theory, IT-12, pp. 399–401, 1966.

[53] G. Bjøntegaard and K. Lillevold, "Context-adaptive VLC coding of coefficients", JVT documentJVT-C028, Fairfax, May 2002.

[54] D. Marpe, G. Blattermann and T. Wiegand, "Adaptive codes for H.26L", ITU-T SG16/6 document VCEG-L13, Eibsee, Germany, January 2001.

[55] Gary J. Sullivan, Pankaj Topiwala, et Ajay Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", SPIE Conference on Applications of Digital Image Processing XXVII, Special Session on Advances in the New Emerging Standard: H.264/AVC, August, 2004

[56] M. Gallant, G. Cote, F. Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding", IEEE Trans. on Image Processing, Vol. 8, No. 12, pp. 1816-1823, Dec. 1999.

[57] T. Koga, K. Iinuma *et al.*, "Motion compensated interframe coding for video conference", Proc. NTC, November 1991.

[58] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," IEEE Trans., Circuits Syst. Video Technol., vol.13, no.7 pp. 614-619, July 2003.

[59] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans., Circuits Syst. Video Technol.,vol.13, no.7 pp.560-576, July 2003.

[60] K. Denolf, C. Blanch, G. Lafruit and J. Bormans, "Initial memory complexity analysis of the AVC codec," Signal Processing Systems, 2002 (SIPS '02), IEEE Workshop, pp.222-227, 16-18 Oct 2002.

[61] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," IEEE Trans. Circuits and Syst. Video Technol., vol.13, no.7, pp.704-716, July 2003.

[62] Hadamard, " Résolution d'une question relative aux déterminants", Bulletin des Sciences Mathématiques Séries 2, 17, Part I, 240–246, 1893.

[63] N. Ahmed, T. Nararajan, and K.R. Rao, " Discrete cosine transform", IEEE Transactions on Computers, 90–93, January 1974.

[64] A. Hallapuro, M. Karczewicz and H. Malvar, "Low Complexity Transform and Quantization – Part I: Basic Implementation", JVT document JVT-B038, Geneva, February 2002.

[65] ISO/IEC 14496-10 and ITU-T Rec. H.264, "Advanced Video Coding", 2003.

[66] H.264 Reference Software Version JM6.1d, http://bs.hhi.de/suehring/tml/, March 2003.

[67] Barbara Burke Hubbard, "See the World According to Wavelets", A.K. Peters, Wellesley, 1996.

[68] ITU-T Recommandation T.800.JPEG2000 Image Coding System-Part I, ITU Std, Jul 2002. http://www.itu.int/ITU-T

[69] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", 2nd edition, Prentice Hall, Upper Saddle River, NJ, 2002.

[70] S. Mallat, "A Wavelet Tour of Signal Processing", Academic Press, 1999.

[71] M. J. T. Smith and T. P. Barnwell III, "A procedure for designing exact reconstruction filter banks for tree structured sub-band coders," IEEE Int. Conf. Acoust., Speech Signal Proc., San Diego, CA, March 1984.

[72] M. J. T. Smith and T. P. Barnwell III, "Exact reconstruction for tree-structured subband coders," IEEE Trans. Acoust. Signal Speech Proc., 34 (3), 431–441, 1986.

[73] F. Mintzer, "Filters for distortion-free two-band multirate filter banks," IEEE Trans. Acoust. Signal Speech Proc., 33 (3), 626–630, 1985.

[74] P. P. Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect reconstruction techniques," IEEE ASSP Mag., 4 (3), 4–20, 1987.

[75] P. P. Vaidyanathan, "Multirate Systems and Filter Banks", Prentice Hall, Englewood Cliffs, NJ, 1993.

[76] T. Q. Nguyen and P. P. Vaidyanathan, "Two-channel perfect reconstruction FIR QMF structures which yield linear phase analysis and synthesis filters," IEEE Trans. Acoust. Signal Speech Proc., 37 (5), 676–690, 1989.

[77] Sweldens W., "The Lifting Scheme : A New Philosophy in Biorthogonal Wavelet Constructions", Wavelet Applications in Signal and Image Processing III, éditeurs A. F. Laine et M. Unser, pp. 68–79. Proc. SPIE 2569, 1995.

[78] Daubechies I. et Sweldens W., "Factoring Wavelet Transforms into Lifting Steps", J. Fourier Analysis and Applications, 4(3) :245–267, 1998.

[79] Calderbank R. C., Daubechies I., Sweldens, W. et Yeo B. L., "Wavelet Transforms that Map Integers to Integers", Applied and Computational Harmonic Analysis, 5(3) :332–369, 1998.

[80] Sweldens W., "The lifting scheme : A custom-design construction of biorthogonal wavelets", Appl. Comput. Harmon. Analysis, 3(2) :186–200, 1996.

[81] Hocine Bekkouche,"Synthèse de bancs de filtres adaptés, application à la compression des images", thèse de doctorat, Université Paris-Sud XI, juin 2007.

[82] Annabelle Gouze, "Schéma Lifting Quinconce pour la Compression d'Images", thèse de doctorat, université de Nice Sophia Antipolis, décembre 2002.

[83] J. Kovacevic and M. Vetterli. "Nonseparable multidimensional perfect reconstruction filter banks and wavelet bases for R^n". IEEE Trans. Inform. Theory, 38(2):533–555, March 1992.

[84] E.P. Simoncelli and E.H. Adelson." Non-separable extensions of quadrature mirror filters to multiple dimensions". Proceedings of the IEEE, 78:652–664, April 1990.

[85] Abdennour Boukaache, Noureddine Doghmane, "Hybrid discrete cosine transform–discrete wavelet transform for progressive image compression", Journal of Electronic Imaging 21(1), 013006 (Jan–Mar 2012).

[86] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6)"H.264/14496-10 AVC REFERENCE SOFTWARE MANUAL", Jan, 2009.

[87] Kim,C. and C.C.J.Kuo, " Feature-based intra-/inter coding mode selection for H.264/AVC ", IEEE Trans.Circuits Syst. Video Technol., 2:441-453, 2007.

[88] Byung-Gyu, K., "Fast selective-intra mode search algorithm based on adaptive thresholding scheme for H.264/AVC encoding", IEEE Trans.Circuits Syst. Video Technol., 18:127-133, 2008a.

[89] Byung-Gyu, K., "Novel inter-mode decision algorithm based on Macroblock (MB) tracking for the P-slice in H.264/AVC video coding", IEEE Trans.Circuits Syst. Video Technol., 18:273-279, 2008b.

[90] Uwe Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Third Edition, ISBN 978-3-540-72612-8 Springer Berlin Heidelberg New York, 2007

[91] Harize S, et al. "A methodology for implementing decimator FIR filters on FPGA". Int J Electron Commun (AEÜ), volume 67, Issue 12, December 2013,pp. 993-1004. http://dx.doi.org/10.1016/j.aeue.2013.05.013

[92] F. Gohzzi, "Optimisation d'une Bibliothèque de Modules Matériels de Traitement d'Images. Conception et test VHDL, Implémentation Sous Forme FPGA", Thèse, No.2789, Ecole doctorale de Sciences Physiques et de l'Ingénieur, Université de Bordeaux, 26 janvier 2004.

[93] "Les composants FPGA s'invitent dans le traitement du signal" , Patrick Méchin Directeur général de Techway, Mesures 775 - MAI 2005 - www.mesures.com

[94] A. Nkesta, "informatique industrielle, circuits logiques programmables, mémoires, PLD, CPLD et FPGA " Editions Ellipses, 1998.

[95] D.Smith " HDL Chip Design : A practical Guide for Designing, Synthesis & Simulating Asics & FPGAs using VHDL or Verilog", Doone Pubns . ISBN : 0965193438

[96] Project Veripage, retrieved from: http://www.angelfire.com/ca/verilog/history.html

[97] Xilinx. Synthesis and simulation design guide. Xilinx Inc, USA. (1998)

[98] Peter J. Ashenden, The VHDL Cookbook, First Edition, 1990,

[99] A. Croisier, D. J. Esteban, M. E. Levilion, and V. Rizo, "Digital filter for pcm encoded signals," U.S. Patent 3 777 130, Apr., 1973.

[100] A. Peled and B. Lie, "A new hardware realization of digital filters," IEEE Transactions on Acoust. Speech and signal processing, vol. 22, p.456-462, Dec. 1974.

[101] Meher, P.K., Chandrasekaran, S. and Amira, A.,"FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", IEEE Transactions on Signal Processing, Vol 56, Issue 7, July 2008, pp.3009 – 3017.

[102] Yajun Zhou, Ping zheng Shi, "Distributed Arithmetic for FIR Filter implementation on FPGA", IEEE International Conference on Multimedia Technology (ICMT), July 2011, pp. 294 – 297.

[103] M. Benouaret et al. , "Real time implementation of a signal denoising approach based on eight-bits DWT", Int. J. Electron. Commun. (AEÜ) Vol 66 (2012), pp. 937–943.

[104] K. A. Kotteri, S. Barua, A. E. Bell, and J. E. Carletta, "A Comparison of Hardware Implementations of the Biorthogonal 9/7 DWT: Convolution Versus Lifting", IEEE Transactions on Circuits and Systems-II: Express Briefs, Vol. 52, No. 5, May 2005, pp. 256 - 260.

[105] Gaurav Tewari, Santu Sardar, K. A. Babu, "High-Speed & Memory Efficient 2-D DWT on Xilinx Spartan3A DSP using scalable Polyphase Structure with DA for JPEG2000 Standard'', 3rd International Conference on Electronics, Computer Technology, IEEE 2011, pp. 138 – 142.

[106] Bellanger M., Bonnerot G. et Coudreuse M., "Digital filtering by polyphase network : application to sample rate alteration and filter banks", IEEE Trans.on Acoust. Speech and Signal Proc., 24(2) :109–114, avril 1976.

[107] Rui Guo and DeBrunner, L.S., "Two High-Performance Adaptive Filter Implementation Schemes Using Distributed Arithmetic", IEEE Transactions on Circuits and Systems II: Express Briefs , Vol 58, issue 9, Sept.2011, pp. 600-604.

[108] Shiann-Shiun Jeng, Hsing-Chen Lin, and Shu-Ming Chang, "FPGA implementation of FIR filter using M-bit parallel distributed arithmetic", IEEE International Symposium on Circuits and Systems (ISCAS '06), 2006, pp. 875 – 878.

[109] http://www.wolfsonmicro.com/products/codecs/WM8731/.website of Wolfson mircoeletronics.

[110] Bukhari K. Z., Kuzmanov G. K., Vassiliadis S.DCT and IDCT implementations on different FPGA technologies, Proceedings of the 13th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC'02). – Veldhoven, The Netherlands, 2002. – P. 232–235.

[111] Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R. On the Enrichment of Functional Delay Fault Tests , Information Technology and Control, 2009. – No. 38(3). pp. 208–216.

[112] Bareiša E., Bieliauskas P., Jusas V., Targamadzė A., Motiejūnas L., Šeinauskas R.Factor of Randomness in Functional Delay Test Generation for Full Scan Circuits, Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 9(105). – P. 39–42.

[113] P. Cassereau, A New Class of Optimal Unitary Transforms for Image Processing, Master's Thesis, Mass. Inst. Tech., Cambridge, MA, May 1985.

[114] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards –Including High Efficiency Video Coding (HEVC)", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1669-1684, December 2012