UNIVERSITÉ DE BRETAGNE OCCIDENTALE
École doctorale SICMA
Institut Supérieur de l'Electronique et du Numérique Brest- L@bISEN

UNIVERSITÉ DE SFAX
Ecole Doctorale Sciences et Technologies
Ecole Nationale d'Ingénieurs de Sfax-LETI

THÈSE DE DOCTORAT

Discipline : Génie Électrique

Présentée par :

Nihel NEJI

(Mastère Electronique)

Conception conjointe sur FPGA d'un décodeur HD H264/AVC utilisant un codage entropique de type CABAC

Soutenue le 17 Décembre 2015, devant le jury composé de :

M.	**Mohamed Ali BEN AYED**	**Professeur, ENET'COM Sfax**	**Président**
M.	**Denis HAMAD**	**Professeur, ULCO Lille**	**Rapporteur**
M.	**Mohamed ATRI**	**Maître de conférences, FSM Monastir**	**Rapporteur**
M.	**Christian BROSSEAU**	**Professeur, UBO Brest**	**Examinateur**
M.	**Maher JRIDI**	**Maitre de conférences, ISEN Brest**	**Encadrant**
M.	**Nouri MASMOUDI**	**Professeur à ENIS Sfax**	**Directeur de Thèse**
M.	**Ayman ALFALOU**	**Professeur à ISEN Brest**	**Directeur de Thèse**

Table des matières

Listes des figures

Listes des tableaux

Acronyme

AC: Alternating Current

AES: Advanced Encryption Standard

ASO: Arbitrary slice ordering

AVC: Advanced video coding

AVS: Audio Video Standard

BAC: Binary Arithmetic Coding

CAB : Codeur Arithmétique Binaire

CABAC: Context adaptive binary arithmetic coding

CAVLC : Context adaptive variable length coding

CCIR: Consultative Committee for International Radio

CDF: Cumulative distribution function

CTB: Coding Tree Block

CTU: Coding Tree Unit

DC: Direct Current

DCT: Discrete Cosine Transform

DES: Data Encryption Standard

DFT: Discrete Fourier Transform

DPCM: Differential pulse-code modulation

DSA: Digital Signature Algorithm

DVD: Digital versatile disc

EDK: Embedded Development Kit

FPGA: Field Programmable Gates Array

FFT: Fast Fourier Transform

FLC: Fixed Length Coding

FMO: Flexible Macroblock Ordering

FRExt: Fidelity Range Extensions

HD: High definition

HVS: Human visual system

IBM: International Business Machines Corporation

ICT: Integer Cosine Transform

IDR: Instantaneous Decoder Refresh

IEC: International Electrotechnical Commission

ITU-T: International Telecommunication Union – Telecommunication sector

IRCT: Inverse residual color transform

ISO: International Organisation for Standardization

JM: Joint model

JPEG: Joint Photographic Experts Group

JVT: Joint video team

KLT: Karhunen-Loève Transform

LFSR: Linear Feedback Shift Register

LSB: Last Significant Bit

MB: Macroblock

MBAFF: Macroblock adaptive frame/field coding

MC : Motion compensation

ME : Motion estimation

MPEG: Moving picture experts group

MSB: Most Significant Bit

MSE: Mean square error

MVD: Motion Vector Differential

NAL: Network Abstraction Layer

NALU: Network Abstraction Layer Unit

NIST: National Institute of Standards and Technology

NSA: National Security Agency

PCE: Peak to Correlation Energy

POF: Phase only Filter

PPS: Picture Parameter Set

PSNR: Peak signal to noise ratio

Q: Quantification

QP: Quantization parameter

RCT: Residual color transform

RLE: Run-Length Encoding

RGB: Red Green Blue

RAC: Randomised Arithmetic Coding

RBN : Randomised Binary Number

RBSP : Raw Byte Sequence Payload

RC: Rapport de Compression

RGB: Red, Green, Blue

RMSE: root-mean-square error

ROI: Region Of Interest

RSA: Ronald Rivest, Adi Shamir, Leonard Adleman.

SEI : Supplemental enhancement information

SI : tranches de commutation

SMPTE: Society of motion picture and television engineers

SPS: Sequence Parameter Set

SP: tranches de commutation

TNRG: True Random Number Generator

TV: Television

VCEG: Video coding experts grou

VLC: Variable Length Coding

VPC : Vecteur de Probabilité Cumulative

WMV: Windows Media Video

YUV /YCrCb: luminance, différence de chrominance rouge, différence de chrominance bleue

YUV : Luminance and chrominance components

Introduction

Contexte général :

En 2010, la télévision analogique vit ses dernières heures pour laisser la place à des diffusions en haute définition HD et à des émissions avec un contenu en 3D.Ce passage est rendu possible grâce à des efforts de recherche en matière de compression de la vidéo tout en garantissant une bonne qualité d'image. La norme de compression vidéo retenue en Europe est la norme H 264. D'un autre côté, la compression du contenu télévisuel a dépassé la station de télévision pour être embarquée dans les caméras numériques, dans les téléphones portables et dans les systèmes de télésurveillance. Aujourd'hui, l'implantation d'algorithme de traitement de signal et de l'image se fait de moins en moins dans des circuits spécifiques de type ASIC. Selon l'ITRS (International Technology Roadmap for Semiconductors), nous sommes passés d'une implantation de niveau circuit à une implantation au niveau système. L'élévation du niveau d'abstraction se fait pour une conception conjointe matérielle/logicielle. Les contraintes en termes de consommation d'énergie, de rapidité de traitement et de surface silicium requise pour ce codec sont alors plus sévères. D'ailleurs, au jour d'aujourd'hui, aucun système industriel ne permet la compression de la vidéo dans ce type d'environnement contraint. En effet, l'élément le plus contraignant dans la chaine de compression est le codeur entropique.

Objectifs

L'objectif de cette thèse est double. D'abord, il s'agit de proposer et de concevoir une architecture novatrice du codage de l'entropie de type CABAC utilisé dans la norme H 264. En effet, cette norme H 264 a été proposée comme nouveau standard par ITU-T et par ISO/IEC pour améliorer les performances des anciens standards comme MPEG-2. Le codeur H 264 et sa version optimisée H 264/AVC (Advanced Video Coding) peuvent offrir un des meilleurs taux de compression actuellement disponible sur le marché. Cependant, la mise en place d'un tel système risque d'augmenter à la fois la complexité de calcul et le coût d'une implantation physique. Plusieurs travaux de recherche se sont focalisés sur la recherche de la meilleure granularité lors de la répartition de l'image en macroblocs, sur l'estimation du mouvement et sur des techniques de prédiction des images futures en se basant sur des images références.

Une autre clé de la chaine de codage H 264/AVC se situe au niveau du codage de l'entropie. Les nouvelles directions pour le codage de l'entropie s'orientent vers un codage arithmétique. Ce type de codage utilise une ancienne technique qui consiste à allouer un nombre de bits important pour les codes à faible occurrence et inversement, un mot de code réduit pour les symboles à forte occurrence. De plus, le CABAC permet d'allouer un nombre de bits fractionnaire et adaptatif pour coder certains symboles d'un message. Par conséquent, l'efficacité du codage est améliorée. En sachant qu'une grande partie du coût d'implantation du codec H 264 vient du codage entropique CABAC, nous proposons d'implanter ce codage avec accélération sur des circuits configurables.

Pour y arriver, une étude bibliographique doit être bien faite sur les solutions d'implantation existantes du standard H 264/MPEG-4-part10. Il s'agit dans un premier temps de comprendre le fonctionnement du codeur CABAC et de le valider par des simulations fonctionnelles. L'étude et l'analyse des fonctionnalités identifiées du décodeur H 264 Haute Définition HD, qui limitent la performance temps réel de cette implantation doivent être poursuivis.

Un autre aspect de cette thèse consiste à intégrer le système sur une plate-forme de type Xilinx Virtex avec la définition de critères d'exploitation et d'évaluation permettant de mesurer les performances de notre système. Après, on doit valider le prototype, ainsi proposé, pour une application réelle de traitement d'image qui consiste à implanter un nouvel algorithme de compression et de cryptage d'images multiples par fusion spectrale à l'aide de méthodes itératives. Cette validation consiste à déterminer les limites physiques réelles sur un prototype FPGA (consommations, rapidité, surface utilisée, …). Puis, comparer l'algorithme proposé avec les standards du domaine pour les concepts es de compression et de cryptage.

Finalement, nous nous intéressons à effectuer le cryptage et la compression de façon simultanée et dépendante. En effet, depuis un certain nombre d'années, les besoins de communication nécessitant la compression et le cryptage n'ont cessé d'augmenter de manière exponentielle. Cela nécessite une course à l'optimisation (nouvel algorithme, consommation, ..) et à l'amélioration des normes actuelles. Afin d'avoir des systèmes robustes, le cryptage et la compression ne doivent plus être deux opérations consécutives car le cryptage suivi de la compression ou la compression suivie du cryptage ont montré leur limite.

Contribution

Ce sujet de thèse se situe dans la continuité des travaux de recherche de l'école nationale d'ingénieurs de Sfax (ENIS). En effet, les thèses soutenues à l'ENIS dans le domaine du codage de la vidéo proposent comme perspective l'étude approfondie du codeur CABAC en vue de son implantation sur cible programmable. Notre thèse est la première cotutelle entre l'université de Sfax (ENIS) - Sfax et l'université de Bretagne occidentale (ISEN) – Brest. Ça été l'occasion de travailler ensemble sur le codage dit CABAC et de découvrir nos recherches respectives. Cette coopération franco-tunisienne est démarrée en Janvier 2011 est intitulée « La conception conjointe sur FPGA d'un décodeur HD H264/AVC à base d'un codage entropique de type CABAC pour des applications de traitement d'images » au sein de deux laboratoires : Laboratoire d'Electronique et technique d'iInformation (LETI) de et le Laboratoire L@bISEN (Institut supérieur d'électroniques numériques) (projet IFC).

L'objectif de cette recherche de doctorat est l'implantation du codage direct et inverse CABAC/ CABAD comme un accélérateur sur des circuits configurables car une grande partie du coût d'implantation physique vient de ce bloc. Aussitôt, on envisage effectuer le cryptage et la compression de façon simultanée et dépendante pour bien assurer la sécurité de compression et transmission des données traitées.

Organisation du mémoire

Ce mémoire est organisé en quatre chapitres.

Le premier chapitre expose le contexte de ce travail. En effet, nous étudions dans une première phase la théorie de compression ainsi que les techniques de la réduction des redondances pour ces différents types spectrales, temporelle, spatiale et statique. Nous nous intéressons plus particulièrement au dernier type qui se base sur les codages à base statistique et arithmétique. Ensuite, dans une deuxième phase, nous nous focalisons sur l'application JPEG et nous détaillons les différentes phases de traitement d'images employés. Plus particulièrement, nous nous intéressons au codage entropique et au fonctionnement avec un codeur entropique à longueur variable et arithmétique. Nous soulignons le fait que l'adaptabilité du codeur à l'entrée a de grande influence sur le taux de compression

Le deuxième chapitre de ce mémoire est dédié à l'étude et l'évaluation de dernières normes de codage vidéo normalisé : H264 et HEVC. D'abord, nous détaillons les techniques de codage vidéo normalisé : prédiction, transformation, quantification et codage entropique. Ensuite, nous présentons les spécificités des deux normes et leur propre technique. Puis, nous présentons leurs résultats d'évaluation software en comparant leurs performances (PSNR, Bitrate). Nous terminons ce chapitre par une étude bibliographique de la complexité au niveau décodeur vidéo des différents sous bloc de la chaine de décodage.

Le troisième chapitre de cette thèse décrit les résultats des synthèses et mesures de puissance réalisés lors de l'implantation sur FPGA Xilinx Virtex des accélérateurs matériels de codage entropique en utilisant les IP CABAC et CABAD. Dans la première partie, nous étudions en détails les phases de codage arithmétique binaire à contexte adaptatif : binarisation, modélisation du contexte et codage arithmétique binaire proprement dit. Nous mettons l'accent sur l'avantage de ce codeur par rapport aux autres codeurs vidéo (CAVLC). Nous validons le fonctionnement des différents blocs du CABAC proposé en détaillant les conceptions réalisés et les rapports de surface et consommation de puissance mesurée.

Nous terminons avec un dernier chapitre présentant une implantation software d'une technique de compression et cryptage simultanés. Dans un premier temps, nous décrivons les techniques de cryptage pour une image fixe et animée : DEC, AES, RSA, …. Ensuite, nous proposons un schéma innovatif qui assure un cryptage conjoint à la compression et nous montrons les performances et les limites du codeur vidéo modifié générant un flux de données protégés.

Finalement, une conclusion vient clôturer ce document dans le but de résumer les résultats obtenus pendant cette thèse et de présenter les perspectives de nos travaux futures

Chapitre 1 : Codage entropique d'une source

1. Introduction

La compression est la technique qui consiste à représenter la même, ou presque la même information avec un minimum de données. Cette méthode a pris ses origines dans le domaine optique. L'équipe VISION du LabISEN a développé un dispositif tout optique pour la compression des images [Alfalou 09]. Bien que les méthodes optiques présentent un intérêt incontestable au niveau de la rapidité des traitements, il est vrai que la résolution des images compressées, la configurabilité et la portabilité des dispositifs optiques restent des inconvénients majeurs. Aujourd'hui, pour les applications industrielles, la compression optique a laissé sa place aux méthodes numériques de compression.

Dans ce chapitre, nous nous sommes intéressés dans un premier temps par décrire par un certain nombre de techniques (unique) utilisés dans la compression. Les algorithmes de réduction de redondances spectrales, temporelles et spatiales sont détaillés. Ensuite, nous mettons l'accent sur les techniques de réduction des redondances statistiques où nous détaillons le principe des deux plus grandes familles des codeurs entropiques à savoir le codeur de Huffman et le codeur arithmétique. Le chapitre se termine par des propositions d'améliorations de ces codeurs entropiques et par des analyses de performance d'un codec JPEG pour des applications de compression d'images.

2. Théorie de la compression

La majorité des méthodes de compression numérique cherche à exploiter les redondances dans les informations à compresser. En effet, la suppression de la redondance d'information contribue à la réduction de la quantité d'information à stoker ou à envoyer. Tous les algorithmes de compression suivent une approche mathématique basée sur le calcul de la quantité d'information H_i propre associée à chaque symbole A_i fournie par la connaissance de la probabilité d'émission de chaque symbole $p(A_i)$ [Rioul 07].

$$H_i = \log_2\left(\frac{1}{p(A_i)}\right) = -\log_2(p(A_i)) \tag{1.1}$$

L'entropie d'une source est alors définie par :

$$H(S) = -\sum_{i=1}^{n} p(A_i)\log_2(p(A_i)) \tag{1.2}$$

Avec n est le nombre des symboles. Ceci introduit la notion d'entropie définie par la quantité moyenne d'information contenue dans une source [Say 05].

En effet, elle présente le lien entre l'information fournie et la distribution de probabilité [Ash 90]. Si un tel symbole est plus probable que d'autres, la quantité d'information correspondante est moins grande [Battail 97]. Dans la base binaire (base 2), l'entropie s'exprime en bit par symbole. Plus l'entropie de la source est grande, plus il y a d'incertitude et donc d'informations liées à la source. Dans cette définition, il est supposé que tous les symboles ont une probabilité d'émission non nulle. On montre que l'entropie est maximale et vaut $\log_2(n)$ lorsque tous ses n symboles sont équiprobables. Cette théorie de l'information définie par Shannon [Shannon 48] détermine les limites de performance des systèmes numériques. En outre, elle anticipe les besoins pratiques pour des applications de télécommunications, par exemple [Reza 94]. Ainsi, cette théorie s'est attachée à coder les flux nécessaires aux transferts des données, ce qui sert à optimiser l'utilisation des voies de communication en recherchant les codages les plus adaptés.

Alors que ce théorème de Shannon permet de trouver le code s'approchant indéfiniment de l'entropie, Hamming et Golay [Ham 86] ont développé le premier algorithme permettant la vérification des erreurs, ce qui a donné naissance à la théorie du codage. Autrement, les caractéristiques d'entropie ont attribué des codages plus performants et plusieurs travaux ont développé des codages statistiques fondés sur la fréquence d'apparition d'une information ou d'un symbol. Ainsi, les méthodes et les techniques utilisées dans la compression sont très diversifiées ; et elles cherchent toutes à représenter toutes les combinaisons théoriques. Chaque méthode dépend principalement du type de données à compresser. Par exemple, on ne compressera pas de la même façon une image qu'un fichier audio [Wagner 93].

Nous pouvons citer :

- La compression physique : elle agit directement sur les données et supprime la redondance de données sur un fichier. Pour ce faire, nous utilisons un algorithme capable de comprimer des données pour un minimum de place.
- La compression logique : elle est effectuée par un raisonnement logique en substituant une information par une information équivalente mais plus petite [Wagner 93]. Elle remplace une lettre, un chiffre ou un nombre par un autre. Pour ce fait, on utilise un algorithme capable de recoder les données dans une représentation différente plus compacte contenant la même information.
- La compression symétrique : la même méthode est utilisée pour compresser et décompresser l'information. Ce type de compression est utilisé dans les transmissions de données [Wagner 93].

- La compression asymétrique : elle requiert plus de travail pour l'une des deux opérations, le plus souvent la compression qui est plus lente que la décompression.
- La compression sans perte : elle réduit la redondance tout en tenant compte ou pas de la corrélation entre les symboles. La décompression permet de retrouver exactement le message originel [Guillois 96].
- La compression avec pertes : c'est une méthode de compression irréversible qui élimine quelques informations pour avoir le meilleur taux de compression possible, tout en gardant un résultat qui soit le plus proche possible des données originales [Guillois 96].

Malgré la diversification des types et méthodes de compression, elles se basent toutes sur les processus de réduction de l'information redondante. Plusieurs types de redondance peuvent exister dans les images et les vidéos. Nous détaillons dans la section suivante les redondances spectrales, spatiales, statistiques et temporelles.

3. Techniques de la réduction des redondances

La redondance désigne l'occurrence multiple d'une donnée qui représente la même information. Toute séquence vidéo contient une très grande redondance spectrale, temporelle, spatiale et statistique. Pour une compression efficace, il est possible de retirer la redondance pour avoir une quantité réduite d'informations. Autrement dit, on néglige les informations redondantes et on met en évidence les passages importants [Murât 78].

3.1. Réduction de la redondance spectrale

Une image en couleur est décrite par un contenu spectral de trois bandes de couleurs fondamentales RGB (Red, Green, Blue) dont les informations portées sont d'égale importance. En fait, ce repère décompose le pixel en trois couleurs de base : rouge, vert et bleu en utilisant 8 ou 10 bits par pixel pour présenter chaque couleur. Leurs associations permettent de parcourir toutes les couleurs. Nous sommes donc obligés de conserver la même résolution pour chaque bande, ce qui nécessite un grand espace de stockage.

La plupart des normes de décompression définissent leurs algorithmes suite à un changement de l'espace trichromatique RGB à l'espace YUV ou YCrCb (luminance, différence de chrominance rouge, différence de chrominance bleue) par une transformation linéaire irréversible. L'espace YUV décompose le pixel de l'image selon trois composantes : la luminance qui varie le niveau de gris du pixel, les deux chrominances bleu et rouge qui permettent d'ajouter la couleur au pixel [Brunel 04]. La décomposition en luminance et chrominance du flux vidéo, spécifiée par le standard CCIR 601 (Consultative Committee for International Radio) [Westwater 97], est donnée par cette équation :

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ C_B &= -0.169R - 0.331G + 0.5B \\ C_R &= 0.5R - 0.419G - 0.081B \end{aligned} \tag{1.3}$$

Cet espace permet une décorrélation spectrale par la séparation du signal de luminance des chrominances [Westwater 97]. Étant donné que la différence d'intensité lumineuse est mieux perçue par l'œil humain, ce changement vers l'espace YUV ne peut qu'améliorer la qualité visuelle du codage en conservant les détails les plus significatifs dans la luminance.

En effet, le sous échantillonnage des signaux U, V améliore les performances de la compression en profitant de la sensibilité psychovisuelle. Autrement dit, la corrélation entre les pixels qui se trouvent dans la même position dans les différentes bandes constituant l'image engendre une redondance spectrale importante qui peut être supprimée par transformation de l'image de l'espace primaire RGB à l'espace secondaire YUV.

Ainsi, plusieurs formats de sous échantillonnage existent. Les plus connus sont les formats :

- 4:4:4 dans lequel chaque pixel est formé par un composant de luminance, une de chrominance rouge et une de chrominance bleue. Dans ce cas, il n'y a pas de compression entre le format RGB et YUV
- 4:2:2 dans le quel une composante de chrominance rouge et une composante de chrominance bleue sont utilisées une fois sur deux lors du parcours de l'image par colonnes. Ainsi, l'image en formation YUV présente une taille égale à 3/4 de la taille de l'image en format RGB.
- 4:2:0 dans le quel une composante de chrominance rouge et une composante de chrominance bleue sont utilisées une fois sur deux lors du parcours de l'image par colonnes et une fois sur deux lors du parcours de l'image par lignes. Dans ce cas, l'image en formation YUV présente une taille égale à la moitié de la taille de l'image en format RGB

Le schéma de la figure 1.1 illustre cette répartition [Jridi 14].

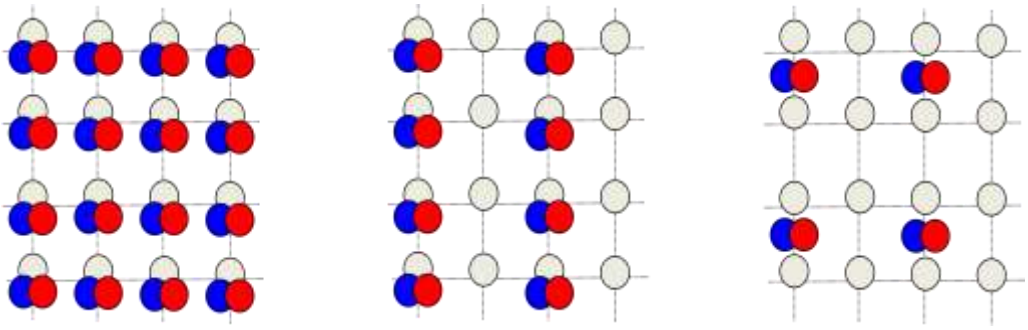


Figure 1. 1 - Représentation du format YUV

(à gauche format 4:4:4, au centre format 4:2:2 et à droite format 4:2:0)

3.2. Réduction de la redondance temporelle

Dans une séquence vidéo, deux images proches se ressemblent aux déplacements près ; ils sont presque identiques. Alors, nous pouvons supposer que la différence entre une image et la suivante est bien faible. Autrement dit, la position d'un bloc de pixels ne varie pas beaucoup d'une image à une autre. On peut se limiter au codage de ce qui est modifié d'une image à une autre dans le but de limiter la quantité d'informations à traiter. C'est le codage Inter frame nommé aussi Inter. Le principe de la redondance temporelle est illustré dans la figure 1.2.

(a) Image I à t_0 (b) Image J à $t_0+\Delta t$ (c) Image de différence

Figure 1. 2 - L'exploitation de la redondance temporelle

Puisque la camera n'a pas changé ni de plan ni de position, on remarque qu'il n'y a pas une grande différence entre ces deux images. En effet, la différence (figure 1.2 (c)) représente l'information résiduelle : elle nous décrit les parties qui ont subi un mouvement entre les deux images I et J successives (figure 1.2 (a) et figure 1.2 (b)) ce qui nous montre clairement les redondances temporelles. L'utilisation des redondances temporelles dans le flux vidéo contribue de façon très importante au gain de compression obtenue. Les techniques utilisées dans ces redondances seront détaillées dans le chapitre 2.

3.3. Réduction de la redondance spatiale

L'ensemble des techniques mises en œuvre pour réduire la redondance spatiale exploite le principe de la corrélation dans une image, indépendamment des autres images. La corrélation en question est dans les zones uniformes plus ou moins grandes dans lesquelles les pixels ont des valeurs très voisines. Nous pouvons diminuer cette redondance en codant chaque image séparément ; il s'agit du codage Intra. Ainsi, on suppose que l'importance d'un pixel particulier de l'image peut être prévue à partir des pixels voisins.

Les algorithmes de compression d'images et de la vidéo utilisent une transformation appelée DCT (Discrete Cosine Transform) ou TCD (Transformée en Cosinus Discrète), sur des blocs de pixels de tailles fixes ou variables allant de 4x4 jusqu'à 32x32 en passant par 8x8 et 16x16, pour analyser efficacement les corrélations spatiales entre pixels voisins de la même image. Cette transformation est toujours suivie d'une opération de quantification pour réduire le

nombre de bits sur lequel les données restantes sont représentées. Enfin, dans la littérature d'autres méthodes ont été proposées, en utilisant les fractales, les ondelettes, ou même le matching poursuit. Cependant, si la corrélation entre pixels dans des trames voisines est grande, c'est-à-dire, dans les cas où deux trames consécutives ont un contenu semblable ou identique, le codeur utilisera le codage Inter qui sera suivi par une transformation.

3.3.1.Transformation en cosinus discrète DCT

Dans les codeurs vidéo, un codage par transformée est appliqué à l'image pour extraire les redondances spatiales. Cette image peut venir d'une trame Intra ou d'une information résiduelle (différence entre trame Intra et image reconstruite après estimation de mouvement). Autrement, les résidus de prédiction et/ou les pixels des trames Intra sont passés dans le domaine fréquentiel par l'étape de transformée pour enlever d'avantage les redondances.

La DCT est employée dans la grande majorité des normes courantes de codage vidéo. Elle a été intégrée dans des standards MPEG1, MPEG2, H261 et H263 [Haskell 97]. La transformée DCT figure aussi dans les nouveaux codeurs vidéo de type H264 et HEVC, moyennant quelques modifications. Cette transformée permet de séparer les hautes fréquences des basses fréquences. Ainsi, le premier coefficient de la matrice transformée représente le coefficient continu DC ou le fondamental du bloc de l'image. Il s'agit du coefficient de la fréquence nulle dont l'amplitude est la plus élevée. Le reste des coefficients sont des coefficients AC ; leurs intensités diminuent à chaque fois qu'on tend vers les hautes fréquences. Le fait de séparer les fréquences hautes et les fréquences basses va ainsi permettre la décorrélation des pixels de chaque sous bloc de l'image et ainsi réduire la redondance spatiales dans l'image.

Tout l'intérêt de la DCT réside dans ses capacités de décorrélation quasi optimales s'approchant ainsi de la transformation KLT (Karhunen-Loève Transform)[1] [Britanak 07].

L'application de la DCT permet ainsi de séparer les basses fréquences des hautes fréquences pour faire la compression ; la décompression est quant à elle réalisée par application de la transformation inverse : IDCT (Inverse DCT). D'un point de vue mathématique, la transformation DCT et la transformation IDCT sont calculées par les équations (1.4) et (1.5).

$$F(u,v) = \frac{2}{N} \times c(u) \times c(v) \times \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathrm{Im}\, g(x,y) \times \cos\left[\frac{\pi}{N} u(x+\frac{1}{2})\right] \times \cos\left[\frac{\pi}{N} v(y+\frac{1}{2})\right] \qquad (1.4)$$

$$\mathrm{Im}\, g(x,y) = \frac{2}{N} \times \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u) \times c(v) \times F(u,v) \times \cos\left[\frac{\pi}{N} u(x+\frac{1}{2})\right] \times \cos\left[\frac{\pi}{N} v(y+\frac{1}{2})\right] \qquad (1.5)$$

[1] La transformation KLT est la transformation linéaire réversible qui permet du supprimer les redondances en décorrélant les données. Elle minimise la bande théorique du débit donné par l'entropie du signal à compresser. Cependant, elle est très peu utilisée en raison de la complexité de calcul qu'elle peut engendrer.

Avec $\left\{ \begin{matrix} C(0) = (2)^{-\frac{1}{2}} \\ c(w) = 1 \quad pour \quad w = 1,2,...,N-1 \end{matrix} \right\}$

Où $F(u,v)$ représente les coefficients DCT, $Img(x,y)$ représente les valeurs de pixel dans la position x, y dans l'image et N représente la taille de l'image ou du block à transformer.

Au final, cette transformation présente une localisation fréquentielle efficace, une compaction de l'énergie qui surpasse celle obtenue avec la DFT, elle est facile d'implémentation et a notamment l'avantage de générer un signal transformé réel, contrairement à la DFT qui génère un signal transformé complexe.

3.3.2. Transformation DCT pour les nouveaux standards vidéo

La norme H.264/AVC [Draft 14] fait recours à une transformée en cosinus discrète entière, appelée ICT (Integer Cosine Transform) semblable à la DCT [Richardson 03]. Cette transformée est définie pour la luminance pour des blocs 4x4 et pour la chrominance pour des blocs 8x8 [Draft 14]. Elle opère sur ces blocs de 4x4 de suite au processus de prédiction et elle distingue deux ensembles de coefficients pour chaque bloc : Le coefficient DC et les quinze autres coefficients AC. Cette transformée, comme la DCT, permet la concentration de l'intensité vers les coefficients de plus faibles fréquences et en premier lieu le coefficient DC de la position zéro. L'équation de l'ICT appliquée à une matrice E de taille 4x4 est donnée par l'équation(1.6) [Draft 14] :

$$F = AEA^T = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} (E) \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix} \qquad (1.6)$$

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

Cette transformée élimine complètement les disparités associées à la transformée inverse qui occasionnaient une disparité du codec dans les normes précédentes. Mais l'intérêt majeur de cette transformation est la réduction de la complexité de calcul [Malvar 03]. En effet, cette transformée, mises à l'échelle, ne manipule que des entiers et n'utilise que des additions et des décalages, devenant ainsi plus flexible pour une éventuelle implantation. Une transformée « Hadamard » [Fournier 72] est aussi appliquée sur les coefficients DC des blocs 4x4 transformés. Il faut noter que pour la norme H264, la taille de l'ICT peut atteindre 16x16. Les matrices de l'ICT de taille 16x16, sont calculées à bases de la matrice de l'ICT de taille 4x4 donnée par l'équation 1.6. L'utilisation de tailles de matrices de plus en plus importantes permet d'avoir une meilleure efficacité de codage c.-à-d., un bon compromis entre le rapport

de compression et la qualité de l'image reconstruite. Cependant, la norme H264, ne définit pas de matrices de taille 32x32 ce qui peut paraitre contradictoire avec ce que nous sommes en train d'avancer. En effet, le calcul des matrices de taille 32x32 à base de matrices 4x4 pour la norme H264 n'est pas possible puisque les propriétés d'orthogonalité de la matrice et d'inversion ne sont plus assurées pour une taille de 32x32. L'inversion de la matrice de l'ICT est une propriété intéressante puisqu'elle permet de calculer l'ICT et l'IICT par les mêmes matrices. Pour remédier à ce problème, la nouvelle norme HEVC a opté pour un changement de la matrice définie dans l'équation 1.6. Ainsi, le calcul des coefficients transformés par une ICT de taille 4x4 définie par la norme HEVC est donné par l'équation 1.6, mais avec un changement dans la matrice A, conformément à l'équation suivante :

$$A = \begin{pmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{pmatrix} \tag{1.7}$$

3.4. Réduction de la redondance statistique

Les données encodées sont relativement proches et les séquences vidéo contiennent une très grande redondance statistique. Cette redondance est définie par le caractère statistique d'émission des symboles par la source. Alors, pour tout symbole dont la fréquence d'apparition est grande, on l'attribut des mots codes plus courts. Et on associe des mots codes plus longs pour les symboles qui se répètent rarement. Cette opération de codage à caractère statistique est appelée codage entropique à longueur variable. Dans les sous sections suivantes nous allons détailler quelques types de ce codage.

3.4.1. Codages à base statistique

3.4.1.1. Le codage de Shannon

Claude Shannon a démontré à la fin des années 40 l'existence d'une méthode permettant de compacter les données sans rien perdre de leur signification [Shannon 48] [Shannon 49]. En effet, on peut calculer la fréquence probable d'apparition des informations ce qui nous permet d'éliminer les redondances et de les encoder d'une manière compacte. Autrement, toute fréquence est représentée par un certain nombre de bits. Tout caractère codé sur 8 bits (ASCII) est décrit par son entropie et par sa fréquence d'apparition. Si le symbole est fréquent, son entropie est élevée et il nécessite moins d'informations pour s'identifier. Donc, au lieu d'allouer 8 bits à ce symbole, on ne lui réserve que le nombre de bits nécessaires à sa reconnaissance.

3.4.1.2. Le codage de Shannon fano

Le codage de Shannon-Fano [Salomon 07] est développé à partir des propriétés statistiques des messages. Cette méthode se base sur le partage des symboles en deux groupes de valeur quasi équivalente. Pour chaque groupe, on attribue la somme des probabilités d'apparition des symboles qu'il contient. On fait un choix arbitraire pour désigner un des groupes par « 0 » et l'autre par« 1 ». De même, ces groupes sont divisés en deux et indiqués par 0 ou 1 jusqu'au avoir un seul symbole dans chaque groupe.

Cette méthode forme alors un arbre binaire constitué de branches (bits d'informations 0 ou 1) et de feuilles (caractères). Le code d'un caractère est déterminé en partant du sommet et en parcourant les branches jusqu'à la feuille correspondante. Les caractères près du sommet sont alors les plus fréquents et nécessitent moins de bits de codage.

3.4.1.3. Le codage de Huffman

L'algorithme de codage de Huffman est un algorithme de codage par bloc, c'est-à-dire que les symboles (ou groupes de symboles) sont représentés à l'aide d'un code binaire à longueur variable [Miroslva 01]. Nous parlons alors des techniques VLC (Variable Length Coding) par opposition aux techniques classiques de binarisation par FLC (Fixed Length Coding). Le principe consiste à trouver le jeu de code optimal afin de minimiser le coût de codage moyen par symbole.

Ainsi, si l'on considère un message M constitué de symboles de l'alphabet $A = \{\lambda_1,\ldots,\lambda_n\}$ ayant les probabilités d'observations $\{P(\lambda_1), \ldots, P(\lambda_n)\}$, il faut trouver un jeu de code préfixé $\{c_1,\ldots, c_n\}$ de longueurs respectives $\{l_1,\ldots, l_n\}$ minimisant le cout de codage moyen donnée par la formule (1.8) :

$$\bar{C} = \sum_{k=1}^{n} P(\lambda_k) \times l_k \qquad (1.8)$$

Huffman définit un algorithme simple de construction d'un tel jeu de code à l'aide d'un arbre binaire équilibré par rapport aux poids $P(\lambda_k)$. Ce code est alors construit par une approche ascendante. Des étiquettes (0 ou 1) associées aux transitions de l'arbre de codage peuvent être inversées.

Les codes de Huffman sont très largement utilisés en pratique pour des alphabets de petite taille, car les encodeurs et les décodeurs qui leur sont associés sont très efficaces [Tjalkens 00]. La complexité est importante et il n'est pas possible de les utiliser sur des séquences de grande taille. Il est donc nécessaire de stocker ou de calculer l'arbre de codage pour toutes les longueurs de séquence possibles.

En pratique, ceci n'est pas réalisable car le nombre d'éléments d'un alphabet croît de manière exponnetielle avec la longueur de la séquence (le coût de calcul est également prohibitif, les codes trop gros et trop complexes à générer). Les codes de Huffman sont donc principalement utilisés sur l'alphabet de base de la source et non sur un alphabet de vecteurs d'éléments de la source. Les performances obtenues demeurent très correctes lorsque la taille de l'alphabet est importante.

Du fait que les codes de Huffman sont à longueur entière, alors que la longueur optimale de chaque mot de code est égale à $(- \log_2 P(\lambda_k))$ et par conséquent a priori irrationnelle, cette technique ne permet pas d'atteindre systématiquement l'entropie de la source lorsque la longueur de la séquence tend vers l'infini. Toutefois, ce code peut s'avérer optimal si les différentes probabilités sont des puissances entières de ½ [Cover 06].

3.4.1.4. Le codage de Huffman adaptatif

Ce code permet de compresser un flot à la volée en faisant une seule lecture de l'entrée. La table des fréquences est élaborée au fur et à mesure de la lecture du fichier, l'arbre de Huffman est modifié à chaque fois que l'on lit un nouveau caractère. Autrement, pour construire l'arbre de Huffman et le mettre à jour, on compte le nombre d'occurrences de chaque caractère et le nombre de caractère déjà lu. On a donc la fréquence de chaque caractère à chaque traitement. On incrémente de 1 le nombre d'occurrences du caractère codé. L'arbre de Huffman-adaptatif est mis à jour à chaque itération et n'a pas besoin d'être envoyé au décodeur qui calcule un arbre exact au codeur. Cette méthode dynamique est moins efficace que la méthode statique parce qu'on ne considère pas la valeur exacte de l'occurrence du fait qu'elle est adaptée. Cependant, cette méthode présente l'avantage d'éviter le stockage de l'arbre et permet aussi de réduire le temps de codage puisqu'il n'est plus nécessaire de parcourir tout le message avant de le coder.

3.4.1.5. Le codage RLE (Run-length Encoding)

Le codage RLE est le codage des répétions. Il est simple et très utilisé. Son algorithme de consiste à identifier et à supprimer les redondances d'informations en les codant sous une forme plus compacte. Autrement, on diminue la taille d'une répétition de chaîne de caractères (passage ou run) à 2 octets. Le premier octet représente le nombre de caractères dans le passage et sa valeur est entre 0 et 255. Le second octet représente la valeur du caractère répété. Le RLE est utilisé pour la plupart des formats de fichiers bitmaps comme le TIFF, le BMP et le PCX.

Par rapport aux autres algorithmes de compression, le RLE ne peut pas atteindre de forts taux de compression. En contrepartie, il est efficace, simple à adapter et rapide en exécution. Ce codage est utile pour des images noir et blanc. Ces algorithmes sans perte assurent un taux de compression en moyenne de l'ordre de 40% pour des données de type texte qui est trop insuffisant pour les données de type multimédia. Pour résoudre ce problème, il faut donc utiliser la compression avec perte. En effet, cette compression permet d'éliminer quelques informations pour avoir le meilleur taux de compression possible tout en gardant un résultat proche aux données originales, ce qui la rend irréversible.

Cette méthode de compression est basée sur le fait que les signaux vidéo contiennent une part importante de données redondantes que l'œil ne peut pas percevoir. Alors, l'objectif de la compression avec pertes est d'éliminer les données redondantes et non pertinentes. Cela engendre une dégradation faible et non visible à l'œil mais taux de compression élevé.

Il existe plusieurs algorithmes de compression avec perte comme Audio MPEG pour la compression du son, JPEG pour des applications de compression des images fixes et H.264 pour la compression des images animées.

3.4.2. Le codage arithmétique

3.4.2.1. Principe de base

Les codeurs arithmétiques ont la capacité de se rapprocher de l'entropie avec une complexité linéaire ; ceci a contribué à leur succès. L'idée directrice de ces codes est assez ancienne. Ces codes sont des codes-bloc vers variable. Ces derniers sont asymptotiquement optimaux lorsque la taille du bloc tend vers l'infini. Le codage arithmétique introduit par Rissanen [Rissanen 76] se différencie des techniques de codage par blocs dans le sens où il n'attribue pas un code à chaque symbole (ou groupe de symbole), mais au message tout entier, ce qui permet de représenter chaque symbole du message sur la fraction adéquate de bits.

Le principe général du codage arithmétique peut être résumé comme suit :

- Pour tout message $S = s_1 s_2 ... s_k$, nous construisons d'une manière récursive l'intervalle $[B_{inf}(S), B_{sup}(S)[$, de longueur P(S), inclus dans [0,1[, de sorte que l'ensemble des intervalles associés à tous les messages s_i constituent une partition de [0, 1[.
- Le codeur arithmétique attribue alors au message S le code binaire c(S) le plus court possible définissant un rationnel binaire permettant d'identifier l'intervalle correspondant au message. Si nous notons L la longueur du code c(S) :

$$\mathrm{Binf}_s^k \leq c(S) < \mathrm{Bsup}_s^k \qquad (1.9)$$

$$\mathrm{Binf}_s^k \leq c(S) + \left(\frac{1}{2}\right)^L < \mathrm{Bsup}_s^k \quad (1.10)$$

L'inégalité (1.10) assure que tout code binaire commençant par c(S) aboutit à un décodage du message S. Á partir de ces inégalités, nous pouvons alors s'apercevoir que pour des messages suffisamment longs, le codeur arithmétique est quasiment optimal puisque l'on peut toujours trouver un code binaire de longueur $(\lceil -\log_2(P(S)) \rceil + 2)$ convenable.

Pour la mise en œuvre de la technique de la construction récursive du code d'un message, nous pourrons se référer à [Witten 87]. La méthode consiste à travailler avec un intervalle du type $[B_{inf}, B_{sup}[$ représentant l'intervalle de probabilité du message constitué des symboles déjà codés/décodés et dont les bornes évoluent de manière similaire à l'encodage et au décodage. Dans la pratique, ces bornes sont calculées sous un format entier (16 ou 32 bits). Pour des séquences binaires suffisamment longues, et dans la mesure où les probabilités sont suffisamment équilibrées au regard du format d'entier utilisé, nous pouvons donc comprimer des informations de manière optimale.

Il existe aussi des implémentations de codec arithmétique sous forme d'automates à état, ne recourant pas à des multiplications réelles pour les calculs de probabilités mais uniquement à des additions, décalages et accès à des tables prédéfinies [Boisson 05].

Ces outils sont très prisés en raison de leur complexité limitée. De nombreuses versions en précision finie sont très utilisées en pratique, car ce codage :

- permet d'approcher l'entropie de la source avec des complexités d'encodage et de décodage qui sont linéaires par rapport à la longueur de la séquence.
- est très flexible dans le sens où la probabilité peut être fournie pour chaque symbole, ce qui permet d'introduire des modèles adaptatifs.
- en précision finie, permet un décodage quasi-instantané, c'est à dire que le nombre de bits à lire avant le décodage d'un symbole est borné par une constante de taille réduite.

Dans le domaine de la compression vidéo, le passage des techniques de type Huffman à des techniques de type arithmétique est assez récent, et date de la norme MPEG-4 au cours de l'année 2003.

Dans ce cadre, l'emploi du codage arithmétique à la place des codes VLC (Varialble Length Coding) a permis d'obtenir en moyenne un gain net en débit de 15% [Boisson 05]. Enfin, le codage arithmétique présente un avantage d'efficacité de compression par rapport au codeur VLC puisqu'un nombre fractionnaire de bits peut être associé au codage d'un symbole.

3.4.2.2. Algorithme de codage

Nous présentons dans cette partie l'algorithme de codage arithmétique (Algorithme 1.1) et nous citons un exemple à sa suite. Comme nous avons expliqué dans la présentation de ce codage, nous utilisons des subdivisions d'intervalle comprises entre 0 et 1 pour représenter les symboles à coder. Ceci dépend des probabilités cumulées de tous les symboles. En effet, nous attribuons à chaque symbole un sous-intervalle inclus dans [0,1] selon son occurrence dans le message.

Par exemple, un symbole X de probabilité 0,2 qui est défini dans la subdivision [0,6, 0,8] peut prendre une valeur de 0,6 à 0,799. Plus le message à coder ne sera long, l'intervalle qui lui représente devient de plus en plus petit. Par conséquent, le nombre de bits utilisé pour représenter cet intervalle augmente.

Algorithme 1.1 : Algorithme de codage arithmétique

Entrées: Symbole : entrée à coder
b_{inf}= 0 ;// initialisation de la borne inférieure de l'intervalle courant
b_{sup}= 1 ;// initialisation de la borne supérieure de l'intervalle courant
Tant que (il y a des symboles à coder) faire
C = symbole à coder ;
Taille= b_{sup}-b_{inf} ;// Calcul de la longueur de l'intervalle courant
b_{sup}= b_{inf} + Taille b_{sup} (C);// mise à jour de la bornesupérieure de l'intervalle courant*
b_{inf}= b_{inf} + Taille b_{inf} (C);//mise à jour de la borne inférieure de l'intervallecourant*
Fin

Nous prenons comme exemple le message : « b », « a », « c », « a ». Le tableau 1.1 représente les différents symboles avec leurs occurrences dans le message à coder.

Tableau 1. 1-Tableau de modèle fixe

Symbole	Occurrence	Probabilité	Intervalle
a	2	0.5	[0, 0.5)
b	1	0.25	[0.5, 0.75)
c	1	0.25	[0.75, 1)

Nous illustrons le processus de codage arithmétique du message « baca » dans le tableau (1.2).Nous constatons d'après cet exemple, que le résultat final de codage dépend de l'intervalle du premier symbole codé (0.609375 ϵ [0.5, 0.75)). Et chaque symbole limitera d'avantage l'intervalle de sortie.

Tableau 1. 2-Le processus de codage arithmétique

Symbole	Taille	b_{inf}	b_{sup}
Initialement		0	1
b	1	0.5	0.75
a	0.25	0.5	0.625
c	0.125	0.59375	0.625
a	0.03125	0.59375	0.609375

La figure 1.3 montre une présentation du processus de codage arithmétique. Comme illustré, à chaque étape, la subdivision de symbole en cours de codage sera élargie. Nous l'avons représenté selon une échelle qui facilite la visualisation de ses bornes.

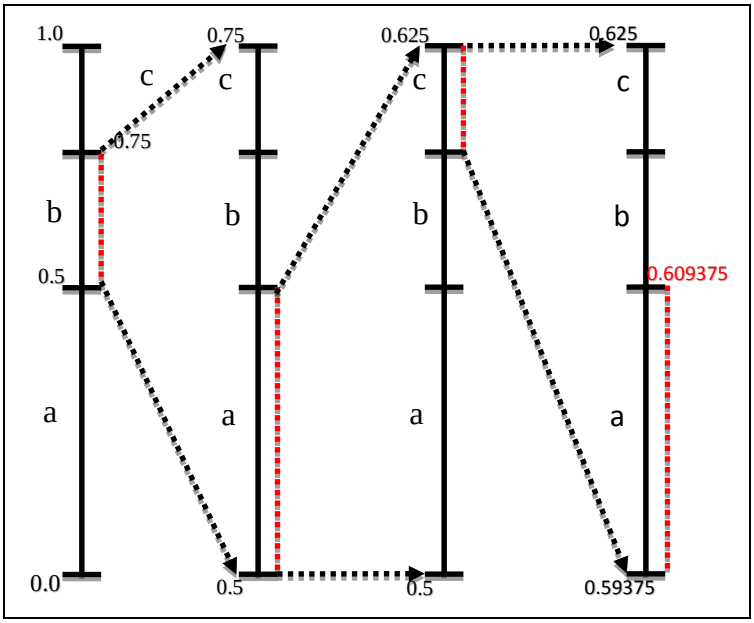


Figure 1. 3- La représentation du processus de codage arithmétique du message «baca»

3.4.2.3. Algorithme de décodage

L'algorithme de décodage fonctionne sur un principe semblable à celui du codage. D'abord, nous transmettons au décodeur la valeur 0.59375 qui, après décodage, devra se traduire au message d'entrée (baca). L'algorithme de décodage est présenté ci-dessous :

Algorithme 1.2 : Algorithme de décodage arithmétique

Entrées : r : nombre en entrée
Tant que (r ≠ 0) faire
C = symbole dont l'intervalle contient r ;
Taille= b_{sup} (C)- b_{inf} (C);// mise à jour de la longueur d'intervalle
r = r - b_{inf} (C);
r= r / Taille;// mise à jour du code
Fin

Le décodeur utilise les mêmes intervalles de probabilités utilisés au niveau codeur. En commençant par l'intervalle [0,1), seul l'intervalle [0,5, 0,75) enveloppe le code transmis, donc le premier symbole doit être «b». Tous les symboles qui suivent sont définis dans l'intervalle [0,5, 0,75). Le tableau 1.3 représente le processus de décodage.

Tableau 1. 3-Le processus de décodage arithmétique

Symbole	Taille	*r*
b	0.25	0.59375
a	0.5	0.375
c	0.25	0.75
a	0.5	0

3.4.2.4. Compression arithmétique

Le codeur arithmétique (CA) est un codeur entropique qui est largement utilisé dans les derniers standards de compression comme JPEG 2000, JBIG, H.264/AVC et HEVC. Comme nous avons vu, le CA utilise un modèle statistique qui estime les probabilités des symboles à coder durant le processus de codage/décodage. À partir de ces probabilités, nous définissons un vecteur de probabilité cumulative VPC afin d'associer à chaque symbole *Si* un sous intervalle de l'intervalle [0,1] dont la longueur est égale à sa probabilité *Pi*. Ainsi, pour chaque nouveau symbole *Si*, le CA propose de continuer avec le sous intervalle accordé à ce symbole et qui se réduit au fur et à mesure du codage de la séquence d'entrée.

Le CA peut être statique ou adaptatif autorisant au codeur la capacité de s'adapter à la variation statistique de la séquence à coder. Le modèle adaptatif est plus performant. Toutefois, il est plus lent à cause des mises à jour des probabilités durant le processus de codage/ décodage.

Nous distinguons deux types de codage adaptatif : codage d'ordre 0 (semi-adaptatif) et codage d'ordre supérieur. Pour le premier codage, nous collectons les statistiques d'apparition des symboles pour le message à coder et nous utilisons ces statistiques pour encoder de façon optimale le message. Ces statistiques doivent être sauvegardées afin de pouvoir les décoder après. Dans ce cas, nous n'utilisons pas de contexte et chaque symbole sera codé de la même façon quel que soit sa position dans le message. Pour des ordres supérieurs, les probabilités des symboles varient selon le niveau de codage. Le changement des valeurs de probabilités se fait au fur et à mesure du codage. De ce fait, après chaque symbole codé, nous mettons à jour sa probabilité (incrémentation par 1). Il existe plusieurs types de CA, le plus important étant le codeur arithmétique binaire(CAB), qui sera détaillé et implanté dans le chapitre trois de cette thèse. En effet, le CAB est capable de réduire la complexité des mises à jour des probabilités dans un schéma de codage adaptatif et de coder tout type de symbole.

Pour un CAB, le VPC s'écrit sous la forme [0, p_0, 1] avec p_0 est égale à la probabilité du symbole 0. Ainsi, l'intervalle [0,1] est divisé en deux parties pour représenter les deux symboles binaires 0 et 1. La subdivision associée au symbole 0 est [0, p_0) et celle au symbole 1 est [p_0,1).

En pratique, nous utilisons une arithmétique entière pour le CA (ou le CAB). Au lieu d'un intervalle [0,1], nous considérons un intervalle initial du type [00000,99999]. De cette manière, le codage et décodage restent les mêmes.

Les bornes des subdivisons seront alors des entiers et à chaque étape nous trouvons l'intervalle contenant l'entier courant. La figure (1.4) montre le principe du codage arithmétique entier du message précédemment cité.

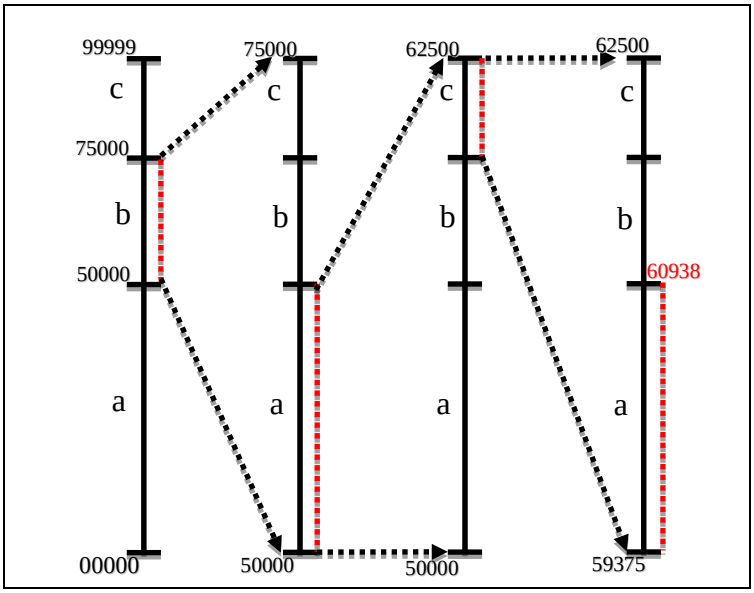


Figure 1. 4- La représentation du processus de codage arithmétique entier du message «baca»

Dans le cas où les poids forts des extrémités (B_{sup} et $B_{inf)}$ seront identiques, ils seront envoyés. A la suite, nous calibrons : décalage et entrée d'un 0 à gauche pour B_{inf} et décalage et entrée d'un 9 à gauche pour B_{sup}. À cause de la précision des calculateurs, ce principe de codage ne fonctionne pas si les deux chiffres de poids fort ne deviennent pas égaux. Par exemple, si l'intervalle est égal [59375,60938], il converge vers [59999,60000]. Pour résoudre ce problème et dans le cas d'une différence de 1 des poids forts, on compare les chiffres suivants. S'ils sont égales à 0 et 9, on les décale $[53750,69389]_1$ avec l'indice 1 pour dire que nous avons décalé le 2éme chiffre. Nous terminons le codage jusqu'avoir l'égalité des chiffres de poids fort (après k décalages), il faut alors afficher ce chiffre suivi de k zéros ou neufs, c'est pour cela que nous devons stocker un bit supplémentaires indiquant si nous l'avons convergé vers le haut ou le bas (pour le CAB, cette information se déduit directement).

4. Application JPEG

Afin de mettre en pratique les notions de redondances et les techniques utilisées dans les sections précédentes, nous proposons dans cette section d'étudier le codeur JPEG et de simuler son fonctionnement pour illustrer les gains en compression.

4.1. Principe

JPEG (Joint Photographic Experts Group) est la norme de compression avec perte la plus utilisée dans les applications industrielles de traitement d'image standardisée par le groupe d'experts de la photographie en 1994. Plusieurs modes de codage sont possibles dont le plus simple est le codage séquentiel basé sur la DCT.

Dans ce mode, d'abord, l'image est divisée de gauche à droite et de haut en bas en blocs de 8x8 pixels. Chaque bloc (8,8) est codé par DCT (Transformée en Cosinus Discret), ensuite les 64 coefficients transformés résultants sont quantifiés. Les coefficients quantifiés sont lus dans un ordre Zigzag et immédiatement codés par un codeur entropique afin de minimiser le stockage des coefficients.

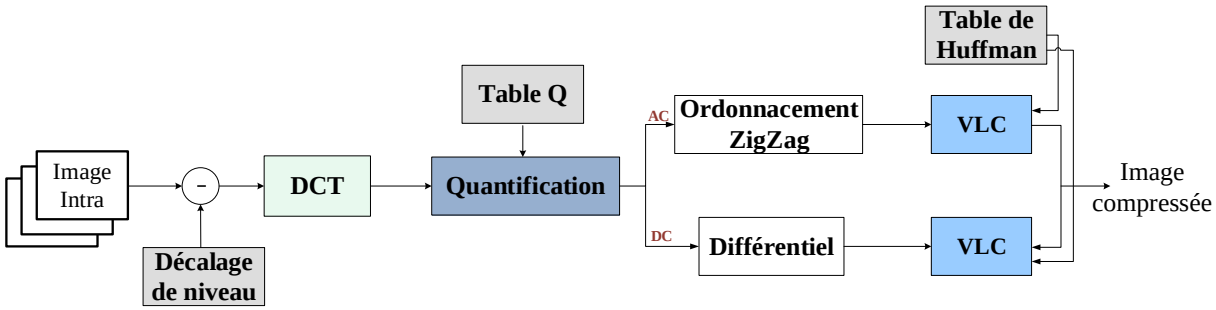


Figure 1. 5- Schéma synoptique du codeur JPEG en mode séquentiel

Le schéma de la figure (1.5) illustre l'algorithme de la compression JPEG en mode séquentiel. En effet, JPEG opère un codage Intra Frame. Il divise l'image en blocs de 8x8. C'est le préalable à tout travail de compression. Deuxième étape, on effectue un décalage de niveau où chaque élément de chaque bloc doit être amené à l'intervalle [-128,127].

Pour faire cela, on soustrait à chaque élément un scalaire 128, cela va permettre d'augmenter la tolérance de précision des 64 coefficients de la DCT. Ces coefficients sont quantifiés selon la matrice de quantification qui est spécifiée à chaque application. L'étape finale, c'est le codage Huffman, qui va compresser le vecteur RLE à la limite de la compression sans perte (VLC) en parcourant des tables de codes prédéfinies.

4.2. Transformation en cosinus discrète DCT

Le principe de la transformation DCT a été détaillé dans la sous-section 2.3.1. Pour chaque bloc de 8x8 pixels, une transformée DCT-2D (en 2 dimensions) est appliquée. Le résultat est une matrice de 8x8 coefficients. Le premier coefficient de la matrice représente le coefficient continu DC et le reste des coefficients sont les coefficients AC. En fait, nous transformons les pixels en valeur de coefficients liés à la fréquence d'analyse de la zone.

Le coefficient le plus élevé correspond à la zone où les pixels changent le plus rapidement d'état alors que les coefficients les plus faibles sont le reflet des zones qui changent le moins fréquemment. Les changements rapides d'intensité du pixel sont représentés alors par les hautes fréquences. Ces hautes fréquences seront négligées vu que l'œil est peu sensible à ces changements. Les formules de la transformation DCT, qui permet de passer de l'espace spatial à l'espace fréquentiel et la transformation DCT inverse sont décrits dans la sous-section 2.3.1 (équations (1.4) et (1.5)).

$$Y = AXA^T \tag{1.11}$$

Avec $A = \begin{pmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -e & -c & -a \\ b & f & -f & -b & -b & -f & f & b \\ c & -g & -a & -e & e & a & g & -c \\ d & -d & -d & d & d & -d & -d & d \\ e & -a & g & c & -c & -g & a & -e \\ f & -b & b & -f & -f & b & -b & f \\ g & -e & c & -a & a & -c & e & -g \end{pmatrix}$ et $\begin{cases} a = \cos(\frac{\pi}{16}), b = \cos(\frac{2\pi}{16}) \\ c = \cos(\frac{3\pi}{16}), d = \cos(\frac{4\pi}{16}) \\ e = \cos(\frac{5\pi}{16}), f = \cos(\frac{6\pi}{16}) \\ g = \cos(\frac{7\pi}{16}) \end{cases}$

La spécificité de la DCT utilisée dans le standard JPEG est qu'elle est réelle. En effet, d'autres standards tels que H264 ou HEVC utilisent des formes de DCT entières ou facilement intégrables. La matrice de la transformation réelle utilisée en JPEG est présentée dans l'équation (1.11).

4.3. Quantification

La quantification consiste à accentuer l'effet de la DCT en séparant les données en hautes fréquences et basses fréquence. La quantification utilisée dans les standards de compression d'images comme le JPEG est une quantification non-linéaire. En ne retenant principalement que les coefficients correspondants aux fréquences basses et en négligeant volontairement les hautes fréquences, par application d'une table de quantification ou d'un seuil. Ainsi nous allons réduire le volume des données de manière appréciable.

Plus concrètement, une division euclidienne est appliquée entre les blocs des coefficients DCT et une table de quantification déjà calculée sur la base d'un facteur de qualité.

Cela a pour but d'éliminer les coefficients haute fréquence dont l'œil humain est peu sensible. Avec cette quantification seulement ceux qui présentent une valeur très faible en dessous du seuil, définis par la table de quantification, auront des coefficients arrondis à l'entier prêt qui est souvent 0.

Le standard JPEG défini deux tables de quantification différentes : une pour la composante de luminance *Y* et une autre pour les composantes de chrominances rouge et bleue *Cr* et *Cb*. Nous représentons les éléments du tableau de quantification par M(u, v). Tout coefficient DCT quantifié à location (u,v) est indiqué par l'équation (1.12):

$$F^q(u,v) = \left\lfloor \frac{F(u,v)}{M(u,v)} \right\rfloor \tag{1.12}$$

où F (u, v) est la valeur du coefficient transformé avant la quantification, et⌊ ⌋ implique l'opération d'arrondissement. On remarque que les coefficients en hautes fréquences seront divisés sur un nombre élevé, et ceux en basses fréquences seront divisés sur un nombre moins élevé. Nous employons un facteur de qualité FQ pour contrôler les éléments de la matrice de quantification selon le débit désiré.

Ses valeurs en pourcentage sont comprises entre 1 et 100. Nous prenons le cas d'un facteur de qualité FQ = 50 et nous illustrons le tableau de quantification de luminosité normalisée (tableau 1.4) pour le standard JPEG.

Tableau 1. 4 - Matrices de quantification normalisée pour la norme JPEG

Table de quantification : Luminance							
6	11	10	116	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table de quantification : Chrominance							
17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

La qualité de l'image finale va dépendre de l'agressivité des coefficients de cette table de quantification. En outre, cette table sera stockée dans l'image JPEG finale derrière un marqueur pour que l'on puisse la restituer. Pour FQ = 100, les éléments du tableau de quantification sont fixés à 1. Pour d'autres facteurs de qualité, les éléments de la matrice de quantification M (u, v) sont multipliés par un facteur de compressionαdéfini comme :

$$\alpha = \frac{50}{\mathrm{FQ}} \quad si \quad 1 \leq \mathrm{FQ} \leq 50 \tag{1.13}$$

$$\alpha = 2 - 2 \times \frac{\mathrm{FQ}}{100} \quad si \quad 50 \leq \mathrm{FQ} \leq 99 \tag{1.14}$$

Nous obtenons nouvelle matrice de quantification M' (u, v) :

$$\mathrm{M'(u,v)} = \alpha \times \mathrm{M(u,v)} \tag{1.15}$$

Dans le cas où FQ = 25 par exemple, α sera égal à 2 et tous les coefficients du tableau 1.4 seront multipliés par deux.

4.4. Codage des coefficients DC et AC

Après la quantification, l'opération d'arrondi est appliquée aux coefficients DCT, le coefficient DC (indiqué par (0.0)) et les 63 coefficients AC sont codés séparément comme représenté dans la figure (1.6.a). Les coefficients DC sont codés en DPCM (Differential pulse-code modulation) afin d'exploiter la corrélation entre les valeurs DC des blocs adjacents et les coder plus efficacement car ils contiennent typiquement la plus grande partie d'énergie d'image. Pour ceci, nous utilisons la valeur de coefficient DC de bloc précédent

$$DIFF = DC_i - DC_{i-1} \tag{1.16}$$

Ce type de codage permet de réduire la redondance inter-pixel. Les coefficients AC sont codés par plage RLE. Autrement, les coefficients AC qui le suivent et qui ont la même valeur sont codés par leur nombre de redondance sur la séquence, après un balayage zigzag comme représenté dans la figure (1.6.b).

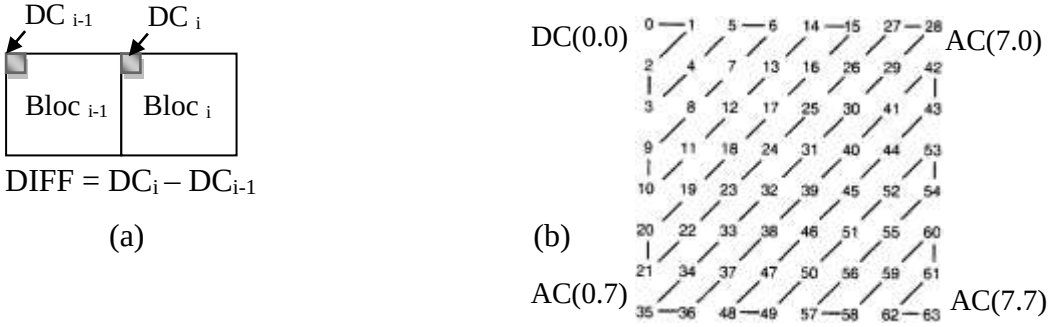


Figure 1. 6- Préparation des coefficients DCT pour le codage par entropie

Nous mentionnons que le balayage zigzag constitue un prétraitement pour le codage entropique des coefficients. Ainsi nous accédons d'abord aux coefficients les plus susceptibles d'être non-nuls vu que l'énergie d'image se concentre principalement dans quelques coefficients de basse fréquence pour la plupart des scènes naturelles. Enfin, les vecteurs DC et AC seront réunis pour le codage entropique Huffman.

4.5. Codage à longueur variable de type Huffman

Le codage entropique est réalisé sur deux étapes. La première étape est le transfert des coefficients quantifiés à un ensemble intermédiaire de symbole. Dans la deuxième étape, des codes de codage par plage sont assignés à chaque symbole. Pour la norme JPEG, un symbole se compose de deux parties : un code de codage par plage pour la première partie suivi d'une représentation binaire de l'amplitude pour la deuxième partie. Deux tableaux de Huffman pour les coefficients DC et AC sont cités dans l'annexe A (tableau A.1 et tableau A.2) comme

définis dans [JPEG 14]. Pour les coefficients DC, les valeurs de DIFF sont classifiées par catégorie en se basant sur la gamme de magnitude appelée CAT. Le tableau (1.5) montre les catégories pour la gamme des amplitudes de JPEG en mode séquentiel. Puisque les valeurs des coefficients sont dans la gamme (-2047,2047), nous distinguons 11 catégories pour les coefficients non nuls. La catégorie nulle est employée uniquement pour définir la fin du bloc EOB (End Of Bloc).

La catégorie CAT est ajoutée avec des bits supplémentaires pour spécifier la valeur actuelle de l'amplitude (DIFF). Si le DIFF est positif, les bits ajoutés sont les bits d'ordre inférieur du DIFF. S'il est négatif, les bits ajoutés sont les bits d'ordre inférieur du DIFF-1.

Tableau 1. 5- Catégories (CAT) de codage séquentiel

Catégorie	Range
0	0
1	-1,1
2	-3,-2, 2,3
3	-7,…,-4,4,…,7
4	-15,…,-8,8,…,15
5	-31,…,-16,16,…,31
6	-63,…,-32,32,…,63
7	-127,…,-64,64,…,127
8	-255,…,-128,128,…,255
9	-511,…,-256,256,…,511
10	-1023,…,-512,512,…,1023
11	-2047,…,-1024,1024,…,2047

Les bits d'ordre inférieur commencent à partir du point où le bit de poids fort des bits ajoutés est 1 si DIFF positif et 0 si DIFF négatif. Par exemple : si DIFF = 8(0000... 01000), les bits ajoutés commencent à partir de 1, par conséquent ils sont 1000. Puisque 8 est dans la gamme de 8 à 15, la valeur du CAT est 4. A partir du tableau A.2 des coefficients AC, le code binaire pour CAT = 4 est 1011, donc le code binaire pour DIFF = 8 est 10111000, où 1011 est le code de VLC de CAT et 1000 est le code binaire ajouté.

Pour un DIFF négatif, comme DIFF = -4 qui est dans la gamme de -7 à -4, CAT = 3 et son code VLC du tableau A.1 est 100. Pour trouver les bits ajoutés, DIFF - 1 = -5(1111…1010), où les bits d'ordre le plus bas sont 010, car le bit de poids fort est 0, donc le code binaire devient 100010. Pour chaque coefficient AC non nul, on associe la paire (RUN, CAT).

Pour le codage séquentiel, le CAT est la catégorie pour l'amplitude d'un coefficient non nul dans l'ordre zigzag, et le RUN est le nombre de zéros précédant un coefficient non nul.

La longueur maximale de RUN est 15. Le codage de RUN supérieur à 15 est fait par (15, 0) dont la longueur de RUN est 15 coefficients nuls suivis d'un coefficient ayant l'amplitude 0. Par conséquent, nous pouvons le considérer comme symbole d'extension avec 16 coefficients nuls [Minh 07]. Enfin, EOB (fin de bloc) indique que le reste des coefficients dans l'ordre de balayage de zigzag sont nuls. Ce symbole est représenté par (0, 0).

4.6.Vérification fonctionnelle du CODEC JPEG

Pour exploiter les performances du CODEC JPEG, nous avons utilisé le codec JPEG de référence dans [Compressor 12]. Ce codec de référence développé avec le logiciel Matlab, prend comme entrée une image non compressée et génère une image compressée selon la norme JPEG, avec la possibilité de changer le rapport de compression en paramétrant le facteur de qualité FQ. Un exemple de compression est montré dans la figure (1.7). La figure (1.7.a) et la figure (1.7.c) représente respectivement l'image d'entrée (originale) et de sortie (décodée); la figure (1.7.b) montre la DCT de l'image originale pour un facteur de qualité 50.

(a) Image originale (b) DCT (c) Image décodée

Figure 1. 7- Compression JPEG standard

(a) Sous-bloc de l'image originale (b) Sous-bloc de l'image décodée

Figure 1. 8- Différence au niveau sous-blocs entre l'image d'entré et de sortie pour le CODEC JPEG standard

Afin de mieux voir l'effet de la compression, nous faisons un zoom de deux sous-blocs de l'image originale et de l'image décodée dans la figure 1.8. Nous pouvons voir une légère différence au niveau nuances des blocs pour le facteur de qualité choisi.

Ces différences seront quantifiées dans les paragraphes suivants en utilisant des paramètres de mesure de qualité objective comme le MSE et le PSNR. Pour analyser les performances du codec de référence, nous faisons varier le facteur de qualité et nous récupérons les paramètres de mesures de performance de la compression.

Trois images de test ont été choisies et six valeurs de FQ ont été considérées (5, 10, 25, 50 ,75 et 90) ce qui correspond à des faibles, moyens et hauts débits. Dans un premier temps nous avons relevé le taux de compression conformément à l'équation (1.17) :

$$\text{Tc (\%)} = 100 \times \frac{\text{taille stream de output (compressé)}}{\text{taille stream de input (original)}} \quad (1.17)$$

Comme nous pouvons le voir dans la figure 1.9, les taux de compression varient entre 70% et 97%. Plus le FQ est faible, plus les coefficients de la tables de quantification sont grands, plus les coefficients quantifiés à zéro sont nombreux et meilleur est le rendement du codeur entropique (grande succession des zéro dans les chaîne à compresser). De plus, le taux de compression varie selon les images à coder.

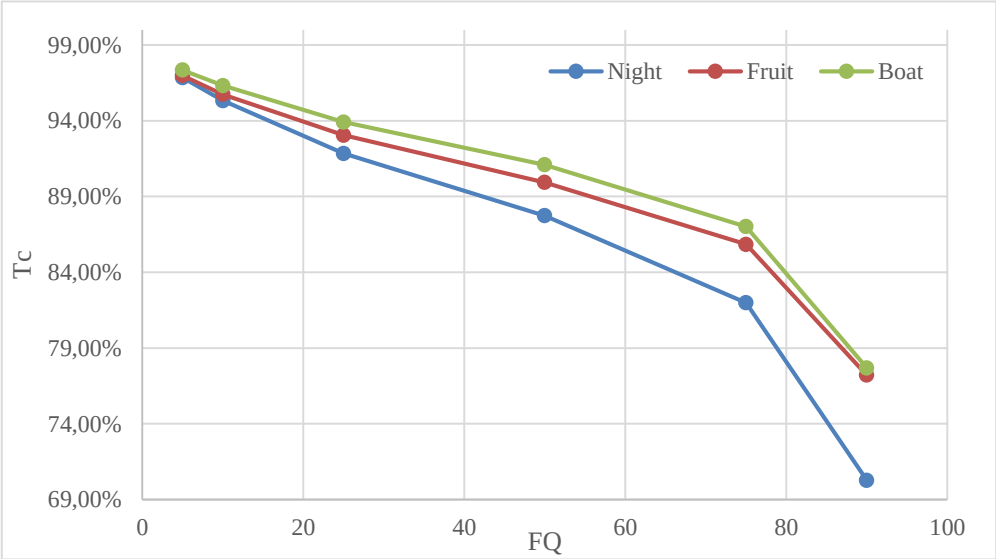


Figure 1. 9- Les variations des taux de compression en fonction du FQ

Par exemple, le taux de compression dans l'image "Night" et toujours plus faible que celui dans l'image "Boat" ou "Fruit" quel que soit le facteur de qualité. Ceci est expliqué par le nombre important de contours dans l'image "Night". En effet, plus il y a de contours dans l'image à compresser, plus les coefficients de la DCT sont non nuls et moins bien est le rendement du codeur entropique. Pour mesurer la qualité des images compressées nous utilisons le PSNR défini par l'équation suivante :

$$\text{PSNR} = 10 * \log_{10}\left(\frac{d^2}{\text{EQM}}\right) \quad (1.18)$$

où d est la dynamique du signal souvent fixé à 255 pour des pixels de taille 8 bits, et EQM désigne l'erreur quadratique moyenne définie pour deux images x et y, de taille M*N par :

$$\text{EQM} = \frac{1}{M*N}\sum_{i,j}(x(i,j) - y(i,j))^2 \quad (1.19)$$

Les variations des PSNR en fonction du facteur de qualité sont illustrées sur la figure 1.10.

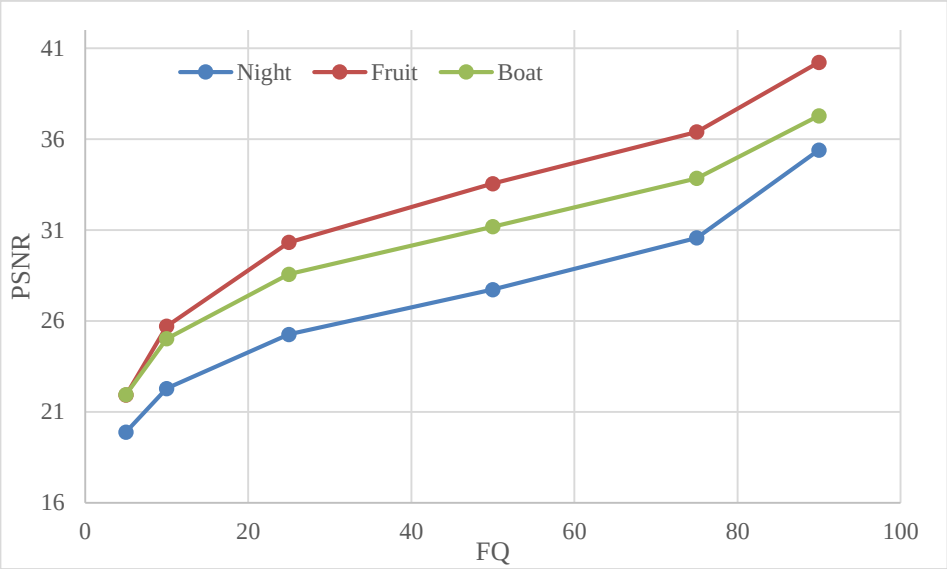


Figure 1. 10- Les variations des PSNR en fonction du FQ

Les valeurs du PSNR sont comprises entre 20dB et 40dB. Plus le facteur de qualité est grand plus la qualité de l'image compressée est élevée ce qui correspond à une valeur de PSNR plus grande. Aussi, il faut noter que la reconstruction d'une image avec beaucoup de contours est souvent accompagnée par un nombre d'erreurs importants et par conséquent par une perte de qualité. Ceci est vérifié en comparant les PSNR obtenus avec l'image "Boat" ou "Fruit" par rapport à ceux obtenus avec l'image "Night" ce qui revient à comparer les valeurs numériques des EQM mentionnées dans le tableau 1.6. Dans ce tableau, nous démontrons les images originales et décodées avec les EQM correspondants pour chaque facteur de qualité.

Tableau 1. 6- Résultats de la compression JPEG pour quelques images de test

Facteur de qualité	Night	Fruit	Boat
5	 EQM=32.095	 EQM=27.256	 EQM=26.140
10	 EQM=23.204	 EQM=16.903	 EQM=17.483
25	 EQM=16.047	 EQM= 9.989	 EQM=11.416

50	 EQM=12.123	 EQM= 6.890	 EQM=8.578
75	 EQM= 8.917	 EQM=4.905	 EQM=6.523
90	 EQM=5.364	 EQM=3.117	 EQM=4.529

4.7. Codage entropique modifié

Afin d'améliorer les performances du codeur de référence, nous présentons dans cette sous-section deux codeurs entropiques adaptés au CODEC JPEG : un codage de Huffman modifié et un codage arithmétique adaptatif et nous comparons leurs performances par rapport au codeur de référence utilisé dans la section précédente.

4.7.1. Huffman modifié

Dans le codeur de référence, le codage entropique est basé sur une table Huffman prédéfinie par le standard au JPEG. En effet, nous utilisons des codes de Huffman construits sur des images de test et connus par défaut du codeur et du décodeur. Ainsi, nous pensons que les codeurs sont moins performants en compression mais on n'a ni de données annexes à stocker ni de calculs pour le codage/décodage (simple recherche dans un tableau). L'optimisation proposée pour améliorer les performances de compression consiste à calculer les tableaux de codage relatifs aux images à compresser. Pour valider, nous comparons les performances trouvées avec celles du codeur JPEG de référence.

Nous présentons un exemple de codage/décodage utilisons notre symbole optimisé dans la figure (1.11). L'image d'entrée est de niveau de gris 8bit 512x512 pixels de format BMP.

La taille de l'image originale est égale à 262144 pixels. Sa taille originale en bit est 2097152 bits et la taille d'image compressée est 137431 bits, ce qui correspond à un taux de compression Tc = 93.447%.

(a) Image originale (b) DCT (c) Image décodée

Figure 1. 11- Compression JPEG avec le codage entropique Huffman modifié

Le PSNR est de 36.445 dB avec EQM égale à 14.742 avec une représentation de 0,5243 bit par pixel (Bpp).

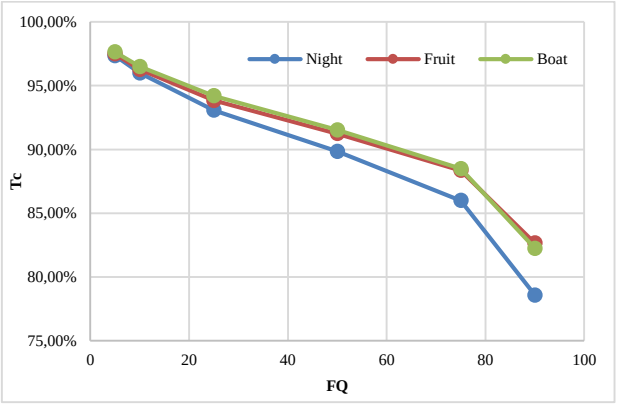


Figure 1. 12- Les variations des taux de compression en fonction du FQ

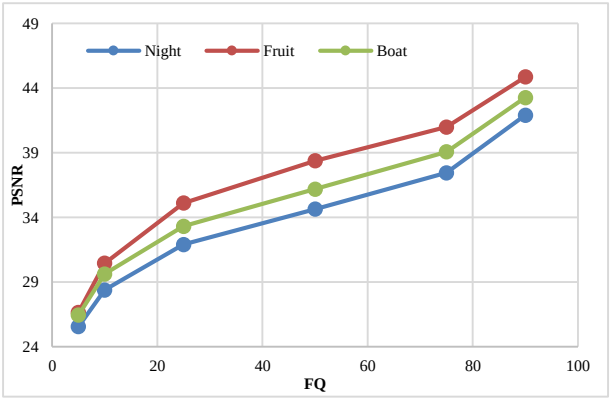


Figure 1. 13- Les variations des PSNR en fonction du FQ

Nous avons conduit les mêmes analyses que dans la section précédente pour relever les valeurs du Tc et du PSNR par rapport aux différents facteurs de qualité. Les résultats de ces simulations sont présentés dans la figure (1.12) et montrent que Tc varie avec notre approche entre 78% et 97% contrairement à une variation entre 70% et 97 % dans le codeur de référence. Quant au PSNR, il présente une variation entre de 25 dB à 44 dB, contre une variation de 20 dB à 40 dB pour le codeur de référence.

Ainsi, nous pouvons mettre en valeur l'apport du changement des tables du codeur de Huffman. Les tables que nous avons considérées sont liées aux images que nous compressons. Cette idée d'adapter le codeur à son contexte d'utilisation a été reprise par les nouveaux codeurs vidéo pour faire des familles de codeurs adaptés aux contextes CA (context adapted).

4.7.2. Codage arithmétique adaptatif

Une autre amélioration du codeur entropique réalisée est décrite dans cette section. Notre nouvelle optimisation consiste à remplacer le codeur Huffman par un codage arithmétique

binaire. Comme nous l'avons décrit dans la sous-section 3.4.2. Ce codage est basé sur l'application d'une suite d'opérations à un intervalle donné afin de modifier ses bornes à chaque ajout d'un symbole. La mise en œuvre sous Matlab de ce codage est faite sur des images fixes de niveau gris 8 bits.

Les résultats de simulation pour la même image d'entrée présentée sur la figure (1.13) (même PSNR et FQ), donnent une taille d'image compressée 155993 bits, soit un Tc= 92.562 % et un Bpp = 0.5951. Ces valeurs sont plus faibles que celle obtenues avec le Huffman modifié. Nous testons après le codeur arithmétique sur les images tests auparavant utilisés. Les mêmes analyses faites dans les sections précédentes ont été reproduites avec le nouveau codeur entropique pour relever les valeurs du PSNR et le taux de compression. Nous avons remarqué que le taux de compression varie entre 77% et 96% ce qui est très proche du taux obtenu avec le codeur de Huffman modifié. De la même façon, les valeurs de PSNR obtenus avec ce codeur sont comprises entre 25 dB et 44 dB et sont donc sensiblement identiques à celles obtenues avec le codeur de Huffman modifié.

4.7.3. Comparaison des techniques utilisés pour le CODEC JPEG

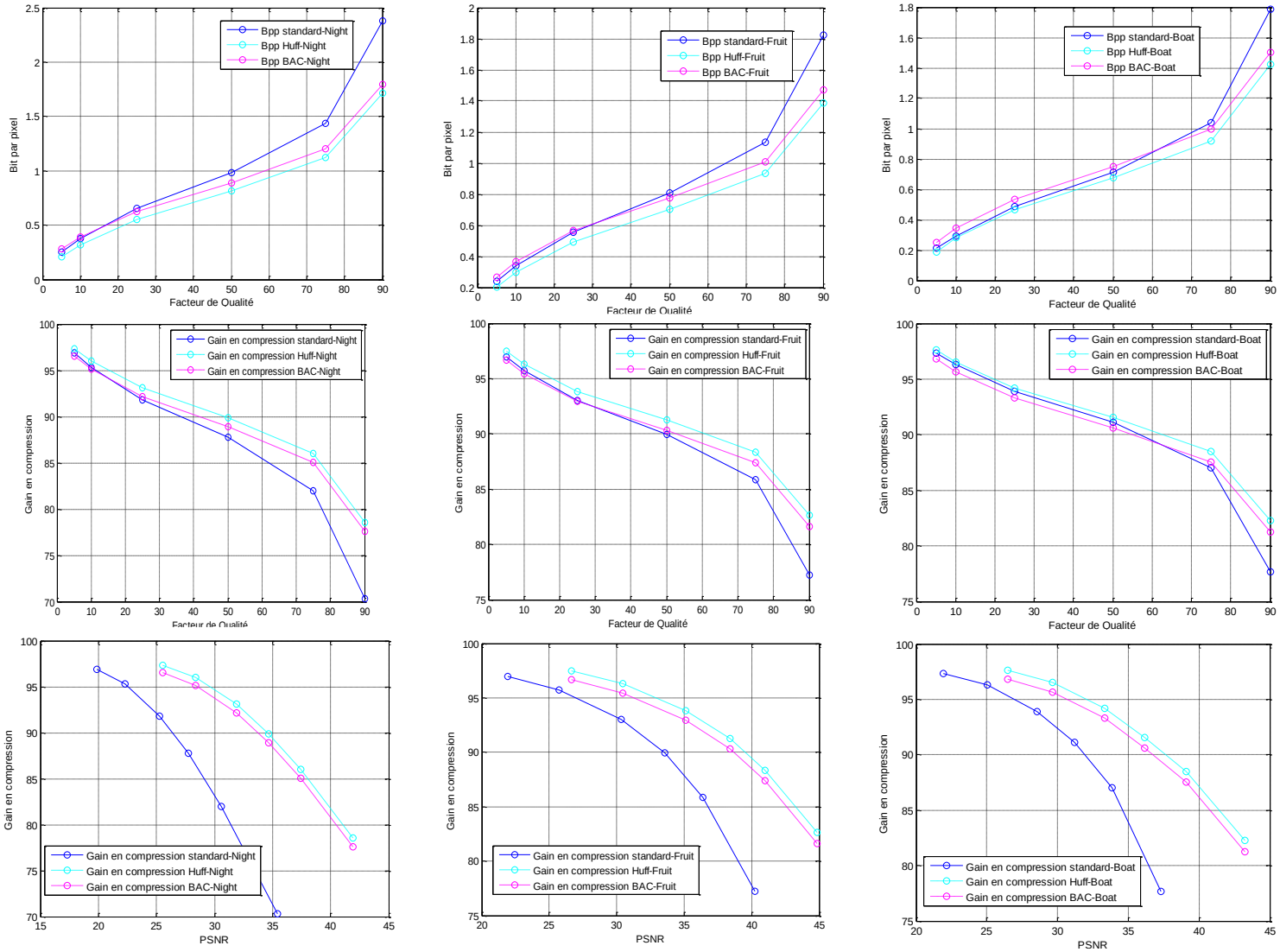
Nous avons obtenu dans les sous-sections précédentes différentes mesures de performances pour 3 types de codeurs entropiques. Nous proposons ici de comparer plus finement leurs performances. Pour cela, nous illustrons les performances trouvées pour l'image Night, Fruit et Boat sur la figure (1.13) : les allures en bleu pour le codage standard, les allures en ciel pour le codage Huffman modifié et les allures en rose pour le codage arithmétique.

Dans la figure 1.14, nous montrons un résumé des différents résultats exposés dans la section précédente. Dans la troisième ligne, le gain de compression en fonction du PSNR est présenté. Le gain de compression est défini par l'équation suivante :

$$\text{Gain de compression (\%) : } = 100 \times \left(1 - \frac{\text{taille stream de output (compressé)}}{\text{taille stream de input (original)}}\right) \qquad (1.20)$$

Les gains en compression obtenus par le codeur Huffman modifié et par le codeur arithmétique sont meilleurs par rapport à ceux obtenus par le codeur de référence pour les mêmes valeurs de PSNR.

Aussi, nous remarquons que les gains en compression obtenus par le codeur de Huffman modifié sont légèrement meilleurs que ceux obtenus par le codeur arithmétique. Néanmoins, il est important de souligner que le codeur arithmétique utilisé n'est pas adapté au contexte comme c'est le cas du codeur de Huffman.



Night Fruit Boat

Figure 1. 14-les performances de compression pour les images Night, Fruit et Boat.

5. Conclusion

Dans ce chapitre, nous avons présenté quelques techniques permettant de réduire les redondances destinées aux images fixes et animées. Nous avons détaillé les types des codeurs entropiques utilisés dans la chaîne de compression. De plus, nous avons implanté deux codeurs entropiques de type Huffman et arithmétique dans le codec JPEG de référence et nous avons montré que les performances du codeur modifié sont meilleures.

Nous pouvons confirmer que les deux méthodes implantées nous donnent des performances meilleures par rapport au JPEG standard de référence en termes de gain de compression et débit binaire. La méthode de Huffman adaptative utilisée nous assure aussi une optimisation intéressante au niveau du flux binaire obtenu au codage pour la même qualité d'image.

Chapitre 2 : Compression vidéo

1. Introduction

Depuis la norme MPEG-2 jusqu'au standard HEVC, des progrès sont réalisés dans le domaine de codage vidéo avec des performances de plus en plus élevées et des outils davantage plus puissants. Un chiffre résume cette évolution ; est qu'avec le standard HEVC, et pour une qualité de vidéo équivalente, 50% de réduction de débit binaire a été effectuée par rapport au standard H.264 des applications de très haute résolution en exécutant un codage parallèle.
L'objectif de ce chapitre est d'étudier quelques techniques de compression utilisées afin de comprendre comment des réductions de débit peuvent être atteintes. Nous allons donc commencer par présenter les principes de la compression vidéo. Ensuite, nous introduirons les normes de codage vidéo et nous détaillerons les deux standards H.264/AVC et HEVC. Nous décrirons leurs codecs de référence et nous préciserons leurs spécificités. Enfin, nous comparerons leurs performances en compression avec des séquences vidéo de test pour différentes résolutions.

2. Aperçu des techniques de codage vidéo

Dans cette dernière décennie, nous avons vécu une véritable explosion dans les applications multimédia avec des nouvelles technologies. Notre besoin est passé d'échange de la parole et du texte à l'envoie des vidéos numériques .Cela pose des problèmes de transmission ou de stockage par leur taille importante [Boudjada 08]. En effet, un signal vidéo est un ensemble d'informations permettant de transmettre, d'enregistrer et de reconstruire une image dans un temps bref. Il est constitué d'une suite de symboles binaires 0 et 1, en l'occurrence 1 485 000 000 bits par seconde pour un signal vidéo Full HD (ou 1080). Les quantités d'informations sont extrêmement volumineuses ce qui induit un temps de transfert prohibitif. Cependant pour des applications de télévision, nous devons utiliser une fréquence d'au moins 25 images par seconde. D'autres applications de ralenti nécessitent des capteurs permettant l'acquisition de 92 images par seconde, comme le VITA 2000 [Vita].
Par conséquent, une compression est bien nécessaire pour réduire le flux des données tout en assurant un transport aisé et un signal reconstruit proche le plus possible à l'original. Ainsi, il est possible de recourir aux nombreux algorithmes de compression vidéo connus, permettant de réduire les flux de données. Nous appelons ces algorithmes « CODEC » (COdage/DECodage).

Ces algorithmes utilisent des techniques pour les calculs de redondance, de prédiction, etc afin de transformer l'image en grandeurs mathématiques permettant d'agrandir le taux de compression. Dans ce contexte, plusieurs standards de compression ont ainsi été proposés afin d'améliorer l'efficacité de codage choisi. L'UIT-T [ITU] et l'ISO/MPEG [MPEG] sont les plus actifs organismes dans la normalisation des systèmes de compression vidéo. Les codeurs issus de ces deux groupes reposent sur des principes de base communs, mais différents en leur configuration choisie selon l'application.

Ce chapitre passe en revue les principes généraux de compression pour des vidéos. Nous rappelons le concept de codage vidéo afin d'avoir une vision détaillée des principes employés dans les systèmes de compression vidéo. Ensuite, un bref aperçu est donné sur les standards de compression les plus courants et en particulier de la norme H.264 (MPEG-4/AVC) et la norme HEVC, leurs spécificités et leurs performances. Nous terminerons par la présentation des types de codages entropiques et leurs propriétés qui sont au cœur de notre préoccupation.

2.1. Introduction au codage vidéo

Le codage vidéo fait partie de nombreuses applications multimédia disponibles aujourd'hui. Pour certaines applications, une bande de transmission limitée ou une capacité mémoire limitée contribuent à une demande de rapport de compression plus élevé. Il faut donc traiter la source vidéo afin de réduire ou/et supprimer les données redondantes, cela mène à une transmission ou stockage efficace du fichier vidéo numérique. En effet, une scène vidéo réelle est continue dans l'espace et dans le temps. Une vidéo numérique est la représentation discrète de cette vidéo, échantillonnée spatialement et temporellement. Elle contient une quantité importante de données et elle se caractérise par son système de représentation des pixels, sa taille et sa fréquence de rafraîchissement (en images/s). Chaque élément d'image (pixel) est représenté par une valeur donnant son niveau de luminance et de chrominance, représentée sur un nombre fini de bits. Pour réduire la quantité d'information nécessaire au codage, de nombreuses méthodes ont été mises en place pour exploiter les redondances dans une séquence vidéo. Il existe plusieurs types de redondances dans une vidéo comme nous avons détaillé dans le chapitre 1 de cette thèse et nous citons en bref :

- la redondance spectrale : données redondantes entre les pixels qui se trouvent dans la même position fréquentielle des différentes bandes constituant l'image;
- la redondance temporelle : données redondantes entre plusieurs images successives ;
- la redondance spatiale : données redondantes à l'intérieur de l'image ;
- la redondance statistique : données redondantes au sein des informations compressées.

Le processus de réduction des redondances ou compression consiste à une exécution algorithmique de la source vidéo qui génère à la fin un fichier compressé. Un algorithme inverse est appliqué au niveau décodeur pour la lecture du fichier compressé. La vidéo décodée contient quasiment le même contenu que la vidéo d'origine. Le diagramme fonctionnel de cette paire d'algorithmes (codeur/décodeur) est présenté par la figure suivante.

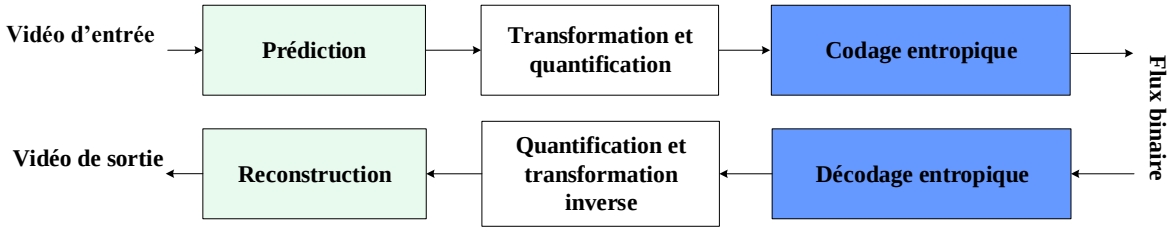


Figure 2. 1. Chaîne directe et inverse d'un code vidéo

Un codeur vidéo consiste à mettre en œuvre des traitements de base (que nous détaillons à la suite) pour produire un flux binaire compressé. Ces traitements sont réversibles et produisent une quantité de données compressées de taille bien inférieure par rapport au flux binaire initiale.

2.2. Les principes généraux de compression d'images

Pour éliminer la redondance à l'intérieur ou entre images, nous effectuons une prédiction intra ou inter. La différence avec l'image originale, plus communément appelé le résidu, est ensuite transformée dans un autre espace pour réduire la redondance du signal. La quantification permet de réduire davantage le nombre de niveaux du signal en tenant compte des propriétés du système visuel humain. Finalement, le codage entropique utilise la distribution statistique des données pour réduire le nombre de bits nécessaire pour coder ces données.

Sachons qu'un module supplémentaire peut être ajouté pour contrôler le débit dans certaines applications afin de contraindre le flux binaire pour respecter un débit fixé.

2.2.1. La prédiction

Toute image est subdivisée en macroblocs. Le codeur calcule une prédiction du macrobloc courant d'après les données déjà codées provenant soit de la même image (prédiction Intra) ou bien à partir d'autres images (prédiction Inter). Le codeur soustrait ensuite la prédiction du macrobloc courant pour former un résidu. L'idée de prédiction consiste à approximer un pixel ou un bloc de l'image à partir des données issues d'une partie de l'image courante déjà encodée est issue de la redondance spatiale de l'image ou bien d'une image précédemment décodée de la séquence travaillant sur la redondance temporelle des données provenant d'autres images.

De cette manière, le décodeur reconstruira la même prédiction qu'à l'encodeur sans transmettre 4des informations supplémentaires. L'enjeu de l'étape de prédiction est majeur dans les schémas de compression : une prédiction efficace réduira de façon considérable la quantité d'informations à transmettre.

Certains codeurs, destinés à des applications de hautes qualités et hauts débits, utilisent uniquement la prédiction spatiale. Il s'agit des codeurs "intra" dont l'intérêt principal est de limiter la complexité du codage. Nous reviendrons plus en détail sur la description des différents types de prédiction (section 3.2.2) à travers la présentation du codage vidéo du standard H.264.

2.2.2. La transformation et la quantification

Divers transformées sont utilisées, notamment les transformations orthogonales [Martin 2010]. L'intérêt de ces transformations est leur gain de codage très intéressant par rapport aux transformations non-orthogonales.

- La transformée de KLT (Karhunen Loève Transform) est un changement de base pour obtenir de nouveaux axes où l'information y est contenue de façon optimale. En effet, le nouvel espace optimalise la décorrélation des couleurs et concentre l'information utile sur deux composantes. Autrement, cette transformation tient compte des propriétés de l'image (luminosité, couleurs, etc. ...) et elle permet de condenser au mieux le signal sur des composantes localisées en fréquence. Cependant, cette transformation est rarement utilisée car sa mise en œuvre est trop complexe. Les transformations utilisées dans les codeurs vidéo tentent de se rapprocher au mieux des performances des performances des transformations KLT.
- La transformée de Fourier consiste à convertir une fonction du temps à valeurs complexes échantillonnées en une fonction à valeurs complexes. Ceci permet de bien localiser les fréquences recherchées de l'information. Cette transformée s'implémente facilement via des algorithmes rapides FFT. Mais son inconvénient majeur est l'écart de performance de codage par rapport aux transformées KLT.
- La transformée en cosinus discrète DCT est un cas particulier de la transformée de Fourier. Elle s'applique à une matrice carrée et son résultat sera représenté dans une matrice de même dimension. Les basses fréquences se localisent en haut à gauche de la matrice résultante, et les hautes fréquences en bas à droite. Cette transformation est la plus utilisée actuellement dans les normes de compression.

Quant à la quantification c'est l'étape de compression non conservative qui diminue la quantité d'information contenue dans les coefficients transformés. C'est la source principale de la compression effective et des pertes qui entraîne une distorsion en termes de qualité de l'image reconstruite. Le principe fondamental de la quantification est de diviser chaque coefficient d'un bloc par l'élément correspondant de la matrice de quantification, et d'en prendre la partie entière, comme nous avons décrit dans le chapitre 1 section 3.3. Nous nous trouvons alors face à un compromis : taux de compression et qualité. N'étant pas réversible, la quantification est l'étape où la notion de pertes d'information va réellement intervenir. Les coefficients quantifiés sont ensuite réorganisés et réordonnancés dans un vecteur unidimensionnel en les balayant par exemple en zigzag pour augmenter le nombre des zéro successifs. Cela conduit à une augmentation de l'efficacité de l'étape suivante de codage entropique.

A mentionner ici que l'expression et la mesure de la qualité vidéo sont considérées comme données d'études en soi. Combien même les techniques de mesure de qualité subjective existent dans la littérature, les standards de codage vidéo ne les considèrent pas. Seules les mesures objectives ont été utilisées. Il s'agit principalement de métriques de distorsion permettant d'évaluer la qualité intrinsèque d'une image comme le PSNR qui correspond au rapport signal à bruit crête-à-crête, ou l'EQM qui équivaut à l'erreur quadratique moyenne. Ces deux critères permettent de mesurer la distorsion entre l'image reconstruite et originale. Prenons par exemple le PSNR qui représente la différence mathématique entre une image et sa référence ; plus sa valeur est élevée plus de vraisemblance entre les images.

2.2.3. Le codage entropique

La dernière étape est le codage de l'information qui consiste à exploiter les propriétés statistiques des coefficients quantifiés. Pour ce faire, nous utilisons des mots courts pour représenter les événements les plus probables et des mots plus longs pour les occurrences rares. Le codage réduit donc la quantité d'information à envoyer par une méthode mathématique ; nous citons par exemple : le codage RLE, le codage de Huffman, le codage arithmétique.

2.2.4. Les balayages entrelacé et progressif

Le balayage entrelacé et le balayage progressif sont deux techniques pour la lecture et l'affichage des séquences vidéo. Le principe du balayage progressif est d'afficher toutes les lignes composant l'image les unes à la suite des autres, de haut en bas et en un seul balayage.

Ce type de balayage accroît la précision de la vidéo puisque chaque image s'affiche d'un coup. L'image obtenue est plus précise mais moins fluide.

Pour le balayage entrelacé, nous transmettons qu'une ligne sur deux du signal original et nous inversons la parité entre deux trames consécutives. Autrement, les lignes impaires de l'image sont affichées suivies des lignes paires. Chaque trame correspond à une image entière, et la fréquence de rafraîchissement de l'écran correspond également à la cadence d'image. L'inconvénient de ce balayage est la visibilité des artefacts tels que le scintillement inter lignes ou la rupture de lignes droites. Par ailleurs, le signal progressif est plus corrélé spatialement et temporellement, mais présente un débit brut deux fois supérieur à celui de l'entrelacé pour un format vidéo similaire. Pour désigner le mode de balayage, le signal vidéo est souvent nommé selon le nombre de lignes qu'il utilise suivi de« p » (pour progressif) ou « i » (pour entrelacé). Deux standards pour la haute définition sont utilisés aujourd'hui :

- 720p : désigne 1280x720 pixels en balayage progressif
- 1080i : désigne 1920x1080 pixels en balayage entrelacé

Les vidéos entrelacés 1080i présentent un avantage au niveau de la résolution. Ainsi, les téléviseurs TV SD affichent l'image à l'écran de façon entrelacée (576i). Cependant, nombreux postes fonctionnent cependant en mode progressif, ce qui permet aux signaux entrelacés entrants d'être convertis en signaux progressifs à l'intérieur de l'écran.

3. Le codage vidéo normalisé

Les normes de compression audiovisuelle ont connu une évolution effrénée qui a permis d'atteindre une grande qualité de restitution. Néanmoins, cette amélioration accroit la complexité des algorithmes employés, qui rend l'implantation d'autant plus ardue, posant de nombreux défis [Saponara 04]. En effet, les normes de compression vidéo développées n'utilisent pas les mêmes méthodes (algorithmes) pour réduire les données.

Nous trouvons alors des résultats différents en termes de débit, de qualité et de latence entre les différentes normes. Aussitôt, il est possible d'effectuer différentes implantations tant que le résultat d'un encodeur est conforme au format de la norme et au décodeur. Ainsi, nous trouvons beaucoup d'implantations adaptées à chaque objectif. Noter que contrairement à un codeur, un décodeur doit implanter toutes les parties requises d'une norme pour pouvoir décoder un flux binaire conforme. Autrement dit, une norme spécifie la façon exacte dont un algorithme de décompression doit rétablir chaque bit d'une vidéo compressée.

3.1. Les normes de compression vidéo

Pendant ces dernières années, plusieurs standards de codage vidéo ont vu le jour et se sont succédé, comme mentionné dans la figure 2.2. Pour répondre à certaines applications, les algorithmes de codage et décodage vidéo ont évolués dont le but est d'améliorer l'efficacité de la compression [Ohm 12]. L'historique des commutés de standardisation montre qu'il y a convergence entre ISO/IEC qui est à l'origine des normes MPEG et le commuté IUT-T qui a standardisé H.26x. En effet, à partir de 2004, un effort commun entre les deux commutés a conduit à la création du standard H.264 AVC.

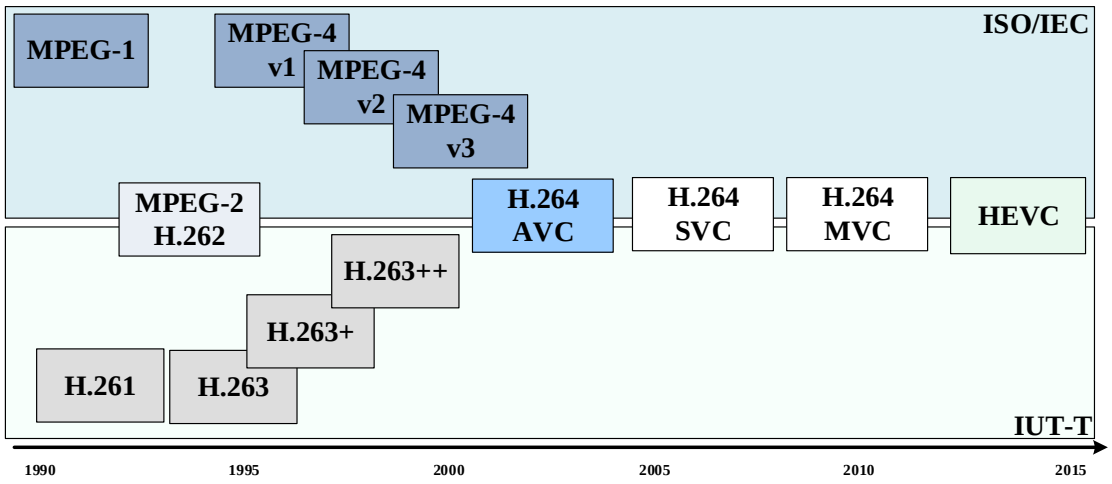


Figure 2. 2- Les standards de codage vidéo

Parmi les codecs les plus répandus, nous citons :

- MPEG-2 (Moving Pictures Expert Group) [ISO 00] : Cette norme a été développée il y a environ 20 ans pour les applications liées à la TV numérique : professionnelle (production audiovisuelle, etc.) et grand public (diffusion vers les postes TV). Elle permet une transmission efficace des signaux de télévision par satellite, par câble ou terrestre. De plus, elle permet aussi la transmission d'autres médias sur l'Internet haut débit (xDSL). Ce standard a été adopté pour les services de TV numérique par voie hertzienne terrestre et satellite. Il est également utilisé comme format de codage du DVD (Digital Video Disc).Vu que la bande passante disponible n'est pas toujours suffisante, l'utilisation de MPEG-2 devient limitée, indiquant ainsi un besoin de compression vidéo à améliorer encore.
- H.264 (ou MPEG-4 partie 10/AVC) : a été développé en fin 2001, le VCEG et le MPEG ont formé une équipe de travail conjointe JVT (Joint Video Team) pour mener à terme le standard de compression vidéo H.264/AVC. L'objectif consiste à doubler l'efficacité de compression en comparaison avec toutes les autres normes vidéo existantes [Wiegand a 03]. Ce codec vidéo répond donc à la double dénomination H.264 (pour l'ITU) et à la partie 10 du

MPEG-4 (pour l'ISO). Il vise des applications de transmission des données multimédia via des canaux sans fil et de la diffusion de vidéos HD sur les ondes radio.

Ainsi, MPEG-4 AVC/H.264 [Richardson 10] [MPEG] permet de diminuer le débit nécessaire (jusqu'à 50 %) par rapport au MPEG-2, ouvrant la voie à de nouveaux services, tels que la vidéo haute définition.

- HEVC (High Efficiency Video Coding) : publié en janvier 2013, ce nouveau standard de compression vidéo est désormais normalisé. Une commercialisation courant 2015 est prévue aux états unis avec des TV UHD équipées d'un tuner satellite intégrés DVBS2-HEVC. En France, Technicolor a sorti le premier Ste Top Box intégrant HEVC pour une commercialisation en 2017 [Technicolor]. Notant que le standard HEVC supporte toutes les résolutions surtout les hautes résolutions (1080i, Ultra HD, 4K). Parmi ses avantages, nous citons : l'élargissement de l'éligibilité aux offres IPTV, le désencombrement des réseaux et le développement des usages multi-écrans. Une efficacité doublée au niveau taux de compression est annoncée. Cette amélioration des performances est obtenue au détriment de la complexité calculatoire des étapes de compression et de décompression.
- Après HEVC : aujourd'hui tandis que de nouveaux amendements sont en train de sortir pour l'amélioration du standard HEVC, d'autres études se focalisent sur l'après HEVC. Les recherches se concentrent sur le traitement de vidéo de plus larges résolutions et sur le streaming. Un autre aspect concerne le codage vidéo faible consommation.

Dans ce sens, en 2013, MPEG a initié une réflexion sur la standardisation du « Green Metadata » [ISO 14] pour le codage vidéo faible consommation. Un résumé du "Green Metadata Standard" est fourni dans [Fernandes 15]. Il est spécifié comment les metadata permettent de réduire le budget énergétique dans la dégradation de la qualité ; et comment on peut sacrifier une faible perte de qualité quand un gain d'énergie important est garanti.

3.2. Caractéristiques de la norme H.264/AVC

La norme de compression vidéo H.264, ou MPEG-4 AVC, est la norme vidéo de choix, au cours de ces dernières années. C'est une norme ouverte et sous licence compatible avec la plupart des techniques de compression disponibles aujourd'hui. Elle est développée conjointement par l'ISO/IEC MPEG (Moving Picture Experts Group) & ITU-T VCEG (Video Coding Experts Group). Ces deux groupes ont été fusionnés sous le nom de JVT en 2001. Les travaux, commencés en 1998 à l'UIT-T, ont abouti à un standard international en mars 2003. La norme UIT-T H.264 est identique à la norme ISO/CEI MPEG-4 Part 10. La technologie employée est connue sous le nom AVC.Elle a permis une plus grande souplesse quand elle est

comparée au codage par blocs classique. Vu que le regroupement des blocs élémentaires en macroblocs de taille et de forme variable (bandes, rectangles, carrés, de résolution dimensions variables) est devenu possible. Le système de codage vidéo disposé est doté de performances accrues en matière de compression avec des outils algorithmiques ajoutés.

Le codec H.264/AVC est adapté à une grande variété de réseaux et de systèmes et il est intégré dans de nouveaux gadgets électroniques tels que les téléphones mobiles et les lecteurs vidéo numériques. Aussi, il est utilisé avec un large éventail d'applications nécessitant des débits variables. Par exemple, dans les applications de vidéo de loisir (diffusion, satellite, DVD), le standard peut offrir une performance située entre 1 et 10 Mbit/s avec une latence élevée. De plus pour des services de télécommunications, il peut offrir des débits de moins de 1 Mbit/s avec une latence réduite. Par conséquent, son acceptation par les utilisateurs a été très rapide (les sociétés de stockage vidéo en ligne, de télécommunications, la vidéosurveillance…).

Dans ce contexte, le standard définit les profils et les niveaux qui déterminent les points de conformité auxquels un décodeur doit se conformer pour être compatible avec la norme. Pour MPEG-4 AVC, il est possible de définir profils chacun ciblant une classe d'applications précise et niveaux avec une énorme variété [Wenger 03] :

- 7 profils (Baseline, Extended, Main, High, High10,High4:2:2, High 4:4:4)
- 16 niveaux qui peuvent être combinés avec les profiles.

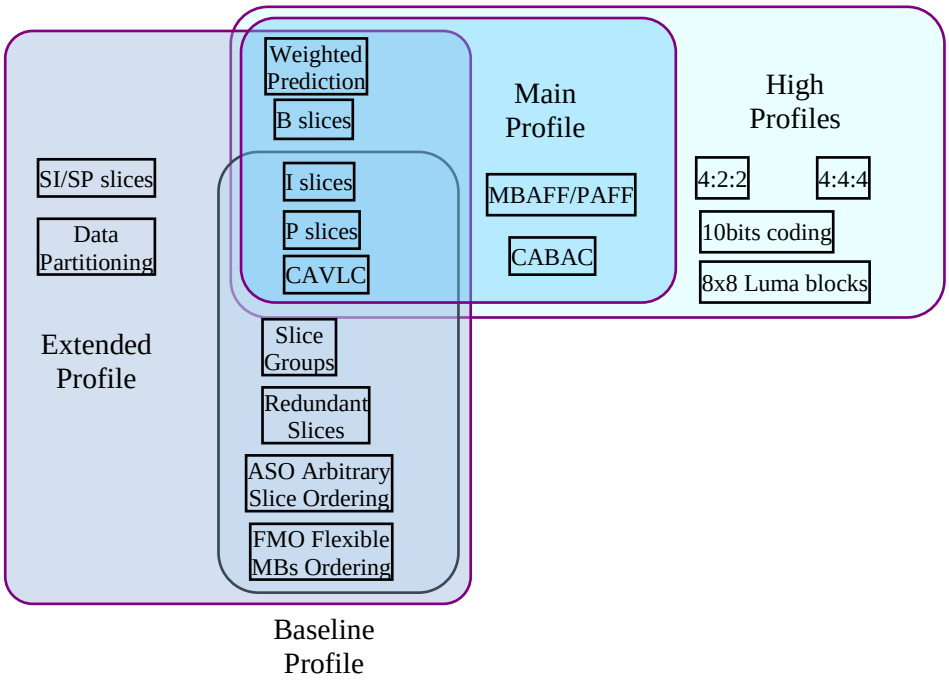


Figure 2. 3- Les différents profils proposés par H.264/AVC

Le profil Baseline est principalement recommandé pour des applications à bas coût (mobiles, visio-conférence). Nous utilisons le profil Main pour des applications grand public de diffusion et de stockage. Le profil Extended sert à des applications de Diffusion en flux (streaming). Le profil High est le profil principal pour la diffusion et le stockage (TNT-HD, HD-DVD, BD). Les autres profils sont destinés à la production et aux applications professionnelles.

Les niveaux représentent les limitations de certains paramètres qui déterminent les ressources employées dans les décodeurs. Nous obtenons un débit de 64 kbps à 240 Mbps pour les profils Baseline et Extended et un débit de 80 kbps à 300 Mbps pour le profil High.

3.2.1. Etude comportementale du codeur de référence

Un codeur H.264 va mettre en œuvre séquentiellement des traitements de base (prédiction, transformation, codage) pour produire un flux binaire compressé. Un décodeur H.264 va exécuter les opérations inverses (décodage, transformation inverse et reconstruction) pour reproduire une séquence vidéo décodée.

Le principe de codage se déroule comme suit. Chaque image de la séquence vidéo est découpée tout d'abord en macroblocs (blocs de 16x16 pixels). Le codeur calcule une prédiction du macrobloc courant d'après les données déjà codées provenant soit du même image (prédiction intra-image) ou d'autres images déjà codées et transmises (prédiction inter-image). Le codeur soustrait ensuite la prédiction ainsi formée du macrobloc courant pour donner un résidu, il s'agit d'une compensation de mouvement lorsque la prédiction est inter-image. C'est cette différence qui est traitée au niveau du codeur présenté par le schéma bloc simplifié de la figure 2.4.

Le traitement se déroule en trois étapes principales :

(i) la transformation des macroblocs résiduelles (au moyen d'une transformée entière qui est une approximation de la transformée en cosinus discrète (DCT)) permettant la décorrélation des dépendances statistiques spatiales [Wiegand b 03] cela permet de rendre le codage entropique plus efficace et de réduire les redondances spatiales de l'erreur de prédiction.

L'implantation de cette transformée ne comporte que des additions et des décalages, cela donne l'avantage de stocker des résiduels entiers et non plus flottants comme dans les prédécesseurs d'H.264.

(ii) l'opération de quantification des coefficients DCT qui va réduire la précision des coefficients de la transformée selon un paramètre de quantification QP.

(iii) le codage entropique des symboles quantifiés. En fait le processus de codage vidéo produit un certain nombre de valeurs numériques qui doivent être codées selon la syntaxe du H.264 pour produire un flux binaire compressé interprétable. Parmi ces valeurs, nous citons :

- les coefficients transformés et quantifiés
- des informations permettant au décodeur de reconstituer la prédiction
- des informations sur la structure des données compressées et les paramètres de compression utilisés pendant l'étape de codage
- des informations globales sur la séquence vidéo

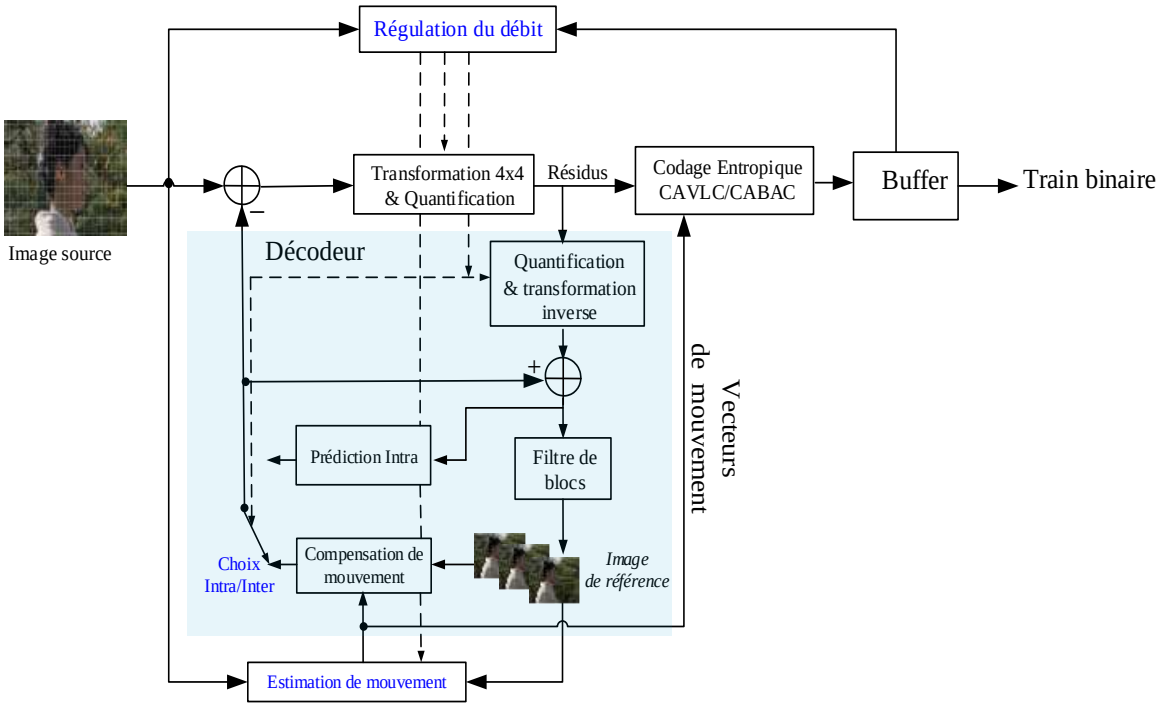


Figure 2. 4- Chaîne de codage vidéo H.264 / AVC

Ces valeurs et ces paramètres sont convertis en mots de code binaires grâce à des algorithmes de codage entropique ou arithmétique. Cela permet de produira une représentation compacte et efficace de l'information. Après cette étape de codage, le flux encodé peut être stocké ou transmis. Il n'y a en fait pas véritablement un seul nouvel élément de codage vidéo dans le VLC (Variable Length Coding) qui fournit à lui seul une amélioration importante de l'efficacité de compression par rapport aux normes vidéo antérieures. C'est plutôt la pluralité de différentes petites améliorations qui s'ajoutent jusqu'à un gain significatif. En effet, les méthodes de prédiction supportées par H.264 sont plus précises et assurent une compression plus efficace. Le codeur H.264/AVC profite des performances fournies par un filtre anti-blocs afin d'éliminer les phénomènes d'artéfacts introduits au niveau de la segmentation des images en blocs. Notons aussi que le codeur H.264/AVC possède une complexité plus importante que le décodeur associé. Cette caractéristique est due aux étapes d'identification des redondances.

3.2.2. Spécificités de H.264/AVC

De nouvelles techniques sont proposées pour permettre au codec H.264 de compresser plus efficacement les vidéos que les normes précédentes avec plus de flexibilité. En effet, nous trouvons l'ordonnancement flexible des macroblocs FMO et l'ordonnancement arbitraire des tranches ASO utilisés pour la restructuration de l'ordonnancement des régions fondamentales dans l'image [H264 15].

Cet ordonnancement améliore la résistance aux erreurs et aux pertes. En outre, les erreurs de transmission sur différents réseaux sont tolérées. Nous citons aussi la couche d'abstraction réseau NAL. Cette dernière permet l'usage de la même syntaxe vidéo dans de nombreux environnements réseau comme les paramètres de séquence SPS et d'image PPS assurant plus de robustesse et de flexibilité. Aussitôt, les tranches de commutation (appelées SP et SI) permettent à un codeur de diriger un décodeur pour s'insérer dans un flux vidéo entrant. Ceci permet du streaming vidéo à débit variable et un fonctionnement en mode truqué. Quand un décodeur a un problème au milieu d'un flux vidéo en utilisant cette technique, il peut se synchroniser. Cette synchronisation est réalisée avec les images présentes à cet endroit malgré l'utilisation d'autres images comme références préalables au déplacement.

3.2.2.1. Syntaxe hiérarchique

Dans le flux vidéo, les données sont hiérarchisées selon une manière bien précise, comme indiqué dans la figure 2.5.

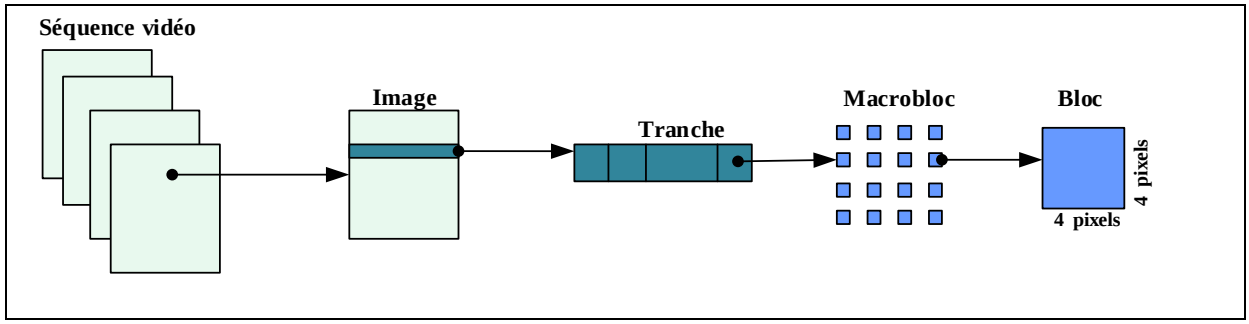


Figure 2. 5- Hiérarchie des données dans le flux vidéo

Toute séquence vidéo contient un ou plusieurs groupes d'images. Chaque image est partitionnée en tranche. Une tranche est constituée d'un ou plusieurs macroblocs adjacents, de dimension fixe et qui couvrent une zone rectangulaire de l'image de 16x16 échantillons de luminance et de 8x8 échantillons de deux composants de chrominance si nous utilisons le format d'image 4:2:0. Un macrobloc se constitue de 16 blocs, chacun de taille 4x4 pixels. L'ordre de transmission des macroblocs dans le flux dépend de leur table d'allocation et ne correspond pas nécessairement à l'ordre de balayage de l'image (figure 2.5).

Les différentes composante de l'image (Rouge, Vert, Bleu) sont codées chacune sur 8 bits, 24 bits pour coder un pixel. Les applications de vidéosurveillance utilisent des flux vidéo en nuance de gris. Donc, 8 ou 10 bits par pixel suffisent.

3.2.2.2. La prédiction Intra et Inter

La prédiction est basée sur l'analyse des redondances. Le procédé inter-trame travaille sur 3 types d'images : Les images I (Intra frame) considérées comme images de référence, les images P qui sont calculées à partir de l'image précédente et les images B (Bidirectionnal/Predicted) qui sont calculées à partir de l'image précédente ou de l'image suivante ou des deux à la fois.

Nous détaillons ces trois types d'images :

- Les images Intra : Ces images de type I sont des images autonomes qui peuvent être codées indépendamment, sans référence à d'autres images. La première image d'une séquence vidéo est toujours une intra-image. Ces images sont nécessaires en tant que points de départ pour de nouveaux téléspectateurs ou en tant que points de resynchronisation si le train de bits transmis est endommagé. Un encodeur vidéo insère automatiquement des images I à intervalles réguliers. L'inconvénient des images I est qu'elles consomment beaucoup plus de bits. Mais, d'un autre côté, elles ne génèrent pas beaucoup d'artefacts.
- Les images Inter prédites : Ces images de type P sont des inter-images prédictives qui font référence aux parties des images I et/ou P antérieures à l'aide de vecteurs de mouvement obtenus par estimation de mouvement. Ces images sont sensibles aux erreurs de transmission puisque il y a une grande dépendance avec les images de référence précédents P et I. C'est la redondance temporelle des images d'une séquence vidéo qui est exploitée.
- Les images Inter bidirectionnelles : Ces images de type B sont des inter-images bi-prédictives qui font référence à une image antérieure ainsi qu'à une image future. Elles sont codées avec une estimation du mouvement par rapport à une image précédente et une image suivante. Elles ne servent jamais de référence.

Le codage doit toujours commencer par une image I. Les images P et B, si elles sont utilisées, doivent être codées avec les images de référence. Le codage d'une séquence vidéo consiste en une succession d'images Intra et Inter agencées de manière à satisfaire les contraintes de débit, de qualité et de complexité. Une séquence d'image Inter Frame se présentera donc comme une suite d'images dépendantes jusqu'à la prochaine image de référence I : (I B B P B B P B B P B B I). Avec un GOP (Groupe Of Picture) de 12, une image I est insérée toutes les 12 frames.

Chaque image vidéo est divisée en macroblocs identiques. Chaque bloc est prédit des images précédemment codés. Il s'agit ici d'une prévision du mouvement en supposant qu'un bloc de pixels de l'image en cours a subit une translation d'un bloc de l'image antérieure. Nous parlons des vecteurs de mouvement qui représentent l'information de mouvement.

Nous distinguons beaucoup d'algorithmes d'estimation du mouvement qui utilisent des techniques de mise en correspondance des blocs. Ainsi, le vecteur de mouvement est obtenu par la réduction d'une fonction de coût mesurant la disparité entre un bloc candidat et le bloc en cours. Nous citons l'algorithme d'estimation du mouvement le plus utilisé le SAD (la somme des différences absolues). Il y a aussi des algorithmes d'estimation du mouvement à des valeurs inférieures au pixel ce qui améliore la qualité de reproduction. Les normes les plus récentes de codage vidéo prévoient une précision d'un ½ de pixel.

La prévision des macroblocs est effectuée à partir des images antérieures (P) ou des images antérieures/postérieures (B). Le meilleur rendement en compression est achevé avec des macroblocs B qui admettent un délai de décodage supplémentaire. La compensation du mouvement est exécutée sur des blocs (16x16 ou 8x8). A sa suite, nous appliquons une transformée, comme nous l'expliquons dans la prochaine section. Le choix de mode de codage consiste à choisir quel est le mode de prédiction qui fournit la meilleure qualité. Pour le codage H.264/AVC, nous spécifions ces modes de codage : 9 modes de prédiction Intra pour les blocs de taille 4x4 (Figure 2.7), 4 modes de prédiction Intra pour les blocs de taille 16x16 (Figure 2.6), 4 partitions possibles pour les modes de prédiction Inter et 4 sous-partitions, 3 prédictions MBAFF Field/Frame et n images de références possibles.

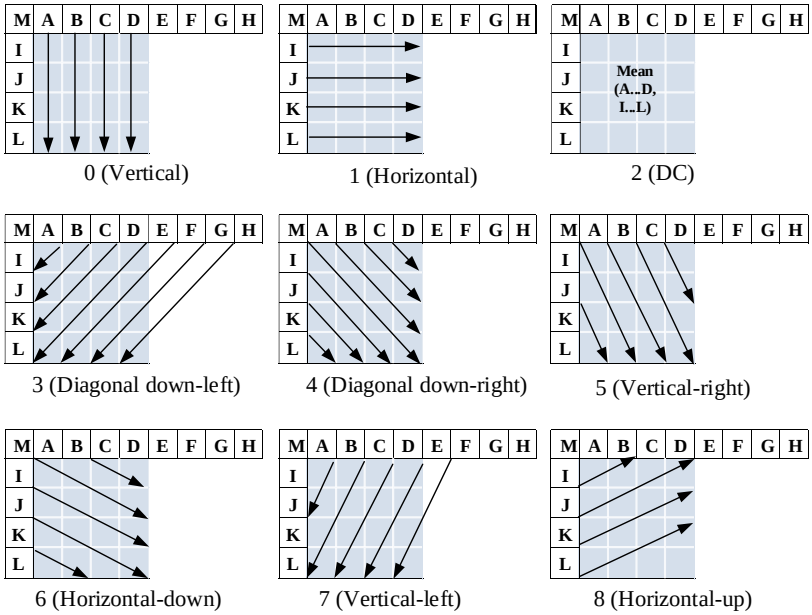


Figure 2. 6. Modes de la prédiction Intra 4x4.

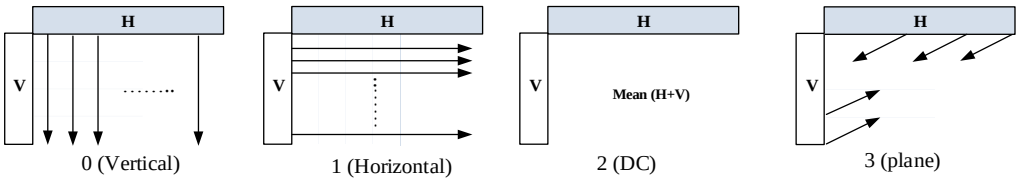


Figure 2. 7. Modes de la prédiction Intra 16x16.

La compensation de mouvement est opérée par rapport à plusieurs images de référence déjà codées. Ceci permet habituellement des améliorations modestes au niveau du débit et de la qualité dans la plupart des scènes. Mais dans certains types de scènes (contenant des flashs rapides et répétitifs), il permet une réduction significative du débit.

Pour la compensation de mouvement, l'utilisation de 7 tailles de blocs permet une segmentation précise des zones en mouvement et la précision au quart de pixel permet une description précise du déplacement de ces zones. Pour la chrominance, la précision se fait au huitième de pixel.

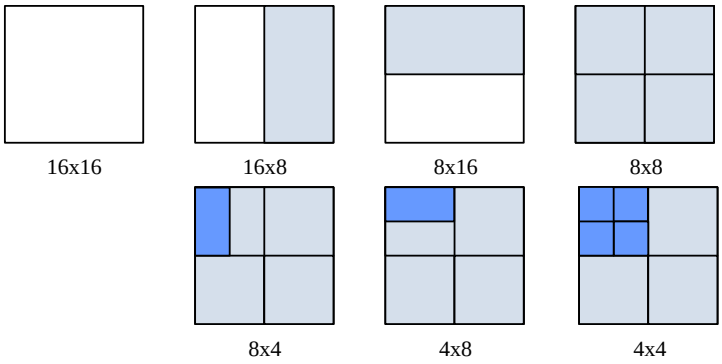


Figure 2. 8. Décomposition d'un macrobloc en partitions (16x16, 8x16, 16x8, 8x8) et sous-partitions (4x8, 8x4, 4x4).

Si une partition (16x16, 16x8 ou 8x16) est choisie, un nombre réduit de bits est utilisé pour coder les vecteurs de mouvement mais le résidu peut être conséquent. Pour les petites partitions, nous obtenons de faibles résidus mais cela nécessite une grande quantité d'information pour représenter le mouvement. Il existe ainsi de nombreuses combinaisons possibles et le choix de la taille des partitions est déterminant pour les performances d'encodage. Des grandes partitions conviennent pour les zones homogènes de l'image alors que des partitions réduites sont utilisée dans les zones détaillées.

3.2.2.3. La quantification et le parcours des données

La quantification est la phase de réduction d'information afin de bien contrôler le débit binaire. Nous pouvons l'appliquer dans différents niveaux pour les blocs. On parle de matrices de quantifications indépendantes. Nous distinguons aussi la quantification adaptative dans laquelle certains blocs seront privilégiés par rapport à d'autres.

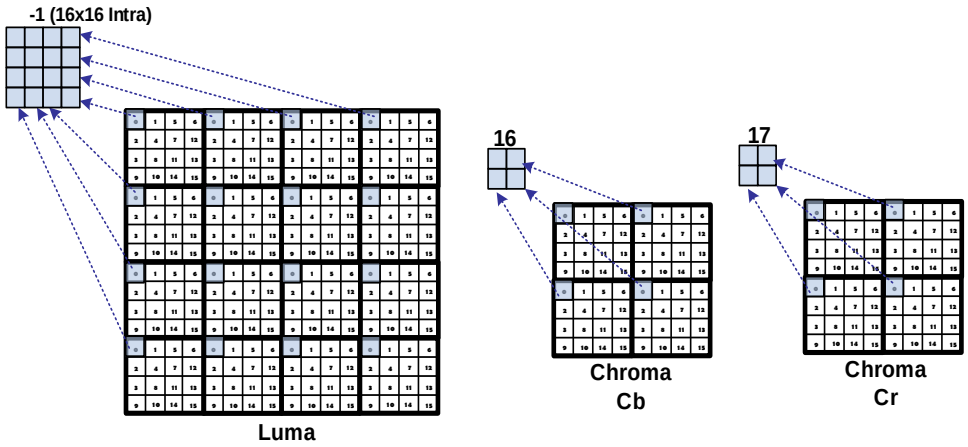


Figure 2. 9- Ordre de codage des blocs dans un macrobloc

Les pas de quantification définis dans la norme s'incrémentent d'une valeur de 12,5% et leur dynamique est augmentée puisque les valeurs vont de 1 à 52. Dans les normes vidéo précédentes, le pas de quantification Qstep augmente par pas fixes Qp. Cela entraîne des zones inaccessibles pour certains quantificateurs. De plus, afin d'obtenir de meilleurs résultats visuels, la quantification de la chrominance est plus fine que celle de la luminance.

Tableau 2. 1- Tableau d'équivalence entre le paramètre QP et le pas de quantification

Qp	**0**	**1**	**2**	**3**	**4**	**5**	**10**	**12**	**…**	**18**	**…**	**30**	**…**	**51**
Qstep	0.625	0.6875	0.8125	0.875	1	1.125	2	2.5	…	5	…	20	…	224

3.2.2.4. Le filtre réducteur d'effets de blocs

La norme H.264/AVC intègre une étape de filtrage qui améliore l'efficacité de la compression et la qualité visuelle des séquences vidéo. Cette étape est en charge de l'élimination des artéfacts visuels indésirables tels que les effets de bloc. Après la reconstruction de l'image, le processus de filtrage lisse les bords horizontaux et verticaux des macroblocs 16x16 et des blocs 4x4. En outre, le filtre anti-blocs (Loop filter ou Deblocking Filter) est le même utilisé au niveau de l'encodeur qu'au niveau du décodeur.

3.2.3. Le codage entropique dans la norme H.264

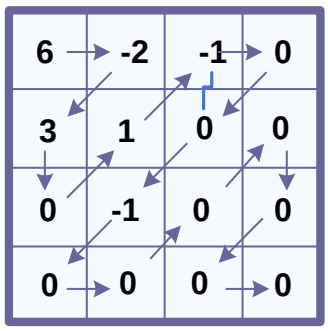
Le codage entropique peut être réalisé de deux manières différentes dans la norme H.264. En effet, nous utilisons deux codages adaptatifs de type Huffman à longueur variable CAVLC (Context-Adaptive Variable Length Coding) et arithmétique binaire CABAC (Context-Adaptive Binary Arithmetic Coding) pour le codage des éléments syntaxiques.

Le premier est une alternative pour le codage des tables de coefficients de transformation [HAN 05] [Park 06] ce deuxième est une combinaison d'une technique adaptative de codage arithmétique binaire qui permet un degré élevé d'adaptation et une réduction plus importante de redondance [Marpe 03].

Une dernière technique simple et hautement structurée de codage à longueur variable est utilisée pour quelques éléments syntaxiques bien distingués, considéré comme Exp-Golomb. Il adopte un format unique (préfixe, suffixe) étendu à la taille nécessaire pour l'élément syntaxique considéré.

3.2.3.1. Le codage adaptatif à longueur variable

Pour coder les coefficients résiduels transformés et quantifiés des blocs de luminance et de chrominance nous utilisons la méthode CAVLC. En exploitant les probabilités d'occurrence de chaque symbole ou séquence de symboles à émettre, nous pouvons leur associer un mot binaire d'une longueur d'autant plus courte que leur occurrence est grande. Cela qui entraîne une énorme compression. Il utilise plusieurs tables VLC pour chaque élément syntaxique. Il s'adapte au contexte actuel en sélectionnant l'une des tables VLC, compte tenu des éléments de syntaxe déjà transmis. Ci-dessous un exemple qui décrit les étapes de codage.



Scan zig_zag	*Evènements*	*Codes VLC*
	6 = (0,0,6)	00100010
	-2 = (0,0,-2)	01001
	3 = (0,0,3)	001010
	1 = (0,1,1)	0110
	-1 = (0,0,-1)	111
	-1 = (1,2,-1)	01011

Cette adaptation offre de meilleures performances au codage entropique en comparaison au codage entropique utilisant une seule table VLC. En outre, CAVLC utilise des techniques de codage telles que niveau d'exécution et de suivi [Richardson 10]. Cependant, le gain de codage est livré avec une augmentation de la complexité, ce qui rend difficile la mise en œuvre de cet algorithme en temps réel.

3.2.3.2. Le codage arithmétique adaptatif contextuel

L'utilisation du codage arithmétique adaptatif CABAC permet d'assigner un nombre de bits fractionnaire à chaque symbole à coder dont les probabilités seront supérieures à 0,5. Les codes adaptatifs générés sont alors adaptés aux statistiques des symboles variables. En outre, nous mentionnons l'adaptation du CABAC au contexte puisque les statistiques des symboles codés sont utilisées pour évaluer les probabilités conditionnelles. Ce qui lui permet d'alterner des modèles de probabilités estimés. Dans la norme H.264/AVC, le moteur fondamental du codage arithmétique et son estimation de probabilités sont définis comme des méthodes de faible complexité, sans multiplication, qui utilisent uniquement les commutations et les

vérifications de tables LUTs. Nous atteignons de bonnes performances de compression grâce à la sélection des modèles de probabilité pour chaque élément de syntaxe selon le contexte de l'élément et l'adaptation probabilité des estimations basées sur les statistiques locales.
Le codage d'un symbole de données comprend les étapes suivantes.

- Binarisation : Seules les décisions binaires (1 ou 0) sont codées. Un symbole évalué (par exemple, un coefficient de transformer ou de vecteur de mouvement) est converti en un code binaire avant le codage arithmétique. Ce processus est similaire au processus de conversion d'un symbole de données en un code de longueur variable, mais le code binaire est également codé, par le codeur arithmétique, avant la transmission. Ensuite, les étapes suivantes sont répétées pour chaque bit du symbole binarisé.
- Sélection de modèles Contexte : Un modèle de «contexte» est un modèle de probabilité pour un ou plusieurs bacs du symbole binarisé. Ce modèle peut être choisi parmi une sélection de modèles disponibles en fonction de la statistique des symboles de données récemment codées.
- Codage arithmétique : Un codeur arithmétique encode chaque symbole en fonction de la probabilité sélectionnée modèle, conformément à l'algorithme donné dans le chapitre 1 section 3.4.2. Notez qu'il y a seulement deux sous-gammes pour chaque cellule (correspondant à «0» et «1»).
- Mise à jour des probabilités : Le modèle de contexte sélectionné est mis à jour, fondé sur la valeur réelle codée (par exemple si la valeur a été "1", la fréquence des "1" augmente).

Le codage arithmétique a été évoqué avec Shannon dans la théorie de l'information et publiée par Abramson dans les années 1960. Diverses mises en œuvre pratiques ont été proposées et de nombreux versions d'algorithmes de codage arithmétique améliorations ont été publiées [Witten 87] [Howard 92]. Ainsi, toutes les versions proposées se basent sur le même principe : au lieu d'attribuer à un mot de code un symbole (ou plusieurs symboles) de la source, on subdivise un intervalle de référence pour définir une valeur représentative du message entier. Cette valeur est alors émise sous sa représentation binaire et est décodée en réception par l'opération de subdivision d'intervalles inverse. L'intérêt de cette approche est que chaque symbole (ou groupe de symboles) de la source se retrouve donc code par un nombre rationnel de bits. Ceci explique le succès des codes arithmétiques qui permettent grâce à cela d'approcher l'entropie de la source avec des complexités d'encodage et de décodage linéaires par rapport à la longueur de la séquence. De plus les probabilités des symboles peuvent être fournies, permettant ainsi d'introduire des modèles adaptatifs. Nous pouvons citer :

(a) Q Codeur

L'algorithme Q-codeur d'IBM adopté en standard JBIG pour la compression d'images. Q-codeur simplifie l'alphabet sur seulement deux signes. Cette procédure permet un rapprochement à la répartition d'intervalle avec des additions au lieu des multiplications comme avec le joint de grade.

(b) MQ Codeur

Le codeur arithmétique MQ est utilisé dans JPEG2000. Il prend comme entrées les valeurs binaires et les contextes associés de modélisation binaire des coefficients dans l'ordre des passes de codage. Ce codeur dérive du codeur QM mis au point par IBM et déjà utilise de façon alternative à un codage de Huffmann, dans le standard JPEG (ISO/IEC 10918-1).

(c) M Codeur

M-codeur est le codeur arithmétique du CABAC pour H.264/ AVC et HEVC. Il utilise des techniques bien spécifiques et il intègre des modèles contextuels très nombreux par rapport aux codeurs anciens. La méthode de « «Free-multiplication » permet une complexité bien réduite de calcul.

4. Codage vidéo au travers la norme HEVC

Le standard HEVC est né fin janvier 2013. Il a été développé par l'UIT-T/ VCEG ainsi que l'ISO/CEI MPEG. Il garantit un gain de compression très important indépendamment de la résolution des vidéos avec notamment plus d'efficacité dans des vidéos Ultra HD. L'industrie adopte cette norme largement pour permettre la résolution plus élevée de vidéo 4K.

Cinq points de conformités ont été définis pour HEVC : HD 1080p (8 bits, 50fps), HD 1080p (8 bits, 60 images par seconde), HD 1080p (8-bit ou 10-bit, 50fps), HD 1080p (8-bit ou 10-bit, 60 images par seconde) et 2160p UHD-1 Phase 1(8-bit ou 10-bit, 50 ou 60 images par seconde). Pour le HEVC, les profiles incluent désormais :

- Profil principal et principal 10
- Profil principal images fixes
- Profil Multiview : Principal Multiview
- Profils évolutifs : Principal évolutif et Principal 10 évolutif
- Profils des extensions de la gamme de formats : Monochrome, Monochrome 12, Monochrome 16, Principal 12, Principal 4:2:2 10, Principal 4:2:2 12, Principal 4:4:4, Principal 4:4:4 10, Principal 4:4:4 12, Principal Intra, Principal 10 Intra, Principal 12 Intra, Principal 4:2:2 10 Intra, Principal 4:2:2 12 Intra, Principal 4:4:4 Intra, Principal 4:4:4 10 Intra, Principal 4:4:4 12 Intra, Principal 4:4:4 16 Intra, Principal 4:4:4 images fixes et Principal 4:4:4 16 images fixes

- Extensions de la gamme de formats, profil haut-débit : Haut-débit 4 :4:4 16 Intra.

Comme H.264, ces profils se décomposent en niveaux définissant des contraintes pour certains paramètres du flux de données. Nous distinguons les niveaux les plus caractéristiques : le 4 (pour le full HD 1080p), le 5 (pour l'UHD-1 en 2160) et les niveaux ≥ 6 pour la future UHD-2 en 4320 x7680. À partir du niveau 4, chaque niveau dans le HEVC se décline en deux tiers (Main et High) pour des applications ne différant que par le débit max ou la taille du buffer.

4.1. Etude comportementale du codeur de référence

Nous commençons par le codage d'une image Intra qui ne fera référence à aucune autre, contrairement au codage des images Inter. Le codage est effectué en utilisant un ensemble de prédiction spatiale à l'intérieure de l'image traitée. Les images inter ne seront pas codées mais uniquement décrite à l'aide des blocs de prédiction (depuis l'image Intra) auxquels s'appliquent des vecteurs de mouvements.

Nous appliquons ensuite une DCT au résiduel dont le résultat sera après transmis avec les informations de prédiction. Afin d'améliorer la qualité et d'enlever les artefacts et autres défauts de prédiction, le codeur décode les images qu'il vient de coder. Par la suite, il les compare et utilise les différences pour finaliser le flux binaire de sortie. Le schéma bloc simplifié de HEVC est illustré dans la figure 2.10.

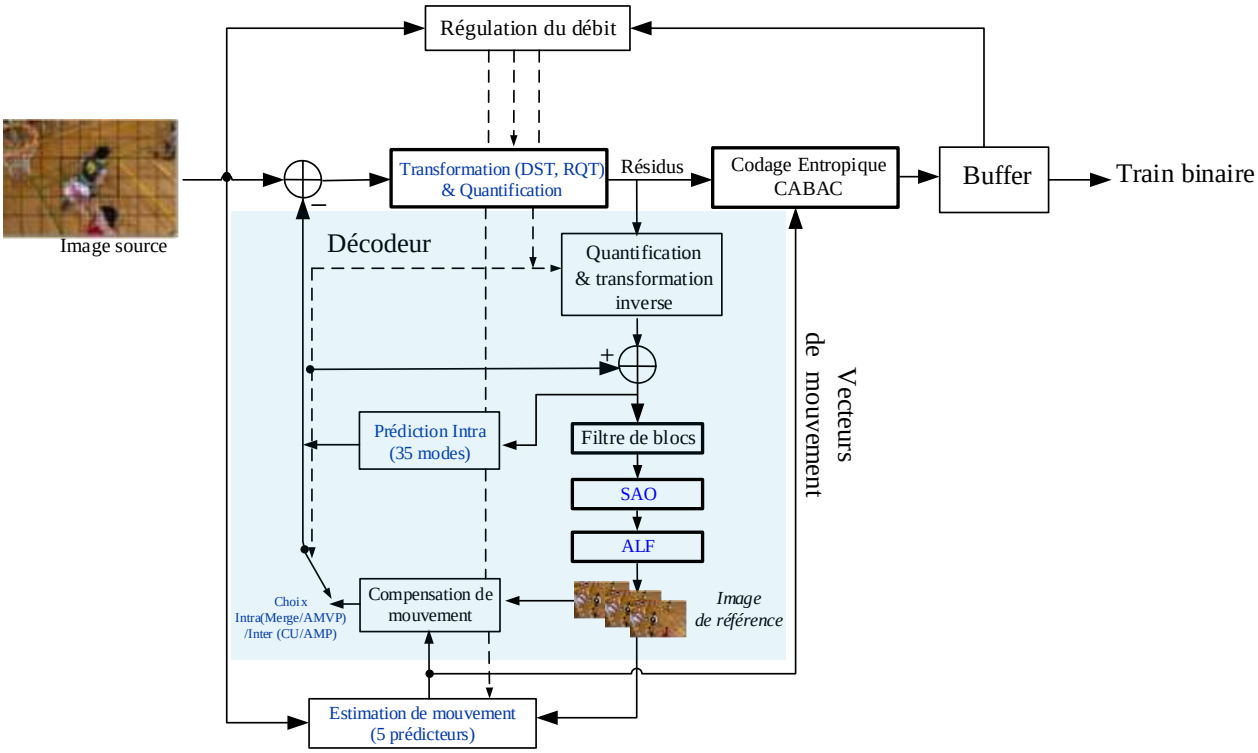


Figure 2. 10- Chaîne de codage vidéo HEVC

4.2. Techniques propres de HEVC

Le fonctionnement du codeur HEVC repose globalement sur les principes du H.264 avec quelques améliorations. Il ne normalise pas l'encodage pareillement aux normes de MPEG. La prédiction inter-image évolue, notamment sur les vecteurs de mouvement indiquant le déplacement des pixels à effectuer sur l'image de référence. Un nouveau filtrage appliqué en sortie de l'image, en plus du filtrage anti-block, est désormais appliqué afin de réduire les effets de bloc. Parmi les améliorations contenues dans HEVC, il y a par exemple le découpage par blocs de différentes tailles, des techniques de « prédictions » permettant d'économiser quelques bits ajoutées. HEVC utilise le principe d'arrangement des données. Elle sépare les unités pour le codage, la prédiction et la transformée. Une unité est tout simplement l'ensemble de donnée affecté à une même région rectangulaire dans la vidéo. L'élément fondamental de codage pour le HEVC est la CTU (Coding Tree Unit) et chaque CTU est divisée en CU (coding units) et chaque CU est divisé en PU (prediction units) et en TU (transform units).

Les CTU peuvent être regroupées en slices afin d'assurer la resynchronisation après une perte de donnée. Nous avons la possibilité d'appliquer un traitement parallèle avec le WPP (wavefront parallel processing) pour ces slices comme illustré dans la figure 2.11.

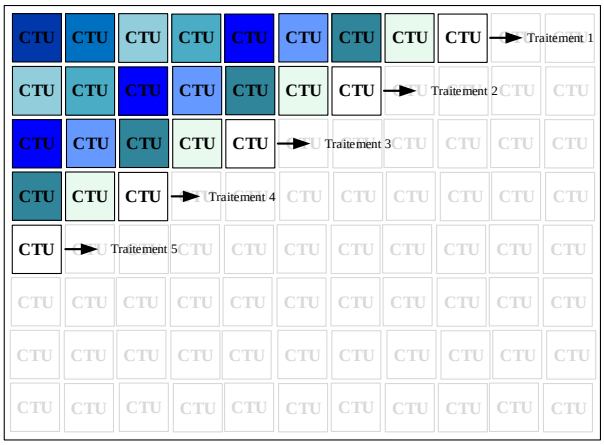


Figure 2. 11- Traitement parallèle avec le WPP en HEVC

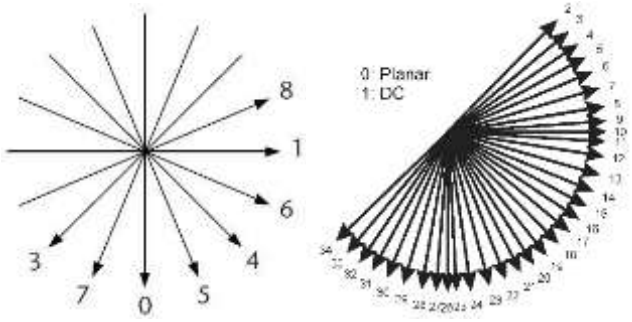
Le codage parallèle traite les CTU en parallèle ligne par ligne. Le traitement d'une ligne peut commencer dès que les deux premières CTU de la ligne immédiatement supérieure sont terminées. Il y a toujours un retard minimum de 2 CTU entre deux lignes consécutives pour conserver les dépendances. Ceci permet d'avoir disponible les CTU voisines pour la prédiction Intra ainsi que la prédiction du vecteur de mouvement. Pour le HEVC, nous distinguons 35 modes de codage Intra pour la luminance (33 directions, Intra DC, Intra Planar). Cela est indépendant de la taille du bloc (entre 4 × 4 et 32 × 32). Pour le PU chroma

on peut choisir seulement entre Planar, V, H, DC et Direct. Le mode chroma est codé directement (sans MPM). Quant au Codage Inter, la taille des blocs est variable, avec structures asymétriques possibles. Comme dans H.264, la prédiction Inter peut être faite en utilisant des références multiples dans une liste. Nous trouvons deux listes pour les slices de type B. Un nouveau mode « Merge » permet de partager l'information de mouvement entre les PU voisins. Il permet de signaler pour une PU que son vecteur de mouvement doit être hérité d'une PU voisine parmi un ensemble de PU candidate à la manière de l'outil précédent. Ce mode permet aussi de corriger un défaut lié à la représentation en arbre quaternaire qui a tendance à sur-partitionner l'image. Une PU codée avec le mode Merge économise alors le coût du vecteur de mouvement ainsi que l'indice de l'image de référence, elle ne nécessite alors que la transmission des résiduels de textures.

Enfin, nous mentionnons que le filtre anti-bloc est toujours utilisé et bien que les tailles plus grandes utilisées dans HEVC réduisent partiellement ce défaut. Deux autres filtres sont aussi définis, le SAO (Sample-Adaptive Offset Filter) et l'ALF (Adaptive Loop Filter). Le SAO est appliqué après le filtrage anti-blocs. Il classifie les pixels de la LCU (largest coding unit) en fonction de leurs intensités puis à appliquer un offset pour chaque catégorie qui est dérivé par rapport à la LCU originale et transmis au décodeur. Pour l'ALF, nous l'appliquons avant la copie de l'image dans le buffer ce qui permet d'améliorer la qualité de l'image.

Pour résumer, le standard HEVC reprend les grands principes de la norme H.264 tout en apportant d'importantes améliorations. Tout d'abord, nous avons une meilleure subdivision des images avec les Coding Tree Blocks d'une taille maximale de 64x64 pixels. Cette subdivision des images est encore améliorée en adoptant une structure arborescente. En effet, les blocs les plus larges augmentent l'efficacité de la compression des images surtout UHD.

En outre, les prédictions intra images sont plus efficaces. Nous distinguons 35 modes pour le HEVC contre 9 modes de détection de redondance spatiale avec le H.264. Le traitement est plus compliqué mais la compression sera meilleure.



(a) (b)

Figure 2. 12- Modes de prédiction Intra (a) H.264 (b) HEVC

Aussi, nous avons une meilleure codification des coefficients résiduels (CABAC) ce qui améliore ainsi le signal sans augmentation du bitrate. Le filtrage est plus performant avec l'ajout d'un nouveau filtre SAO (Sample Adaptive Offset). Et enfin, il y a des nouvelles fonctionnalités qui favorisent le traitement parallèle. Ces améliorations rendent le HEVC plus efficace et puissant avec une complexité de la compression et une charge pour les CPU supérieures.

4.3. Evaluation des performances de H.264 et HEVC

4.5.1. Résultats expérimentaux

Nous avons testé l'encodage de la séquence Blowing Bubbles (416x240) dont le « frame rate » est égal à 50 avec l'HM 10 (encodeur HEVC) pour 3 pas de quantification différents : 22, 31 et 51 comme illustré dans la figure 2.13.

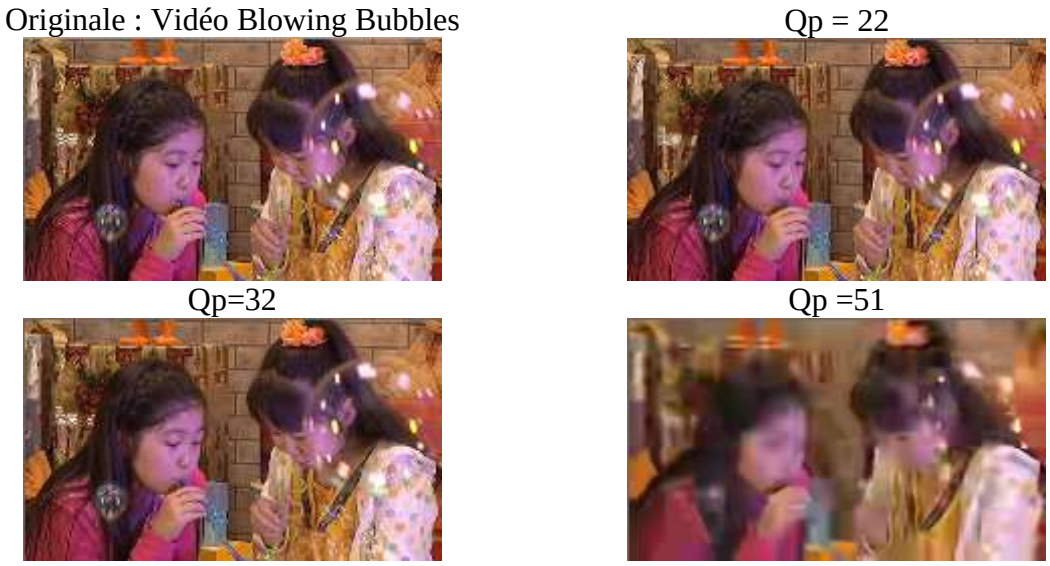


Figure 2. 13- Comparaison d'image originale avec des images compressés par l'HM 10 pour différents pas de quantification

Nous relevons les débits et taux de compression associés (Tableau 2.2). Nous obtenons des débits qui varient de 2170 kbps à 35 kbps avec des taux de compression de 28:1 à 1680:1.

Tableau 2. 2 Gain en compression d'HM 10 pour la séquence de test BlowingBubbles

Pas de quantification	débit	Taux de compression
Non compressé	60 Mbps	
22	2170 kbps	28 :1
32	568 kbps	105 :1
51	35 kbps	1680 :1

Après, nous avons utilisé le JM 18 (encodeur H.264) pour le comparer avec l'HM 10. Nous nous servons des différents séquences de test avec divers résolutions et caractéristiques (qui

contiennent du mouvement rapide ou/et lent, des textures complexes ou simples, avec des composants synthétiques). Ces séquences vidéo utilisées sont :

- Basketball pass, BQsquare, Bowling bubbles, Racehorses (416x240)
- Basket Ball Drill, PartyScene, BQMall, Racehorses (832x480)
- Vidyo1, Vidyo3, Vidyo4 (1280x720)
- BQterrace, Cactus, Kimonio, ParkScene (1920x1088)

Nous pouvons observer les différences entre les codeurs selon plusieurs critères : PSNR pour la composante luminance (Figure 2.14, Figure 2.16, Figure 2.18, Figure 2.20) et le temps de calcul (Figure 2.15, Figure 2.17, Figure 2.19, Figure 2.21). En effet, nous avons présenté les gains en débit et les PSNRs correspondantes pour un codage CABAC et 100 frames à coder. Les simulations menées permettent de comparer les performances en termes de qualité, débit et temps de codage de H.264 vs HEVC.

- la différence en termes de qualité est bien significative. En effet, les courbes présentées au-dessous montrent que le débit obtenu avec le standard HEVC est en moyenne 2 fois plus réduit que celui obtenue par le standard H.264. Cette tendance est bien vérifiée avec divers séquences et résolutions.
- le temps de codage de HEVC est plus important que celui du H.264, ceci confirme la complexité algorithmique de nouveau codec. Ainsi, il est proportionnel au nombre de frames à coder. Si le nombre d'images à coder a doublé, le temps l'a aussi. La qualité n'est pas liée à ce paramètre. Nous remarquons aussi que plus la séquence est mobile, plus le temps de codage est long. Enfin, les tests nous permettent aussi de conclure que la qualité n'est pas altérée par le format de l'image à coder.

Des comportements similaires sont retrouvés pour les différentes vidéos. Les figures au-dessous montrent l'efficacité de la compression HEVC en terme du PSNR luminance / débit par rapport aux MPEG-4 AVC. En utilisant le codeur HEVC, le gain en débit est approximativement de 50%.

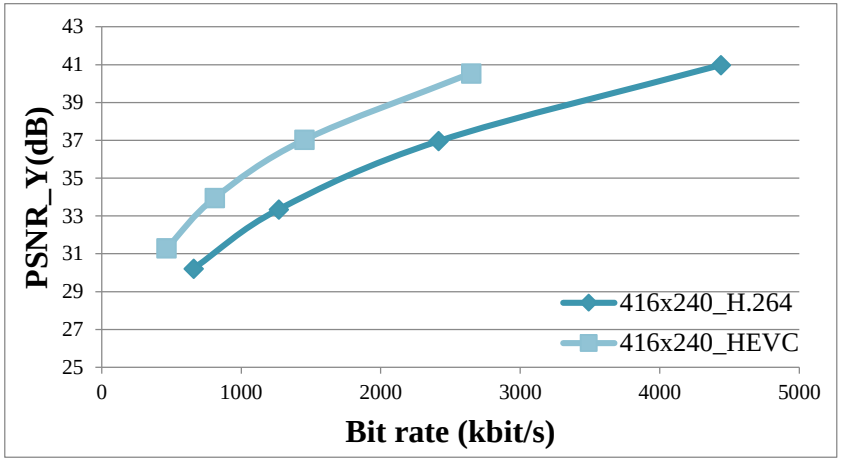


Figure 2. 14 Evaluation de performances pour des séquences de test 416x240

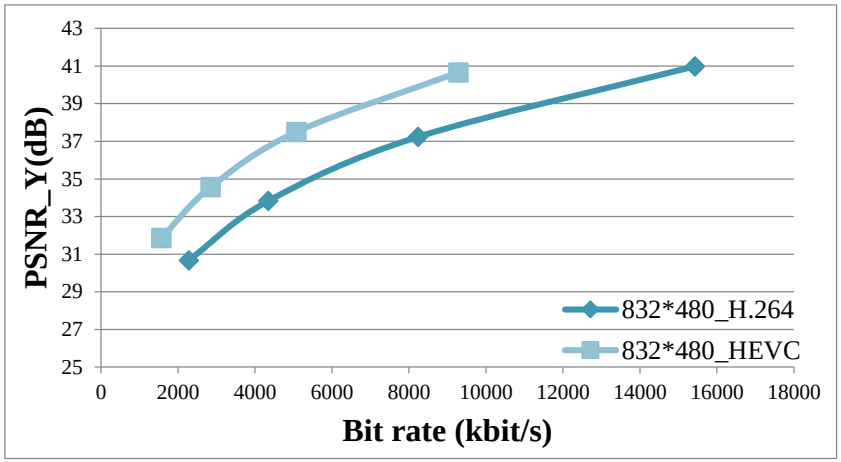


Figure 2. 15 - Evaluation de performances pour des séquences de test 832x480

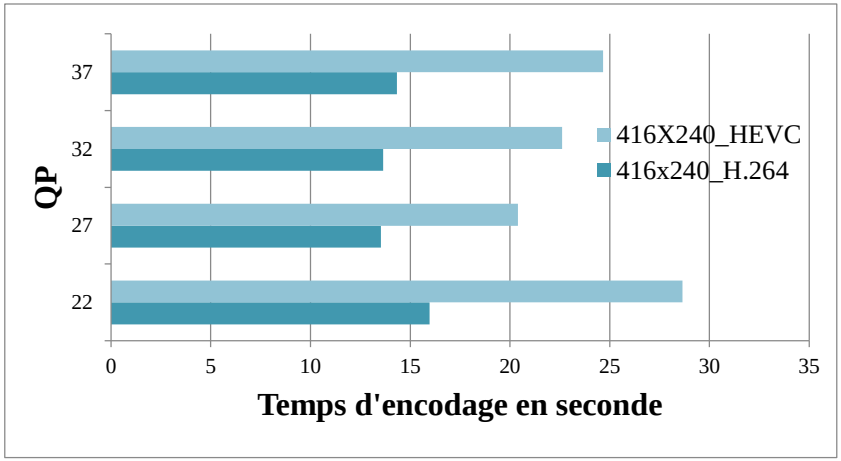


Figure 2. 16- Temps d'encodage avec divers pas de quantification pour des séquences de test 416x240

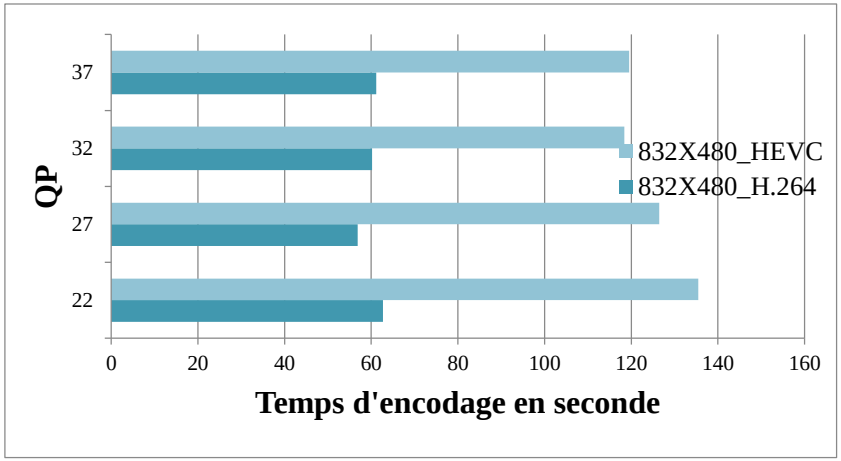


Figure 2. 17- Temps d'encodage avec divers pas de quantification pour des séquences de test 832x480

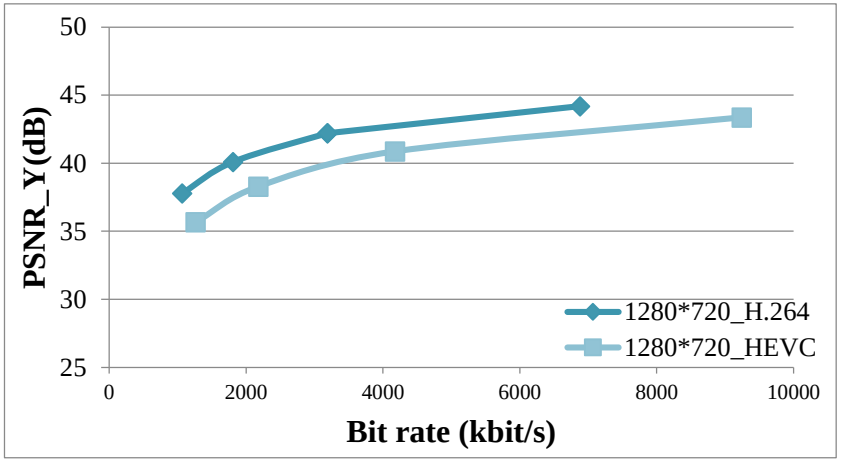


Figure 2. 18- Evaluation de performances pour des séquences de test 1280x720

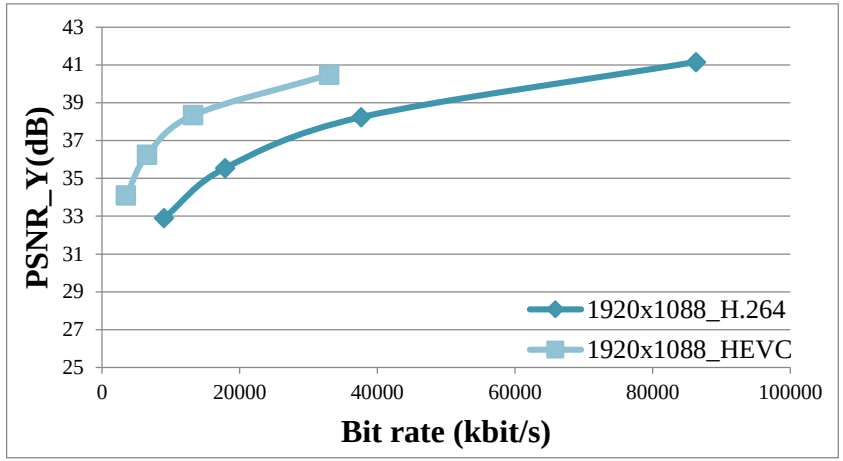


Figure 2. 19- Evaluation de performances pour des séquences de test 1920x1088

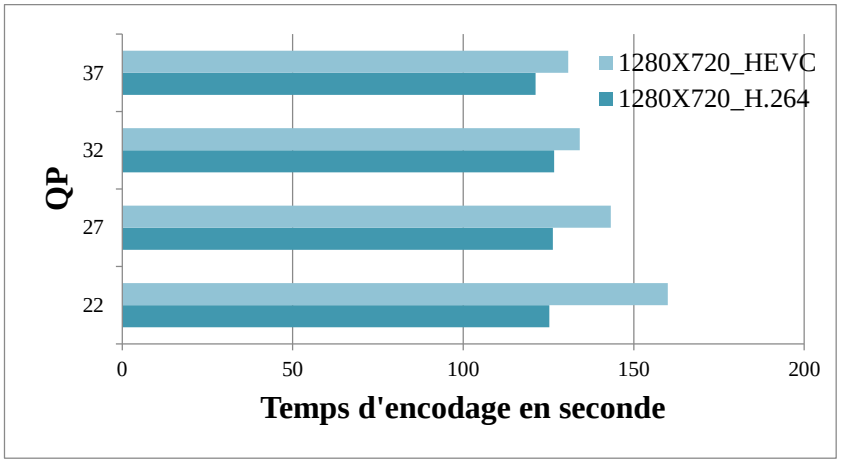


Figure 2. 20- Temps d'encodage avec divers pas de quantification pour des séquences de test 1280x720

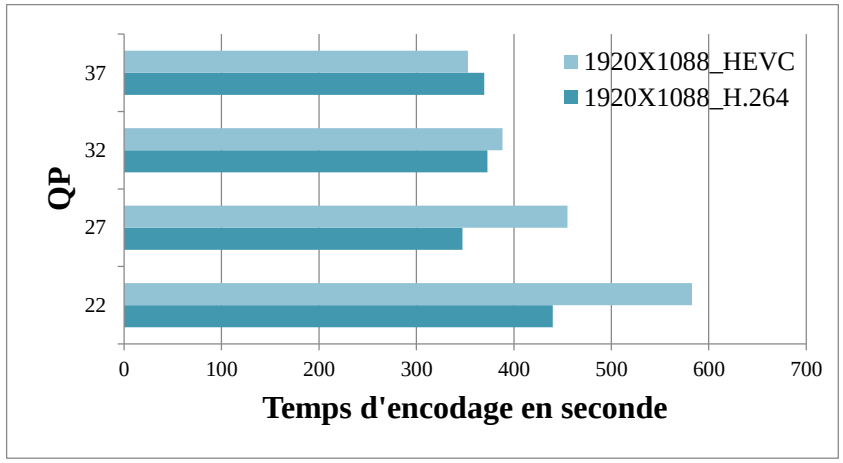


Figure 2. 21- Temps d'encodage avec divers pas de quantification pour des séquences de test 1920x1088

4.5.2. Etude bibliographique de la complexité dans les décodeurs H.264 et HEVC

La complexité d'implantation pour n'importe quelle application dépend forcément de l'occupation de la surface en silicium et le temps d'exécution. Ces paramètres ont un impact direct sur la surface, le coût et aussi la consommation d'énergie des circuits. Pour cela, il est nécessaire d'identifier les modules les plus complexes d'une norme pour pouvoir les intégrer matériellement afin de minimiser le temps d'exécution.

Ainsi, divers analyses basées sur le profilage du décodeur H.264/AVC et HEVC ont été réalisées dans la littérature. D'après les résultats présentés dans [Chebbeh 2012], nous constatons que la complexité calculatoire est fortement localisée au niveau décodage entropique (28 % du temps processeur) et au niveau transformation et quantification inverse (26%). Le reste du temps de calcul est réparti entre le filtrage anti-blocs, la prédiction et la reconstruction. Par ailleurs, la figure 2.12 illustre le profilage du décodeur H.264/AVC et la figure 2.13 présente un profilage du décodeur HEVC [Bossen 12]. Cette analyse de la complexité calculatoire du décodage HEVC est différente à celle de H.264/AVC. En fait, la compensation de mouvement représente 43% du temps de calcul, puis le décodage entropique (24%),

Alors, nous pouvons affirmer que le décodage entropique constitue une partie bien complexe dans le décodage de flux vidéo (H264 et HEVC). A partir de ce constat, nous avons décidé d'implanter ce bloc (chapitre 3) afin d'améliorer les performances temporelle de ce traitement.

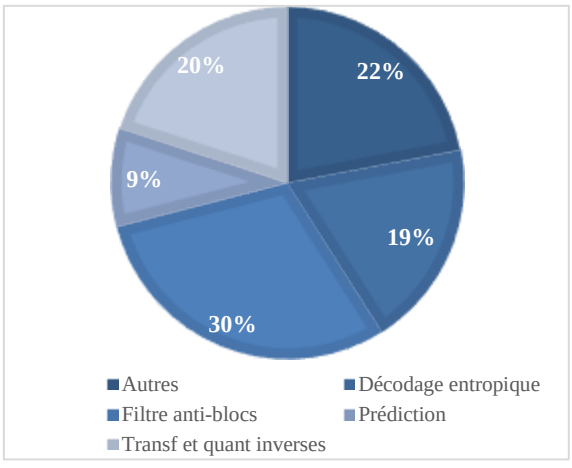


Figure 2. 22- Profilage du décodeur H.264/AVC [Chebbeh 2012]

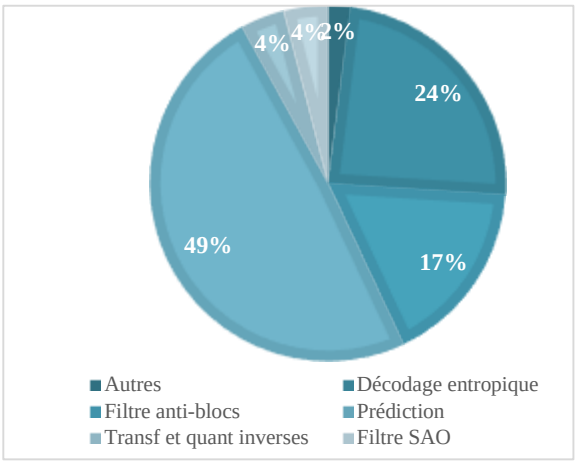


Figure 2. 23- Profilage du décodeur HEVC [Bossen 12]

5. Conclusion

Dans ce chapitre, nous avons étudié brièvement les dernières normes de codage vidéo. Ensuite, nous avons détaillé le fonctionnement des techniques de compression associées aux normes H.264/AVC et HEVC. Ces derniers standards fournissent des performances nettement supérieures en comparaison aux normes antérieures. Lors des tests, nous avons utilisé des codeurs logiciels H.264/AVC et HEVC. HEVC inclut de nombreuses techniques nouvelles qui permettent de compresser les vidéos beaucoup plus efficacement que les normes et fournit plus de flexibilité aux applications en environnement réseau. Bien que le HEVC n'apporte pas une rupture technologique par rapport aux normes de codage vidéo précédentes, il comporte plusieurs modifications impliquent des calculs très complexes. L'analyse de la complexité algorithmique à travers les résultats de profilage montre que le codeur entropique est le goulot d'étranglement au niveau de la complexité matérielle. Dans le chapitre suivant, nous nous proposons de décrire une solution hardware considéré comme accélérateur matériel.

Chapitre 3 : Codage entropique CABAC

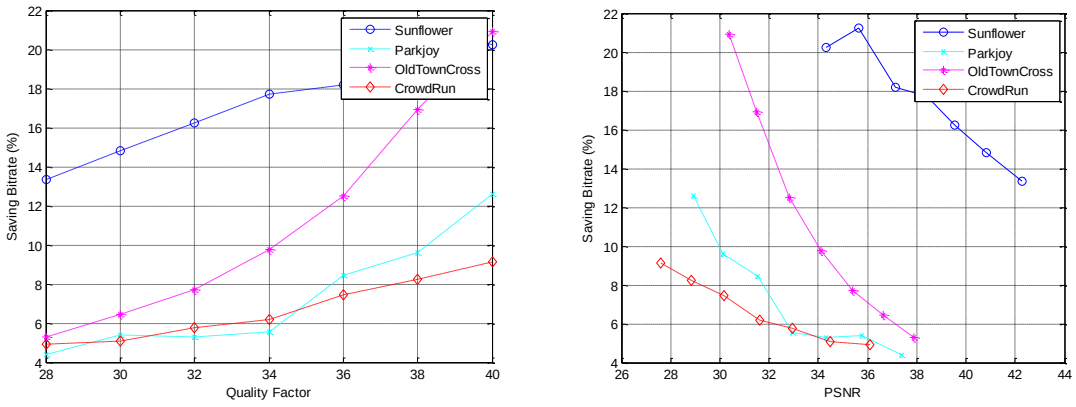
1. Introduction

Les choix de codage entropique pour H.264 peuvent être sélectionnés à partir soit de CABAC (codage arithmétique binaire adaptatif selon le contexte), soit de CAVLC (codage adaptatif de longueur variable selon le contexte). Nous voulons dire par contexte «les conditions environnantes » et nous parlons d'un codage adaptatif qui atteint un niveau élevé d'efficacité de la compression en ajustant les paramètres du codage pour qu'ils correspondent le plus possible au contenu de l'image. Le tableau 3.1 compare les méthodes de codage entropique.

Tableau 3. 1. Comparaison des méthodes de codage entropique

	MPEG-2 VLC	MPEG-4 AVC/H.264	
		CAVLC	CABAC
Méthode de codage	Codage à longueur variable		Codage arithmétique
Procédé d'adaptation au contexte	Non	Unité : coefficients DCT	Unité : 1 bit
Efficacité de codage	Moyenne	Très bonne	Excellente
Plus importantes applications	DVD télévision numérique	Téléphone cellulaires vidéodiffusion	Blu-ray, HD-DVD, AVC-HD de nouvelle génération

MPEG-2 ne prend pas en charge cette fonction d'adaptation au contexte et effectue le codage entropique par simple VLC au moyen d'une table de codes fixes. Le VLC simple n'est pas optimal et l'efficacité de compression décroît en présence d'une structure d'image de plus en plus complexe. En revanche, CAVLC et CABAC peuvent compresser les images de manière hautement efficace en ajustant leur action en fonction de chaque image avec une grande précision. La figure suivante présente les performances des deux codeurs adaptatifs en utilisant le JM 16.0 au niveau 4.



(a) (b)

Figure 3. 1. Performance de CABAC vs CAVLC pour des séquences HD
(a) avec QF variable ; (b) avec PSNR variable

Nous montrons la moyenne du gain en compression pour différents séquences vidéos (Crowd run, Park joy, Old town cross, Sun flower) pour divers facteurs de qualité (28-40) (Fig.1-a) et PSNR (34-43 db) (Fig.1-b). Un gain de performance à l'ordre de 4-21% est considéré par rapport à la version CAVLC (réduction de débit pour une qualité égale). Alors, CABAC est capable d'effectuer un codage entropique idéal presque aux limites des prévisions théoriques en raison de la combinaison du codage arithmétique avec une modélisation du contexte. CABAC s'avère une contribution importante à l'efficacité des algorithmes de codage HD c'est pour cela qu'il est retenu par le dernier standard de codage vidéo HEVC.

2. Etude du CABAC

CABAC combine un codage arithmétique binaire adaptatif avec une modélisation du contexte. L'encodage d'un élément par CABAC se déroule en trois étapes principales : la binarisation, la modélisation du contexte et le codage arithmétique binaire**.**

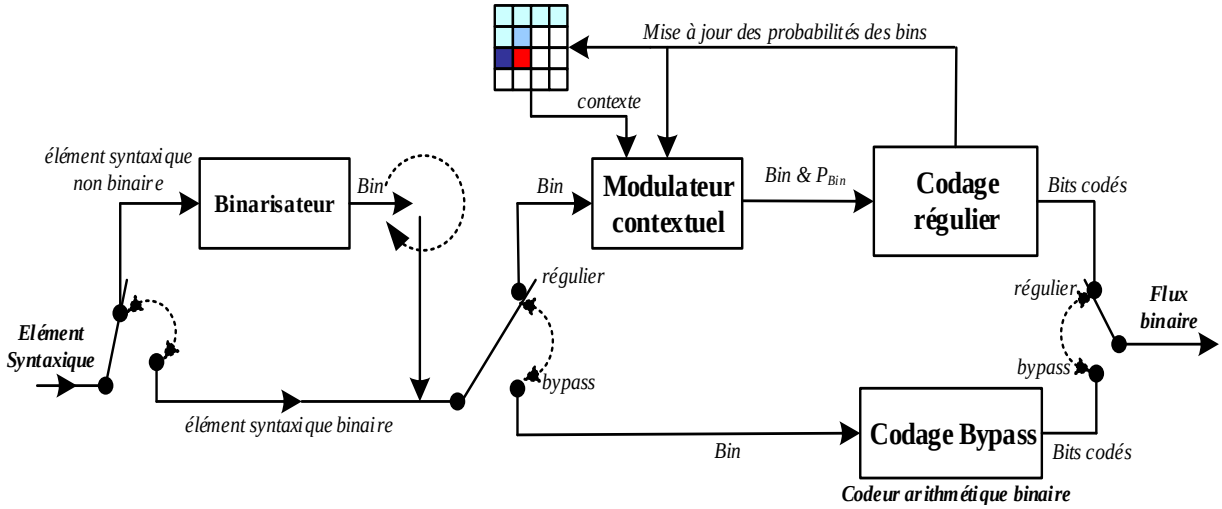


Figure 3. 2. Schéma général du codage CABAC

2.3. La binarisation

Elle permet de diminuer la taille des éléments à coder en représentant chacun des éléments à coder par un mot binaire unique (bin string). Ce prétraitement permet d'utiliser par la suite un codage arithmétique binaire adaptatif plutôt qu'un codage arithmétique m-aire adaptatif qui nécessite d'estimer beaucoup plus de probabilités, d'utiliser des multiplications et qui est beaucoup plus complexe. Cette binarisation peut donc être vue comme une conversion en code à longueur variable, mais ce code est ensuite encore codé par un codage arithmétique. Cette binarisation peut être réalisée à l'aide de trois méthodes :

- le code unaire (U) ou unaire tronqué (TU) : pour tout entier non signé $x \geq 0$, le code unaire correspond à x bits à "1" avec un "0" pour terminer. Le code unaire tronqué s'applique à tout $0 \leq x \leq S$, pour $x < S$ il correspond au code unaire et pour $x = S$ le code ne contient que x bits à "1" (pas de "0").
- le code Exp-Golomb d'ordre k (EGk) : c'est une version modifiée des codes Exp-Golomb. Ce code s'applique à un entier non signé x et se compose d'un préfixe et d'un suffixe. Le préfixe est un code unaire correspondant à la valeur $l(x) = \lfloor \log_2(x/2^k + 1) \rfloor$. Le suffixe correspond à une représentation binaire de $x+2^k (1-2^{l(x)})$ qui utilise $k + l(x)$ bits.
- le code à longueur fixe (FL) : pour $0 \leq x < S$, le code FL de x est donné par sa représentation binaire, mais limitée à $l_{FL} = \lceil \log_2 S \rceil$ bits.

Ces méthodes élémentaires sont ensuite concaténées pour donner les binarisations utilisées dans la norme. Ces méthodes dérivées sont au nombre de trois. La première est la concaténation d'un préfixe FL sur 4 bits et d'un suffixe TU avec S = 2. Ce code permet de représenter le CBP (coded_block_pattern qui indique quels blocs d'un macrobloc contiennent des coefficients non nuls), le préfixe traduit la partie du CBP correspondant à la luminance et le suffixe la partie du CBP correspondant aux chrominances. La seconde est la concaténation d'un préfixe TU et d'un suffixe EGk, elle est dénommée UEGk. Elle s'applique aux différences de vecteurs de mouvement mvd. Si mvd = 0, le code sera "0", autrement le préfixe est TU avec S = 9. Si $|mvd| \geq 9$, le suffixe est le code EG3 correspondant à la valeur $|mvd| - 9$ précédé du bit de signe ("1" pour - et "0" pour +). Sinon quand $0 < |mvd| < 9$ le suffixe n'est que le bit de signe. La troisième est aussi un code UEGk, mais cette fois-ci elle s'applique aux valeurs absolues des coefficients transformés (abs_level -1 car on ne transmet pas les 0). Le préfixe est alors un code TU avec S = 14 et le suffixe est un code EG0.

2.4. La modélisation du contexte

Une des propriétés les plus importantes du codage arithmétique est la possibilité d'utiliser une interface entre la modélisation et le codage. La modélisation permet d'assigner aux symboles un modèle de distribution de probabilité qui est ensuite utilisé par le codeur arithmétique pour produire une représentation de ces symboles en adéquation avec ce modèle de distribution. Autrement dit, il s'agit d'adapter la probabilité d'apparition d'un symbole binaire à son contexte comme : type de l'information codée, valeurs des symboles déjà traités dans un voisinage spatio-temporel... Il est donc important d'avoir des modèles statistiques bien adaptés aux données et de les garder à jour tout au long du codage.

La norme H.264/AVC (main profile) définit 459 modèles statistiques représentés par leur indice de contexte γ, modèles qui dépendent de la donnée à coder et du type d'image auquel elle appartient (I, P ou B). Nous spécifions une fonction permettant le calcul de l'indice de contexte à utiliser pour coder le symbole. Les éléments de syntaxe présentés dans le Tableau 3.2 sont divisés en deux parties : la première 0≤ γ≤ 84 correspond aux données relatives aux macroblocs et au mode de prédiction, la seconde 85≤ γ ≤459 correspond aux résidus des coefficients transformés. Il existe donc une fonction permettant de calculer pour chacune de ces parties. Pour la première partie, on obtient γ selon :

$$\gamma = \Gamma_s + \chi_s \tag{3.1}$$

où Γ_s est l'offset d'indice de contexte, il correspond à la borne inférieure de l'indice définie dans le tableau 3.2, et χ_s est l'incrément d'indice de contexte de l'élément de syntaxe S, il est fonction du symbole et de l'indice du bin à coder. Pour la seconde partie, on obtient en appliquant :

$$\gamma = \Gamma_s + \Delta_s(\text{ctx\_cat}) + \chi_s \tag{3.2}$$

où est utilisé en plus un offset Δ_s dépendant de la catégorie de contexte (ctx_cat). Ces catégories proviennent du type de transformée qui a été utilisée :

- Luma_Intra16_DC : catégorie 0 - Chroma_DC : catégorie 3
- Luma_Intra16_AC : catégorie 1 - Chroma_AC : catégorie 4
- Luma_4×4 : catégorie 2

Les offset Δ_s associés sont alors donnés par :

Tableau 3. 2. Valeurs des offsets Δ_s dépendant de la catégorie et de l'élément

Éléments syntaxiques	**Catégorie de contexte**					
	0	**1**	**2**	**3**	**4**	**5**
coded_block_flag	0	4	8	12	16	na
significant_coeff_flag	0	15	29	44	47	0
last_significant_coeff_flag	0	15	29	44	47	0
coeff_abs_level_minus1	0	10	20	30	39	0

C'est donc χ_s qui permet de s'adapter et de garder à jour le modèle de contexte. Le calcul de cet incrément est basé sur le voisinage du bloc à coder et les fonctions utilisées sont définies par la norme. Nous présentons un exemple pour le codage d'un bloc C avec A à gauche et B au-dessus pour le mvd (différence de vecteurs de mouvement). Pour son premier bin, nous avons :

$$\chi_{mvd}(C) = \begin{cases} 0 & si \quad e(A,B) < 3 \\ 1 & si\ 3 \le e(A,B) < 32 \\ 2 & si \quad e(A,B) > 32 \end{cases} \tag{3.3}$$

avec e(A,B)= |mvd(A)|+|mvd(B)|

Le mvd est binarisé à l'aide d'un code UEG3 avec S = 9. Les bin 2 à 9 du préfixe utilisent les 4 autres modèles de contexte définis dans le tableau 3.3 (bin 2 -> modèle 3, 3 -> 4, 4 -> 5, 5 -> 6, 6-9 -> 6). Et, les autres bin correspondant au signe et à l'EG3 sont codés en utilisant le codage "bypass".

Tableau 3. 3. Les éléments syntaxiques et leurs indices de contexte associés

	Les éléments syntaxiques	**Types de Slices**			
		SI	**I**	**P, SP**	**B**
slice_data( )	mb_skip_flag			11-13	24-26
	mb_field_decoding_flag	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	0-10	3-10	14-20	27-35
	transform_size_8x8_flag	na	399-401	399-401	399-401
	coded_block_pattern (luma)	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	77-84	77-84	77-84	77-84
	mb_qp_delta	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	68	68	68	68
	rem_intra4x4_pred_mode	69	69	69	69
	prev_intra8x8_pred_mode_flag	na	68	68	68
	rem_intra8x8_pred_mode	na	69	69	69
	intra_chroma_pred_mode	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0			54-59	54-59
	ref_idx_l1				54-59
	mvd_l0[ ][ ][0]			40-46	40-46
	mvd_l1[ ][ ][0]				40-46
	mvd_l0[ ][ ][1]			47-53	47-53
	mvd_l1[ ][ ][1]				47-53
sub_mb_pred( )	sub_mb_type			21-23	36-39
residual_block_cabac( )	coded_block_flag	85-104	85-104	85-104	85-104
	significant_coeff_flag [ ]	105-165 277-337	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
	last_significant_coeff_flag[ ]	166-226 338-398	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
	coeff_abs_level_minus1[ ]	227-275	227-275 426-435	227-275 426-435	227-275 426-435

2.5. Le codage arithmétique binaire

Nous avons vu que les modèles permettent de définir les probabilités d'apparition des bin ("0" ou "1"), ces probabilités sont : pLPS (Least Probable Symbol) et pMPS (Most Probable Symbol). Le codage arithmétique consiste à diviser récursivement un intervalle qui est défini par L sa borne inférieure et R sa longueur. La division de cet intervalle s'effectue de la façon suivante [Marpe 03]:

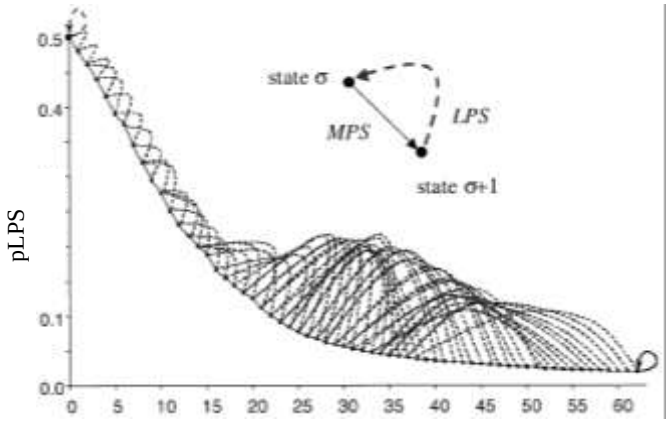
$$R_{LPS} = R.\ pLPS \text{ et } R_{MPS} = R - R_{LPS} \quad (3.4)$$

Pour s'affranchir de la multiplication, CABAC utilise une table construite à partir de Q_ρ et p_σ. Q_ρ est un ensemble de 4 approximations de R (0 à 3), et p_σ est un ensemble de 64 probabilités possibles pour pLPS ($0 \leq \sigma \leq 63$). Ces probabilités $p_\sigma \in [0.01875, 0.5]$ sont calculées à partir de l'équation récursive suivante :

$$p_\sigma = \alpha . p_{\sigma-1} \text{ avec } \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63} \text{ et } p_0 = 0.5 \quad (3.5)$$

Pour coder un bin, nous calculons tout d'abord l'approximation Q(R) de la longueur R de l'intervalle, cette approximation est définie par son indice à qui est calculé selon :

$$\rho = (R >> 6) \,\&3 \quad (3.6)$$



State Idx	LPS prob	MPS prob	State Idx	LPS prob	MPS prob
0	0.50	0.50	32	0.10	0.90
1	0.48	0.52	33	0.09	0.91
2	0.46	0.54	34	0.09	0.91
3	0.44	0.56	35	0.08	0.92
4	0.42	0.58	36	0.08	0.92
5	0.40	0.60	37	0.08	0.92
6	0.38	0.62	38	0.07	0.93
7	0.36	0.64	39	0.07	0.93
8	0.34	0.66	40	0.06	0.94
9	0.32	0.68	41	0.06	0.94
10	0.30	0.70	42	0.06	0.94
11	0.29	0.71	43	0.05	0.95
12	0.28	0.72	44	0.05	0.95
13	0.26	0.74	45	0.05	0.95
14	0.25	0.75	46	0.05	0.95
15	0.24	0.76	47	0.04	0.96
16	0.22	0.78	48	0.04	0.96
17	0.21	0.79	49	0.04	0.96
18	0.20	0.80	50	0.04	0.96
19	0.19	0.81	51	0.04	0.96
20	0.18	0.82	52	0.04	0.96
21	0.17	0.83	53	0.03	0.97
22	0.16	0.84	54	0.03	0.97
23	0.15	0.85	55	0.03	0.97
24	0.15	0.85	56	0.03	0.97
25	0.14	0.86	57	0.03	0.97
26	0.13	0.87	58	0.03	0.97
27	0.13	0.87	59	0.02	0.98
28	0.12	0.88	60	0.02	0.98
29	0.11	0.89	61	0.02	0.98
30	0.11	0.89	62	0.02	0.98
31	0.10	0.90	63	0.01	0.99

Figure 3. 3. Règles de transition pour la mise à jour des probabilités LPS (pointillés) et MPS

Nous avons maintenant σ et σ (issu du modèle et mis à jour par le bin précédent), la table nous donne la valeur du produit pσ. Qρ correspondant à R_{LPS}. Ceci permet de calculer le nouvel intervalle $R = R - R_{LPS}$. Si le bin est le MPS alors le sous-intervalle bas (le plus proche de 0) est sélectionné, la borne inférieure L de cet intervalle est inchangée et sa longueur R est celle qui a été calculée précédemment. Sinon (LPS) c'est le sous-intervalle haut qui est sélectionné, la borne inférieure est mise à jour selon L = L + R et sa longueur est affectée à $R = R_{LPS}$.

Dans les deux cas, le modèle de contexte est mis à jour en calculant la nouvelle valeur de σ.

Si le bin est le MPS, alors il faut augmenter la probabilité pMPS du modèle est réalisant :

$$\sigma = \min(\sigma + 1, 62) \quad (3.7)$$

Sinon c'est la probabilité pLPS du modèle qu'il faut augmenter, cela est donné par une table (TransIdxLPS[σ]) (tableau 3.4) définissant l'état σ atteint sachant qu'un LPS a été observé à l'état σ actuel.

```
if( binVal  = =  valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
else {
    if( pStateIdx  = =  0 )
        valMPS = 1 – valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
```

Tableau 3. 4. Table de transition d'état

pStateIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

Nous présentons dans la figure suivante les différentes étapes effectuées au cours du codage arithmétique (calcul Q(R), R_{LPS}, R) et nous mettons la table des rangeTabLPS en annexe (Tableau A.29).

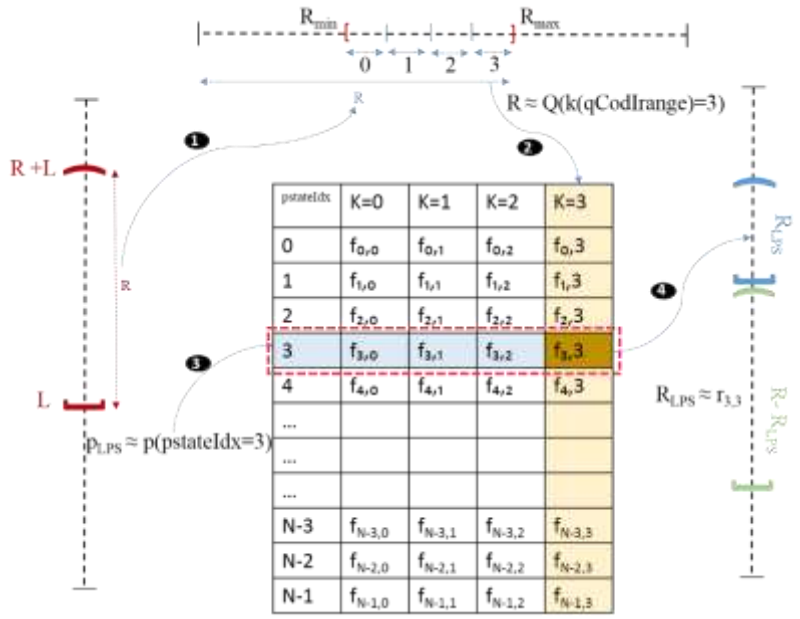


Figure 3. 4. Étapes du codage arithmétique

Le codage alternatif "bypass" est utilisé pour simplifier et accélérer le codage d'informations binaires caractérisées par des répartitions équiprobables comme les bits de signe, bits les moins significatifs... Dans ce cas, les modèles de contexte ne sont pas pris en compte et il est considéré que les probabilités de MPS et LPS sont équiprobables. L'intervalle sera donc divisé par 2 à chaque bin codé. Ceci permet également le codage direct des symboles binaires. Alors, le codeur arithmétique peut être équiprobable (tous les symboles ont la même probabilité

d'apparition) ou adaptatif, auquel cas les probabilités sont mises à jour en fonction des symboles codés. Cette adaptation permet d'affiner ces probabilités pour chacune des séquences traitées et d'améliorer les performances du codeur. Ainsi, la modélisation par contextes ne soit efficace que si les symboles présentant des répartitions de probabilités concentrées autour de certaines valeurs. Quant à l'initialisation, elle permet d'affecter au départ une valeur à chaque probabilité d'apparition des symboles et elle n'est pas toujours utilisée. Autrement, nous débutons le codage des informations d'orientation prédites avec des probabilités des symboles moyennes calculées sur un ensemble de séquences ce qui permet d'améliorer les performances du codeur arithmétique.

3. Analyse du processus CABAC

3.1. Format des données d'entrée et de sortie

La norme H264/AVC organise le flux binaire, généré par l'encodeur dans une séquence d'unités syntaxiques appelées NALU (Network Abstraction Layer Units) et illustré sur la figure (3.5). Ces unités sont reliées pour former un flux binaire continu. Chaque unité est constituée d'un entête et d'un champ d'information. Pour les NALUs comprenant la quantité d'information vidéo, le champ utile contient un ensemble de macroblocs. Á son tour, chaque macrobloc contient un entête et des données résiduelles. Les deux premières NALUs sont réservées respectivement au codage des SPS et PPS qui comprennent des informations à propos de la séquence. Les éléments syntaxiques sont encodés avec une longueur variable ou fixe.

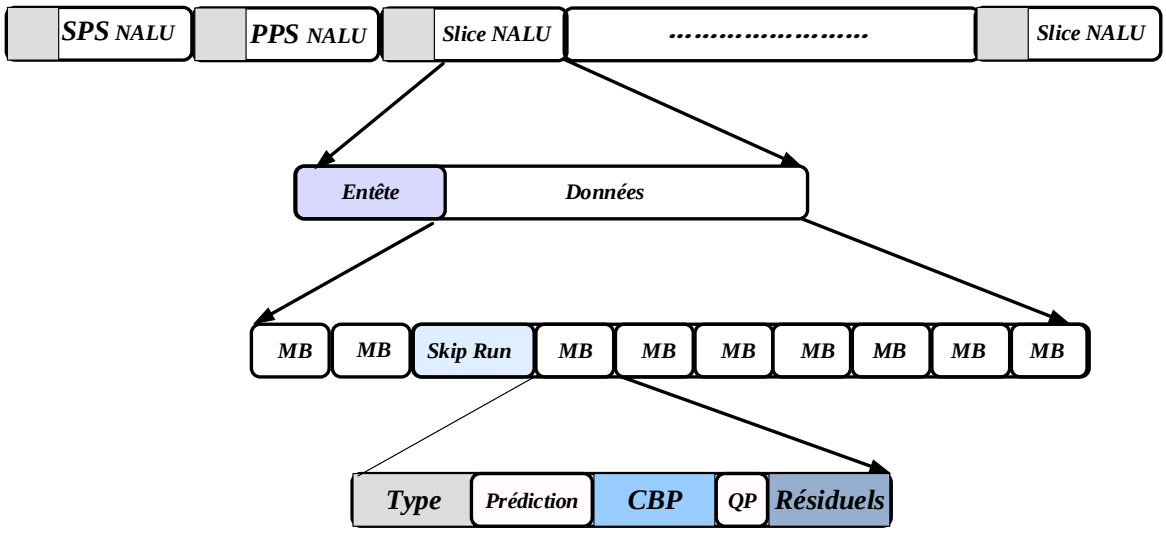


Figure 3. 5. Structure hiérarchique du flux binaire

Les données résiduelles des blocs sont codées en utilisant CAVLC et tous les éléments syntaxiques sont codés par l'Exp-Golomb. Les syntaxes pour coder un MB dans le profil de base de la norme H.264 sont indiquées dans le tableau (3.5).

Tableau 3. 5. Les éléments syntaxiques du macrobloc

Paramètres	**Description**
Type des MBs	Méthode de prédiction de chaque MB codé
Motif de bloc codé	Indiquer quel bloc dans le MB contenant les coefficients à coder
Paramètre de quantification	Transmis de la valeur du QP précédente
Données résiduelles	Coefficient de chaque bloc 4x4 ou 2x2

Le VLC est utilisé pour coder les coefficients résiduels transformés et quantifiés des blocs de luminance et de chrominance. Il traite chaque bloc 4x4 et bloc 2x2 en zig-zag et nous codons les coefficients de chaque bloc de la façon citée par la suite. Lorsque tous les blocs des images (Intra ou Inter) ont été transformés, quantifiés et codés, il faut les mettre en forme pour la transmission et/ou le stockage. La normalisation ne définit pas le codeur mais le train binaire, c'est-à-dire l'organisation des données et leurs caractéristiques. Les données codées héritent leurs caractéristiques des techniques utilisées pour mettre en œuvre les quatre modules du codage présentés précédemment, c'est-à-dire les méthodes d'estimation/compensation de mouvement (précision, taille de fenêtre. . .), de transformation, de quantification et de codage entropique. L'organisation des données correspond à la structure des données pour la transmission et/ou le stockage.

3.2. Processus de codage

3.2.1. Processus de codage de tranche et d'unité NAL

Le standard H.264 fournit plus de flexibilité aux applications à travers une large variété de réseaux. En effet, la particularité de la norme H.264 est la standardisation de la couche réseau. Deux couches constituent donc le codec, à savoir, la couche réseau NAL (Network Abstraction Layer) assurant la transmission et la couche vidéo VLC (Video Layer Coding) portant les informations des MBs de la séquence compressée. La couche NAL organise le flux VCL dans des unités appelées NALU (NAL Unit) afin de faciliter l'intégration et le transport de la vidéo codée sur une variété de réseaux. Chaque NALU est constituée d'un entête de 8 bits réservée à la couche réseau et un champ qui porte l'information utile de la couche vidéo. Trois types de NALU sont principalement utilisés :

- La NALU SPS (Sequence Parameter Set) : C'est la première NALU dans l'ordre du bitstream, elle contient les paramètres communs de la séquence dans un ordre fixé par le standard telle que : le profil, le level, les coefficients de filtrage, le format de la chrominance (4 :2 :0 ou 4 :2 :2. . .). . .
- La NALU PPS (Picture Parameter Set) : C'est la seconde NALU après celle du SPS, elle permet de déterminer la taille de l'image, le nombre d'image, nombre de tranche (Slice) par image, le mode de codage entropique utilisé : CAVLC ou CABAC et d'autres paramètres. La NALU Slice contient les informations vidéo qui constituent les pixels de la séquence vidéo. Elle commence par un entête de slice, suivie par les codes binaires des MBs. Chaque MB est précédé par un entête qui contient les informations propres à lui comme le QP, le mode de prédiction, le VM,... L'ordonnancement du codage de ces informations est présenté par la figure 3.6, notons que ça diffère selon le type d'image en cours de traitement et qu'il y a des éléments itératifs (boucle).

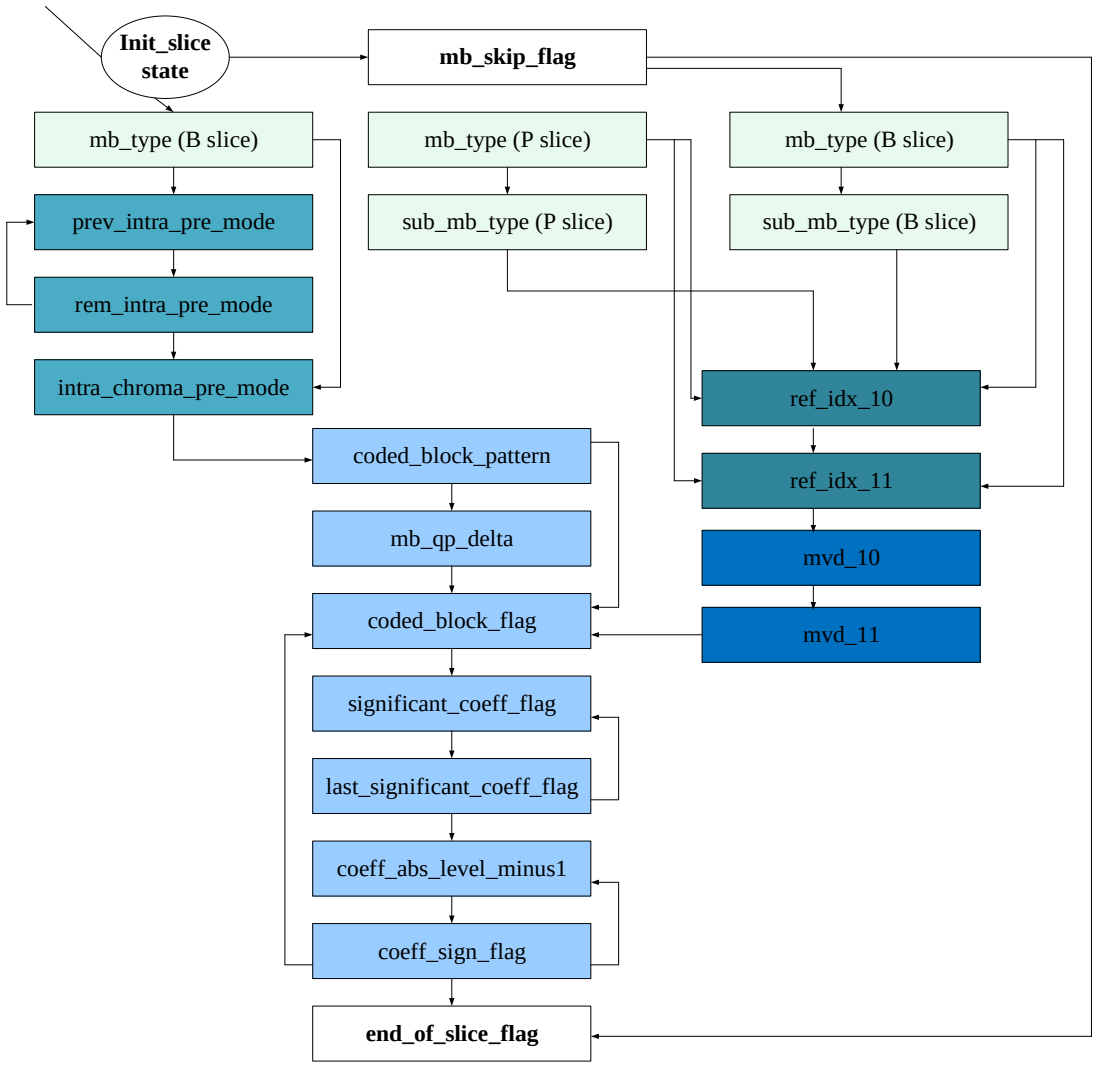
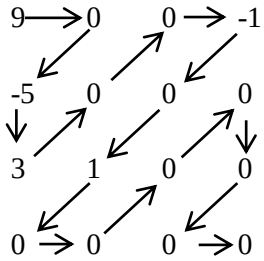


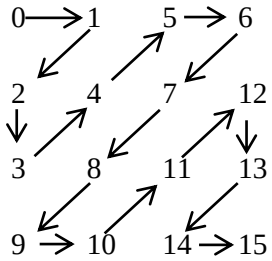
Figure 3. 6. Transmission d'état pour les éléments syntaxiques

3.2.2. Processus de codage des coefficients transformés

Nous distinguons divers éléments syntaxiques pour les coefficients résiduels : last_significant_coeff _prefix, last_significant_coeff_suffix, significant_coeff_group_flag, significant_coeff_flag, coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag, coeff_sign_flag, coeff_abs_level_remaining. Nous présentons un exemple du codage d'un bloc 4x4 pour la norme H.264 et HEVC afin de saisir les étapes effectuées et voir utilité des divers éléments syntaxiques cités.

Pour H.264, le parcours des données du bloc 4x4 se déroule comme ainsi :



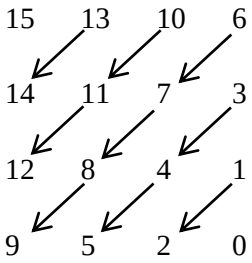


Alors, les paramètres de codage valent pour chaque coefficient :

Significant_coeff_flag	1	0	1	1	0	0	1	0	1
Last_significant_coeff_flag	0		0	0			0		1
Coeff_abs_level_minus1	8		4	2			0		0
Coeff_sign_flag	0		1	0			1		0

1	0	0	1	0	1	0	0	0	0	1	0	0	1	1	1	8	0	4	1	2	0	0	1	0	0
Mappage significatif																signe et valeur des coefficients									

Pour HEVC, le parcours des données du bloc 4x4 se déroule comme ainsi :



Alors, les paramètres de codage valent pour chaque coefficient (HM 8.0):

Last_significant_coeff_x	3									
Last_significant_coeff_y	0									
Significant_coeff_flag	1	0	1	0	0	0	1	0	1	1
Coeff_abs_level_greater1_flag	0		0				1		1	1
Coeff_abs_level_greater2_flag							1			
Coeff_abs_level_remaining							0		3	7
Coef_sign_flag	0		1				0		1	0

3	0	1	0	1	0	0	0	1	0	1	1	0	0	1	1	1	1	0	1	0	1	0	0	3	7

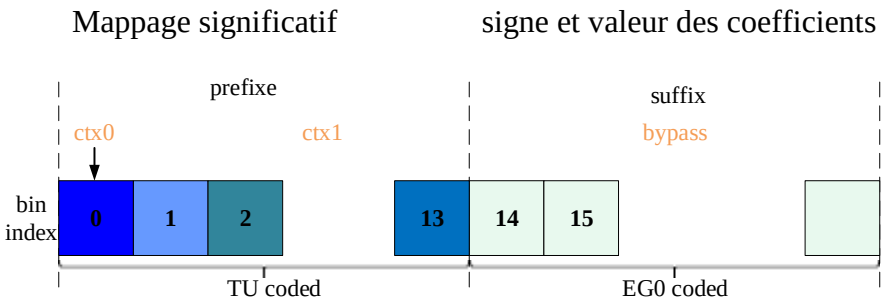


Figure 3. 7. Binarisation des valeurs absolues des coefficients résiduels pour H.264

Pour le H.264, chaque coefficient est composé de deux parties comme illustré dans la figure 3.7. Les 14 premiers bins, générés avec la binarisation unaire tronquée, sont codés par le BAC régulier. Les bins restants, générés par la binarisation Exp-Golomb d'ordre 0, sont codées par le BAC alternatif. Également, le signe du coefficient (un seul bin) est codé avec un BAC alternatif. Pour le HEVC, seuls les deux premiers bins du coefficient (coeff−abs−level−greater1−flag and coeff−abs−level−greater2−flag)) sont codés par le BAC régulier. La partie restante du coefficient (coeff−abs−level−remaining) est codé avec un BAC alternatif. "coeff−abs−level−remaining" est binarisé avec une binarisation unaire pour le préfixe, et une binarisation à longueur fixe pour le suffixe (32 bins maximum) [Budagavi 12].

3.2.3. Processus d'analyse syntaxique CABAC

Le codage est essentiellement symétrique au décodage. Nous pouvons distinguer trois processus dans l'analyse syntaxique CABAC : processus d'initialisation pour les variables de contexte et de codage arithmétique, processus de binarisation et processus de codage arithmétique binaire (figure 3.8). Après l'initialisation des paramètres, l'élément syntaxique subit une binarisation et puis, un contexte sera bien sélectionné pour chaque bin afin d'extraire la probabilité nécessaire au codage. L'état du moteur arithmétique de codage est représenté par une valeur de la variable codILow pointant sur l'extrémité inférieure d'un sous-intervalle et une valeur de la variable codIRange spécifiant la gamme correspondante de ce sous-intervalle.

3.2.3.1. Processus d'initialisation pour les variables de contexte et de codage arithmétique

Dans la procédure d'initialisation de l'encodeur, codILow est mis égal à 0, et codIRange est mis égal à 0x01FE. De plus, un firstBitFlag est mis égal à 1, et les compteurs bitsOutstanding et symCnt sont mis égaux à 0. La précision minimale de registre requise pour codILow est de 10 bit et pour CodIRange elle est de 9 bit. La précision requise pour les compteurs bitsOutstanding et symCnt devrait être suffisamment grande afin d'empêcher le débordement des registres en question.

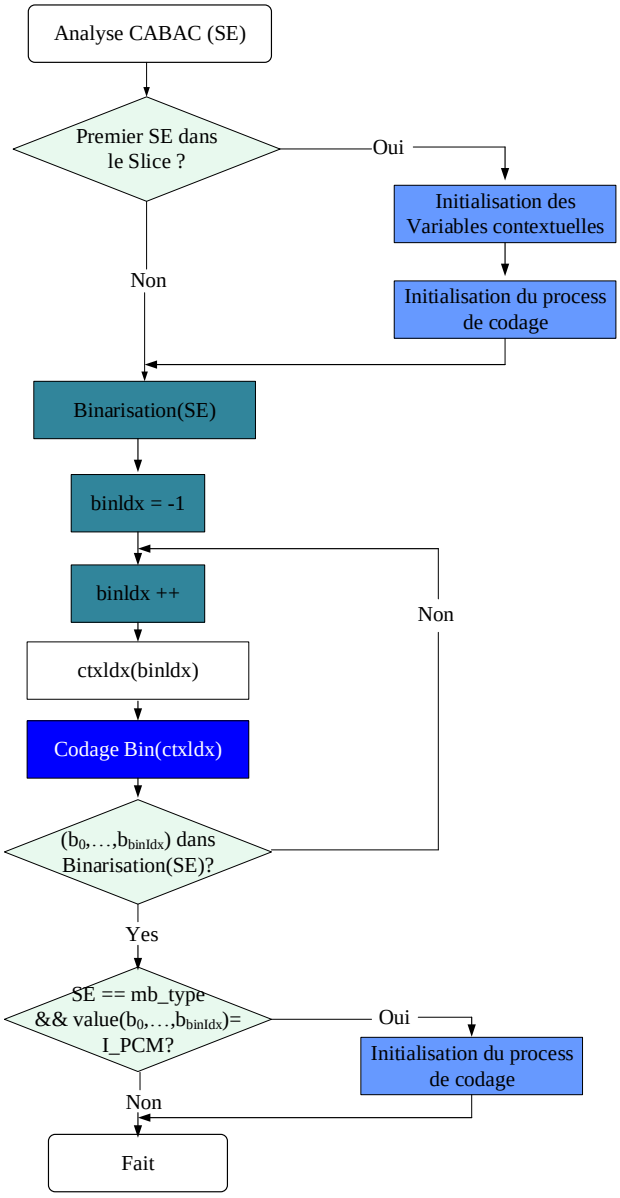


Figure 3. 8. Illustration du processus d'analyse syntaxique pour un élément syntaxique

Lorsque MaxBinCountInSlice note le nombre total maximal de décisions binaires à coder en une tranche, la précision minimale de registre requise pour les variables bitsOutstanding et symCnt est donnée par Ceil (Log2 (MaxBinCountInSlice + 1)). En outre, ce processus génère les variables de contexte CABAC initialisées indexées par ctxIdx. Du Tableau A.1 au Tableau A.13 figurent les valeurs des variables n et m utilisées dans l'initialisation des variables de contexte allouées à tous les éléments syntaxiques. Pour chaque variable de contexte, nous initialisons les deux variables pStateIdx et valMPS. La variable pStateIdx correspond à un indice de probabilité d'état et la variable valMPS correspond à la valeur du symbole le plus probable. Les deux valeurs allouées à pStateIdx et valMPS pour l'initialisation sont dérivées de SliceQPY.

```
preCtxState = Clip3( 1, 126, ( ( m *Clip3( 0, 51, SliceQPY ) ) >> 4 ) + n )
si ( preCtxState <= 63 ) {
                    pStateIdx = 63 - preCtxState
                    valMPS = 0
                    } sinon  {
                    pStateIdx = preCtxState - 64
                    valMPS = 1
                    }
```

Pour chacun des types de tranche, la liste des ctxIdx inclue les valeurs de m et n nécessaires pour l'initialisation. Pour les types de tranche P, SP et B, l'initialisation dépend aussi de la valeur de l'élément syntaxique cabac_init_idc. Les noms des éléments syntaxiques n'affectent pas le processus d'initialisation. ctxIdx égal à 276 est associé au «end_of_slice_flag» et au «mb_type», qui spécifie le type de macrobloc « I_PCM ». Dans ce cas, les valeurs initiales associées à ctxIdx égal à 276 sont spécifiées comme pStateIdx = 63 et valMPS = 0, où pStateIdx = 63 représente un état de probabilité de non-adaptation. Les résultats de ce processus sont les valeurs codILow, codIRange, firstBitFlag, bitsOutstanding, et symCnt du moteur de codage arithmétique.

3.2.3.2. Processus de binarisation

Le processus de la binarisation génère les bins de l'élément syntaxique. Une valeur spécifique de la variable de décalage d'indice de contexte (ctxIdxOffset) et une valeur spécifique de la variable maxBinIdxCtx sont associées à chaque binarisation. Lorsque deux valeurs pour chacune de ces variables sont spécifiées pour un élément syntaxique, la valeur dans la rangée supérieure se rapporte à la partie préfixe tandis que la valeur dans la rangée inférieure se rapporte à la partie suffixe de la binarisation de l'élément syntaxique correspondant.

Nous spécifions divers processus de binarisation : le processus de binarisation unaire (U), le processus de binarisation unaire tronqué (TU), le processus de binarisation Exp-Golomb de kième ordre unaire concaténé (UEGk), et le processus de binarisation de longueur fixe (FL).

Le choix du processus dépend forcément du type d'élément syntaxique. La binarisation pour l'élément syntaxique mb_type, consiste en une chaîne de segments donnés par un enchaînement de chaînes binaires de préfixes et de suffixes. La binarisation UEGk est utilisée pour mvd et coeff_abs_level_minus1, et la binarisation du coded_block_pattern consiste aussi en un enchaînement de chaînes binaires de préfixes et de suffixes. Pour ces processus de binarisation, la chaîne binaire de préfixes et celle de suffixes sont indexées séparément au moyen de la variable binIdx. Les deux ensembles de chaînes binaires de préfixes et de suffixes sont désignés respectivement comme la partie préfixe de la binarisation et la partie suffixe de la binarisation.

Le Tableau A.15 illustre les types de binarisation des éléments syntaxiques, la variable de décalage d'indice de contexte (ctxIdxOffset) et maxBinIdxCtx associées à chaque binarisation.

3.2.3.3. Processus de codage pour une décision binaire

Ce processus spécifie comment chaque décision d'une chaîne binaire est analysée syntaxiquement pour chaque élément syntaxique. Lors de l'analyse syntaxique de chaque segment, la variable binIdx est incrémentée de 1 à partir de binIdx mise égale à 0 pour le premier segment. Lorsque la binarisation de l'élément syntaxique correspondant consiste en une partie de binarisation de préfixe et de suffixe, la variable binIdx est mise égale à 0 pour le premier segment de chaque partie de la chaîne de segments (partie préfixe ou partie suffixe). Si bypassFlag est égal à 1, le processus de codage est appliqué afin d'analyser syntaxiquement la valeur des segments provenant du binarisateur. Sinon (si bypassFlag est égal à 0), l'analyse syntaxique de chaque segment est établie.

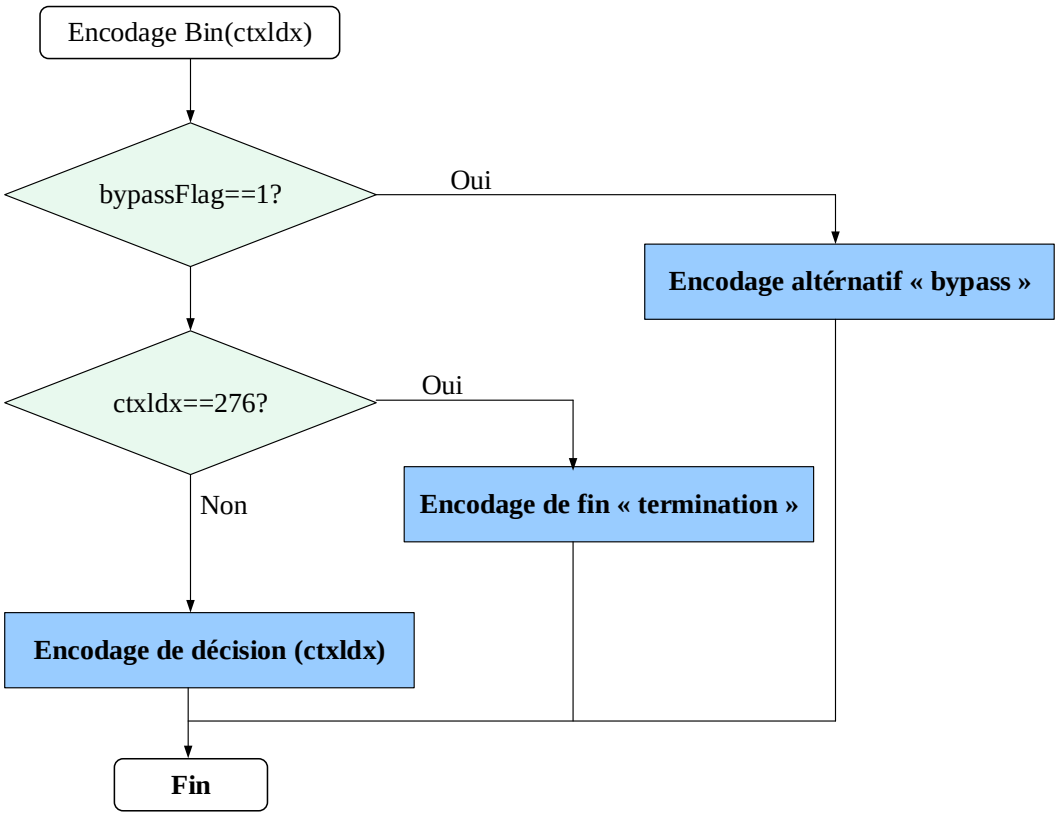


Figure 3. 9. Processus de codage arithmétique pour un segment

Etant donné binIdx, maxBinIdxCtx et ctxIdxOffset, ctxIdx est calculé et étant donné ctxIdx, nous codons la valeur du segment provenant du binarisateur. Les allocations des incréments de ctxIdx (ctxIdxInc) à binIdx pour toutes les valeurs de ctxIdxOffset est données dans les tableaux A.23 -25. Le ctxIdx est la somme de ctxIdxOffset et de ctxIdxInc ou la somme de

ctxIdxOffset, ctxIdxBlockCatOffset (ctxBlockCat) et ctxIdxInc (ctxBlockCat) (Tableau A.26). La Figure 3.9 illustre l'ensemble du processus de codage arithmétique pour un seul segment. Afin de coder la valeur d'un segment, l'indice de contexte ctxIdx est passé au processus de codage arithmétique Encodage Bin (ctxIdx).

Si bypassFlag est égal à 1, Encodage altérnatif est invoqué. Sinon, si bypassFlag est égal à 0 et que ctxIdx est égal à 276, Encodage de fin est invoqué. Sinon (si bypassFlag est égal à 0 et si ctxIdx n'est pas égal à 276), Encodage de décision est appliqué.

Le diagramme d'Encodage de décision est donné à la figure 3.10.

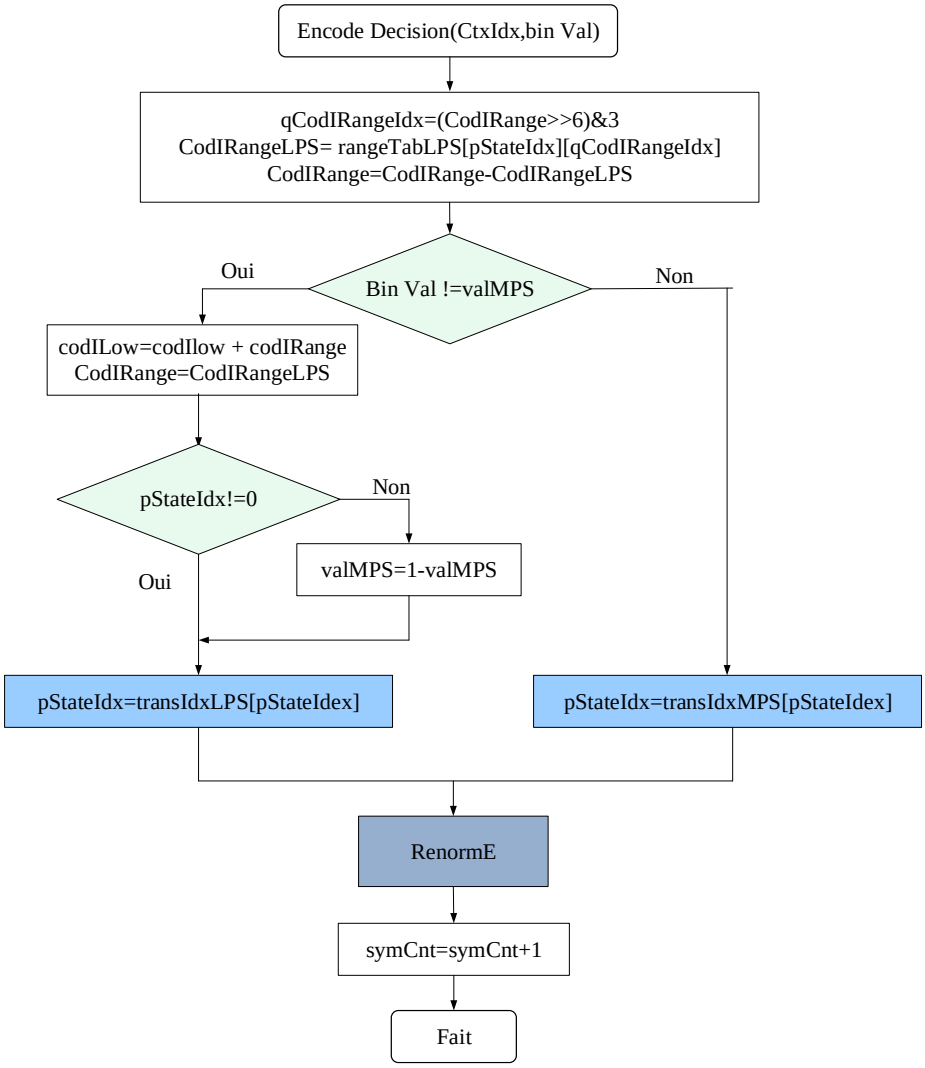


Figure 3. 10. Diagramme de codage d'une décision

Comme nous avons indiqué, le codage arithmétique se fonde sur le principe d'une subdivision à intervalle récursif. Un sous-intervalle de code donné initialement avec la gamme codIRange sera subdivisé en deux sous-intervalles ayant respectivement la gamme pLPS * codIRange et codIRange - pLPS * codIRange dont pLPS et 1 − pLPS les probabilités des LPS et MPS.

En fonction de la décision, le sous-intervalle correspondant sera choisi comme le nouvel intervalle de code, et une chaîne de code binaire pointant dans cet intervalle représentera la séquence de décisions binaires observées. Chaque contexte est spécifié par la probabilité pLPS du LPS et la valeur de MPS (valMPS), qui est 0 ou 1.

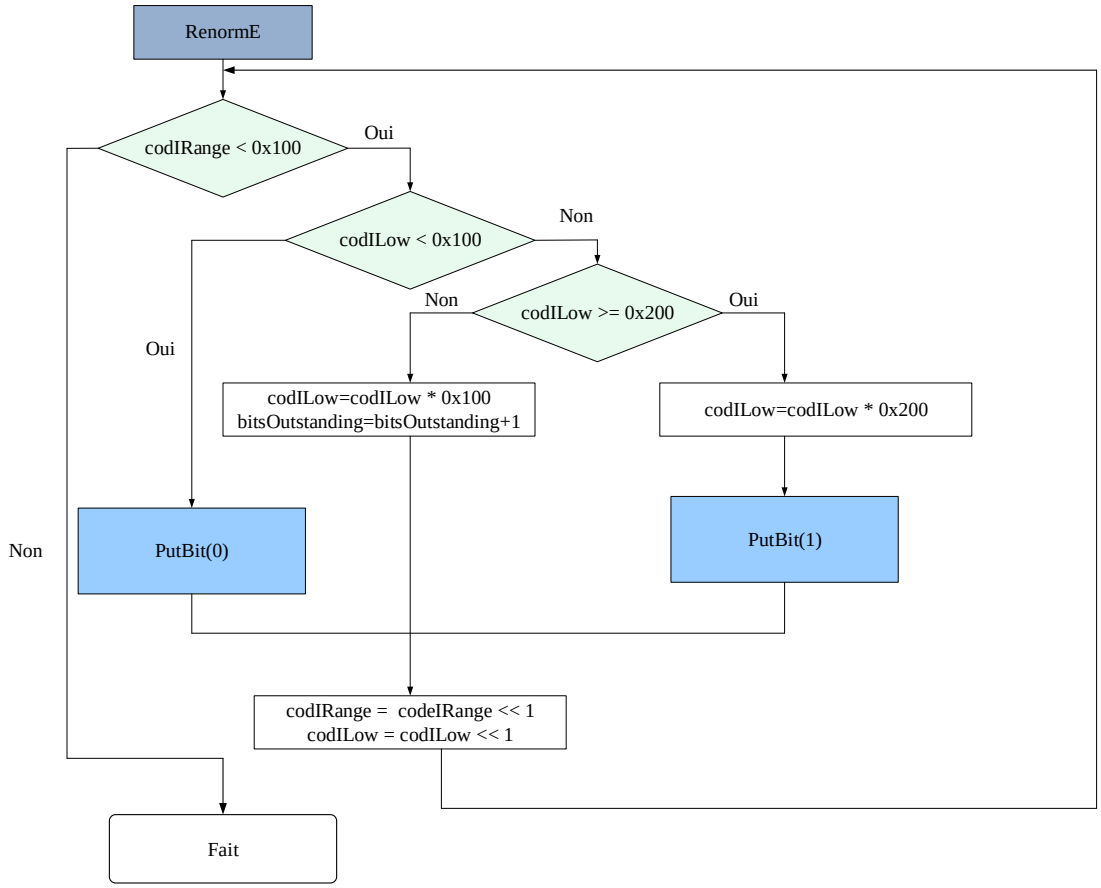


Figure 3. 11. Diagramme de renormalisation dans le codeur

Le diagramme de la renormalisation est donné à la figure 3.11. Les entrées sont les variables codIRange, codILow, firstBitFlag, et bitsOutstanding. Les résultats de ce processus sont zéro ou plusieurs bits écrits dans la charge utile RBSP et les variables mises à jour codIRange, codILow, firstBitFlag, et bitsOutstanding. La valeur de codIRange est d'abord comparée à 0x0100. Si codIRange est supérieur ou égal à 0x0100, aucune renormalisation n'est nécessaire et le processus RenormE est terminé. Sinon (si codIRange est inférieur à 0x0100), nous entrons dans la boucle de renormalisation. Dans cette boucle, nous discutons la valeur de codILow s'il est supérieur ou égal à 0x0100. Si oui, nous générons un bit « 0 », sinon nous vérifions s'il est supérieur ou égale à 0x0200. Si oui, nous multiplions codILow par 0x0200 et nous générons un bit « 1 », sinon nous le multiplions par 0x0100 et un compteur « bitsOutstanding » sera incrémenté. Après la génération du bit (sois « 0 » soit « 1 »), la valeur de codIRange et codILow seront doublées.

La procédure PutBit ( ) décrite par la figure 3.12 génère les bits de sortie. Elle utilise la fonction WriteBits (B, N) qui écrit N bits avec la valeur B au flux binaire et avance le pointeur du flux binaire de N positions de bit. Cette fonction suppose l'existence d'un pointeur de flux binaire avec une indication de la position du prochain bit que le processus de codage doit écrire au flux binaire.

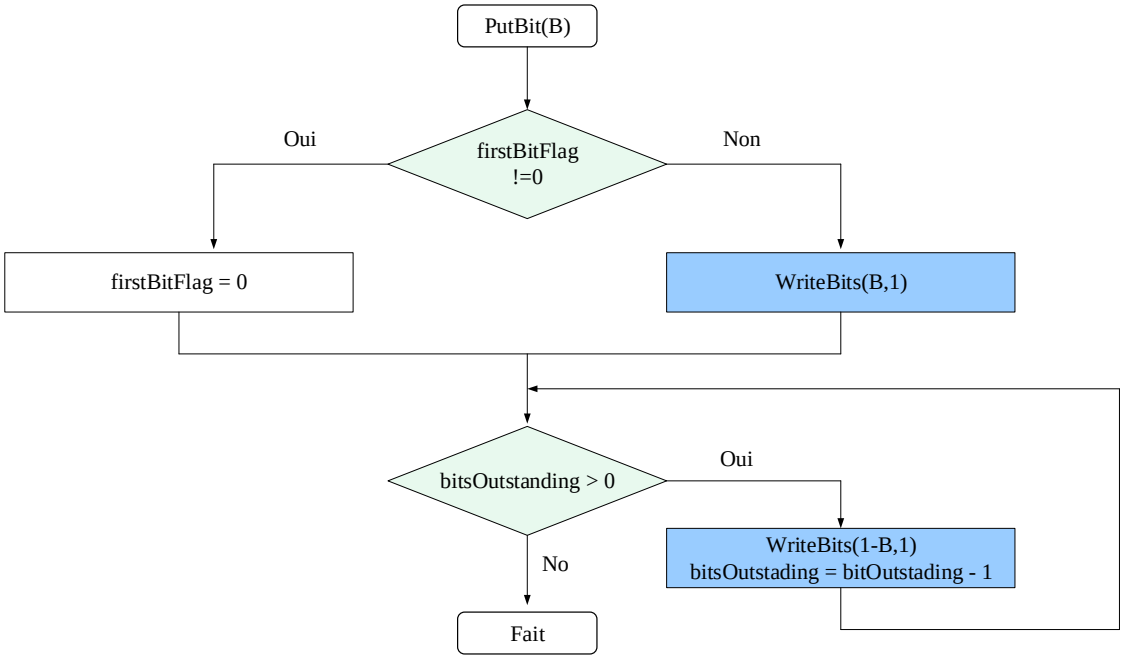


Figure 3. 12. Diagramme de PutBit(B)

Le processus de codage d'aiguillage pour décisions binaires s'applique à toutes les décisions binaires avec bypassFlag égal à 1. La renormalisation est incluse dans la spécification de ce processus comme indiqué à la figure 3.13. D'abord, la valeur de codILow est doublée. Puis, nous discutons la valeur de Bin val. S'il est « 0 », nous terminons le processus sinon (s'il est « 1 »), nous additionnons codIRange à l'ancien codILow pour trouver un nouveau codILow qui sera comparé à 0x0400. Si codILow est supérieur à 0x0400, nous générons un bit « 1 » et nous soustrayons 0x0400 à la suite. Sinon, si codILow est inférieur à 0x0200, nous générons un bit « 0 », sinon nous soustrayons 0x0200 du codILow et un compteur « bitsOutstanding » sera incrémenté. A la fin, un autre compteur « SymCnt » sera incrémenté.

Lorsque la valeur de binVal à coder est égale à 1 (figure 3.14), le codage CABAC est terminé et nous appliquons la procédure de vidage (figure 3.15). Dans cette procédure de vidage, le dernier bit écrit par WriteBits (B, N) est égal à 1. Lors du codage d'end_of_slice_flag, ce dernier bit est interprété comme le rbsp_stop_one_bit.

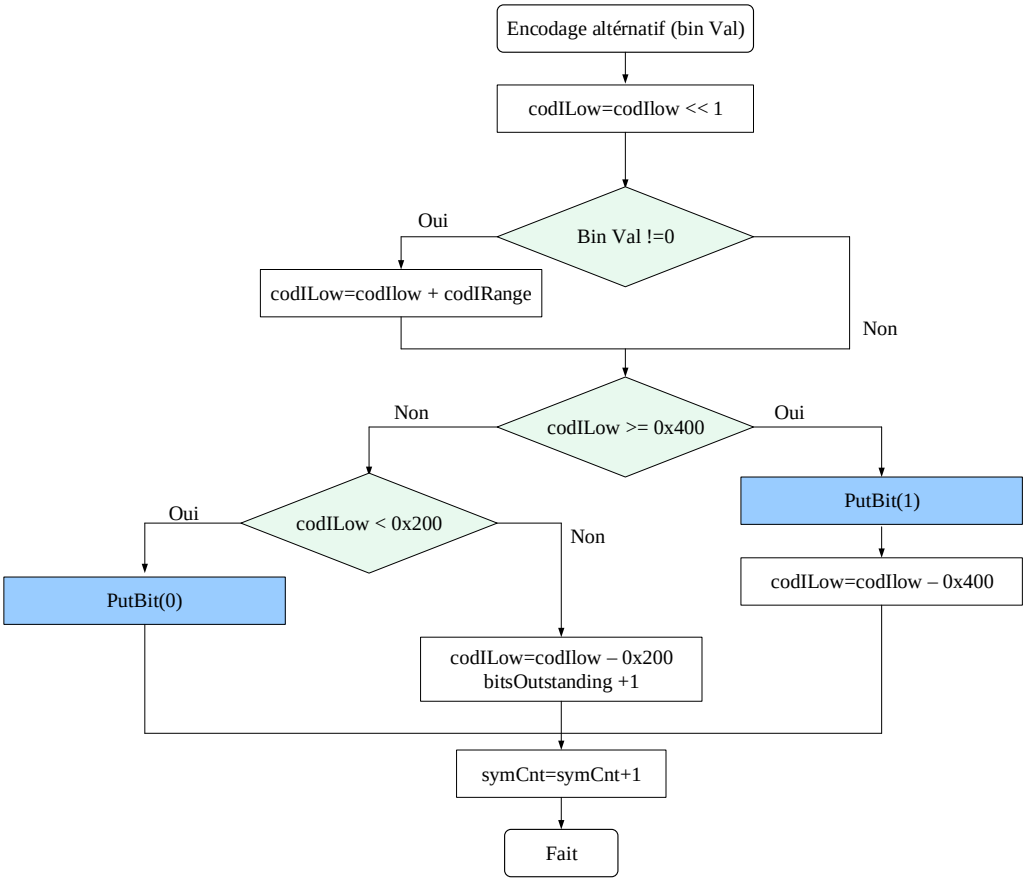


Figure 3. 13. Diagramme de l'aiguillage de codage

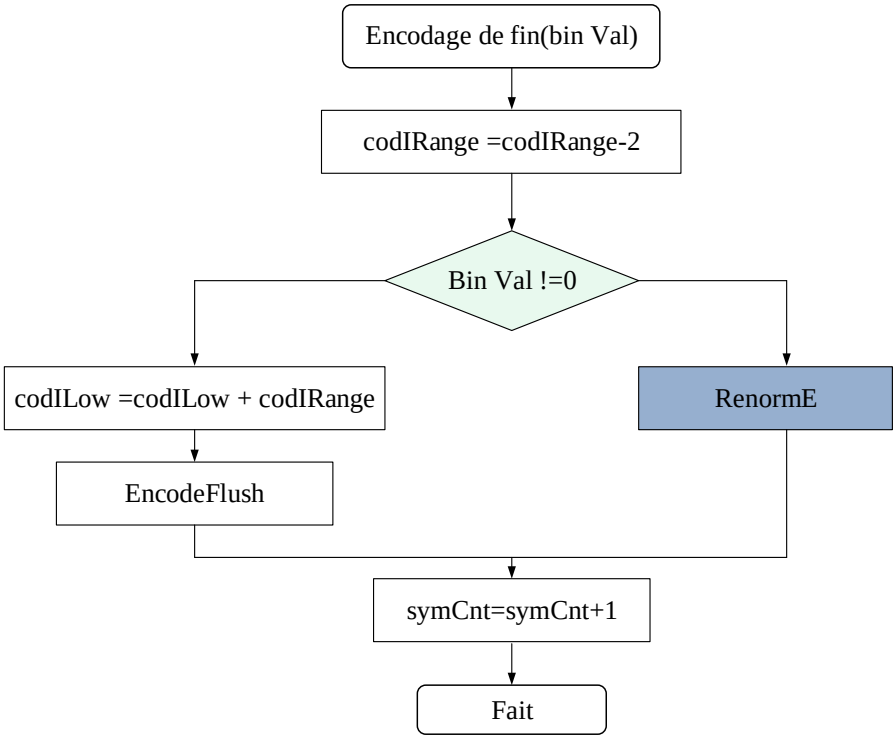


Figure 3. 14. Processus de codage pour une décision binaire avant achèvement

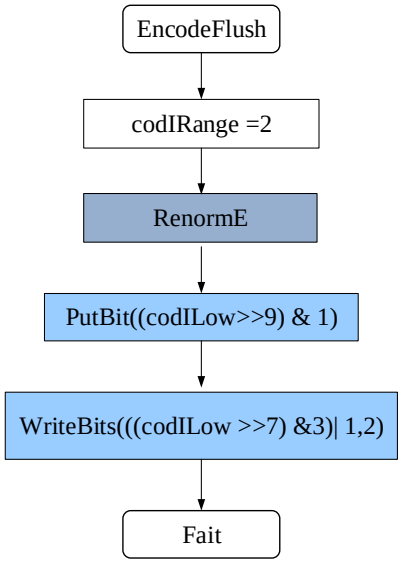


Figure 3. 15. Procédure de vidage à achèvement

4. Implantation du CODEC entropique sur une plateforme FPGA

Les FPGA (Field Programmable Gates Array) sont des composants logiques de haute densité et reconfigurables qui permettent, après programmation, d'implémenter en matériel des circuits décrits en langage HDL. Son architecture correspond à une matrice de portes logiques séparées par des réseaux d'interconnexion comme le montre la figure 3.16.

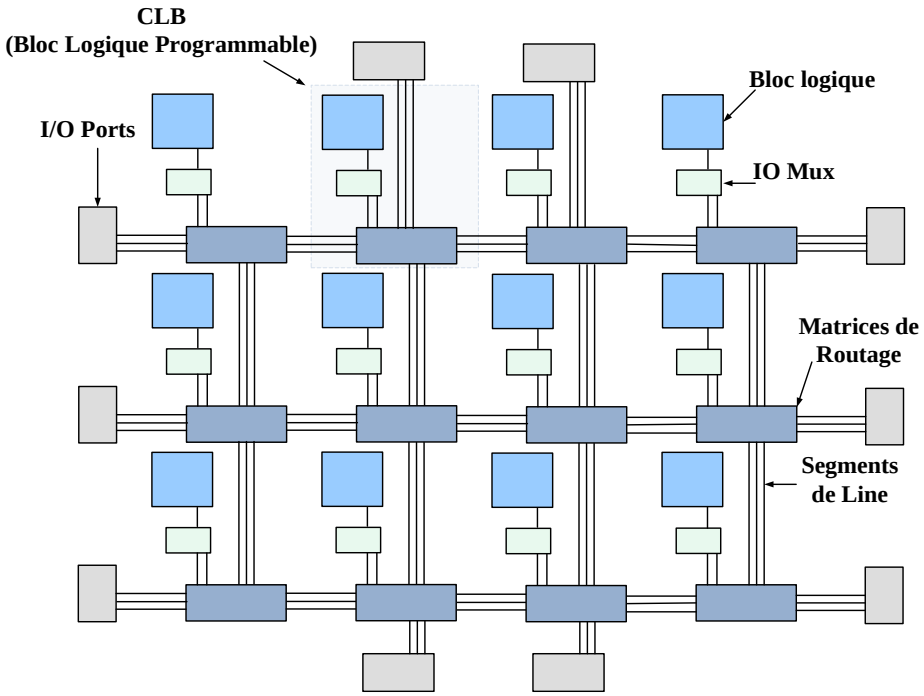


Figure 3. 16. Architecture de FPGA (XILINX)

Les I/O Ports (Input Output Ports) sont des cellules d'entrées-sorties qui permettent d'interfacer le FPGA avec l'environnement extérieur.

Afin de réaliser des fonctions complexes à partir des fonctions de base que représentent les CLBs (Configurable Logic Block), il est nécessaire de disposer de ressources d'interconnexion entre ces différentes cellules. Ce qui permet d'obtenir un circuit logique fonctionnant jusqu'à une fréquence de plusieurs centaines de MHz.

Actuellement, nous distinguons deux principaux producteurs de circuits logiques programmables : Xilinx et Altera. L'architecture du CLB varie fortement d'un producteur à l'autre. Chaque bloc CLB [35] est composé d'un nombre de sous blocs, définis en fonction de type du FPGA, nommé Slices. Ces derniers contiennent un ensemble de tables de vérité LUTs (Look-Up Table), qui représentent le cœur du FPGA. Ces cellules sont constituées d'éléments logiques programmables où l'on trouve des bascules, des LUTs, des multiplexeurs et des portes logiques.

Grâce à l'évolution de la technologie, l'architecture interne d'un FPGA est bien évoluée en complexité. La structure des CLB et la structure des Slices sont changés en augmentant le nombre de LUT par Slice et le nombre des entrées et des sorties par LUT. Il est montré que le nombre de Slices augmente d'une génération à l'autre à l'exception de la famille Virtex-5 [thèse yousri]. Nous trouvons par exemple 1 956 000 CLBs pour les Virtex 7 de Xilinx [Xilinx]. Ainsi, la fréquence d'horloge des FPGA (centaines de MHz) permet un temps de traitement d'une application bien réduit. Depuis quelques générations de FPGAs, de nombreux éléments sont ajoutés aux ressources classiques sur certains modèles comme les blocs mémoire, les multiplieurs, les DSP, les blocs processeur...

Les FPGAs classiques peuvent se comporter comme un espace de stockage de variables limité en terme d'espace disponible. Pour un stockage plus important, certains fabricants de FPGAs ont introduit des blocs mémoires à l'intérieur des FPGAs, de 10 à 90 Mbits. Sinon, nous pouvons accéder à des mémoires externes comme les RAMs (Random Access Memory), ROMs (Read Only Memory), ... et la taille des blocs mémoires varient suivant les constructeurs et le type de FPGA. Les ressources logiques des FPGAs permettent de réaliser aussi toutes sortes d'opérations arithmétiques. Ainsi, les FPGAs disposent des multiplieurs câblés qui limitent en particulier les multiplications coûteuses. De plus, le temps de calcul pour ces opérations est optimisé. En particulier, nous citons les DSP Blocs composés des multiplieurs embarqués de taille variable, additionneur/soustracteur et une unité accumulateur. Ces IPs intégrés au sein de FPGA servent dans plusieurs algorithmes de traitement de signal.

Avec l'apparition des très grands FPGAs, il est maintenant possible d'intégrer des algorithmes complexes sur une seule puce. Nous pouvons alors implanter divers blocs IP en logiciel ou/et

en matériel. En effet, certains FPGAs disposent de cœurs de processeurs (coeur Power PC pour Xilinx ou cœur ARM pour Altera). Aujourd'hui, les FPGA ont atteint des capacités de plusieurs millions de portes logiques équivalentes. Elles sont devenues des éléments incontournables des flots de conception de circuits numériques complexes. Leur apport au niveau de la phase de prototypage des systèmes en cours de conception est indéniable. Elles présentent une alternative d'utilisation intéressant qui dispose un compromis entre les flexibilités du processeur et les performances de la logique câblée.

Dans notre travail nous utilisons la plateforme XC6S avec un FPGA de type LX45. En 2009, Xilinx a lancé les FPGA-Spartan 6, en technologie 45 nm, avec 2 catégories différentes (LX et LXT), et qui offre un équilibre optimal entre faibles risques, faibles coûts, basse consommation et performances.

Notre carte offre également une technologie de gestion de l'alimentation avancée, jusqu'à 43 661 cellules logiques, 200 entrées-sorties, un support mémoire avancé et des tranches DSP de 390 MHz fonctionnant sous une tension de 1,26V ce qui fait de la carte FPGA Atlys Spartan 6 un support idéal pour les systèmes numériques complets conçus sur des processeurs embarqués, tels que le MicroBlaze de Xilinx.

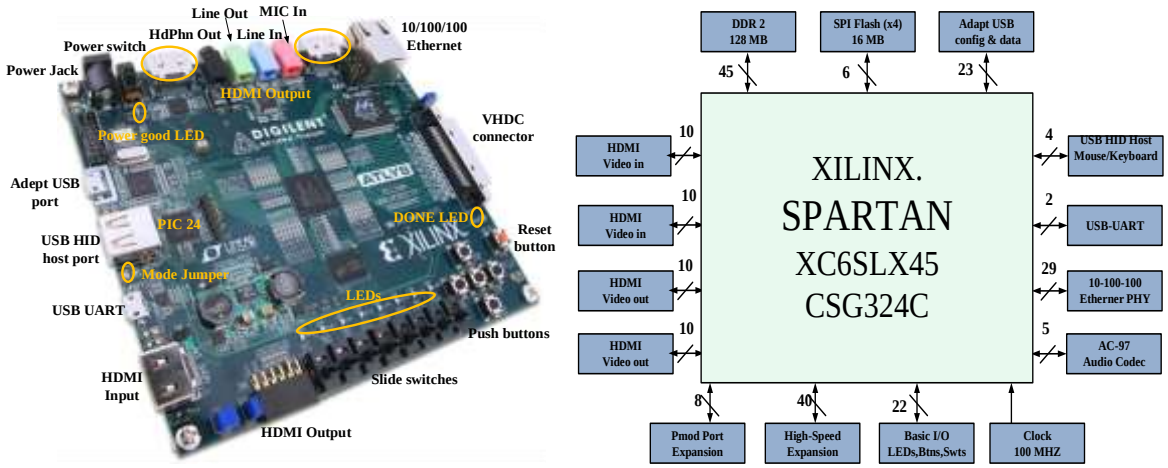


Figure 3. 17. FPGA spartan 6

4.1. Conception des systèmes numériques

4.1.1. Les conceptions conjointes (Codesign)

Pour les systèmes temps réel, la tendance actuelle est de tenir compte dès la spécification des interactions entre les deux parties du hardware et du software. La technique de conception de systèmes impose une séparation du hardware et software le plus tard possible dans le cycle de conception. Ce qu'on appelle le "hardware/software codesign".

Autrement dit, nous sommes face à un système sur puce intégrant des composants logiciels et matériels. Alors, après la spécification, qui décrit le comportement fonctionnel du système, une phase de partitionnement est effectuée. Nous distinguons trois parties :

- Une partie logicielle (programme exécutable sur un processeur) et le processeur peut être un GPU ou un processeur reconfigurable
- Une partie matérielle (circuits hardware : IP ou accélérateur matériel)
- Une interface de communication entre les deux parties.

Beaucoup de détails architecturaux de communication et d'ordonnancement des opérations sont intégrés. Ainsi, une description au niveau RTL est générée pour la partie matérielle et un code source pour la partie logicielle du système. Ce partionnement doit être vérifié et validé avant de passer à la synthèse et de mise en œuvre. Nous pouvons effectuer des analyses de performances de l'architecture à travers des simulations.

Suite à la synthèse logique de la partie RTL, les fonctions logiques seront placées et routées sur le plan de la puce (figure 3.18). Ce processus est réalisé à l'aide des outils de synthèses comme Synplify [Synplicity 07], XST [Xilinx], etc. La partie logicielle sera compilée pour générer une image hexadécimale.

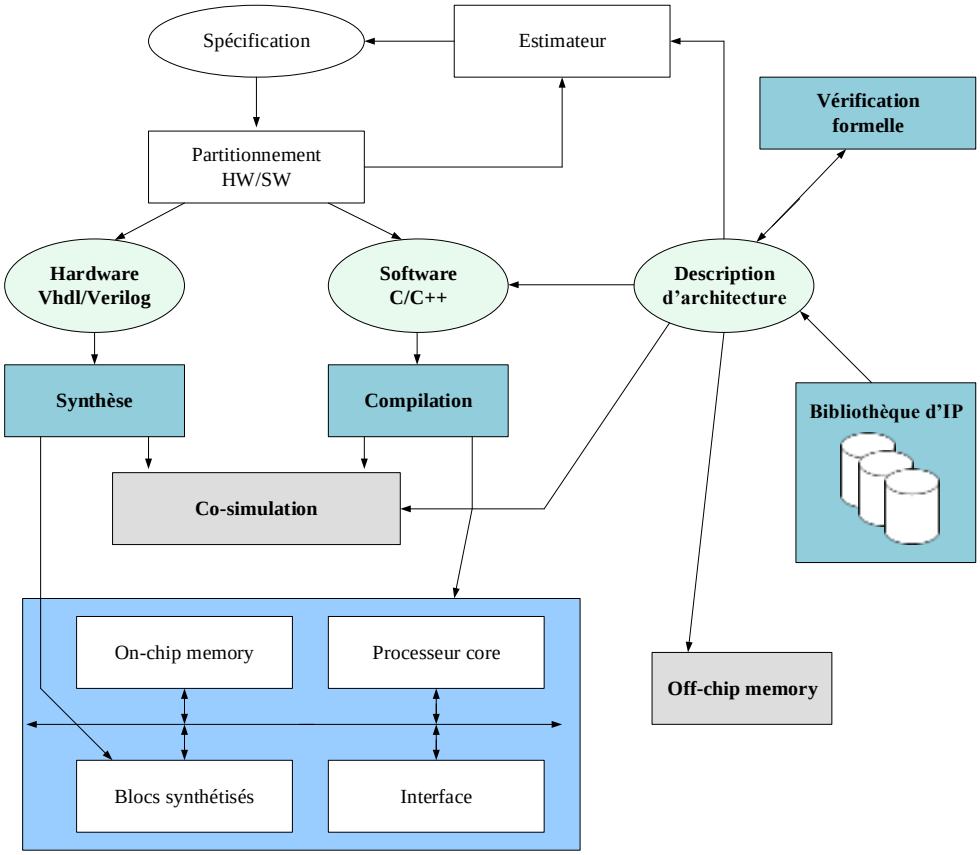


Figure 3. 18. Flot de conception standard [L'EXCELLENT 06]

Afin d'obtenir les performances en surface et en consommation d'énergie, nous effectuons des simulations après la synthèse logique et nous passons aux tests réels sur les cartes FPGA. Ce flot de conception conjointe logicielle/matérielle est adopté par plusieurs outils et compagnies ont adopté cette approche de codesign.citons EDK pour Embedded Development Kit proposé par Xilinx. Actuellement, la tendance est donc l'assemblage sur la même puce plusieurs composants pour répondre au mieux aux exigences des systèmes embarqués. Nous trouvons des processeurs, des DSPs et des accélérateurs matériels qui consistent à confier une fonction bien précise effectuée par le processeur à un circuit destiné à effectuer cette fonction de façon plus efficace (unité de calcul intensif). Les accélérateurs matériels intègrent des fonctions de calculs, des contrôleurs d'entrée/sortie ou des réseaux de communication et des mémoires considérées comme des IPs. Plusieurs accélérateurs matériels sont proposés comme des processeurs reconfigurables ou des accélérateurs pour un traitement complet (comme la compression d'image JPEG).

Distinguons aussi les composants virtuels (IPs) qui présentent des circuits génériques avec des outils associés accessibles. Ça peut apparaître sous forme de HDL synthétisable (IP Logiciel) ou une librairie technologique spécifique et un layout personnalisé (IP Matériel). L'intégration de coeurs de processeurs, tel que NIOS, MicroBlaze, … est possible et nous pouvons décrire des applications bien complexes par l'assemblage de quelques IPs.

4.1.2. Les processeurs embarqués

Actuellement, la majorité des processeurs sont destinés aux puces spécialisées (les téléphones mobiles, les tablettes, etc). Aussitôt, le cœur de processeur idéal doit fournir le maximum de performances sur une surface bien réduite. Cette partie du traitement constitue donc un élément critique dans un système. Elle peut être un processeur Hardcore comme le PowerPC, ARM, … ou bien un processeur softcore comme le MicroBlaze, NIOS-II, … ou même un processeur softcore Open source comme Leon , ORI200, … Pendant la conception, nous ne pouvons pas choisir le nombre de processeurs hardcore présents dans le FPGA mais nous mettons autant de processeurs softcore que la puce le permet. En effet, le processeur softcore est réalisé à partir de la logique programmable présente sur la puce et il n'occupe pas beaucoup en la surface de le FPGA. Alors, le FPGA nous garantit un prototypage rapide avec une spécialisation de ces processeurs bien approximés pour concevoir un système sur puce.

4.2. Accélérations proposées du module de codage entropique pour H.264/AVC et/ou HEVC

Dans cette partie, nous nous intéressons au module de codage entropique des derniers standards de normes vidéo H264 et HEVC. En effet, nous proposons une accélération matérielle du CABAC dans le sens directe et inverse du codage. Distinguons alors l'IP de l'encodeur et l'IP du décodeur.

4.2.1. Les IPs proposées pour l'encodeur CABAC

L'architecture de l'encodeur CABAC est composée de plusieurs étages comme montré dans la figure 3.19. Nous classons un bloc de BAC (régulier et altérnatif), un bloc de RAM pour sauvegarder les modèles contextuels et les élements syntaxiques, un bloc des tables (rangTabLPS, initialisation et transmission d'état) et un bloc de contrôle. Nous détaillons à la suite les diverses architectures des sous-blocs de l'encodeur CABAC et nous précisons les résultats de synthèse et les rapports de puissance générés.

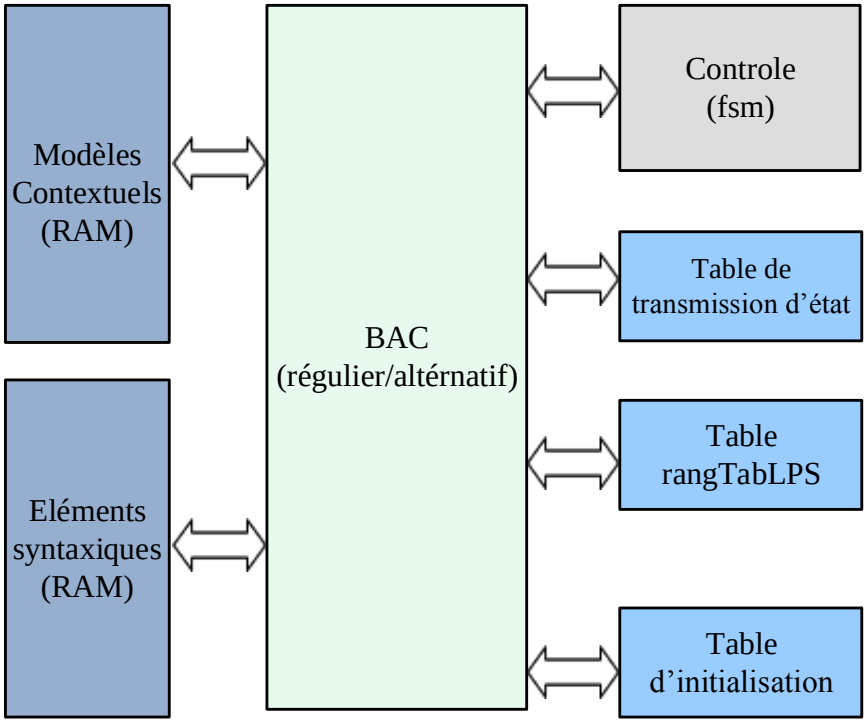


Figure 3. 19. Architecture de l'encodeur CABAC

4.2.1.1. Architecture matérielle d'implantation

Nous représentons dans cette section les architectures conçus des différents sous-blocs de l'encodeur CABAC illustré dans la figure 3.20. Nous distinguons 3 phases principales : la gestion de contexte, le codage arithmétique et la génération de sortie.

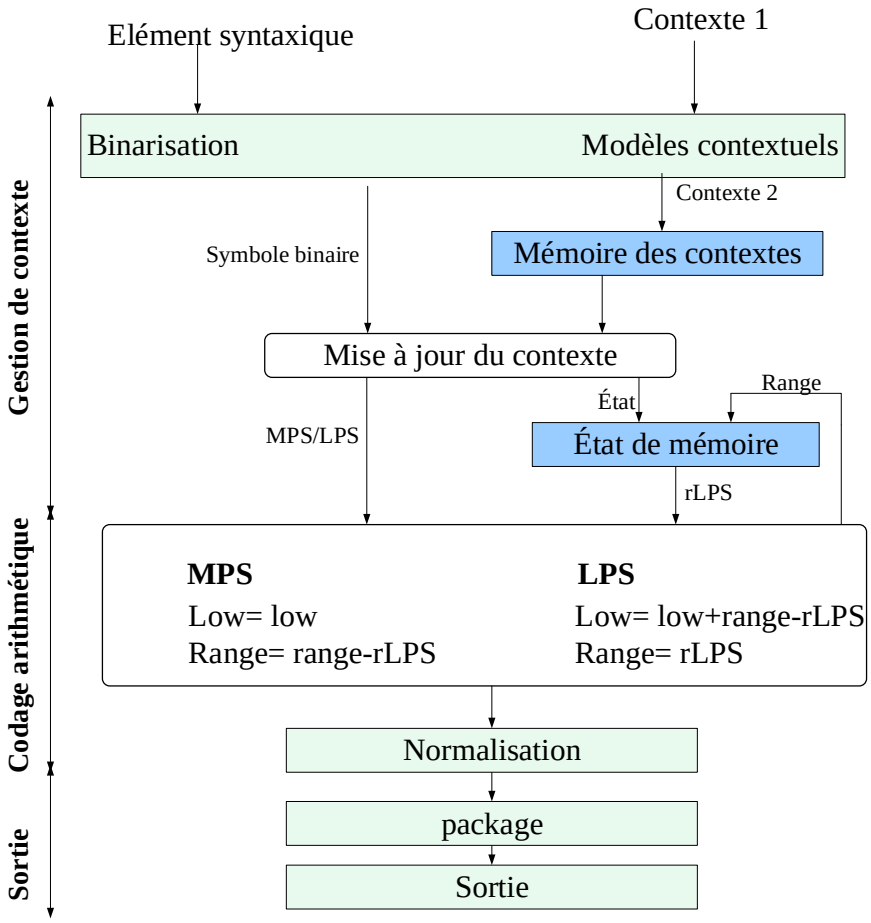


Figure 3. 20. Schéma bloc du CABAC

En premier temps, l'élément syntaxique subit la binarisation et le choix du contexte correspondant sera effectué. A la suite, la chaîne des bins de l'élément syntaxique sera envoyé bin par bin au codage arithmétique où s'effectue la mise à jour des intervalles codIRange et codILow. Ce calcul est assuré par la génération du rLPS de l'état selon le contexte courant. Un bloc de normalisation est indispensable après comme expliqué dans le processus de codage (section 3.2). Les bits de sortie (bitstream) seront à la fin empaquetés et envoyés par 32 bits. Les différentes étapes de codage de chaque bin sont récapitulées dans la figure 3.21.

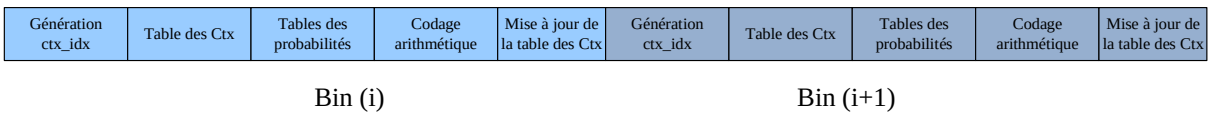


Figure 3. 21. Codage séquentiel de la chaîne des bins

Tout d'abord, nous proposons dans la figure 3.22 l'architecture du binarisateur dont les entrées sont l'élément syntaxique et le type de l'élément, et les sorties sont la chaines des segments (bins) avec la longueur associée. Dedans, nous distinguons les différents sous-blocs associés pour les 7 types de binarisation (FL,Unaire, …).

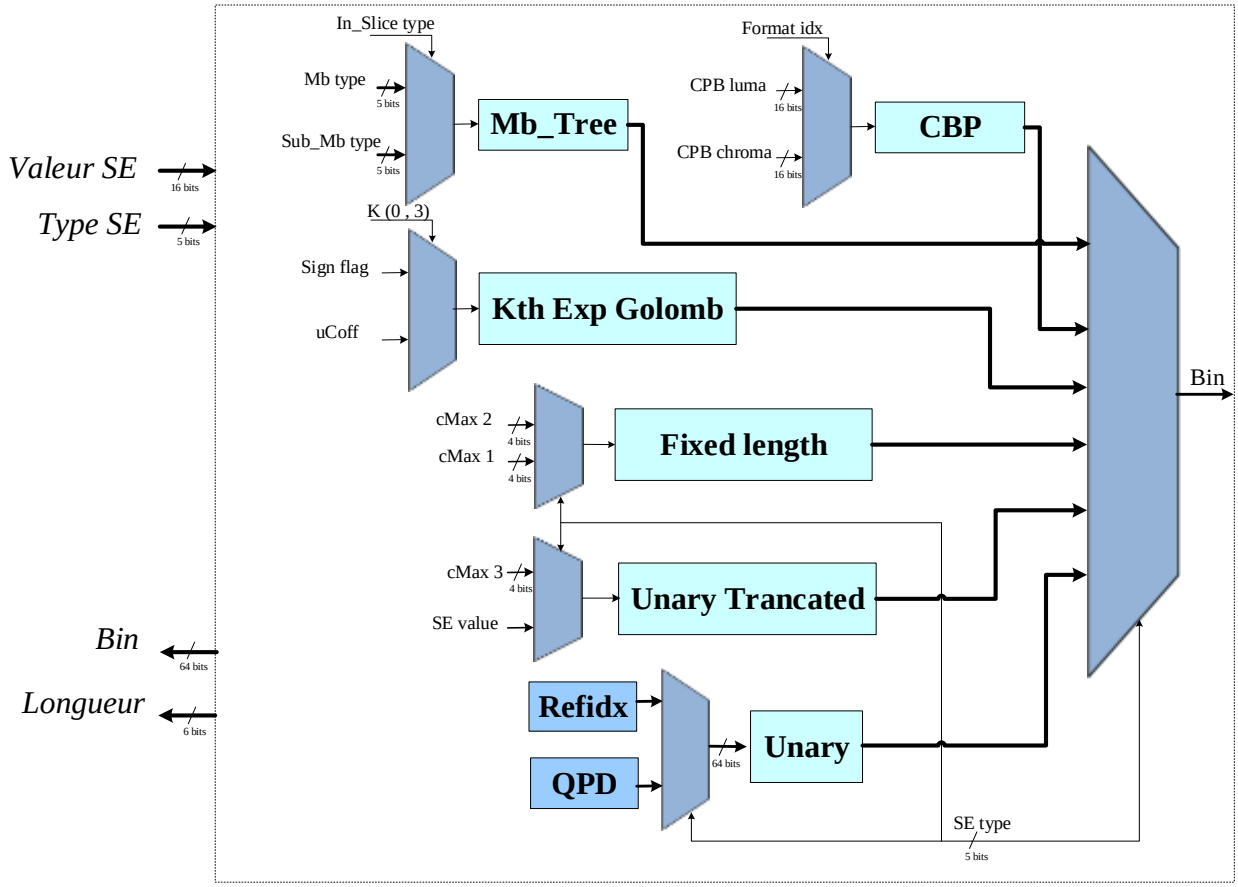


Figure 3. 22. Circuit du binarisateur

La binarisation des éléments syntaxiques mb_type ou sub_mb_type est présenté par le bloc Mb_Tree. Les chaînes de segments pour les types de macrobloc dans les tranches I, les types de macrobloc dans les tranches P/SP/ B et pour les types de sous-macrobloc dans les tranches P/SP/B sont spécifiés dans le draft.

Pour la binarisation de longueur fixe (FL), l'entrée est le paramètre cMax (1,2) avec cMax1 =1 et cMax 2= 7. Cette binarisation est en générale utilisée pour les symboles plus ou moins équiprobables. Pour “ref_idx”, le processus de la binarisation unaire est déclenché. Le code unaire est optimal pour un symbole caractérisé par une répartition de probabilité exponentielle. En cas de binarisation unaire tronquée, si les valeurs d'élément syntaxique sont inférieures à cMax3, le processus de binarisation U est invoqué,sinon, la chaîne de segments est une chaîne binaire de longueur cMax3 avec tous les segments égaux à 1.

La binarisation Exp-Golomb de kième ordre unaire concaténé (UEGk) est la concaténation de deux codes binaires : un code préfixe (TU) et un code suffixe (EGk de l’entier) comme décrit dans l’algorithme 3.1.

Algorithme 3.1 : Algorithme du codage Exp-Golomb de kième ordre unaire concaténé [Bardone 08]

symbol ← xynElVal-UCoff;

```
    TANT QUE (1){
    //la première partie unaire du UEGK
      Si (symbol>=(unsigned int)(1<<k)) {
      ECRIRE ('1'); // parte unaire
      symbol ← symbol -(1<<k);
      k++;    }
      Sinon {
        ECRIRE ('0'); //le zero de fin de la partie unaire
        symbol ← xynElVal-UCoff;
        TANT QUE (k--) // la deuxième partie du UEGK
        ECRIRE ((symbol >>k) & 0x01 );
        Stop;       }      }
         FIN TANT QUE
       FIN Si
    FIN TANT QUE
Fin
```

Le processus dépend des paramètres sign Flag et uCoff. Pour K= 3 (UEG3), nous trouvons sign Flag=1 et uCoff=9 pour l'élément syntaxique « mvd » (Tableau 3.6). Et pour k=0 (UEG0), sign Flag est égal à 0 et uCoff=14 pour l'élément syntaxique « coeff_abs_level_minus1 » (Tableau 3.7).

Tableau 3. 6. Binarisation du mvd (0 ≤ abs (mvd)) ≤ 9

mvd	Binarisation
0	0
1	10
2	110
3	1110
….	….
9	111111111

Tableau 3. 7. Binarisation UEG0 de quelques valeurs absolues des coefficients résiduels

Valeur	Préfixe	Suffixe
1	0	
2	10	
3	110	
4	1110	
….	….	….
13	1111111111110	
14	11111111111110	
15	11111111111111	0
16	11111111111111	100
17	11111111111111	101
18	11111111111111	11000
….	….	….

Après la binarisation, les bins seront traités un par un afin de sélectionner les probabilités correspondantes rLPS. Une mise à jour est nécessaire à effectuer suite à chaque état choisi comme le montre la figure 3.23.

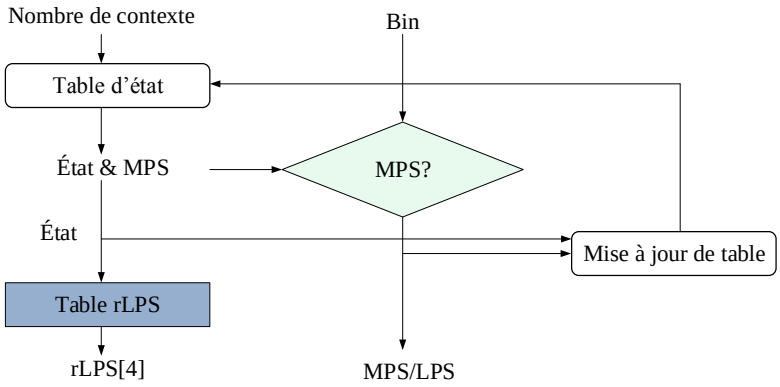


Figure 3. 23. Circuit du management de sélection du contexte, mémorisation du RLPS et la transition d'état

La probabilité, le MPS et les intervalles courants range et low seront à la suite transmis au bloc du BAC. Les sorties du bloc de codage sont les nouveaux range et low et les bits de sortie. L'architecture proposée est ainsi donnée sur la figure 3.24. Le circuit illustré démontre aussi la sélection des probabilités rLPS et le processus de mise à jour après le codage de chaque bin.

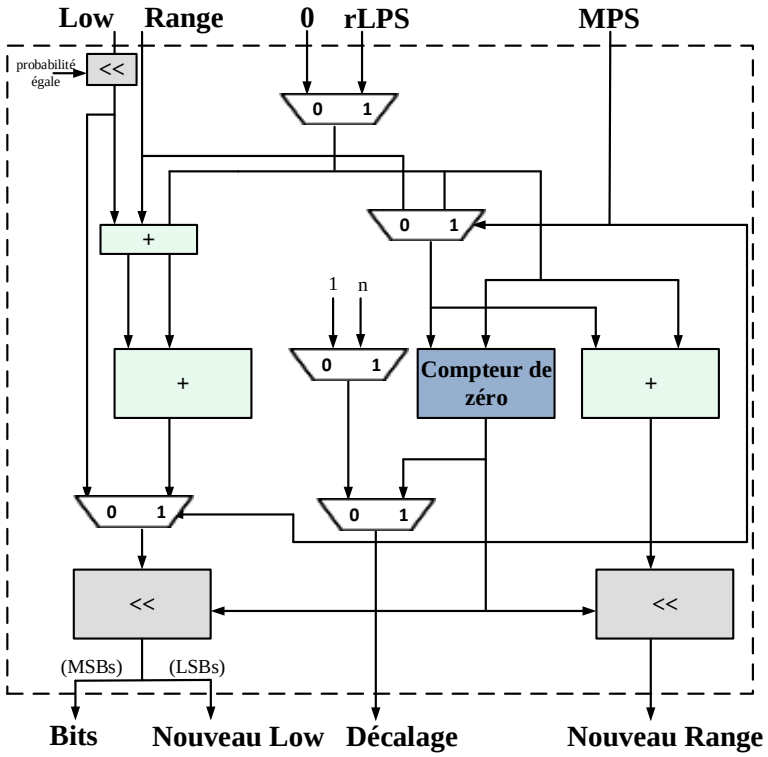


Figure 3. 24. L'architecture proposée pour le BAC régulier

Nous signalons à la fin que nous avons deux types de normalisation utilisées : normalisation régulière utilisé avec le BAC régulier et normalisation alternatif utilisé avec le BAC alternatif. Les algorithmes 3.2 et 3.3 décrivent les étapes effectuées avec chaque normalisation.

Algorithme 3.1 : Algorithme de la normalisation régulière

```
 TANT QUE (range < 256)
Si (low <256) alors {
Si low <512 alors {
   low ← low − 512
   PutBit(1)
   Sinon
   low ← low − 256
   bitsOutstanding ← bitsOutstanding + 1}
Sinon
PutBit(0)}
FIN Si
range ← range << 1
low ← low << 1
   FIN TANT QUE
```

Algorithme 3.1 : Algorithme de la normalisation alternative

```
 Si (low < 1024) alors {
    Si low < 512 alors {
    PutBit(0)
    Sinon
    low ← low − 512
    bitsOutstanding bitsOutstanding + 1}
    Sinon {
PutBit(1)
low ← low − 1024}
FIN Si
```

4.2.1.2. Résultats de synthèse

Nous présentons ci-dessous deux tableaux de synthèse résumant les différentes binarisateurs matériels nécessaire pour le binarisateur global du CABAC. Nous avons utilisé l'outil ISE de la société Xilinx pour synthétiser le code VHDL sur une cible FPGA Spartan 6. La fréquence maximum d'utilisation obtenue lors de la synthèse est environ 287 MHz.

Tableau 3. 8. Résultats de synthèse obtenus pour les binarisateurs matériels du CABAC

	U	UT	FL	TREE
Slices(%6822)	45	16	7	24
LUTS(%27288)	106	33	19	55
Flip Flop(%1072)	67	12	7	10
Slices Registers(%54576)	138	24	7	20
Slices(%6822)	886.643	886.643	729.26	886.643

	MQD	EG3	EG0	CBP
Slices(%6822)	39	16	20	8
LUTS(%27288)	121	30	39	14
Flip Flop(%1072)	75	12	21	5
Slices Registers(%54576)	76	42	60	12
Slices(%6822)	286.859	886.643	886.643	696.452

Tableau 3. 9. Résultats de synthèse obtenus pour le binarisateur global du CABAC

	Binarisateur global
Slices(%6822)	376
LUTS(%27288)	916
Flip Flop(%1072)	540
Slices Registers(%54576)	924
Max Frequency (MHZ)	286.859
Memory (%6408)	157
Number of RAMB16BWERs (%116)	6
Number of RAMB8BWERs (%232)	1

Après effectuer la synthèse du codeur arithmétique sur notre cible FPGA spartan 6, nous trouvons les résultats concernant l'implantation récapitulés dans le tableau 3.10. La fréquence maximum d'utilisation obtenue lors de la synthèse est 696 MHz pour le BAC régulier et 886 Mhz pour les BACs alternatifs et de finalisation.

Tableau 3. 10. Résultats de synthèse obtenus pour les trois codeurs arithmétiques : régulier, alternatif et de finalisation

	régulier	*alternatif*	*finalisation*
Slices(%6822)	14	11	9
LUTS(%27288)	52	28	33
Flip Flop(%52)	19	2	1
Slices Registers(%54576)	19	4	2
Max Frequency (MHZ)	696.452	886.643	886.643

4.2.1.3. Estimation de puissance

Les rapports de puissances obtenus après la synthèse du binarisateur et le BAC sont illustrés dans les tableaux 3.11, 3.12 et 3.13. Au niveau binarisateur, le rapport maximal est obtenu pour le binarisateur mqd d'environ 41.5 mW et le rapport minimal est obtenu pour le binarisateur CBP d'environ 36.6 mW, un rapport quasiment identique à celui du binarisateur global. Quant au BAC, nous avons un rapport de 57.51 mW pour la BAC régulier et 40.09 mW pour les deux autres BACs.

Tableau 3. 11. Rapports de puissance obtenus pour les binarisateurs du CABAC

	U	**UT**	**FL**	**TREE**
puissance dynamique (mW)	4.49	3.03	2.39	3.16
Clocks	3.50	2.22	1.8	2.12
Logic	0.20	0.11	0.3	0.24
Signals	0.52	0.40	0.21	0.52
Ios	0.27	0.30	0	0.29
puissance statique (mW)	36.22	36.19	36.18	36.20
puissance totale (mW)	40.71	39.22	38.57	39.36
	MQD	**EG3**	**EG0**	**CBP**
puissance dynamique (mW)	5.25	3.00	3.50	0.46
Clocks	3.81	2.36	2.83	0.46
Logic	0.03	0.11	0.12	0
Signals	0.19	0.27	0.28	0
Ios	1.22	0.27	0.27	0
puissance statique (mW)	36.23	36.19	36.20	36.15
puissance totale (mW)	41.48	39.19	39.70	36.61

Tableau 3. 12. Rapports de puissance obtenus pour le binarisateur global

	Binarisateur global
puissance dynamique (mW)	10.49
Clocks	6.80
Logic	0.70
Signals	1.64
Ios	1.36
puissance statique (mW)	36.33
puissance totale (mW)	46.82

Tableau 3. 13. Rapports de puissance obtenus pour les trois codeurs arithmétiques

	régulier	***alternatif***	***finalisation***
puissance dynamique (mW)	21.08	14.89	3.88
Clocks	3.63	9.30	2.58
Logic	0.23	0.42	0.07
Signals	1.34	3.67	0.47
Ios	15.89	1.5	0.77
puissance statique (mW)	36.53	35.77	36.21
puissance totale (mW)	57.61	40.09	40.09

4.2.2. Les IPs proposées pour le décodeur CABAD

Nous allons décrire l'implantation matérielle du décodeur CABAC et ses principaux blocs fonctionnels. La description de chaque bloc est quasiment identique au celui de l'encodeur. Nous avons parcouru les mêmes tables de rangTabLPS, initialisation et transmission d'état. En outre, le bloc RAM des modèles contextuels est aussi utilisé.

4.2.2.1. Architecture matérielle d'implantation

Les conceptions réalisées au niveau décodeur sont quasi symétriques aux ceux de l'encodeur. Le circuit du décodeur régulier est présenté par la figure 3.25. Le paramètre « offset » est d'origine le dernier paquet envoyé dans le flux binaire d'entrée. En effet, le flux binaire sera lu 32 bits puis 32 bits et son décalage sera effectué au fur et à mesure du décodage. Entre temps, les offset et range subissent toujours des mises à jour et normalisation pendant le processus de décodage.

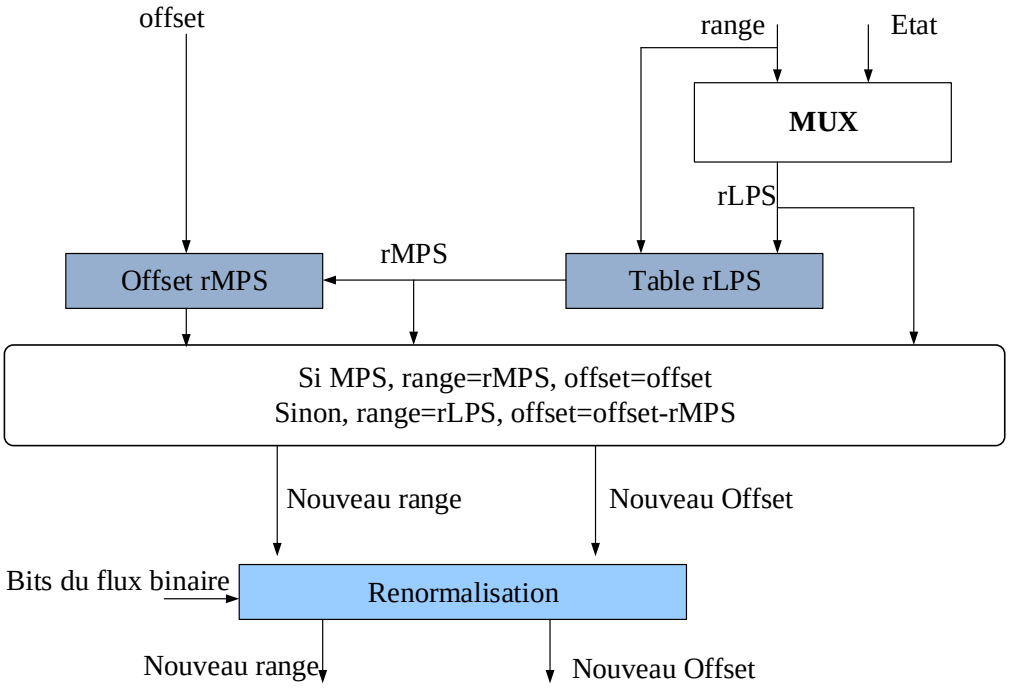


Figure 3. 25. Circuit du décodage régulier

L'architecture détaillée du BAD est donnée sur la figure 3.26. La sortie de ce bloc contient les bins qui seront à la suite débinarisés. Le bloc de débinarisation en ces divers types n'est pas présenté car il est quasi-similaire au bloc de binarisation.

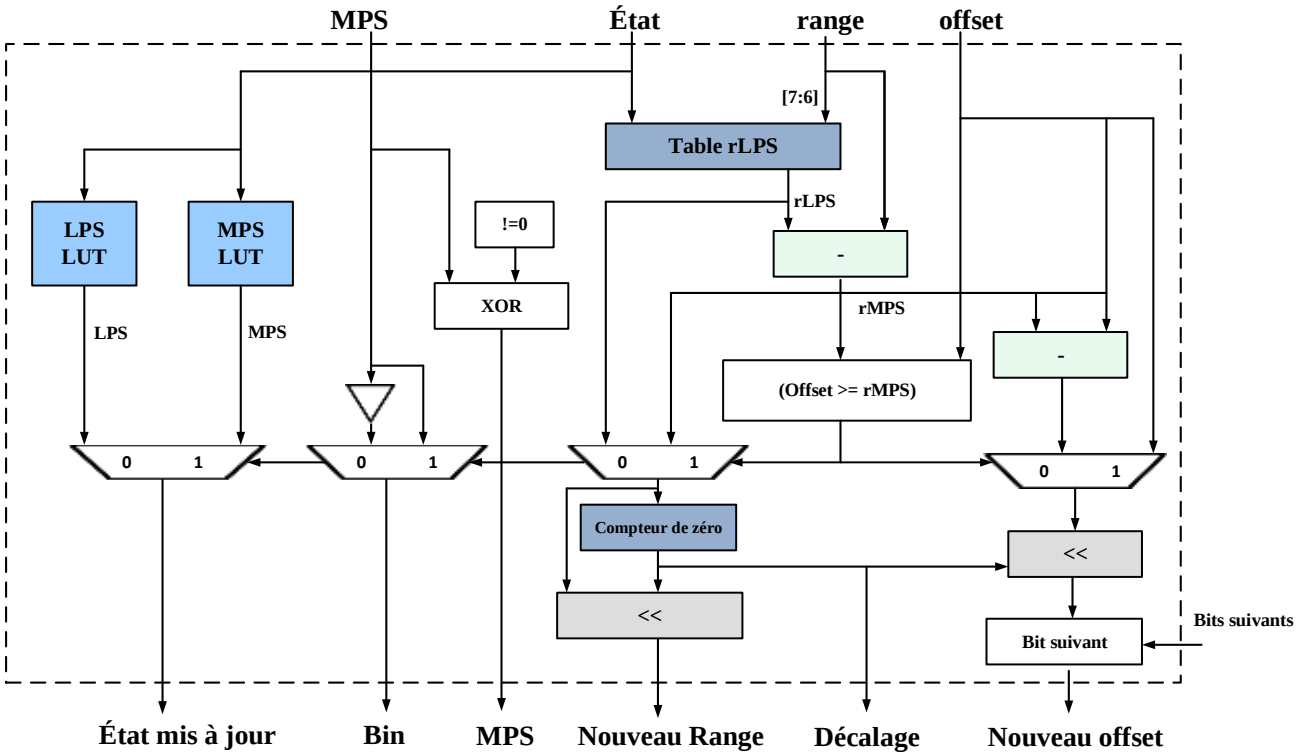


Figure 3. 26. Architecture du BAD régulier

4.2.2.2. Résultats de synthèse

Les deux tableaux ci-dessous présentent les résultats de synthèse résumant les différentes débinarisateurs matériels nécessaire pour le débinarisateur global du CABAD.

Les fréquences de fonctionnement de chacun des blocs sont estimées lors de la synthèse avec un minimum de 552 MHz pour le débinarisateur MQD.

Tableau 3. 14. Résultats de synthèse obtenus pour les débinarisateurs matériels du CABAD

	U	**UT**	**FL**	**TREE**
Slices(%6822)	57	22	12	24
LUTS(%27288)	145	42	17	41
Flip Flop(%1072)	8	4	3	5
Slices Registers(%54576)	13	8	8	10
Max Frequency (MHZ)	886.643	886.643	886.643	886.643
	MQD	**EG3**	**EG0**	**CBP**
Slices(%6822)	57	24	47	5
LUTS(%27288)	150	63	127	7
Flip Flop(%1072)	12	8	19	6
Slices Registers(%54576)	13	19	25	14
Max Frequency (MHZ)	552.181	886.643	886.643	625.293

Tableau 3. 15. Résultats de synthèse obtenus pour le débinarisateur global

	Débinarisateur global
Slices(%6822)	233
LUTS(%27288)	551
Flip Flop(%1072)	54
Slices Registers(%54576)	69
Max Frequency (MHZ)	552.181

Pour le débinarisateur global, les résultats de la synthèse montrent que le système fonctionne avec une fréquence maximale d'environ 552 MHz. Nous trouvons les résultats concernant l'implantation décodeur arithmétique sur notre cible FPGA récapitulés dans le tableau 3.16. La fréquence maximum d'utilisation est 255 MHz pour le BAD régulier et 310 Mhz environ pour les BADs altérnatifs et de finalisation.

Tableau 3. 16. Résultats de synthèse obtenus pour les trois décodeurs arithmétiques : régulier, alternatif et de finalisation

	régulier	*alternatif*	*finalisation*
Slices(%6822)	65	12	17
LUTS(%27288)	122	26	32
Flip Flop(%52)	62	18	24
Slices Registers(%54576)	95	35	48
Max Frequency (MHZ)	254.415	310.025	312.842

4.2.2.3. Estimation de puissance

Les rapports de puissances obtenus après la synthèse du débinarisateur et le BAD sont illustrés dans les tableaux 3.17, 3.18 et 3.19. Au niveau binarisateur, les rapports sont bien comparables (à l'entour de 39.6 mW environ). Le rapport du débinarisateur global est 53.15 mW.

Tableau 3. 17. Rapports de puissance obtenus pour les débinarisateurs

	U	**UT**	**FL**	**TREE**
puissance dynamique (mW)	3.33	0.47	3.6	4.34
Clocks	1.76	0.21	1.70	2.47
Logic	0.10	0.07	0.04	0.17
Signals	0.81	0.13	0.61	0.76
Ios	0.66	0.06	0.97	0.94
puissance statique (mW)	36.20	36.15	36.20	36.22
puissance totale (mW)	39.53	36.61	39.83	40.56
	MQD	**EG3**	**EG0**	**CBP**
puissance dynamique (mW)	3.76	3.53	3.42	3.16
Clocks	2.22	1.76	1.77	2.07

Logic	0.10	0.18	0.11	0.01
Signals	0.78	0.89	0.89	0.12
Ios	0.66	0.71	0.66	0.96
puissance statique (mW)	36.21	36.20	36.20	36.20
puissance totale (mW)	39.97	39.74	39.62	39.35

Tableau 3. 18. Rapports de puissance obtenus pour le débinarisateur global

	Débinarisateur global
puissance dynamique (mW)	16.71
Clocks	3.78
Logic	0.56
Signals	2.75
Ios	9.61
puissance statique (mW)	36.44
puissance totale (mW)	53.15

Pour le BAD, nous avons un rapport de 112.6 mW pour le BAD régulier, 63.31 mW pour le BAD alternatif et 78.19mW pour le BAD de fin. Ces rapports sont presque doublés à ceux du BAC direct.

Tableau 3. 19. Rapports de puissance obtenus pour les trois décodeurs arithmétiques

	régulier	***alternatif***	***finalisation***
puissance dynamique (mW)	75.06	26.68	41.29
Clocks	4.24	2.09	2.46
Logic	1.33	0.22	0.33
Signals	1.82	0.60	0.80
Ios	67.66	23.77	37.70
puissance statique (mW)	37.54	36.63	36.90
puissance totale (mW)	112.60	63.31	78.19

4.3. Test et validation du CODEC entropique sur FPGA

4.3.1. Environnement logiciel et matériel de la plateforme

Xilinx commercialise toute une gamme d'outils de développement pour exploiter ses composants. Les principaux outils sont :

- Xilinx ISE : Environnement de développement intégré
- Xilinx EDK : Environnement de développement intégré ciblant les processeurs intégrés au FPGA
- Xilinx ChipScope Pro : Outil de débogage temps-réel des circuits FPGA.

4.3.2. Validation fonctionnelle

Les modules de vérification dédiés sont des solutions intégrées au design cible. Le but principal est de pouvoir observer les signaux internes des circuits sans avoir à faire appel à des entrées/sorties supplémentaires sur les FPGA. Ce module devient alors une composante intégrée au système vérifié dans le FPGA. Il représente un coût en surface non négligeable, mais il donne un accès aux signaux, permettant d'observer des erreurs et donc réduire le temps nécessaire à la validation. Un exemple connu de ce genre de module est l'outil Chipscope Pro de Xilinx [Xilinx], spécialement conçu pour la vérification et le débogage des FPGA de Xilinx (Figure 3.27).

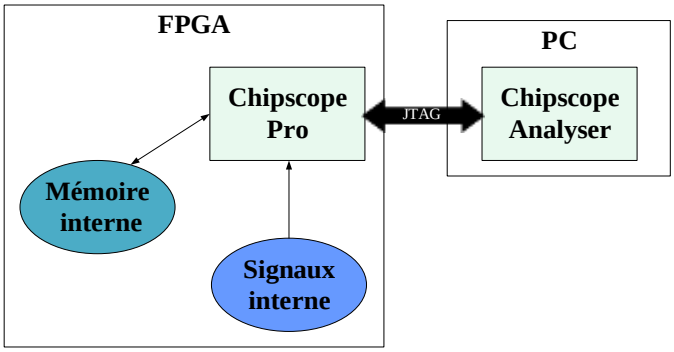


Figure 3. 27. Schéma de fonctionnement du module Chipscope Pro de Xilinx.

Figure 3. 28. Banc de test pour la validation réelle avec chipscope sur FPGA SPARTAN 6

Ce module utilise la connexion JTAG entre le FPGA et un ordinateur avec un logiciel de vérification pour capter les signaux observés dans le FPGA. Ainsi, il n'y a pas besoin d'entrées/sorties réservées pour la vérification. Chipscope Pro utilise les unités de mémoire interne des FPGA, les BRAM (Block RAM), pour mémoriser les signaux après ou avant un signal de déclenchement. Les avantages de Chipscope sont de pouvoir être synchronisé à l'horloge du système et de pouvoir intégrer les circuits de vérification directement en VHDL dans un design.

Les principaux inconvénients de ces outils sont que les BRAM des FPGA sont limités en espace de stockage, 2.1 Mbits pour le Spartan6 LX45. Également, les ressources de logique et de routage requises pour les modules de commande et la redirection des signaux peuvent être insuffisantes. Le principal concurrent de Xilinx, Altera, fournit un outil similaire à Chipscope pour ses propres FPGA : SignalTap II [Altera 13].

La figure 3.28 démontre une validation réelle sur notre cible FPGA des différents modules VHDL implantés. Comme exemple, la figure 3.29 et figure 3.30 représentent la sortie des binarisateurs et débinarisateurs, nous avons utilisé l'outil ChipScope afin de visualiser les sorties auquel nous appliquons toutes les combinaisons possibles des valeurs d'entrée.

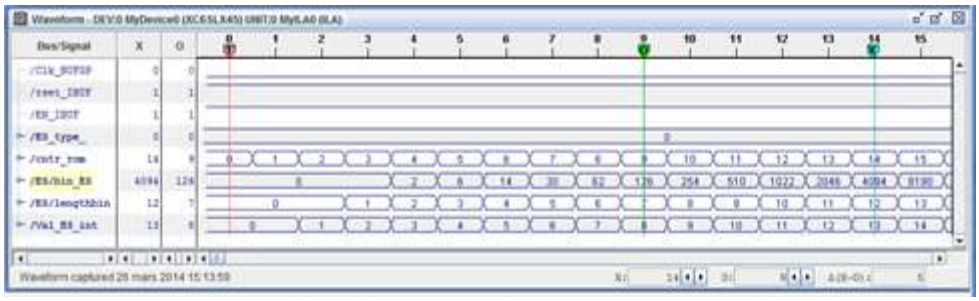


Figure 3. 29. Validation du binarisateur

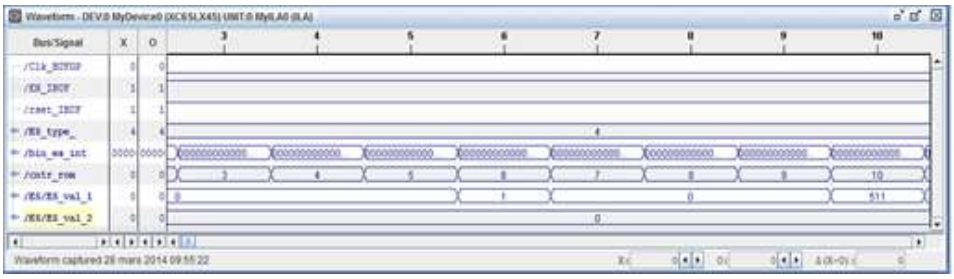


Figure 3. 30. Validation du débinarisateur

5. Conclusion

Dans ce chapitre nous avons détaillé les principes de fonctionnement de l'encodeur/décodeur CABAC. En raison de ces performances élevées et des fonctionnalités supportées, ce codeur entropique est adopté dans les derniers standards vidéo.

Nous nous sommes basés sur les processus décrits dans le draft et les différents algorithmes associés afin d'implanter et validé l'architecture du codec entropique proposée. Ainsi, nous avons réalisé une mesure de la puissance consommée des différents modules VHDL réalisés. La conception proposée permet d'atteindre l'objectif fixé concernant le prototypage rapide. Toutefois, un faible pourcentage des ressources matérielles a été utilisé pour réaliser les modules développés spécifiquement en langage matériel. Ceci nous permettrait d'envisager ajouter d'autres traitements de la chaine de CODEC vidéo comme l'estimation de mouvement.

Chapitre 4 : Cryptage et compression vidéo simultanés

1. Introduction

Au cours des deux dernières décennies, plusieurs travaux de recherche sont effectués en compression vidéo en raison des enjeux économiques [Alfalou 09]. Les industriels du domaine ont émis un grand intérêt pour des techniques de compression performantes et efficaces en termes de débit, qualité et complexité. En parallèle, le besoin de la protection des contenus vidéo est devenu un enjeu majeur aussi bien pour des applications militaires que civiles. Il a pris plus d'importance vu que les contenus numériques peuvent être facilement copiés ou modifiés. Cependant, le besoin de sécuriser des informations ne date pas d'aujourd'hui. En effet, depuis l'antiquité, on a cherché à envoyer des messages cachés i.e. sans que des personnes extérieures ne puissent les comprendre. Pour le faire, dans la littérature, deux méthodes sont utilisées : (1) la stéganographie qui consiste à cacher un dans un autre message pour que la personne non autorisée ne le trouve pas ; (2) la cryptographie qui consiste à rendre le message incompréhensible par l'ennemi [Havet 10].

Dans cette thèse, nous nous intéressons à cette deuxième catégorie. Traditionnellement, la cryptographie a été utilisée par les militaires, notamment durant la première guerre mondiale. Une étude extensive de l'art de cryptographie est illustrée dans [Kahn 96]. Aujourd'hui, la sécurisation de la vie privée (échange d'information par exemple) est devenue une préoccupation majeure vue l'utilisation massive des caméras de surveillance pour sécuriser les bâtiments, les rues... La confidentialité et le contrôle d'accès de données visuelles aux utilisateurs autorisés sont donc bien exigés. Pour de simples raisons (la protection de la vie privée, la protection commerciale, …). Diverses techniques sont utilisées pour assurer la protection des vidéos. Parmi ces techniques nous pouvons citer le tatouage, le chiffrement, la prise des empreintes digitales [Puech 2004] [Margot 00].

Dans ce contexte, la compression et le cryptage vidéo simultanés présente une solution originale, déportant la sécurité au flux vidéo codé. D'un premier abord, le cryptage et la compression paraissent comme deux technologies non apparentées. Chacune d'elles effectue une transformation sur les données pour garantir la sécurité ou la compression. La compression peut nuire au niveau de sécurité exigé et le cryptage risque de réduire l'efficacité de la compression. Lorsque ces deux opérations sont effectuées simultanément sur les données, nous allons voir que ni la sécurité, ni l'efficacité de compression ne doivent être

sacrifiées En plus, il faut souligner qu'un système de crypto-compression ne doit pas augmenter la complexité pour respecter les contraintes temps-réel.

Ce chapitre a pour but de présenter les aspects, les techniques et les algorithmes de cryptage d'images et de vidéos. Les procédures les plus utilisées de compression et cryptage simultanés mises en œuvre dans les standards de la compression d'images fixes et animées seront ensuite détaillées. L'objectif est de montrer les hautes performances, mais aussi les limites des codeurs vidéo modifiés générant ses flux de données protégés.

Notre contribution dans ce domaine consiste à proposer et valider une méthode de compression et de cryptage simultanés. Notre technique est basée sur une inversion des intervalles de codage du codeur arithmétique binaire à l'aide d'un nombre généré aléatoirement. Ce chiffrement à flot synchrone consiste à combiner (généralement pas un ou-exclusif bit à bit) le message en clair avec une suite binaire secrète. La suite binaire secrète provient d'un générateur de nombre pseudo-aléatoire ou d'une clé biométrique. Les résultats de simulations en terme de complexité algorithmiques et de niveau de sécurité sont comparés favorablement avec des méthodes récentes. Enfin, nous revenons sur la robustesse des techniques de cryptage et compression contre les attaques cryptographiques.

2. Généralités sur le cryptage

La cryptographie est un ensemble des techniques permettant de protéger un message au moyen d'un code secret. Elle rend l'information inintelligible et assure l'authentification des partenaires (identification). Elle permet de limiter l'accès aux données (mot de passe) ce qui garantit la confidentialité et le contrôle d'accès. Les techniques de cryptage et décryptage sont basés sur des notions mathématiques liées à la sécurité de l'information [Buchmann 04]. Avec la puissance évolutive des ordinateurs et les grandes avancées mathématiques, la cryptographie a fait d'énormes progrès ce qui élargit son domaine d'utilisation : les transactions bancaires, les documents confidentiels de l'armée ou de gouvernement, l'envoi d'émail, les conversations avec téléphone cellulaires, etc. Dans la littérature, on distingue plusieurs classes et techniques de cryptage.

2.1. Cryptosystèmes classiques

La cryptographie est essentiellement basée sur l'arithmétique permettant de transformer le message visible en un message codé (crypté).

Les transformations utilisées sont des fonctions mathématiques appelées algorithmes cryptographiques permettant de crypter des messages avec un paramètre appelé clé. Afin de

transmettre un message de manière confidentielle, nous lui appliquons une transformation qui le rend incompréhensibles (secret) ; c'est ce qu'on appelle le cryptage. Cette modification donne un texte illisible (cryptogramme) à partir d'un texte en clair.

Inversement, le décryptage est le procédé inverse dont l'unique objectif est de retrouver le message original (en clair). Le décryptage est l'action consistant à retrouver le texte en clair sans connaître la clé de cryptage comme le montre la figure 4.1.

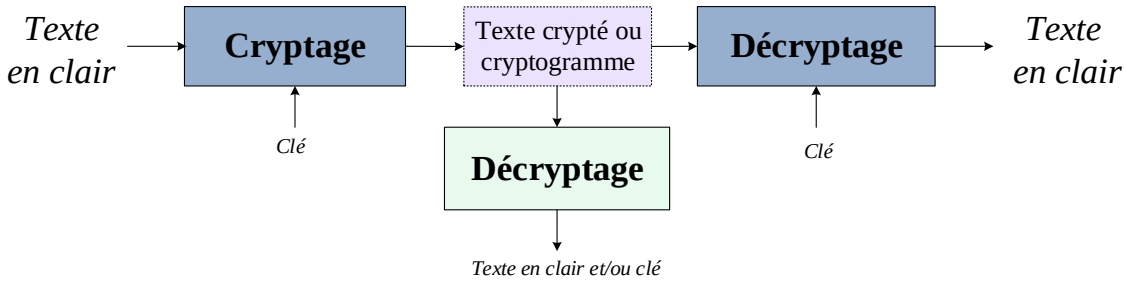


Figure 4. 1 - Schéma bloc d'un cryptosystème

Un cryptosystème est alors principalement constitué par des algorithmes de chiffrement et de déchiffrement

2.2. Classification des algorithmes cryptographiques

Chaque système de cryptage est composé d'un algorithme de codage et une ou plusieurs clés de sécurité en se basant sur des théories mathématiques pour rendre la communication inviolable. Parmi une grande variété de mécanismes de cryptage, nous distinguons deux grands types de cryptographie : la cryptographie à clé publique (ou cryptographie symétrique) et la cryptographie à clé privé (ou cryptographie asymétrique) [Buchmann 04].

2.2.1. Cryptographie symétrique

La cryptographie à clé symétrique ou privée est la technique de cryptage la plus utilisée en raison de sa simplicité. Dans la cryptographie à clé symétrique, le cryptage et le décryptage sont effectués en utilisant la même clé secrète comme le montre la figure 4.2.

Figure 4. 2 - Schéma bloc de la cryptographie symétrique

Le problème majeur de cette méthode est qu'il faut trouver le moyen de transmettre d'une manière sécurisée la clé au correspondant [Whitman 07].

2.2.2. Cryptographie asymétrique

La cryptographie à clé asymétrique ou publique utilise une clé pour le chiffrement et une autre pour le déchiffrement des messages, comme illustré sur la figure 4.3.

Figure 4. 3 - Schéma bloc de la cryptographie asymétrique [Havet 10]

La première clé de cryptage ou clé publique est connue par tout le monde, alors que la deuxième clé de décryptage également connue sous le nom de clé privée est secrète. Chaque utilisateur possède une paire de clés (privée et publique) : une pour le cryptage, l'autre pour le décryptage. Les clés publiques et privées sont mathématiquement liées et les données cryptées avec la clé publique ne peuvent être décryptées qu'avec la clé privée correspondante, ce qui garantit d'avantage la confidentialité des messages cryptés. En cryptographie à clé publique, les fonctions à sens unique avec trappe [Acrypta] sont utilisées. Pour ce type de fonction, le sens inverse est difficile sans certaines informations (la trappe est la clé dans ce cas). Dans la cryptographie à clé privée, l'utilisateur distribue sa clé publique et garde secrète sa clé privée. Dans ce type d'application, tout le monde peut lui écrire en utilisant la clé publique, mais seul l'utilisateur destinataire pourra décrypter et donc lire le message avec sa clé privée [Vivet 97]. Des exemples typiques de cryptage asymétrique sont RSA, ElGamal [Bresson 04] [Vaudenay 06] et les courbes elliptiques [Blake 99]. Les longues clés sont utilisées pour rendre le système plus sécurisé. Les longueurs de clés typiques sont 768, 1024, 2048 bits (RSA, ElGamal) [Bresson 04]. Les clés basées sur les courbes elliptiques ont des longueurs typiques de 150 à 200 bits. Les longueurs de clé des différents algorithmes ne peuvent pas être comparées directement parce qu'elles reposent sur des propriétés mathématiques différentes, par exemple, certains chercheurs disent que 170 bits pour les courbes elliptiques fournissent à peu près le même niveau de sécurité que 1024 bits pour RSA [Cheddad 10].

2.3. Techniques de cryptage pour une image fixe et animée

2.3.1. L'algorithme DES

L'algorithme DES (Data Encryption Standard) [NIST a] crypte le texte bloc par bloc contrairement à d'autres méthodes qui cryptent les données bit par bit ou octet par octet

(algorithmes de cryptage par flots). C'est un algorithme à clé secrète inventé en 1975 par IBM [NIST a]. Il est très populaire ; il est utilisé dans le standard de chiffrement du gouvernement américain et l'armée, le système de mots de passe UNIX, ect. Cet algorithme symétrique utilise des clés de 56 bits et chiffre le texte par bloc de 64 bits.

Il consiste à effectuer des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens. Son exécution est donc très lente et la taille de clé est faible permettant une attaque systématique en un temps raisonnable [DES]. Chaque utilisateur choisit sa propre clé et il doit la communiquer aux divers destinataires pour permettre le décodage de ses messages ; ce qui constitue le point faible du DES. On utilise généralement le Triple DES, qui est l'enchainement de 3 DES simples dans l'ordre DES / DES^{-1} / DES, sans avoir de meilleurs performances.

2.3.2. L'algorithme AES

Le standard AES (Advanced Encryption Standard) est un des standards de chiffrement avancé adopté en 2001 par le NIST [NIST b]. Il est destiné à remplacer le DES (trop faible au regard des attaques actuelles), pour les organisations du gouvernement des États-Unis. Cet algorithme opère sur des blocs de 128 bits qu'il transforme en blocs cryptés de 128 bits par une séquence d'opérations ou "rounds". Il supporte différentes combinaisons pour [longueur de clé]-[nombre de rounds] : [128-10], [192-12], [256-14]. L'AES répond aux mêmes exigences que le 3DES, qui est son concurrent le plus direct, par contre il est beaucoup plus sûr et flexible que son prédécesseur car il présente une plus grande résistance aux attaques par dictionnaires de clés. Mais, la gestion des clés est non simplifiée. Son utilisation est aussi bien pratique car il consomme peu de mémoire, il est moins complexe et plus facile à implémenter.

2.3.3. L'algorithme RSA

Cet algorithme de cryptographie asymétrique a été inventé en 1977 par trois mathématiciens : Ron Rivest, Adi Shamir et Len Adleman [wikipedia a]. Il est basé sur les nombres premiers. Son principe repose sur le fait qu'il est difficile et très long de factoriser un grand nombre en deux facteurs premiers. Il s'agit de générer un nombre à partir de deux grands nombres premiers (plus de cent chiffres pour le représenter en système décimal) qui doivent être secrets. Ce système de chiffrement est beaucoup utilisé dans le commerce électronique (paiement en ligne, etc...), les cartes bancaires et les logiciels (OpenSSH…).

Contrairement aux systèmes de chiffrement symétriques, le système RSA ne nécessite pas de transfert de clé entre l'expéditeur et le destinataire. Aucune personne en plus des utilisateurs concernés ne peut comprendre le message chiffré. Autrement, le propriétaire de la clé privée uniquement peut lire le message chiffré avec la clé publique correspondante, donc le RSA assure la confidentialité.

En plus, le système RSA est basé sur les fonctions à sens uniques. Pour inverser la fonction, il faut avoir la clé privée. Le propriétaire de la clé privée peut signer un message (avec la clé privée). Une signature déchiffrée avec la clé publique prouvera donc l'authenticité du message. Malheureusement, cet algorithme est beaucoup plus lent que n'importe quel système symétrique. En plus, il existe plusieurs attaques contre le système RSA comme les attaques qui ont montré la vulnérabilité de cette approche comme : Wiener [Stern 04], Hastad [Stern 04], l'attaque du milieu [Boneh 00]. Afin de se prémunir contre les puissances de calculs grandissantes, on utilise des clés de grandes tailles (actuellement de 2048 bits).

2.3.4. L'algorithme de cryptage par flot

Cet algorithme de cryptographie symétrique est constitué d'un générateur de nombres pseudo-aléatoires, qui étend la clé en une suite chiffrante, et d'une opération de chiffrement « XOR » entre un bit à la sortie du générateur et un bit provenant des données ou entre des mots binaires de taille supérieure [Berbain 07]. On peut aussi appliquer l'opération d'addition dans un groupe. Le chiffrement par flot peut être obtenu aussi par la conversion d'un chiffrement par bloc en utilisant un mode opératoire qui permet de chaîner plusieurs blocs et traiter des données de taille quelconque. Des exemples de chiffrements par flot : A5/1 (1994) utilisé dans les téléphones mobiles [Briceno 99] de type GSM, RC4 (1987) utilisé par le protocole WEP du Wi-Fi, Py, ect [Rivest 92].

Un chiffrement par flot permet d'atteindre un très haut niveau de performances en termes de vitesse de chiffrement et d'efficacité matérielle. Une des principales caractéristiques des algorithmes de chiffrement par flot est qu'ils arrivent à traiter les données de longueur quelconque et n'a pas besoin de les découper. Mais, son inconvénient majeur est la nécessité d'échange au préalable d'une clé secrète aussi longue que le message à crypter.

2.3.5. L'algorithme de cryptage sélectif

Pour assurer la confidentialité, le niveau de protection peut être adapté en fonction de l'application et du temps disponible. En effet, un contenu vidéo destiné aux militaires doit être complètement crypté, alors que dans l'industrie du spectacle, il suffira de cacher une partie

d'un film pour qu'il soit de mauvaise qualité et par conséquent immuniser contre les attaques. Ainsi, la sécurisation des transferts de contenus multimédias peut se faire soit par cryptage total soit par cryptage partiel (sélectif), où les utilisateurs peuvent appliquer une sécurité proportionnelle ou réglable en fonction du niveau de protection désiré [Norcen 03]. Alors, un cryptage sélectif peut être suffisant pour de nombreuses applications, excepté les applications militaires et ou civiles nécessitant un cryptage total.

Le cryptage sélectif d'une image ne crypte qu'une partie précise des informations de toute l'image (exemple : les hautes fréquences). Il peut être utilisé par exemple pour des images acquises par une caméra de surveillance [Rodrigues 06]. Le cryptage sélectif diminue la quantité de données à chiffrer, et par conséquent le temps de calcul. De plus, ce cryptage peut être intégré à l'intérieur d'un codeur (JPEG, H264/AVC) générant un flux de sortie conforme aux normes. De ce fait, un décodeur classique aura accès à l'information basse résolution et un décodeur adapté (ayant la clé secrète) pourra décoder correctement l'information cryptée. Du point de vue sécurité, le cryptage sélectif garantit un certain niveau de confidentialité qui reste toujours inférieur à celui d'un chiffrage complet.

2.3.6. Autres techniques

Ils existent différentes autres approches que nous pouvons différentier selon où nous effectuons le cryptage lors de la compression vidéo : avant ou après compression, avant ou après transformation et quantification, lors du codage entropique ou après (sur le flux binaire). Si le cryptage est effectué avant la compression, nous avons une grande influence négative sur les performances de compression vidéo. Ainsi, si la totalité de l'information visuelle doit être caché, cette approche ne présente pas le bon choix vu que nous pouvons deviner très facilement les contours d'objets. Une autre solution consiste à crypter les données de l'image et coder des informations modifiées, par exemple une permutation des positions des pixels est proposé dans [Carrillo 08]. Pour des standards de compression vidéo avec perte comme H.264, nous nous demandons toujours si le décryptage après compression avec perte est possible et comment nous gardons les performances de compression qui est plus facile dans la chaîne de compression [Dufaux 08], [Dufaux b 08], [Lian 07].

2.3.6.1. Cryptage dans l'encodeur vidéo

Divers travaux proposent un cryptage dans la chaîne de compression vidéo. Une modification du mode de prédiction Intra, qui doit être signalée dans le flux binaire, a été proposée dans

[Noorkami 07], [Ahn 04], [Lian 06], [Lian 08], [Su 10]. Dans [Li 05], l'auteur a proposé d'effectuer des permutations de l'information envoyée sur les macroblocs inter-prédits ayant le même nombre de vecteurs de mouvement (les informations de partition de macroblocs, le mode d'inter-prédiction et les données de vecteur de mouvement). Une modification des vecteurs de mouvement et MVD a été proposé dans [Noorkami 07], [Thomas 07], [Li 05], [Lian 05], [Ahn 04], [Lian 08], [Su 10], [Liu 07].

Certains proposent des cryptages des signes des éléments syntaxiques [Thomas 07], [Lian 05], [Su 10], [Won 06], et d'autres parlent du cryptage des suffixes des codes de Golomb exponentiel [Noorkami 07], [Lian 08]. De nombreux travaux modifient les signes des coefficients transformés [Dufaux a 08], [Noorkami 07], [Lian 07], [Lian 05], [Lian 06], [Lian 08], [Shahid 09], [Won 06], [Kim 07], [Lee 06], [Li 08]. Dans [Lian 06], ils ont proposé de crypter les coefficients DC pour des macroblocs intra. Ainsi, les auteurs de [Shahid 09] cryptent les coefficients non nuls qui subissent un codage Exp- Golomb avant le codage arithmétique binaire. Dans [Magli 06] et [Grangetto 07], les auteurs parlent du cryptage des bits les moins significatifs uniquement d'un coefficient (cryptage transparent). Mais, toutes ces techniques de cryptage sélectif ne permettent pas d'apporter une sécurité maximale. Beaucoup d'algorithmes non sécurisés sont employés en ajoutant des constantes avec les coefficients [Lee 06], ce qui réduit l'efficacité de la compression.

2.3.6.2. Cryptage dans le module du codeur entropique

Avant le codage entropique de chaque bloc 4x4, les coefficients non nuls sont mappés à une séquence de balayage par le zigzag. Une modification dans l'ordre de mappage a été proposé dans [Dufaux b 08] et [Su 10]. Beaucoup de travaux parlent aussi de la compression et cryptage conjointes pour les deux types des codeurs entropiques CAVLC et CABAC. Avec le CAVLC, un grand nombre de données est encodé avec les codes d'Exp- Golomb, qui peuvent être fréquemment cryptés [Bergeron 05], [Lian 05], [Lian 08]. Les mots de code des éléments syntaxiques peuvent être permutés comme les « run before » dans [Mian 07]. Si le CABAC est employé pour le codage des données, seuls les bits codés en mode « bypass », comme le MVD, doivent être cryptées afin de garder les performances du codeur [Zou 10], [Stutz 11]. La sécurité repose sur l'incertitude de l'état de décodage arithmétique, ce qui signifie que l'élément syntaxique à décoder et l'état modèles contextuels [Richardson 03] doivent être devinés, ainsi que divers variables de décodage (CodIRange et codIOffset représentés avec 16 bits) ce qui rend le décryptage très complexe pour un attaquant.

2.3.6.3. Cryptage dans le flux binaire

Le cryptage partiel du flux H.264 peut réduire la quantité de données à chiffrer, ce qui conduit à des améliorations de performance de système de diffusion [Hellwagner 09]. On peut se limiter sur le chiffrement d'IDR ou le chiffrement d'entête d'une NALU [Almasalha 08] qui nécessite un décodeur non classique mais qui peut être insuffisant pour assurer la sécurité [Hellwagner 09].

Une approche pour réduire la complexité de calcul du cryptage sélectif du flux binaire avec un chiffrement sécurisé [Massoudi 08] .Dans [Fan 07], il est proposé d'employer un AES pour les parties les moins importantes du flux binaire. Puisque le flux de bits est constitué des NALUs, on peut appliquer un cryptage simple sur le format entier comme le cryptage par paquet dans JPEG2000 [Engel 09]. On trouve aussi des systèmes de cryptage qui éliminent les séquences de marquage [Hellwagner 09]. Un cryptage des NALUs en conservant les entêtes des unités a été proposé dans [Arachchi 09], [Hellwagner 09] et [Zou 06]. Plusieurs propositions [Iqbal 06],[Iqbal 07],[Kuschnig 08] cryptent des fractions du flux binaire H.264/AVC avec des algorithmes classiques comme l'AES. En outre, Il existe une norme pour la diffusion sécurisée de données RTP, y compris H.264 : AVC / SVC [ISMA 07].

2.3.7. Discussion

Les travaux cités au-dessus ne permettent pas de garantir une sécurité absolue des images et des vidéos (JPEG, JPEG 2000, H264, HEVC). Toutefois, une erreur sur un bit crypté peut propager des erreurs importantes. Si l'image contient des zones homogènes, ses blocs identiques sont identiques après cryptage. Dans ce cas, l'entropie de l'image n'est pas maximale. Comme nous avons cité, certains cryptages garantissent un niveau de confidentialité limité. D'autres échangent des clés secrètes trop longues. De plus, la plupart des systèmes de cryptage augmentent la taille des données, ou encore la complexité calcul. Pour pallier à ces inconvénients, nous allons démontrer dans la partie suivante une technique explorée qui combine la compression de données avec le cryptage dans un seul processus.

3. Compression vidéo et cryptage simultanés proposés

Le cryptage est la façon la plus utile pour résoudre le problème de la confidentialité. Cependant, les algorithmes classiques de chiffrement ne sont pas capables de crypter une grande quantité de données pour des applications en temps réel. La compression et cryptage simultanés peut être alors une solution robuste qui n'entraine pas une augmentation du temps

de calcul tout en assurant une certaine sécurité et gardant la taille des données codées. Dans cette section 3, nous proposons des techniques de cryptage pour des séquences vidéo comprimées au format de vidéo H.264 et HEVC. Nous allons les décrire une par une en montrant leurs schémas de blocs et en expliquant les algorithmes utilisés. Une méthode de cryptage conjointe à la compression vidéo est aussi présentée à la fin de cette section et qui sera comparée avec les simples procedures de cryptage introduits.

3.1. Principe de compression

Le schéma de la compression choisi dans cette partie est indiqué dans la figure 4.4. Il s'agit d'un schéma de compression simplifié qui peut être utilisé pour une compression d'image ou une partie de la chaîne de compression vidéo. Après la décomposition en blocs de l'image, nous effectuons la transformation et la quantification des coefficients de l'image. Puis, nous opérons un mappage zig zag pour arranger les coefficients d'une manière spécifique et nous terminons par un codage entropique des symboles quantifiés. Nous utilisons un codeur entropique de type BAC pour les données vidéo binarisées (voir chapitre 1 pour les détails et annexe pour l'algorithme).

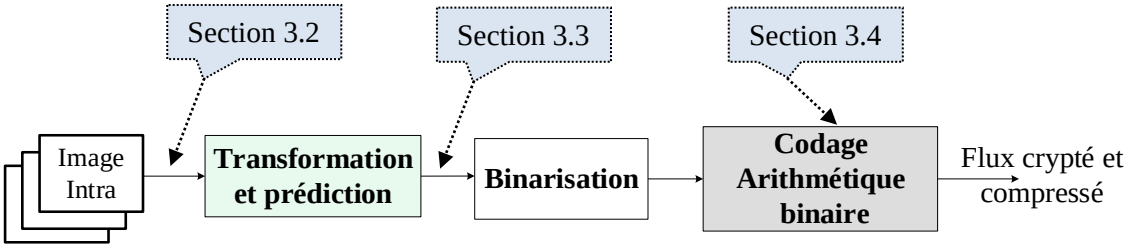


Figure 4. 4 –Modèle de la chaîne de compression vidéo adopté
avec un cryptage des données dans différents niveaux :
(3.2) cryptage des données brutes (3.3) cryptage des coefficients transformés et quantifiés
(3.4) cryptage conjoint au codage entropique

Toutefois, nous pouvons appliquer une ou plusieurs techniques de cryptage dans la chaîne de compression. Dans notre travail, nous adoptons divers méthodes pour crypter les données vidéo. Elles sont appliquées dans divers niveaux dans la chaîne et elles sont détaillées dans les sous-sections suivantes. Comme illustré dans la figure 4.4, le codeur vidéo peut être interrompu à tout niveau pour effectuer le cryptage. On distingue 3 étages différents :

- Cryptage des données brutes réalisé avant la compression (décrit dans la section 3.2)
- Cryptage des coefficients transformés et quantifiés réalisé après la quantification (décrit dans la section 3.3)

- Cryptage simultané au codage entropique appliqué sur les informations binaires de l'image (décrit dans la section 3.4)

Avant de se lancer dans les différents tehcniques de cryptage, il est opportun de préciser les moyens avec les quels on peut juger la qualité du cryptage. On touche ici un sujet assez sensible et complexe à la fois. En effet, du moment que l'on connait la méthode de chiffrement, un jeu exhaustif de clé de cryptage peut être appliqué afin de décrypter le contenue de l'image ou de la vidéo.

Néanmoins, quelque critères d'évalaution du niveau de chiffrement existent dans la littérature. La métrique classique utilisée pour notre comparaison est le PSNR considérée comme une mesure très indicative qui dépend des encodeurs et leurs formats [Horner 92]. Vu que le PSNR et l'EQM (équations 1.18), nous proposons d'utiliser en plus du PSNR, PCE (Peak to Correlation Energy) [Horner 92], qui est un critère de performance très utile pour obtenir une comparaison efficace entre une image cible et une image de référence. Le PCE peut être vu comme une mesure de similitude entre l'image cible et l'image de référence. Pour obtenir le PCE, il faut corréler les spectres de l'image cible et référence et analyser le pic de corrélation. Le PCE représente le rapport entre l'énergie contenue dans le pic de corrélation et l'énergie contenue dans le plan de corrélation.

$$\mathrm{PCE} = \frac{\text{Energie du pic de corrélation } (\mathrm{E_{pic}})}{\text{Energie du plan de corrélation}(\mathrm{E_{plan}})} \tag{4.1}$$

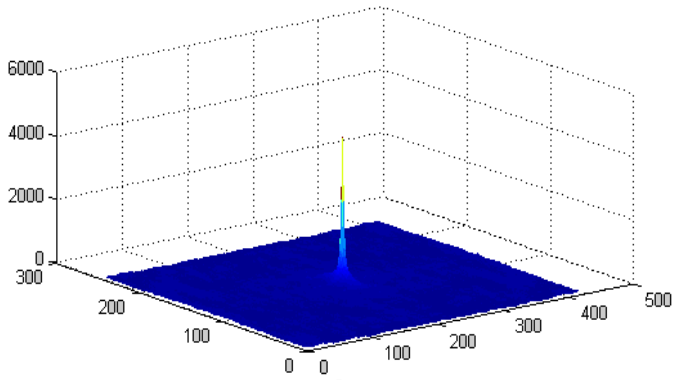
$$\mathrm{E_{pic}} = \sum_{x=x_0-t}^{x=x_0+t} \sum_{y=y_0-t}^{y=y_0+t} |C(x,y)|^2 \tag{4.2}$$

$$\mathrm{E_{plan}} = \sum_{x=1}^{x=n} \sum_{y=1}^{y=m} |C(x,y)|^2 \tag{4.3}$$

où x_0 et y_0 représentent la position du pic de corrélation et C représente le plan de corrélation. En effet, la corrélation consiste à multiplier le spectre d'une image cible avec toutes sortes de filtres [Alfalou 10]. Plusieurs filtres ont été proposés, comme le filtre de phase pure (filtre POF: Phase Only Filter) [Horner 84], le filtre adapté [Lugt 64], le filtre composite [Tocnaye 97] et le filtre segmenté [Alfalou 99].

Pour illustrer l'importance du critère PCE, la figure 4.5 (a) montre l'image Intra d'origine cible sur laquelle la compression et le cryptage seront appliqués (dans les sections suivantes Dans le cas de corrélation de cette image avec elle même (autocorrélation), nous pouvons constater que le plan de corrélation (figure 4.5 (b)) montre un pic d'amplitude important. de important : il s'agit ici d'autocorrélation.

(a)

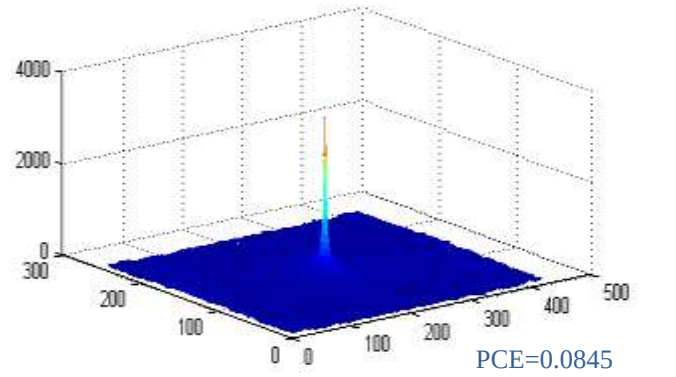


(b)

Figure 4. 5 - Image intra de la vidéo d'entrée

La figure 4.6 montre le résultat de la décompression de l'image Intra d'origine (figure 4.5) suite à l'application des traitement de la figure 4.4. Il faut noter que dans un premier temps, aucun cryptage n'est appliqué à l'image d'entrée. Ainsi, l'mage reconstruite après un codage Intra a subit une compression sans perte (BAC). La dégradation de la qualité visuelle est dûe alors à la DCT et à la quantification. Les valeurs des PSNR sont relevés sur la composante luminance pour des raisons de simplification de la de présentation. Notons aussi que l'image Intra est de taille 420*240 pixels.

(a)



(b)

Figure 4. 6- Image reconstruite après le codage intra

Pour un pas de quantification QP = 32, nous avons trouvé un PSNR=21.4804 dB, PCE=0.0845 et un rapport de compression RC= 83.278 %. Cette figure va nous servir comme référence pour mesurer les performances de notre méthode de compression et de cryptage proposée.

Dans les parties qui suivent, nous allons voir l'influence d'application des simples techniques de cryptage sur la qualité de vidéo reconstruite en comparant les métriques considérées (PSNR, PCE et RC) à chaque fois. Autrement, nous testerons le cryptage XOR avant et après

la transformation et la quantification des données vidéo, nous appliquerons un cryptage simultanné au codage entropique et nous combinerons les deux approches ensemble (RAC + XOR). Ceci nous permettra de choisir à la fin l'approche qui nous donne le bon compromis entre les métriques cités.

3.2. Compression et cryptage des données brutes (C&C DB)

Comme un premier niveau de sécurité, nous avons intégré un chiffrement à flot synchrone. Ce chiffrement consiste à combiner (généralement pas un ou-exclusif bit-à-bit) le message en clair avec une suite binaire secrète. La suite binaire secrète provient d'un générateur de nombre pseudo-aléatoire ou par une clé biométrique.

Dans notre chaîne de compression nous avons utilisé une clé biométrique pour ce premier niveau de sécurité. Cette clé est générée à partir d'un échantillon biométrique (une empreinte digitale/ un gabarit de minuties). Au début, nous avons extrait la valeur minimale et la valeur moyenne des pixels contenus dans la clé biométrique pour fabriquer une image pseudo-aléatoire. La taille de la clé aléatoire est la même que l'image Intra à crypter. Ensuite, nous appliquons le cryptage tel qu'il est indiqué dans la figure 4.7.

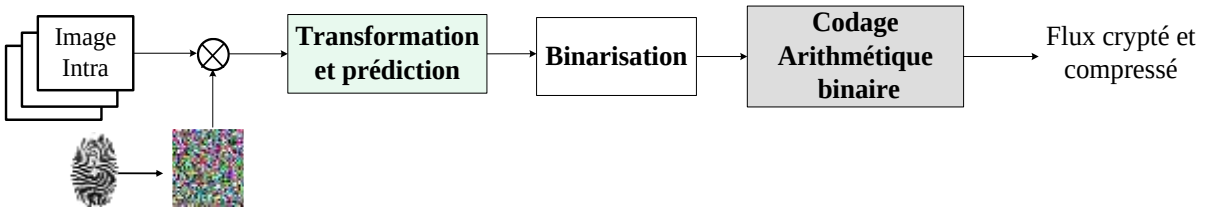


Figure 4. 7– Cryptage à base de clé biométrique appliquée sur les données brutes avant compression vidéo

En effet, cette technique est bien simple. Elle est basée sur un « OU exclusif » entre l'image à chiffrer et la clé qui doivent être en binaire. Chaque bit des données brutes (Image Intra) est combiné avec un bit de la clé de cryptage secrète selon la table de vérité de la fonction booléenne XOR. Ainsi, si le bit d'information et le bit de clé sont identiques, le bit du cryptogramme émis par l'algorithme sera 0, sinon il sera 1. Pour le décryptage, les bits sont à nouveau combinés un à un avec un bit de la même clé biométrique utilisé dans le cryptage. Notre clé est absolument privée dans cette approche. L'algorithme du cryptage XOR est présenté ci-dessous. La clé utilisée pour cette description algorithmique est une image de même taille que l'originale qui contient des 0 et des 1 distribués de façon aléatoire.

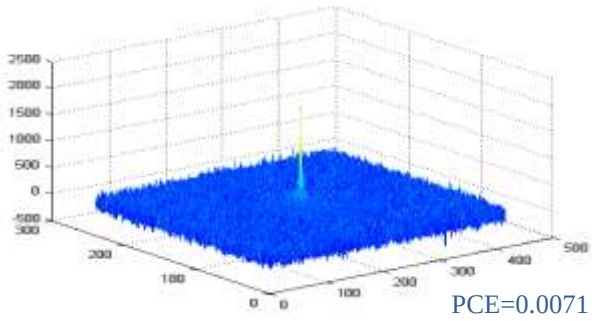
Algorithme 4.1 : Algorithme du cryptage à flot synchrone avec une clé de cryptage biométrique

```
i ← 1
TANT QUE image est différent de la fin d'image FAIRE
  c ← prochain bit d'image          //Lecture des pixels binaires d'images
      ECRIRE (c XOR clé [i] )       //Application d'XOR entre bit d'image et bit de clé
      i ← i + 1
FIN TANT QUE
Fin
```

Au niveau du développement de notre méthode, nous avons utilisé une clé de longueur fixe et nous avons réutilisé la clé autant de fois qu'il est nécessaire pour couvrir la longueur totale de l'information à crypter. Ceci est lié forcément à la résolution vidéo utilisée.

La figure 4.8 (a) représente l'image reconstruite après le cryptage à flot synchrone et la compression. La corrélation entre la figure 4.8 (a) et la figure 4.5 (a) est illustrée dans la figure 4.8 (b). Nous trouvons cette fois un PSNR = 11.828 dB, un PCE = 0.0071 et un RC = 23.152 %.

(a)



(b)

Figure 4. 8- Image décodée après compression et cryptage des données brutes

La valeur du PCE est bien faible par rapport au PCE trouvé dans le cas de compression sans cryptage (PCE=0.0845). Ceci s'explique par le fait que la méthode de corrélation est sensible aux contours qui sont restés visible après le décodage. De plus, le rapport de compression est faible comparé à la valeur de 83.278 % obtenue sans cryptage, pour le même pas de quantification. Ceci s'explique par le fait que la clé aléatoire a été appliquée sur les données brutes. Les redondances spatiales entre pixels voisins et blocs de pixels voisins ont été supprimées. Par conséquent, le nombre de coefficient quantifiés à zéro a été réduit et le codage arithmétique n'est plus efficace. On voit donc tous les inconvénients de faire un cryptage suivi par une compression.

3.3. Compression et cryptage des coefficients transformés et quantifiés (C&C CQ)

Pour pallier les défauts de faible cryptage cités en 3.2, nous avons effectué le cryptage à flot synchrone à base de clé biométrique après la transformation et quantification des données vidéo comme montré dans la figure 4.9. L'algorithme de cryptage utilisé est équivalent à celui de l'algorithme 4.1, sauf que le ou-exclusif s'applique sur les données transformés et quantifiés.

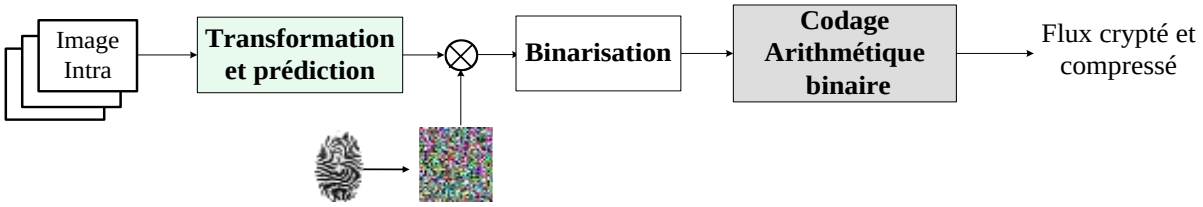


Figure 4. 9- Cryptage à base de clé biométrique appliqué sur les coefficients transformées et quantifiés

Lorsque nous adoptons cette optimisation, nous utilisons la même clé de cryptage qu'auparavant et nous obtenons un PSNR = 11.764 dB, un PCE = 0.0073 et un RC = 1.381 %. Les résultats sont présentés dans la figure 4.10.

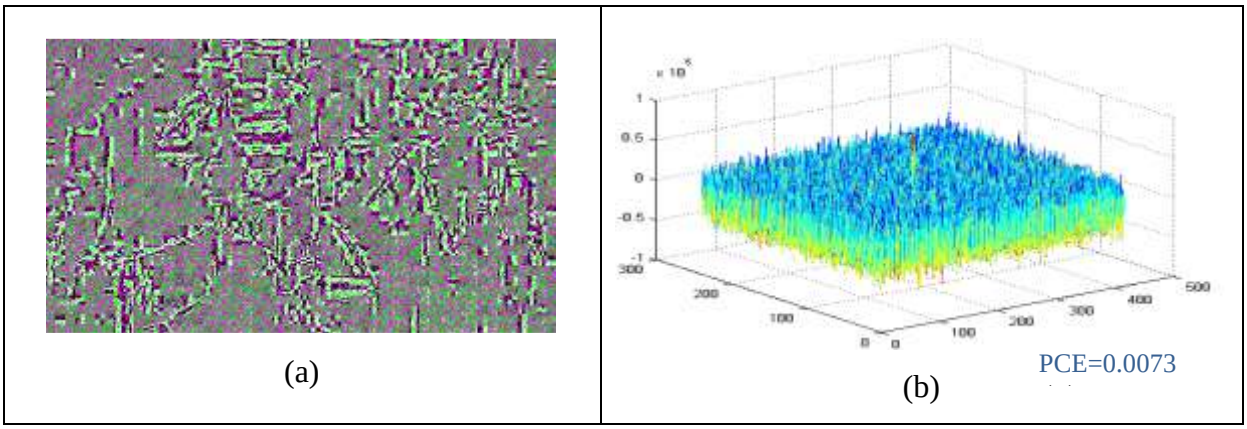


Figure 4. 10- Image décodée après compression et cryptage des coefficients transformées et quantifiés

Au niveau de la qualité de l'image reconstruite 4.10 (a), le PSNR est proche de ce que nous avons trouvé avec (C&C DB) (11.61≈11.58) alors que l'image reconstruite a un niveau de cryptage visuel plus intéressant que celui de (C&C DB). C'est pour cette raison qu'il intéressant d'utiliser un autre paramètre de mesure objective de la qualité des images. En effet, le PCE indique une valeur plus faible que celle obtenue dans la section 3.2. Enfin, le rapport de compression s'est complètement dégradé par rapport à la configuration sans cryptage (83.278

% 1.50). Ceci s'explique par le fait que l'utilisation de la clé aléatoire a réduit le nombre de zéros consécutifs dans les hautes fréquences du spectre et par conséquent, le BAC n'est plus efficace.

Pour remédier à ce problème, nous allons présenter dans la section suivante une approche de compression et cryptage simultanés permettant d'améliorer le rapport de compression.

3.4. Compression et cryptage conjoints (C&C C)

3.4.1. Algorithme RAC

Le codage arithmétique utilisé dans notre chaîne est une technique de compression sans perte qui représente les données à coder comme une valeur fractionnaire dans le segment scalaire [0, 1[[Langdon 84], [Say 05]. Pour une description détaillée de l'algorithme, les lecteurs intéressés peuvent se référer au chapitre 3 de la thèse. De nombreuses d'approches dédiées à coder et reconstruire des données cryptées avec les codeurs des images fixes et animées sont décrites dans la littérature, [Pande 11], [Grangetto 06], [Varalakshmi 10], ect. Nous proposons dans cette partie une technique de cryptage simultané à la compression dont le but était de changer le flux binaire généré sans modifier les performances du codeur notamment en terme de rapport de compression. Notre méthode est basée sur l'algorithme du RAC (Randomized Arithmetic Coding) et décrite dans la figure 4.11. Le principe de fonctionnement de ce cryptage RAC se repose en grande partie sur celui du codeur arithmétique binaire (BAC). En effet, au début du processus, il y a acquisition des données à compresser et à crypter simultanément avec la clé de cryptage noté R dans la figure 4.11. Les statistiques sur les entrées binaires sont réalisées pour déterminer la taille des données à compresser ainsi que le nombre de 1 et de 0 dans la chaîne à compresser. Avec ceci, une initialisation des bandes inférieure 'low' et supérieure 'sup' est réalisée en plus de l'estimation de la taille théorique des données compressées. Ensuite, pour tout élément de la chaîne binaire à compresser, il y a une mise à jour de ces bandes. Tant qu'il y a des données à compresser (la fin du fichier n'est pas atteinte), l'algorithme utilise les bandes 'low' et 'sup' pour se situer dans l'une des trois conditions E1, E2 ou E3. Si la condition E3 est satisfaite, il y a mise à jour des bandes, complémentation du MSB des bandes et incrémentation du compteur 'E3_compt'. Sinon, la condition E1 ou E2 est satisfaite. C'est précisément à cet endroit que la partie cryptage peut commencer.

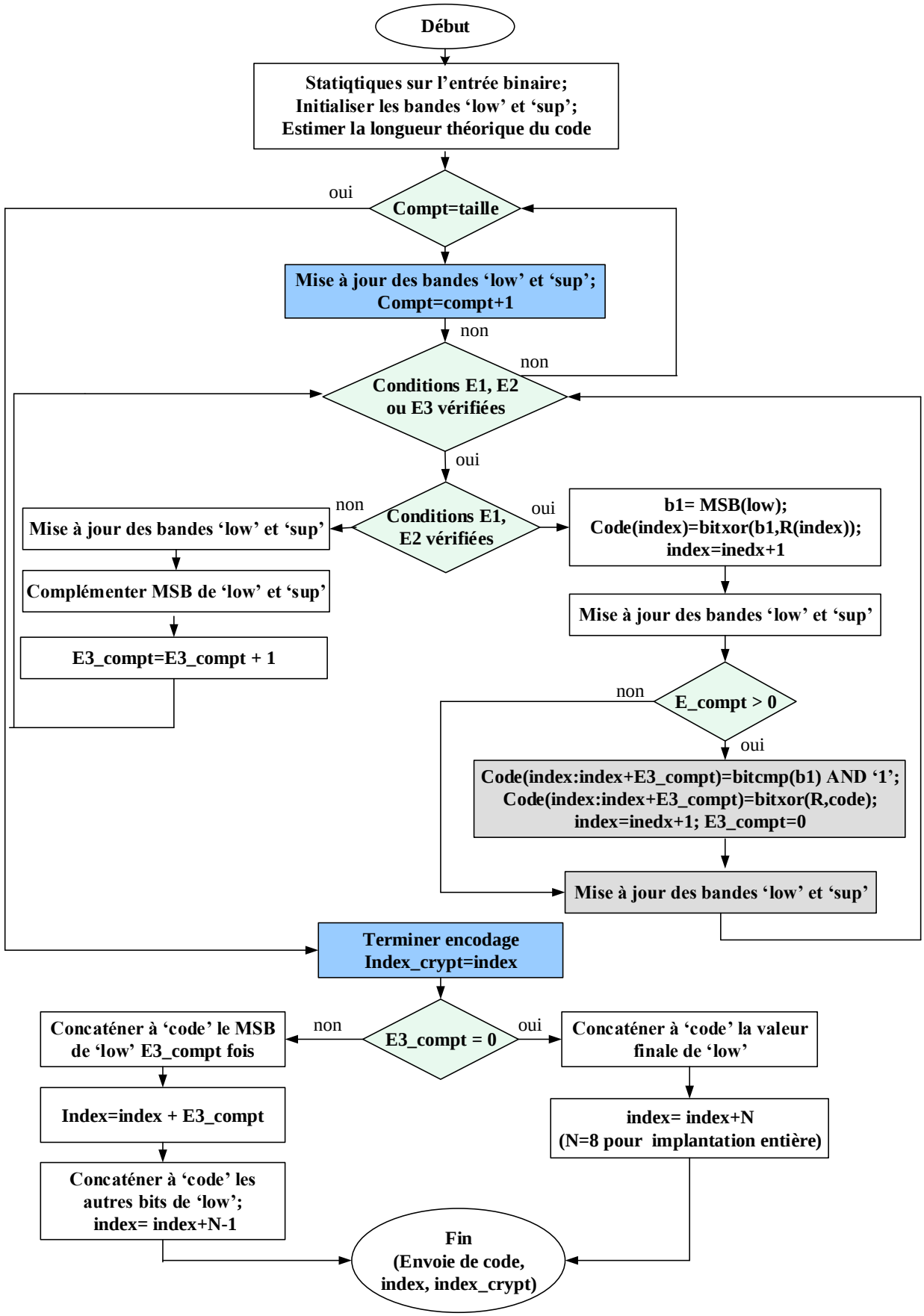


Figure 4. 11- Principe de la technique proposée de cryptage simultané à la compression

Le MSB (bit b1) de la bande 'low' est récupéré, ce qui permet de trouver le code pour cet indice, qui est la combinaison avec un ou exclusif entre b1 et la valeur du vecteur aléatoire R prise à cet indice. Nous avons fait le choix de faire le cryptage à ce niveau pour augmenter la robustesse du cryptage. Il est important de signaler que le nombre de fois où la condition E3 est satisfaite est donné par le compteur E3_compt. Tant que la valeur de ce compteur est non nulle, nous réalisons un ou exclusif entre la valeur du complément de b1 prise E3_compt fois (conformément à l'algorithme BAC) et les bits correspondants dans le vecteur aléatoire R.

Ces opérations continuent jusqu'à la fin des données à compresser. C'est à ce moment que nous passons au processus de fin d'encodage. Ici, nous relevons l'indice des données compressées à ce moment précis ce qui correspond à la taille intermédiaire du code. La variable «index_crypt» est une donnée utile du processus du cryptage puisqu'elle indique la taille du vecteur aléatoire R qui a été utilisé lors du processus de crypto-compression. Cette taille sera très utile pour le décryptage des informations. Au final, la variable 'index' indiquera la taille des données compressées et la variable «index_crypt» indique la taille du vecteur aléatoire utilisé pour le cryptage. A la fin du processus, nous pouvons trouver la relation suivante entre ces deux compteurs :

$$index = \begin{cases} index_{crypt} + N\,, & si\ E3_{compt} = 0 \\ index_{crypt} + E3_{compt} + N - 1\,, & sinon \end{cases} \quad (4.4)$$

Rappelons que N=8 pour une implantation numérique. Une fois l'encodage est fini, l'algorithme renvoie le code compressé et crypté et les variables index et index_crypt qui représentent la taille du code et de la clé utilisé pour le cryptage, respectivement.

3.4.2. Structures LFSR pour la génération de nombres pseudo-aléatoires

Concernant la génération de nombres pseudo-aléatoires nous avons opté pour des structures appelés LFSR (Linear Feedback Shift Register). Ce choix est justifié par la très faible complexité de d'implémentation matérielle des LFSR et par l'excellente qualité statistique des nombres produits. Plus précisément, un LFSR binaire de longueur L est composé d'un registre à L bascules. Ce registre est contrôlé par une horloge externe qui définit les périodes de fonctionnement du LFSR. Au cours de chaque période de temps, chaque bascule décale son contenu vers la droite. Le retard engendré par les temps de propagation des bascules est utilisé pour construire un nombre aléatoire. La figure 4.12 représente le schéma bloc du codeur entropique intégrant le cryptage.

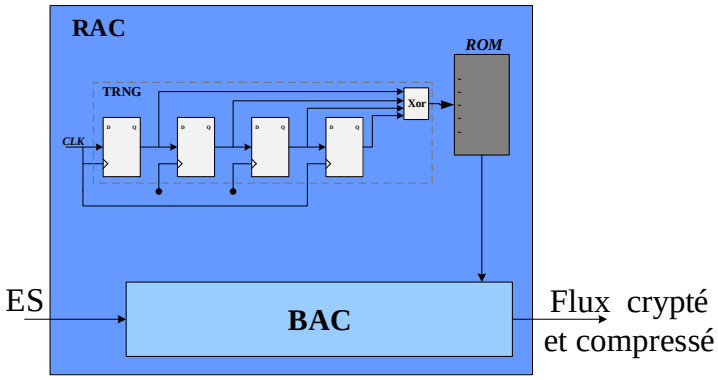


Figure 4. 12- Schéma bloc du RAC

Les valeurs d'initialisation des bascules jouent un rôle important pour la propriété statistique des nombre générés. En effet, ces valeurs d'initialisation définissent le polynôme de rétroaction d'un LFSR et doivent restés secrètes lors d'un chiffrement par flot. Même si l'algorithme de Berlekamp-Massy permet de le reconstruire à partir de 2L bits de la suite chiffrant [Berlekamp 68], [Massey 69]. C'est pour cette raison qu'ils sont souvent utilisés comme module de base dans les générateurs pseudo-aléatoires dédiés, mais au sein d'un dispositif plus complexe.

Par exemple, pour augmenter la non linéarité du code, les LFSR peuvent être utilisés en parallèle. C'est le cas de l'algorithme de Geffe où la mise en cascade de plusieurs LFSR de longueurs différentes suivies d'une combinaison logique de chaque LFSR permet d'avoir une meilleure robustesse des codes générés [Geffe 73].

Au cours de ces travaux de thèse, nous nous sommes intéressés aux implantations matérielles de ces structures LFSR. L'ensemble de ces travaux a fait l'objet d'une publication dans une conférence internationale [Neji 13].

3.4.3. Résultats de simulation

Nous avons réalisés des simulations avec l'outil Matlab de la méthode de crypto-compression utilisé. Le schéma bloc de la figure 4.13 montre les différents blocs qui constituent la méthode proposée. L'algorithme RAC est celui expliqué dans la section 3.3.1 et le nombre aléatoire est généré par la fonction rand de Matlab.

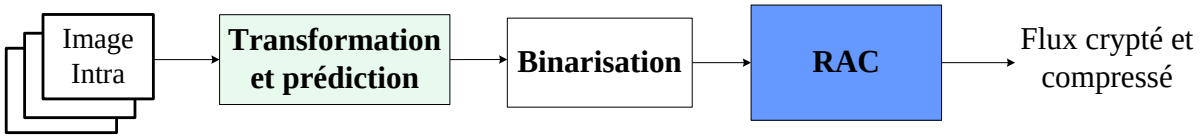
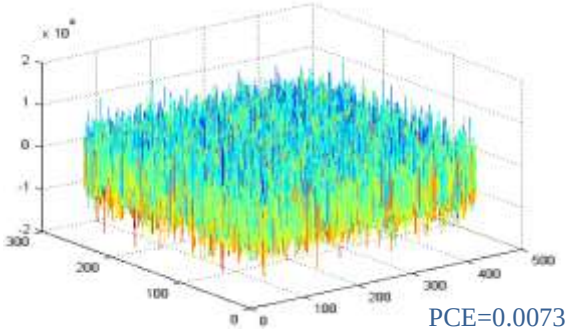


Figure 4. 13- Cryptage simultané au codage entropique RAC

L'image cryptée de cette méthode proposée est présentée dans la figure 4.14. Nous avons obtenu un PSNR = 12.081 dB, un PCE = 0.0858 et un RC = 83.278%. Ces PSNR et PCE sont très proches de ceux obtenus dans C&C CQ (section (3.3)) avec un rapport de compression beaucoup plus important (+88%). Ce rapport de compression est identique au rapport obtenu dans le cas sans cryptage. C'est la valeur maximale que nous pouvons atteindre. C'est tout l'intérêt de l'algorithme RAC proposé qui assure le cryptage sans changer les statistiques de la séquence à compresser. Quand les statistiques de la séquence (nombre des 1 et des 0) ne changent pas, l'algorithme RAC possède des performances identiques à celles de l'algorithme BAC en termes de compression.

(a)



(b)

Figure 4. 14- Image décodée après compression et cryptage RAC

3.4.4. Processus de décryptage

Non seulement la méthode proposée permet de compresser les données avec des performances identiques au cas sans cryptage, mais elle permet aussi d'assurer le décryptage des informations de façon parfaite. Quand les données cryptées et compressées sont envoyées, on envoie aussi (en clair ou en crypté) la valeur de la variable index_crypt.

Etant donné que les utilisateurs autorisés à voir le contenue de l'image se partagent une clé de longueur suffisamment importante, ils n'ont qu'à adapter la longueur de la clé par rapport à la valeur de index_crypt en prenant que les premiers éléments de la clé. Pour ces premiers éléments qui correspondent à index_crypt, on applique un ou exclusif bit-à-bit entre le code et la clé R et pour les autres éléments, aucune action n'est envisagée.

Enfin, on applique l'algorithme de décodage arithmétique BAD tel qu'il a été détaillé dans le chapitre 3. La figure 4.15 donne une idée synthétique de cet algorithme de décryptage.

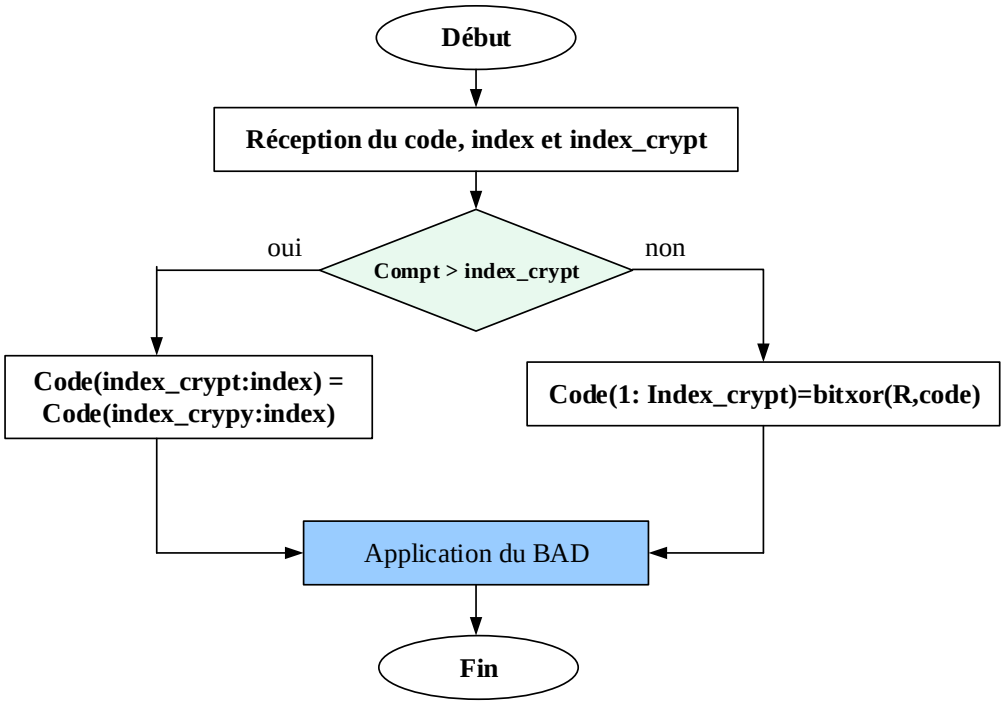


Figure 4. 15- L'algorithme de décryptage proposé

Une fois le décryptage terminé, on applique l'ordre zigzag inverse, la déquantification et l'IDCT pour trouver l'image décryptée. Pour une image d'entrée de la figure 4.16.a, l'image compressée et non cryptée est mentionnée dans la figure 4.16.b.

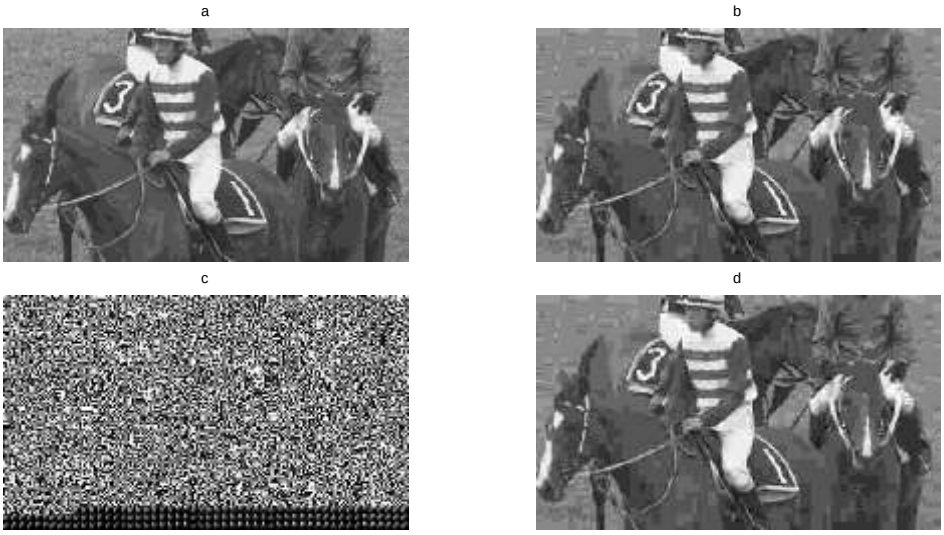


Figure 4. 16- Résultats de décryptage. (a) image d'origine, (b) image compressée et non crypté, (c) image compressée et cryptée avec une mauvaise clé, (d) image compressée et cryptée avec la bonne clé.

Un décryptage qui reconstruit exactement à l'origine l'image décryptée est censé converger vers cette image. C'est ce qui est illustrée dans la figure 4.16.d où le décryptage donne la

même image compressée. A contrario, une image reconstruite sans connaitre la clé de cryptage est illustrée dans la figure 4.16.c.

3.5. Compression et cryptage double (C&C D)

Le cryptage peut agir sur un seul niveau (avant la DCT, après la DCT ou dans le BAC) comme expliqué dans les sections précédentes, mais peut aussi agir sur plusieurs niveaux simultanément. En fait, nous voulons voir si la concaténation de deux méthodes de cryptage dans divers blocs de la chaîne de compression supplante les insuffisances auparavant cités. Ici, nous introduisons deux niveaux de cryptage. Un premier niveau utilisant la clé biométrique dans l'un des étages comme décrit dans les parties 3.2 et 3.3 (avant la DCT (figure 4.17) (C&C D1) ou après la phase de quantification (4.18) (C&C D2)).

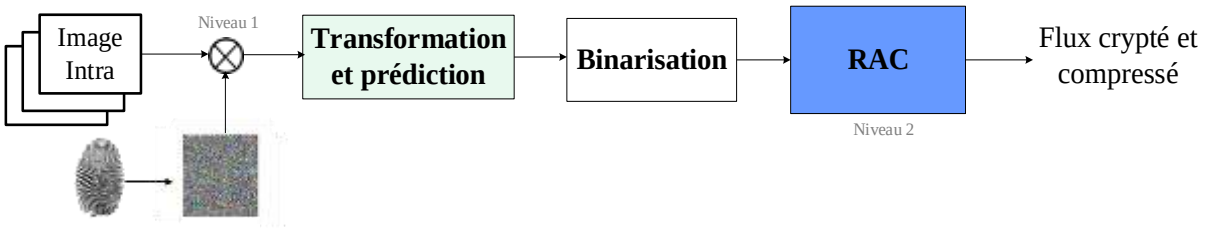


Figure 4. 17- Cryptage double avec une clé biométrique appliquée sur les données brutes

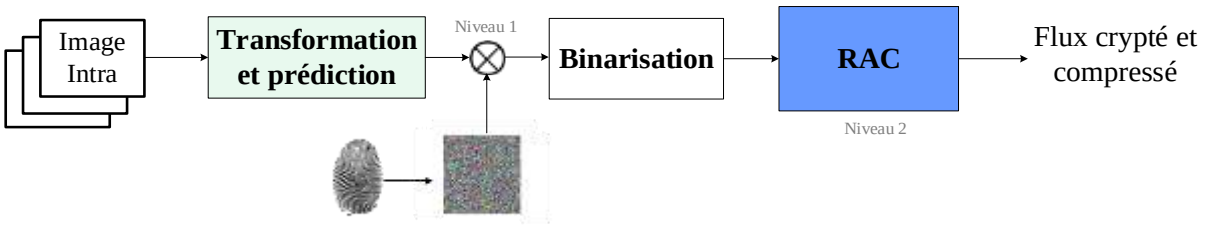
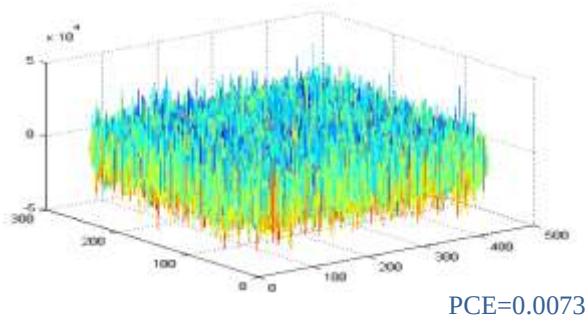


Figure 4. 18- Cryptage double avec une clé biométrique appliquée sur les coefficients transformés et quantifiés

Le deuxième niveau de sécurité se situe au niveau du codeur entropique BAC dont l'algorithme de compression est modifié comme décrit dans la partie (C&C C). La chaîne de crypto-compression procède alors simultanément à la compression et au cryptage des données dans deux niveaux. En effet, un cryptage dans l'encodeur vidéo et un autre cryptage dans le module du codeur entropique sont effectués à la fois.

Les figures 4.19 et 4.20 représentent les images décodées après compression et cryptage double avec une clé biométrique appliquée sur les données brutes et sur les coefficients transformées et quantifiées respectivement.

(a)



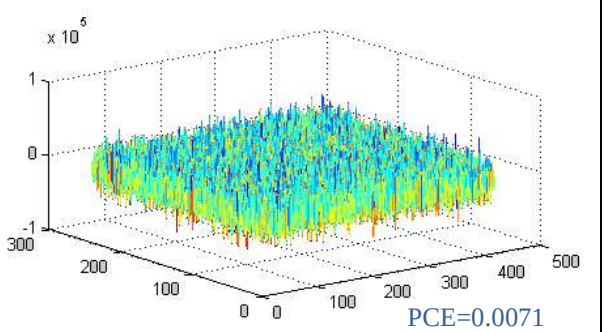
(b)

Figure 4. 19- Image décodée après compression et cryptage double
avec une clé biométrique appliquée sur les données brutes

Lorsque le cryptage avec une clé biométrique est appliqué sur les données brutes, nous avons obtenu PSNR = 11.385 dB, PCE = 0.0073 et RC = 23.152 % (figure 4.17). Et si le cryptage avec une clé biométrique est appliqué sur les coefficients transformées et quantifiées, nous avons obtenu PSNR = 10.358 dB, PCE = 0.0071 et RC = 1.381 % (figure 4.18).

Ces deux techniques nous donnent alors deux PCE légèrement différents, une différence de 1 dB au niveau PSNR et un rapport de compression moins bien que celui trouvé avec le RAC.

(a)



(b)

Figure 4. 20- Image décodée après compression et cryptage double
avec une clé biométrique appliquée sur les coefficients transformés et quantifiés

3.6. Résultats des tests et comparaison des approches proposées

Plusieurs simulations ont été tournées avec des séquences de tests diverses et variées. Le but de changer les séquences de test est de voir le comportement de la méthode proposée par rapport à la quantification, la résolution par exemple ou à la nature même de séquence à tester. Les résultats pour la séquence Racehorses sont récapitulés dans le tableau 4.1. La compression et cryptage conjoints (C&C C) nous donne le meilleur rapport de compression. Les mesures des PSNR et des PCE donnent des résultats très similaires avec plusieurs

méthodes. Nous ne pouvons pas conclure à ce niveau par rapport à la qualité du cryptage, mais nous pouvons d'ores et déjà affirmer que le cryptage RAC donne les meilleurs résultats.

Tableau 4. 1- Récapitulatifs des résultats de test pour la séquence SD Racehorses

	PSNR	PCE	RC %
Compression sans cryptage (CSC)	21.4804	0.0845	83.278
Compression et cryptage des données brutes (C&C DB)	11.828	0,0071	23.152
Compression et cryptage des coefficients transformés et quantifiés (C&C CQ)	11.764	0,0073	1.381
Compression et cryptage conjoints (C&C C)	12.081	0,0073	83.278
Compression et cryptage double avec une clé appliquée sur les données brutes (C&C D1)	11.385	0,0073	23.152
Compression et cryptage double avec une clé appliquée sur les DCT coefficients (C&C D2)	11.358	0,0071	1.381

Pour évaluer et comparer les différentes techniques de compression et cryptage décrites dans les sous sections 3.1 à 3.5, divers formats principaux ont été testés : QCIF, SD, 720p et 1080p dans une version

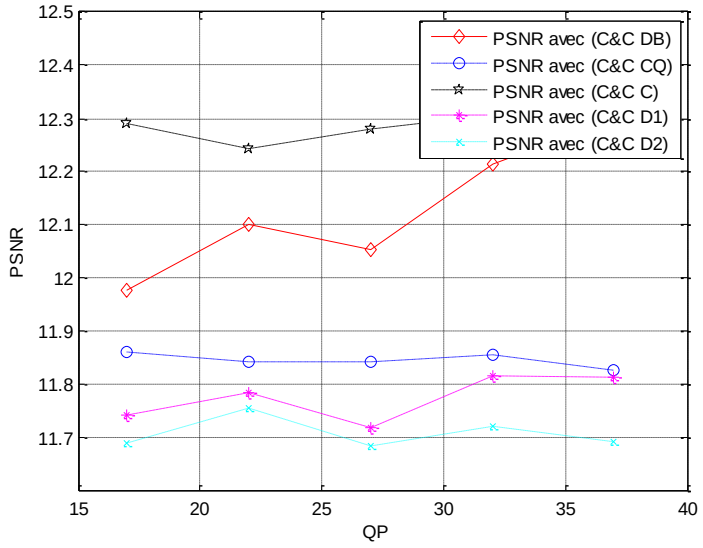


Figure 4. 21- La moyenne de variation du PSNR en fonction de QP pour les méthodes de compression et cryptage proposées appliquées sur cinq séquences QCIF

YUV 4:4:4 à 10 bits et dans une configuration ou toutes les images sont codées en Intra. Nous avons varié les pas de quantification QP entre des valeurs faibles, moyennes et importantes. Les séquences de test sont des séquences QCIF (Forman, Coastguard, Carphone, Suzie et Mobile). La moyenne des variations de PSNR en fonction des QP sont présentées sur la figure

4.21. La variation de PSNR en fonction des QP pour chaque séquence vidéo est présentée dans l'annexe (figure A.1).

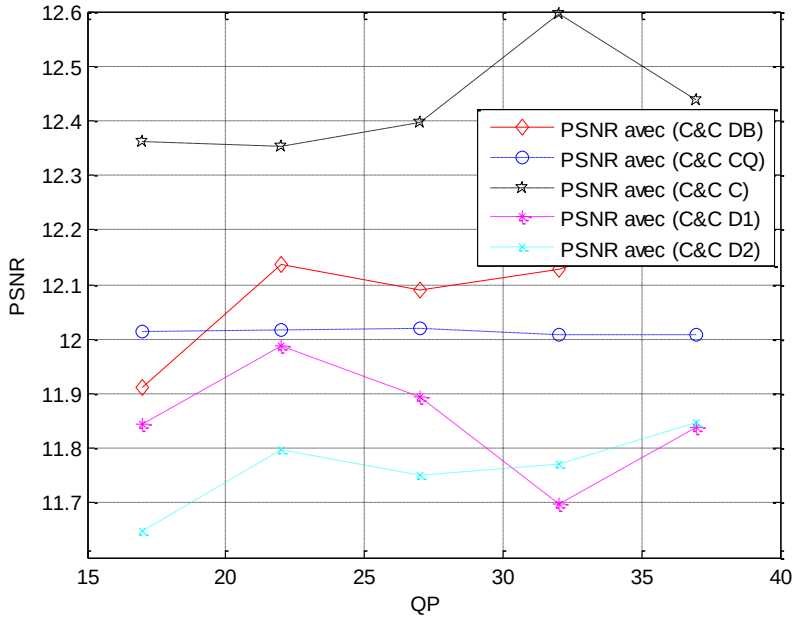


Figure 4. 22- La moyenne de variation du PSNR en fonction de QP pour les méthodes de compression et cryptage proposées appliquées sur quatre séquences SD

De la même façon, pour des séquences SD (BasketballPass, BlowingBubbles, BQSquare et RaceHorses), nous présentons la moyenne des variations de PSNR en fonction des QP dans la figure 4.22 et la variation de PSNR de chaque séquence vidéo dans l'annexe (figure A.2).

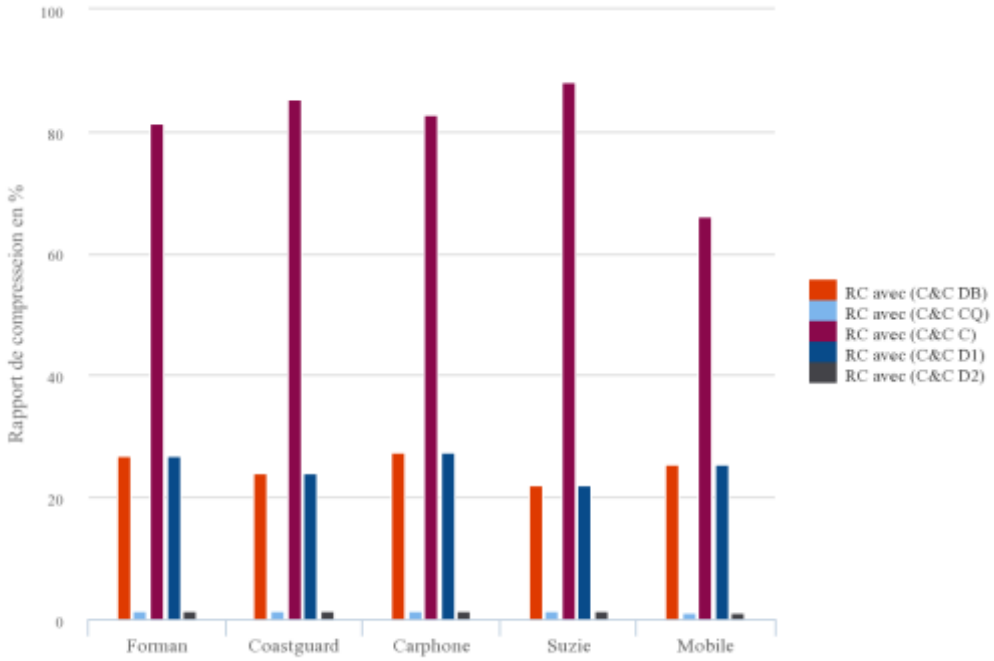


Figure 4. 23- Rapport de compression pour des séquences vidéo QCIF pour QP = 32

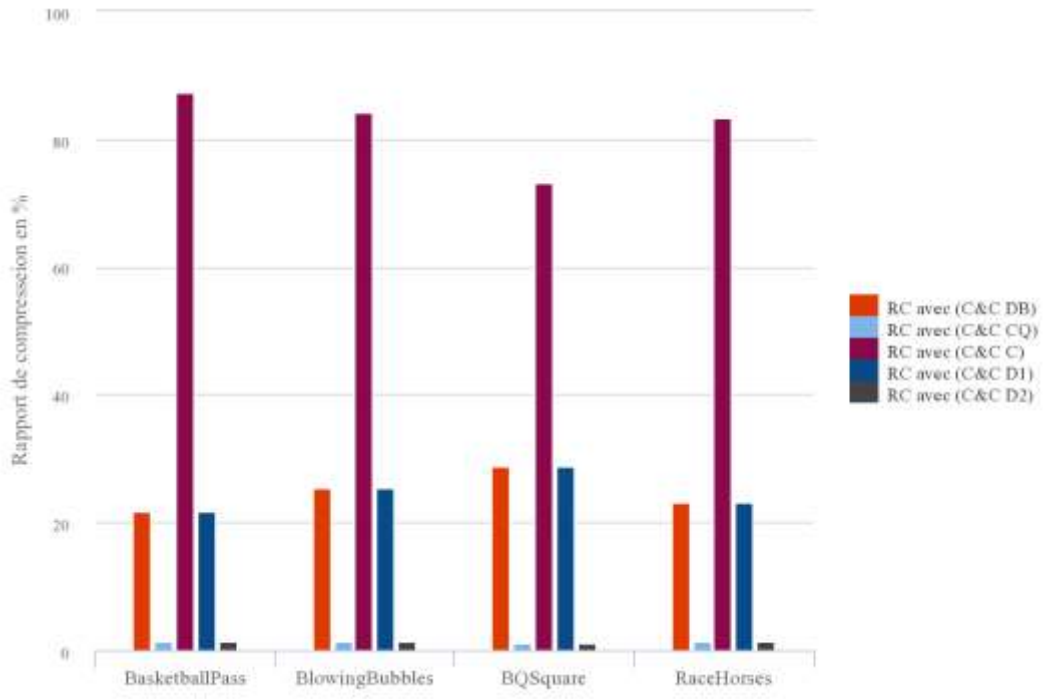


Figure 4. 24- Rapport de compression pour des séquences vidéo SD pour QP = 32

Pour les deux résolutions, nous obtenons le même ordre de croissance des PSNR pour les différentes méthodes de compression et cryptage. Toutefois, la compression et cryptage conjoints (C&C C) nous donne le PSNR le plus élevé, ensuite nous obtenons dans l'ordre décroissant les autres méthodes (C&C DB) (C&C CQ) (C&C D1) (C&C D2).

Les rapports de compression pour les même séquences QCIF et SD citées avant sont présentés sous forme histogramme dans les figures 4.23 et 4.24 pour un QP = 32. La compression et cryptage conjoints (C&C C) nous attribut le meilleur RC, nous obtenons des RC respectivement décroissants par (C&C DB), (C&C D1), (C&C CQ) et (C&C D2).

Tableau 4. 2- PCE pour des séquences QCIF pour différents valeurs de QP

		37	32	27	22
Forman	(CSC)	0,6626	0,8479	0,6626	0,6626
	(C&C DB)	0,5065	0,7189	0,5065	0,5065
	(C&C CQ)	0,5739	0,5807	0,5739	0,5739
	(C&C C)	0,5739	0,5807	0,5739	0,5065
	(C&C D1)	0,4390	0,7189	0,5065	0,5739
	(C&C D2)	0,5739	0,7189	0,5065	0,6414
Coastguard	(CSC)	0,3325	0,3370	0,3370	0,3325
	(C&C DB)	0,1174	0,1099	0,1097	0,1174
	(C&C CQ)	0,1242	0,1099	0,1097	0,1242
	(C&C C)	0,1174	0,1099	0,1097	0,1242

	(C&C D1)	0,1310	0,1099	0,1136	0,1310
	(C&C D2)	0,1242	0,1099	0,1136	0,1174
Carphone	(CSC)	0,1917	0,1294	0,1294	0,1917
	(C&C DB)	0,0367	0,0167	0,0167	0,0367
	(C&C CQ)	0,0423	0,0235	0,0235	0,0423
	(C&C C)	0,0395	0,0167	0,0235	0,0451
	(C&C D1)	0,0423	0,0235	0,0167	0,0395
	(C&C D2)	0,0451	0,0167	0,0167	0,0423
Suzie	(CSC)	1,0321	0,9207	1,0321	1,0321
	(C&C DB)	1,0052	0,8199	1,0052	0,9104
	(C&C CQ)	0,7557	0,7087	0,7557	0,7557
	(C&C C)	1,0052	0,6809	1,0052	0,9104
	(C&C D1)	0,9104	0,7087	0,6009	0,7557
	(C&C D2)	0,4461	0,6253	0,7557	1,0652
Mobile	(CSC)	0,0404	0,0400	0,0404	0,0404
	(C&C DB)	0,0017	0,0016	0,0017	0,0016
	(C&C CQ)	0,0018	0,0013	0,0018	0,0018
	(C&C C)	0,0017	0,0010	0,0017	0,0017
	(C&C D1)	0,0019	0,0016	0,0019	0,0018
	(C&C D2)	0,0020	0,0016	0,0018	0,0017

Les mesures des PCE pour les séquences QCIF et SD sont illustrées dans les tableaux 4.2 et 4.3. Nous remarquons que la variation de QP n'influe pas beaucoup sur les valeurs de cette métrique. Nous obtenons des PCEs très proches avec (C&C DB), (C&C C), (C&C CQ), (C&C D1), et (C&C D2) et bien inférieurs aux PCEs de la compression sans cryptage (CSC).

Des résultats similaires ont été obtenus avec d'autre résolution (SD) comme indiqué dans le tableau 4.3.

Tableau 4. 3- PCE pour des séquences SD pour différents valeurs de QP

		37	32	27	22
BasketballPass	(CSC)	0.4340	0.4340	0.4340	0.4340
	(C&C DB)	0.4340	0.4340	0.4340	0.4340
	(C&C CQ)	0.3966	0.3966	0.3966	0.3966
	(C&C C)	0.3966	0.3966	0.3966	0.4340
	(C&C D1)	0.3966	0.3966	0.3966	0.4340
	(C&C D2)	0.4340	0.3966	0.3592	0.4340
BlowingBubbles	(CSC)	0.0675	0.0675	0.0675	0.0675
	(C&C DB)	0.0675	0.0675	0.0675	0.0675
	(C&C CQ)	0.0660	0.0660	0.0660	0.0660
	(C&C C)	0.0660	0.0675	0.0646	0.0675
	(C&C D1)	0.0675	0.0660	0.0675	0.0660

	(C&C D2)	0.0646	0.0646	0.0646	0.0632
BQSquare	(CSC)	0.0186	0.0186	0.0186	0.0186
	(C&C DB)	0.0186	0.0186	0.0186	0.0186
	(C&C CQ)	0.0337	0.0337	0.0337	0.0337
	(C&C C)	0.0337	0.0337	0.0337	0.0487
	(C&C D1)	0.0186	0.0487	0.0337	0.0186
	(C&C D2)	0.0487	0.0487	0.0487	0.0337
RaceHorses	(CSC)	0.0845	0.0845	0.0845	0.0845
	(C&C DB)	0.0845	0.0845	0.0845	0.0845
	(C&C CQ)	0.0858	0.0858	0.0858	0.0858
	(C&C C)	0.0871	0.0858	0.0871	0.0871
	(C&C D1)	0.0858	0.0858	0.0871	0.0845
	(C&C D2)	0.0845	0.0845	0.0858	0.0871

Le temps d'exécution de la compression et cryptage conjoints illustré dans le tableau 4.4 est très inférieur par rapport à la compression double. Non seulement le deuxième niveau de cryptage n'est pas efficace (ne renforce pas le niveau de cryptage et détruit le rapport de compression). Mais aussi, les performances en terme de temps d'exécution sont très faibles.

Tableau 4. 4- Rapport de temps pour des séquences QCIF

Vidéos	Rapport de temps (compression et cryptage / compression simple) (%)		
	(C&C C)	(C&C D1)	(C&C D2)
Foreman	0.1	54.59	6.84
Coastguard	0.803	51.052	3.791
Carphone	3.39	52.209	9.013
Suzie	0.351	58.175	7.827
Mobile	0.898	48.447	6.867

4. Transfert sécurisé d'images par combinaison de techniques de cryptage et compression

Tout cryptosystème doit nécessairement être résistant contre les attaques. Ceci dépend forcément de son algorithme de chiffrement qui ne doit jamais être cassé, autrement la reconstruction du message chiffré en clair doit être difficile ou impossible. La cryptanalyse des algorithmes de chiffrement est essentielle pour en trouver les failles.

Pour l'effectuer, nous nous servons des simulations ou attaques. Pour étudier le niveau de ses sécurités, nous essayons de casser les fonctions cryptographiques qui composent les algorithmes. Nous introduisons dans cette section les différentes notions de sécurité et d'attaques possibles et nous analysons ensuite la robustesse du système de crypto-compression proposé.

4.1. Notion de sécurité

La qualité la plus importante que doit posséder un cryptosystème est la sécurité. Il doit résister aux attaques qui auraient pour but d'extraire le texte original. La sécurité est liée à la capacité de deviner les valeurs des données chiffrées. Par exemple, de point de vue confidentialité, il est préférable de chiffrer les bits qui semblent les plus aléatoires. Cependant, en pratique, une attaque est plus difficile sur les coefficients AC non nuls d'une image JPEG que sur ses coefficients DC qui sont fortement prévisibles [Van 02]. Néanmoins, le grand souci dans la conception d'un système sûr en cryptographie est la durée nécessaire pour le casser, qui doit être supérieure à sa durée de validité. La tendance actuelle est de chercher à prouver la sécurité d'un système sur la base d'hypothèses, sur la puissance de calcul requise ou sur la quantité de texte. Si la cryptanalyse est effectuée sur la base d'hypothèses, on considère comme exemplaire l'hypothèse fondamentale, connue sous le nom de principe de Kerckhoff, qui indique que l'adversaire connaît complètement l'algorithme de cryptage, à l'exception de la clé secrète qui est inconnue. Donc, la sécurité du cryptosystème repose entièrement sur la clé secrète. Cette hypothèse signifie que la sécurité d'un schéma de cryptage ne doit pas reposer sur la confidentialité du schéma, c'est-à-dire la fonction de chiffrement employée, mais sur la confidentialité de la clé.

Pour juger la réussite d'une attaque, beaucoup d'éléments doivent être tenu en compte comme la complexité, le temps de calcul, la quantité et la qualité des informations déduites de l'attaque (la clé secrète, l'algorithme de chiffrement découvert sans,...). La complexité de l'attaque se caractérise par le temps en nombre d'opérations effectuées, par la mémoire nécessaire et par la quantité de données requises.

4.2. Les attaques cryptographiques

La cryptanalyse a bien aidée à renforcer la sécurité des systèmes, qui reste toujours relative ou imparfaite. En effet, leur résistance parait parfois faible devant certaines attaques. Nous pouvons identifier de nombreuses attaques contre les cryptosystèmes. Deux classes d'attaques sont distinguées: les attaques actives qui font ralentir, dégrader ou empêcher la communication, d'envoyer des informations parasites (nous agissons sur l'information afin de tromper le destinataire ou de faire modifier ou disparaître ces informations; par exemple, nous modifions le message (suppression, ajout,…), nous retardons sa transmission, nous répétons son envoi) et les attaques passives qui interceptent un message et exploitent tous ses informations (nous touchons la confidentialité des données sans modifier les informations

interceptées, ie, nous récupère le message, la clé secrète, etc.). Une attaque est souvent caractérisée par les données qu'elle nécessite supposées connues par les attaquants. Nous distinguons ces méthodes [Akkar 04] :

- L'attaque à texte chiffré seulement : L'attaquant connait le texte chiffré de plusieurs messages ; il fait des hypothèses sur le texte et il tente de retrouver la clé secrète ou le texte clair en observant seulement le texte chiffré.
- L'attaque à texte en clair connu : L'attaquant dispose des messages ou parties de message clairs et de leur version chiffrée. La tâche est de retrouver la clé utilisée pour chiffrer ces messages ou un algorithme qui permet de décrypter n'importe quel nouveau message chiffré avec la même clé.
- L'attaque à texte en clair choisi : L'attaquant connait le texte chiffré de plusieurs messages et le texte en clair correspondant et il peut choisir les textes en clair à chiffrer. La tâche consiste à retrouver la clé utilisée pour chiffrer ces messages ou un algorithme qui permette de décrypter n'importe quel nouveau message chiffré avec la même clé. Les chiffrements asymétriques sont notamment vulnérables à cette attaque.
- L'attaque adaptative à texte en clair choisi (cas particulier de l'attaque à texte en clair choisi) : Non seulement l'attaquant peut choisir les textes en clair mais il peut également adapter ses choix en fonction des textes chiffrés précédents. Dans une attaque à texte en clair choisi, l'adversaire est juste autorisé à choisir un grand bloc de texte en clair au départ tandis que dans une attaque à texte en clair adaptative, il choisit un bloc initial plus petit et ensuite il peut choisir un autre bloc en fonction du résultat pour le premier et ainsi de suite.
- L'attaque à texte chiffré choisi : L'attaquant peut choisir différents textes chiffrés à décrypter. Les textes décryptés lui sont alors fournis.
- L'attaque adaptative à texte chiffré choisi : C'est une attaque à texte chiffré choisi où le choix du texte chiffré peut dépendre du texte en clair reçu précédemment.

Il existe nombreux types d'attaques cryptographiques employés qui se diffèrent selon le type du système. Nous présentons quelques types d'attaques. Nous citons au début l'attaque en force brute. L'attaquant dans ce cas essaie toutes les combinaisons de clefs possibles jusqu'à l'obtention du texte clair. Avec des ordinateurs de plus en plus performants et des méthodes de calculs distribués, l'attaque en force restera toujours un moyen de casser des systèmes de chiffrement, mais c'est la plus coûteuse en temps de calcul et en mémoire à cause de la recherche exhaustive. Nous trouvons aussi l'attaque à l'aide de l'analyse statistique.

L'attaquant ici possède des informations sur les statistiques du message clair comme les fréquences des lettres. En outre, nous distinguons l'attaque d'une tierce personne. Cette attaque intervient dans une transaction entre deux personnes. Une troisième personne s'interpose de manière transparente entre les deux et termine la transaction normalement en captant les messages et en transmettant d'autres messages. Il peut donc ainsi intercepter et même modifier les messages envoyés sans que les deux entités s'en aperçoivent.

Nous spécifions l'attaque à l'aide du temps d'opération : Cette méthode est basée sur la mesure du temps nécessaire pour effectuer des chiffrements ou des déchiffrements. Ainsi cette étude permet de mieux cibler la longueur de la clé utilisée et a donc pour but de limiter le domaine des clés à explorer pour une cryptanalyse classique. L'analyse fréquentielle examine les répétitions des lettres du message chiffré afin de trouver la clé. Elle est inefficace contre les chiffrements modernes tels que DES, RSA. Elle est principalement utilisée contre les chiffrements mono-alphabétiques qui substituent chaque lettre par une autre et qui présentent un biais statistique. Nous indiquons aussi la cryptanalyse linéaire qui s'agit de trouver des approximations linéaires entre les bits de sortie, les bits d'entrée et les bits de la clé. Le nombre de clés à envisager pour déchiffrer le message est restreint.

A la fin, nous citons la cryptanalyse différentielle : c'est la comparaison des sorties de l'algorithme quand on lui met en entrée deux messages ayant une différence fixe. On étudie comme varient les sorties si les deux messages ne diffèrent que par un seul bit. Si en déplaçant ce bit à l'intérieur des messages, certains bits des sorties restent inchangés, on a alors trouvé une faille dans l'algorithme et celui-ci est attaquable.

4.3. Analyse de sécurité du système de cyrpto-compression utilisé

4.3.1. Analyse par histogrammes

L'histogramme d'une image naturelle contient les propriétés statistiques de l'image. En plus cet histogramme montre souvent un motif (pattern) comme c'est le cas de l'image présentée dans la figure 4.25 qui présente la première image intra de la séquence RaceHorses.

A contrario, l'histogramme de l'image compressée et cryptée avec la méthode proposée ressemble plus à une loi normale comme indiqué dans la figure 4.26. Rappelons qu'en statistiques, une loi normale est l'une des lois de probabilité les plus adaptées pour modéliser des phénomènes naturels issus de plusieurs événements aléatoires.

Ainsi, dans le cadre d'une cryptanalyse, il devient très difficile de décrypter l'image en se basant sur des analyses statistiques de l'image à décrypter.

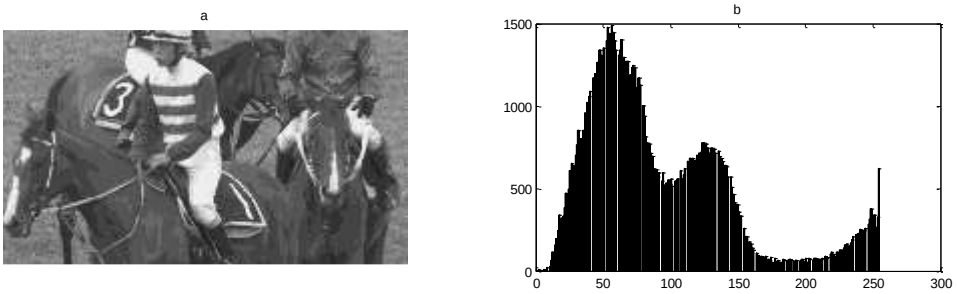


Figure 4. 25- (a) Image d'origine en clair à crypter, (b) histogramme de (a)

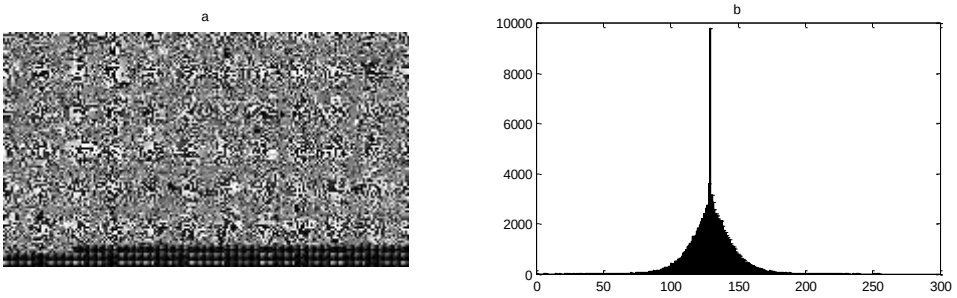


Figure 4. 26- (a) Image compressée et cryptée à base de RAC, (b) histogramme de (a)

4.3.2. Analyse par corrélation temporelle

Une des principales difficultés du cryptage des images provient de la grande corrélation entre les pixels voisins. En effet, nous pouvons souvent établir des relations linéaires entre les pixels voisins en horizontale, verticale ou en diagonale. Pour tester la robustesse de notre algorithme, nous pouvons appliquer une analyse sur la corrélation temporelle entre les pixels voisins avant le cryptage et après le cryptage. Il s'agit d'une analyse classique dans le domaine de la cryptanalyse comme c'est le cas dans [Farajallah 13].

Nous sélectionnons de façon aléatoire 10000 paires de pixels adjacents en horizontale, verticale ou en diagonale de l'image d'origine et de l'image cryptée. Ensuite, nous calculons le coefficient de corrélation conformément aux équations suivantes :

$$\rho_{x,y} = \frac{\mathrm{cov}(x,y)}{\sqrt{D(x)\times D(y)}} \tag{4.5}$$

où

$$\begin{aligned} \operatorname{cov}(x,y) &= \frac{1}{N}\sum_{i=1}^{N}\left(x_i - E(x)\right)\times\left(y_i - E(y)\right) \\ D(x) &= \frac{1}{N}\sum_{i=1}^{N}\left(x_i - E(x)\right)^2 \\ E(x) &= \frac{1}{N}\sum_{i=1}^{N} x_i \end{aligned} \tag{4.6}$$

Dans ces équations, x_i et y_i sont les valeurs de pixels adjacents dans l'image d'origine et leur correspondant dans l'image cryptée. La figure 4.27 montre la corrélation des pixels voisins dans la direction horizontale des images prises des figures 4.25.a et 4.26.a.

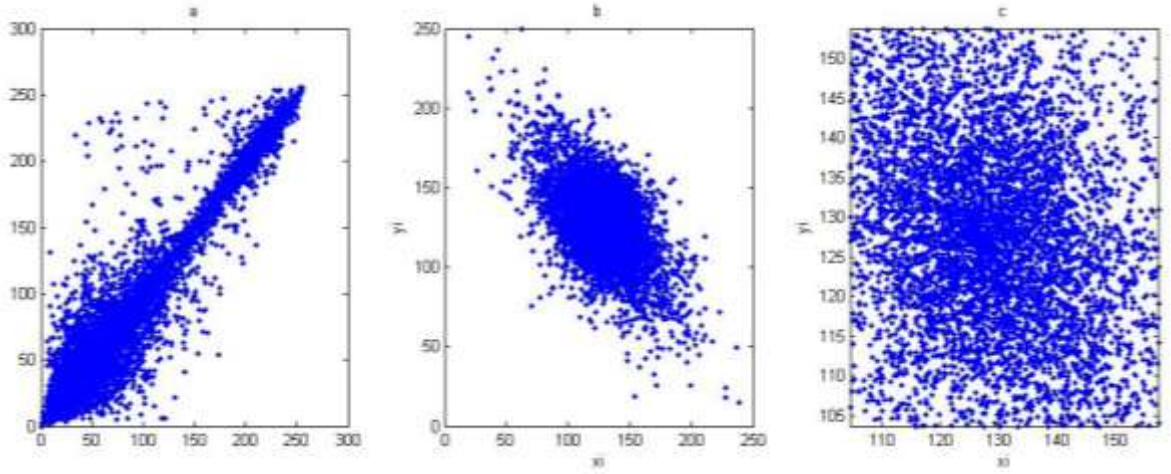


Figure 4. 27- Répartition des pixels voisin horizontalement : (a) dans l'image d'origine, (b) dans l'image cryptée, (c) zoom sur l'image cryptée

En observant le nuage de points dans l'image en clair de la Figure 4.27.a, nous pouvons voir qu'il y a une relation de linéarité entre les pixels voisins. Cette propriété de voisinage disparait avec l'image cryptée. Le zoom sur cette répartition illustré dans la Figure 4.27.c, montre que n'importe quel pixel dont l'intensité est comprise entre 110 et 150 peut prendre n'importe quelle valeur de pixel voisin. Ceci rend le décryptage des images difficile. Pour donner une analyse quantitative ce que nous avançons, nous pouvons nous baser sur les équations du coefficient de corrélation expliqué dans l'équation 4.5. Le tableau 4.5 illustre les différentes valeurs de ce coefficient pour les 3 directions et pour deux séquences vidéo.

Tableau 4. 5- Coefficient de corrélation temporelle

Séquence	Direction	Image d'origine	Image cryptée
Horses (416*240)	Horizontale	0.942	-0.496
	Verticale	0.933	-0.524
	Diagonale	0.895	0.303

Tulipe (176*144)	Horizontale	0.960	-0.531
	Verticale	0.945	-0.541
	Diagonale	0.922	0.381

Nous pouvons voir que pour les images naturelles, le coefficient de corrélation est très proche de 1 qui est la valeur maximale. Plus le coefficient de corrélation est proche de 1, plus les pixels voisins sont fortement corrélés. Dans le cas de l'image cryptée, le coefficient de corrélation présente des valeurs proches de 0.5 ce qui montre la non-corrélation des pixels voisins. Cette valeur aurait pu être plus faible, mais la distribution des pixels montrée dans la figure 4.26.b montre un pic dans l'histogramme autour de la valeur 130. L'écart type autour de ces points n'est pas très élevé ce qui rend le coefficient de corrélation plus élevé (mais reste toujours moins important que dans l'image d'origine). Notons que le signe négatif souligne la relation descendante entre les pixels voisins. Ceci veut dire qu'un coefficient de corrélation qui vaut -1 montre aussi la forte corrélation entre les pixels voisins.

4.3.3. Sensibilité de l'image d'origine à l'attaque

Le test de la sensibilité de l'image d'origine aux attaques permet de conclure sur la robustesse de l'algorithme proposé. L'idée de ce test est de comprendre ce qui se passe lors d'un changement d'un seul bit de l'image d'origine. Soit P1 l'image d'origine. Si un seul bit de l'image P1 change, on obtient l'image P2. Ensuite, le cryptage de P1 et de P2 par la même méthode et la même clé donne les images cryptées C1 et C2. La figure 4.28 illustre le schéma de test de sensibilité adoptée.

La sensibilité de l'image d'origine aux attaques revient à calculer le nombre de bits différents entre C1 et C1 divisé par le nombre total des bits. Ceci s'appelle la distance de Hamming et s'exprime mathématiquement par cette équation :

$$d_{Hamming}(C1,C2) = \frac{\sum_{k=1}^{L\times H\times 8} C1(k) \oplus C2(k)}{L\times H\times 8} \tag{4.7}$$

L et H désignent la taille de l'image d'entrée codée sur 8 bits et $\oplus$ désigne l'opérateur ou exclusif (mesure de nombres de bits différents). Les différents tests réalisés montrent qu'en moyenne le changement d'un bit dans l'image d'origine permet d'avoir une distance de Hamming de 0.24 ce qui est équivalent au changement de 24% de l'image de sortie (changement entre C1 et C2 égal à 24 %).

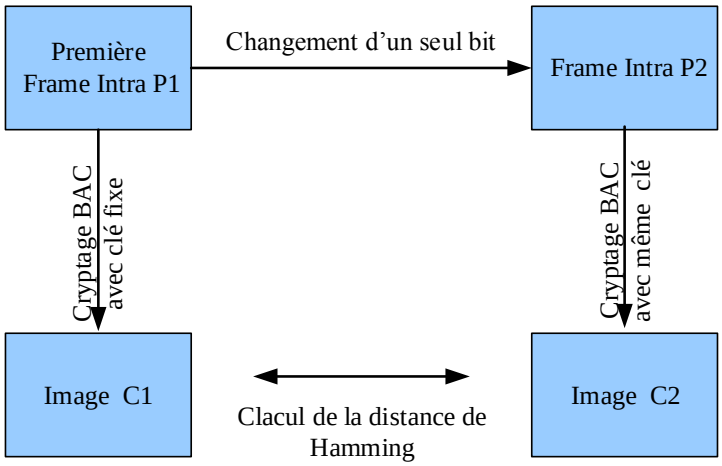


Figure 4. 28- Le schéma de test de sensibilité adoptée

4.3.4. Sensibilité de la clé de cryptage à l'attaque

Le test de la sensibilité de clé de cryptage aux attaques ressemble beaucoup à celui de la section précédente. Nous utilisons les images P1 et P2 obtenus dans la section précédente par un changement d'un seul bit. Ensuite, nous appliquons le cryptage RAC, mais cette fois ci avec un changement d'un seul bit au niveau de clé. Ainsi, l'image P1 est cryptée avec une clé et l'image P2 est cryptée avec une deuxième clé qui diffère de la première d'un seul bit. Le schéma bloc de la figure 4.29 illustre le scénario de mesure de sensibilité de la clé du cryptage aux attaques.De la même manière nous calculons la distance de Hamming entre C1 et C2. Nous trouvons que la distance moyenne est de 0.25 % ce qui signifie que le taux de changement des pixels est de 25%. Ceci démontre la robustesse de la méthode de cryptage proposée, sans oublier qu'il s'agit d'une méthode qui réalise simultanément la compression.

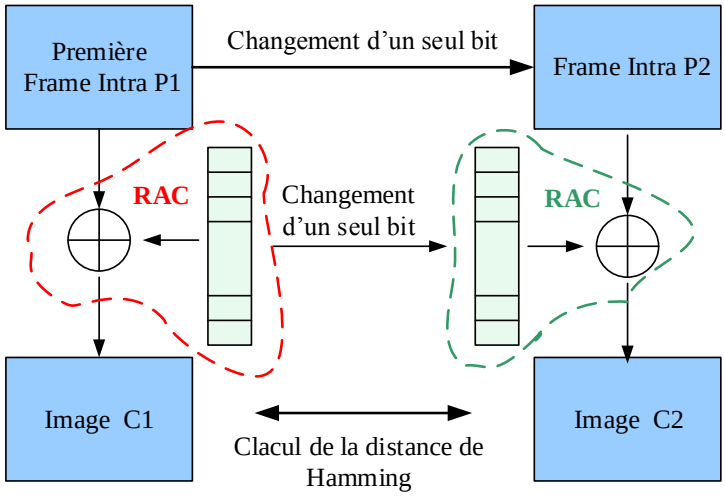


Figure 4. 29- Scénario de mesure de sensibilité de la clé du cryptage aux attaques

D'un autre côté, nous signalons que pour un décryptage avec une mauvaise clé donne toujours une image bruitée. Aussi, en cas de l'obtention de la bonne clé de cryptage, un décryptage avec un «index_crypt» erroné ne permet de visualiser l'image en clair. Ceci constitue un niveau de sécurité supplémentaire. Si la valeur de «index_crypt» dépasse la taille du fichier à décrypter, l'algorithme de décryptage répond par une erreur.

4.3.5 Sensibilité du schéma du cryptage au bruit

La robustesse au bruit du schéma de crypto-compression proposé est étudiée dans cette section. Pour une image (ou Intra Frame) d'entrée, une DCT, une quantification et une application du rangement par ordre zigzag est appliquée. Ensuite, nous cryptons cette image et nous la décryptons avec la clé secrète.

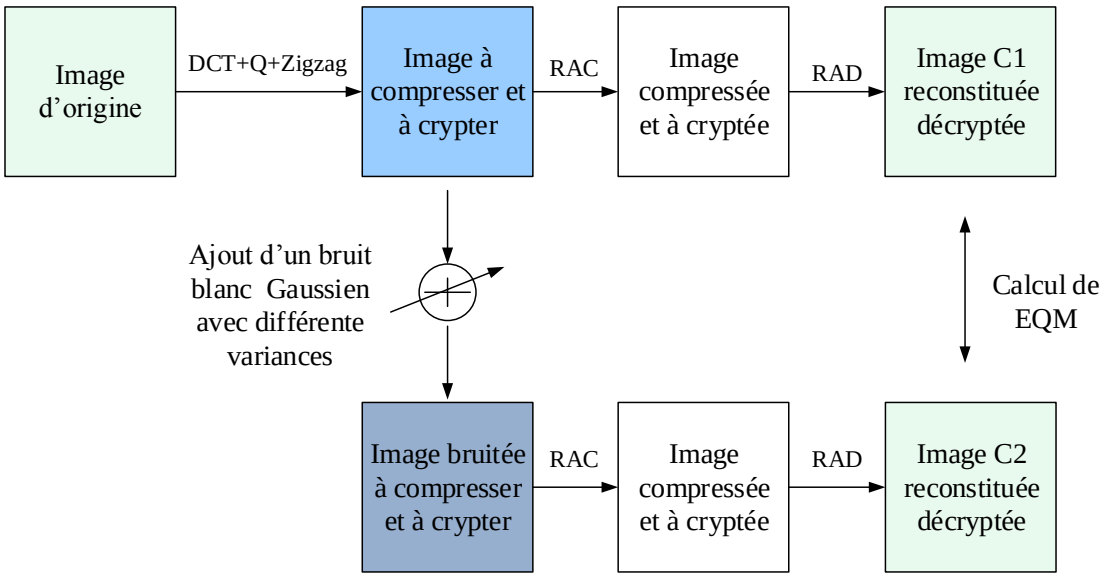


Figure 4. 30- Sensibilité du schéma du cryptage au bruit

Nous obtenons ainsi une image C1 reconstituée après compression et décryptée après cryptage. D'un autre côté, nous ajoutons un bruit banc Gaussien (AWGN) avec une moyenne nulle et une variance de bruit allant de 0,02 à 0,4. L'application du RAC suivie du RAD (Randomized Arithmetic Decoding) nous donne une image C2 reconstituée et décryptée après application du bruit. Le synoptique de cette attaque est détaillé dans la figure 4.30.

Pour tester la robustesse du schéma du cryptage au bruit, nous utilisons une mesure de distance entre les deux images C1 et C2. Cette mesure peut être assurée par l'évaluation l'erreur quadratique moyenne (EQM) entre C1 et C2. Nous avons tracé les différentes valeurs de l'EQM par rapport à la variance du bruit comme indiqué dans la figure 4.31.

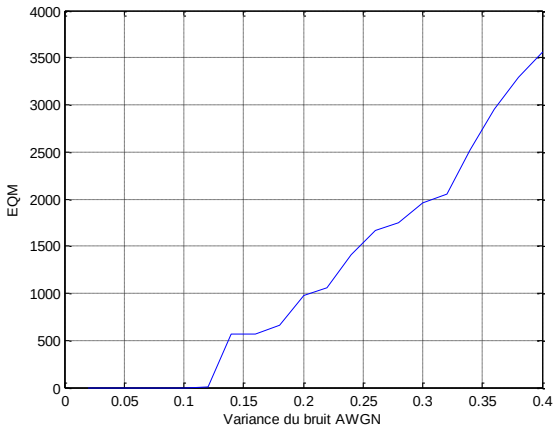


Figure 4. 31- Évaluation de l'EQM pour différentes valeurs de variance du bruit Gaussien

Le pas de la puissance du bruit est fixé à 0,02. Il est évident que plus la variance du bruit est élevée plus la différence entre C1 et C2 est grande.

Figure 4. 32- Résultats de la résistance au bruit. (a) Image d'origine, (b) Image cryptée et décryptée avec la clé secrète sans présence de bruit, (c) Image cryptée et décryptée avec la clé secrète avec présence de bruit de variance égale à 0,3, (d) Image cryptée et décryptée avec la clé secrète avec présence de bruit de variance égale à 0,4.

Ces essaies démontrent que nous pouvons décrypter les images et nous pouvons les reconnaitre malgré les interférences et que le système de crypto-compression peut tolérer un certain niveau de bruit. Pour avoir la preuve de ce que nous avançons, la figure 4.32.a et 4.32.b montrent l'image d'origine et l'image C1 respectivement. Pour des valeurs de puissances de variance de bruit égales à 0,3 et à 0,4 la figure 4.32.c et 4.32.d montrent respectivement les images reconstituées et décryptées. Enfin, nous soulignons que d'autres tests d'attaques ont été réalisés. Comme celui de la robustesse aux occlusions. Ce test n'a malheureusement pas été concluant.

5. Conclusion

Dans ce chapitre, nous avons présenté quelques techniques et algorithmes de cryptage vidéo. Nous avons détaillé une procédure de compression et cryptage simultanés mise en œuvre dans le standard de la compression d'images fixes et animées. Ainsi, nous avons essayé de montrer les performances et les limites du codeur vidéo modifié générant un flux de données protégés.
Si nous comparons les extraits de séquences avec l'originale dans les différentes méthodes proposées, nous ne reconnaissons pas les détails. Notre combinaison du cryptage et compression nous permet de gagner du temps de calcul en conservant le taux de compression initial. La différence importante des rapports de compressions permet de voir un avantage certain au cryptage conjoint à la compression (C&C C). Il présente le meilleur compromis entre RC, PCE, PSNR et temps de calcul puisque c'est un seul niveau de cryptage qui nécessite moins de calcul.
Le cryptage double que nous réalisons n'est pas assez efficace et nécessitera d'autres tests de vérifications sur des images P ou B et peut-être l'utilisation des contextes au niveau codage entropique à rendre adaptatif.

Conclusion et perspectives

Les travaux présentés dans ce document s'intéressent à l'étude et l'implantation numérique de bloc de codage entropique pour le traitement d'images.

Lors d'une première phase, nous avons réussi à simuler la chaîne de codage vidéo H264/AVC en assurant une identification et caractérisation des fonctions matérielles. Une étude comparative est faite pour divers types de codeurs entropiques Huffman standard et adaptatif, codage arithmétique binaire statique et adaptatif pour justifier notre intérêt d'implanter par la suite le meilleur. L'étude et l'application du codage à la compression d'images fixes et animés sont assurées afin de prouver l'efficacité des différents types de codage entropique sans perte, pour une application JPEG comme exemple. Pour y arriver, nous avons étudié l'amélioration de la chaîne de compression en se basant sur des blocs bien définis dans la norme. Une optimisation de la chaîne par l'exploration d'autres fonctions pour le Codec JPEG est effectuée comme le décalage de niveau et le codage arithmétique destinés à améliorer le taux de compression. De plus, nous avons implanté deux codeurs entropiques de type Huffman et arithmétique dans le codec JPEG de référence et nous avons montré que les performances du codeur modifié sont meilleures. Nous pouvons confirmer que les deux méthodes implantées nous donnent des performances meilleures par rapport au JPEG standard de référence en termes de gain de compression et débit binaire. La méthode de Huffman adaptative utilisée nous assure aussi une optimisation intéressante au niveau du flux binaire obtenu au codage pour la même qualité d'image.

Dans le deuxième chapitre, nous avons étudié brièvement les dernières normes de codage vidéo. Ensuite, nous avons détaillé le fonctionnement des techniques de compression associées aux normes H.264/AVC et HEVC. Ces derniers standards fournissent des performances nettement supérieures en comparaison aux normes antérieures. Lors des tests, nous avons utilisé des codeurs logiciels H.264/AVC et HEVC. HEVC inclut de nombreuses techniques nouvelles qui permettent de compresser les vidéos beaucoup plus efficacement que les normes et fournit plus de flexibilité aux applications en environnement réseau. Bien que le HEVC n'apporte pas une rupture technologique par rapport aux normes de codage vidéo précédentes, il comporte plusieurs modifications impliquent des calculs très complexes. L'analyse de la complexité algorithmique à travers les résultats de profilage montre que le

codeur entropique est le goulot d'étranglement au niveau de la complexité matérielle. Dans le chapitre suivant, nous nous proposons de décrire une solution hardware considéré comme accélérateur matériel.

Après, nous avons détaillé les principes de fonctionnement de l'encodeur/décodeur CABAC. En raison de ces performances élevées et des fonctionnalités supportées, ce codeur entropique est adopté dans les derniers standards vidéo. Nous nous sommes basés sur les processus décrits dans le draft et les différents algorithmes associés afin d'implanter et validé l'architecture du codec entropique proposée sur FPGA Xilinx Virtex. Ainsi, nous avons réalisé une mesure de la puissance consommée des différents modules VHDL réalisés. La conception proposée permet d'atteindre l'objectif fixé concernant le prototypage rapide. Toutefois, un faible pourcentage des ressources matérielles a été utilisé pour réaliser les modules développés spécifiquement en langage matériel. Ceci nous permettrait d'envisager ajouter d'autres traitements de la chaine de CODEC vidéo comme l'estimation de mouvement.

Enfin, nous avons présenté quelques techniques et algorithmes de cryptage vidéo. Nous avons détaillé une procédure de compression et cryptage simultanés mise en œuvre dans le standard de la compression d'images fixes et animées. Ainsi, nous avons essayé de montrer les performances et les limites du codeur vidéo modifié générant un flux de données protégés.

Si nous comparons les extraits de séquences avec l'originale dans les différentes méthodes proposées, nous ne reconnaissons pas les détails. Notre combinaison du cryptage et compression nous permet de gagner du temps de calcul en conservant le taux de compression initial. La différence importante des rapports de compressions permet de voir un avantage certain au cryptage conjoint à la compression (C&C C). Il présente le meilleur compromis entre RC, PCE, PSNR et temps de calcul puisque c'est un seul niveau de cryptage qui nécessite moins de calcul.

A la suite de ces travaux, nous pouvons envisager l'utilisation des modules IP développés pour l'implantation matérielle/logicielle de la norme H.264 pour des profils avancés sur notre plateforme ou sur une plateforme comportant un processeur embarqué puissant (comme le processeur PowerPC d'un circuit Virtex de Xilinx).

Bibliographie

[Acrypta]: www.acrypta.com/lexique/trapdoor.html.

[Ahn 04]: J. Ahn, H. J. Shim, B. Jeon et I. Choi, “Digital video scrambling method using intra prediction mode,” in Proc. Adv. Multimedia Information Process, LNCS 3333, pp. 386–393, Decembre 2004.

[Akkar 04] : Attaques et méthodes de protections de systèmes cryptographiques embarqués, Mehdi-laurent Akkar, 160 pages, 2004

[Alfalou 99] : A. Alfalou, G. Keryer, et J. L. de Bougrenet de la Tocnaye, Optical Implementation of Segmented Composite Filtering, Applied Optics, vol. 38, 1999.

[Alfalou 09] : Ayman Alfalou et C. Brosseau, Optical image compression and encryption methods, Adv. Opt. Photon, 1, pp.589-636, 2009.

[Alfalou 10] : A. Alfalou et C. Brosseau, Understanding Correlation Techniques for Face Recognition: From Basics to Applications, 2010.

[Almasalha 08] F. Almasalha, N. Agarwal et A. Khokhar, “Secure multimedia transmission over RTP,” in Proc. 8th IEEE ISM, pp. 404–411, Decembre 2008.

[Altera 13]: Altera, Design Debugging Using the SignalTap II Logic Analyzer,‖ in in Quartus II Handbook Version 13.0, vol. 3, no. Mai 2013.

[Arachchi 09]: H. Kodikara Arachchi, X. Perramon, S. Dogan et A. M. Kondoz, “Adaptation-aware encryption of scalable H.264/AVC video for content security,” Signal Process. Image Commun, vol. 24, pp. 468–483, 2009.

[Ash 90]: Robert B. Ash, "Information Theory ", Dover Publications, Inc, New-York, 1990.

[Bardone 08] : Davide Bardone, Elias S.G. Carotti, Juan Carlos De Martin , Adaptive Golomb Codes For Level Binarization In The H.264/AVC FRExt Lossless Mode, Conference: Signal Processing and Information Technology, ISSPIT 2008.

[Battail 97] : Gérard BATTAIL, " Théorie de l’Information ", Masson, 1997.

[Berbain 07] Côme Berbain, Analyse et conception d'algorithmes de chiffrement à flot, 1 vol.146 p, Paris 7, 2007

[Bergeron 05] C. Bergeron et C. Lamy-Bergor, "Compliant selective encryption for H.264/AVC video streams," in Proc. IEEE Workshop MMSP, pp. 1–4, Octobre 2005.

[Berlekamp 68] E.R. Berlekamp. Algebraic Coding Theory. McGraw-Hill, 1968.

[Blake 99] Ian F. Blake, G. Seroussi et N. P, Smart.El liptic curves in cryptography. Cambridge University Press, New York, NY, USA, 1999.

[Boisson 05] : G. Boisson, Représentations hautement scalables pour la compression vidéo sur une large gamme de débits/résolutions, Université de Rennes-1, Janvier 2005.

[Boneh 00] D.Boneh , A. Joux et P.Q. Nguyen, Why textbook ElGamal and RSA encryption are insecure, LNCS , volume 1976. – Springer-Verlag, 2000.

[Bossen 12]: F.Bossen, B. Bross, K. Sühring, and D. Flynn, "HEVC complexity and implementation analysis," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1684–1695, Dec. 2012.

[Boudjada 08]: Amed Boudjada. Codage video distribué. PhD thesis, Université Mentouri, 2008.

[Bresson 04] E. Bresson, CRYPTOGRAPHIE : Chiffrement asymétrique, SGDN/DCSSI. Laboratoire de cryptographie, 2004

[Briceno 99] M. Briceno, I. Goldberg et D. Wagner, A pedagogical implementation of A5/1. http://jya.com/a51-pi.htm, 1999.

[Britanak 07] : V. Britanak, P. Y. Yip, and K. R. Rao, Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations. Academic Press, London, 2007.

[Brunel 04] : Lionel Brunel. Indexation vidéo par l'analyse de codage. Automatic. Université Nice Sophia Antipolis, 2004.

[Buchmann 04] J.Buchmann, Introduction to cryptography, Springer Science & Business Media, 335 pages, Juillet 2004

[Budagavi 12]: M. Budagavi and V. Sze, coeff−abs−level−remaining Maximum Codeword Length Reduction, document JCTVC-J0142, Joint Collaborative Team on Video Coding (JCT-VC), Juillet, 2012.

[Carrillo 08] P. Carrillo, H. Kalva et S. Magliveras, "Compression independent object encryption for ensuring privacy in video surveillance," in Proc. ICME, pp. 273–276, Juin 2008.

[Chebbeh 2012] : T. Chebbeh « Etude et Intégration de la Chaîne de Traitement Intra du Décodeur H.264/AVC sous une plateforme Embarquée » 2012 l'Ecole Nationale d'Ingénieurs de Sfax.

[Cheddad 10] Abbas Cheddad, Joan Condell, Kevin Curran et Paul Mc Kevitt. Digital image steganography: Survey and analysis of current metho ds.Signal Processing, 90(3):727-752, Mars 2010.

[Compressor 12]. http://code.google.com/p/jpeg-compressor/

[Cover 06] : Thomas M. Cover, Joy A. Thomas, Elements of Information Theory, Wiley-Interscience,200 .

[DES] Data Encryption Standard, http://fr.wikipedia.org/wiki/Data_Encryption_Standard

[Draft 14] : Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC MPEG & ITU-T VCEG), "Advanced video coding for generic audiovisual services," Joint Video Team (JVT), Février 2014.

[Dufaux a 08] F. Dufaux et T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," IEEE Trans. Circuits Syst. Video Technol, vol. 18, pp.1168–1174, Aout 2008.

[Dufaux b 08] F. Dufaux et T. Ebrahimi, "H.264/AVC video scrambling for privacy protection," in Proc. IEEE ICIP, pp. 1688–1691, Octobre 2008.

[Engel 09] D. Engel, T. Stutz et A. Uhl, "A survey on JPEG2000 encryption,"Multimedia Syst, vol. 15, pp. 243–270, 2009.

[Farajallah 13] : Mousa Farajallah, Z. Fawaz , Safwan El Assad, Olivier Déforges, Efficient image encryption and authentication scheme based on chaotic sequences The 7th International Conference on Emerging Security Information, Systems and Technologies, Aug 2013, Barcelone, Spain, p.150-155, 2013

[Fan 07] Y. Fan, J. Wang, T. Ikenaga, Y. Tsunoo et S. Goto, "A new video encryption scheme for H.264/AVC," in Proc. Adv. Multimedia Information Process, pp. 246–255, 2007.

[Fernandes 15]: F. Fernandes, X. Ducloux, Z. Ma, E. Faramarzi, P. Gendron, et J. Wen, "The green metadata standard for energy-efficient video consumption," MultiMedia, IEEE, vol. 22, no. 1, pp. 80–87, Janvier 2015.

[Fournier 72] : Alain Fournier, Transformée de Hadamard. Application de la théorie de l'information à l'étude de l'opérateur humain, 266 p, 1972

[Geffe 73] P.R. Geffe, How to protect data with ciphers that are really hard to break, Electronics, pages 99-101, 1973.

[Girod 93] B. Girod. What's wrong with mean-squared error? Pages 207‑220, 1993.

[Grangetto 06] Grangetto, M. Magli et E. Olmo, G. Multimedia Selective Encryption by Means of Randomized Arithmetic Coding, Multimedia, IEEE Transactions on 8(5) 905-917, 2006.

[Grangetto 07] M. Grangetto, E. Magli et G. Olmo, "Conditional access to H.264/AVC video by means of redundant slices," in Proc. IEEE ICIP, vol. 6, pp. 485–488, Septembre 2007.

[Guillois 96] : Jean-Paul Guillois, Techniques de compression des images, Hermès, Lavoisier, 252 p, 1996

[Ham 86] : R. W. Hamming, "Coding and Information Theory", 2nd Ed. Prentice Hall, Englewood Cliffs, 1986.

[HAN 05]: J.H. Han, M.Y. Lee, Y.H. Bae, and H.J. Cho, "Application Specific Processor Design for H.264 Decoder with a Configurable Embedded Processor", ETRI Journal, vol. 27, Octobre 2005.

[Haskell 97] : Haskell B G., Puri A et Netravalli A. N,"Digital Video: An introduction to MPEG-2", Chapman and Hall, New York, USA, 1997.

[Havet 10] Frédéric Havet, Histoire de la cryptographie, MASCOTTE, commun I3S (CNRS/UNSA)-INRIA Sophia Antipolis, Fête de la science, pp. 21-24, Octobre 2010.

[Hellwagner 09] H. Hellwagner, R. Kuschnig, T. Stutz et A. Uhl, "Efficient in-network adaptation of encrypted H.264/SVC content," Elsevier J. Signal Process. Image Commun, vol. 24, pp. 740–758, Juillet 2009.

[Horner 84] J. Horner et P. Gianino, Phase-Only Matched Filtering, Applied Optics, vol. 23, pp.812_816, 1984.

[Horner 92]: J. L. Horner, Metrics for assessing pattern-recognition performance, Applied Optics, vol. 31, p. 165-166, 1992.

[Howard 92]: P.G. Howard and J.S. Vitter. Practical implementations of arithmetic coding. Image and Text Compression, 13(7):85–112, 1992.

[H264 15]: H264. https://fr.wikipedia.org/wiki/H.264, 2015.

[H265 15]: Http: //www.h265.net/

[JPEG 14] : http://www.jpeg.org/

[Iqbal 06] : R. Iqbal, S. Shirmohammadi et A. El-Saddik, "Secured MPEG-21 digital item adaptation for H.264 video," in Proc. ICME, pp. 2181–2184, Juillet 2006.

[Iqbal 07] : R. Iqbal, S. Shirmohammadi et A. El Saddik, "A framework for MPEG-21 DIA based adaptation and perceptual encryption of H.264 video," Proc. SPIE Multimedia Comput. Netw., vol. 6504,pp. 1–12, 2007.

[ISMA 07] : ISMA Encryption and Authentication Specification 2.0, Internet Streaming Media Alliance, San Francisco, CA, Novembre 2007.

[ISO 00]: ISO/IEC 13818-2: Information Technology, "Coding of Moving Pictures and Associated Audio Information. Part 2: Video", 2000.

[ISO 14]: ISO/IEC 23001-11: Information-Technology, “MPEG Systems Technologies - Part 11: Energy-Efficient Media Consumption (Green Metadata),” Oct. 2014.

[ITU] : http://www.itu.ch

[Jridi 14] : Maher JRIDI, cours "Compression des données dans les nouveaux standards de codage vidéo", ISEN-Brest 2014.

[Kahn 96] : David Kahn, The Code breakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet Hardcover, Decembre1996.

[Kim 07] : Y. Kim, S. Yin, T. Bae et Y. Ro, “A selective video encryption for the region of interest in scalable video coding,” in Proc. TENCON IEEE Region 10 Conf, pp. 1–4, Octobre 2007.

[Kuschnig 08] : R. Kuschnig, I. Kofler, M. Ransburg et H. Hellwagner, “Design options and comparison of in-network H.264/SVC adaptation,” J. Vis. Commun. Image Representation, vol. 19, pp. 529–542, Decembre 2008.

[Langdon 84] G.G. Langdon .An introduction to arithmetic coding .IBM Journal of Research and Development, vol. 28, pp.135‑149, 1984.

[Lee 06] H.J.Lee et J.Nam,“Low complexity controllable scrambler/descrambler for H.264/AVC in compressed domain,” in Proc.ACM Multimedia, pp. 93–96, 2006.

[Lian 05] S. Lian, Z. Liu, Z. Ren et Z. Wang, “Selective video encryption based on advanced video coding,” in Pro. PCM Multimedia, Adv. Multimedia Inform. Process, LNCS 3768, pp. 281–290, 2005.

[Lian 06] S. Lian, Z. Liu, Z. Ren et H. Wang, “Secure advanced video coding based on selective encryption algorithms,” IEEE Trans. Consumer Electron, vol. 52, pp. 621–629, Mai 2006.

[Lian 07] S. Lian, Z. Liu, Z. Ren et H. Wang, “Commutative encryption and watermarking in video compression,” IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 6, pp. 774–778, Juin 2007.

[Lian 08] S. Lian, J. Sun, G. Liu et Z. Wang, “Efficient video encryption scheme based on advanced video coding,” Multimedia Tools Aplicat, vol. 38, pp. 75–89, Mars 2008.

[Li 05] Y. Li, L. Liang, Z. Su et J. Jiang, “A new video encryption algorithm for H.264,” in Proc. 5th ICICS, pp. 1121–1124, Decembre 2005.

[Li 08] C. Li, X. Zhou et Y. Zhong, “NAL level encryption for scalable video coding,” in Proc. Adv. Multimedia Inform. Process, pp. 496–505, Decembre 2008.

[Liu 07] Y. Liu, C. Yuan et Y. Zhong, “A new digital rights management system in mobile aplications using H.264 encryption,” in Proc. 9th Int. Conf. Adv. Commun. Technol, vol. 1, pp. 583–586, Fevrier2007.

[Lugt 64] A. V. Lugt, Signal Detection by Complex Filtering, IEEE Transactions on information theory, vol. 10, pp. 139_145, 1964.

[L’EXCELLENT 06] : Fabien L’EXCELLENT, ‘Mise En OEuvre D’une Application De Traitement D’image Sur La PlateForme Ml310’, Mémoire d’ingénieur CNAM, Centre Régional Associe De Bourgogne, Centre D’enseignement De Dijon, Décembre 2006.

[Magli 06] : E. Magli, M. Grangetto et G. Olmo, “Conditional access to H.264/AVC video with drift control,” in Proc. IEEE ICME, pp. 1353–1356, Juillet 2006.

[Malvar 03] : Malvar H., Hallapuro A., Karczewicz M., et Kerofsky L., "Low-Complexity Transform and Quantization in H.264/AVC", IEEE Trans. Circuit Syst. Video Technol., vol. 13, pp. 598-603, Juillet 2003.

[Marpe 03]: Marpe, D.; Schwarz, H.; Wiegand, T., "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.13, no.7, pp.620-636, Juillet 2003

[Margot 00] : P.Margot, P.J.Saukko G.C.Knupfer, J.A.Siegel, Fingerprint Sciences, Encyclopedia of Forensic Sciences. London : Academic Press, p 1054-1058, 2000.

[Martin 2010] : Aurélie Martin, Représentations parcimonieuses adaptées à la compression d'images, PhD thesis, Université Rennes 1, 2010

[Massey 69] : J.L. Massey, Shift-register synthesis and BCH decoding, IEEE Transactions on Information Theory, 15:122-127, janvier 1969.

[Massoudi 08] A. Massoudi, F. Lef`ebvre, C. De Vleeschouwer, B. Macq et J.J. Quisquater, "Overview on selective encryption of image and video, challenges and perspectives", EURASIP J. Inform.Security, vol. 2008, pp. 18, 2008.

[Mian 07] C. Mian, J. Jia et Y. Lei, "An H.264 video encryption algorithm based on entropy coding" in Proc. 3rd Int. Conf. IIH-MSP, pp. 41–44, 2007.

[MPEG]: http://www.mpeg.org.

[Minh 07] : Vu Duc Minh. Protection de séquence d'images comprimés et portables : Application à la vidéo surveillance, LIRMM, Montpellier, France, 2007.

[Miroslva 01] : Miroslva Balik et Milan Simanek, "A method for the construction of minimum redundancy codes", Proceeding of the Prague Stringology conference, Septembre 2001.

[Murât 78] Kunt, Murât, et Ottar Johnsen. "Codes de blocs a préfixe pour la réduction de redondance d'images." Annales des Télécommunications. Vol. 33. No. 7-8. Springer-Verlag, 1978.

[Neji 13] N.NEJI, M.JRIDI, A.ALFALOU, N.MASMOUDI « A CABAC Codec Design of H264/AVC with secure Arithmetic Coding » IS&T/SPIE Electronic Imaging, California USA, 2013

[NIST a] NIST. FIPS 46-3, Data Encryption Standard (DES), Octobre 1999. Cf. [En ligne] Adresse : http://csrc.nist.gov/encryption/tkencryption.html.

[NIST b] NIST. FIPS 197, Advanced Encryption Standard (AES), November 2001. Cf. [En ligne] Adresse : http://csrc.nist.gov/encryption/aes/.

[Noorkami 07] M. Noorkami et R. M. Mersereau, "Digital video watermarking in P-frames," in Proc. SPIE Conf. Security Steganography Watermarking Multimedia Contents IX, vol. 6505. Janvier 2007.

[Norcen 03] R. Norcen, M. Podesser, A. Pommer, H.P. Schmidt et A. Uhl: « Confidential storage and transmission of medical image data », Computers in Biology and Medicine, vol. 33, pp. 277-292, 2003.

[Ohm 12]: J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards #x2014fIncluding High Efficiency Video Coding (HEVC)," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1669–1684, décembre 2012.

[Pande 11] A.Pande, Algorithms for Secure Multimedia Delivery over Mobile Devices and Mobile Agents, 2011.

[Park 06]: S.M. Park, M.Y. Lee, S.C. Kim, K.S. Shin, I.K. Kim, H.J. Cho, H.B. Jung, and D.D. Lee, "VLSI Implementation of H.264 Video Decoder for Mobile Multimedia Application", ETRI Journal, 525-528, Août 2006.

[Puech 2004] W.Puech et J.M. Rodrigues. Sécurisation d'Image par Crypto-Tatouage. CORESA'04 : 9ème Conférence Nationale de Compression et Représentation des Signaux Audiovisuels, Lille (France), pp.215-218, Mai 2004

[Reza 94] : Fazlollah M. Reza, " An Introduction to Information Theory ", Dover Publications, Inc, New-York, 1994.

[Richardson 03] : I. E. G. Richardson, H.264, MPEG-4 Video Compression:Video Coding for Next Generation Multimedia. New York, Wiley, 2003.

[Richardson 10]: Iain E. Richardson, "The H.264 Advanced Video Compression Standard", John Wiley and Sons, second edition, 2010.

[Rioul 07] Rioul, O., 2007. Théorie de l'information et du codage. Hermes Science – Lavoisier

[Rissanen 76] : J.J. Rissanen . "Generalized Kraft inequality and arithmetic coding". IBM Journal of Research and Development, 200–203, May 1976

[Rivest 92] R. Rivest. The RC4 encryption algorithm, RSA Data Security, 1992.

[Rodrigues 06] J. Rodrigues et W. Puech, Protection en temps réel des visages dans une séquence d'images par cryptage partiel et sélectif, Proc. 11th Colloque Compression et Représentation des Signaux Audiovisuels,CORESA'06, Caen, France, novembre 2006.

[Salomon 07] : David Salomon, Data Compression, Springer-Verlag London, ISBN 978-1-84628-603-2, 2007

[Saponara 04] : S. Saponara, K. Denolf, G. Lafruit, C. Blanch, and J. Bormans, "Performance and Complexity Co-evaluation of the Advanced Video Coding Standard for Cost-Efective Multimedia Communications," EURASIP J. Adv. Signal Process., vol. 2004, no. 2, p. 2144371, Mar. 2004.

[Say 05] : K. Sayood. "Introduction to Data Compression". Morgan Kaufmann, 2005.

[Shahid 09] Z. Shahid, M. Chaumont et W. Puech, "Fast protection of H.264/AVC by selective encryption of CABAC," in Proc. IEEE ICME, pp. 1038–1041, Juin 2009.

[Shannon 48] : Claude E. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal, vol. 27, p. 379-423 et 623-656, Juillet et Octobre, 1948.

[Shannon 49] : Claude E. Shannon, Communication Theory of Secrecy Systems, Bell System Technical Journal, Vol 28, p. 656-715, Octobre 1949.

[Spartan 6]: http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/silicon_devices/fpga/spartan-6.html

[Stern 04] Jacques Stern, Louis Granboulan, Phong Nguyen et David Pointcheval, Conception et preuves d'algorithmes cryptographiques, cours de magistère, école normale supérieure, 2004.

[Stutz 11] T. Stutz et A. Uhl, A survey of H.264 AVC/SVC encryption, IEEE Transactions on Circuits and Systems for Video Technology, doi:10.1109/TCSVT.2011.2162290, 325-340, 2011.

[Su 10] P.C. Su, C.W. Hsu et C.Y. Wu, "A practical design of content protection for H.264/AVC compressed videos by selective encryption and fingerprinting," Multimedia Tools Aplicat, vol. 52, pp.529–549, Janvier 2010.

[Synplicity 07] : Synplicity Inc., 'Outil de synthèse logic « Synplify »', disponible sur le site : www.synplicity.com, 2007.

[Sze 09]: A high throughput CABAC algorithm using syntax element partitioning, Vivienne Sze, Anantha P. Chandrakasan, ICIP, Cairo, Egypt, 2009

[Technicolor]: http://www.technicolor.com/en/solutions-services/connected-home/set-top-boxes/satellite-set-top-boxes/dst839

[Tjalkens 00] : T.Tjalkens, The complexity of minimum redundancy coding, In Proc.Intl. Conf. Inform. Theory, page 373, June 2000.

[Thomas 07] N. Thomas, D. Lefol, D. Bull et D. Redmil, "A novel secure H.264 transcoder using selective encryption," in Proc. IEEE ICIP, pp. IV-85–IV-88, Septembre 2007.

[Tocnaye 97] J. L. de Bougrenet de la Tocnaye, E. Quémener, et Y. Pétillot, Composite Versus Multichannel Binary Phase-Only Filtering, Applied Optics, vol. 36, 1997.

[Varalakshmi 10] L.M.Varalakshmi et Dr. G. Florence Sudha2 Performance Enhancement of Binary Randomized Arithmetic Coding for Multimedia Encryption, International J. of Recent Trends in Engineering and Technology, Vol. 3, Mai 2010.

[Van 02] M. Van Droogenbroeck et R. Benedett: Techniques for a Selective Encryption of Uncompressed and Compressed Images. In Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS), Ghent, Belgium, Septembre, 2002.

[Vaudenay 06] S.Vaudenay, A Classical Introduction to Cryptography: Applications for Communications Security, Springer, 2006

[Vita]: VITA 2000, FMC-IMAGEON Rev.B - Hardware User Guide, www.em.avnet.com

[Vivet 97] [En ligne] Adresse : http://thomas.vivet.free.fr/crypto.html.

[Wagner 93] : Charles Wagner, " De l'image vers la compression ", Rapport de Recherche de l'INRIA, Septembre 1993.

[Wenger 03] S. Wenger, "H.264/avc over ip," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 13, pp. 645 – 656, july 2003.

[Westwater 97] : Westwater R et Furht B, Real-Time Video Compression Techniques and Algorithms , Florida Atlantic University, 1997.

[Whitman 07] M. Whitman et H. Mattord: Principles of information security. Course Technology Press, pp. 56, USA, 2007.

[Wiegand a 03]: T. Wiegand and G. Sullivan: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. (ITU-T Rec H.264|ISO/IEC 14496-10 AVC) Mars 2003.

[Wiegand b 03] T. Wiegand, G.J. Sullivan, G. Bjntegaard, et A. Luthra. Overview of the h.264/avc video coding standard. IEEE Trans. on Circuits and Systems for Video Technology, 13(7):560–576, juillet 2003.

[wikipedia a] [En ligne] Adresse : http://fr.wikipedia.org/wiki/Chiffrement_RSA.

[Witten 87]: I.H. Witten, R.M. Neal, et J.G. Cleary. Arithmetic coding for data compression. Communications of the ACM, 30(6):520–540, June 1987.

[Won 06] Y. G. Won, T. M. Bae et Y. M. Ro, “Scalable protection and access control in full scalable video coding,” in Proc. 5th IWDW, LNCS 4283, pp. 407–421, Novembre 2006.

[Xilinx] : Xilinx Inc., ‘Outil de synthèse logique ISE-V9.2i’.

[Zou 06] Y. Zou, T. Huang, W. Gao et L. Huo, “H.264 video encryption scheme adaptive to DRM,” IEEE Trans. Consumer Electron., vol. 5, pp. 1289–1297, Novembre 2006.

[Zou 10]: D. Zou et J. Bloom, “H.264 stream replacement watermarking with CABAC encoding,” in Proc. IEEE ICME, pp. 117–121, Juillet 2010.

Annexe A

Tableau A. 1- Coefficients DC de compression Huffman de la luminance

Catégorie (CAT)	Code binaire
0	00
1	010
2	011
3	100
4	101
5	110
6	1110
7	11110
8	111110
9	1111110
10	11111110
11	111111110

Tableau A. 2- Coefficients AC de compression Huffman de la luminance

(RUN,CAT)	Code binaire	(RUN,CAT)	Code binaire
0.0 (EOB)	1010	5,3	1111111110011110
0,1	00	5,4	1111111110011111
0,2	01	5,5	1111111110100000
0,3	100	5,6	1111111110100001
0,4	1011	5,7	1111111110100010
0,5	11010	5,8	1111111110100011
0,6	1111000	5,9	1111111110100100
0,7	11111000	5,10	1111111110100101
0,8	1111110110	6,1	1111011
0,9	1111111110000010	6,2	111111110110
0,10	1111111110000011	6,3	1111111110100110
1,1	1100	6,4	1111111110100111
1,2	11011	6,5	1111111110101000

1,3	1111001	6,6	1111111110101001
1,4	111110110	6,7	1111111110101010
1,5	11111110110	6,8	1111111110101011
1,6	1111111110000100	6,9	1111111110101100
1,7	1111111110000101	6,10	1111111110101101
1,8	1111111110000110	7,1	11111010
1,9	1111111110000111	7,2	111111110111
1,10	1111111110001000	7,3	1111111110101110
2,1	11100	7,4	1111111110101111
2,2	11111001	10,6	1111111111001011
2,3	1111110111	10,7	1111111111001100
2,4	111111110100	10,8	1111111111001101
2,5	1111111110001001	10,9	1111111111001110
2,6	1111111110001010	10,10	1111111111001111
2,7	1111111110001011	11,1	1111111001
2,8	1111111110001100	11,2	1111111010000
2,9	1111111110001101	11,3	1111111010001
2,10	1111111110001110	11,4	1111111010010
3,1	111010	11,5	1111111010011
3,2	111110111	11,6	1111111010100
3,3	111111110101	11,7	1111111010101
3,4	1111111110001111	11,8	1111111010110
3,5	1111111110010000	11,9	1111111010111
3,6	1111111110010001	11,10	1111111011000
3,7	1111111110010010	12,1	1111111010
3,8	1111111110010011	12,2	1111111111011001
3,9	1111111110010100	12,3	1111111111011010
3,10	1111111110010101	12,4	1111111111011011
4,1	111011	12,5	1111111111011100
4,2	1111111000	12,6	1111111111011101
4,3	1111111110010110	12,7	1111111111011110
4,4	1111111110010111	12,8	1111111111011111

4,5	1111111110011000	12,9	1111111111100000
4,6	1111111110011001	12,10	1111111111100001
4,7	1111111110011010	13,1	11111111000
4,8	1111111110011011	13,2	1111111111100010
4,9	1111111110011100	13,3	1111111111100011
4,10	1111111110011101	13,4	1111111111100100
5,1	1111010	13,5	1111111111100101
5,2	11111110111	13,6	1111111111100110

Le tableau A.2 de codage AC se compose d'un code binaire Huffman pour chaque événement composé possible et il montre les codes binaires pour toutes les combinaisons possibles de RUN et CAT. Le format du bit additionnel est le même que le codage de DIFF dans des coefficients DC. Pour le $k^{ième}$ coefficient AC dans l'ordre de balayage de zigzag, COEF(k), les bits ajoutés sont les bits d'ordre inférieur de COEF(k) si COEF(k) est positif, ou les bits d'ordre inférieur de COEF(k) - 1, si COEF(k) est négatif.

Voici un exemple afin d'expliquer la manière de créer le vecteur Huffman pour faire le codage par entropie. Des coefficients quantifiés d'un bloc de luminance sont montrés dans la figure A.1. Supposant que le coefficient DC dans le bloc précédent de luminance était égale à 29, nous voulons trouver les codes binaires pour le codage des coefficients DC et AC.

31	18	0	0	0	0	0	0
−21	−13	0	0	0	0	0	0
0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure A. 1-Exemple de coefficients quantifiés d'un bloc 8x8

On a DIFF = 31 - 29 = 2. CAT = 2 et selon le tableau 1, le code Huffman pour cette valeur de CAT est 011. Pour trouver les bits ajoutés, puisque DIFF = 2 > 0, et 2 = 000... 0010, donc les bits ajoutés sont 10. Par conséquent, le code binaire pour coder ce coefficient DC est 01110.Le balayage commence à partir du premier coefficient AC non nul, qui a une valeur de 18. La valeur de CAT pour 18 est 5, et puisqu'il n'y a aucun coefficient AC nul avant lui, puis

RUN = 0. Par conséquent, on aura (0, 5). D'après le tableau A.2, le code binaire pour (0, 5) est 11010. Les bits d'ordre inférieur de COEF(k) = 18 = 000... 010010 sont 10010. En conséquence, le premier code binaire AC est1101010010. Le troisième coefficient AC non nul dans le balayage est -13, qui a un coefficient nul avant lui. Donc, RUN = 1 et, la valeur de CAT correspondante est 4. En consultant le tableau 1, le code binaire pour (RUN = 1, CAT = 4) est 111110110.

COEF(k) -1 = -13 - 1 = -14 = 111... 110010. Donc, le deuxième champ est 0010, et le code binaire entier est 1111101100010. Puisque 5 est le dernier coefficient AC non nul, donc le codage se termine ici et la fin du code du bloc qui est défini comme (0, 0) est transmise. Son code binaire est 1010.

Dans le processus de décodage, c'est l'inverse de toutes les étapes précédente en commençant par le décodage Huffman jusqu'à la DCT inverse (IDCT) pour reconstituer finalement l'image. Rappelons que le codage entropique est un codage sans perte et par conséquent, les coefficients de la DCT sont correctement reconstruits.

Annexe B

Tableau B. 1. Simulations du JM18 pour différents résolutions

- 416x240

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
BasketballPass	2378.88	13.998	41.879	1328.40	11.827	38.188	704.24	15.180	34.817	376.16	14.013	31.895
BlowingBubbles	4821.52	14.361	40.198	2394.64	12.588	36.205	1193.76	11.675	32.765	598.80	16.386	29.731
BQSquare	5059.68	14.926	40.537	2813.28	12.503	36.168	1482.48	11.114	32.424	768.40	10.370	29.084
RaceHorses	5494.08	20.605	41.306	3124.24	17.221	37.267	1699.20	16.622	33.389	888.88	16.546	30.149

- 832x480

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
BasketballDrill	8799.44	57.324	41.004	4684.32	50.545	37.591	2435.52	67.563	34.838	1326.32	57.243	32.360
PartyScene	20314.16	58.213	40.685	11566.16	51.825	36.129	6394.80	49.085	32.165	3369.92	67.667	28.501
BQMall	10348.72	51.752	40.680	5259.68	48.643	37.630	2905.60	57.091	34.473	1601.52	48.796	31.273
RaceHorses	22270.96	83.672	41.564	11444.24	76.642	37.585	5656	66.978	33.843	2829.92	71.007	30.585

- 1280x720

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
Vidyo1	8241.44	115.698	43.461	3915.92	114.379	40.970	2151.36	103.385	38.519	1295.36	130.104	35.929
Vidyo3	10783.44	122.288	43.080	4858.40	131.081	40.670	2497.92	132.015	37.916	1425.52	214.185	35.065
Vidyo4	8703.76	137.762	43.571	3750.72	133.312	40.967	1892.80	144.650	38.436	1058.24	119.400	35.976

- 1920x1088

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
BQTerrace	127829.52	496.040	41.176	53627.68	398.973	36.892	21565.52	409.173	33.490	10096.16	386.367	30.554
Cactus	100572.40	414.468	40.106	39310.96	320.692	37.277	20683.76	329.020	34.994	11330.08	295.576	32.600
Kimono	48848.48	366.724	42.374	26254.96	337.365	40.646	14347.36	363.256	38.449	7595.84	356.913	35.911
ParkScene	68080.40	482.111	40.959	31509.92	421.877	38.139	15012.48	389.790	35.242	7098.16	349.710	32.545

Tableau B. 1 Simulations du HM 10 pour différents résolutions

- 416x240

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
BasketballPass	1666.080	19.877	42.3141	918.160	17.612	38.8385	504	15.759	35.7738	276.240	14.316	33.1062
BlowingBubbles	3028.480	26.803	39.6485	1602.880	23.208	36.4498	859.200	19.697	33.5389	449.920	20.906	30.9044
BQSquare	3755.712	23.532	40.2459	2107.872	18.804	36.5892	1242.048	16.628	33.6140	784.512	14.295	30.9462
RaceHorses	2140.896	44.356	39.9545	1181.808	39.054	36.2284	634.320	38.376	32.9070	346.464	32.069	30.1973

- 832x480

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR- Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR- Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR - Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR - Y (dB)
BasketballDrill	6042.080	114.850	42.0259	3141.12	112.908	38.9764	1652.24	101.253	36.201	892.320	96.236	33.771
PartyScene	14950.96	116.136	40.2966	8636.08	132.729	36.4008	4998.64	125.235	33.037	2778.24	124.123	29.940
BQMall	8302.272	82.744	40.8835	4674.91	69.574	38.4318	2738.21	86.626	35.714	1566.91	84.560	32.914
RaceHorses	7857.696	228.175	39.4319	3867.65	190.526	36.1903	2016.43	160.673	33.317	1051.15	173.249	30.829

- 1280x720

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
Vidyo1	6458.688	157.576	44.3849	3147.552	135.703	42.4098	1823.520	125.140	40.3515	1091.616	123.332	38.0228
Vidyo3	8030.112	174.374	43.7556	3473.088	145.750	41.7736	1968.384	132.767	39.5604	1178.976	128.220	37.1806
Vidyo4	6148.032	6148.032	44.4163	2946.432	148.408	42.4086	1633.056	144.523	40.3704	920.064	140.710	38.1307

- 1920x1088

	22			27			32			37		
	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)	Bit rate (kbit/s)	Encoding time (sec)	PSNR-Y (dB)
BQTerrace	69153.696	664.733	39.1542	26592.672	460.112	36.5396	12766.272	375.084	34.6400	6938.400	342.438	32.6324
Cactus	39374.880	532.220	39.3377	14730.960	391.628	37.6604	7580.560	353.422	35.8604	4081.360	334.149	33.8265
Kimono	8536.090	583.020	42.5784	4300.800	551.041	40.6463	2273.050	474.283	38.3347	1243.008	408.684	36.0528
ParkScene	15132.787	551.379	40.8770	7378.790	416.965	38.5524	3693.773	350.715	36.1730	1812.672	326.361	33.9042

Annexe C

Algorithme C.1 : Algorithme du codage arithmétique régulier

```
rangeIdx ← range (7 ! 6)
rangeLPS ←rangeLUT[pStateIdx[ctxIdx]][rangeIdx]
rangeMPS ←range − rangeLPS
Si (bin = valMPS[ctxIdx]) alors{
 range ← rangeMPS
 pStateIdx[ctxIdx] = pStateLUT[pStateIdx[ctxIdx]][0] }
 Sinon {
  range ← rangeLPS
  low ←low + rangeMPS}
   Si (pStateIdx[ctxIdx] = 0) alors{
    valMPS[ctxIdx] ← 1 − valMPS[ctxIdx] }
    pStateIdx[ctxIdx] ←pStateLUT[pStateIdx[ctxIdx]][1]}
   FIN Si
 Renorm(range, low, bitsOutstanding)
FIN Si
Fin
```

Algorithme C.2 : Algorithme du codage arithmétique de finalisation

```
range ←  range − 2
Si (bin = 1) alors {
  low   low + range
  range  ←  2
  Renorm(range, low, bitsOutstanding)
  Encodeflush(low) }
  Sinon
     Renorm(range, low, bitsOutstanding)
  FIN Si
Fin
```

Algorithme C.3 : Algorithme du codage arithmétique altérnatif

```
low ← low << 1
Si (bin = 1) alors
  low  ←  low + range
  FIN Si
Renorm bypass(range, low, bitsOutstanding)
Fin
```

Annexe D

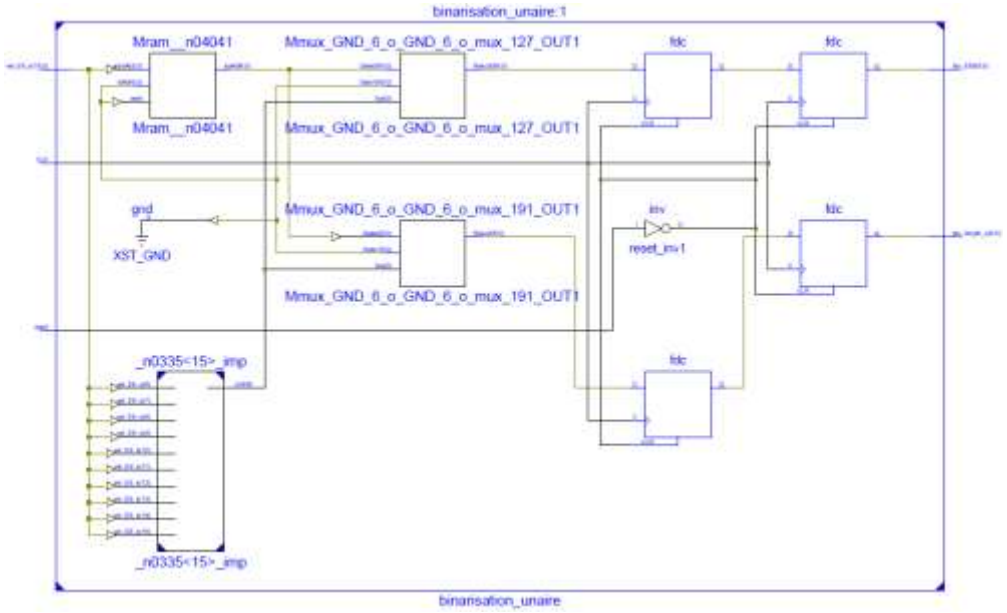


Figure D. 1. Architecture synthétisée du binarisateur unaire

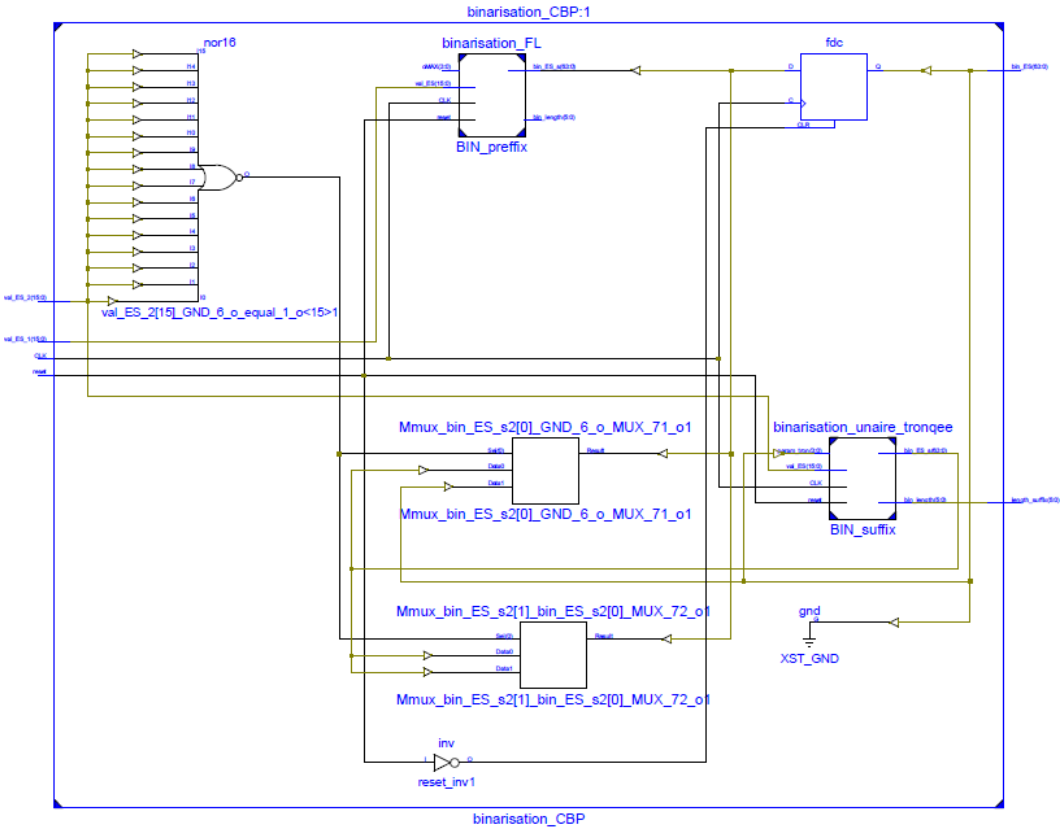


Figure D. 2. . Architecture synthétisée du binarisateur CBP

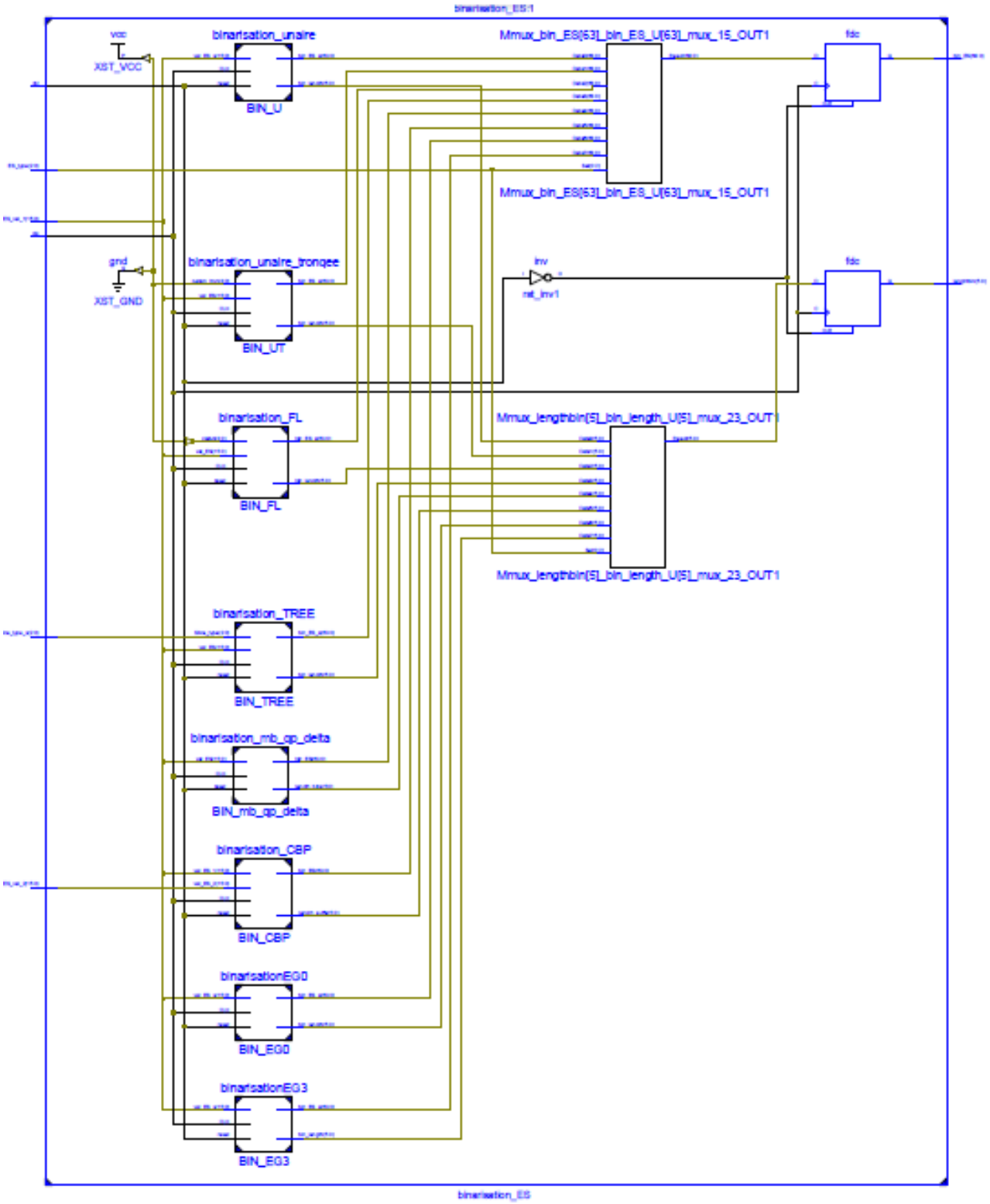


Figure D. 3. . Architecture synthétisée du binarisateur global

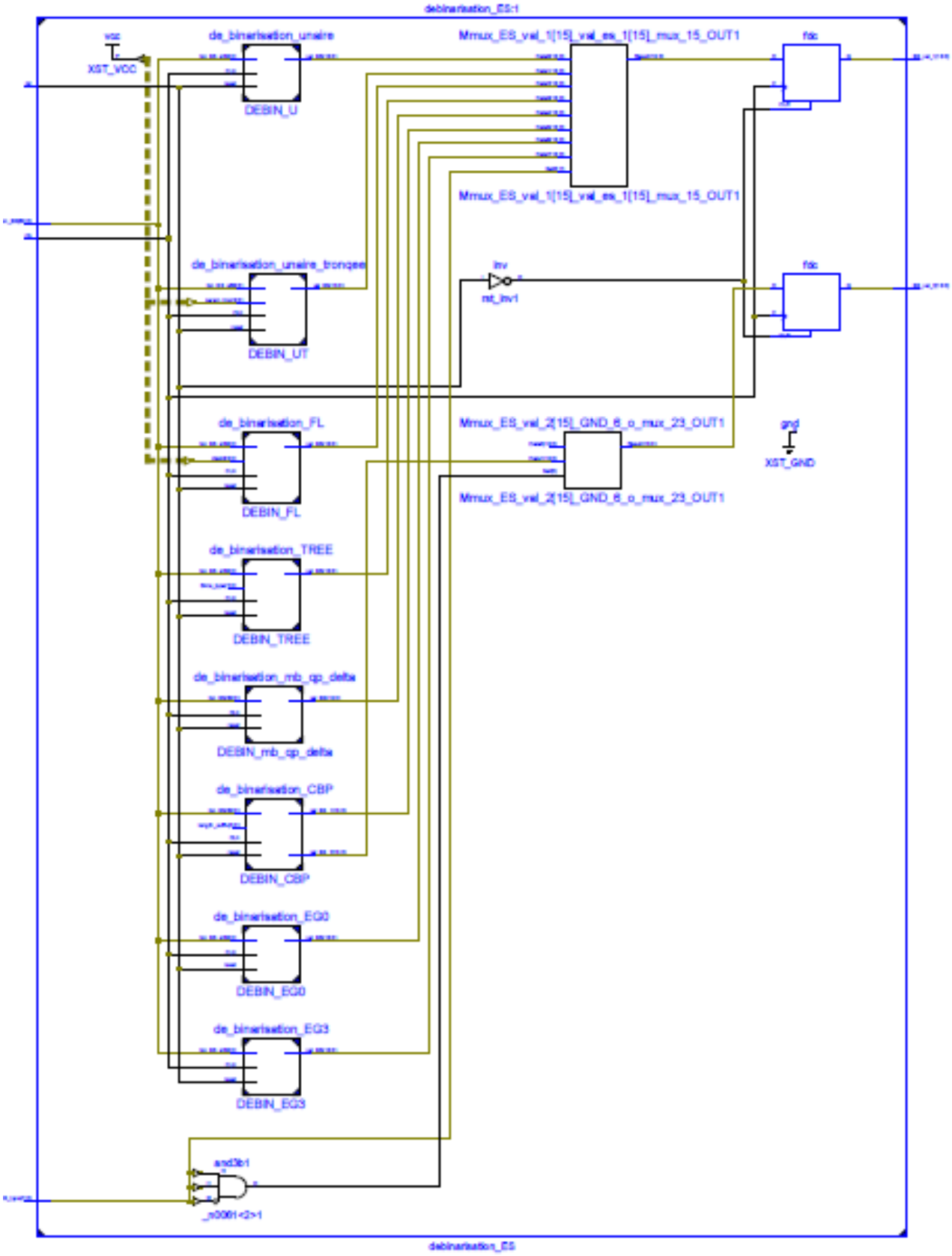


Figure D. 4. . Architecture synthétisée du débinarisateur global

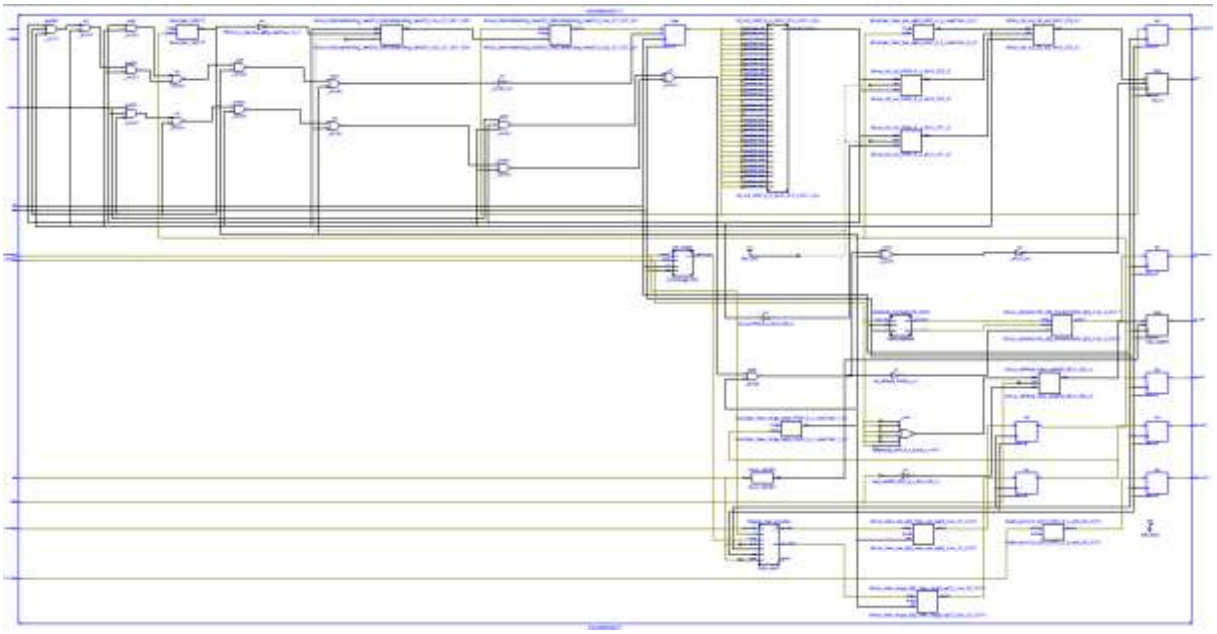


Figure D. 5. . Architecture synthétisée du BAC régulier

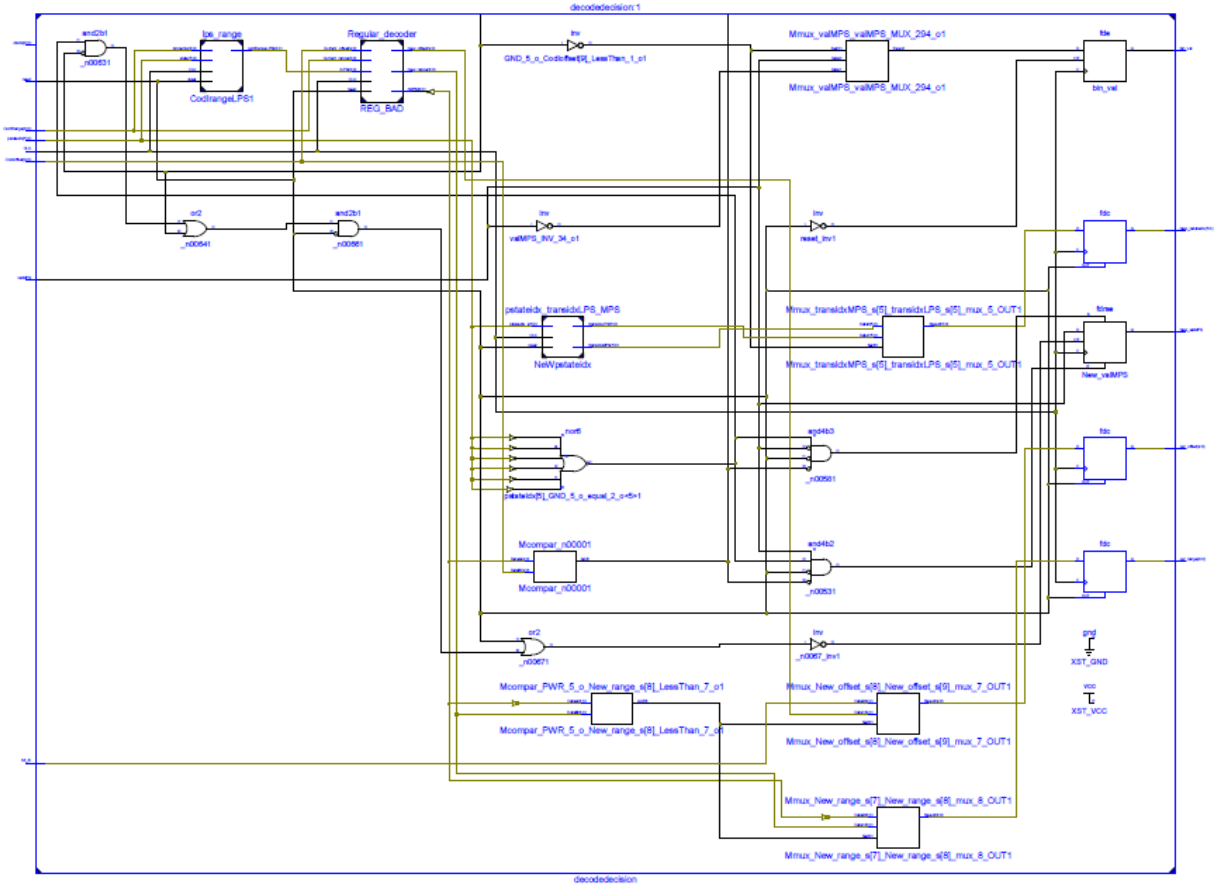


Figure D. 6. Architecture synthétisée du BAD régulier

Figure D. 7. Architecture synthétisée du BAC de finalisation

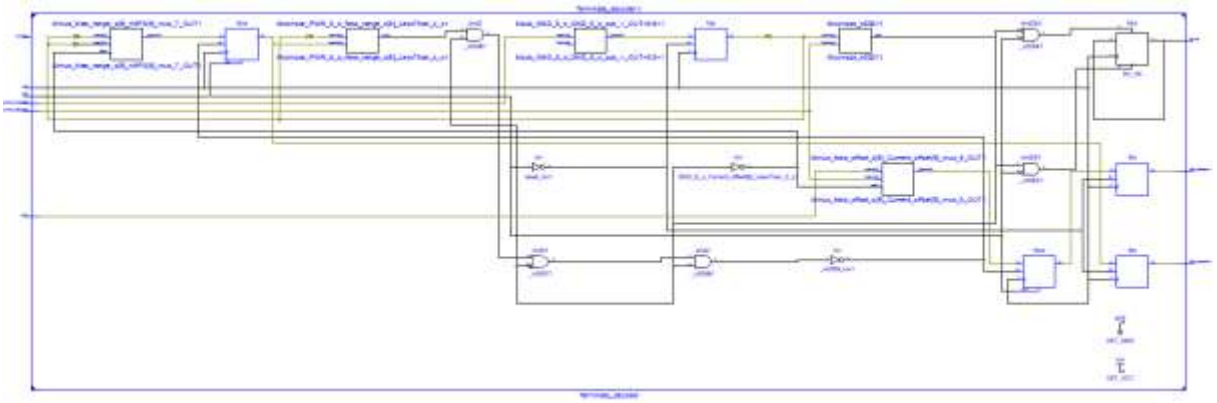


Figure D. 8. Architecture synthétisée du BAD de finalisation

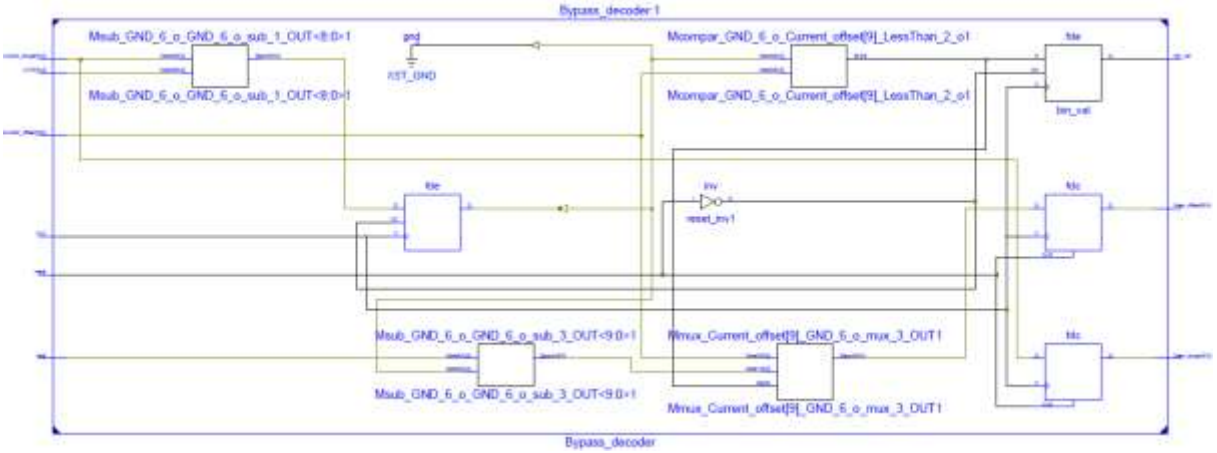


Figure D. 9. Architecture synthétisée du BAD altérnatif

Annexe E

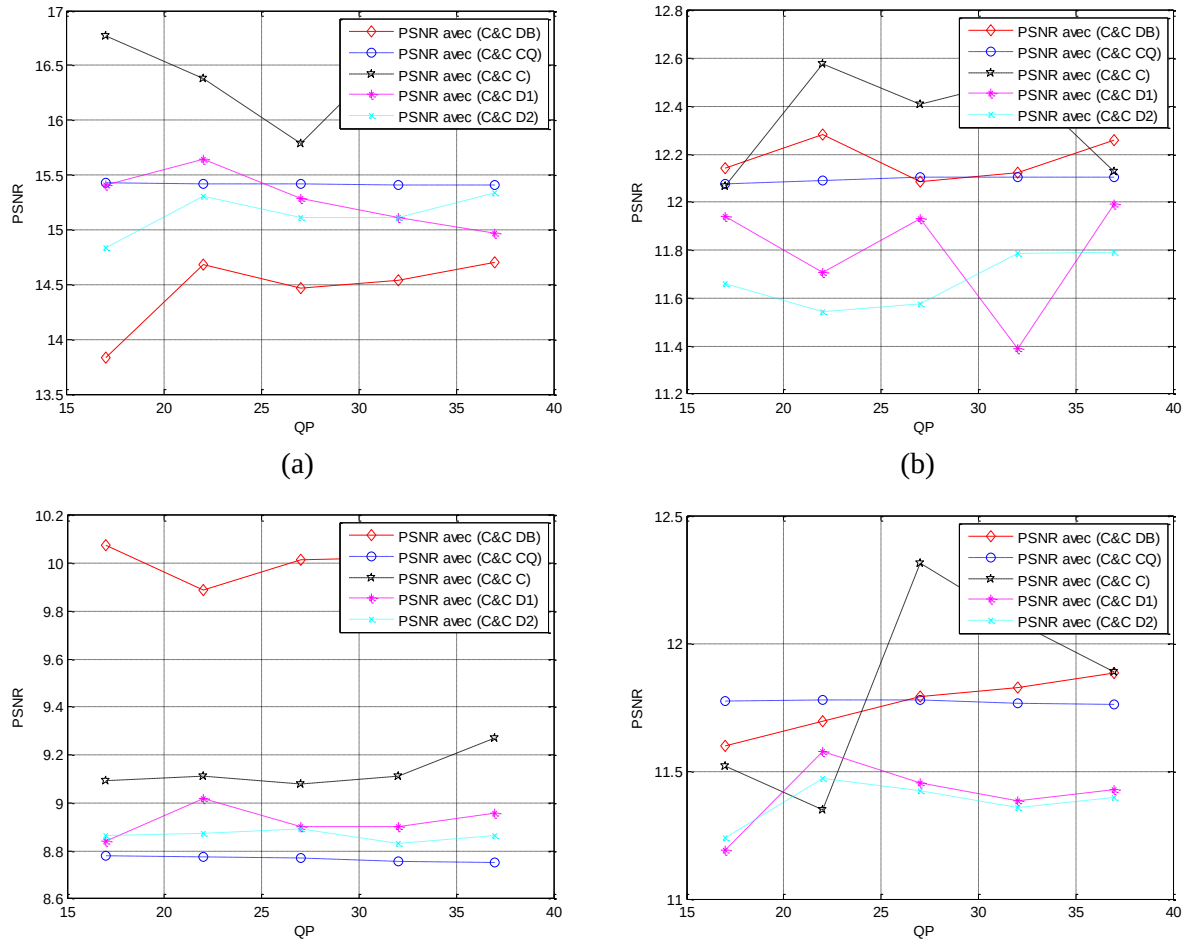
Algorithme E.1 : Algorithme de BAC

Entrées : U_0, L_0 : intervalles initiaux.
Scale_ E_3 : paramètre de condition E_3
Symbole : entrée binaire {0, 1} à coder
U= U_0 ;
L= L_0 ;
Avoir le symbole {0, 1}
L = L + F_{j-1} * (U-L+1)
U = l + F_j * (U-L+1)-1
Tant que (MSB de U et L sont égaux à B ou la condition E_3 est vérifiée) faire
si (MSB de U et L sont égaux à B) // B = {0.1}
Envoie de B // Mettre (0) ou (1)
Décaler L à gauche par 1 bit et mettre **0** dans LSB
Décaler U à gauche par 1 bit et mettre **1** dans LSB
Tant que (Scale_ E_3 > 0) alors
Envoie du complément de B
Décrémenter Scale_ E_3
Fin
Fin
si (la condition E_3 est vérifiée)
Décaler L à gauche par 1 bit et mettre **0** dans LSB
Décaler U à gauche par 1 bit et mettre **1** dans LSB
Inverser MSB de L et U
Incrémenter Scale_ E_3 // Accumuler Out-bit
Fin
Fin

Algorithme E.2 : Algorithme du RAC

Entrées : U_0, L_0 : intervalles initiaux.
RBN : nombre aléatoire
Scale_ E_3 : paramètre de condition E_3
Symbole : entrée binaire {0, 1} à coder
U= U_0 ;
L= L_0 ;
Avoir le symbole {0, 1}
L = L + F_{j-1} * (U-L+1)
U = l + F_j * (U-L+1)-1
Tant que (MSB de U et L sont égaux à B ou la condition E_3 est vérifiée) faire
si **(**RBN = 0**)**
MSB_L= complément de MSB_L
Fin
si (MSB de U et L sont égaux à B) // B = {0.1}
Envoie de B // Mettre (0) ou (1)
Décaler L à gauche par 1 bit et mettre **0** dans LSB
Décaler U à gauche par 1 bit et mettre **1** dans LSB

Tant que (Scale_ E_3 > 0) alors
Envoie du complément de B
Décrementer Scale_ E_3
Fin
Fin
si (la condition E_3 est vérifiée)
Décaler L à gauche par 1 bit et mettre **0** dans LSB
Décaler U à gauche par 1 bit et mettre **1** dans LSB
Inverser MSB de L et U
Incrémenter Scale_ E_3// Accumuler Out-bit
Fin
Fin



(a) (b)

(c) (c)

Figure E. 1. Variation de PSNR en fonction de QP pour les séquences SD (a) BasketballPass, (b) BlowingBubbles, (c) BQSquare et (d) RaceHorses pour les méthodes de compression et cryptage proposées

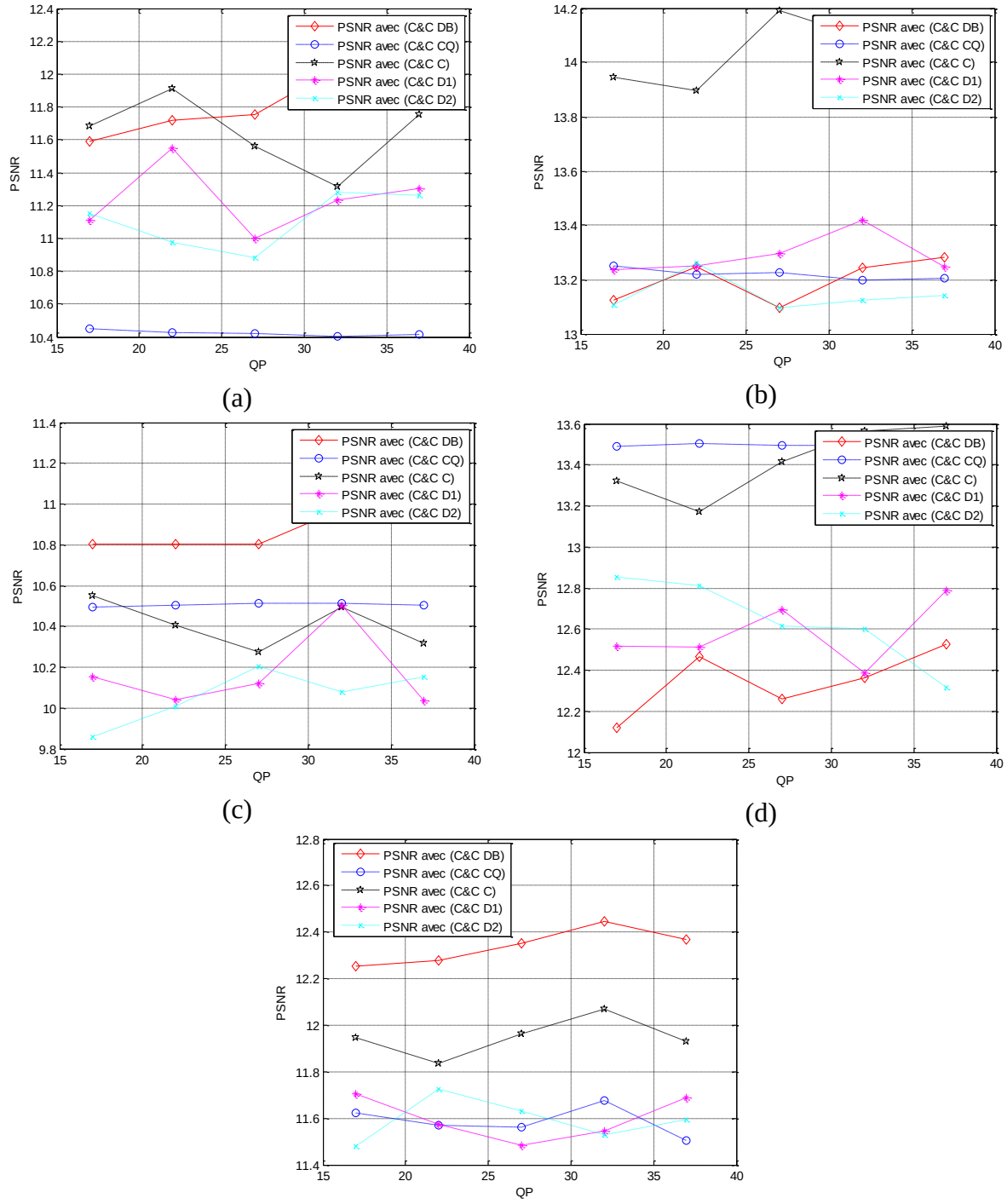


Figure E. 2. Variation de PSNR en fonction de QP pour les séquences QCIF (a) Forman, (b) Coastguard, (c) Carphone, (d) Suzie et (e) Mobile pour les méthodes de compression et cryptage proposées

Tableau E. 1. Variation de PSNR des séquences QCIF pour différents valeurs de QP

		37	32	27	22
Forman	(CSC)	23.2499	23.0478	21.1995	24.8656
	(C&C DB)	12.2268	12.0402	11.7554	11.7194
	(C&C CQ)	10.4172	10.4022	10.4205	10.4247
	(C&C C)	11.7564	11.3144	11.5589	11.9096
	(C&C D1)	11.3047	11.2309	10.9993	11.5483
	(C&C D2)	11.2622	11.2777	10.8808	10.9787
Coastguard	(CSC)	25.5036	25.1747	25.7230	26.6643
	(C&C DB)	13.2831	13.2450	13.0969	13.2465
	(C&C CQ)	13.2043	13.1984	13.2263	13.2200
	(C&C C)	14.1056	14.0945	14.1898	13.8971
	(C&C D1)	13.2467	13.4194	13.2973	13.2499
	(C&C D2)	13.1409	13.1234	13.0966	13.2611
Carphone	(CSC)	21.6633	21.9719	21.5739	22.2914
	(C&C DB)	11.0868	10.9751	10.8048	10.8022
	(C&C CQ)	10.5053	10.5124	10.5130	10.5043
	(C&C C)	10.3147	10.4933	10.2765	10.4070
	(C&C D1)	10.0385	10.5060	10.1190	10.0402
	(C&C D2)	10.1535	10.0762	10.2033	10.0102
Suzie	(CSC)	25.9847	25.2875	26.1856	26.3979
	(C&C DB)	12.5284	12.3628	12.2582	12.4678
	(C&C CQ)	13.5130	13.4959	13.4953	13.5023
	(C&C C)	13.5876	13.5649	13.4144	13.1709
	(C&C D1)	12.7896	12.3879	12.6970	12.5123
	(C&C D2)	12.3145	12.6017	12.6160	12.8123
Mobile	(CSC)	21.0837	20.9292	21.1995	21.7688
	(C&C DB)	12.3673	12.4445	12.3516	12.2762
	(C&C CQ)	11.5024	11.6781	11.5611	11.5708
	(C&C C)	11.9299	12.0711	11.9639	11.8376
	(C&C D1)	11.6906	11.5447	11.4828	11.5757
	(C&C D2)	11.5939	11.5276	11.6303	11.7257

Tableau E. 2. Variation de PSNR des séquences SD pour différents valeurs de QP

		37	32	27	22
BasketballPass	(CSC)	24.4849	25.0708	25.9012	25.0294
	(C&C DB)	14.6982	14.5388	14.4696	14.6835
	(C&C CQ)	15.4063	15.4094	15.4174	15.4210
	(C&C C)	16.4592	16.6777	15.7837	16.3803
	(C&C D1)	14.9726	15.1104	15.2844	15.6482
	(C&C D2)	15.3340	15.1118	15.1100	15.3106
g B	(CSC)	22.1214	21.8485	21.1667	22.6790

	(C&C DB)	12.2596	12.1237	12.0845	12.2826
	(C&C CQ)	12.1076	12.1048	12.1068	12.0915
	(C&C C)	12.1295	12.5100	12.4092	12.5772
	(C&C D1)	11.9949	11.3890	11.9320	11.7081
	(C&C D2)	11.7923	11.7886	11.5772	11.5420
BQSquare	(CSC)	15.7155	16.5312	16.7153	16.7795
	(C&C DB)	9.9861	10.0230	10.0164	9.8876
	(C&C CQ)	8.7527	8.7549	8.7711	8.7763
	(C&C C)	9.2715	9.1131	9.0786	9.1106
	(C&C D1)	8.9560	8.9030	8.9024	9.0184
	(C&C D2)	8.8662	8.8312	8.8906	8.8729
RaceHorse	(CSC)	21.0709	21.4804	21.8769	22.1303
	(C&C DB)	11.8818	11.8279	11.7915	11.6964
	(C&C CQ)	11.7613	11.7642	11.7779	11.7784
	(C&C C)	11.8896	12.0810	12.3150	11.3477
	(C&C D1)	11.4298	11.3848	11.4551	11.5783
	(C&C D2)	11.3972	11.3584	11.4213	11.4693

Tableau E. 3. Rapport de compression pour des séquences QCIF pour différents valeurs de QP

		37	32	27	22
Forman	(CSC)	82.2312	81.3807	80.6044	79.7631
	(C&C DB)	28.1558	26.8550	24.7795	22.5121
	(C&C CQ)	1.4714	1.4671	1.4547	1.4468
	(C&C C)	82.2312	81.3807	80.6044	79.7631
	(C&C D1)	28.1558	26.8550	24.7795	22.5121
	(C&C D2)	1.4714	1.4671	1.4547	1.4468
Coastguard	(CSC)	85.4465	85.3814	84.9533	83.8408
	(C&C DB)	25.3973	23.9943	22.1961	19.9845
	(C&C CQ)	1.5079	1.5166	1.5070	1.4822
	(C&C C)	85.4465	85.3814	84.9533	83.8408
	(C&C D1)	25.3973	23.9943	22.1961	19.9845
	(C&C D2)	1.5079	1.5166	1.5070	1.4822
Carphone	(CSC)	83.1727	82.7037	81.9008	80.8670
	(C&C DB)	29.1202	27.4604	25.4986	22.9599
	(C&C CQ)	1.5058	1.4989	1.4806	1.4717
	(C&C C)	83.1727	82.7037	81.9008	80.8670
	(C&C D1)	29.1202	27.4604	25.4986	22.9599
	(C&C D2)	1.5058	1.4989	1.4806	1.4717
Suzie	(CSC)	88.4015	88.2218	87.5848	86.8278
	(C&C DB)	23.6678	21.9329	20.3523	18.1064
	(C&C CQ)	1.5462	1.5370	1.5343	1.5237
	(C&C C)	88.4015	88.2218	87.5848	86.8278
	(C&C D1)	23.6678	21.9329	20.3523	18.1064
	(C&C D2)	1.5462	1.5370	1.5343	1.5237
Mobile	(CSC)	67.5886	66.0788	64.5144	62.6480
	(C&C DB)	27.1190	25.5001	23.6709	21.4469
	(C&C CQ)	1.2055	1.1788	1.1505	1.1147

	(C&C C)	67.5886	66.0788	64.5144	62.6480
	(C&C D1)	0.0468	25.5001	23.6709	21.4469
	(C&C D2)	1.2055	1.1788	1.1505	1.1147

Tableau E. 4. Rapport de compression pour des séquences SD pour différents valeurs de QP

		37	32	27	22
BasketballPass	(CSC)	87.9024	87.4191	86.6140	86.1748
	(C&C DB)	23.1899	21.6915	20.0025	17.8877
	(C&C CQ)	1.4352	1.4305	1.4232	1.4121
	(C&C C)	87.9024	87.4191	86.6140	86.1748
	(C&C D1)	23.1899	21.6915	20.0025	17.8877
	(C&C D2)	1.4352	1.4305	1.4232	1.4121
BlowingBubbles	(CSC)	85.0715	84.2231	83.4011	82.5538
	(C&C DB)	27.0979	25.5151	23.5333	21.2357
	(C&C CQ)	1.4128	1.3946	1.3924	1.3747
	(C&C C)	85.0715	84.2231	83.4011	82.5538
	(C&C D1)	27.0979	25.5151	23.5333	21.2357
	(C&C D2)	1.4128	1.3946	1.3924	1.3747
BQSquare	(CSC)	74.0329	73.0644	71.9957	70.7151
	(C&C DB)	30.5024	28.8154	26.7595	24.1551
	(C&C CQ)	1.2778	1.2619	1.2460	1.2287
	(C&C C)	74.0329	73.0644	71.9957	70.7151
	(C&C D1)	30.5024	28.8154	20.0025	24.1551
	(C&C D2)	1.2778	1.2619	1.2460	1.2287
RaceHorses	(CSC)	83.6379	83.2779	82.5530	81.5138
	(C&C DB)	24.6990	23.1518	21.3258	19.1611
	(C&C CQ)	1.3893	1.3807	1.3660	1.3522
	(C&C C)	83.6379	83.2779	82.5530	81.5138
	(C&C D1)	24.6990	23.1518	21.3258	19.1611
	(C&C D2)	1.3893	1.3807	1.3660	1.3522