Architecture auto-adaptative pour le transcodage vidéo

Michael Guarisco

Nancy-Université
Université Henri Poincaré

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

➢ Contact SCD Nancy 1 : theses.sciences@scd.uhp-nancy.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Nancy-Université
Université Henri Poincaré

U.F.R. Sciences et Techniques, Mathématiques, Informatiques, Automatique
École Doctorale Informatique Automatique Electronique Electrotechnique Mathématiques
Département de formation Doctorale Électronique Électrotechnique

Architecture auto-adaptative pour le transcodage vidéo

THÈSE

présentée et soutenue publiquement le 14/11/2011

pour l'obtention du

Doctorat de Nancy Université – Nancy 1

(spécialité Systèmes Électroniques)

par

Michael GUARISCO

Composition du jury

Président :	Guy GOGNIAT	Professeur, Lab-STICC de Lorient
Rapporteurs :	Christophe JEGO	Professeur, IMS de Bordeaux
	Guy GOGNIAT	Professeur, Lab-STICC de Lorient
Examinateurs :	Serge WEBER	Professeur, UHP (Directeur de thèse)
	Hassan RABAH	Professeur, UHP (Co-directeur de thèse)
	El-Bay BOURENNANE	Professeur, Université de Bourgogne

Laboratoire d'Instrumentation Électronique de Nancy
Faculté des Sciences - 54506 Vandoeuvre-lès-Nancy

Remerciements

J'adresse mes remerciements sincères à Monsieur Christophe Jégo, Professeur à l'université de Bordeaux ainsi qu'à Monsieur Guy Gogniat, Professeur à l'université de Bretagne-Sud, qui m'ont fait l'honneur d'accepter de juger cette thèse en qualité de rapporteurs.

Je remercie Monsieur El-Bay Bourennane, Professeur à l'université de Bourgogne pour avoir accepté de participer au jury.

Je remercie chaleureusement Monsieur Serge Weber de m'avoir accueilli dans son laboratoire et en qualité de co-directeur de thèse pour m'avoir accordé sa confiance et m'avoir soutenu pendant ces années de thèse. Pour les même raisons et également en qualité de co-directeur de thèse, je remercie Monsieur Hassan Rabah pour les conseils qu'ils a pu m'apporter durant l'encadrement de mes travaux.

Je tiens à remercier aussi Monsieur Yves Berviller pour m'avoir guidé dans mes premières années d'enseignement.

Bien entendu, ces remerciements ne seraient pas complets si je ne remerciais pas mes collègues et membres du LIEN, en pensant notamment à Patrice Roth dont l'humour restera dans les annales, Pierre Schmitt et Christine Daudens pour ses conseils administratifs.

Je remercie avant tout mes parents, Christine et Patrice, sans qui je n'en serai pas là. Merci à mes amis pour le stress occasionné par des phrases clés comme "alors cette thèse ? ça avance ?" ou encore "après trois années, je n'ai toujours pas compris ce que tu faisais".

Je remercie, pour finir, Aurélie pour son soutien.

Michael Guarisco

Septembre 2011

Table des matières

Table des figures

Liste des tableaux

Introduction générale

1 Contexte général

1.1 historique

Grâce aux progrès de l'électronique, de l'informatique et du transport de média, les technologies de l'information ont pris une grande place dans notre quotidien. Depuis les années 90, les systèmes de visualisation de contenu multimédia audiovisuel n'ont cessé de se diversifier. Les moyens de transport de ces informations se sont aussi largement étendus. Aujourd'hui, l'ADSL avec tous ces moyens de diffusion filaire ou non-filaire, la 3G, amènent une grande variété de canaux de transmission avec des caractéristiques plus ou moins variables selon les cas. Il existe donc de nombreux terminaux de type portable (smartphone, ordinateur portable voire ultra-portable ...) ou fixe (Téléviseur, ordinateur de bureau, ...). Cette multitude de terminaux sont connectés entre eux ou à un serveur distant par le biais de réseaux très hétérogènes. Ils sont également pourvus de capacités plus ou moins importantes en termes de puissance de calcul, résolution d'affichage, autonomie .... La demande et surtout l'hétérogénéité en termes de multimédia ne cesse donc de croitre avec les nouvelles technologies.

C'est sans compter sur l'évolution mais surtout la coexistence de nombreuses méthodes de compression des contenus audiovisuels. En effet, un panorama rapide des standards de compression permet de se rendre compte de la cohabitation de standards anciens et récents (toutes les versions de MPEG, de MPEG-1 à 4 en passant par les différentes versions au sein même d'une de ces appellations) ainsi que de standards concurrentiels (WMV ou VC-1 et MPEG par exemple).

L'encodage multimédia est une problématique suffisamment large pour que de nombreux domaines soient concernés. En effet, la compression vidéo peut tout aussi bien toucher le monde informatique avec des solutions d'encodage (ou décodage) logicielle, que le monde électronique avec des solutions matérielles de type ASIC ou FPGA. Dans le monde du logiciel, il existe des solutions qui utilisent de manière très classique les processeurs des ordinateurs alors que d'autres utilisent des processeurs plus spécifiques qu'on retrouvent dans les routeurs et autres matériels conçus pour gérer les canaux de transmission de l'information qu'on appelle les processeurs réseau.

Les ASICs sont des circuits très spécialisés apportant une intégration maximale des technologies les plus optimales. Ces types de circuits possèdent malheureusement une flexibilité très faible et le coût de fabrication est relativement élevé. Les FPGAs permettent quant à eux d'apporter une flexibilité bien meilleure grâce à leur possible reconfiguration (qu'elle soit partielle ou totale). Ces circuits sont de plus faciles à mettre à oeuvre lors de la conception et de l'implémentation de leur fonctionnalité. Le prototypage est en effet d'une grande facilité sur ce genre de circuits puisqu'il est possible de les configurer et reconfigurer (quasiment) sans aucune limite. Du fait de leur construction et de leur généricité, les FPGAs ne peuvent toutefois pas être aussi performants sur une fonctionnalité donnée qu'un ASIC conçu expressément pour cette fonction. Plus récemment, un nouveau type d'architecture est utilisée pour du calcul rapide. Il s'agit de l'utilisation des GPU (Graphical Processor Unit) qui prennent place sur les cartes graphiques de nos ordinateurs conventionnels. Leur architecture fortement dédiée au traitement d'image généralement en trois dimensions permettent nativement un traitement de l'image en deux dimensions optimales. Le processeur classique fait alors office de contrôleur de ce nouveau type de ressources.

1.2 Travaux actuels sur le sujet

Les avancées technologiques dans le domaine des équipements mobiles ont conduit à une très grande variétés de terminaux de visualisation, de la mobilité extrême avec des faibles résolutions, des puissances de calcul réduites comme dans les smartphones aux plus hautes résolutions avec les écrans Haute Définition (HD) des téléviseurs ou écran d'ordinateurs personnels possédant des capacités de calcul élevés. La majorité de ces terminaux sont de nos jours reliés au réseau internet et peuvent utiliser des services de vidéo à la demande ou des services de streaming. Mais fournir une diffusion de qualité pour tous ces périphériques aux caractéristiques si différents est un réel challenge. De plus, les réseaux de transport de l'information sont également très diversifiés, ce qui contribue à la complexité de la diffusion de contenu audiovisuel. Du point de vue des terminaux, il existe effectivement des terminaux pouvant décoder de

multiples standards de compression alors que d'autres sont plus restreint (car vieillissant ou limités techniquement). Du côté du réseau de transmission, la bande passante utilisable peut varier énormément d'un canal à un autre. Ces deux types d'hétérogénéités combinées présentent une multitude de cas d'usage. La communication entre deux appareils (un serveur et un terminal) devient de plus en plus complexe. C'est alors que l'adaptation du contenu vidéo entre les deux dispositifs devient réellement nécessaire. De plus, il existent un nombre important de standards circulant sur les réseaux de télécommunication modernes tels qu'Internet ou le réseau 3G (*UMTS : Universal Mobile Telecommunication System*). Dans cette partie sont dans un premier temps exposées brièvement les problématiques de transmission de contenus audiovisuels ainsi que les méthodes de compression usuelles de ces contenus. Puis, un passage en revue des techniques de base de transcodage est effectué. Enfin, les solutions concrètes de mise en oeuvre de ces techniques seront explorées.

2 Motivation

Les FPGAs, par rapport aux ASICs, apportent donc, certes au détriment de performances moindres, une flexibilité accrue. Même si le principe de reconfiguration dynamique sur FPGA est une problématique ancienne [EV62], elle reste tout à fait d'actualité face à des fonctionnalités toujours plus étendues et des demandes en puissance de calcul, en réduction de consommation et en flexibilité également toujours plus importantes. L'adaptation à l'environnement est aussi un des intérêts de la reconfiguration dynamique offert par les FPGAs.

Comme nous l'avons vu, la triple hétérogénéité entre les terminaux de visualisation, les réseaux de transport de l'information ainsi que celle des standards de compression utilisés forment un maillage duquel découle de nombreuses possibilités de couplage. Lors de la diffusion d'un contenu multimédia sur un réseau vers des terminaux divers, une adaptation est souvent nécessaire. Le transcodage est une des solutions qui permet une adaptation de ce contenu à un type de terminaux et un type de réseau de communication donnés. Cependant, au vu de la complexité grandissante

des standards de compression et de la versatilité de ceux-ci, les architectures de transcodage sont le plus souvent dédiées à un type de transcodage, qu'il soit homogène (le contenu vidéo reste sous le même standard mais certains paramètres comme le débit binaire, la résolution ou le nombre d'images par seconde sont modifiés) ou hétérogène (le standard de compression est modifié). Une architecture sur circuit est alors uniquement réservée à un type de transcodage, un aspect qui ne satisfera qu'un nombre limité de cas d'usage. Le calcul reconfigurable prend tout son sens ici puisqu'il devient alors théoriquement possible de modifiée l'architecture, ou du moins sa ou ses fonctionnalité pour qu'elle s'adapte au dit cas d'usage.

Un second aspect moderne vient également motiver cette étude et les choix qui ont été faits dans son cadre. En effet, les extensions scalables naissantes et plus particulièrement celle du standard vidéo H.264/AVC (pour Advance Video Coding) appelée SVC (Scalable Video Coding) sont extrêmement bien adaptée aux problématiques d'adaptation de contenu vidéo et aux terminaux de visualisation de plus en plus hétérogènes proposés sur le marché (de la télévision HD 1080 à la résolution VGA - voire inférieure - de certains smartphones ou autres terminaux mobiles). Malgré cela, le transcodage reste une étape souvent nécessaire dans de nombreux cas. Ce transcodage est particulièrement gourmand en termes de calculs qui plus est avec les nouvelles méthodes d'encodage par couche proposées par SVC qui permettent un choix quasi infini dans l'agencement des différentes couches et qui demandent donc un transcodage très adaptatif selon les besoins. Ce mémoire de thèse présente une architecture de transcodage auto-adaptative permettant le transcodage de

3 Contribution

L'objectif du travail de recherche présenté dans ce manuscrit est d'apporter une solution architecturale innovante basée sur la reconfiguration partielle et dynamique qu'offrent certains FPGAs. Plus précisément, il s'agit dans ces travaux de définir une architecture reconfigurable comportant un squelette de base et pouvant adapter ses fonctionnalités selon des besoins de base (type de transcodage, mode de transmission,

...) et également s'adapter aux conditions changeante de l'environnement de lequel l'architecture se situe.

Ces travaux ont été menés dans le cadre du projet ANR TOSCANE (Transmission vidéo Optimisée Source/CAnal écheloNnablE) qui visait le développement d'un système de diffusion s'appuyant sur une technique de codage dite codage conjoint de source et de canal pour améliorer la couverture globale de diffusion des contenus audiovisuels.

Ainsi, ces travaux ont permis de mettre au point une première architecture destinée au transcodage de flux vidéo encodés grâce au standard H.264/AVC ou SVC. L'objectif final étant toutefois de réaliser une architecture de base permettant de multiple transcodage, le développement d'une seconde architecture reconfigurable a permis de poser les fondements d'un cadre de travail pour le transcodage de divers sources et dans différents scénarios. Les architectures ont pu être validées par simulation grâce à différents outils de conception et vérification mais elles ont également été implémentées et testées sur des plateformes de prototypage FPGAs propre à Xilinx. Les travaux de cette thèse sont originaux dans leur approche puisque actuellement, peu de recherche existe quant à l'utilisation d'architectures reconfigurables dynamiquement dans le transcodage vidéo et encore moins en ce qui concerne l'utilisation de ces architectures dans le traitement de flux vidéos codés sous forme scalable.

4 Plan du mémoire

Le mémoire de thèse est scindé en deux parties distinctes : la première dresse un état de l'art des problématique de transmission vidéo et des solutions qui existent en terme de transcodage. Est abordée dans un premier temps la transmission vidéo, un chapitre qui mettra en évidence les problématiques de l'envoi sur différent réseaux de différents standards de compression sur différents terminaux. Puis, dans un second chapitre, les méthodes de transcodage à notre disposition seront passées en revue et discutées. Le choix d'un type de transcodage plutôt qu'un autre dans la suite des travaux sera également justifié grâce à cette partie de l'étude. Enfin, dans le troisième et

dernier chapitre de cette première partie, les moyens de mise en oeuvre de ces méthodes de transcodage seront analysés. Seront discutées les architectures matérielles de type ASIC ou FPGAs issues de la littérature ainsi que les solutions logicielles utilisant des processeurs conventionnels issus du monde informatique, mais également les architectures récentes à base de processeurs graphiques (GPU) et les solutions utilisant des processeurs réseau.

La seconde partie de ce manuscrit est orientée sur les contributions aux architectures reconfigurables de transcodage apportées par ce travail de thèse. Le premier chapitre est dédié à la première architecture que la précédente partie de l'introduction sur les contributions présente brièvement, l'architecture de transcodage dédiée au standard H.264/AVC et à son extension scalable SVC. La reconfiguration du circuit est alors utilisée ici au niveau codec (à chaque reconfiguration, c'est au moins une grande partie des fonctionnalités du standard qui sont mises à jour), contrairement au chapitre suivant qui détaillera une évolution de l'architecture où la reconfiguration dynamique est maintenant utilisée au niveau fonctionnel (la reconfiguration est plus fine, et peut se faire indépendamment sur chaque fonction du transcodeur). Un dernier chapitre tentera de généraliser le concept d'architecture reconfigurable pour le transcodage à une plus large gamme de codec et de cas d'utilisation. Pour cela un cadre est défini pour obtenir une méthode généraliste permettant la création autonome d'un transcodeur quelconque à partir d'une batterie de fonctions prédéfinies.

La dernière partie de ce mémoire dresse un bilan du travail de thèse ainsi que des nombreuses perspectives ouvertes par ces travaux.

Chapitre 1

La transmission vidéo

Sommaire

1.1 Introduction

1.1.1 Objectifs de la transmission

Depuis l'avènement des télécommunications, la tendance est à une dématérialisation des contenus multimédias au profit d'une centralisation de ces contenus et d'un accès à la demande pour l'utilisateur. Grâce aux récentes techniques de transmission et notamment l'ADSL, les capacités de bande passante ne cesse de s'accroitre et offrent la possibilité d'obtenir en temps réel des contenus audiovisuels dont la qualité et la résolution augmentent avec l'évolution des techniques de codage. Les terminaux de visualisation dont les capacités en termes de résolution sont aussi en constant progrès influent largement sur les besoins en codages et transmissions efficaces.

1.1.2 Les codecs

Les codecs (compression des termes CODeur et DECodeur) représentent un ensemble de moyens de compression de l'image et du son. Certains de ces moyens sont destructifs alors que d'autres ne le sont pas. Ces techniques mises bout à bout permettent de représenter de manière compacte le flux vidéo brute en (1) éliminant les redondances classiques d'une séquence d'images (spatiales et temporelles) et en (2) éliminant des informations peu discernables pour l'oeil humain. Le premier codec de compression numérique est le standard H.120 développé par le COST 211 en 1984. Peu efficace, ce standard servira de base au H.261, une recommandation de l'UIT-T (Union internationale des télécommunications), puis au standard MPEG-1 issu du groupe de travail MPEG (*Moving Picture Experts Group*. Ces deux derniers donneront naissance par la suite aux standards H.262, H.263, MPEG-2, MPEG-4, jusqu'à fusionner pour devenir H.264 - MPEG-4 part 10, dernier standard des plus aboutis. Alors que beaucoup de ces standards sont présents sur les réseaux de communication, passer de l'un à l'autre peut devenir une nécessité pour obtenir une meilleure efficacité de codage ou offrir une rétrocompatibilité pour un terminal ne disposant pas de la possibilité de décoder des standards récents. Ce passage est appelé transcodage. On distingue deux

types de transcodage : le transcodage hétérogène et le transcodage homogène. Dans le premier cas, un flux vidéo encodé dans un standard est totalement transposé dans un autre standard. Dans le second cas, le flux vidéo en entrée et en sortie du transcodeur sera encodé dans le même standard, mai sces caractéristiques auront été modifiées. Il peut s'agir d'une réduction de la qualité, de la résolution spatiale ou de la résolution temporelle.

1.1.3 Les canaux de transmission

L'optimisation au niveau de la transmission sur le canal ne fait pas partie des objectifs des travaux présentés ici. Mais quelques notions de transmission canal permettent de mieux appréhender la problématique du sujet. On peut distinguer d'abord deux types de transmission, la transmission filaire et la transmission sans fil. Dans les transmissions filaire : canal téléphonique (avec raccordement analogique ou RNIS), câbles coaxiaux, fibre optique ... mais on retrouve notamment les technologies DSL (*Digital Subscriber Line*) qui permettent l'utilisation de la classique paire torsadée téléphonique pour transmettre des informations à haut débit. Aujourd'hui, l'ADSL domine ce milieu (A pour *asymmetric*, signifiant que le débit montant et descendant sont différents). Du côté des transmissions sans fil, on peut citer le wifi (*wireless fidelity*), la téléphonie mobile à partir de la création du réseau 3G (3ème Génération ou UMTS pour *Universal Mobile Telecommunication System*) permettant de transmettre toute sorte de contenus. Tous ces réseaux de transport de l'information possèdent des caractéristiques propres et parfois variables (essentiellement pour les réseaux sans fil) et ne permettent pas tous de transmettre en temps réel des contenus audiovisuels encodés en haute qualité et haute résolution. Le transcodage devient là encore une nécessité.

1.2 Concepts généraux de compression des signaux audiovisuels

1.2.1 Objectifs de la compression vidéo

Depuis la numérisation des signaux vidéo, on a eu de cesse de vouloir optimiser son stockage et surtout sa transmission. Pour ce faire, des techniques de compression s'appuyant sur des constatations visuelles ont été mises au point. Le codage et le décodage du signal vidéo sont regroupés et connus sous le nom de codec. Tout principe de compression s'appuie sur la faculté de certaines transformations du signal à éliminer les redondances d'informations ou à les regrouper efficacement. On distingue deux grands types de compression : les compressions sans perte et les compressions avec pertes. La compression sans perte ne fait que regrouper les redondances afin de limiter leur impacte sur la taille du flux vidéo. Mais le poids de ces fichiers compressés sans perte reste relativement important et la compression avec perte est souvent inévitable. Ce type de compression permet l'élimination sélective d'informations. Pour résumer et simplifier, il s'agit de tirer parti des défauts de l'oeil humain qui est plus sensible à certains types d'information qu'à d'autres. Les standards MPEGs sont de bons exemples de compression avec perte ou des taux de compression de l'ordre de 2% sont atteignables par rapport au stockage d'une séquence vidéo donnée en brute. De récents standards tels que MPEG-4 AVC permettent une compression encore deux fois plus efficace pour la même qualité d'image.

1.2.2 Echantillonage

Une image brute est représentée par une succession de pixels. Un pixel couleur est typiquement formé de trois composantes couleurs : le rouge, le vert et le bleu. Pour chacune de ces composantes, on utilise généralement une représentation sur 8 bits de la valeur de la composante, permettant ainsi d'atteindre plus de 16 millions de nuances de couleur.

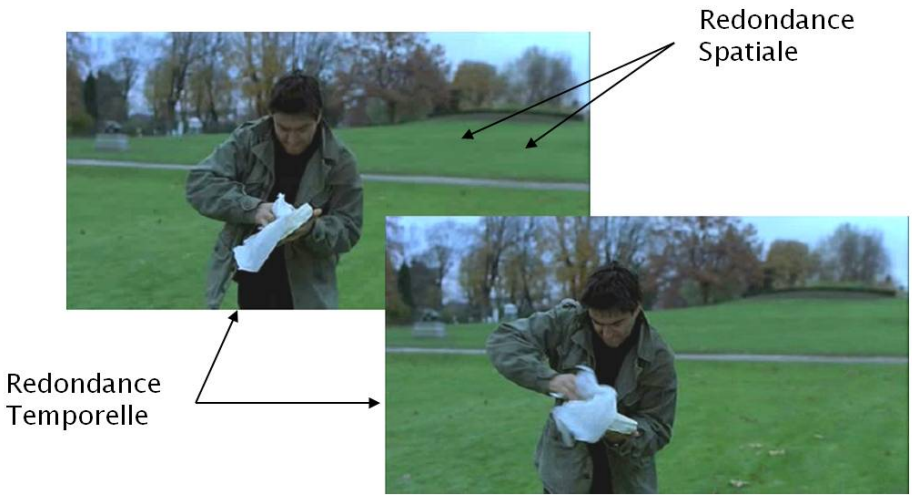


FIGURE 1.1 – Les deux types de redondances visuelles d'une séquence vidéo

1.2.2.1 Espace colorimétrique et échantillonage

En introduction, nous avons parlé de représentation RVB (rouge, vert, bleu) car c'est la représentation utilisée pour la majorité des types d'affichage, mais il en existe d'autres (CMJN, TSV, ...) et notamment le YUV (Y pour la luminance du pixel, U pour la chrominance bleue et V pou rla chrominance rouge). En RVB, l'oeil est sensible aux trois couleurs de façon similaire et il n'est pas envisageable d'éliminer certaines informations d'une composante pour alléger le signal. C'est par contre une possibilité dans le format YUV puisque l'oeil est bien moins sensible à la couleur d'une image qu'à sa luminosité. C'est pourquoi il est possible dans ce format de réduire la résolution des composantes de couleur U et V. Normalement, dans ce mode de représentation un pixel est représenté à la fois par sa luminance, mais aussi par 3 composantes de couleur (rouge, vert, et bleu) et non pas seulement deux. Ces composantes transformées dépendent des composantes de bases RGB et s'accordent selon les équations 1.1. YUV permet historiquement une rétrocompatibilité avec les anciens équipement d'affichage noir et blanc ou seule la composante Y est utilisée.

$$Y = kr.R + kg.G + kb.B$$
$$C_b = B - Y$$
$$C_r = R - Y$$
$$C_g = G - Y$$

Ces quatre composantes étant totalement interdépendante, il est possible de n'en retenir que trois pour la transmission ou le stockage pou reconstruire la quatrième au décodage. On passe alors du système RGB au système YUV et vice et versa grâce à 1.1.

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = 0.564(B - Y)$$
$$Cr = 0.713(R - Y)$$
$$et$$
$$R = Y + 1.402Cr$$
$$G = Y - 0.344Cb - 0.714Cr$$
$$B = Y + 1.772Cb$$

Comme il est dit précédemment, l'oeil humain est plus sensible à la luminance qu'aux couleurs d'une image. On a su tirer parti de cette caractéristique en instaurant des standards d'échantillonnage qui ne contiennent pas autant d'information de chrominance que d'information de luminance.

On trouve donc ces formats parmi les plus usités : le 4 :4 :4 (figure 1.2), qui contient deux fois plus d'information de chrominance que de luminance (même quantité d'information pour la chrominance rouge, la chrominance bleue et la luminance) ; le 4 :2 :2 (figure 1.3) qui contient au total autant d'information de chrominance que de luminance ; et le 4 :2 :0 (figure 1.4) qui contient deux plus d'information de luminance que de luminance.

Ce dernier format est le plus utilisé dans les standards de compression. Par pixel, il contient donc une moyenne de 12 bits.

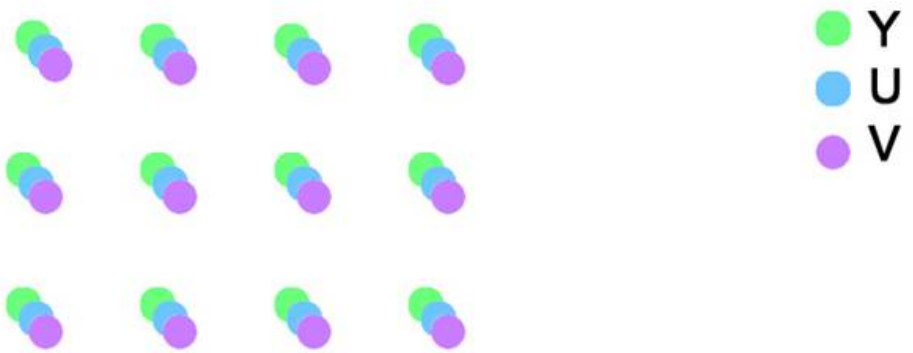


FIGURE 1.2 – Echantillonnage 4 :4 :4

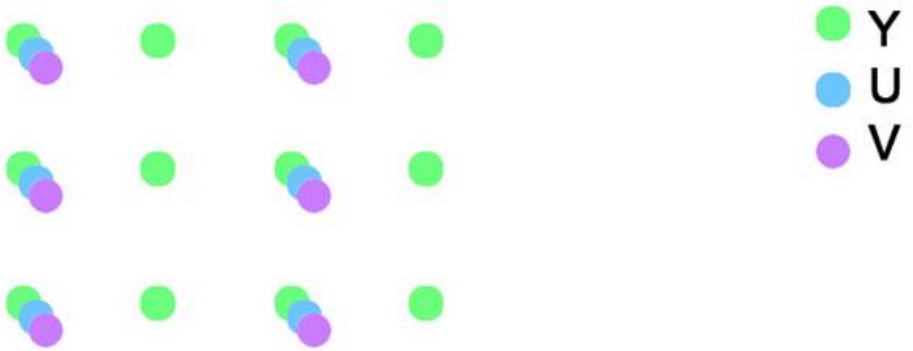


FIGURE 1.3 – Echantillonnage 4 :2 :2

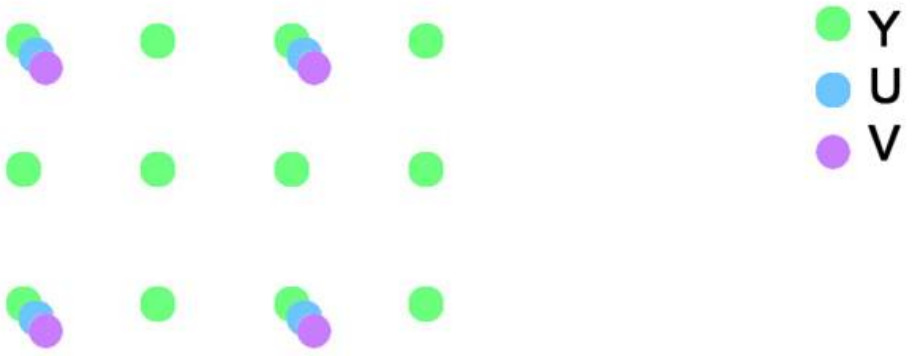


FIGURE 1.4 – Echantillonnage 4 :2 :0

1.2.2.2 Format

Certains standards, comme nous le verrons dans la suite de ce chapitre ne peuvent traiter qu'un nombre de formats limité. D'autre en revanche (les plus récents tels que H.264 AVC) sont conçus pour traiter tous les formats. En pratique, le format CIF7 est un regroupement de formats (tableau 1.1) standardisés. On peut ajouter à ce groupe les récents formats haute définition.

TABLE 1.1 – Formats standards d'affichage vidéo

Format	*Résolution de la luminance*	*Bit/image*	*Application typique*
SQCIF	128×96	147456	Vidéo mobile
QCIF	176×144	304128	Vidéo Conférence
CIF	352×288	1216512	Surveillance
4CIF	704×576	4866048	Télévision, DVD
720p	1280×720	11059200	Télévision HD
1080p	1920×1080	24883200	Stockage HD

Tous ces formats ne sont pas supportés par tous les standards de compression. H.261 ayant été conçu pour une transmission sur réseau RNIS ne supporte que de faibles résolutions CIF et QCIF. H.263 évolue dans ce sens en supportant en plus les résolutions SQCIF, 4CIF et 16CIF. Quant au H.264, dernier né de L'UIT, il supporte tout type de résolution.

1.2.3 Principe d'un encodeur-decodeur

Les codecs de compression vidéo fonctionnent tous sur les mêmes bases, tirant parti des redondances visuelles d'une séquence vidéo (figure 1.1). Le fonctionnement global d'un encodeur est présenté sur la figure 1.5

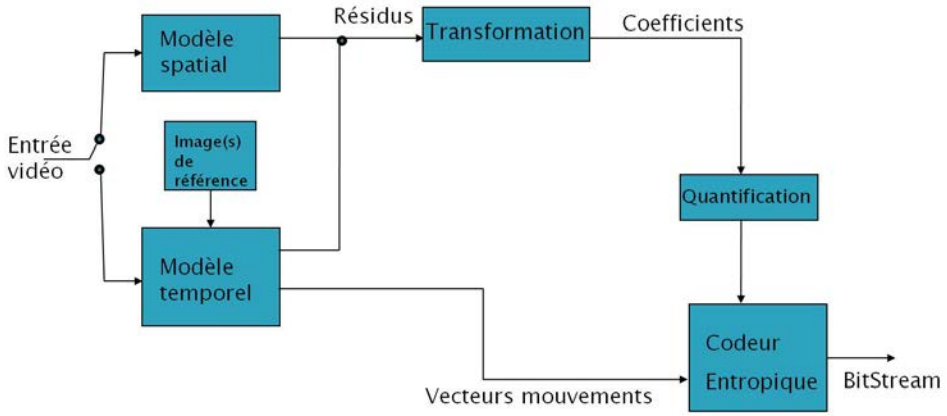


FIGURE 1.5 – Principe de la compression vidéo

L'encodeur se compose de trois étages : le modèle temporel, le modèle spatial et le codeur d'entropie. C'est donc l'addition de ces trois étages qui vont permettre d'exploiter les redondances. A la sortie du modèle temporel, on obtient des résidus et un champ de vecteurs de mouvement qui exprime la compensation du mouvement entre deux images successives. Les résidus sont la représentation des erreurs commises dans la prédiction du mouvement et sa compensation. Ils sont uniquement le résultat de la différence entre l'image reconstruite par prédiction et l'image d'origine. Ces mêmes résidus forment l'entrée du modèle spatial. Comme le montre la figure 1.1, il existe des redondances locales au sein même d'une images. Le modèle spatial utilise cette propriété et permet de les réduire. La transformation inclue dans ce modèle permet de décorréler les pixels en les passant dans un autre domaine (domaine transformé). On obtient alors des coefficients qui sont ensuite quantifiés afin d'éliminer les moins significatifs (qui sont alors remplacés par une valeur nulle). Plus le pas de quantification est élevé, plus le nombre de coefficients éliminés sera important. Les éléments issus des deux modèles sont alors encodés grâce au codeur entropique qui permet de compresser ces informations sans perte, uniquement en leur attribuant des mots-codes

judicieusement choisis.

Le décodeur quant à lui reconstruit la vidéo à partir du flux de sortie de l'encodeur. Les coefficients et les vecteurs de mouvement sont décodés par un décodeur d'entropie, à la suite duquel le modèle spatial se charge de reconstruire la trame résiduelle à partir des coefficients. Ensuite à partir d'une trame précédemment reconstruite, des vecteurs de mouvement et de la trame résiduelle, le décodeur obtient l'image de départ.

1.2.3.1 Modèle Temporel

Statistiquement, il est fort probable de trouver beaucoup de redondances entre deux images consécutives dans une séquence vidéo. Une faible proportion de pixel (ou de groupe de pixels) va être modifiée. On défini le résidu comme étant la différence entre deux trames successives. Un exemple est donné pour les trames des figure 1.6 et 1.7. La différence entre les deux trames est représentée en figure 1.8.

FIGURE 1.6 – Trame 1 (Images issues de [Ric03]

FIGURE 1.7 – Trame 2

On note clairement que dans les zones statiques, les résidus ont des valeurs proches de 0. Cependant les résidus qui correspondent aux zones mobiles de l'image possèdent une dynamique encore trop importante pour que l'encodage soit efficace. Un mécanisme de représentation du mouvement entre deux images a été mis au point. Il s'agit de créer un champ de vecteurs collant au mieux au mouvement qui permettra alors d'effectuer une prédiction de l'image suivante et permettra surtout d'abaisser fortement les valeurs des résidus. Or plus les résidus sont proches de zéro, plus l'encodage sera efficace. La figure 1.9 montre la représentation d'un champ de vecteur qui correspond à l'attribution d'un vecteur par bloc de 16x16 pixels. Ce champ de vecteur est obtenu après estimation du mouvement. Après compensation du mouvement (en utilisant l'image précédente et les vecteurs de mouvement), on obtient les résidus de la figure 1.10 qui sont plus proche de zéros que les résidus sans compensation de mouvement.

FIGURE 1.8 – Différence entre les trames

1.2.3.2 Modèle Spatial

Le modèle spatial correspond au second étage de l'encodeur. Il permet de réduire les redondances spatiales. C'est sur la trame résiduelle qu'est appliqué ce traitement. En regroupant les valeurs redondantes au sein d'une même zone, il sera plus aisé de trier les coefficients qui ont de l'importance de ceux qui en ont moins. Il existe différents outils permettant la transformation du domaine pixel dans le domaine coefficient.

1.2.3.2.1 La transformée en cosinus discret La transformée en cosinus discret (ou DCT) est reprise de la norme de compression des images fixes JPEG. Tout comme le fait une transformation de Fourier, la DCT permet de transposer les valeurs des pixels dans le domaine fréquentiel. Le domaine fréquentiel est choisi car les valeurs des pixels d'une image admettent une grande continuité. Ainsi, on distingue les basses fréquences qui représentent des "'aplats"' dans l'image, c'est à dire des zones dans lesquelles les valeurs des pixels sont homogènes et les hautes fréquences qui représentent

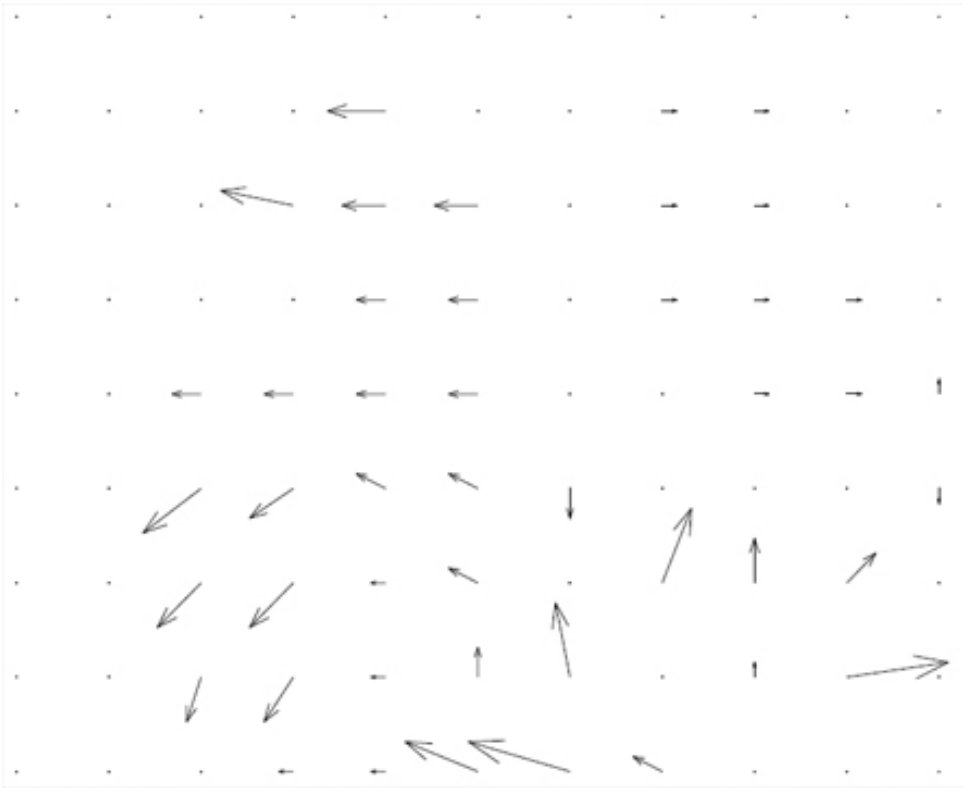


FIGURE 1.9 – Champ de vecteurs

des changements brutaux dans les valeurs de pixels voisins. Le hautes fréquences sont typiquement présentes lors de l'apparition de contours francs dans une image. Cette séparation des composantes hautes et basses fréquences de l'image permet un triage sélectif des informations utiles. En effet, l'oeil humain étant assez peu sensible aux hautes fréquences, celles-ci pourront être éventuellement éliminées pour gagner par la suite en taux de compression. La DCT se calcule sur une matrice carrée NxN. L'équation 1.1 donne la relation mathématique qui lient les valeurs des pixels de la matrice et celles des coefficients de sortie.

$$DCT(i,j) = \frac{1}{\sqrt{2}}C(i)C(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} pixel(x,i)\cos\left(\frac{(2x+1)i\pi}{2N}\right)\cos\left(\frac{(2y+1)j\pi}{2N}\right) \quad (1.1)$$

La reconstruction se fait par l'opération inverse mathématiquement décrite dans l'équation 1.2.

FIGURE 1.10 – Résidus avec compensation de mouvement

$$pixel(x,y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i)C(j)DCT(i,j) \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right) \tag{1.2}$$

Le choix de la DCT est basée sur une décroissance rapide de la valeur des coefficients (des basses fréquences vers les hautes fréquences et comparé à une transformée de Fourier), ce qui évite, lors de l'élimination des coefficients de hautes fréquences de perdre trop d'information, les plus importantes étant regroupées dans les basses fréquences. Classiquement, la DCT est réalisée sur des blocs (ou matrices) de 8x8 pixels ce qui peut entrainer, à fort taux de compression, un effet de bloc sur l'image.

En terme d'implémentation, la complexité d'une transposition directe des équations précédentes serait important pour une double boucle sur chaque pixel (coût $O(N^2)$). L'implémentation est donc simplifiée par l'utilisation d'une matrice C préétablie (équation 1.3). Ainsi, le coût des opérations nécessaires revient à $O(N)$.

$$C(i,j) = \begin{cases} \frac{1}{\sqrt{N}} & \texttt{si } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2j+1)i\pi)}{2N}\right) & \texttt{si } i > 0 \end{cases} \tag{1.3}$$

Le calcul des coefficients de la DCT revient ensuite à effectuer une multiplication matricielle entre C, la matrice des pixels et la transposée de C.

La transformée DCT a été utilisée dans quasiment tous les standards de compression jusqu'à l'arrivée de H.264 ou MPEG-4 part 10 qui utilise une transformée dérivée de la DCT dite transformée entière.

1.2.3.2.2 La transformée entière Cette transformée opérant sur des blocs 4x4 des résidus obtenus après différence entre prédiction (intra ou inter) et image courante originale, est basée sur la DCT mais en diffère par certains points :

1. Les calculs se font sur des entiers sans perte de précision.
2. La transformée inverse ne génère aucune erreur. Ce qui signifie que le résultat de la transformée direct suivi de la transformée inverse donnera exactement le bloc de départ, sans la moindre erreur.
3. Les opérateurs utilisés, contrairement à la DCT, restent simples. Seuls des additionneurs et des décaleurs sont nécessaires.
4. Les facteurs de mise à l'echelle sont intégrés dans la fonction de quantification, ce qui réduit le nombre de multiplications.

Ce type de transformée entière est utilisée dans le standard H.264.

1.2.3.3 Quantification

La transformation est sans perte. Sa fonction inverse permet théoriquement de retrouvé les valeurs des pixels exactes. La quantification, elle, permet d'éliminer les hautes fréquences déjà proches de la valeur nulle. Elle écarte donc les coefficients insignifiants et permet de réduire la dynamique des autres coefficients (les basses fréquences). Cette opération est réversible, mais elle engendre des pertes. La qualité de la vidéo ainsi que le débit binaire du flux compressé sont fortement corrélés à cette

fonction qui est paramétrable et éliminera plus ou moins de coefficients selon le paramètre choisi. Les transcodeurs permettant la réduction de débit binaire se servent essentiellement de cette fonction afin de modifier le flux en termes de débit binaire.

1.2.3.4 Encodeur d'entropie

L'encodeur d'entropie consiste en un encodage sans perte. Il doit représenter les informations de manière plus compactes selon des paramètres statistiques. En effet, plus un symbole (ou valeur) a de chance d'apparaitre dans les coefficients, et plus le mot-code que l'encodeur lui attribuera sera court. Au contraire un symbole peu fréquent se verra attribuer un mot-code plus long. Certains de ces codeurs sont dits adaptatifs (tels que CAVLC) car ils adaptent leur façon d'encoder les valeurs des coefficients selon le contexte, c'est à dire selon les informations qui ont déjà été encodées par le codeur. Le tableau 1.2 compare brièvement les caractéristiques principales des différents codeurs entropiques.

TABLE 1.2 – Méthodes de codage entropique

	MPEG-2 VLC	*H.264 CAVLC*	*H.264 CABAC*
Méthode de codage	VLC(codage à longueur variable)	VLC	Codage arithmétique
Adaptation au contexte	Aucune	Niveau coefficients transformés	Adaptation au bit près
Efficacité	Moyenne	Bonne	Excellente
Applications	DVD, TNT	Vidéo diffusion	Support HD (Blu-ray ...)

1.2.3.4.1 L'encodage VLC Le codage à longueur variable (VLC : *Variable Length Coding*) se base sur la probabilité d'apparition des éléments d'un ensemble. C'est une variation du codage de Huffman et est utilisé essentiellement dans MPEG-2. Le codage

de Huffman est réalisé pour un ensemble de symboles et un message à transmettre en ordonnant les symboles par ordre de fréquence d'apparition dans le message. On procède ensuite à un arbre qui permet d'attribuer un mot code à chaque symbole. Du fait de l'organisation des symboles et de la création spécifique de l'arbre, il en résulte que les symboles qui apparaissent le plus dans le message sont codés avec des mots codes courts alors que les moins fréquents sont associés à des mots codes plus longs. Le VLC utilise une table fixe qui attribue chaque valeur à un mot code donnée. Exp-Golomb et CAVLC font partie de la famille des VLC mais CAVLC est dit adaptatif et permet une compression plus efficace grâce à cette caractéristique.

1.2.3.4.2 L'encodage Exp-Golomb L'encodage Exponential-Golomb est un code à longueur variable relativement simple qui est utilisé dans H.264 pour encoder des informations dans les différentes entêtes présentes dans le flux, mais aussi pour encoder les vecteurs de mouvement issus de la prédiction du mouvement (prédiction Inter) ou plutôt pour l'encodage de la différence entre les vecteurs de mouvement réellement calculés et les vecteurs de mouvement prédits par rapport aux vecteurs de mouvement trouvés précédemment. Ce qui permet de n'avoir à coder que de faibles différences (si les prédictions de vecteurs de mouvement sont efficaces) grâce au système Exp-Golomb. Le tableau 1.3 montre l'attribution figée des mots codes binaires aux valeurs réelles.

Le nombre de zéros précédent le 1 (ces 0 et ce 1 étant appelés bits de préfixe) permet au décodeur de connaitre la longueur totale du mot code.

1.2.3.4.3 L'encodeur CAVLC Utilisée dans H.264, tous profils confondus, CAVLC (*Context-based Adaptive Variable Length coding* ou codage adaptatif de longueur variable selon le contexte) est une méthode de compression dérivée s'appuyant sur VLC mais possédant des propriétés d'adaptation au niveau de table d'attribution des mots codes. Les tables sont en effet mises à jour en fonction des éléments déjà encodés par CAVLC. Cette adaptation existe à deux niveaux. Premièrement au niveau bloc : certains éléments qui permettent de coder par exemple le nombre de coefficient non-nuls dans

TABLE 1.3 – Attribution des mots codes

Numéro de code	*Mots codes*
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

le bloc courant sont trouvés dans une table qui est modifiée en fonction des caractéristiques des blocs qui se trouvent au dessus et à droite du bloc courant. De même, à l'intérieur même d'un bloc, lorsqu'il s'agit d'encoder les valeurs proprement dites des coefficients, la table utilisée pour encoder la valeur courante dépend des valeurs des coefficients précédents déjà encodés. Ainsi, on optimise le codage en fonction du contenu de l'image. Les détails de fonctionnement de CAVLC sont appréciés plus loin

dans ce manuscrit.

Dans ces travaux, le CAVLC est largement privilégié car il apparait dans tous les profils de la norme H.264, à la différence de CABAC.

1.2.3.4.4 L'encodeur CABAC CABAC (*Context-based Adaptive Binary Arithmetic Coding* ou codage arithmétique binaire adaptatif selon le contexte) est un encodeur arithmétique bien plus complexe que ne l'est CAVLC. Il n'est d'ailleurs pas présent sur tous les profils, mais uniquement à partir du profil *main*. Le gain en termes de débit binaire par rapport à l'utilisation de CAVLC se situe entre 10 et 15%.

1.2.3.5 Codage scalable

Le codage scalable n'est pas une problématique nouvelle, mais ce type de codage même ancien a du mal à se faire une place dans les transmissions vidéo quotidiennes. L'idée du concept de codage scalable est d'obtenir, à partir d'une seule source encodée dans un format spécifique, une multitude de "'sous-flux"' vidéo possédant des caractéristiques différents, mais forcément inférieurs en terme quantitatif à la vidéo de base. On doit donc pouvoir recevoir, à partir d'un même flux, une vidéo avec des paramètres personnalisés ou adaptés à la chaîne de transmission et au terminal de visualisation, tels que la résolution spatiale, la résolution temporelle (le nombre d'image par seconde) et/ou la qualité visuelle. Un des enjeux de ce codage scalable est d'obtenir les mêmes performances en termes de qualité et de débit que du codage non-scalable. MPEG-2 présentait déjà des bases de codage scalable [ISOb] tel qu'il est utilisé dans les standards plus récents (notamment H.264 SVC). Cette recommandation prévoit donc un encodage par couches. Dans un premier temps, une couche de base est encodée suivant des paramètres très bas en termes de résolution, qualité et nombre d'image par seconde. D'autres couches viennent néanmoins s'ajouter à cette couche de base. Ces couches sont différenciées en trois catégories : les couches de rehaussement qualitatif qui ajoutent des informations sur les coefficients DCT (ou les coefficients de transformée entière selon le standard utilisé) permettant d'améliorer la qualité visuelle de la séquence vidéo. Les couches de réhaussement spatial permettent un passage d'une

résolution donnée vers une résolution supérieure. Ces couches sont obtenues par prédiction à partir de la couche de base (l'augmentation de la résolution par interpolation est un exemple de prédiction spatiale). Enfin, les couches de réhaussement temporel qui doivent venir ajouter des images au flux de base utilisent un système de vecteur de mouvement afin d'être prédites par rapport à la couche de base. MPEG-2 supporte donc déjà cette forme de scalabilité, mais MPEG-4 Visual apporte un nouveau type de scalabilité [ISOa], il s'agit d'une scalabilité fine qui permet d'obtenir beaucoup plus de variantes qu'avec l'encodage et les combinaisons limitées de quelques couches. Afin d'obtenir cette forme de scalabilité, l'encodage des coefficients DCT dans les couches de réhaussement sont de type plan de bit. Un ordonnancement de ce type permet un arrangement selon l'ordre d'importance des informations et une troncature du flux binaire permettra d'éliminer uniquement des informations de faible importance. Des méthodes de transcodage ou plutôt de modification du débit binaire par troncature de la couche de réhaussement organisée en bit-plane ont été proposée en [WWL+01]. Le codage scalable permet donc de répondre à une demande similaire à celle du transcodage. Le codage scalable permet de spécifier dés l'encodage le format et l'organisation des données pour une modification du débit binaire aisée, alors que le transcodage permet, à partir de n'importe quel format d'obtenir la meilleure forme de transmission pour un canal donné. Pourtant, il est clair que ces deux techniques ne doivent pas être vues comme des techniques concurrentes, mais plutôt comme des techniques complémentaires.

1.3 Canaux de transmission et décodeurs cibles

Une fois les séquences vidéos encodées selon un certains standard, il convient à présent de les transmettre à d'éventuels utilisateurs.

1.3.1 Canaux de transmission

Les canaux de transmission ou supports de transmission sont relativement hétérogènes. Leur but principal est bien entendu la transmission d'un équipement connecté au réseau à un autre de données, quelle qu'elles soient, de données vidéo dans notre cas. Sur ces canaux, le signal à transmettre est toujours codé. Le principe du codage dépend du type de support physique sur lequel l'information va circuler.

La figure 1.11 modélise une transmission de données entre deux équipements. On distingue les équipement terminaux de traitement de données (ETTD) qui désigne la machine qui interprète les données transmises et les équipements terminaux de circuit de données (ETCD) qui est une sous-partie de l'ETTD destinée à connectée cette dernière avec le réseau auquel elle est reliée.

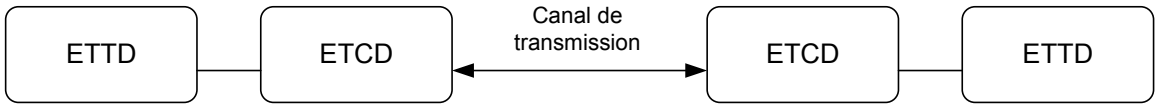


FIGURE 1.11 – Canal de transmission entre deux équipements

On compte trois grands types de supports physiques de transmission :

- Les supports filaires (câbles, paires cuivrées)
- Les supports optiques (fibres optiques)
- Les supports aériens (wifi, UMTS ...)

Chacun de ces supports est caractérisé par une bande passante et une capacité. La bande passante correspond à l'intervalle de fréquence sur lequel le signal transmis ne subit pas de perte supérieure à 3dB. Schématiquement, la figure 1.12 illustre cette caractéristique.

Quant à la capacité de la ligne de transmission, il s'agit du nombre de bits maximum pouvant être transmis en 1 seconde et s'exprime en bps (bits par seconde). La capacité d'un canal est liée à sa bande passante W et au rapport signal sur bruit S/N(qui dépend de la qualité de la ligne, de la présence ou non de parasites, etc...). Cette capacité C peut donc être caractérisée de la façon suivante :

$C = Wlog_2(\frac{S}{N} + 1)$

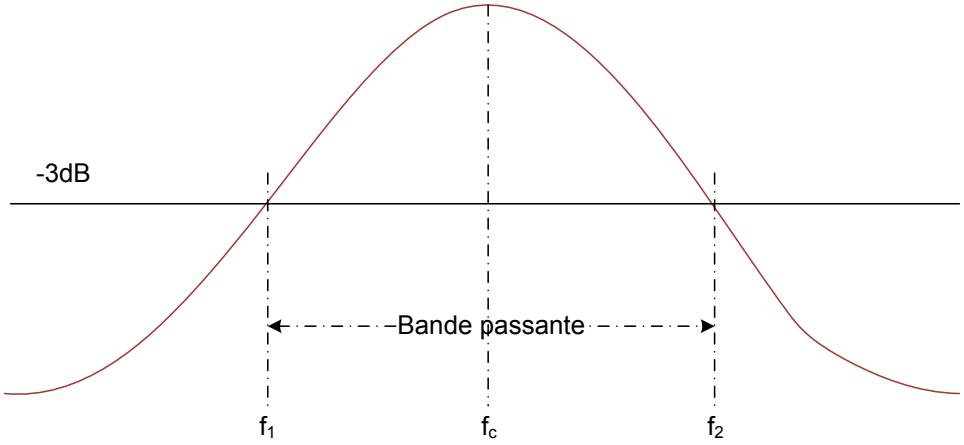


FIGURE 1.12 – Illustration de la bande passante d'un canal de transmission

Les protocoles de transmission sont des standards permettant la communication d'informations entre deux machines par le biais d'un réseau, quel qu'il soit. Il existe de nombreux protocoles, essentiellement sur Internet. Ces protocoles sont alors regroupés dans une suite qu'on nomme TCP/IP (*Transmission Control Protocol/Internet Protocol*) qui contient, en plus des principaux protocoles utilisés TCP et IP, des protocoles non orientés connexion (c'est à dire des protocoles qui ne garantissent pas la bonne transmission des données du fait qu'il n'existe pas de contrôle de transmission, contrairement à TCP par exemple) tels que UDP (*User Datagram Protocol*). Il existe d'autres protocoles dédiés à certains types de transmission. RTP et RTCP (*Real Time Transfer Protocol*) en sont un exemple. Ces deux protocoles permettent la transmission de contenus audiovisuels en temps réel sur IP. Ils ajoutent simplement au protocole inférieur (L'UDP dans le cas de RTP) des entêtes contenant des numéros de séquence, un identificateur de la source de diffusion et des marqueurs temporels permettant la lecture du flux par le client en temps réel. Le RTCP fournit un moyen de contrôle sur la qualité du flux transmis.

1.3.2 Décodeurs cibles

Les terminaux de visualisation de contenus audiovisuels sont également très variés. Il existe beaucoup d'équipements domestiques filaires visés par la diffusion vidéo. De

la télévision à définition standard à la celle ayant une capacité d'affichage HD 1080p, en passant par les possibilités de visualiser des séquences sur son ordinateur personnel, ces équipements sont multiples et omniprésents. Ils possèdent de surcroît des capacités très différentes que ce soit en termes de résolution spatiale d'affichage, de capacité de calcul ou encore de standards de compression vidéo décodables. L'ordinateur personnel possède un avantage sur les autres équipements dans le sens où ses capacités de décodages des différents standards peuvent évoluer simplement par l'installation de nouvelles solutions logicielles. Les téléviseurs intégrant d'ores et déjà un chipset de décodage spécifique à un standard de compression par exemple ont le lourd inconvénient de manquer de flexibilité et par conséquent de présenter une évolutivité quasi nulle lors de l'apparition de nouveaux standards (C'est par exemple le cas avec les TV intégrant un décodeur de télévision numérique terrestre décodant du MPEG-2 alors que certaines chaînes commencent à émettre du MPEG-4).

De plus, avec la généralisation des réseaux sans fil (3G, wifi ...), le nombre d'équipements mobiles ne fait que croître, possédant chacun ses caractéristiques. Là encore, on retrouve une hétérogénéité dans la taille de l'affichage, la capacité de calcul parfois limité par l'utilisation de batteries, ou encore de la présence, comme sur les terminaux filaires, de décodeurs matériels non évolutifs.

1.3.3 Influence sur le type de codage

Les deux points précédents accumulés, il apparait clairement que les possibilités de transmission sont multiples et qu'aucun couple standard de compression - canal de transmission ne sera optimal pour tous les terminaux de visualisation. La multiplication des standards, des moyens de transmission et des équipements clients amènent une très large problématique d'adaptation et de rétrocompatibilité. Aucun codage n'est à même de satisfaire la transmission sur tous supports vers tous terminaux. Les serveurs de contenus vidéo ne peuvent vraisemblablement pas abriter une multitude du même flux codé de façon spécifique pour les différentes transmissions prévues. De ce fait, des adaptations sont requises. Le transcodage fait partie de ces moyens d'adap-

tation. Le transcodage permet à partir d'une seule source (théoriquement, une source encodée avec une qualité et une résolution importantes) de fournir un flux dédié à un type de support de transmission et à un type de terminal. Le transcodage peut être envisagé à divers endroits afin de répondre à différentes contraintes. La figure 1.13 illustre la topologie d'un réseau classique type ADSL et indique les endroits où peut avoir lieu le transcodage. Les points rouges de la figure indiquent les emplacements possibles d'un système de transcodage. Le transcodage au niveau serveur est envisageable, mais demande une capacité de calcul importante du fait de la possible multiplicité des connexions à ces serveurs et donc du nombre important de transcodages différents à réaliser. Au niveau DSLAM, les capacités de calcul sont faibles, mais ils peuvent être utilisé pour du filtrage de couche dont on parlera dans la suite. Le modem-routeur (ou *Set Top Box*) peut également agir en tant que transcodeur pour d'éventuelles transmissions vers des terminaux mobiles qui ne requièrent qu'une faible résolution.

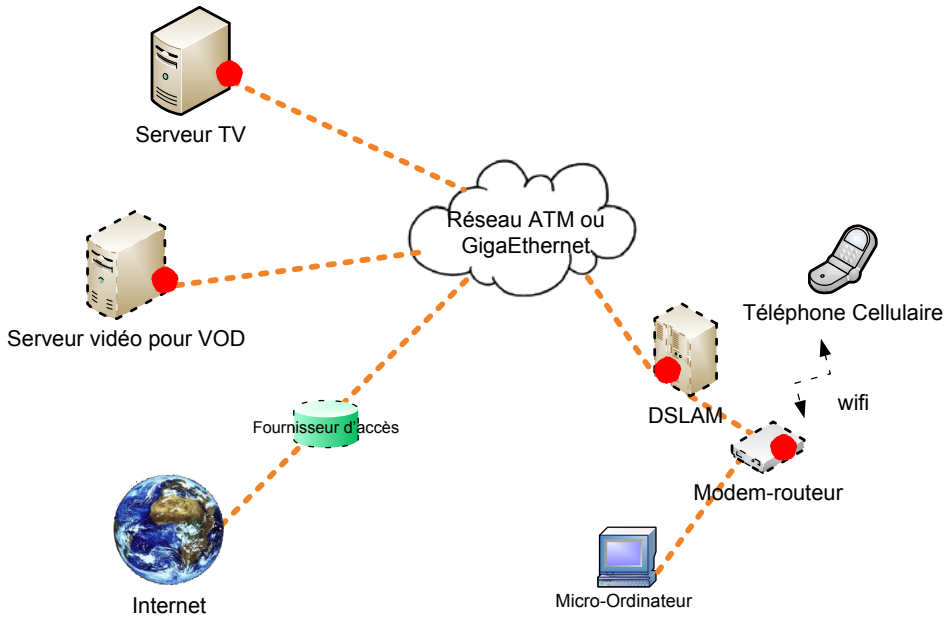


FIGURE 1.13 – Lieux de transcodage

Le codage scalable est également une solution proposé pour l'adaptation d'un contenu audiovisuel à une pluralité de terminaux et de transmissions. Un flux vidéo est encodé par couches successives, une couche de base représente la séquence avec une qualité et une résolution temporelle et spatiale minimale. Des couches de réhaussement

apportent une amélioration sur un de ces trois points. Ces couches peuvent être transmises ou non, ou filtrées lors du passage dans les éléments du réseau, au même titre que les opérations de transcodage. Cet encodage par couche ne permet que le transcodage homogène (le standard reste le même en entrée et en sortie du processus de filtrage). L'aspect de codage scalable est développé plus loin dans ce rapport.

1.4 Conclusion

Ce premier chapitre a permis de poser les bases du codage vidéo. La majorité des notions abordées ici seront réutilisées dans la suite et éventuellement discutées plus profondément. La problématique concernant le codage vidéo, et la transmission du flux compressé ont été mis en avant afin de saisir l'intérêt des techniques de transcodage mais également celui du codage scalable. Il est clair que l'adaptation de ces contenus devient une nécessité. Disposant d'un large panel de standards de compression, le transcodage est un moyen très efficace pour adapter le contenu à tout type de support de transport et de terminaux. Dans le chapitre qui suit, les méthodes de transcodage vont être détaillées.

Chapitre 2

Adaptation et transcodage vidéo

Sommaire

2.1 Introduction

On retrouve dans les premiers travaux de transcodage, des études concentrées sur la réduction du débit binaire pour atteindre les capacités de transmission en termes de bande passante d'un canal de transmission. Les recherches se sont aussi concentrées sur la conversion entre flux vidéo à débit variable (VBR) et flux vidéo à débit constant (CBR). Cette conversion avait pour objectif de faciliter la transmission du flux.

Plus tard, lorsque les équipements mobiles sont apparus, leur faibles capacité de décodage ainsi que leur faible résolution d'affichage (des caractéristiques qui tendent à être fortement améliorées aujourd'hui) ont conduit les chercheurs à se concentrer sur des techniques permettant de réduire les besoins en terme de capacité de calcul, notamment en modifiant le nombre d'image par seconde de la vidéo ou la résolution de celle-ci. Il existe également des recherches basées sur le renforçage de la protection aux erreurs durant le transport. Certains de ces types de transcodage sont illustrés sur le figure 2.1. La figure montre un flux original en MPEG-2 (à gauche) et les différentes opérations qu'on peut effectuer par transcodage : transcodage vers M-JPEG avec réduction de la résolution temporelle et spatiale, transcodage homogène avec réduction du débit binaire par requantification et donc réduction de la qualité ou encore transcodage vers MPEG-4 avec réduction de la résolution spatiale et temporelle.

Dans tous les cas de figures, il est généralement toujours possible d'utiliser une approche dans le domaine pixel de type *full decode/full recode* qui réalise de manière classique un décodage total du flux vidéo suivant le standard utilisé pour l'encodage puis, une fois la vidéo brute obtenue, le système se charge de réencoder totalement le flux dans un nouveau standard avec des caractéristiques prédéterminées. Cette façon de transcoder une vidéo, bien qu'elle puisse être dépourvue de tout problème de conversion et qu'elle puisse obtenir une qualité optimale est une aberration en termes d'utilisation de ressources. D'autres méthodes permettent heureusement d'obtenir de bons résultats en réduisant considérablement le coût d'implémentation. La course à l'efficacité et à l'optimisation du ratio ressources utilisées - performances qualitatives ont permis de mettre à jour différentes méthodes de transcodage.

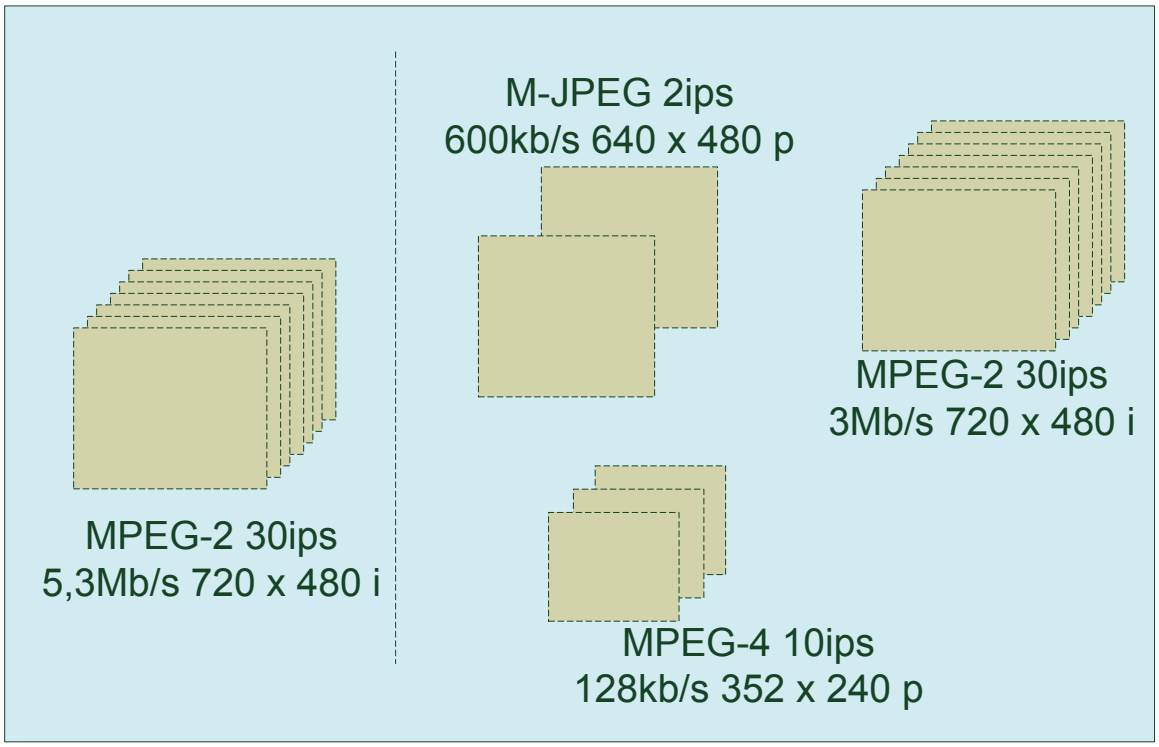


FIGURE 2.1 – Opérations de transcodage classiques

Dans ce chapitre, les codages classiques comprenant une estimation du mouvement (et une compensation) ainsi qu'une transformation fréquentielle qui permettent respectivement de réduire les redondances temporelles et les redondances spatiales seront étudiés. Dans de telles approches, les images des séquences vidéo sont découpées en blocs ou plus précisément (et dans un premier temps) en macrobloc (MBs). Chacun de ces blocs représentent une matrice de 16 pixels par 16 pixels en ce qui concerne la luminance (Y) alors qu'ils ne représentent qu'une matrice de 8x8 pixels pour chaque composante de chrominance (Cr et Cb ou U et V). Les premières méthodes de transcodage historiquement apparues étant celles permettant la réduction du débit, c'est tout naturellement que cet aspect est abordé en premier.

2.2 Les opérations de transcodage

2.2.1 Réduction de la résolution spatiale

Il existe de nos jours des terminaux de visualisation aux antipodes. D'une part on trouve des affichages Haute Définition voire d'ici quelques année une démocratisation de l'Ultra Haute Définition (seize fois plus d'informations qu'en HD) alors que d'autre part, on trouve des afficheurs compactes, sur des équipements mobiles essentiellement dont la résolution ne dépasse pas le VGA voire le CIF. La possibilité d'afficher les contenus audiovisuels sur des écrans très larges poussent les fournisseurs de ces contenus à les encoder à haute résolution. Le besoin de transcodage en résolution devient alors réellement nécessaire pour permettre aux petits équipements mobiles d'accéder à ces contenus encodés à l'origine en haute résolution.

De même que nous l'avons vu précédemment, les architectures de transcodage passant par un décodage complet puis un réencodage permettent d'obtenir de très bon résultats en terme de qualité au détriment de l'optimisation de l'implémentation. Des alternatives ont cependant pu voir le jour grâce à certains travaux de recherche [BC98, BC00].

La réduction de la résolution spatiale pose, au delà de la transformation des résidus, des problèmes concernant les vecteurs de mouvement qui doivent être de nouveau calculés ou interpolés en réutilisant les vecteurs calculés à l'encodeur. Un des premiers travaux à faire l'objet de ce type de recherche est décrit en [SSV97]. D'autres apportent des solutions concernant une réduction de la résolution spatiale à partir des coefficients transformés (dans le domaine DCT) [ZYB98, SG00].

2.2.2 Réduction de la résolution temporelle

La résolution temporelle définie le nombre d'image par seconde que transporte un flux vidéo. La réduction de cette résolution et donc du nombre d'image par seconde permet de réduire d'une part le débit binaire du flux, mais également de réduire les ressources nécessaires au décodage de ce dernier. La qualité des images, elle, n'est pas

affectée par ce type de transcodage. Le choix de se transcodage se fait généralement en fonction des limitations des performances de calcul du récepteur visé. On préférera une réduction de la qualité ou de la résolution si le récepteur est capable de décoder le nombre d'image par seconde original.

Ce type de transcodage pose à nouveau un problème au niveau des vecteurs de mouvement. En effet, comme il est précisé plus haut, pour les transcodages utilisant une réduction de la résolution spatial ou une réduction du débit par réduction de la qualité, les vecteurs de mouvement peuvent être réutilisés, ou interpolés grâce à des transformations relativement simples. Ici, certaines images étant éliminées, les vecteurs de mouvement doivent également être mis à jour. La figure 2.6 illustre la réestimation de ces vecteurs qui doivent être recalculés en fonction des images qui n'ont pas été supprimées.

Là encore, la problématique n'est pas nouvelle et de nombreuses études ont pu être menées pour pallier ce problème [YSL99, HWL98].

2.2.3 Réduction du débit par réduction de la qualité

Une réduction de débit ne doit pas se contenter de permettre une réduction du débit binaire afin de cibler un canal de transmission aux capacités réduites. Le système doit également proposer une qualité optimale. Comme nous l'avons vu précédemment, les meilleures performances en termes de qualité peuvent être obtenues en décodant totalement le flux puis en le réencodant, avec, par exemple, une quantification plus importante. La figure 2.2 montre une telle architecture. En effet, les meilleures performances sont atteintes en recalculant les vecteurs de mouvement qui peuvent être légèrement différent étant donnée les modifications qu'on est susceptible d'appliquer sur le flux vidéo. La démonstration de ces performances optimales est réalisée en [SKZ96].

Au delà de cet aspect de recalcul des vecteurs de mouvement qui apporte certes une qualité optimale au transcodage mais qui nécessite malheureusement une quantité de ressources importante, il existe une alternative. Les vecteurs de mouvement peuvent être en effet réutilisés [KHHH96, BC00]. Cette approche permet une réduction des res-

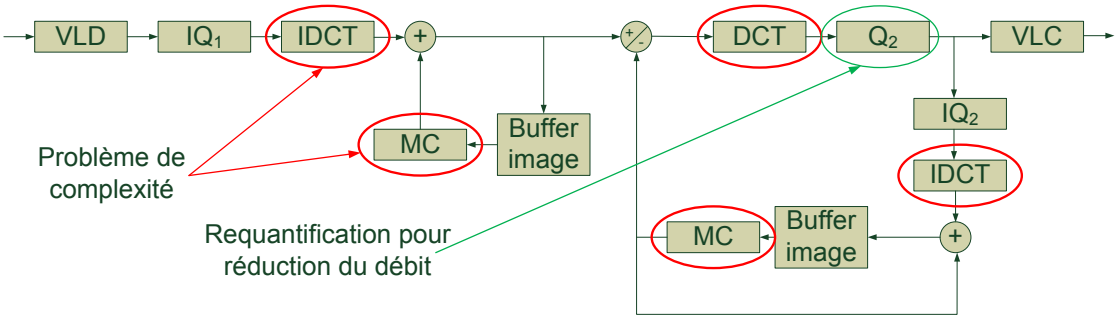


FIGURE 2.2 – Approche en cascade dans le domaine pixel

sources nécessaires. Durant ces dernières années, les travaux dans le domaine de la réduction du débit se sont concentrés sur les problèmes de propagation des erreurs dans les prédictions (ou *drift*) ainsi qu'à la réduction de la complexité. Ce *drift* (figure 2.3) peut être expliqué par la perte des informations de haute fréquence de l'image lors de la requantification. Ces pertes engendrent alors une différence entre les images prises comme référence lors des prédictions. De ce fait, les prédictions ne peuvent être justes et les erreurs peuvent se propager d'image en image jusqu'à ce qu'une nouvelle image de référence (image I en générale) vienne rafraichir le transcodeur. On pourrait trouver deux types de systèmes de transcodage. Il s'agit des systèmes en boucle ouverte et des systèmes en boucle fermée. Ces deux principes permettent de comprendre les problèmes classiques et leurs solutions qu'on trouve dans les méthodes de transcodage.

2.3 Fonctions mises en oeuvre

2.3.1 Transformée entière

L'architecture en boucle fermée permet la transformation des macroblocs pour la réduction du débit binaire dans le domaine des coefficients (domaine DCT). Malheureusement la compensation de mouvement nécessitant des images de référence dans le domaine spatial (au niveau pixel), une transformée inverse (et respectivement une transformée directe) est nécessaire afin d'obtenir l'image dans ce domaine spatial. Cer-

FIGURE 2.3 – L'effet de drift ou propagation d'erreurs sur la qualité de l'image. A gauche, l'image de référence, à droite, l'image obtenue après transcodage

taines études ont cependant montré qu'il était possible de supprimer cette phase de transformation du domaine DCT au domaine spatial [CM95].

De même que la réduction du débit binaire par la réduction du nombre d'information (et donc de la qualité globale de l'image), la réduction spatiale peut également se faire dans le domaine de la transformée ou domaine DCT. Il est possible en effet de ne retenir que les informations concernant les basses fréquences de chaque bloc et de regrouper certains de ces blocs obtenus entre eux afin d'en créer un nouveau. Une technique de regroupement de ces blocs est proposée en [CM95]. Imaginons que les macroblocs soient constitués de 8x8 pixels, les blocs de coefficients transformés ont alors le même aspect. Il suffit donc, dans le cas d'une réduction de la résolution par deux, de ne conserver que les informations de basse fréquence de chacun de ces blocs, soient des blocs de 4x4 pixels pris au début de chaque bloc 8x8 (en haut à gauche selon l'organisation en zig-zag des coefficients des basses fréquences vers les hautes fréquences). Ensuite, quatre de ces blocs ainsi obtenus vont venir former un nouveau bloc 8x8. Certaines approches plus subtiles permettent cependant de retenir des in-

formations au niveau des hautes fréquences au lieu de les éliminer complètement. Ces techniques sont développées en [SC98] et [VSDP98]. Ces techniques utilisent des filtres appliqués de manière horizontale puis verticale sur chaque bloc afin d'obtenir un nouveau bloc qui prend en compte aussi bien les informations de basse fréquence que les informations de haute fréquence.

2.3.2 Quantification

La fonction de quantification intervient lors de la réduction du débit binaire par requantification. La requantification est un procédé comprenant à la fois la quantification inverse, de manière à retrouver les coefficients DCT d'origine (avec l'erreur de quantification) et la quantification directe, qu'on utilise avec un paramètre de quantification Q_p différent (et forcément inférieur) du paramètre de quantification utilisé lors de l'encodage original de la vidéo.

2.3.3 Codage entropique

Le codage entropique, sous quelque forme que ce soit, est utilisé dans tous les types de transcodage pour décoder les coefficients DCT quantifiés. Pour les transcodages classiques de réduction de débit, réduction spatiale ou temporelle, le codage entropique ne joue qu'un rôle de décodage ou réencodage à longueur variable. Lors d'un transcodage qui vise à modifier le type d'encodeur entropique utilisé, sont impact est bien entendu plus important. En modifiant le type d'encodage entropique, on peut améliorer la compression en passant de CAVLC à CABAC, ou permettre à un décodeur pourvu seulement d'un décodeur CAVLC (type baseline) de lire un flux encodé précédemment dans un profil plus élevé (*extended* par exemple) utilisant CABAC pour le convertir en CAVLC. Ce type de modification s'applique plus particulière à H.264 qui autorise ces différentes formes de codage entropique. Le brevet [PB06] propose une méthode afin d'obtenir une conversion de l'un ou l'autre des codeurs entropiques. La figure 2.4 schématise l'action de ce transcodage entropique.

Le module de contexte CABAC et de traitement CABAC permet le codage ou le

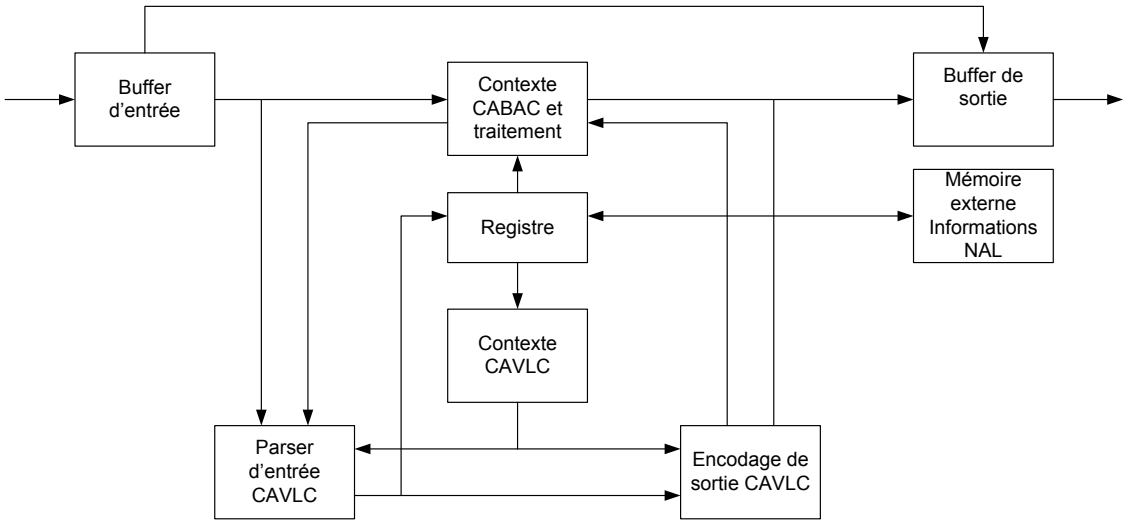


FIGURE 2.4 – Architecture du transcodage entropique pour H.264

décodage entropique en CABAC. Les modules Contexte CAVLC, Parser CAVLC et Encodage CAVLC permettent la même opération pour CAVLC. Lors d'un transcodage CABAC vers CAVLC, le module Contexte CABAC décode les éléments binaires et les envoie vers les modules d'encodage CAVLC. Lors de l'opération inverse, ce sont les modules CAVLC qui décodent le flux puis l'envoient au module de codage CABAC. Les entêtes de NAL devant être mises à jour suivant le nouveau codage, elles sont également modifiées.

2.3.4 Vecteurs de mouvement

Les problèmes de transformation des vecteurs de mouvement lors d'opération de transcodage ont été brièvement abordés précédemment. Lors d'une réduction de la résolution, une groupe de macrobloc (dont le nombre dépend du ratio entre la taille de l'image originale et la taille de l'image transcodée) est transformé en un seul macrobloc. Alors que chaque macrobloc se voit attribuer un vecteur de mouvement lors de l'estimation de mouvement à l'encodeur, cette transformation nécessite également de revoir le mappage des vecteurs de mouvement. Quand, par exemple, la résolution en largeur et en hauteur d'une séquence vidéo est divisée par deux, ceci signifie qu'un groupe de quatre macroblocs doit être transformé en un seul macrobloc. De même le

groupement des quatre vecteurs de mouvement associé à ce groupe de quatre macrobloc va alors être modifié pour ne représenté plus qu'un seul vecteur de mouvement correspondant au nouveau macrobloc unique. La figure 2.5 illustre la problématique de transformation de ces vecteurs de mouvement. Certains travaux tels que [SG00] et [YWL00] ont mis en avant ces problématiques.

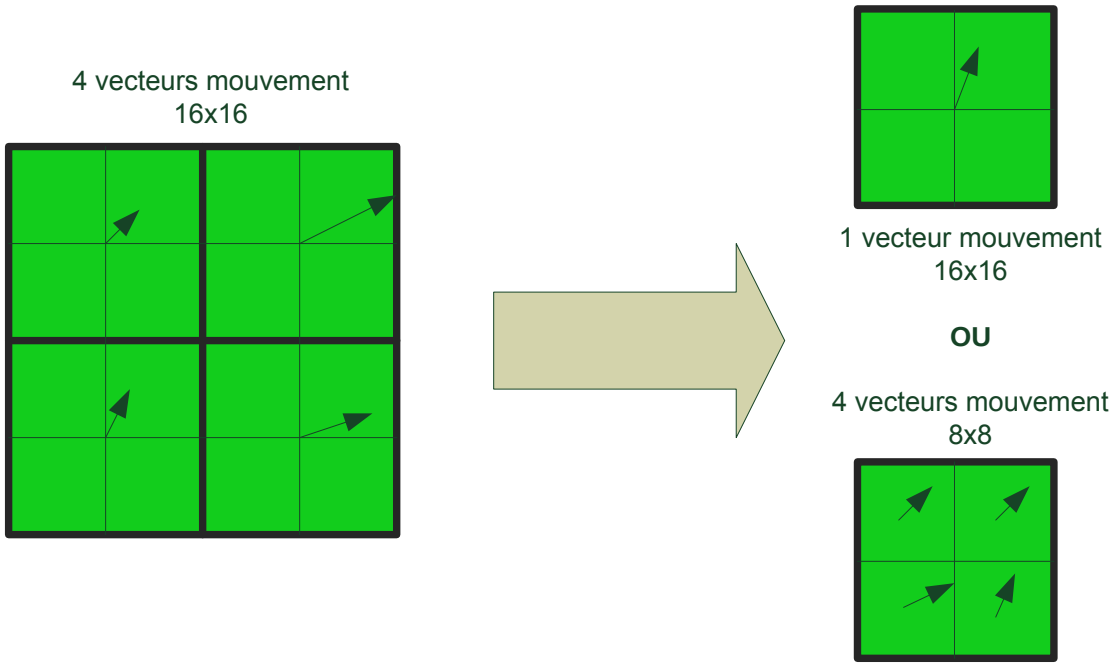


FIGURE 2.5 – Mappage des vecteurs de mouvement

Comme le montre la figure pour transformer nos quatre vecteurs de mouvement en un seul, il est possible d'appliquer une moyenne pondérée ou un filtre médian. Cette transformation classique est appelée 4 :1 mapping. Dans certains standards récents, il est cependant possible d'attribuer des vecteurs de mouvement à des blocs plus petits (8x8 par exemple). C'est le cas dans des standards tels que MPEG-4 Visual [ISOa], H.263 [IT] et le récent H.264 [H26]. Dans ces standards, il est donc possible d'attribuer chaque vecteur du groupe (qui comporte des vecteurs attribués à des macroblocs 16x16) à des blocs plus petit (8x8) dans l'image à résolution réduite. Dans ce cas, on parle de 1 :1 mapping car les vecteurs ne sont pas transformés, ils sont simplement réattribuer. [SG00] comporte également une étude sur les différents type de mappage et leur performance en terme de qualité.

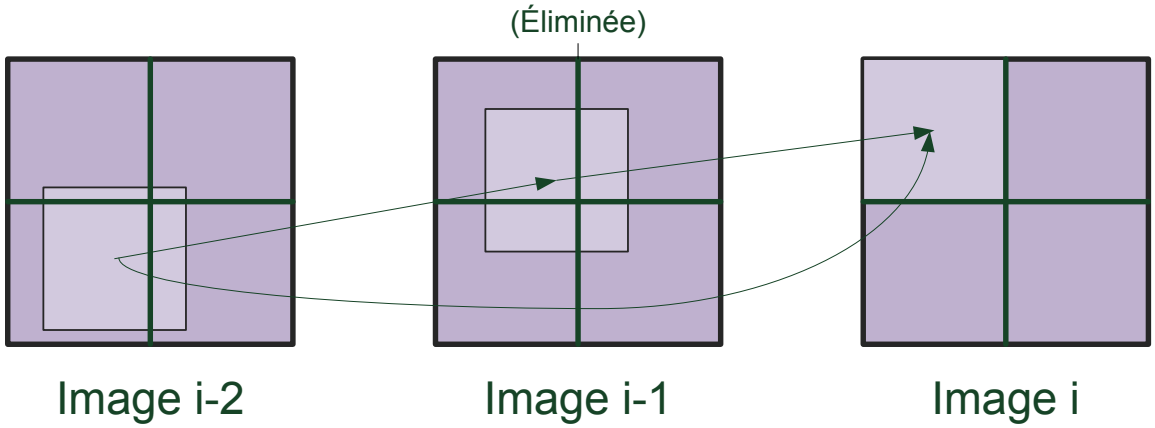


FIGURE 2.6 – Réestimation des vecteurs de mouvement

Dans les architectures à boucle fermée, une réestimation des vecteurs ainsi que des résidus (du fait que la compensation de mouvement se faisant grâce aux vecteurs et que celle ci détermine l'aspect des résidus) sont nécessaires.

2.3.5 Estimation de mouvement

[YSL99, HWL98] expliquent comment la réestimation des vecteurs de mouvement peut-être menée. A partir de l'image actuelle, il est possible de rechercher l'origine du vecteur en cours de réestimation dans la précédente image qui n'a pas été éliminée. Dans [YSL99] et [VYLS02], une technique particulière d'approximation du vecteur de mouvement est utilisée. Le codage n'est toutefois pas optimal et il s'avère que pour plus d'efficacité, un raffinement est nécessaire. Dans [HWL98], un algorithme est développé afin d'obtenir une estimation du vecteur de mouvement. Cet algorithme prend en compte le nombre d'images éliminées entre l'image qui va devenir l'image de référence dans le flux transcodé et la taille du vecteur de mouvement attribué au bloc courant.

Il est à noté qu'il existe également des études qui ont porté sur une détermination automatique des images à supprimer pour atteindre un nombre d'image par seconde donné. Ces algorithmes sont développés en [HWL98] et [LH97].

2.4 Décodage total et recodage

Le décodage total suivi d'un réencodage du flux vidéo est souvent utilisé à titre de comparaison car il donne souvent les meilleurs résultats. En effet, aucune erreur n'est introduite puisque le décodage complet permet d'obtenir les pixels (au niveau YUV) d'origine (avec leur dégradation éventuelle suite à l'encodage) et ensuite de réencoder en répétant les processus d'estimation de mouvement, de compensation, ainsi que de prédiction Intra. Le réencodage total permet de définir les meilleurs modes de prédiction et donc d'obtenir les meilleures performances de compression pour un débit donné. Lors de transcodage optimisées, des concessions sont toujours faites sur le choix des modes de prédiction qui ne sont pas toujours redéfinis, mais qui, au prix de performances moindres, allègent la complexité.

Souvent, on trouve des transcodeurs avec décodage et recodage totale dans le domaine logiciel. Les implémentations de transcodeur [LBH04, BC98] présente dans un cas une solution logicielle utilisant des blocs entiers de décodage et de codage de différents standards. Dans cet esprit, le transcodeur est très modulaire puisque le changement total d'un bloc de décodage ou de codage change l'application visée. La modularité au sein d'architecture à décodage partielle et recodage partielle est difficilement atteignable et ces architectures sont fortement spécifiques. Comme le montreront les figures 2.10 et 2.2, ce type de décodage donne les meilleures performances, mais au prix d'une complexité accrue. Dans [DCNLVdW09], la solution de décodage total d'un flux AVC pour le recodage en SVC est illustrée, mais démontre que la complexité est bien trop grande. Les résultats de cette étude sembleraient indiquer en plus que pour des taux de compression élevés, le réencodage total devient moins performant en terme de qualité que le décodage et recodage partiel, mais cette étude ne concerne qu'un cas précis de transcodage homogène de H.264 AVC vers SVC.

2.5 Décodage partiel et recodage

Comme dit précédemment, on peut distinguer deux types d'architectures : celles en boucle ouverte et celles en boucle fermée. Les figures 2.7 et 2.8 illustrent respectivement ces différences. Dans le cas d'un système en boucle ouverte, le flux vidéo encodé est tout d'abord décodé afin d'obtenir les coefficients transformés et quantifiés et les informations de vecteurs de mouvement. Les coefficients sont alors déquantifiés puis requantifiés selon un nouveau paramètre de quantification permettant d'abaisser le débit binaire en supprimant encore un peu plus d'information sur les hautes fréquences de l'image. Les nouveaux coefficients obtenus passent alors par le codeur entropique afin d'obtenir un code à longueur variable non destructif (contrairement au procédé de quantification). Il est également possible, au lieu de passer par le procédé de requantification (qui peut générer un effet de fourmillement sur l'image), d'effectuer une sélection fréquentielle sur les coefficients de la transformée afin d'éliminer directement (en les remplaçant simplement par une valeur nulle) certaines valeurs constituant les hautes fréquences de l'image. Ce procédé ainsi que son implémentation est décrit en [DCB+09]. Il existe également une solution de sélection fréquentielle au niveau du bitstream lui même en éliminant les mots codes issus de la compression entropique qui représentent les coefficients correspondant aux hautes fréquences. Cette problématique n'est pas récente, on la retrouve ici [Ele95].

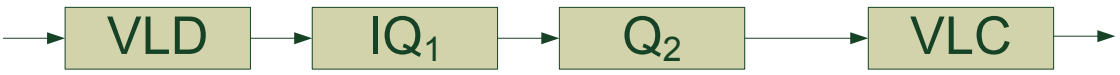


FIGURE 2.7 – Architecture de transcodage en boucle ouverte

Malheureusement, la simplicité des systèmes en boucle ouverte est anéantie par les effets de propagation d'erreur. En effet, l'image I, première image d'un groupe d'images (GOP : Group of Pictures, voir figure 2.9) sert de référence à l'image suivante, généralement une image P. Cette référence étant dégradée de par le transcodage, et le décodeur s'en servant à titre de référence, les vecteurs de mouvement peuvent ne plus être aussi précis et induire des erreurs. De plus, les résidus calculés en parallèle des vecteurs de mouvement à l'encodage et qui permettent normalement de compenser les

erreurs de prédiction de type inter sont eux aussi dégradés par le transcodage et sont donc dans l'incapacité de corriger les erreurs de prédiction. Des erreurs sont alors de nouveau introduites dans l'image P reconstruite. L'image suivante se basant sur cette reconstruction erronée et utilisant des vecteurs de mouvement et des résidus peu précis engendrera également d'autres erreurs. De ce fait les erreurs se propagent d'image en image jusqu'à ce qu'une nouvelle image I (un nouveau groupe d'image) apparaisse à l'entrée du décodeur et rafraichisse ce dernier. L'avantage en terme d'implémentation de ces systèmes en boucle ouverte tient dans le fait qu'il n'y a pas besoin de buffer pour d'éventuelle image de référence puisque tout ce passe au niveau coefficient et non pas au niveau pixel.

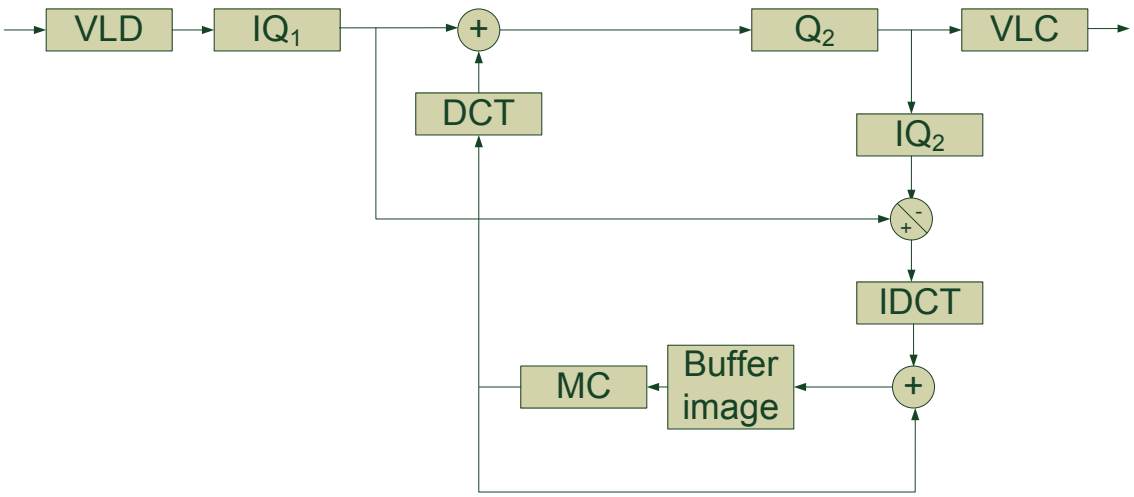


FIGURE 2.8 – Architecture de transcodage en boucle fermée

La seconde approche de transcodage à des fins de réduction du débit est un système en boucle fermée. Ce type de système est représenté sur la figure 2.8. Le but de ce type d'architecture est d'éviter la propagation d'erreurs qu'engendrent les systèmes en boucle ouverte. Pour ce faire, il faut éliminer les différences entre image de référence à l'encodeur et image reconstruite suivant les prédictions. Ces architectures sont en fait une simplification et une optimisation des transcodeurs de type *full decode - full recode* qui eux évitent toutes ces erreurs (au prix de besoins en ressources importants [AG96]).

Comme on peut le voir dans la figure 2.2, il existe deux boucles de reconstruction, chacune utilisant à la fois la transformée directe et la transformée inverse (qu'il

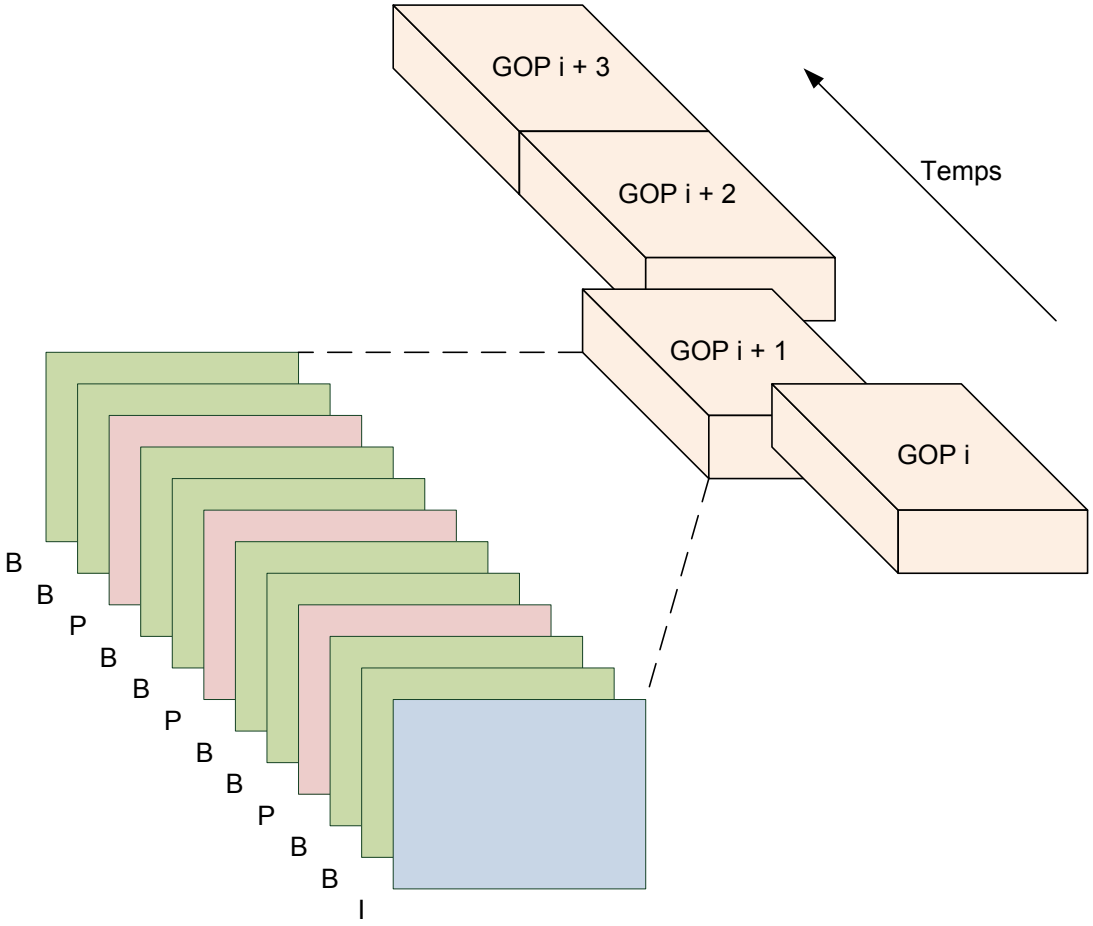


FIGURE 2.9 – Illustration de la succession des groupes d'images et de leur contenu en termes de type d'image

s'agisse de transformée entière comme dans le cas de H.264 ou de transformée type DCT comme dans MPEG-2 par exemple). L'algorithme simplifié présenté ici (figure 2.8) ne contient qu'une seule boucle de reconstruction.

Dans [VCS03], une comparaison succincte en terme de qualité est effectuée entre les systèmes vus précédemment. Les comparaisons ont été réalisées sur une courte séquence vidéo de 90 images en faible résolution (CIF) encodée grâce au standard MPEG-1. Le transcodage est réalisé à partir de cette source pour la transformer en un flux MPEG-4. Le graphique en figure 2.10 montre pour chaque image la qualité en terme de PSNR correspondante.

Il est clairement visible ici le problème souligné plus haut face aux architectures

en boucle ouverte, à savoir la propagation d'erreur. On note la baisse de qualité d'une image sur l'autre jusqu'à ce qu'une image I vienne à nouveau rafraichir le décodeur.

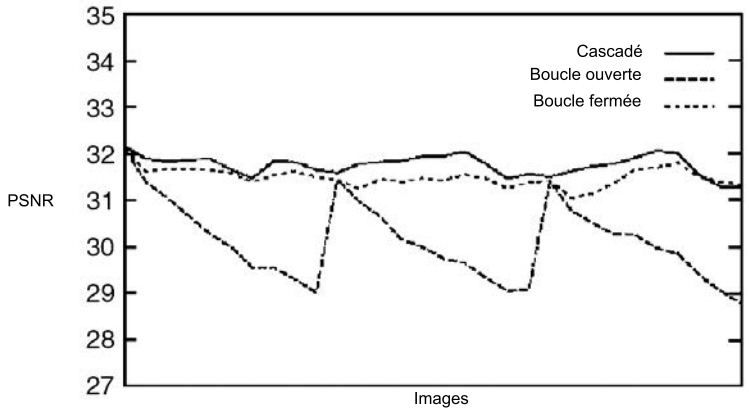


FIGURE 2.10 – Résultat de la qualité de l'image du transcodage obtenu avec les différents types d'architecture

On peut également noter la proximité des résultats concernant l'architecture en boucle fermée par rapport à l'architecture de type *full decode - full recode*. De ce fait l'optimisation concernant la baisse d'utilisation de ressources et de la complexité ne vient pas impacter fortement sur les résultats qualitatifs du transcodage.

2.6 Filtrage de paquets

Dans le codage scalable, les différentes de l'encodage peuvent être éliminées pour obtenir un flux plus léger au prix d'une réduction de la qualité ou de la résolution spatiale ou temporelle de l'image reconstruite selon le type de la couche de réhaussement éliminée. Dans [KPKH08] et [KKH09], un proxy de type RTSP/RTP est développé pouvant adapté le flux vidéo H.264/SVC à un client unique. Le protocole RTSP (*Real Time Streaming Protocol*) est utilisé pour produire une interaction entre le client et le serveur de contenu audiovisuel. L'utilisateur, à partir de la machine cliente peut alors lire le flux ou l'interrompre via une interface graphique.

H.264/SVC, tout comme AVC, est organisé sous forme de NALU (*Network Abstrac-*

tion Layer Unit). L'entête de chaque NAL est composée d'un octet pour AVC, mais de quatre octets pour SVC (figure 2.11).

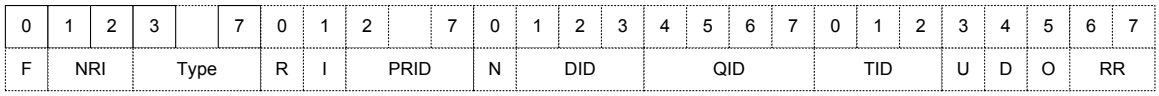


FIGURE 2.11 – Entête d'une unité NAL SVC

Les identifications de couche nommées TID (pour la scalabilité temporelle), DID (pour indiquer les dépendances hiérarchiques entre les couches de réhaussement) et QID (pour la scalabilité SNR) servent à différencier les couches de réhaussement. Chaque triplet de valeur unique correspond à une couche de réhaussement. Les NALUs sont paquétisés dans le protocole RTP [dSsKhJY10]. Les NALUs qui ont le même marqueur de temps peuvent être agrégés dans le même paquet RTP, mais un NALU trop long pour la taille du paquet RTP pourra également être découper en deux parties. Les deux paquets RTP sont alors indiqués comme des unités fragmentées dans leur entête.

L'adaptation SVC au niveau de l'équipement de filtrage (en l'occurrence un routeur wifi) est relativement simple. En fonction des informations reçues depuis le client (résolution requise, bande passante du canal ...) grâce à la communication RTSP, le routeur choisi quelles couches de réhaussement il convient de retransmettre au client et quelles couches doivent être éliminées. Selon ces règles, les paquets RTP sont lus jusqu'aux entêtes des NALUs pour obtenir le triplet (DID, TID et QID). Ce triplet unique pour chaque couche permet de différencier ces dernières et la retransmission au client des couches choisies.

La simplicité des opérations de traitement étant relativement réduite, ce type d'adaptation est très rapide et peut supporter la distribution de plusieurs flux vers plusieurs clients, en temps réel. L'implémentation de l'algorithme de filtrage a été réalisé sur un processeur de routeur de type MIPS32 à 200MHZ. L'utilisation de ce processeur n'atteint les 100% que lors du traitement d'un flux vers 6 clients différents pour une séquence 4CIF à 30ips et à un débit binaire de 2056kbps.

2.7 Conclusion et discussion

Les méthodes de transcodage passées ici en revue sont très hétérogènes. En terme de principe d'abord et par conséquent en terme de complexité et de coût mémoire. Certaines des méthodes présentes une complexité réduite, notamment pour le filtrage de couche, mais ne permettent en contrepartie qu'un seul type d'adaptation homogène et interdisent la modification intrinsèque du flux. Les modèles de transcodage de type décodage total - recodage total sont les plus efficaces en terme de qualité de transcodage, mais aussi les plus demandeuses de capacité de calcul et de mémoire (des buffers images sont fortement nécessaires). Elles ont cependant l'avantage de présenté une éventuelle souplesse puisque le remplacement du module de codage ou de celui de décodage est trivial. D'un autre côté, les optimisations réalisées pour le transcodage dans certaines architectures, le transcodage au niveau coefficients, permet de réduire largement la complexité, et donc les besoins en capacités de calcul et buffer d'images intermédiaire. Ces architectures sont relativement spécifiques et visent un transcodage donné. Elles sont peu flexibles du fait des fortes optimisations dédiées à ce transcodage, homogène (réduction de débit binaire, de résolution ... dans le même standard) ou hétérogène (changement de standard de compression avec éventuellement des modifications des caractéristiques du flux au niveau pixels).

De plus, les paramètres de l'environnement (bande passante du canal de transmission, changement du terminal de visualisation ...) sont mal gérés, souvent pas du tout, du fait du manque de flexibilité des différentes architectures. Différents scénarios peuvent être envisagés lors d'une transmission de contenu audiovisuel et les méthodes vues dans ce chapitre ne suffisent pas à permettre une bonne adaptabilité à l'environnement du transcodage.

C'est donc après avoir étudié et comparé ces techniques de transcodage qu'un choix a été fait concernant l'approche des travaux présentés ici. La prochaine section présente des implémentations concrètes, matérielles ou logicielles, qui utilisent certaines des techniques de transcodage vues ici.

Chapitre 3

Etude comparative des Implémentations de transcodage

Sommaire

3.1 Introduction

Les algorithmes de compression vidéo devenant de plus en plus complexes, et les caractéristiques des séquences vidéo encodée telles que la résolution, la qualité d'image, le nombre d'images par seconde augmentant de plus en plus, de nombreuses architectures basées sur des dispositifs au fonctionnement différent ont vu le jour. Ce chapitre propose un survol des produits utilisés pour l'implémentation d'architecture de transcodage en lien avec les méthodes de transcodage théoriques vues au chapitre précédent.

3.2 Les solutions logicielles

3.2.1 Implémentation sur processeurs génériques

Du fait que les codes de référence de la plupart des standards de compression vidéo se trouve être implémenté sur processeur et décrit dans un langage type C ou C++, se développe dans un premier temps une majorité d'implémentation logicielle. Ces implémentations sont généralement conçues pour être exécutées sur une plateforme monoprocesseur. Mais certaines études proposent une approche basée sur des plateformes multiprocesseurs. Il existe deux catégories d'architectures logicielles qui permettent d'effectuer du transcodage. Il y a d'abord les solutions ne permettant qu'un type de transcodage donné. Que ce transcodage soit hétérogène ou homogène, l'architecture ne permet que de passer d'un standard avec des caractéristiques données à une autre standard (ou au même standard dans le cas d'un transcodage homogène). Le second type d'architecture sont des solutions plus flexibles, voire reconfigurable, qui permettent un choix multiple de transcodage.

Les premières implémentations logicielles de transcodage touchant au standard H.264 sont basées sur l'outil logiciel de référence qu'est le JM (*Joint Model*) ou le JSVM (*Joint Scalable Video Model*) dans le cas de l'extension scalable de H.264.

[JZy09] propose une implémentation de transcodeur de MPEG-2 vers H.264. La compléxité du transcodage est fortement réduite grâce à la réutilisation des informa-

tions du flux concernant les vecteurs de mouvement. Dans MPEG-2, Les vecteurs de mouvement sont issus de bloc 16x16, alors que dans H.264, un vecteur de mouvement peut être associé à des blocs de taille variable (de 16x16 jusqu'à 4x4). Cependant, statistiquement, les vecteurs de mouvement sont le plus souvent associés à des blocs 16x16 (à au moins de 50%). A partir de ce constat, les auteurs, au lieu de recalculer tous les vecteurs de mouvement pour le codage en H.264, réutilisent les vecteurs de mouvement des blocs 16x16 qui présentent une énergie résiduelle très faible. Les autres sont alors décomposés en plus petit blocs et on cherche la combinaison de découpage qui donnera le meilleur résultat. L'algorithme a été implémenté logiciellement à partir du logiciel de référence JM.

Le gain en temps de calcul est relativement important. Les auteurs utilisent deux séquences test de 300 images en QCIF (176 x 144) à 25 ips, soit des séquences de 12 secondes. Alors que le transcodeur JM sans optimisation met respectivement 82.2 et 76.4 secondes pour le transcodage des deux séquences, le transcodeur optimisé réalise le transcodage en 42.1 et 35.3 secondes, soit un gain en temps d'exécution de l'ordre de 50% selon la séquence. Le débit binaire ainsi que la qualité des images ne sont quasiment pas touchés par cette optimisation.

[MFA+07] propose l'inverse : H.264 vers MPEG-2. Les auteurs utilisent une architecture cascadée au niveau pixel (c'est à dire un décodeur complet suivi d'un encodeur), voir figure 3.1 (volontairement simplifiée par rapport à la figure originale).

De même que pour l'architecture de transcodage précédente, les auteurs partent du constat que l'estimation de mouvement est le processus d'encodage le plus complexe. Le module d'adaptation situé entre l'encodeur et le décodeur permet de contrôler le module d'estimation du mouvement au niveau de l'encodeur et de le court circuiter si les informations contenues dans le flux H.264 sont réutilisables pour l'encodage MPEG-2. Ce module d'adaptation permet également de convertir les coefficients de la transformée entière en coefficients DCT directement utilisables pour l'encodage MPEG-2. L'implémentation de cet algorithme est réalisée sur un processeur de PC à 3GHz et testée sur des séquences 720p. Pour une qualité et un débit tout à fait similaire, l'implémentation décrite ici permet de gagner 30% en temps d'exécution face au

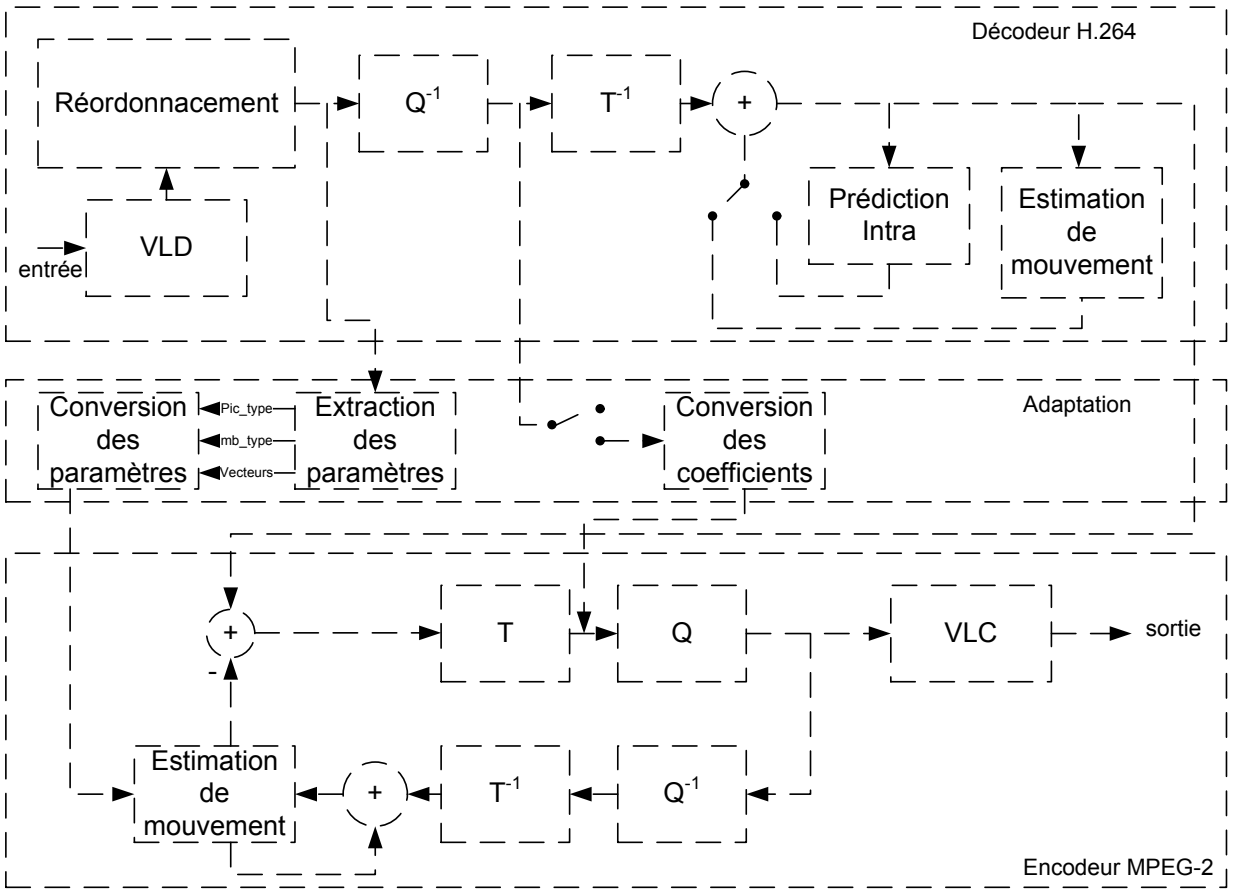


FIGURE 3.1 – Architecture de transcodage

transcodeur de référence.

Certes les algorithmes de réduction de la complexité de l'encodage H.264 permettent d'améliorer largement les temps d'exécution, mais il est clair que ces implémentations sont loin de tenir des contraintes de temps réel. Visiblement, les implémentations logicielles sont peu adaptées à ces impératifs.

Il existe des références commerciales de transcodeur [Dia09].

[DCNLVdW09] propose un transcodage de AVC vers SVC et compare différentes techniques. Afin d'éviter de cascader un décodeur AVC suivi d'un encodeur SVC, l'étude s'appuie sur des techniques vues au chapitre *Adaptation et transcodage vidéo* évitant de multiplier le nombre de modules MCP (*Motion Compensated Prediction*) par le nombre de couche SVC à produire.

Souvent, les implémentations logicielles prennent place dans des proxys (qui re-

présentent un intermédiaire entre le fournisseur du contenu, le serveur, et le client utilisateur). Ces proxys, doté de processeurs plus ou moins puissants peuvent assurer quelques formes de transcodage [LBH04, hMD02, WHZ+01, SLB04, DBKW05]. On peut classer les implémentations sur proxy en deux catégories : Les architectures monolithiques qui ne permettent le transcodage que d'un type de flux vers un autre type connu à l'avance et les architectures de transcodage dites heuristiques qui s'adaptent au contenu entrant et modifie le type de transcodage en fonction du terminal utilisé pour la visualisation du contenu audiovisuel.

[LBH04] propose un cadre d'implémentation sur proxy d'un transcodeur adaptatif et reconfigurable. Les auteurs utilisent une librairie Microsoft nommée DirectShow. Cette librairie contient des modules permettant de faire l'acquisition de flux vidéo (depuis un réseau, une webcam ...), de décoder ce flux ou de l'encoder dans divers standards, et enfin, de visualiser ou renvoyer le flux sur le réseau. Ce cadre possède trois niveaux. Le premier est le niveau de spécification qui détermine les besoins de l'utilisateur, décrit les types de flux et leur encapsulation réseau, et définit des conditions de reconfiguration de l'architecture de transcodage. La seconde couche est dédiée à la traduction de ces besoins décrits dans un langage haut niveau en une séquence de modules fonctionnels (issus de la librairie). La troisième et dernière couche concerne l'exécution de l'application. La reconfiguration d'un ou plusieurs modules est réalisée lorsqu'une condition décrite dans la couche de spécification est rencontrée (modification de la bande passante du canal de transmission, changement de terminal de visualisation ...). La reconfiguration peut ne pas engendrer de changement de module, mais simplement le changement d'un paramètre (par exemple, le paramètre de quantification, qui peut être modifié à la volée, sans être la cause d'une interruption du traitement du flux). Dans l'autre cas, lorsque la reconfiguration nécessite un changement de module, alors le traitement du flux vidéo, ainsi que sa transmission sont interrompues.

3.2.2 Architectures sur GPU

Ce nouveau paradigme émergeant vise à utiliser les capacités des cartes graphiques de nos ordinateurs dont la puissance n'est que rarement utilisée de manière optimale. Il est encore difficile de lire des travaux concernant le transcodage vidéo, mais il existe déjà des études qui ont pour objectifs l'encodage vidéo sur processeurs graphiques [CFAK10, CAK+09]. La puissance des processeurs graphiques (GPU : *Graphics Processing Units*) tient dans la parallélisation massive de leurs unités de traitement. Pour donner un exemple, une carte NVIDIA GeForce 8800GTS contient 96 processeur de flux travaillant à 1.2GHz. La figure 3.2 donne un aperçu de l'évolution des capacités de calcul des GPUs face aux processeurs classiques.

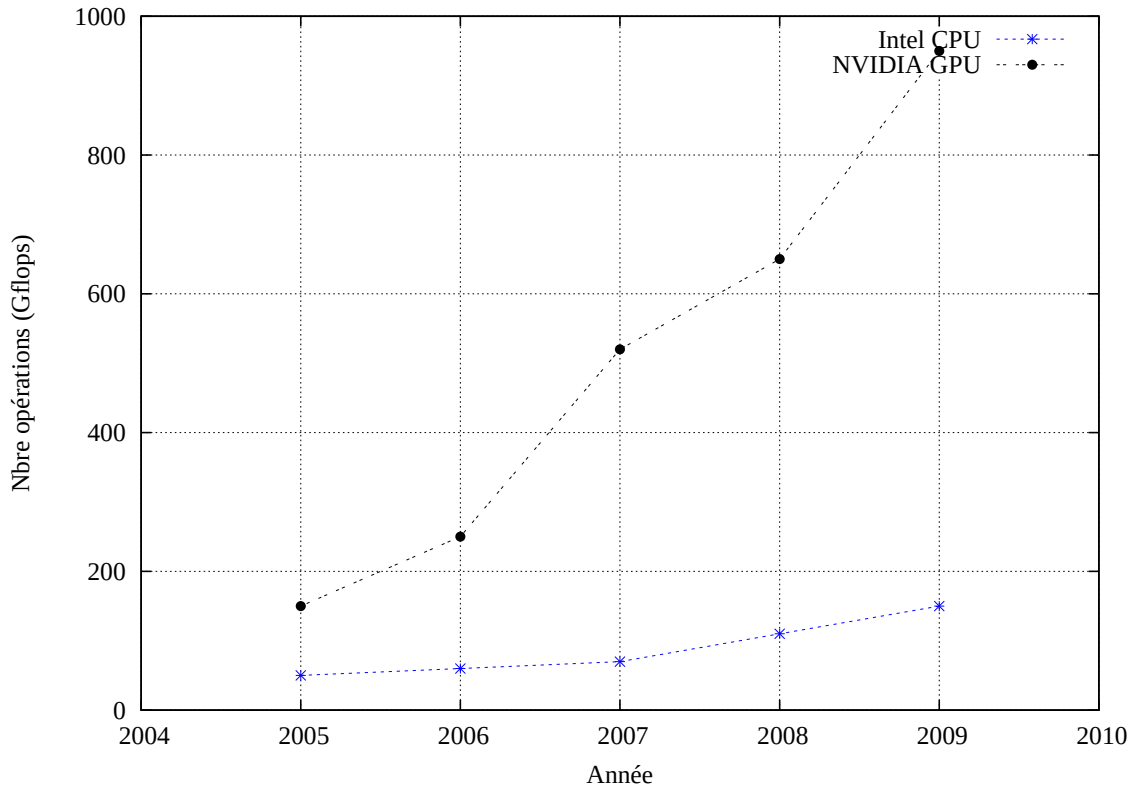


FIGURE 3.2 – Capacités de calcul des GPUs et CPUs

De part leur construction, les GPUs ne sont avantageux que pour des applications requérant beaucoup de calculs qui peuvent être réalisés en parallèle. Le fait que le GPUs ne possède que peut de modules de contrôle par rapport aux CPUs ne leur

permet pas d'être efficaces sur des applications comportant beaucoup d'instructions conditionnelles.

Le plus grand challenge concernant l'implémentation de traitement vidéo sur GPUs consiste en la structuration des fonctionnalités de l'encodage de manière hautement parallèle. L'encodage entropique, par exemple, s'adapte très mal à l'architecture des GPUs du fait qu'ils ne sont pas étudiés pour réaliser des instructions conditionnelles alors que les codeurs entropiques classiques comportent peu de calculs mais beaucoup de conditions. Par contre, l'estimation de mouvement, qui est une des fonctions les plus complexes de l'encodage, se prête tout à fait à l'architecture GPU, les calculs (des SAD par exemple) étant similaires et indépendants. Le GPU ne représente alors qu'un accélérateur matériel qui, couplé au processeur et bien utilisé, saura être efficace. Ceci était vrai pour les méthodes de recherche des vecteurs de mouvement basé sur la SAD (ces modes sont présents dans MPEG-1/2 et H.263). Pour H.264, par contre, on n'encode pas un vecteur entier, mais la différence entre le vecteur réel et une prédiction de ce vecteur. Or, pour prédire ce vecteur, sont utilisés les vecteurs précédemment calculés. Cette prédiction apporte donc une dépendance entre les calculs des vecteurs ce qui ouvre un nouveau challenge pour la parallélisation de ce type de calcul.

Les résultats de l'encodage sur GPU sont assez mitigés. L'idée proposée est de découpée l'image en groupe de macroblocs. Chacun de ces groupes est distribué dans un *thread* du GPU. La taille de ces groupes modifie donc le nombre de *threads* par image. Dans ces travaux, les groupements vont de un macrobloc (soient 3600 *threads*) à 40 (soient 90 *threads*). Plus la parallélisation est importante (jugée par le nombre de *threads*), et plus l'encodage sera rapide. Selon les séquences utilisées (toutes en 720p à 60 images par seconde), les meilleurs temps d'exécution (pour la parallélisation la plus forte) vont de 835.05ms à 1688.50ms pour une image. L'étude propose également une comparaison avec l'encodage sur CPU à simple coeur et CPU multi-coeur (en l'occurrence quatre). Il s'avère que l'augmentation de performance comparée à un simple coeur varie entre 1.5 et 3.5 fois (toujours avec la plus forte parallélisation). Par contre, la comparaison avec un multi-coeur montre que l'augmentation des performances est nulle, le processeur, pour certaines séquences faisant au moins aussi bien, voire deux

fois plus rapidement pour d'autres séquences.

Pour ce qui est du transcodage, il n'y a pas encore d'étude académique majeure dans le domaine, même si on peut trouver des applications commerciales de transcodage. Parmi ces solutions, on trouve celle de NVIDA, Badaboom [Kow08]. Cette solution logicielle utilisant le langage de programmation propre à NVIDIA, le langage CUDA, n'est pas détaillé très clairement par les concepteurs, mais semble pouvoir s'acquitter de transcodage de séquences haute résolution 1080p avec une vitesse d'exécution de 90 images par seconde avec la même carte graphique qu'utilisée dans les travaux précédents. L'autre constructeur leader de carte graphique ATI n'est pas en reste puisqu'il propose également une solution de transcodage logicielle utilisant les GPUs de la marque. Nommée AVIVO Video Converter [Wil09] cette solution ne semble tout de même pas optimale. Tout en donnant des résultats similaires en terme de rapidité et de qualité de transcodage que Badaboom, le CPU semble être utilisé à 100% au lieu des 30% qu'utilise Badaboom. L'utilisation des GPUs semblent donc bien différentes.

Au vu de ces résultats plus qu'intéressants pour ces solutions commerciales, il apparait clairement que les GPUs sont de sérieux acteurs pour le futur du transcodage et de l'adaptation temps réel.

3.2.3 Les processeurs réseaux

Les processeurs réseaux sont une alternative à l'utilisation de proxy pour le transcodage. On trouve ces processeurs dans les équipements d'interface réseau de moyenne gamme ou haut de gamme. L'idée de l'utilisation de processeurs réseau pour le transcodage vidéo n'est pas nouvelle [SEG06], mais la puissance de ces circuits augmentant toujours plus jusqu'à utiliser aujourd'hui des processeurs multicoeurs engendre un regain d'intérêt pour ces architecture [SEG09]. On trouve chez Intel, la série IXP dont le IXP2800 possède un coeur XScale entouré de seize *microengines*, de petits processeurs qui permettent de délester le processeur principal de tâches légères mais récurrentes. Ces processeurs cumulés permettent d'atteindre une vitesse de traitement de 25 Giga opérations par seconde. D'autres constructeurs proposent des NPU (*Network Proces-*

sing Unit) similaires (IBM, Agere, Motorola, ...). L'architecture d'un processeur réseau IXP1200 d'INTEL est représentée en figure 3.3.

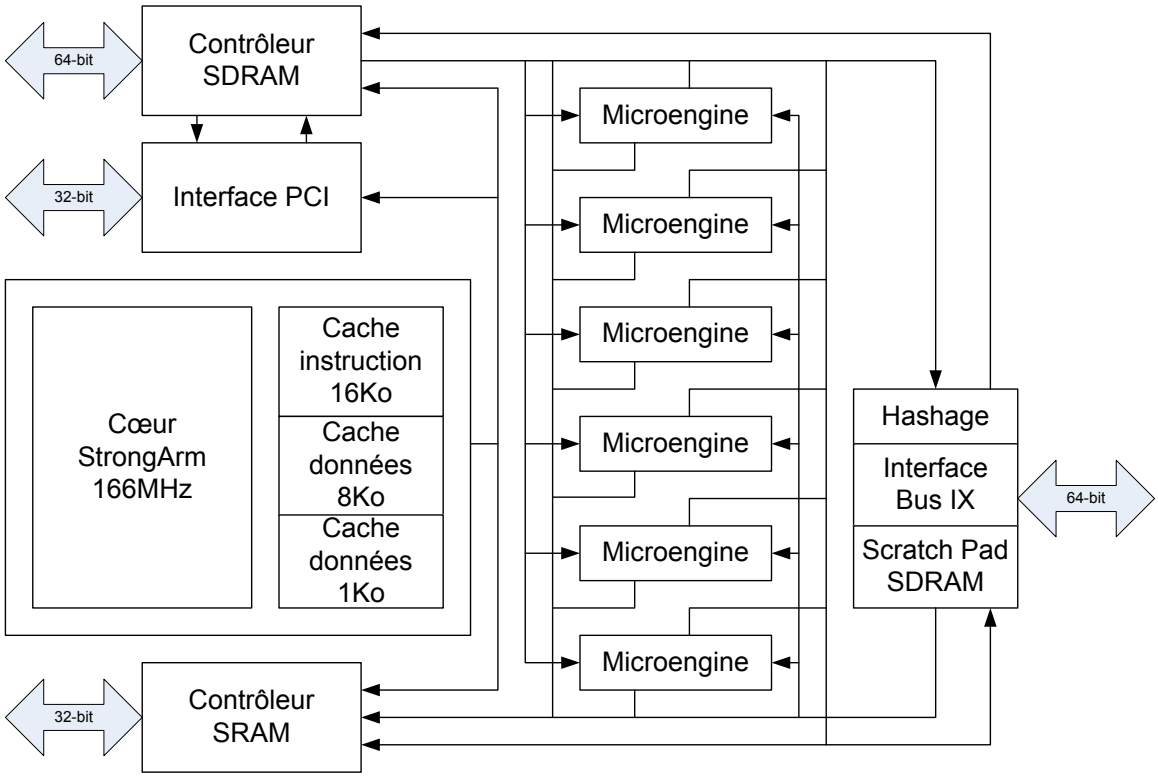


FIGURE 3.3 – Architecture du IXP1200

Chaque *microengine* peut être programmé pour exécuter un ensemble de tâches. Le processeur principal (ici un *StrongARM*) gère les communication entre les différents *microengine* de l'architecture et effectue également des tâches de gestion du protocole. Dans les travaux présentés en [SEG09] qui sont relativement uniques (d'autres architectures similaires n'existent pas selon les auteurs), les microengines sont utilisés pour assurer des tâches fixes. Dans le processeur utilisé, il existe 6 microengines, tous sont utilisés dans la mise en oeuvre de l'architecture de transcodage. Le microengine 0 permet la récepetion des paquets qui sont ensuite placés dans une FIFO afin d'éviter la perte de paquets lors de des processus de transcodage dont la latence peut être élevée. Le microengine 1 est associé à une tâche de classification des paquets. Cette classification permet de gérer la présence de plusieurs flux vidéo indépendants. Les microengines 2, 3 et 4 sont dédiés au transcodage. Deux types de transcodage sont étudiés sur

cette architecture. Il s'agit de traiter le cas d'une classique requantification au niveau des coefficients DCT, ou celui d'une réduction de la résolution temporelle par élimination d'image (*frame dropping*). Dans le cas de la requantification, les trois microengines sont nécessaires alors qu'un seul peut effectuer les opérations de réduction du nombre d'image par seconde. Enfin le dernier microengine (5) est utilisé pour la transmission vers l'extérieur.

Chaque microengine possède quatre *threads* matériels. Des mesures ont été effectués avec différentes implémentation qui utilise 1, 2, 3 ou 4 de ces *threads*. Les séquences vidéo utilisées sont de résolutions spatiales variables qui atteignent au maximum 256x256 pixels avec une résolution temporelle de 30ips (image par seconde). Pour atteindre le traitement temps réel, quelque soit les dimensions de l'image, pour 30ips, la latence par image ne doit pas dépasser théoriquement 33ms. L'architecture présentée peut atteindre le temps réel pour al requantification à partir de deux *threads* avec lesquels la séquence la plus lourde obtient une latence de 32.9ms par image. Avec quatre *threads*, on arrive à 19.9ms par image. Au vu des résultats donnés ici, il est évident que cette architecture ne peut convenir qu'à des séquences de faible résolution. Les auteurs précisent cependant que l'architecture peut être transposée sur des processeur de réseau plus performants tels que le IXP2400 ou le 2800 qui contiennent plus de microengines supportant plus de *threads*. Il n'existe actuellement malheureusement pas de transposition de ce type d'architecture sur des processeurs plus puissants.

3.2.4 Implémentation sur DSP

[SRS+04, Vun07, PSG+08] Les implémentations sur DSP pures sont relativement peu nombreuses (DSP : *Digital Signal Processor*). On trouve plutôt des SoC (*System on Chip*) utilisant des DSPs dans leur architectures. Comme le montre [Vun07], un simple portage d'une application d'encodage H.264 depuis une implémentation pour processeur d'utilisation générale (GPP) vers une implémentation pour DSP n'est pas d'une grande efficacité. Dans cet exemple, des séquences de 144 x 144 pixels sont encodées. Pour le processeur classique (Pentium 4 @ 2,4GHz), le temps d'encodage est de 26 se-

condes alors que ce temps passe à 65 secondes avec de grands efforts d'optimisation sur un DSP DM642 (sans optimisation, le temps d'encodage est égal à plus de 37 minutes !). Même si l'auteur reconnait que l'implémentation peut encore être optimisée, il est clair que l'utilisation d'un seul DSP pour toute l'application n'est pas envisageable pour réaliser des opérations de codage ou transcodage en temps réel.

Il existe cependant des circuits plus complexes comportant des DSPs qui sont très orientés pour le traitement vidéo et dont les performances sont bien entendu nettement supérieures. On peut cité notamment le TMS320DM6446 dont l'architecture est illustrée sur la figure suivante 3.4.

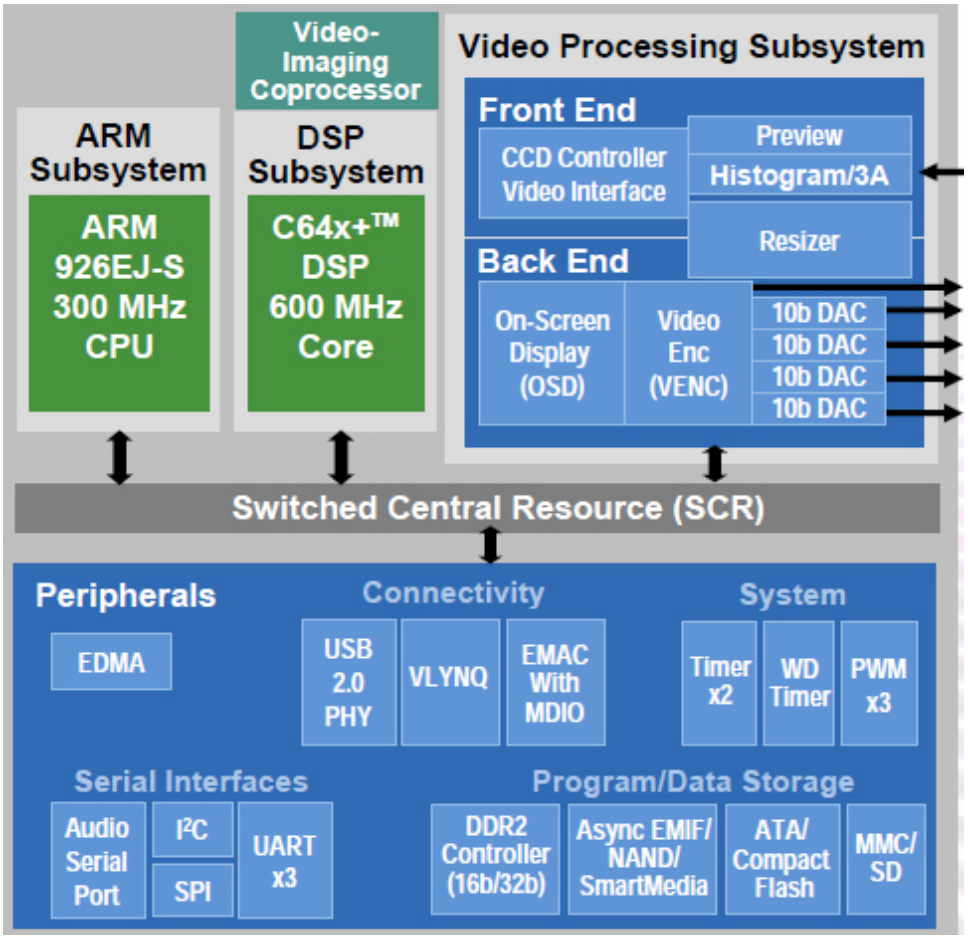


FIGURE 3.4 – Architecture du TMS320DM6446

Le DM6446 permet l'encodage de séquence 720p @ 30ips en H.264 profil de base à une moyenne de 590MHz.

3.3 Les solutions matérielles

Bien que les solutions logicielles offrent la plupart du temps une grande souplesse d'utilisation de part le matériel utilisé et le type de programmation qui peut être facilement mis à jour, les implémentations matérielles ont cet avantage d'être bien plus dédiées à un type d'utilisation, en l'occurrence le transcodage. Cet avantage permet d'obtenir des performances bien souvent nettement meilleures au détriment de la souplesse d'utilisation. Cependant, un compromis peut se trouver en l'existence de matériel reconfigurable.

3.3.1 Architectures sur FPGA

Les FPGA (*Field Programmable Gate Array*) sont composés d'un réseau de blocs logiques programmable par le stockage dans des cellules mémoire d'éléments binaires (ou bits de configuration). Le réseau reliant ces blocs logiques est lui aussi programmable. L'architecture d'un FPGA est illustré en figure 3.5 et la composition interne d'un bloc logique en figure 3.6. Ces illustrations sont valables pour des FPGA contenant des LUT à 4 entrées, il en existe de plus récentes versions à 6 entrées. En plus de cette architecture de base, certains FPGA contiennent des blocs DSP, des blocs MAC (*Media Access Controller*) pour une communication Ethernet, des blocs RAM, etc ... [JZy09, NLCC07, BKI04, MFA+07]

Les circuits FPGA proposent une fonctionnalité intéressante qu'est la possibilité de reconfiguration dynamique et partielle. Une reconfiguration dynamique permet la modification d'une fonction et donc de la programmation des LUTs et du réseau de communication, en cours de fonctionnement. Le fait que cette reconfiguration puisse être en plus partielle signifie que la reconfiguration ne peut avoir lieu que dans une zone limitée du FPGA sans que le traitement effectué dans le reste du circuit ne soit interrompu ou modifié. Ces types de reconfiguration s'illustrent sur la figure 3.7.

Ces particularités sont pourtant peu usitées dans la littérature. L'utilisation de la reconfiguration dynamique et partielle parait souvent négligée. Les architectures proposées dans la littérature basé sur des technologies FPGA portent fréquemment sur

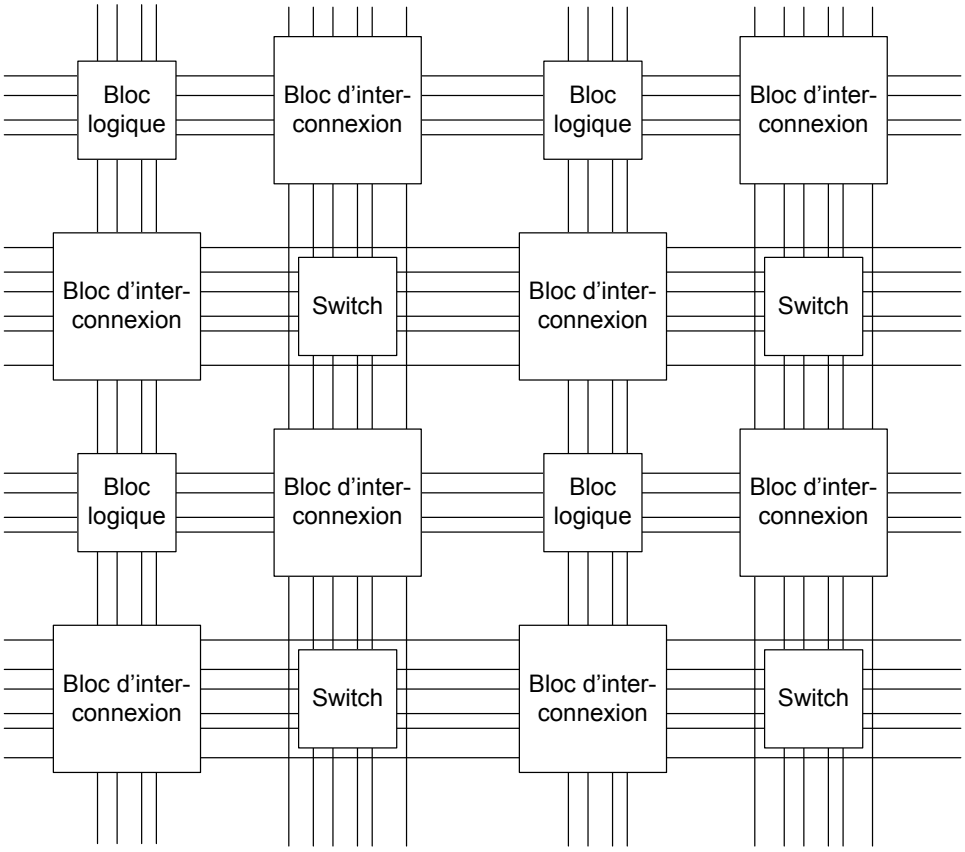


FIGURE 3.5 – Architecture globale d'un FPGA

un seul composant d'encodage ou de transcodage (Pour CAVLC [CLSG06, LC06, RB07, GM07], la prédiction Intra [SPC05, DCB+09] ou la quantification et la transformée [CWYL06, CWLY09]).

[HL09] propose une architecture VLSI (*Very Large Scale Integration* pour l'accélération du traitement des coefficients DTC et de l'estimation et la compensation du mouvement dans ce domaine. L'architecture est basée sur un réseau d'éléments de traitement. La force de cette architecture est de pouvoir recevoir différents algorithmes.

Même si le sujet des travaux de cette thèse est essentiellement orienté sur le transcodage, l'encodage, et a fortiori le décodage, peuvent être étudiés car contribue largement, même en étant modifiés, aux aspects de transcodage. Des encodeurs H.264 ont été réalisés sur FPGA [BDG+07] permettant de produire un bitstream vidéo en temps réel issu d'une vidéo brute de 1024 x 768 pixels à 15 images par seconde et ceci à 50MHz pour l'implémentation sur FPGA. Cet encodeur a été retenu pour son principe

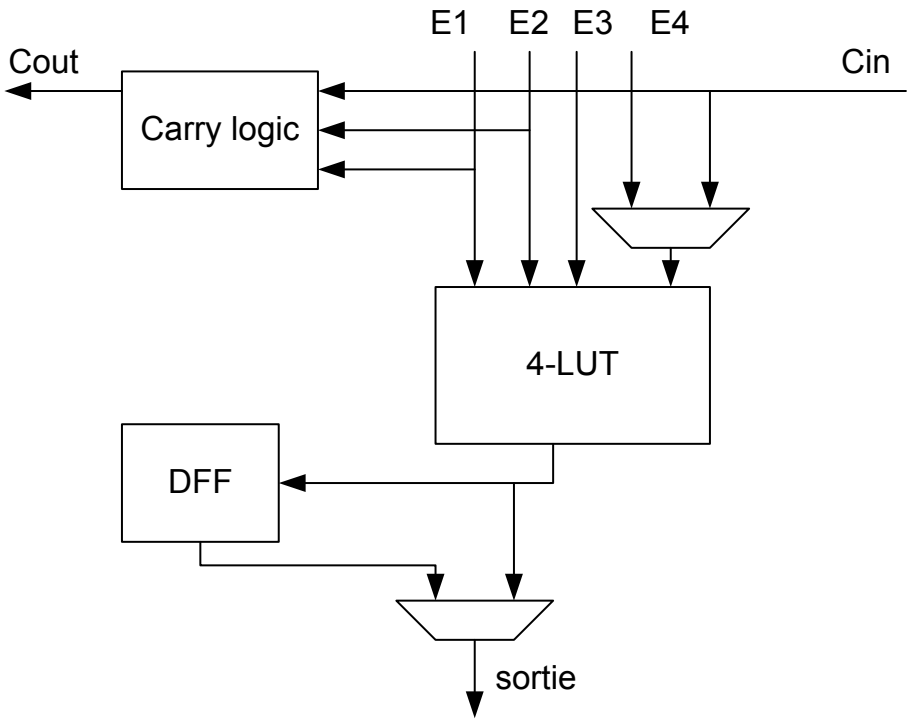


FIGURE 3.6 – Détail d'un bloc logique

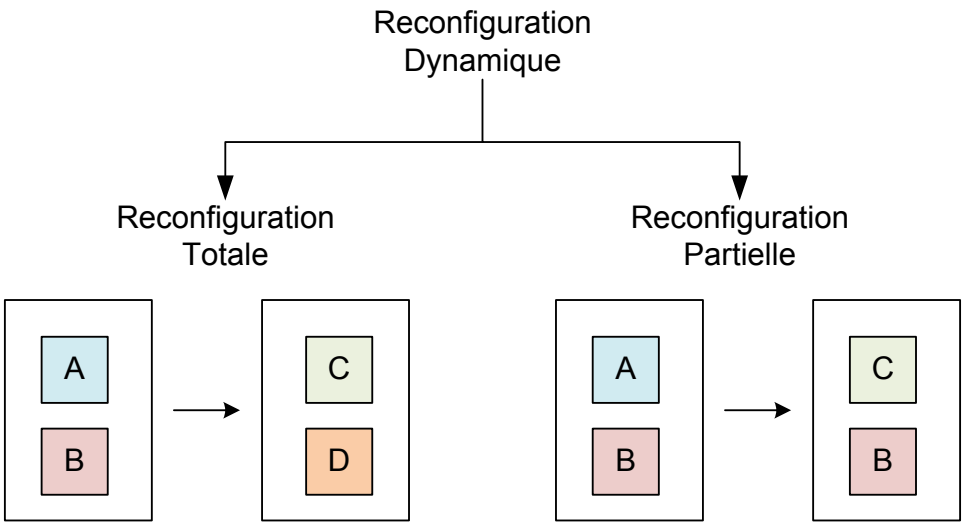


FIGURE 3.7 – Mode de reconfiguration d'un FPGA

de pipeline comme l'illustre la figure 3.8

En termes d'occupation du circuit, l'implémentation occupe 12000 *slices* d'un FPGA Virtex II Pro. Afin d'améliorer les performances de l'architecture en temps d'exécution,

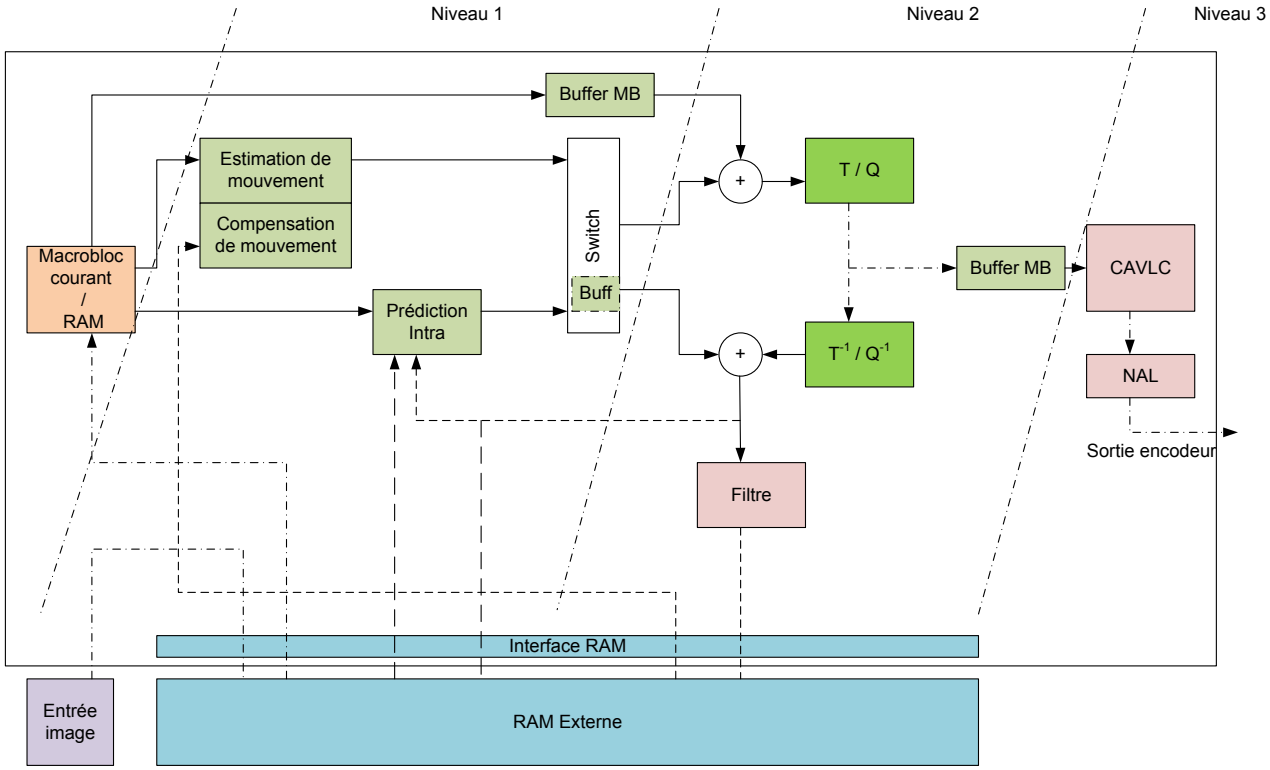


FIGURE 3.8 – Architecture de l'encodeur

les auteurs ont réalisé une implémentation VLSI SoC sur technologie UMC (Une compagnie de fabrication de semi-conducteur) à 0,18 micron.

3.4 Les systèmes sur puce

Les *Sytems on Chip* ou SoC ou systèmes sur puce sont des systèmes complets de type ASIC embarqués sur une puce qui peuvent contenir de la mémoire, un ou plusieurs processeurs, des interfaces, des accélérateurs matériels ou d'autre éléments amenant à la réalisation d'une fonction donnée. Dans le domaine du transcodage, il existe plusieurs SoC commerciaux et notamment l'architecture Mustang de Thomson [Hen08]. Ce SoC est dédié à l'encodage et au décodage, mais également au transcodage. L'architecture interne de ce circuit est illustré en figure 3.9.

Les architectures basées uniquement sur des DSPs ont des problèmes de bande passante et manque d'intercommunication. L'architecture présentée ici contient en plus

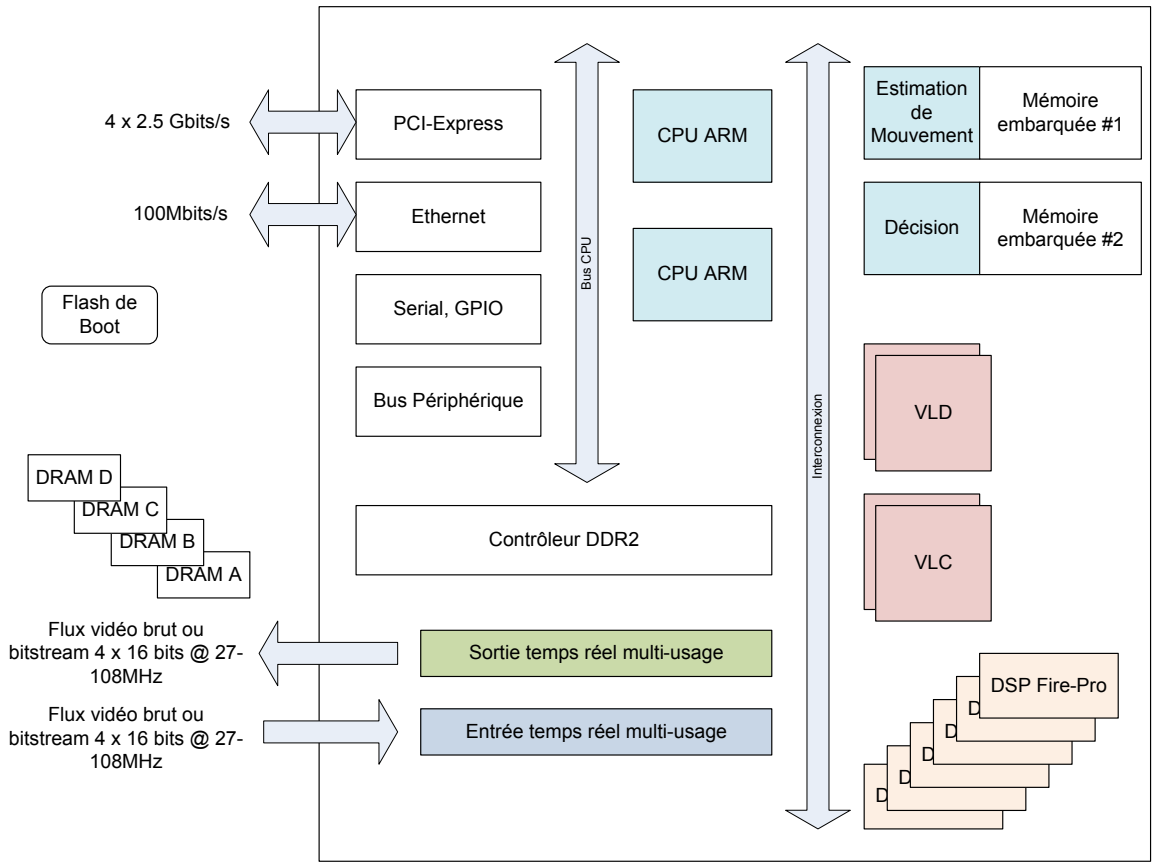


FIGURE 3.9 – Architecture Mustang

des blocs dédiés à certaines tâches lourdes de l'encodage H.264 qui représente une multiplication par 10 de la complexité face au standard MPEG-2. Quatre blocs de DRAM sont utilisés pour le stockage temporaire des données reliées à des bus 32 bits fonctionnant à 533MHz, ce qui équivaut à une bande passante totale de 32Gbits/s. On trouve également un groupe de 6 DSPs dédiés au traitement vidéo qui embarque des fonctions telles que la transformée entière de H.264 qui est alors réalisée en seulement deux cycles, sachant que chaque unité travaille à 333MHz. Ce circuit est dit évolutif du fait de la présence de ces DSP programmables. Cette flexibilité est somme toute restreinte du fait de la spécificité des accélérateurs matériels présents dans le circuit et des DSPs programmables mais fortement orientés vers un standard donné. Le circuit contient également des blocs dédiés à l'encodage et au décodage entropique incluant CAVLC et CABAC. Chacun des blocs d'encodage et de décodage est doublé pour réalisé plu-

sieurs tâches en parallèle. Deux processeurs ARM à 533MHz sont également présents, le premier à des fins de contrôle du circuit, le second pour effectuer des calculs sur le flux vidéo et contrôler le débit.

D'autres constructeurs proposent également des SoC d'encodage/transcodage de MPEG-2 vers H.264. C'est le cas de Fujitsu [Fuj09] qui, sans proposer de fonctionnalité ou de performances plus intéressante que le précédent circuit, semble avoir mis l'accent sur une consommation d'énergie réduite pour une utilisation dans des équipements mobiles (1W lors de processus de transcodage les plus complexes réalisables, selon le constructeur). Ce circuit a une fréquence de travail de 216MHz. Il n'est par contre visiblement encore moins flexible que le précédent, aucune programmation interne ne pouvant être effectuée.

Ces circuits, hautement spécialisés, sont très efficaces en termes de capacité de calcul et de traitement de l'image haute définition. Leur inconvénient majeur est le manque de flexibilité pour des évolutions des normes de codage futures.

3.5 Conclusion et discussion

Les implémentations les plus flexibles, reprogrammables, telles que les implémentations sur GPP ou DSP dont le code peut être mis à jour de façon simple, ne sont toutefois pas les plus performantes et avec l'avènement de résolutions toujours plus élevées et de quantité de données à traiter également en constante hausse, ces implémentations ne peuvent rivaliser pour le traitement temps réel, que ce soit pour des opérations d'encodage ou de transcodage. D'un autre côté, on trouve des solutions très performantes, essentiellement chez les industriels. Ces solutions ont l'inconvénient d'être relativement figées de par leur conception du type ASIC. Même si on voit émerger des SoCs qui embarquent des processeurs (généraux et dédiés) et donc un peu plus de flexibilité, la présence d'accélérateurs matériels totalement figés ne promettent pas une évolutivité à long terme. Il semble alors intéressant de s'orienter vers ce problème de flexibilité. Le chapitre suivant ouvre la voie à une architecture efficace permettant le traitement de scénarios de transcodage et qui apporte une réelle évolutivité dans le

temps.

Chapitre 4

Architecture proposée pour le transcodage

Sommaire

4.1 Introduction

Dans le chapitre précédent, les architectures de transcodage existantes ont été survolées et, grâce à cet état de l'art, des choix architecturaux ont été faits afin d'obtenir un bon compromis entre qualité du transcodage et complexité réduite de mise en oeuvre. Ces choix sont explicités dans ce chapitre.

Le but de cette première architecture présentée ici est confiné au transcodage homogène de H.264, qui, en partant d'un flux SVC à plusieurs couches de réhaussement, permet de produire un flux AVC ne comportant qu'une couche. Cette transposition est appelée *AVC rewriting* et est prévue dans la norme SVC. Les travaux de ce chapitre ont porté sur l'optimisation matérielle de cette phase de réécriture.

4.2 Description de l'architecture proposée

4.2.1 Modèle de l'architecture

La figure 4.1 donne une vue globale de l'architecture proposée. Cette architecture consiste en plusieurs modules, certains statiques, d'autres reconfigurables (RPM). Les modules statiques incluent des fonctions de paquetisation, d'analyseur et constructeur de NALU, de gestion de l'interface de reconfiguration (ICAP) et un processeur, qui dans notre cas est un Microblaze de Xilinx. Dans cette partie statique, le flux vidéo est dépaquétisé et stocké en SDRAM dans le format NALU. Le parser (analyseur) se charge ensuite d'effectuer une lecture des NAL successives et les envoient au décodeur adéquat selon qu'elles appartiennent à la couche de base ou à une couche de réhaussement. Des informations additionnelles sont également communiquées au processeur qui se charge de gérer la reconfiguration des zones reconfigurables à travers l'ICAP du système. Cette étape de reconfiguration est détaillée plus loin. Le flux vidéo transcodé est ensuite paquétisé après une reconstruction classique des NALU. La zone reconfigurable est organisée en modules reconfigurables de différentes tailles. Le but de cette organisation en zones séparables est d'apporter une flexibilité à l'architecture globale. Grâce à cette organisation, les tâches nécessaires à la réalisation des différents scéna-

rios de transcodage sont implémentées de manière modulaire. Une forme de décodage pour un type de couche SVC peut être aisément remplacée par une autre forme dédiée à la lecture d'un autre type de couche.

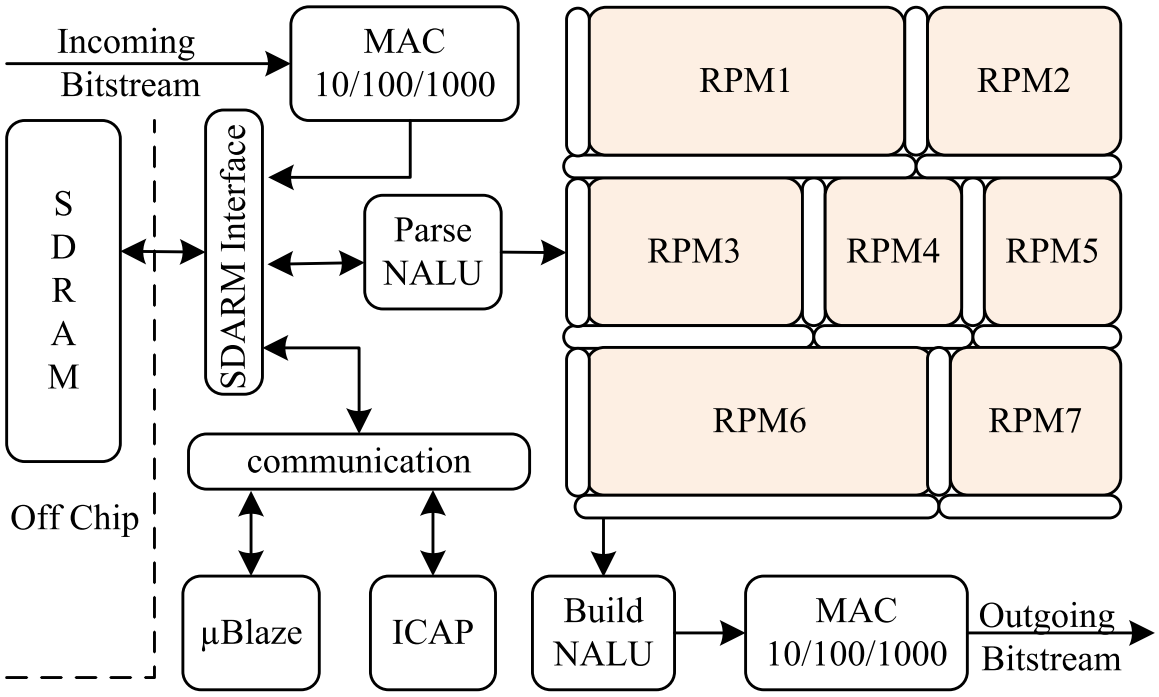


FIGURE 4.1 – Architecture reconfigurable proposée pour le transcodage de SVC vers AVC

L'architecture proposée est conçue de façon modulaire et peut aisément s'adapter à différentes entrées et différents besoins en sortie qui requièrent, dans le cas présenté ici, un débit binaire ou une résolution spatiale donnés. Le transcodage est réalisé ici au niveau pixel et les fonctions de décodage entropique, quantification inverse et transformation inverse sont donc nécessaires. L'architecture complète est également pipelinée et cette organisation particulière engendre le besoin d'utiliser des buffers entre les principales fonctions du standard H.264. CAVLC, la quantification et la transformée inverse agissent sur des blocs de pixels de taille 4x4. La prochaine section détaille les opérations de décodage SVC et de réencodage AVC.

La figure 4.2 montre les différentes partitions qu'il est possible d'implémenter dans l'architecture et réalisant chacune une action précise sur le flux vidéo. Ces partitions

particulières sont développées afin de répondre aux caractéristiques des scénarios définis plus haut. Il faut toutefois savoir que le nombre de partitions n'est pas limité à celle présentée dans le schéma. Le nombre de partition, ainsi que leur type dépend du scénario SVC (nombre et type de couches de réhaussement). La première partition implémentée ici (M1) correspond à une partie du décodage de la couche de base qui est compatible AVC.

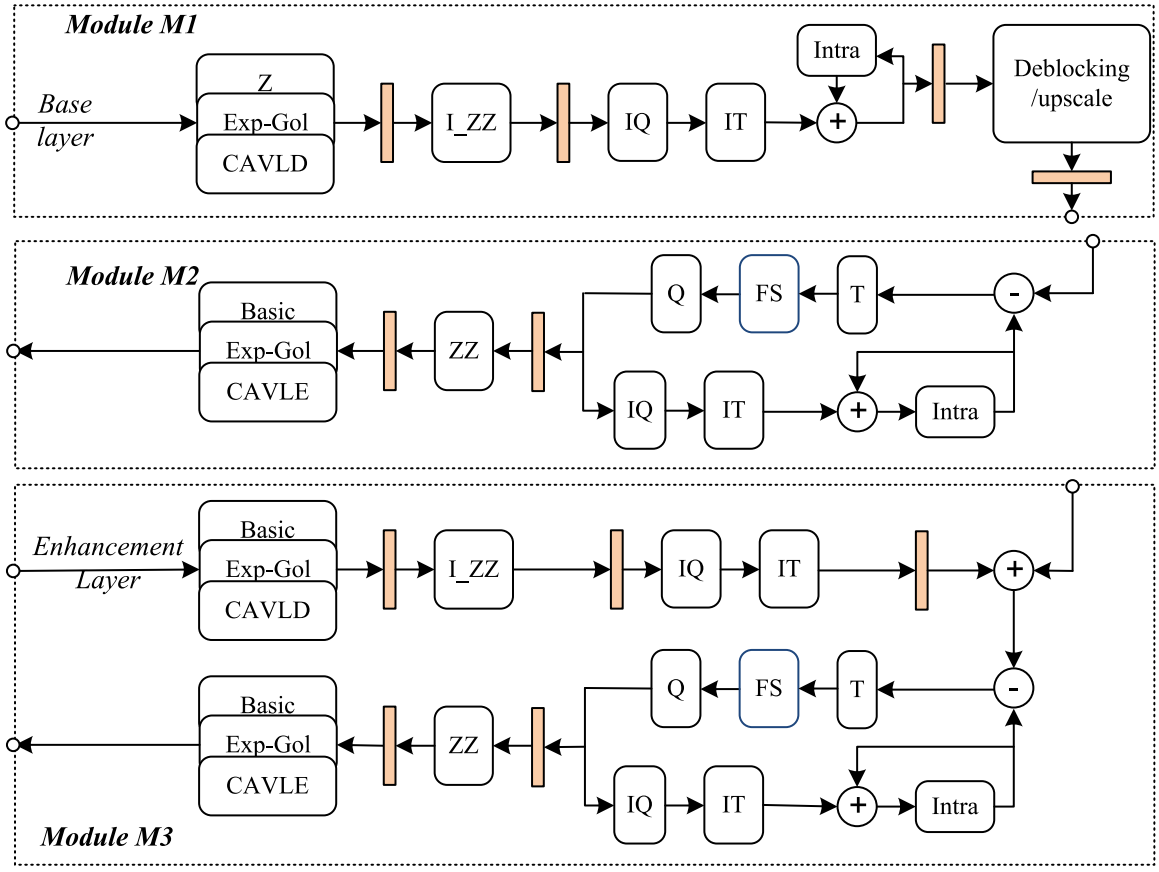


FIGURE 4.2 – Partitions disponibles

La figure 4.3 montre les étapes d'ordonnancement du traitement des blocs résiduels (B*i*). On note la présence d'une forte latence en raison de l'utilisation dans la prédiction intra du résultat des dernières opérations de transformée/quantification inverse. De ce fait, nous verrons plus loin qu'il est indispensable de veiller à réduire la latence au sein de chaque module traversé par les blocs. Le bitstream est d'abord lu à partir de la mémoire externe. Un bus PLB (Processor Local Bus) est ensuite utilisé pour

transmettre les informations à l'analyseur qui sépare les NALU qui contiennent des données différentes. Chaque tranche de donnée est ensuite stockée temporairement dans une FIFO contrôlée par une machine à états finis. Un contrôleur analyse le flux vidéo. Le décodeur entropique décode ensuite le bitstream. Le but de ce décodeur est de reconstruire les blocs 4x4 à partir des mots codes utilisés dans le codage entropique. Ces blocs 4x4 sont alors au niveau coefficients quantifiés et sont alignés en zigzag. Pour retrouver la forme du bloc original, un processus de zigzag inverse est réalisé. Ce nouvel agencement nécessite un buffer intermédiaire pour stocker les valeurs des coefficients. La quantification inverse est ensuite réalisée sur chaque coefficient du bloc issu du décodage entropique. H.264 introduit un nouveau type de quantification qui prend en compte un paramètre de quantification Q_p mais aussi la position de chaque coefficient dans le bloc. La matrice de quantification inverse est définie au début du processus. L'implémentation de la transformée entière décrite par [MHKK03] est utilisée. Une implémentation de la transformée inverse dite *butterfly* permet de n'avoir recourt qu'à des additionneurs et des décalages. La transformée est réalisée en deux passes : la première est faite sur les colonnes alors que la seconde sur les rangs. Dans l'implémentation proposée, les deux passes sont réalisées en un seul cycle d'horloge. Après cette transformation inverse, les résidus représentant la différence entre une prédiction et une image originale sont obtenus. La prédiction Intra peut alors être lancée de la même façon qu'elle a été réalisée à l'encodeur. Cette prédiction est exécutée sur l'addition des blocs résiduels et de la prédiction du bloc courant. Comme ces opérations sont toutes effectuées sur des blocs de l'image (4x4 ou 8x8 pour la prédiction Intra), le standard recourt à un filtre de déblocage pour atténuer les effets de bloc dans l'image décodée.

Le module reconfigurable de décodage est construit sur le même modèle mais certains détails dépendent du type de la couche de réhaussement visée. Dans l'exemple donné ici, illustré par la figure 4.4, une couche de réhaussement en qualité est décodée dans la première zone reconfigurable. Ce décodage utilise les mêmes étapes que le décodage de la couche de base. Le résultat du décodage de la couche de réhaussement peut ensuite être ajouté à celui de la couche de base pour obtenir une vidéo de

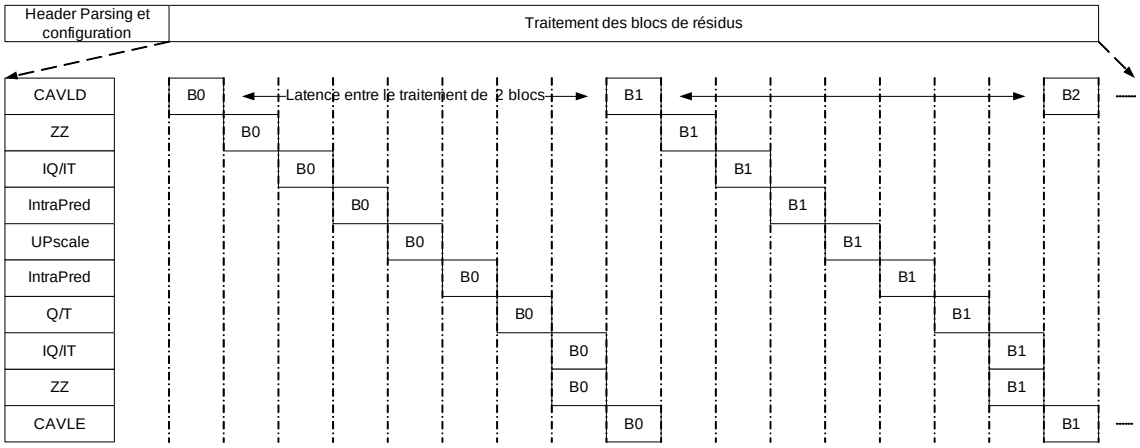


FIGURE 4.3 – Ordonnancement du traitement des blocs résiduels

meilleure qualité.

Pour la prédiction inter images, comme les informations de mouvement peuvent être réutilisées dans le processus de transcodage, il n'est pas nécessaire de recalculer ces vecteurs. Dans le cas d'une réduction de débit binaire, les informations de mouvement sont simplement ajoutées au bitstream final après les image intra dégradées. Dans le cas d'une réduction de la résolution spatiale, le processus est un peu plus complexe car les informations de mouvement doivent être ajustées en fonction de la nouvelle résolution. Cependant le calcul est beaucoup plus simple qu'au niveau de l'encodeur [NLCC07].

Le processus d'encodage dépend des besoins en sortie (selon le terminal de visualisation utilisé et la bande passante disponible). Cette partie de réécriture du flux est placée dans une des zones reconfigurables et peut également être modifiée en cours de fonctionnement lors d'un changement des caractéristiques de l'environnement. Ceci inclut différente méthodes pour adapter le flux vidéo parmi lesquelles la requantification complète ou la sélection fréquentielle [DCB+09] qui agit sur la qualité de l'image. Ces méthodes permettent de réduire le débit binaire du flux vidéo pour l'adapter au canal de transmission.

4.2.2 Modèle de l'environnement

L'environnement dans lequel le transcodeur prend place est illustré sur la figure 4.4.

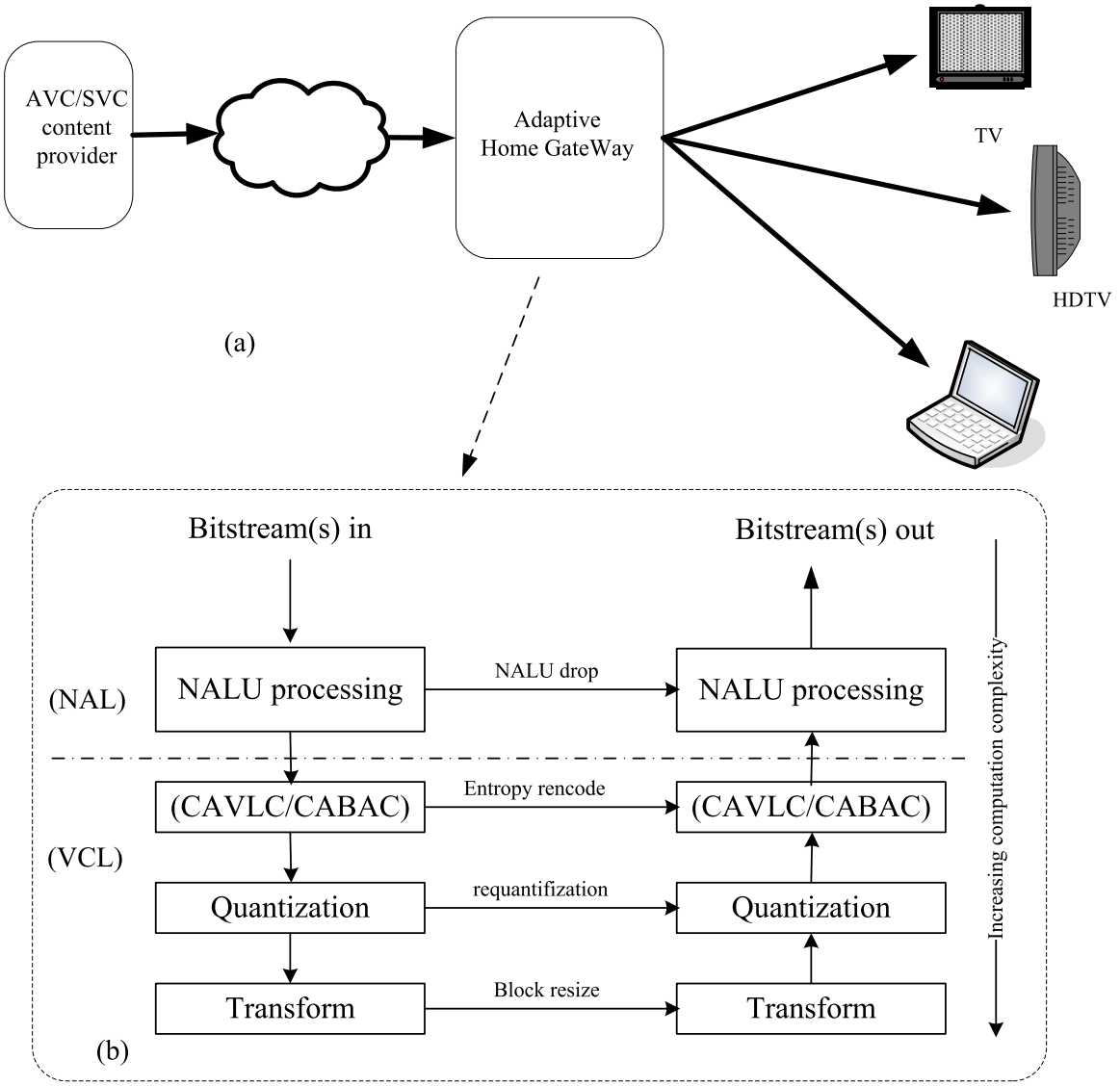


FIGURE 4.4 – Système d'adaptation vidéo : (a) architecture globale, (b) Différents niveaux d'adaptation dans H.264/AVC-SVC.

4.2.3 Décision d'adaptation

Certains des modules de décodage et d'encodage ont été développés en utilisant des outils propriétaires. Ces modules sont synthétisés séparément et leur bitstream de configuration (à ne pas confondre avec le bitstream du flux vidéo) est stocké dans une mémoire externe. Chacun d'eux peuvent être implémenté en cours de fonctionnement dans le FPGA par l'intermédiaire du processeur présent dans le système et l'interface de reconfiguration ICAP [PK06]. L'analyseur de flux vidéo contenu dans le système contient en plus des fonctionnalités exigées par le standard H.264, une partie de contrôle permettant l'envoi d'informations au microprocesseur. L'analyseur détecte ainsi les différentes couches utilisées dans le flux et peut communiquer ces informations au microprocesseur. Ce dernier, en croisant ces information est les besoins en sortie du transcodeur choisit les modules matériels à mettre en oeuvre pour le transcodage.

Deux implémentations différentes sont proposées afin de transcoder un flux AVC et un flux SVC. La première proposait une réduction du débit binaire permettant d'adapter le flux à un canal de transmission, la seconde permet, grâce à une certaine flexibilité, une rétrocompatibilité complète (et non plus uniquement de la couche de base) entre un flux SVC et un lecteur purement AVC.

4.3 Concepts mathématiques

La partie qui suit est consacrée à la présentation des fonctionnalités du transcodage sous leur forme mathématique. Cette présentation est tout d'abord nécessaire pour exposer les principes utilisés pour le traitement de la vidéo, et ensuite pour expliquer leurs implémentations respectives. Dans la majorité des standards de compression, l'estimation du mouvement puis la compensation de celui-ci est l'opération la plus complexe. Cependant, dans de récents travaux, on trouve des techniques de transcodage qui permettent de réutiliser les informations des vecteurs de mouvement contenues dans le flux original. Comme nous l'avons démontré dans la première partie de

ce rapport, la problématique n'est pas nouvelle et est apparue dés la fin des années 90, mais de récents travaux ont pu prouver qu'il était possible d'améliorer encore les résultats de transcodage. C'est le cas dans les travaux suivants [LM07, MKF+09, CRG+09].

Parmi ces techniques modernes, il existe des méthodes qui permettent d'estimer les vecteurs de mouvement pour une réduction arbitraire de la résolution spatiale de la vidéo. Ces méthodes vont même jusqu'à proposer des résultats en terme de qualité d'image qui concurrencent directement les estimations de mouvement à recherche complète, mais ceci avec une complexité de calcul amoindrie. Cet aspect n'est pas l'objet direct des travaux de thèse présentés ici, mais leur connaissance permettra plus tard de réfléchir à de nouvelles perspectives. Dans ce chapitre seront détaillées les fonctionnalités basiques du transcodage vidéo ainsi que leur implémentation. Il s'agit de la transformée entière, de la quantification, de la sélection fréquentielle, de la prédiction Intra et du codage entropique, en l'occurrence CAVLC.

4.3.1 Transformée en cosinus et transformée entière

La transformée en cosinus discret (DCT) est une des transformations les plus populaire dans de nombreux standards de compression vidéo. Le dernier né des standards utilise quant à lui une transformation dérivée de cette DCT, qui est de complexité inférieure, il s'agit de la transformée entière. Cette transformée est directement liée à la DCT à deux dimensions, mais n'utilise que des coefficients entiers. Les coefficients résultant de la transformation entière seront donc également de type entier. Et afin de conserver cette caractéristique, il existe des facteurs de mise à l'échelle, ceux-ci étant inclus dans les processus de quantification directe et inverse comme nous le verrons un peu plus loin.

La transformée en cosinus discret de $X(i,j)$, d'un bloc de taille $N \times M$, est définie comme suit :

$$Y(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} X(i,j)C(i,u)C(j,v) \tag{4.1}$$

où

$$C(p,q) = \begin{cases} \frac{1}{\sqrt{N}} & , q = 0 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2p+1)q\pi}{2N}\right) & , q \neq 0 \end{cases} \tag{4.2}$$

La simplification de ces expressions considérant que $N = M = 4$ prend la forme suivante :

$$Y = (C\,X\,C^T) \odot E_f \tag{4.3}$$

où C et E_f sont donnés par :

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \tag{4.4}$$

$$E_f = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} \tag{4.5}$$

et

$$a = \frac{1}{2} \;\; et \;\; b = \sqrt{\frac{1}{2}} \cos(\frac{\pi}{2}) \tag{4.6}$$

Le symbole $\odot$ signifie que chaque élément de $(C\,X\,C^T)$ est multiplié par le facteur de mise à l'échelle qui se trouve dans la même position dans la matrice E_f. C^T est la transposée de C.

La transformée à deux dimensions $4x4$ est calculée en deux passes. Une première passe est effectuée en ligne sur une dimension, et une seconde, toujours à une dimension est effectuée en colonne. L'algorithme 1 donne le pseudo-code pour une transformée en cosinus discret à deux dimensions.

L'architecture traditionnelle dite *butterfly* (papillon) pour la transformée à une dimension est utilisée dans cet algorithme.

Algorithm 1 La transformée directe en cosinus discret à deux dimensions

1: **for** $row = 1$ to $Nbrows$ **do**
2: `Application de la transformée en ligne`
3: **end for**
4: **for** $col = 1$ to $Nbcols$ **do**
5: `Application de la transformée en colonne`
6: **end for**

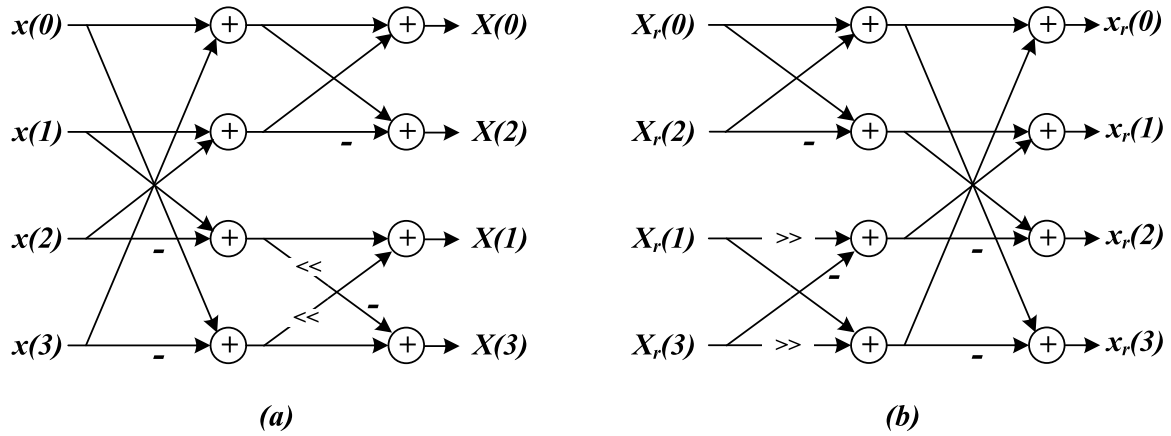


FIGURE 4.5 – Transformée entière directe(a) et inverse (b) à une dimension

4.3.2 Quantification

La transformation entière permet une classification fréquentielle des coefficients se trouvant dans un même bloc. Ceci implique une dynamique importante dans les valeurs des coefficients résultant de cette transformée. Un des objectifs de la quantification est alors de réduire cette importante dynamique tout en offrant un bon compromis en minimisant les pertes engendrées par ce troncage des coefficients. La force de la quantification d'H.264 tient uniquement dans le paramètre de quantification Qp. L'équation 4.7 donne la forme mathématique de cette quantification où $X(i,j)$ et $X_q(i,j)$ représentent respectivement les coefficients issus de la transformée et ces mêmes coefficients quantifiés. La formule utilisés dans [MHKK03] est utilisée ici.

$$
\begin{aligned}
X_q(i,j) = sign(X(i,j) \times [(|X(i,j)| \times A(Q_M, i, j) \\
+f2^{17+Q_E}) \gg (17+Q_E)] \\
where\ \ Q_M = Qp \mod 6 \\
and\ \ Q_E = \frac{Qp}{6}
\end{aligned}
\tag{4.7}
$$

Le paramètre f est compris entre 0 et $1/2$ et est choisi par l'encodeur. Le processus de quantification dépends aussi de deux matrices contenant uniquement des constante (voir la figure 4.8). L'utilisation de ces matrices dépend du paramètre de quantification Qp choisi.

$$
M = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \end{bmatrix}, \quad S = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 16 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 18 & 29 & 23 \end{bmatrix}
\tag{4.8}
$$

Les matrice A et B sont définies dans l'équation 4.9, où la valeur de Q_M est dérivée du paramètre de quantification Qp et indique quelles valeurs contenues dans les matrices seront utilisés dans le processus de quantification. Chaque coefficient quantifié est ainsi obtenu en utilisant les coefficients de A qui occupent la même position. Une opération similaire est effectuée lorsqu'il s'agit de la quantification inverse, mais cette fois avec la matrice B.

$$
A, B = \begin{bmatrix} a & c & a & c \\ c & b & c & b \\ a & c & a & c \\ c & b & c & b \end{bmatrix}
\tag{4.9}
$$

où $a = M(Q_M, 0), b = M(Q_M, 1), c = M(Q_M, 2)$ pour A *et* $a = S(Q_M, 0), b = S(Q_M, 1), c = S(Q_M, 2)$ pour B

La quantification induit théoriquement des divisions et donc une arithmétique à virgule flottante. Afin d'éviter ces opérations complexes, la matrice d'encodage est im-

plémentée sous forme d'une LUT (*Look Up Table*), la matrice M pour l'encodage et la matrice S pour le décodage comme le montre l'équation 4.8. A partir de ces LUT, deux nouvelles matrices sont créées, respectivement A et B et sont construites selon l'équation 4.9.

La quantification inverse au niveau du décodeur utilise également une LUT qui fournit les facteurs de quantification inverse.

L'algorithme 2 montre les principes de quantification dans le standard H.264, avec QP le paramètre de quantification.

Algorithm 2 Quantification

$Q_M = Q_p mod 6$

$Q_E = \frac{Q_p}{6}$

$a = M(Q_M, 0)$

$b = M(Q_M, 1)$

$c = M(Q_M, 2)$

$$A = \begin{bmatrix} a & c & a & c \\ c & b & c & b \\ a & c & a & c \\ c & b & c & b \end{bmatrix}$$

for $i = 0$ to 3 **do**

  for $j = 0$ to 3 **do**

    $X_q(i,j) = sign(X(i,j)) \times \left[\left(|X(i,j)| \times A(i,j) + f2^{17+Q_E}\right) >> (17 + Q_E)\right]$

  end for

end for

4.3.3 Sélection fréquentielle

Le transcodage par sélection fréquentielle (également connue sous le nom de troncature) consiste en une élimination de certains coefficients représentant des hautes fréquences contenues dans le bloc en cours de traitement. Ce type de transcodage permet d'ajuster le débit binaire selon la bande passante disponible. Dans certains cas, la sé-

lection fréquentielle est préférée à la simple requantification car elle n'induit jamais de fourmillement de l'image contrairement à la requantification [DCB+09]. Même si cette dernière est souvent plus interessante en terme de rapport signal sur bruit (ou PSNR) et en terme de réduction du débit, la sélection fréquentielle est utilisée pour éviter une dégradation de la qualité subjective de la séquence vidéo. Cette transformation engendre une élimination des détails fins de l'image, mais sa complexité est très faible.

La sélection fréquentielle se contente d'annuler un certain nombre de coefficients en commençant toujours par les coefficients de plus haute fréquence. Ce nombre est déterminé par un seuil qui varie de 0 (aucun coefficient n'est modifié) à 16 (tous les coefficients sont mis à zéros) pour un traitement sur des blocs 4×4. L'algorithme 3 donne le détail de cette procédure. Dans cet algorithme, $coeff\_pos$ correspond à la position des coefficients en partant du premier coefficient en haut à gauche du bloc et suivant l'ordre zig-zag classique. Le paramètre fs détermine quant à lui le seuil de la sélection. new_coeff correspond à la nouvelle valeur du coefficient alors que $coeff$ correspond au coefficient actuel.

Algorithm 3 Sélection Fréquentielle

1: **for** $coeff\_pos = 1$ to 16 **do**
2: **if** $coeff\_pos > fs$ **then**
3: $new\_coeff \leftarrow 0$
4: **else**
5: $new\_coeff \leftarrow coeff$
6: **end if**
7: **end for**

4.3.4 Prédiction Intra

L'objectif de la prédiction Intra est de créer des macroblocs aussi similaires que possible aux macroblocs originaux. La prédiction est effectuée à partir des blocs déjà encodés puis décodés en utilisant les pixels adjacents au bloc en cours de prédiction. Il

existe 9 types de prédiction pour les blocs 4×4 et quatre modes pour les blocs 16×16. Ces modes sont décrits dans la figure 4.6. Le mode choisi pour la prédiction sera celui capable de minimiser la différence entre le bloc prédit et le bloc original. Ainsi les valeurs des résidus (différence entre l'image prédite et l'image originale) s'approchent de zéro et l'encodage entropique gagne en efficacité.

Dans le standard H.264, la prédiction Intra permet deux manière de codage : sur des blocs 16×16 et sur des blocs 4×4. Chacun de ces groupes utilise un mode de prédiction basé sur un schéma donné utilisant les pixels voisins en luminance et en chrominance. L'intra 16×16 est utilisé pour prédire des zones dans lesquelles on trouve peu de détail alors que la prédiction par bloc 4×4 est utilisée là où on trouve les plus hautes fréquences. Un type de prédiction est utilisée pour tout un macrobloc. Pour la prédiction 16×16, quatre modes sont possibles : les modes vertical, horizontal, DC et plan. Chacun d'eux utilise les pixels voisins du bloc courant pour le prédire. La décision du mode de prédiction est basée sur le calcul de la somme des différences absolues (SAD : *Sum of Absolute Differences*) entre le macrobloc prédit et le macrobloc original. Le mode qui permettra de minimiser la SAD sera donc retenu.

Pour les blocs 4×4, il existe neuf modes de prédiction qui sont détaillés dans la figure 4.6

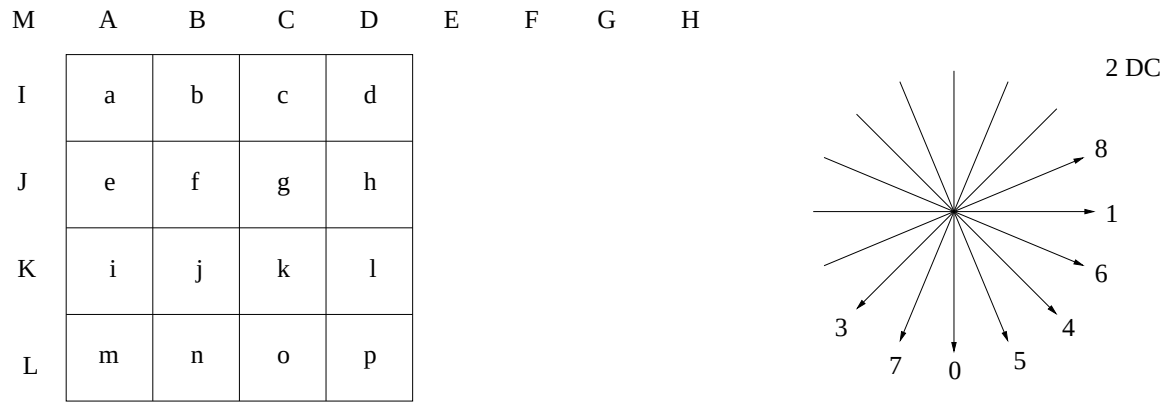


FIGURE 4.6 – Modes disponibles pour la prédiction Intra

Dans le cas du transcodage, le meilleur mode de prédiction n'est pas obligatoirement recalculé. La recherche étant faite à l'encodeur, le transcodeur peut se contenter

de récupérer le mode utilisé pour le macrobloc courant et de le réutiliser. Le mode de prédiction utilisé est bien sûr encodé dans l'entête du macrobloc au niveau du bitstream et peut aisément être lu par le système de transcodage [BKI04]. La prédiction Intra, au niveau de l'implémentation est paramétrable et peut exécuter tout type de mode sans être reconfigurée. Dans cette implémentation, la prédiction intra est utilisé au niveau du transcodeur afin d'éviter les problèmes de *drift* [Leo08], qui peuvent réduire considérablement la qualité de l'image.

4.3.5 Codeur entropique CAVLC

CAVLC est l'acronyme de Context-based Adaptive Variable Length Coding. Ce codeur entropique utilise un encodage à longueur variable et joue un rôle très important dans la compression d'image. Les éléments qui apparaissent le plus souvent au niveau des coefficients à encoder se voient assigner des mots-codes binaires plus courts alors que les éléments rares sont associés à des mots codes plus longs. CAVLC est dit adaptatif car l'encodage du bloc courant dépend des données contenues dans les blocs encodés précédemment. Plus précisément dans les blocs qui entourent spatialement le bloc courant. Ces sont des informations extraites du bloc supérieur et du bloc de gauche qui sont retenues pour déterminer l'encodage du bloc courant. En effet, dans une image, des blocs proches au niveau spatial ont de grande chance d'être similaire, le codeur entropique s'adapte donc réellement en essayant de prédire la meilleure façon d'encoder un bloc en fonction des blocs précédents. Un autre type d'adaptation apparait dans CAVLC, alors qu'un bloc est en cours d'encodage. Une adaptation au niveau des valeurs contenues dans le bloc elle même est effectuée. Le mot-code attribué à un coefficient dépend en effet de la valeur du coefficient précédent, ou, dans le cas du premier coefficient, du nombre de coefficients non-nuls et du nombre de coefficients égaux à 1 ou -1 dans le bloc en cours d'encodage.

Généralement, CAVLC encode chaque bloc en cinq étapes indépendantes. Toutes ces étapes génèrent des éléments syntaxiques. Ces éléments seront ensuite concaténés afin d'obtenir une représentation compacte d'un bloc et de l'injecter dans le bitstream

sous forme d'une séquence binaire. La première étape permet de générer l'élément syntaxique appelé "`CoeffToken`" qui encode à la fois le nombre total de coefficients non-nuls "`TotalCoeffs`" à l'intérieur du bloc et le nombre de coefficients égaux à 1 ou -1 ("`TrailingOnes`"). Cet élément dépend également du nombre de coefficients non-nuls dans les blocs adjacents (ce qui représente encore une forme d'adaptation). La seconde étape génère un élément syntaxique qui code à la fois le nombre de coefficients égaux à 1 ou -1 (dans la limite de 3) et leur signe respectif. L'éléments "`TotalZeros`" est une représentation de la quantité de valeurs nulles qui précèdent le dernier coefficient non nul. Après avoir extrait ces informations intervient l'encodage des valeurs de tous les coefficients non nuls présents dans le bloc. Dans cette étape, l'architecture proposée plus loin améliore la fonction de codage entropique en remplaçant les classiques LUT décrites dans la norme [H26] par un calcul simple fonction de la valeur absolue du coefficient et de son signe. La prochaine étape permet la représentation compacte du nombre de zéros qui apparait devant chaque coefficient non-nul. Cette étape est appelée "`RunBefore`" et est effectuée selon l'ordre zig-zag classique d'ordonnancement des coefficients. L'algorithme 4 représente le pseudo-code des étapes décrites ici pour l'encodage entropique.

CAVLC permet un encodage sans perte des valeurs issues de la quantification des coefficients du domaine fréquentiel. Le taux de compression

CAVLC est le codeur entropique d'H.264 pour le profile de base. Il tire son avantage de plusieurs caractéristiques qui apparaissent le plus fréquemment dans les coefficients de blocs à encoder. Pour une implémentation efficace de ce codeur, une architecture parallèle a été choisie pour optimiser les temps d'exécution. En effet, l'architecture présentée plus loin se doit de réaliser un encodage en temps réel.

4.4 Complexité Arithmétique

La complexité arithmétique d'une fonction permet d'appréhender les difficultés d'implémentation et les besoins en ressources de ladite fonction. Certaines des fonctions vues précédemment sont extrêmement simples alors que d'autres sont caracté-

Algorithm 4 Context-based Adaptive VLC

```
1: for each quantized bloc do
2:   Extraction des variables intermédiaire
3:   Encodage de CoeffToken à l'aide de LUT
4:   Encodage de TrailingOnes à l'aide de LUT
5:   Encodage de TotalZeros à l'aide de LUT
6:   for chaque coefficient non-nul do
7:     Calcul de l'index de table
8:     Encodage du niveau du coefficient
9:   end for
10:  for chaque coefficient non-nul do
11:    Encodage de RunBefore
12:  end for
13: end for
```

risées par une complexité accrue, réclamant un effort d'implémentation en termes de parallélisme, de pipeline. Lors de transformation par bloc de 4x4 pixels, la complexité en termes de nombre d'opérations dans la table 4.1.

L'encodage entropique demande une quantité de ressources particulièrement importante face aux autres fonctions. La complexité élevée est due aux différentes itérations réalisées dans chaque module d'encodage des éléments syntaxiques. Ces boucles déroulées font en effet entrer en jeu un grand nombre d'opérations. Le nombre d'itérations à effectuer dans chaque module dépend du nombre de coefficients nuls dans le bloc en cours d'encodage. Ce phénomène peut engendrer des temps d'exécution différent entre deux blocs différents. Dans les travaux qui suivent est proposée une méthode permettant de fixer le temps d'exécution du traitement d'un bloc et de calculer les éléments syntaxiques en une passe, et non plus un nombre d'itérations variable compris entre 0 et 16 (toujours selon le nombre de coefficients nuls dans le bloc 4x4).

Chaque algorithme de traitement a été développé afin de compter le nombre de chaque opération. Pour donner un exemple, qu'on retrouve dans le tableau 4.1, on

compte 96 additionneurs dans la transformée entière. The comptage est bien sûr réalisé sans optimisation de parallélisation. Dans l'implémentation matérielle, les besoins en ressources matérielles peuvent être modifiées en réalisant des efforts de parallélisation, ou de pipeline.

TABLE 4.1 – Complexité de calcul des principales fonctions

Opérations	*Additionneurs*	*Décalages*	*Comparateurs*
Transformation entière	96	32	0
Prédiction Intra	59	10	6
Quantification	15	0	0
CAVLC	111	4	72
Sélection fréquentielle	1	0	1
Opérations	*Multiplieurs*	*Diviseurs*	*Total*
Transformation entière	0	0	128
Prédiction Intra	0	0	75
Quantification	9	0	24
CAVLC	16	2	203
Sélection fréquentielle	0	0	2

4.5 Architecture globale du transcodeur

Cette section décrit globalement l'architecture du transcodeur. Ce transcodeur comprend un décodeur, un module d'adaptation suivi d'un réencodeur. La figure 4.7 présente cet aspect. On y distingue l'ensemble des opérations de traitement du flux compressé pour le décodage puis de traitement de l'image afin de réencoder le flux selon les besoins de l'environnement. Le module d'adaptation présenté ici est un module générique abritant diverses fonctions selon les besoins (décimation d'images, adaptation de résolution spatiale ...).

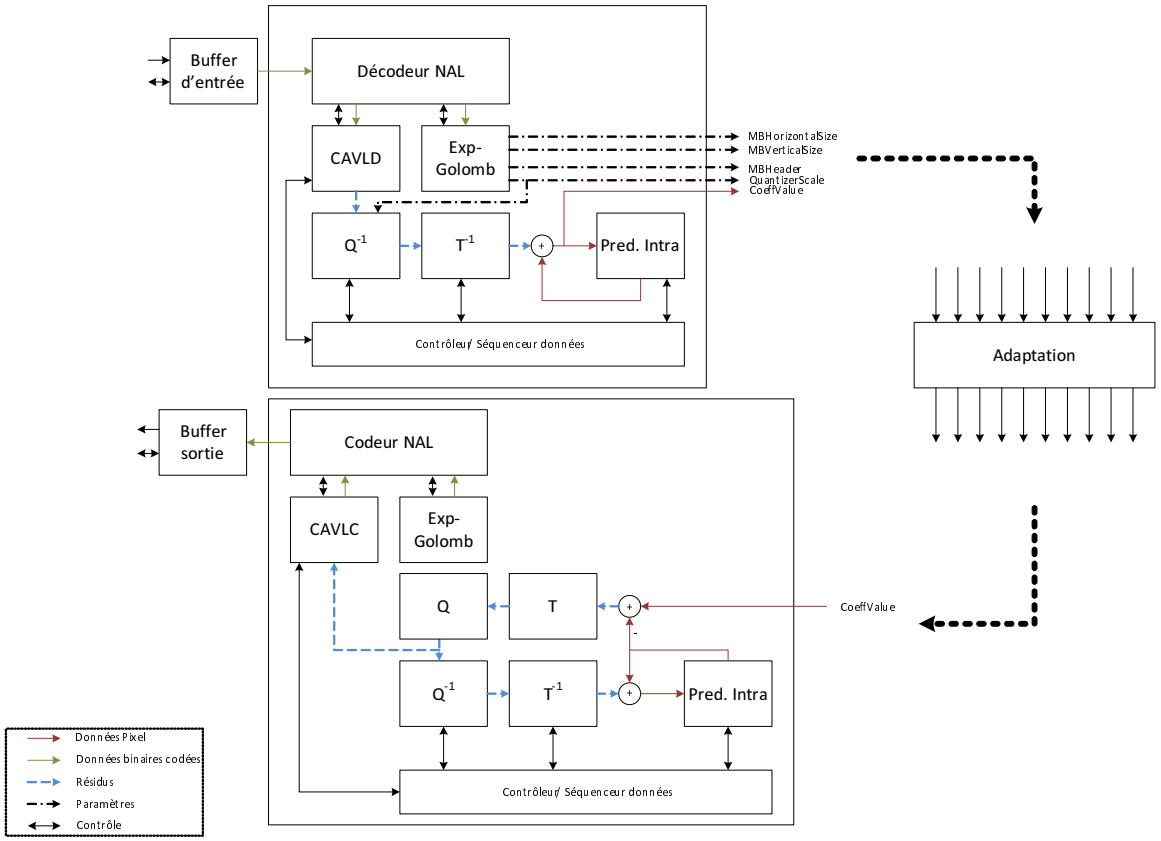


FIGURE 4.7 – Vue globale du transcodeur

Dans les sections suivantes, nous allons détailler les éléments constitutifs de cette architecture que nous avons tenté d'optimiser, du point de vue de l'encodeur ou de celui du décodeur, les deux étant très proches sur la forme.

4.6 Description des modules fonctionnels du transcodeur

Dans cette section nous souhaitons décrire individuellement l'ensemble des composants de l'architecture globale. Nous détaillerons les diverses opérations qui constituent les modules fonctionnels du transcodeur ainsi que la façon de les implémenter sur notre circuit cible.

4.6.1 Etude d'une implémentation classique d'un encodeur Intra

Cette section va nous permettre d'avoir une vue d'ensemble d'un codeur H.264 intra. Nous nous contenterons de présenter un encodeur dans le sens ou un décodeur présente les mêmes modules de fonctionnement car un décodeur (hormis le décodage entropique) est déjà intégré dans l'encodeur. Ce décodeur s'intègre dans une boucle de retour permettant d'établir une image de référence qui sera totalement similaire à l'image de référence obtenue avec un décodeur réel. La figure 4.8 présente l'ensemble du processus de codage, toujours en excluant la partie de prédiction inter.

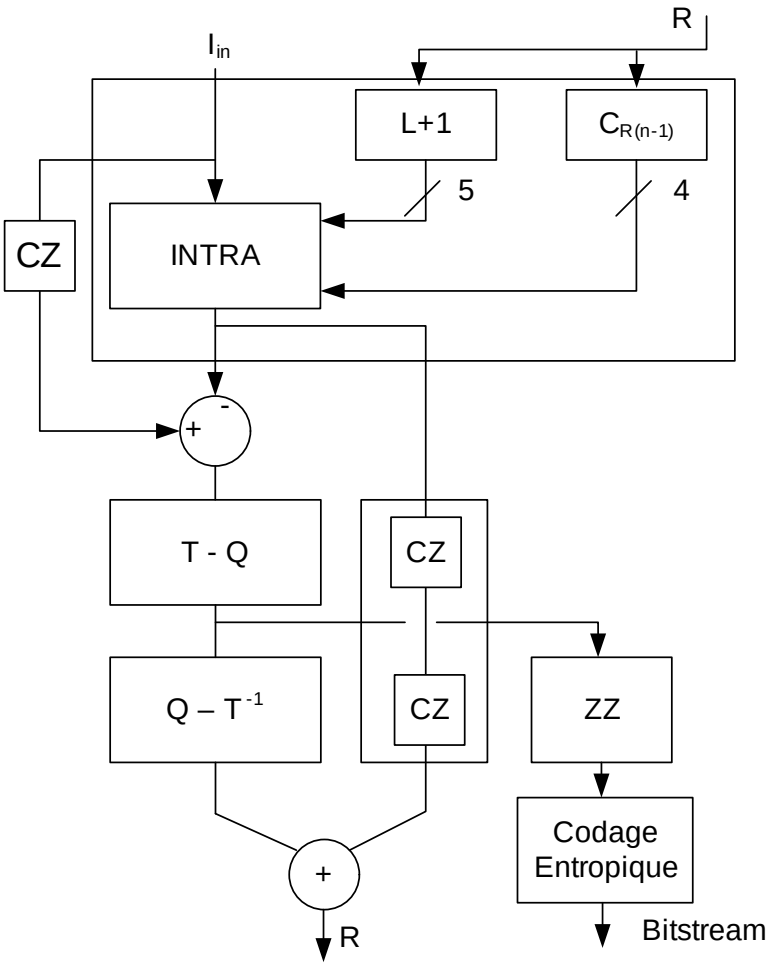


FIGURE 4.8 – Vue globale d'un encodeur intra

Cette implémentation préliminaire, suivant strictement l'implémentation logicielle d'H.264, nous a permis de repérer des zones critiques que nous avons tenté d'optimi-

ser. Dans un premier temps, nous avons constaté que certaines des implémentations séquentielles (puisque suivant l'implémentation logicielle) engendraient une latence importante comme nous avons déjà pu le voir précédemment en ce qui concernait le transcodeur dans sa globalité. En effet, une forte latence apparait entre le traitement de deux blocs distincts. La figure 4.9 illustre la simulation d'une telle architecture. On notera la latence globale de 85 cycles d'horloge pour le traitement d'un seul bloc 4x4. Le traitement du bloc *n* nécessitant le résultat du traitement du bloc *n-1*, cette latence est valable pour tous les blocs et non uniquement le premier.

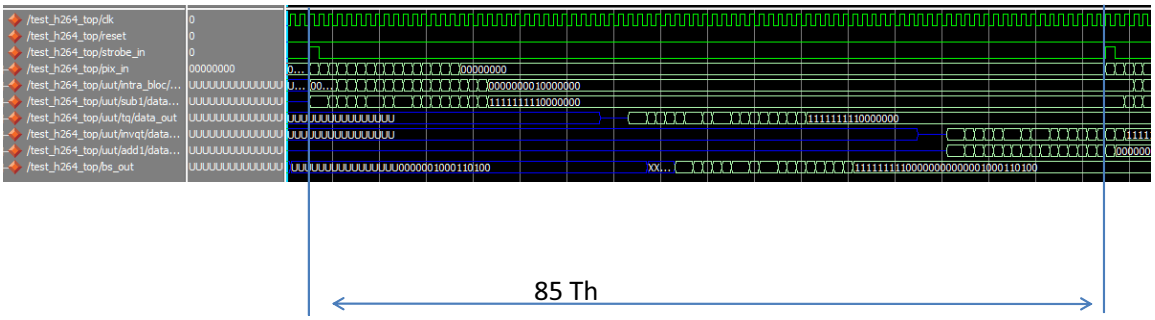


FIGURE 4.9 – Extrait de la simulation de l'ensemble de l'encodeur

Cette latence est du essentiellement aux traitements de prédiction intra, de transformée et de quantification du bloc. Chacun de ces modules sera détaillé dans la suite. Une telle architecture, même si elle est loin d'être optimisée permet néanmoins d'atteindre des performance respectables (tableau 4.2).

TABLE 4.2 – Performance de notre architecture de référence

Résolution d'image	*Fréquence d'horloge*
HD720p (1280x720px)	125MHz
Temps de traitement par image	*Nombre d'images par seconde*
39,168ms	25,5 ips

4.6.2 Transformée et quantification

Plusieurs implémentations sont envisageables. L'implémentation classique est une implémentation sérialisée, construite autour de bus de 8 ou 16 bits permettant de transmettre un pixel ou un coefficient (selon la fonction du bus) à la fois [Kes10,Kor08,Li06]. La majorité des fonctions présentes dans l'encodeur nécessitent la prise en compte de tous les éléments d'un bloc (aussi bien dans le cas des blocs 4x4 que des blocs 8x8 dans le cas de l'encodage intra ou des blocs à dimensions variables dans le cas de la prédiction inter). Comme on peut le constater sur la figure 1, le traitement d'un bloc *n* en début de chaîne nécessite d'avoir au préalable entièrement traité le bloc *n-1*. L'ensemble des fonctions d'H264 ne peut donc être vu comme un pipeline en raison de cette boucle de retour. Ainsi, le temps de traitement d'un seul bloc entraîne une latence pénalisant l'ensemble du système de traitement. Si on souhaite optimiser le temps de traitement (codage ou décodage) il est nécessaire de chercher à réduire la latence de chaque fonction travaillant sur un bloc entier. Chacune des fonctions correspondant à cette description peut faire néanmoins l'objet d'une étude attentive afin de dégager les points entraînant une latence excessive avec l'objectif de réduire cette latence en proposant d'autres solutions architecturales.

Cette étape permet de convertir un ensemble de pixels résiduels (ces pixels sont dits résiduels car il s'agit de la différence entre le pixel réel et le pixel prédit, quel qu'elle soit, intra ou inter), le plus couramment regroupés en blocs carrés ou rectangulaires depuis le domaine spatial au domaine fréquentiel. Cette transformée doit avoir accès à l'ensemble des pixels d'un bloc 4x4 soit 16 pixels au total. Une implémentation simple de la transformée entière (directe) peut être vu de la façon présentée par la figure 4.10

Il s'agit d'une implémentation de type *butterfly* qu'on retrouve dans beaucoup d'implémentation et qui est due à [MHKK03]. Nous noterons cette architecture U, qui constituera notre opérateur de base pour la transformée entière. La représentation de cette implémentation est ici incomplète car il n'est représenté uniquement l'opérateur qui permettra de calculer la transformée entière. Comme on peut le constater, il n'y a que 4 entrées sur cette opérateurs, et ceci simplement parce que le calcul s'effectue sur

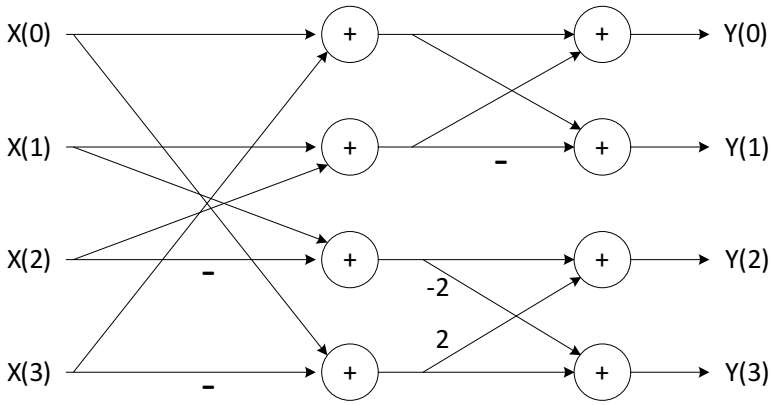


FIGURE 4.10 – Implémentation *butterfly* pour la transformée directe

des colonnes ou des lignes. Le calcul doit également se produire en 2 passes : une passe pour les colonnes et une seconde pour les lignes. L'opération simple présentée ici doit donc être dupliquée pour un total de 8 fois (4 colonnes et 4 lignes dans un bloc 4x4).

4.6.2.1 Implémentation séquentielle de la transformée

Lorsque les transmissions de données s'effectuent au sein de l'architecture sur des bus ne convoyant qu'un pixel à la fois (par coup d'horloge), il n'est pas utile de dupliquer l'opérateur de base.

Dans une implémentation classique de cette transformée [Kes10, Kor08] les pixels sont acheminés un par un dans la zone de calcul. La figure 4.11 montre l'architecture de transformation. Les pixels sont dans un premier temps accumulés, quatre à quatre dans un buffer d'entrée à décalage. Une fois que la première colonne de pixels occupe le buffer, le calcul est lancé sur la première colonne qui sera stockée dans un buffer bloc et qui, une fois le calcul sur les colonnes terminé, contiendra le résultat intermédiaire de la transformation, c'est à dire le résultat de la première passe. Ensuite le contrôleur lancera le calcul sur les lignes.

Ce calcul de type récursif introduit une longue latence. Si on considère qu'un pixel est acquis dans le buffer par cycle (d'horloge), le traitement d'un bloc entier prendra 32 cycles qui correspondent au temps d'acquisition de l'ensemble des pixels du bloc (16 cycles) et au temps de sortie de chaque coefficient sur le bus de sortie (OUT). Le

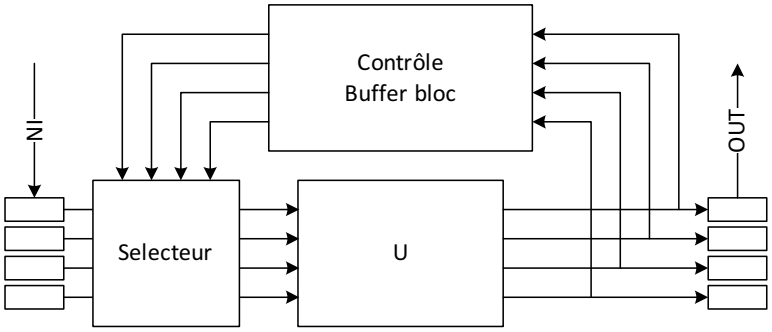


FIGURE 4.11 – Transformée directe ou inverse selon le choix de l'opérateur de base

calcul et le stockage du résultat intermédiaire sont réalisés en parallèle l'acquisition des pixels. Cette transformée est suivi par la quantification, qui comme nous le verrons à une latence qui peut être fortement réduite, mais elle est également suivi par son inverse afin de reconstruire ce bloc et l'utiliser dans la prédiction du bloc suivant. L'inverse de la transformée entière peut être réalisée en utilisant la même architecture. L'opérateur de base change alors pour prendre la forme de la figure 4.12.

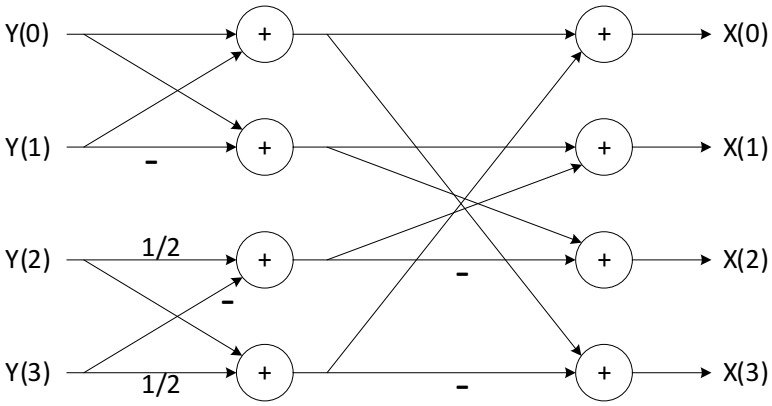


FIGURE 4.12 – Implémentation *butterfly* pour la transformée inverse

De même, la latence d'une telle architecture de transformée inverse sera équivalente à celle de la transformée directe, soit 32 cycles. Comme nous l'avons dit, l'ensemble du bloc devant être traité (prédit, transformé puis quantifié) puis reconstruit par les fonctions inverses avant de pouvoir être utilisé dans la prédiction du bloc suivant, ces temps de latence s'ajoutent et entre l'envoi d'un bloc dans la chaîne de traitement et l'envoi du bloc suivant s'écouleront 85 cycles. Certes, une telle implémentation peut

être envisagée dans le sens où dans de telles conditions, une vidéo en 720p et 25 images par seconde pourra être traitée en utilisant une fréquence de fonctionnement de 125 MHz mais des optimisations sont cependant envisageables.

4.6.2.2 Implémentation parallèle de la transformée

Il est alors intéressant, afin de limiter la latence en nombre de cycles de l'architecture de traitement de la transformée d'envisager une parallélisation. Il est en effet possible de réaliser une transformée en multipliant les opérateurs de base, ceci engendrant la modification de la taille des bus de transmission des données entre chaque fonction. De manière totalement opposée à l'utilisation d'un opérateur de base réalisant séquentiellement tous les calculs de la transformée, il est envisageable de dupliquer cet opérateur afin de réaliser une architecture purement combinatoire comme l'illustre la figure 4.13.

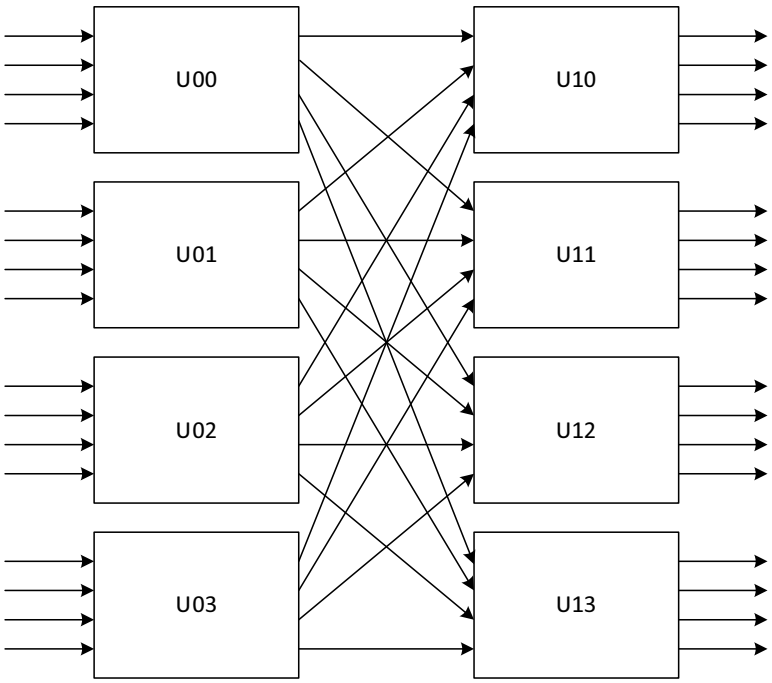


FIGURE 4.13 – Implémentation parallèle de la transformée

Les inversions entre les deux colonnes d'opérateurs permettent de transposer la matrice des coefficients et donc de traiter les lignes après les colonnes du bloc de pixels tout en réutilisant le même opérateur.

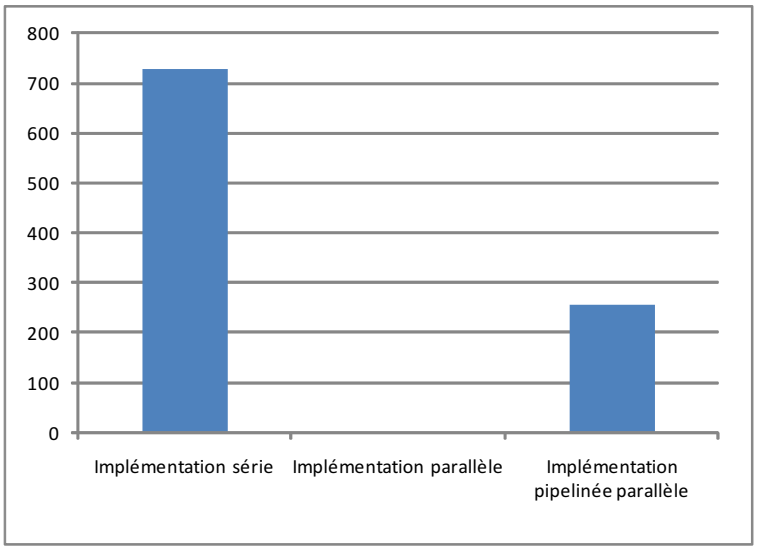
TABLE 4.3 – Comparaison des différentes implémentations proposées

Type d'architecture	*Latence (nombre de cycles)*	*Surface (Slice)*	*chemin critique*
Architecture séquentielle	32	583 (728 slices registers)	nop
Architecture parallèle	1	992	9,508 ns
Architecture parallèle pipelinée	2	992	4,6717 ns

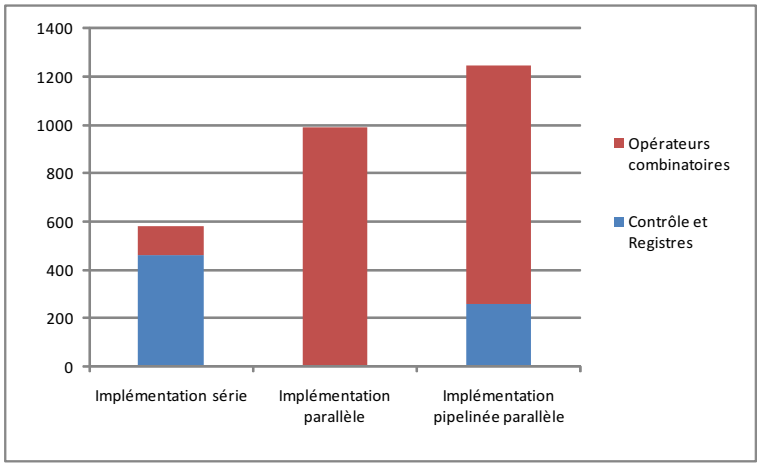
Le gain de cette structure totalement parallèle est assez important. En effet comme on peut l'observer dans la table 4.3, autant la latence est diminuée drastiquement, autant la surface n'est augmentée que de 36%. Le contrôleur de la version série de cette transformée utilisant une surface non négligeable au sein du FPGA, comme le montre le diagramme ci dessous. Il est également possible de recourir à un pipeline entre les deux colonnes d'opérateurs de l'architecture parallèle. Ainsi, le chemin critique est réduit et l'ensemble peut travailler à une fréquence plus élevée. Cette modification entraîne néanmoins une augmentation des ressources en termes de registres et introduit une latence cyclique non nulle. Les diagrammes 4.14 montrent la proportion de slice LUT (b) et de slice registre (a) utilisées par les différentes architectures.

4.6.2.3 Quantification

La quantification permet une compression plus efficace grâce au codeur entropique. Cette quantification permet une diminution de la dynamique des coefficients de sortie de la transformée. Dans les cas typiques, une majorité de coefficients (essentiellement ceux trouvés dans les hautes fréquences) sont réduits à zéro. En termes d'architecture, la quantification est relativement simple. En partant de la définition mathéma-



(a) Slice Register



(b) Slice LUT

FIGURE 4.14 – Occupation des différentes implémentations sur Virtex 5

tique donnée dans la section 4.3, et toujours avec notre objectif de parallélisation massive de l'architecture, nous pouvons en déduire une architecture contenant un macro-opérateur (défini par des opérateurs tels qu'une multiplication ou un décalage à droite) qui sera dupliqué pour chaque coefficient à calculer, 16 pour un bloc 4x4. Les zones de calcul QM varient légèrement d'un opérateur à l'autre dés lors que ce calcul dépend de la position du coefficient dans le bloc. Chaque sous-bus transportant un coefficient étant dédié, le type de calcul à l'intérieur de ces zones ne varie que lorsque le paramètre de quantification QP varie. La figure 4.15 montre la parallélisation de ces opérateurs.

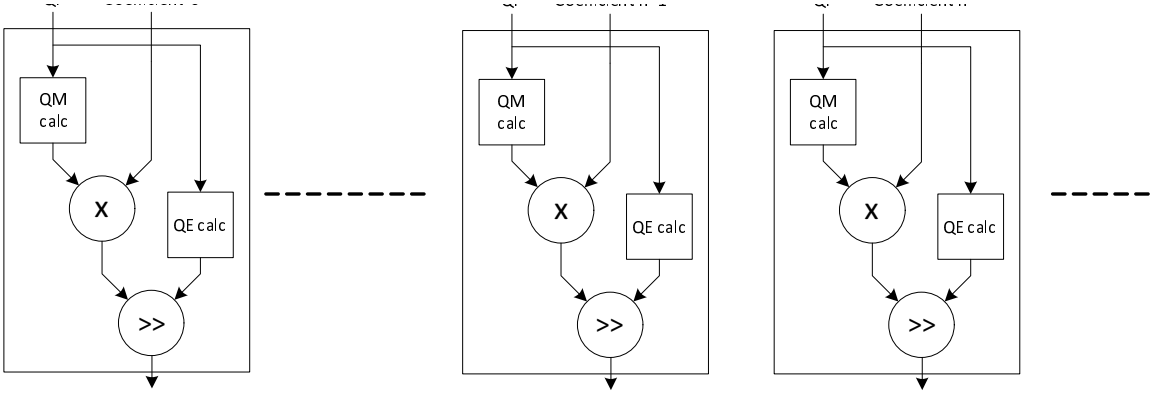


FIGURE 4.15 – Illustration de la parallélisation de l'unité de base de la quantification

4.6.3 Codage entropique

Le codage entropique dans le format H.264 est constitué essentiellement par le codage exp-Golomb et le codage en CAVLC (Contexte-based Adaptative Variable Length Coding). Ces deux éléments peuvent cohabiter au sein d'une même archicture afin de partager certaines phases du décodage comme l'acquisition et le stockage local des données. Le diagramme 4.16 présente cette architecture dans son ensemble.

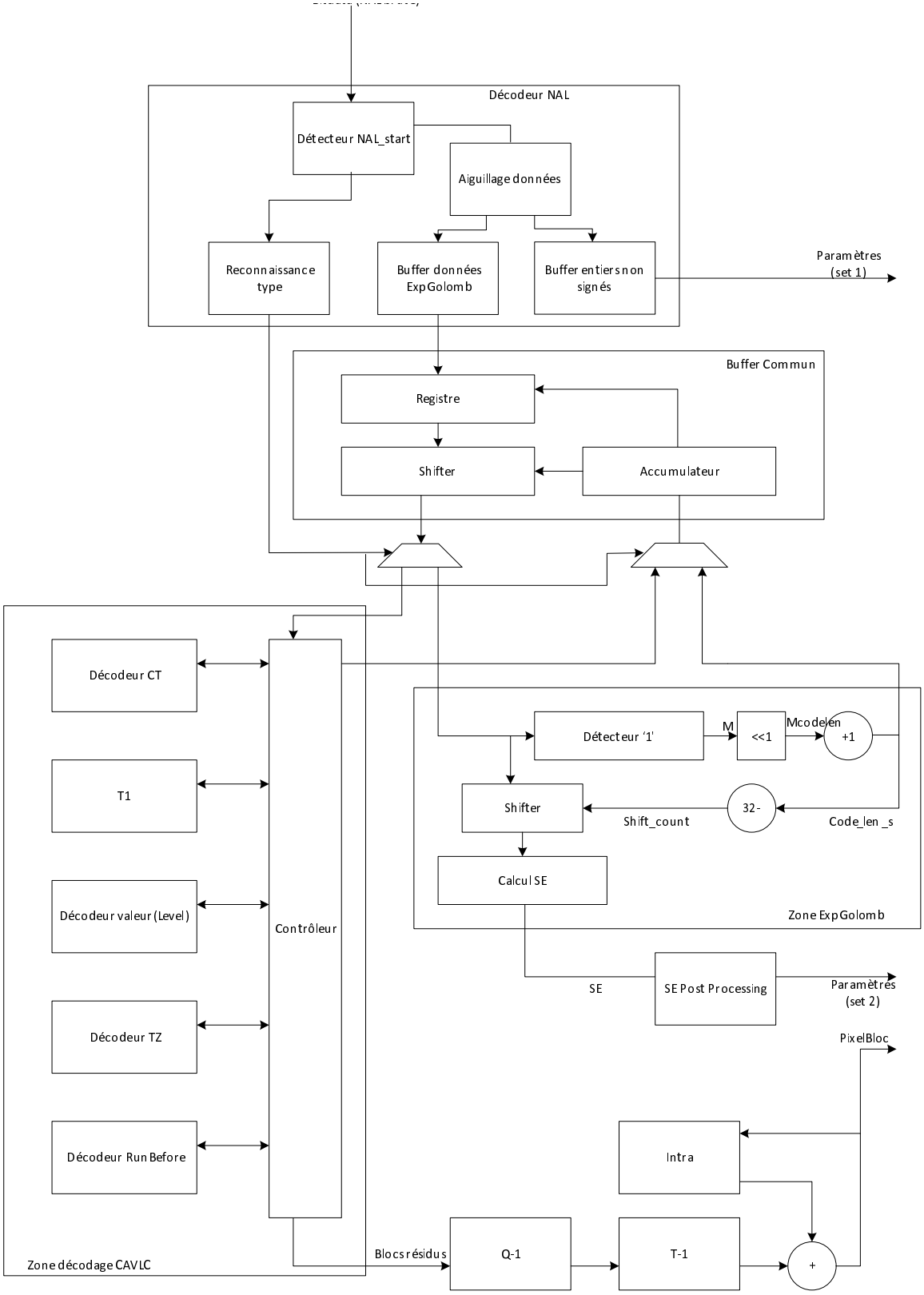


FIGURE 4.16 – Ensemble du processus de décodage entropique

Le décodeur entropique de notre architecture est composé d'un décodeur exp-Golomb et d'un décodeur CAVLC (Context-based Adaptative Variable Length Coding).

4.6.3.1 Codage exp-Golomb

Un grand nombre d'éléments constituant l'entête de NAL est codé grâce au système exp-Golomb (tableau 4.4. Ce codage est un codage simple [IT] qui est défini de manière systématique comme dans l'équation 4.10. L'information est contenue dans le code qui suit directement un nombre M de '0' puis un '1'. Le nombre de zéros indique la longueur du code de la donnée utile (INFO) qui se trouve directement après le '1' du code.

TABLE 4.4 – Méthode de codage pour différents éléments syntaxiques

Eléments syntaxique	*Description*	*Type de codage*
SPS/PPS	Entêtes et paramètres	Exp-Golomb
Type de MB	Type de prédiction pour chaque MB	Exp-Golomb
Paramètre de quantification	Indique la différence entre le nouveau QP et le précédent	Exp-Golomb
Indice d'image de référence	Identifie l'image de référence pour une prédiction inter	Exp-Golomb
Vecteurs de mouvement	Indique une différence à partir d'une prédiction de mouvement	Exp-Golomb
Données résiduelles	Coefficient des blocs 4x4 luma ou 2x2 chroma	CAVLC

TABLE 4.5 – Quelques exemples de codage exp-Golomb

Valeur	mot-code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

$$[Mzeros][1][INFO] \tag{4.10}$$

La valeur de la donnée utile est trouvée en utilisant à la fois le nombre de zéros M et le suffixe INFO. Ainsi, la valeur du code est déterminée par l'équation 4.11. Quelques exemple de valeur et de leur codage respectif est donné dans le tableau 4.6.3.1.

$$valeur_code = 2^M + INFO - 1 \tag{4.11}$$

Les éléments syntaxiques ne sont pourtant pas extrait directement de 4.11, mais sont interpretés selon leur type. En effet, dans la transmission des éléments syntaxique, on trouve quatre types différents :

- les codes exp-Golomb non-signés (ue)
- les codes exp-Golomb signés (se)
- les codes exp-Golomb mappés (me)
- les codes exp-Golomb tronqués (te)

L'architecture proposée (figure 4.17) est dérivée de l'architecture proposée en [DWMZ03].

Comme on peut le voir sur la figure 4.16, le buffer d'entrée décrit ici sera commun avec le buffer du décodeur CAVLC.

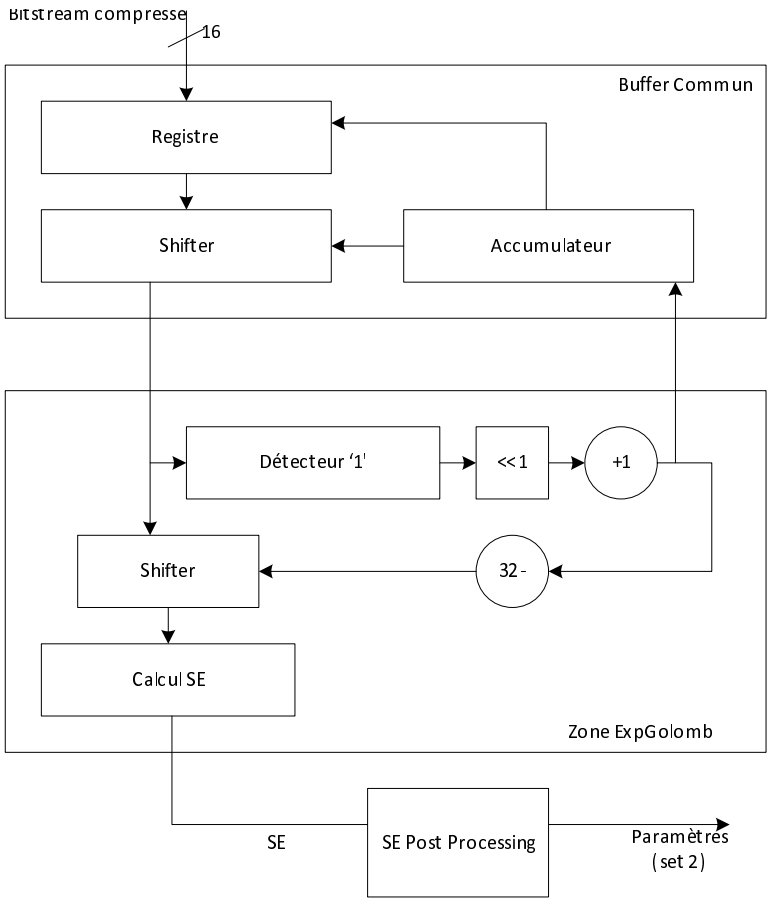


FIGURE 4.17 – Processus de décodage des éléments codés en exp-Golomb

Cette architecture propose une approche classique du décodage exp-Golomb. Une fois le buffer chargé, une décalage permet au module détecteur du premier '1' de définir la valeur de M. Ensuite, à partir de cette valeur, nous pouvons déduire la taille du code en multipliant par deux M (ou par un décalage binaire à gauche pour notre architecture matérielle) et en ajoutant 1. On peut ainsi faire remonter l'information au buffer d'entrée afin que le système opère un décalage de la taille de ce code sur le bitstream. Une fois le décalage réalisé, on a accès à la valeur binaire du code qu'il suffit d'interpréter en post-processing pour extraire l'élément syntaxique. Cette interprétation est fonction du type de NAL en cours de traitement, de la position du code et des valeurs des codes précédents [IT].

4.6.3.2 Codage CAVLC

La grande majorité des implémentations de CAVLC encode ou décode les coefficients de façon itérative. Pour l'encodage ou le décodage, le temps de traitement de CAVLC est alors dépendant des données à encoder, du nombre de coefficient non-nul, essentiellement. L'originalité de notre architecture est basée sur l'utilisation massive de parallélisme permettant d'une part un calcul rapide et d'autre part un temps de traitement figé. L'architecture de calcul (à différencier de l'architecture de contrôle qui permet la distribution des coefficients) est purement combinatoire. Nous avons tester deux types d'alimentations de données. Dans un premier temps une alimentation séquentielle fut utilisée, c'est à dire qu'un seul coefficient était envoyé dans l'architecture de traitement par coup d'horloge. Puis dans notre effort de réduction de latence global, nous avons pu procéder à une alimentation parallèle de l'ensemble des 16 coefficients d'un bloc 4x4.

La figure 4.18 décrit l'architecture globale utilisée dans le calcul des éléments syntaxiques (ES) du processus. On y distingue trois grandes parties :

- L'acquisition des coefficients et le prétraitement extrayant les variables nécessaires au calcul des ES
- Le calcul des ES en parallèle
- La construction du bitstream et la préparation au stockage ou envoie

La parallélisation est également exploitée à l'intérieur des modules de calcul des ES qui le nécessitent. Par exemple, nous avons parallélisé le calcul des *level* de chaque coefficient non-nul malgré leur forte dépendance. Cette parallélisation est visible sur la figure 4.19.

La totalité des coefficients non-nul est alors encodée en un cycle d'horloge. Chaque module permet d'obtenir deux informations : la taille du mot-code concernant l'ES et sa valeur. Certains coefficients du bloc 4x4 sont nuls (le nombre dépend essentiellement du paramètre de quantification : plus celui-ci est élevé, plus on trouvera de coefficient nul à l'intérieur d'un bloc) ; de ce fait, alors que dans une implémentation séquentielle, le calcul s'arrêterait au coefficient non-nul, dans notre architecture parallèle, le calcul

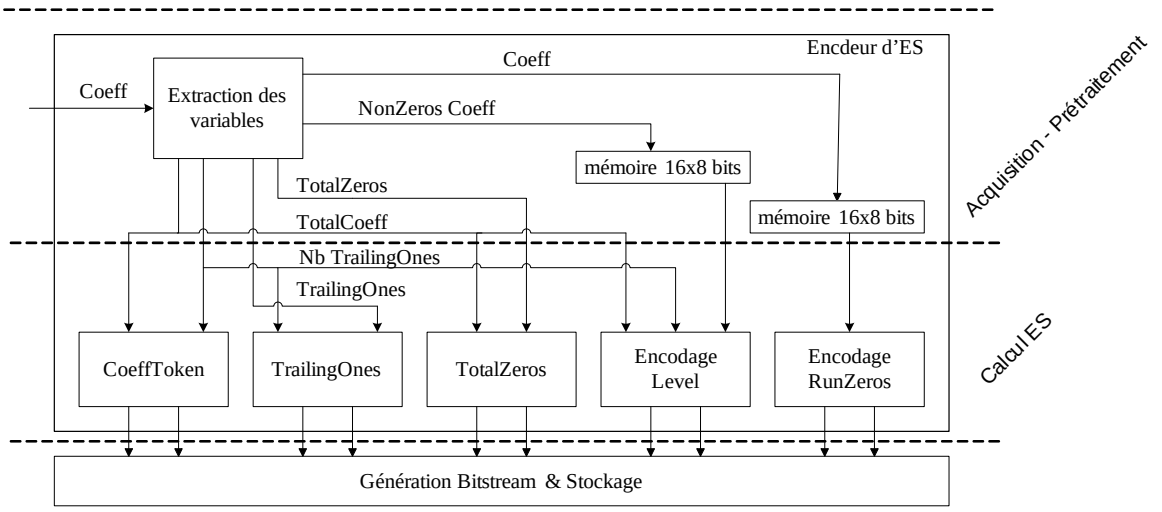


FIGURE 4.18 – Vue global de l'architecture du CAVLC

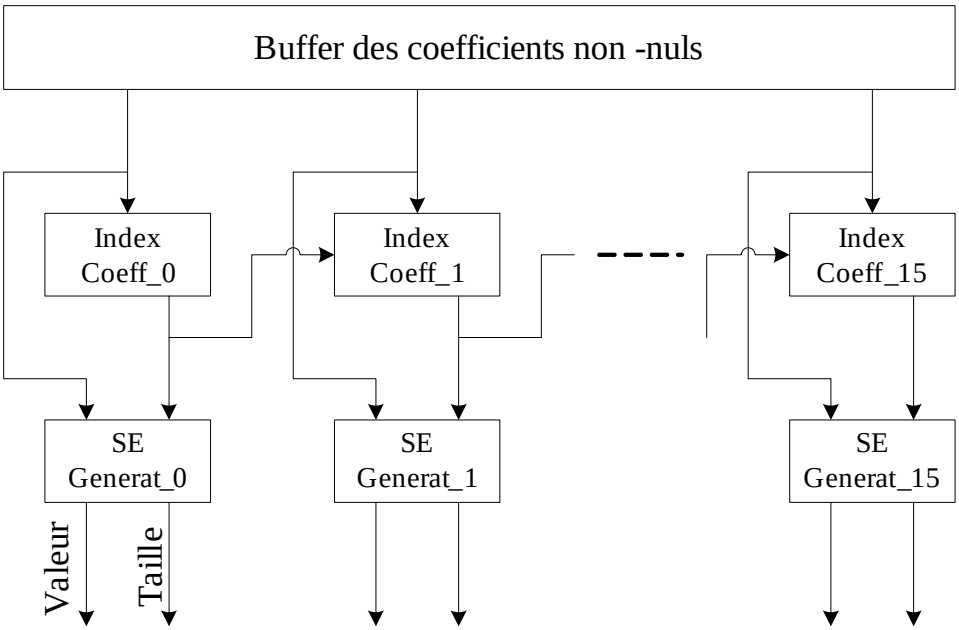


FIGURE 4.19 – Parallélisation du calcul de chaque ES dit *level*

est aussi effectué sur les coefficients nuls, mais le résultat concernant la taille est nul. Ainsi, dans la suite du processus, le génération du bitstream ne prendra pas en compte de coefficients nuls puisque la taille de leur ES sera nulle.

4.6.4 Transfert des données et hiérarchie mémoire

L'acquisition des données est réalisée grâce à un bus ; dans notre cas, il s'agit d'un bus PLB (Processor Local Bus) de Xilinx puisque nous avons travaillé exclusivement avec du matériel Xilinx. Ce choix a permis de mettre en place facilement une architecture d'acheminement des données. La figure 4.20 illustre la méthode d'acquisition ainsi que la hiérarchie mémoire des données compressées. Dans notre cas, les données ont été placées en mémoire externe afin de faciliter la réalisation expérimentale.

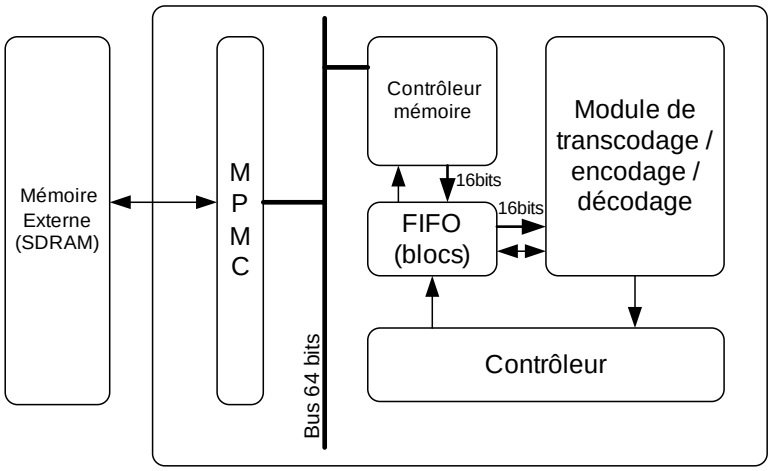


FIGURE 4.20 – Hiérarchie mémoire

Les données compressées sont acheminées vers le module de transcodage sur un bus de 16 bits. Dans le module de transcodage on trouve par contre des chemins données de 256 bits afin de réaliser les opérations du décodage ou de l'encodage de façon parallèle comme présenté dans les sections précédentes.

4.6.5 Communication entre fonctions

Dans l'architecture qui nous a servi de référence, du fait du traitement pipeliné du flux vidéo, il existe des buffers entre chaque étape du pipeline. Ces buffers sont représentés par des rectangles oranges dans la figure 4.2. Ces buffers n'apparaissent qu'entre le décodage et le recodage entropique et sont donc basés sur la même architecture. La figure 4.21 détaille l'organisation de ces buffers. La figure indique un buffer conçu pour

le passage de blocs pixels 4x4, certains traitements exigent de pouvoir procéder à des opérations sur des blocs plus large (pour l'Intra 16x16 ou le filtre de déblocage, par exemple), les modules responsables de ces traitements contiennent alors leur propre buffer de travail. Il existe également une variante de ces modules de transmission pour les blocs de coefficients qui eux nécéssite 16bits par coefficient (au lieu des 9 pour le domaine pixel).

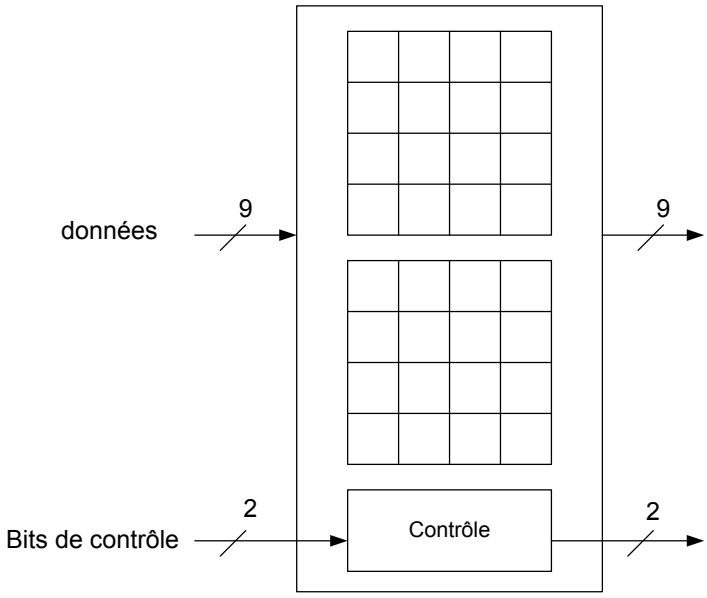


FIGURE 4.21 – Buffer du pipeline

Les bits de contrôle contiennent une synchronisation au niveau pixel (ou coefficient, selon la position du buffer dans l'architecture) et une synchronisation au niveau bloc. Le buffer est organisé en deux bancs mémoire qui travaillent en *ping-pong*, c'est à dire que lorsqu'un des bancs est en mode écriture, l'autre est en mode lecture. Une fois que le buffer en écriture est plein et le buffer en lecture vide (l'architecture est développée de façon à ce que les deux événements se produisent au même instant) la fonction des bancs s'inverse, celui qui était précédemment en lecture est utilisé en écriture et inversement pour l'autre.

Dans notre architecture à large parallélisation, les données ne sont plus acheminées de la même manière entre chaque module fonctionnel. En effet, il s'agit maintenant de

TABLE 4.6 – Ressources utilisées pour la communication inter-fonction

Type de ressource	*Communication série*	*Communication parallèle*
Bascules *Flip-Flop*	20	256
RAM 32x1	32	0
Eléments logics basic	25	0

pouvoir obtenir en un seul cycle d'horloge l'ensemble des coefficients ou des pixels d'un bloc 4x4. Ainsi, on simplifie largement le contrôle des données au détriment de la taille du bus qui augmente considérablement. L'interface entre les modules se résument alors un buffer composé de bascules, le tout contrôlé afin de répondre aux exigences des étapes de transcodage (voir figure 4.3). Le tableau 4.6.5 met en avant les ressources nécéssaires à la réalisation des deux types de communication que nous venons de présenter.

4.6.6 Scénarios étudiés

L'étude réalisée et décrite dans cette section est limitée au transcodage homogène, dans le sens où seul le codec de compression H.264/AVC ainsi que son extension scalable sont considérés. Dans un flux SVC, seul la couche de base est compatible avec les décodeurs purement AVC. Cette étude propose donc une méthode de transcodage de SVC vers AVC en reconstruisant un flux AVC à partir des couches SVC présentes et utiles à l'utilisateur final.

La scalabilité à trois dimensions du standard H.264 (fréquence temporelle, résolution spatiale, qualité échelonnable) implique de nombreux scénarios envisageables requérant une architecture de traitement (décodage ou transcodage) très flexible. Le but de ces travaux est de réaliser une telle architecture en exploitant la reconfiguration matérielle dynamique et partielle. Afin de démontrer la fonctionnalité et l'efficacité de la solution proposée, un flux SVC comprenant une couche de base et une couche de réhaussement de type résolution spatiale. L'architecture est adaptative et on dis-

tingue trois cas selon les besoins de l'utilisateur et le terminal qui servira à visualiser le contenu vidéo :

1. scénario 0 : La couche de base est utilisé seule en éliminant l'autre couche afin d'obtenir simplement un flux basse résolution, avec une qualité variable grâce à un transcodage de type transrating par requantification ou sélection fréquentielle.
2. scénario 1 : Augmentation de la résolution spatiale par prédiction de la résolution supérieure. Il s'agit de la prédiction intercouche prévue par SVC. Le flux AVC de sortie aura alors une résolution supérieure, mais le flux SVC à l'origine sera le même que dans le scénario précédent.
3. scénario 2 : Prise en compte du flux de réhaussement spatial du flux SVC afin d'apporter les informations manquantes à la prédiction inter-couche précédente. Le débit binaire du flux SVC est plus élevé, mais la qualité visuelle en est forcément améliorée.

En plus de ces trois scénarios, une scalabilité à un niveau plus fin est introduite du fait que pour chaque scénario, il est possible d'effectuer un transcodage de la couche de base par transrating comme dans le scénario 0. Le principe de couche de SVC ne permet qu'une adaptation par échelle de débit binaire relativement large alors qu'avec ce type de scalabilité, une adaptation précise du flux vidéo à la bande passante disponible est permise.

4.7 Performances de l'architecture proposée

Les algorithmes décrits dans la section Concepts mathématiques sont utilisés pour construire et simuler les différents scénarios avec Matlab. La description matérielle de ces architectures a été réalisée sous Xilinx ISE. Des simulations comprenant les événements de reconfiguration ont également été réalisées afin d'observer le comportement du flux de sortie du transcodeur lors d'un événement de reconfiguration. Le temps de reconfiguration d'un emplacement reconfigurable a été calculé grâce à un module ma-

tériel permettant cette mesure. Le temps de reconfiguration des zones reconfigurables a ensuite été insérer dans les simulations et la sortie du flux décodé a pu être analysée en terme de qualité.

4.7.1 Surface

L'architecture de base que représente le décodage de la couche SVC de base a été implémentée dans une zone statique du FPGA. Les zones reconfigurables son développées afin de recevoir les modules de décodage ou de ré-encodage. Chacune de ces branches peuvent être vues comme des modules distincts reconfigurables. Selon les besoins de l'utilisateur, l'une ou l'autre des branches est implémentée et la couche de réhaussement du flux SVC est utilisée ou non. Cette architecture est très flexible et est capable de réaliser le transcodage d'un flux HD 1080p type SVC en un flux AVC simple couche. Ces performances sont obtenues à une fréquence de travaille faible de 64MHz grâce à une forte parallélisation de décodage des différentes couches.

Cet algorithme a été implémenté sur une plateforme matérielle. Les résultats de l'implémentation sur Virtex 5 de Xilinx est présentée dans la table 4.7. En terme de mémoire, la prédiction Intra nécessaire à l'évitement de la dégradation de l'image par *drift* est le plus consommateur. Le CAVLC reste cependant le module nécessitant le plus de cellule logique pour sont implémentation.

La table 4.8 présente les résultats d'implémentation en ce qui concerne l'utilisation des ressources pour le codeur entropique CAVLC. Les résultats de synthèse montrent que la fréquence maximale d'entrée des coefficients dans le module CAVLC est de 152MHz. Cependant la majorité des modules internes de CAVLC travaillent à une fréquence plus basse du fait qu'ils agissent au niveau bloc et non pas au niveau coefficient. La fréquence de fonctionnement de ces modules est donc divisée par 16.

TABLE 4.7 – Utilisation des ressources FPGA pour chaque fonction

Blocks	Utilisation des ressources matérielles			
	Registres	LUT	DSP 48Es	Buffers
Transformation	797	3359	0	1
Quantification	0	175	3	0
CAVLC	37	2555	0	4
Prédiction	4018	225	0	0

En terme de LUT, l'architecture statique occuperait 112% du circuit alors que l'architecture reconfigurable occupe respectivement 56 et 77% d'un Virtex 5 LX50T. Afin d'implémenter l'architecture statique, un circuit de surface plus importante devra être envisagé. La consommation dans un FPGA provenant essentiellement de la consommation statique qui est fortement dépendante de la taille de ce dernier, un plus gros circuit consommera plus d'énergie. En prenant partie de la reconfiguration dynamique, on peut donc réduire la consommation d'un tel système. Un interêt certain pour les systèmes mobiles.

4.7.2 Consommation

Des estimations ont également été réalisées concernant la consommation d'énergie de l'architecture proposée dans ces travaux. Cette consommation de l'architecture reconfigurable (AR) est comparée à une version statique de l'architecture (AS). Ces estimations sont menées sur différents circuits afin de mettre en valeur l'utilité de la reconfiguration partielle puisqu'elle permet de réaliser les mêmes fonctions sur un circuit de surface inférieure. L'architecture statique est implémentée sur un circuit Virtex-5 LX85T, alors que l'architecture reconfigurable peut être implémentée dans un LX50T. La table 4.11 montre des résultats somme toute prévisibles. L'architecture implémentée

TABLE 4.8 – Résultats d'implémentation FPGA pour CAVLC

	Registres	LUTs	BufGCTRL
preprocessing	32	28	1
memory480	5	75	45
CoeffToken	0	1	0
TotalZeros	0	52	0
MemoryEL	0	16	1
MemoryERZ	0	16	1
EncodeLevel	0	1906	0
EncodeRunZero	0	417	0

TABLE 4.9 – Comparaison d'architectures pour CAVLC

Paramètres	Etudes comparées		
	[CHT+06]	[RB07]	Proposed
Portes logiques	23,600	6,855	28,152 (351/27,801)
Horloge (MHz)	100	50	63/4

sur un circuit plus gros consomme plus d'énergie, à peine plus en terme de consommation dynamique, mais beaucoup plus en terme de consommation statique. La fréquence de chaque architecture est fixée à 64MHz qui permet d'assurer le transcodage de vidéo HD 1080p.

4.7.3 Flexibilité

Même si les techniques de reconfiguration offre une flexibilité accrue [SVK01], le temps de reconfiguration ne peut pas être négligé. Quand un changement apparait dans le flux SVC ou que l'utilisateur change délibérément de type de visualisation, l'architecture est reconfigurée partiellement. Pendant ce temps, le processus de déco-

TABLE 4.10 – Utilisation de ressources matérielles

Resources	*AS*	*AR Sc.1*	*AR Sc.2*	*Virtex 5 LX50T*
Registres	16984	10501	11335	28800
LUT	32295	16260	22349	28800
DSP48	18	9	12	48
Buffers	22	11	16	32

TABLE 4.11 – Comparaison de la consommation d'energie

Consommation (mW)	*AS* *V5 LX85T*	*AR Sc.1* *V5 LX50T*	*AR Sc.2* *V5 LX50T*
Dynamique	53,19	28,43	35,3
Statique	644,78	367,42	367,42
Totale	697,97	395,85	402,72

dage dans la zone considérée est bien entendu désactivé mais le décodage de la couche de base est toujours actif. Une perte de qualité de la vidéo se fera donc sentir au niveau visuel, sans pour autant interrompre complètement l'image.

Pour la simulation et pour étudier l'impact de la reconfiguration sur le flux vidéo transmis, un cas d'utilisation simple à partir de différents scénarios SVC a été choisi comme suit : un flux vidéo est encodé en deux couches, une couche de base à basse

qualité et une couche de réhaussement en qualité. Dans un premier temps, l'architecture est capable de décoder seulement la couche de base car le transcodeur au niveau de cette couche est toujours actif, alors que la couche de réhaussement n'est prise en compte qu'à la demande et le décodeur à ce niveau n'est donc pas implémenté à la mise sous tension. Dans le même temps, le flux vidéo est réencodé en un flux AVC avec le module M3. Le temps de configuration du module M2 a été mesuré. Les résultats sont données sur la figure 4.22 et la table 4.12. L'Analyse dans la table correspond au temps nécessaire à la détection de la couche de réhaussement contenue dans le flux SVC par l'analyseur (*parser*). A ce moment précis, le module M2 est vide. RT0 correspond au temps de reconfiguration du module M2. Pour assurer cette mesure, un module matériel indépendant scanne à la fois l'analyseur qui est capable de lui transmettre un signal à partir du moment où il détecte la couche de réhaussement et une sortie de validation de M2 qui indique que le module est fonctionnel.

La séquence de test *foreman* à 25 images par seconde est encodée en intra sur deux couches suivant le standard H.264. Le processus de configuration est lancé quand la couche de réhaussement est détectée. La figure 4.23 montre un exemple de mesure de PSNR avant et après la reconfiguration pour cette séquence. Le temps total avant l'apparition de la couche de réhaussement dans le flux de sortie est la somme du temps d'analyse et du temps de reconfiguration qui correspond à 614ms. La séquence vidéo commence donc visuellement par une basse qualité puis cette dernière est améliorée par la couche de réhaussement. La reconfiguration du module de décodage de la couche de réhaussement est réalisée pendant que le module dédié à la couche de base procède au décodage de celle-ci. Il n'y a donc pas d'interruption dans le flux malgré la reconfiguration d'une partie du circuit.

Un autre cas a pu être étudié. La figure 4.24 montre l'évolution du débit de sortie pendant l'adaptation du circuit à une bande passante variable qui chute en l'occurrence brutalement. La figure décrit une situation où la bande passante devient très faible. L'architecture éliminera dans un premier temps la couche de réhaussement, puis adaptera la qualité de la vidéo par requantification afin de réduire le débit binaire pour qu'il corresponde à la bande passante disponible en utilisant une méthode de réduc-

tion de débit binaire dans le module M3. Dans le cas où la bande passante ne serait que de quelques ko/s plus faible, l'architecture procèderait alors simplement à une réduction du débit binaire et non à une élimination de la couche de base. La figure 4.25 montre la qualité de la vidéo entre (a), un flux AVC issu d'une seule couche, et (b), un flux reconstitué grâce aux deux couches.

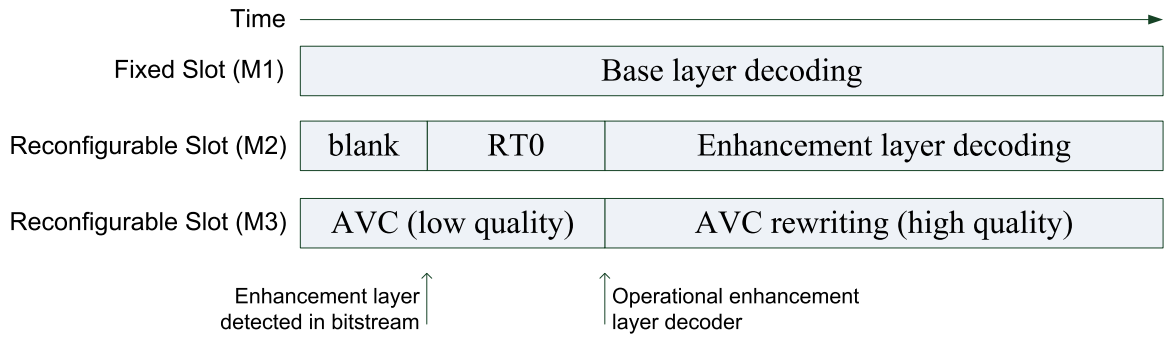


FIGURE 4.22 – Analyse du temps de reconfiguration

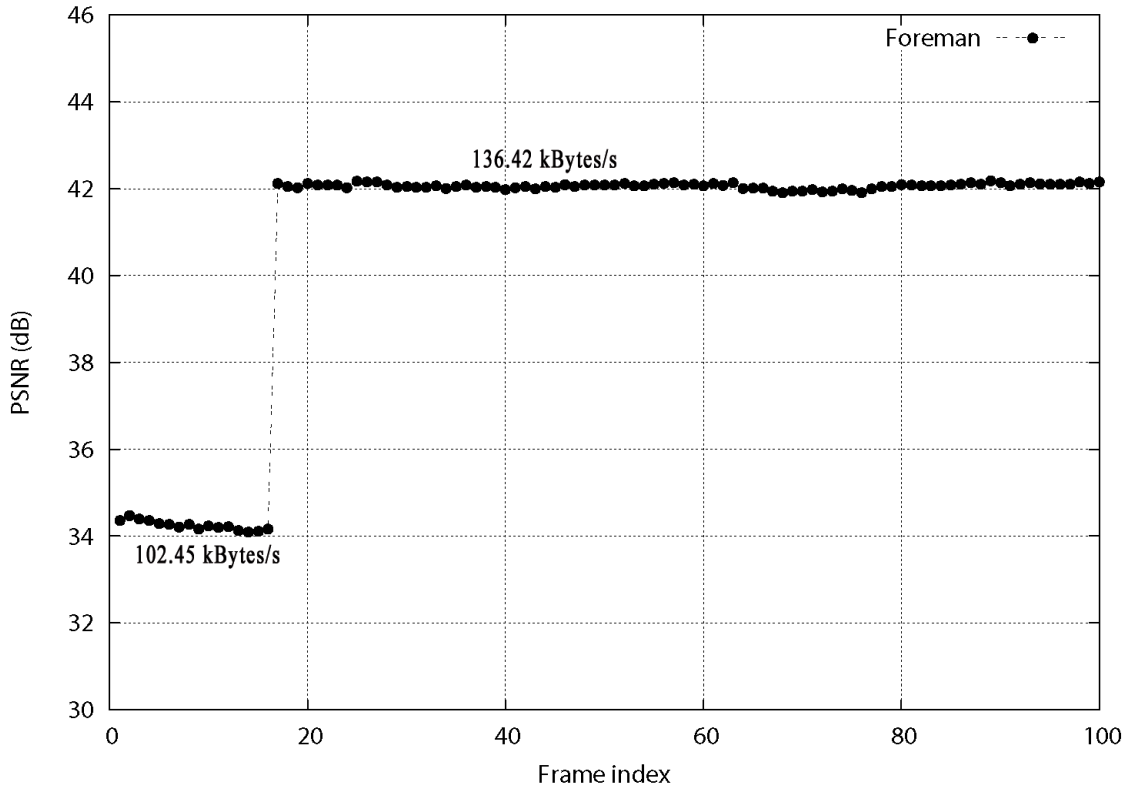


FIGURE 4.23 – Adaptation dynamique : mesures objectives PSNR

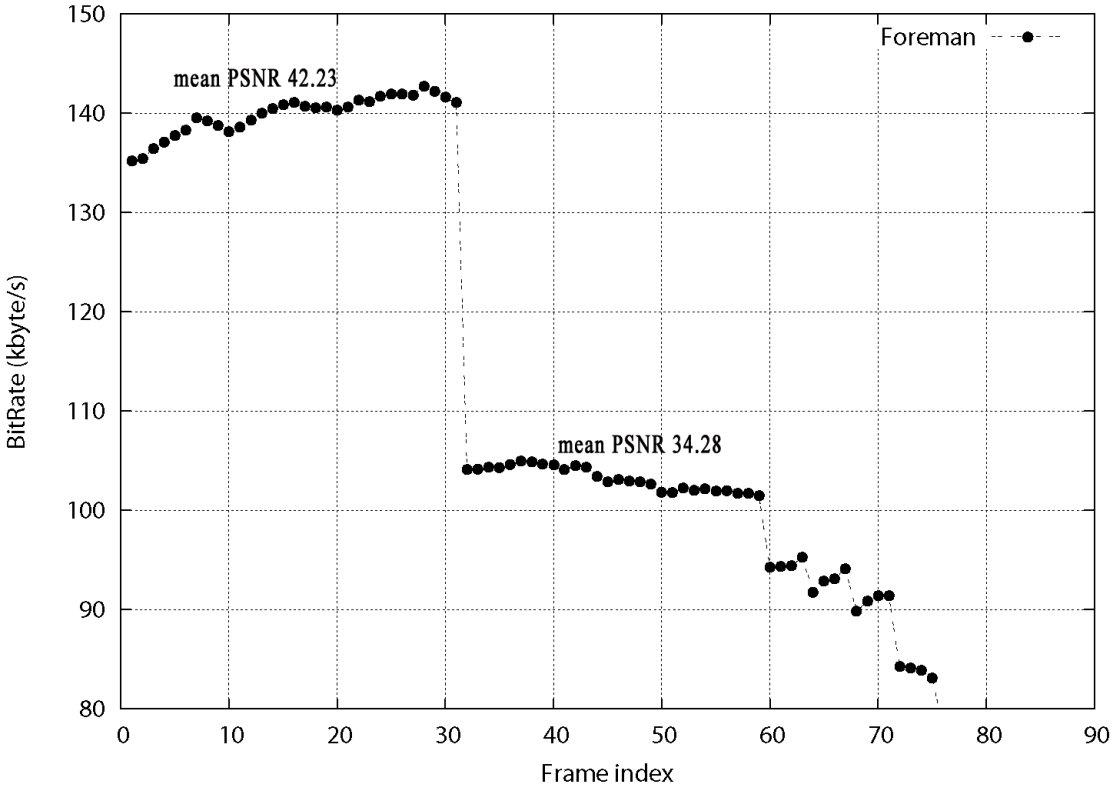


FIGURE 4.24 – Adaptation dynamique : fluctuations du débit binaire dans le temps

TABLE 4.12 – Temps de reconfiguration

Etape	*Temps (ms)*
Analyse du flux	182
RT0	432

4.7.4 Résultats en termes de transcodage

Le paramètre de qualité d'image PSNR (Peak Signal to Noise Ratio) pour l'évaluation des performances d'un transcodage est défini dans les équations 4.12 et 4.13

$$PSNR(db) = \frac{20 \times \log_{10}(Valeur\_max\_pixel)}{\sqrt{MSE}} \tag{4.12}$$

(a) scénario 1 (b) scénario 2

FIGURE 4.25 – Détail d'une séquence CIF : Foreman

MSE dans l'équation 4.13 représente la moyenne des erreurs au carré de l'image reconstruite définie comme

$$MSE = \frac{1}{N} \times \sum_{i} \sum_{j} [f(i,j) - F(i,j)]^2 \tag{4.13}$$

où N est le nombre total de pixels, $F(i,j)$ est la valeur du pixel au point (i,j) dans l'image reconstruite et $f(i,j)$ est la valeur du pixel dans l'image d'origine au point (i,j).

Comme les méthodes de requantification complète et de sélection fréquentielle donnent des résultats différents en termes de qualité d'image, différentes méthodes de transrating ont donc été simulées puis implémentées. Comme le montre la figure 4.26, dans le scénario 1, le PSNR est toujours meilleur pour la requantification que pour la sélection fréquentielle. Cependant, au niveau de la qualité visuel ressenti par l'utilisateur, il s'avère qu'il existe une différence. L'outil VQM (Visual Quality Metering) permet d'avoir une mesure objective de la qualité visuelle d'une séquence vidéo. Alors que le PSNR ne prend en compte les images que de manière indépendante, VQM permet également d'apprécier la qualité d'une vidéo sur la résolution temporelle. Par exemple, un papillotement entre les images, désagréable à l'oeil pourra être détecté par VQM, mais pas par la mesure du PSNR. Dans le cas présent, la requantification

introduit en effet un papillotement de l'image. Contrairement au PSNR, plus le résultat du VQM est proche de zéro, plus le résultats est bon. Le $DeltaQ_p$ correspond à la différence entre le paramètre d'encodage de la vidéo d'origine et celui de la requantification. Comme le montre la figure 4.27, lorsque la sélection fréquentielle dépasse un certain seuil, la requantification devient un meilleur moyen de transcodage en terme de qualité. Un simple abaque est implémenté dans l'architecture permettant de définir quelle technique sera plus efficace pour une réduction de débit donnée.

Dans cette première simulation, une séquence vidéo test en haute définition 1080p 30fps a été utilisée. Le PSNR est une moyenne obtenu sur toutes les images de la séquence. La figure 4.28 montre le taux de distorsion du flux transcodé. La simulation est réalisée sur les images Intra seulement et donne le pire cas en terme de débit binaire.

Les tests sont effectués avec un flux SVC dans lequel le paramètre de quantification de la couche de base est relativement bas ($Q_p = 10$). Un détail de la séquence est donné sur la figure 4.29 pour différencier les scénarios 1 et 2.

Dans le premier scénario, seulement la couche de base est utilisée et sa résolution spatiale est incrémentée par prédiction pour obtenir un flux haute résolution faible qualité. Le second scénario donne une résolution aussi élevé mais cette fois avec une qualité élevée puisque la couche de réhaussement est utilisée. Le débit binaire augmente donc inexorablement mais la qualité est également augmentée. Ce débit binaire peut toujours être ajusté plus finement grâce au transrating possible au niveau de la réécriture du flux AVC.

4.7.5 Implémentation

L'architecture proposée dans cette section est construite sur les bases de l'architecture permettant le transrating AVC. Cette précédente architecture de réduction du débit binaire permet la modification de la qualité de la vidéo. Dans ce type de transcodeur, la qualité et par conséquent le débit peuvent être ajustés finement à la bande passante disponible sur le réseau de transmission. Pour effectuer cette échelonabilité en qualité, une requantification complète des coefficients dans le domaine fréquen-

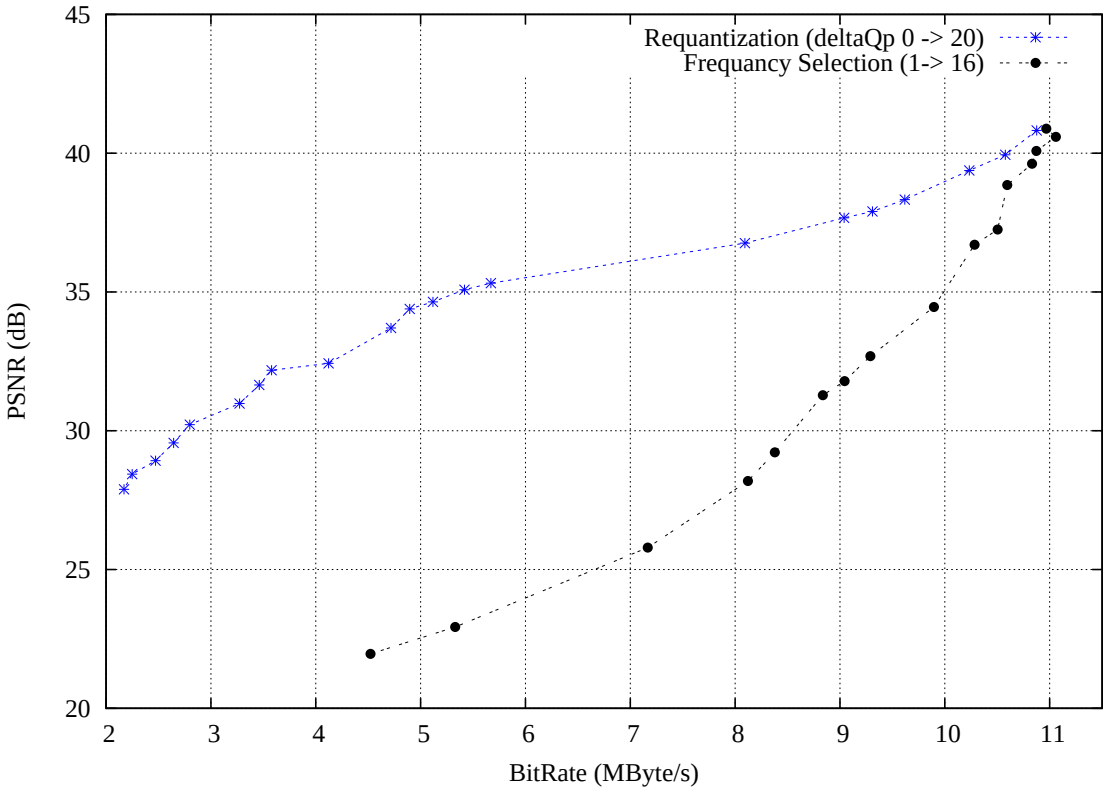


FIGURE 4.26 – Taux de distortion pour la requantification et la sélection fréquentielle

tielle est réalisée. Bien entendu, des techniques d'évitement de propagation d'erreur sont employées. On utilise également le principe de sélection fréquentielle pour réduire le débit binaire, cette opération est plus simple que la requantification en terme de complexité, une comparaison de ces deux techniques est réalisée plus loin. Ces procédés permettent d'introduire un nombre de valeurs nulles plus important dans les blocs de coefficients transmis et rendent ainsi le codage entropique plus efficace. Dans l'architecture proposée ici, la reconfiguration intervient afin de répondre à des besoins variés en termes de transcodage. Certains modules peuvent être implémentés ou non, selon les besoins ou simplement remplacés par d'autres.

Un système vidéo complet et adaptatif a été implémenté sur une plateforme de développement Xilinx ML505 équipée d'un FPGA Virtex-5 et d'une mémoire externe DDR-2 (entre autres composants non nécessaire à notre implémentation). Le flux vidéo compressé à transcoder est stocké dans cette mémoire externe pour les besoins

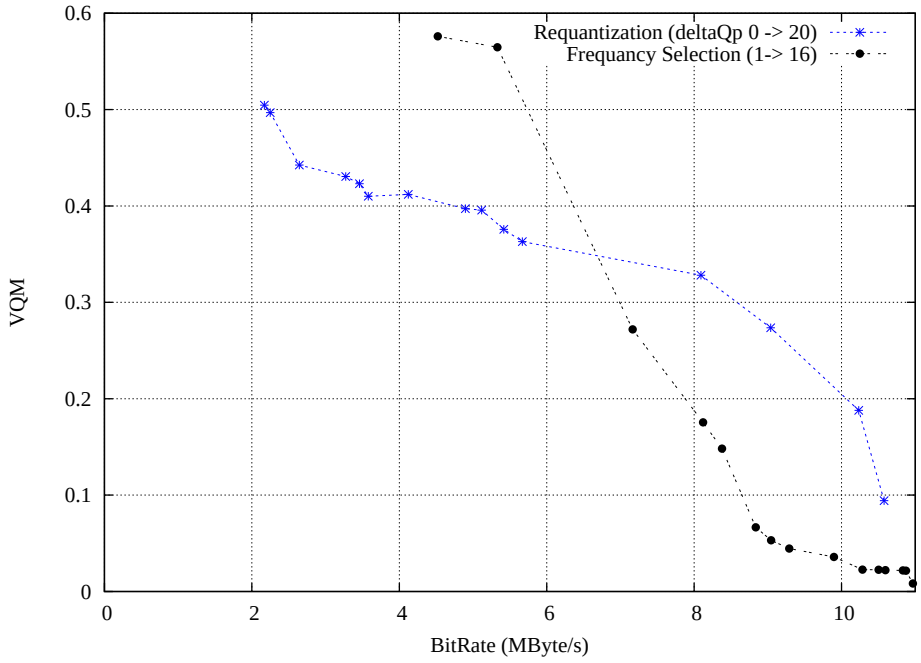


FIGURE 4.27 – Mesure du VQM pour la requantification et la sélection fréquentielle

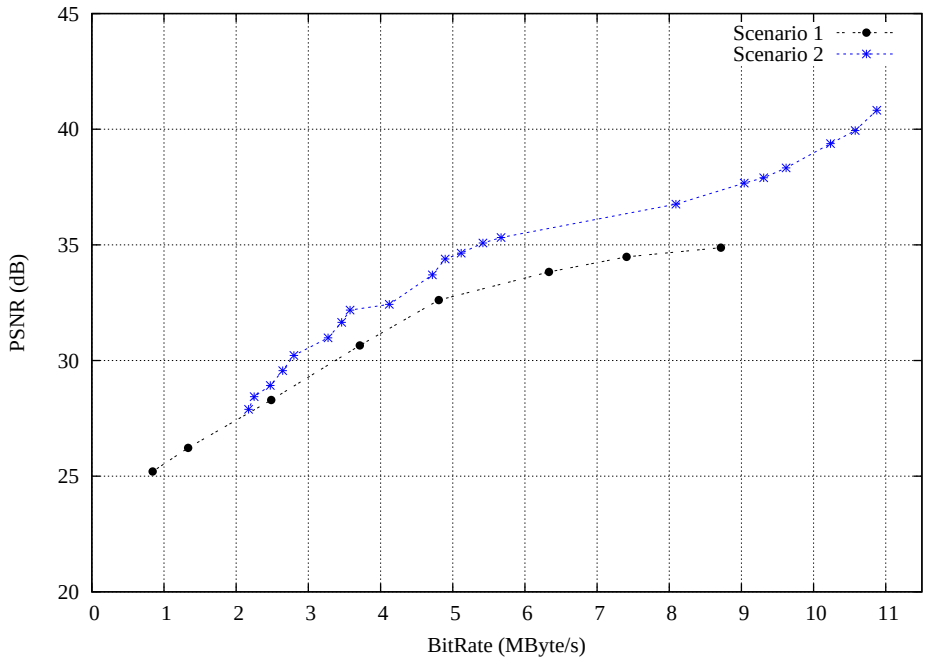


FIGURE 4.28 – Taux de distortion pour les Scenario 1 (a) et Scenario 2 (b)

de l'expérimentation. Ce flux est encodé en SVC avec une couche de base 720p et une couche de réhaussement 1080p. Dans ce banc de test, une autre carte comportant un

(a) scénario 1 (b) scénario 2

FIGURE 4.29 – Détail d'une séquence HD 1080p 30fps : Clouds

décodeur compatible AVC uniquement est utilisée et raccordée à un moniteur afin de visualiser le résultat du transcodage. Cette seconde carte communique avec la précédente en étant capable de lui envoyer des informations concernant un débit maximum acceptable sur le canal de transmission (en l'occurrence un lien ethernet). Cette information permet de simuler une variation du débit maximum sur le réseau et d'observer le comportement de la carte en amont réalisant le transcodage.

Le système de compression complet implémenté est présenté sur la figure 4.30.

4.8 Conclusion et discussion

Cette architecture est un premier pas en avant qui va nous permettre d'aller plus loin dans les mécanismes de reconfiguration pour le traitement vidéo. Dans la suite de ce mémoire, nous allons explorer une nouvelle voie dans le domaine, permettant de généraliser l'architecture à de nombreux autres types de transcodage vidéo.

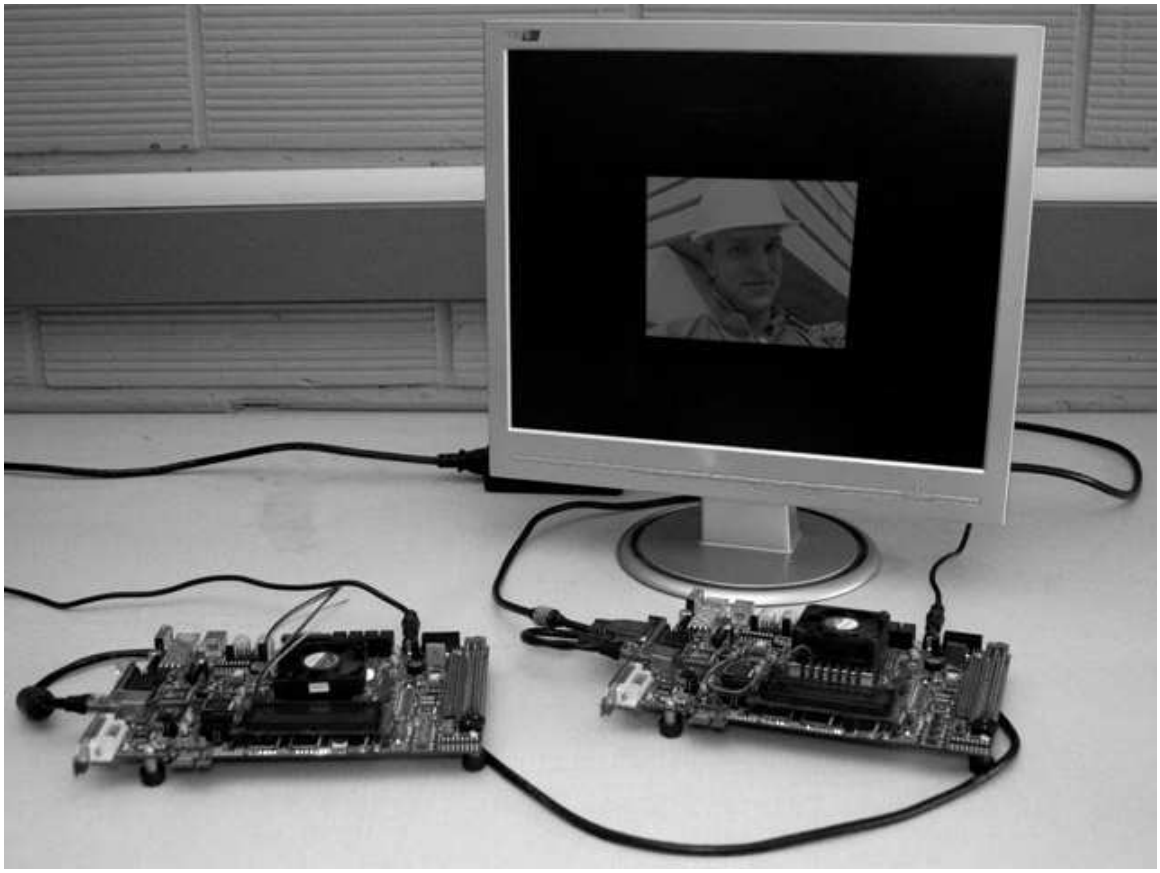


FIGURE 4.30 – Transcoding system.

Chapitre 5

Optimisation de l'architecture pour une meilleure adaptabilité

Sommaire

5.1 Introduction

Dans le chapitre précédent, nous avons détaillé une première architecture dédiée nous permettant un transcodage de SVC vers AVC. Dans ce travail de thèse, nous avons souhaité aller plus loin en cherchant à généraliser cette architecture à tout type de transcodage.

5.2 Extension à différents scénarios de transcodage

Dans de précédents travaux utilisant des versions software de ce type d'adaptation [LBH04] sont basée sur un langage de type XML afin de décrire la spécification du transcodeur à haut niveau. Nous avons réutilisé cette méthode en développant notre propre type de fichier que nous avons dénommé HASL (Hardware Adaptation and Specification Language). La figure 5.1 montre un exemple de ce à quoi notre fichier de spécification haut niveau peut ressembler.

Dans ce fichier de configuration, nous distinguons les macro-tags et les tags. Les macro-tags servent à définir la fonction globale de l'architecture. L'exemple donné en figure 5.2, montre un macro-tag qui, une fois interprété par le contrôleur de configuration, entrainera la génération de la fonction demandée à bas niveau. Ces macro-tags représentent des macro-fonctions monolithiques du type, encodeur, décodeur ou transcodeur. Ces informations contiennent également les éléments qui permettent de définir plus précisément la fonction prévue : le standard utilisé (mpeg-2, mpeg-4, H.264 ...), la qualité voulue, ses connexions avec les autres modules du système.

Le fichier contient aussi les informations réseaux qui vont permettre de configurer les modules de communication en entrée et en sortie. Comme on le distingue dans l'exemple de code donné en 5.1, les protocoles de transmission utilisés (TCP, RTP, UDP) en amont et souhaité en aval sont implicitement décrit dans la partie $< Network >$. Une fois que le contrôleur a détecté la fonction à réaliser ainsi que ses paramètres, il est capable de générer l'architecture à bas niveau à l'aide d'une librairie d'unités fonctionnelles (LUF). Cette librairie est détaillée dans la partie suivante.

```
<HASL>
<Network>
  <Input id="IN" src="udp://videoserver.lien.uhp-nancy.fr:4500"
     />
  <Output id="OUT" src="rtp://LIEN10.lien.uhp-nancy.fr:4501"/>
</Network>
<Process>
  <Transcoder id="T" ilink="IN" fmt_in="31" olink="OUT" frame-
     rate="25" qp="18" fmt_out="32"/>
</Process>
<Reconfiguration>
  <Probe id="P1" element="CONTEXT" parameter="bandwidth" unit="
     ko/s"/>
  <Probe id="P2" element="CONTEXT" parameter="packet-loss" unit
     ="%"/>
  <Probe id="P3" element="IN" parameter="incoming-stream-type"
     />
  <Probe id="P5" element="OUT" parameter="outgoing-stream-type"
      />
  <Probe id="P5" element="CONTEXT" parameter="terminal-
     resolution" unit="pix" />
</Reconfiguration>
</HASL>
```

FIGURE 5.1 – Exemple de fichier de spécification

```
<Transcoder id="T" ilink="R" olink="OUT" frame-rate="25"
   quality="80" fmt_in="31" fmt_out="32"/>
```

FIGURE 5.2 – Exemple de macro-tag au niveau process

Notre architecture est également conçue pour être reconfigurer dynamiquement (plus d'information à ce sujet sont données dans la partie de ce chapitre concernant l'implémentation de l'architecture matérielle). Certains paramètres peuvent en effet changer lors de la transmission. Ces changements concernent par exemple, une modification de la bande passante du canal ou une augmentation du taux de perte de paquets (essentiellement en transmission sans fil), ou encore un changement venant de l'utilisateur sur la résolution de la vidéo. La reconfiguration permet à l'architecture de s'adapter d'elle-même à ces changements d'environnement. Le fichier de spécification requiert également la description d'un groupe de sonde qui votn conditionner le mécanisme de reconfiguration. De telles sondes sont capables de communiquer avec le contrôleur de configuration qui peut quant à lui décider selon certaines conditions toujours décrite dans le fichier de spécification si une reconfiguration est nécéssaire ou non. Ces fonctions de reconfiguration évenementielle sont à distinguer de la configuration d'origine décrite dans le fichier HASL qui restent statiques lors du fonctionnement.

```
1 <Reconfiguration>
2   <Probe id="P1" element="OUT" parameter="packet-loss" op="
      superior" value="10" unit="%"/>
3   <Probe id="P2" element="OUT" parameter="bandwidth" op="
      inferior" value="1000" unit="ko/s"/>
4 </Reconfiguration>
```

FIGURE 5.3 – Exemple de description des sondes de mesure

Les sondes sont décrites dans le fichier de spécification haut niveau et conditionnent quels événements vont engendrer une reconfiguration dynamique (il est possible de définir des sondes qui vont mesurer la bande passante disponible au niveau du canal de transmission ou bien une sonde capable de détecter un changement de résolution de l'affichage chez le client). Chaque sonde retourne une mesure au processeur présent dans l'architecture qui procédera si besoin à une modification d'une ou plusieurs

fonctions de l'architecture ou changera simplement un des paramètres d'une fonction.

5.3 Définition des unités fonctionnelles

Cette librairie contient trois types d'unités fonctionnelles. La première catégorie regroupe les fonctions qui permettent la communication externe (encapsulation et désencapsulation). Les unités élémentaires de traitement de la vidéo contiennent toutes les fonctions élémentaire que l'on trouve dans les standards de compression vidéo de type mpeg. La troisième et dernière catégorie regroupe les unités de mesure ou sondes.

5.3.1 Unités de communication

Les unités de communication sont des modules d'encapsulation ou de désencapsulation qui permettent d'obtenir les données issues d'un type de protocole (UDP, TCP/IP ...) ou au contraire d'encapsuler les données sorties du transcodeur sous un certain protocole (qui peut se trouver être différent du protocole utilisé en entrée).

5.3.2 Unités de traitement vidéo

L'essentiel des constituants de la librairie concerne les unités de traitement vidéo. Ces modules matériels, écrits en langage haut niveau HDL, décrivent le comportement des fonctions basics que l'on trouve dans les standards mpeg. Certaines de ces fonctions ont été détaillées au chapitre précédent. Il s'agit des modules de prédiction de mouvement ou de prédiction Intra, de la quantification, de la transformée type DCT ou entière, de filtre de déblocage, etc ... Dans cette librairie, afin de rendre les fonctions compatibles entre elles, un effort de standardisation des entrées/sorties a été mené. Ainsi, lors d'un séquence de reconfiguration, une fonction peut être remplacé aisément par une autre sans modifié également les chemins entre les fonctions précédente et suivante.

5.3.3 Unités de mesures

Les sondes de mesure sont des modules spécifiques. Ils sont implémentés selon les besoins d'adaptation décrit dans le fichier de spécification haut niveau. Dans ce dernier, les sondes sont associées au signal qu'elles doivent mesurer (généralement une entrée ou une sortie du système). Ces unités mesurent tout changement dans le contexte d'utilisation du transcodeur. (augmentation ou baisse de la bande passante, modification du terminal utilisé afin de visualiser la vidéo, ...). Il peut y avoir trois types de sondes. (1) Une sonde peut mesurer un paramètre d'entrée à partir du flux (type de flux, débit binaire, ...), (2) ces mesures étant également faisable en sortie du système. Finalement, (3) une sonde peut également récuperer des informations sur le contexte d'utilisation (demande de l'utilisateur, capacité du canal de transmission). Chaque sonde retourne des informations au processeur responsable de gestion de la reconfiguration qui décide ou non d'une adaptation matérielle. Ce processus est décrit plus loin.

5.4 Définition des zones reconfigurables

Le modèle est illustré sur la figure 5.4. En haut de cette modélisation, le contrôleur de configuration et d'adaptation est utilisé pour lire un fichier de spécification défini par l'utilisation du transcodeur. Le contrôleur est alors capable de générer une architecture à bas niveau en utilisant une banque de fonctions que nous appelons la librairie d'unité fonctionnelles (LUF). Cette librairie contiens différents types d'actionneurs et de sondes dédiées à l'accomplissement de différentes mesures.

L'architecture à bas niveau est conçue pour convenir à tout type de transcodage respectant une approche cascadées traditionnelle dans le domaine pixel [AWSZ05] ou dans le domaine DCT. La figure 2.2 illustre cette approche de transcodage. Les connections entre les modules reconfigurables n'ont pas besoins d'être modifiées lorsqu'un des MR est reconfiguré parce qu'un MR est conçu pour recevoir un type de fonction précis (ou un groupe de fonctions comme dans le cas des fonctions de quantification

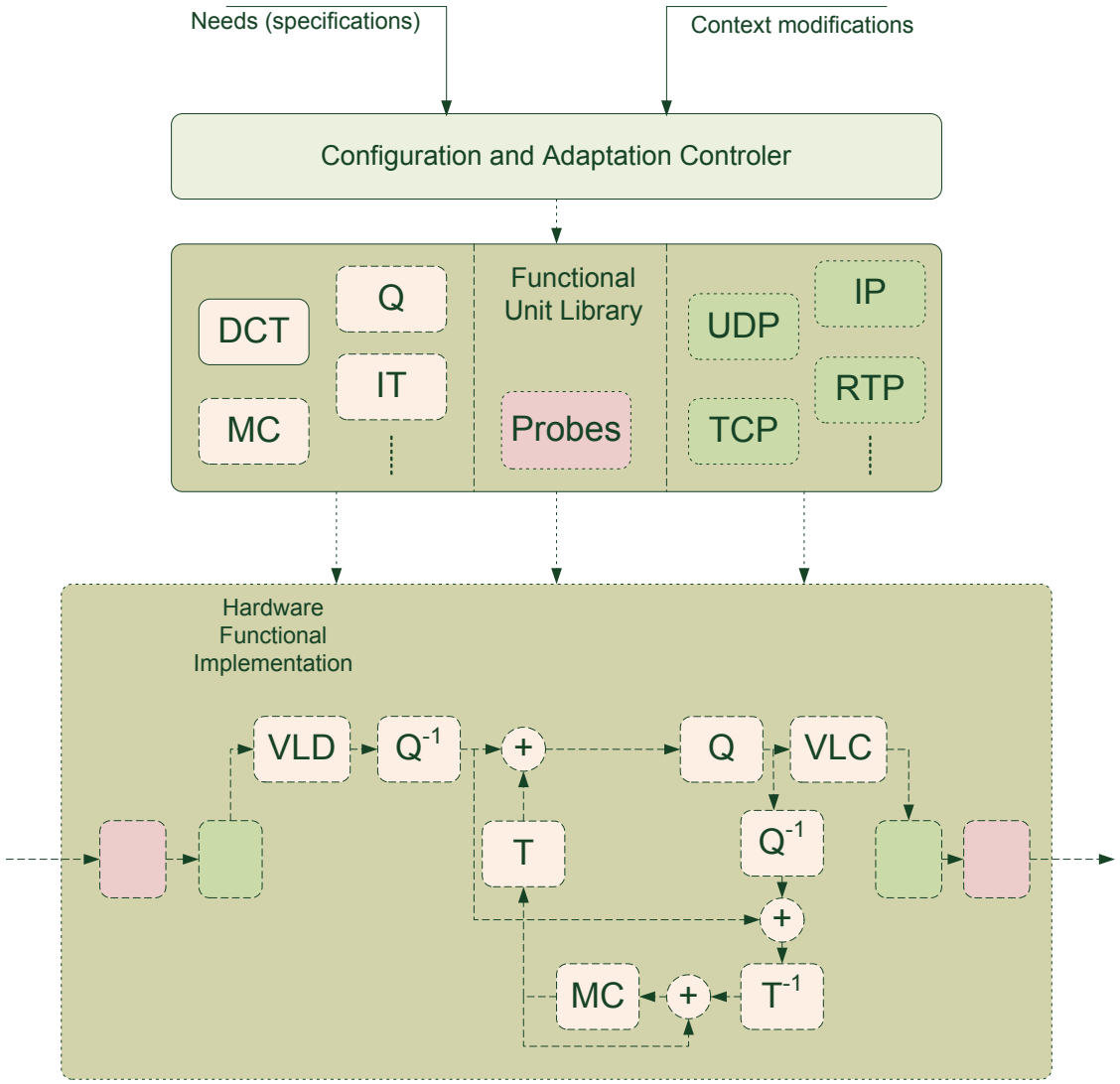


FIGURE 5.4 – Framework

et de transformée que nous avons regroupées afin d'occuper un seul module reconfigurable). Les données circulant d'un MR à un autre, même si la fonction du module est modifiée (mais toujours avec une fonction similaire), restent du même type (mêmes tailles de vecteurs, même signaux de contrôle). La taille des MR est alors prévue pour accueillir la fonction qui prend le plus de place sur le FPGA. Par exemple, le codage entropique est plus complexe en H.264 qu'en MPEG-2 et prend donc une quantité de ressources plus importante ; nous avons donc determiné la taille du module reconfigurable dédié au codage entropique afin qu'il soit assez large pour contenir cette fonc-

tion. Il va de soi qu'il saura également accueillir des fonctions de taille inférieure, mais qu'une partie des ressources du module sera inutilisée.

Notre architecture est basée sur un FPGA qui offre une grand souplesse d'utilisation de part ses possibilités de reconfiguration. En effet certains FPGA permettent une reconfiguration dynamique en cours de fonctionnement et cette reconfiguration peut être de plus seulement partielle, c'est à dire que les modifications peuvent n'avoir lieu que dans une zone prédéfinie du FPGA sans pour autant que le fonctionnement du reste du FPGA en soit perturbé. La figure 5.5 montre les différents composants de l'architecture que nous proposons. Ces différents modules peuvent être classés en deux catégories : les composants statiques et les composants reconfigurables (RPM ou MR : Module Reconfigurable). Les modules statiques contiennent essentiellement l'interface de reconfiguration ICAP (Internal Configuration Access Port) et un processeur, qui, dans notre cas, se trouve être un processeur MicroBlaze de Xilinx. Ce processeur est utilisé à des fins de gestion de la configuration et reconfiguration ainsi que de gestion du contexte et changement de contexte.

D'autres éléments statiques sont présents. La staticité de ces modules s'explique par le fait qu'ils seront utilisés dans n'importe quel contexte, aussi bien que dans n'importe quelle demande de l'utilisateur. C'est par exemple le cas de l'unité MAC (Media Access Control) utilisée pour communiquer entre notre FPGA et un lien Ethernet.

D'autre part, les zones reconfigurables (assimilables aux modules reconfigurables) ont été définies afin d'implémenter des unités fonctionnelles de traitement des données. Les connexions entre les MR sont statiques et chaque zones reconfigurables possèdent des entrées/sorties statiques. De ce fait, un effort d'homogénéisation des entrées/sorties à été fait entre les modules.

Certains RPM sont dédiés à un seul type de tâche. Par exemple, le RPM1 et 12 sont exclusivement destinés à recevoir des sondes afin d'effectuer des mesures sur différentes paramètres souhaités du réseau. Les RPM2 et 11 sont utilisés respectivement pour désencapsuler et réencapsuler le flux vidéo dans un protocole de transport. Les RPM3 et 10 servent à extraire les NAL (Network Abstraction Layer) ou au contraire à paquétiser les éléments compressés par le codeur entropique dans des NAL. Les RPM4

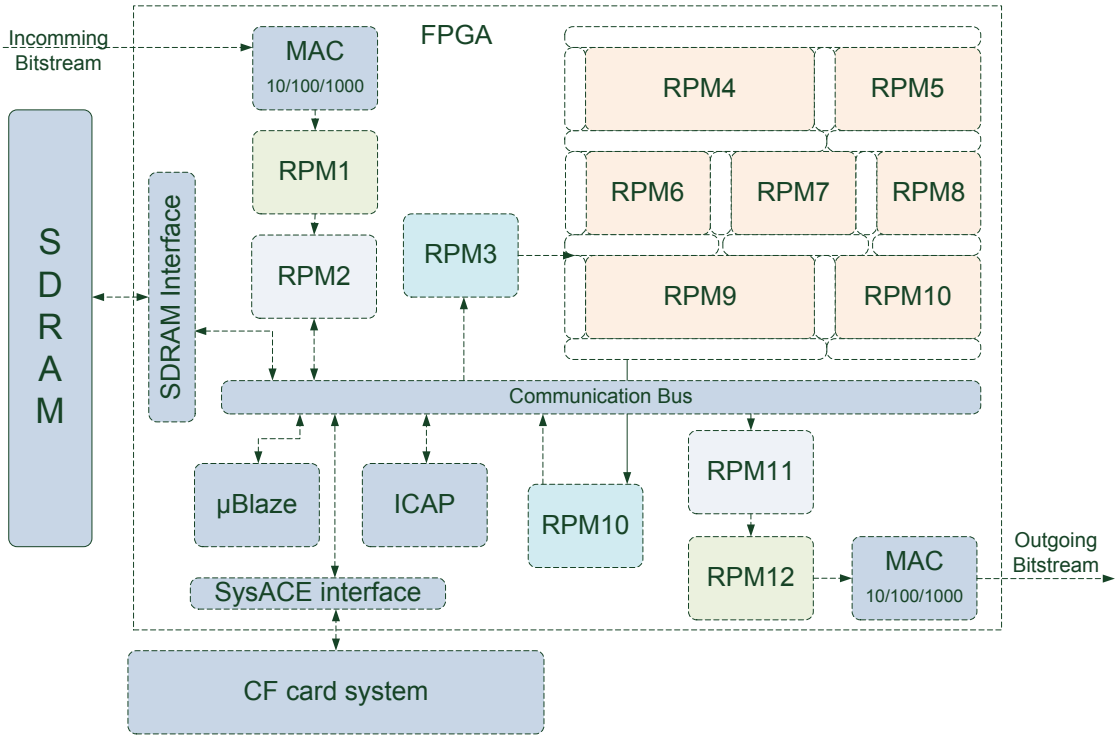


FIGURE 5.5 – Vue globale de l'architecture reconfigurable

à 9 sont dédiés à la réception de fonction de base du traitement vidéo. Dans l'architecture présentées ici, une mémoire externe (sous la forme d'une carte mémoire Compact Flash) et son interface de communication avec le FPGA sont également présents afin d'accéder aux différents bitstream partiels (les bitstream de configuration du FPGA, à ne pas confondre avec le bitstream du flux vidéo) stockés dans cette mémoire.

5.5 Gestion de l'adaptation en ligne

5.5.1 Processus matériel d'adaptation

Le Microblaze, un processeur implémenté dans le système sur FPGA, a deux tâches à réaliser. La première consiste en la lecture du fichier de spécification et donc en la configuration originale de l'architecture alors que la seconde est de gérer la reconfiguration éventuelle en récupérant les informations issues des sondes de mesure. Le fichier de spécicification haut niveau est stocké dans un premier temps manuellement

dans une mémoire externe à laquelle le processeur peut aisément accéder. Un programme écrit en langage C est chargé dans la RAM du processeur et consiste en une série d'instructions qui permettent d'interpréter le fichier de spécification. Une fois l'analyse des besoins en terme de fonctionnalité principale (encodage, transcodage ou décodage), de standard utilisé, et également en terme de d'adaptation nécéssaire au cours du fonctionnement, le processeur peut lancer la configuration de chaque RPM. Des fonctions dédiées en C (code de la figure 5.6) permettent un accés au bitstream partiel de configuration ainsi qu'une communication avec l'ICAP (Internal Configuration Access Port) du FPGA qui gère la configuration initiale ou la reconfiguration des zones reconfigurables. Dans le cas du système développé ici, les bitstreams partiels sont stockés dans une carte *Compact Flash*.

```
1 XHwIcap_CF2Icap(&HwIcap, "h264IT.bit");
```

FIGURE 5.6 – Fonction de reconfiguration

Après avoir lu le fichier de configuration initiale ou après avoir reçu une information venant d'une sonde engendrant une adaptation, le processeur analyse les nouveaux besoins et décide quelle zone de l'architecture doit être reconfiguré. Une fois que le module à modifier est identifié, le processeur lance un ordre de reconfiguration à travers l'ICAP pour instantier le bonne fonctionnalité à la bonne place.

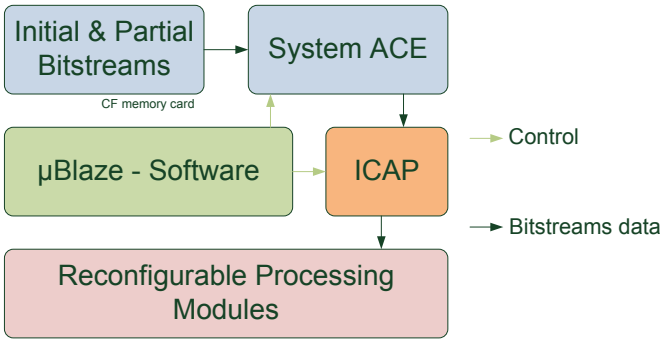


FIGURE 5.7 – Processus de configuration

5.6 Implémentation de l'architecture

L'architecture de base qui contient uniquement des zones reconfigurables sans y avoir associé une fonctionnalité comme décrit plu haut, est générée avec les outils Xilinx XPS. Les RPMs sont décrits dans des fichiers HDL indépendants. Dans l'outils XPS, lors de l'élaboration de l'architecture, seuls des composants sont instanciés, uniquement en déclarant leurs ports d'entrées/sorties, sans leur attribuer de comportement. Les fichiers HDL décrivant le comportement de chaque fonction sont générés à part en utilisant l'outil ISE. Les unités fonctionnels sont donc synthétisées afin d'obtenir un fichier *netlist*. Pour un RPM, on obtient deux ou plusieurs netlists différentes qui peuvent être instanciées dans le RPM quand une reconfiguration est requise et qui bien entendu possèdes des entrèes/sorties similaires. Afin d'obtenir les différents bitstream partiels, l'outil PlanAhead est utilisé. Dans cet outil, on place manuellement les zones reconfigurables des RPMs, puis on génère une configuration initiale et autant de bitstream partiel qu'il existe de fonctions différentes. Le résultat en terme de placement des composants est illustré sur la figure 5.8.

5.7 Résultats et comparaison

En terme de performance, une architecture matérielle surpasse ici une architecture software. Le tableau 5.1 donne une breve comparaison entre quelques points clés. Dans [LBH04] un Pentium 4 à 2Ghz est utilisé. Il apparait qu'à une charge de 70%, seul un flux d'une séquence vidéo de 273x205 pixels à un débit d'image de 25 par seconde peut être traité. L'architecture matérielle proposée ici permet quant à elle de traiter des flux Haute définition. La consommation d'énergie est également à prendre en compte. L'architecture matérielle est sur ce point bien plus performante.

En raison de la latence de reconfiguration, un blackout apparait quand un évenement de reconfiguration apparait. Une façon d'approximer théoriquement le temps de reconfiguration est d'appliquer la formule suivante : $\frac{taille\_du\_bitstream}{4} \times \frac{1}{frquence\_ICAP}$. Par exemple, un RPM contenant un bitstream de 195ko (un bitstream partiel qui corres-

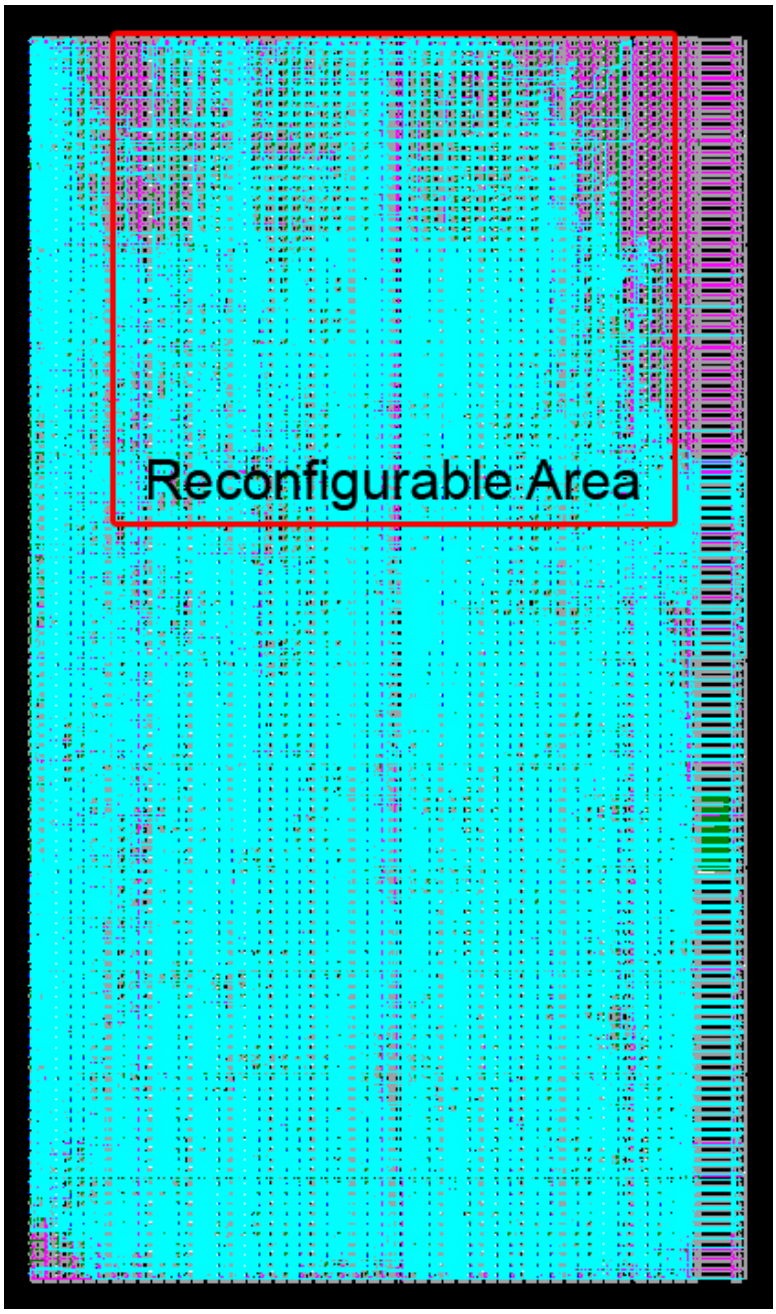


FIGURE 5.8 – Occupation du FPGA

pond à la fonction d'encapsulation UDP), le processus de reconfiguration prend approximativement 0.5 ms. Comparer au système développé en [LBH04], l'architecture proposée ici réduit drastiquement le blackout puisque ce dernier est estimé à 11.6 ms pour l'implémentation logicielle.

TABLE 5.1 – Comparaison entre solution logicielle et matérielle

Elements de comparaison	*[LBH04]*	*Architecture proposée*
Fréquence de travail	2GHz	100MHz
Résolution d'image	273x205	1920x1080
Image par seconde	25	30
Consommation d'énergie	71.8W	697mW
Blackout lors d'une reconfiguration	11.6ms	0.5ms

5.8 Conclusion

Pour clore ce chapitre et pour résumer les travaux effectués, une architecture embarquant une forme d'adaptation originale et efficace a été développée. En parallèle, l'architecture matérielle a été conçue pour convenir aux besoins et à la flexibilité nécessaire au traitement vidéo générique, permettant l'usage de nombreux standards que l'on peut trouver actuellement sur le réseau Internet. Ce chapitre présente donc une architecture flexible capable d'une adaptation automatisée et une réponse aux problématiques de transport réseau et aux flucuations des capacités de ce dernier. Les architectures figées ne peuvent se permettre de traiter un nombre important de scénarios. La reconfiguration au niveau fonctionnel, et non plus au niveau codec comme présenté au chapitre précédent, permet d'adresser une multitude de cas mais augmente la complexité de conception. Le but final d'une telle architecture est d'être livrée avec une bilbiothèque de composants exhaustive regroupant les fonctionnalités de tous les standards en cours d'utilisation apportant ainsi une universalité à l'architecture. Il reste cependant un poitn à explorer. Il s'agit en effet de garantir la pérénité de l'architecture en donnant la possibilité d'inclure de nouveaux composants dans le futur. La problé-

matique vient du fait de la staticité du nombre de ressource à l'intérieur des zones reconfigurables qui empêche toute implémentation d'une fonctionnalité nécéssitant plus de ressources. Des études en cours de réalisation proposent un découpage des fonctions afin de répartir une tâche sur plusieurs zones reconfigurables.

Conclusion et Perspectives

9 Bilan des travaux

Ce travail de thèse a été orienté vers une la constitution d'une architecture polyvalente pour le traitement vidéo et plus particulièrement pour le transcodage. Dans les travaux précédents ayant pour thématique le transcodage, aucune solution n'apportait un bon compromis entre performances et flexibilité. Dans la littérature ou dans les applications commerciales, des architectures efficaces peuvent être trouvées mettant l'accent sur l'une ou l'autre problématique. La conciliation des deux est rare. Or, on ne peut négliger les aspects performances lorsqu'on aborde le traitement vidéo temps réel et les aspects flexibilité du fait de la multiplicité des standards de compression, de la variété des transmissions envisageables et des moyens de lecture de contenu audiovisuel toujours plus hétérogènes. Avec les travaux présentés ici, nous apportons une base d'architecture matérielle réunissant des qualités de souplesse et de performances.

Dans cette optique, nous avons développé dans un premier temps une architecture non reconfigurable de transcodage offrant des capacités de traitement importantes. Cette première architecture a permis de valider les modules de calcul propre à la compression d'images (DCT, transformée, prédiction ...) qui sont détaillés tout au long de ce rapport. Cette première architecture a fait l'objet d'un effort d'optimisation afin d'obtenir de bonnes performances, chacun des éléments de traitement ayant été décrits puis testés séparément. Cette architecture a en effet été conçue dans l'esprit de la rendre ensuite plus flexible et le choix de l'implémentation sur circuit reconfigurable était justifié. C'est donc dans un second temps qu'une première étape de recherche de flexibilité a été conduite. Le transcodage choisi pour appuyer la réalisation de l'ar-

chitecture est une transposition du codage scalable SVC en un flux unique AVC. Ici encore, même si l'architecture gagnait en souplesse, elle n'était dédiée qu'à un type de transcodage. Nous avons donc enfin fait le choix de vouloir rendre cette architecture bien plus flexible et, en réutilisant les éléments réalisés précédemment, nous avons développé la dernière phase de cette architecture qui permet à présent de gérer de nombreux type de transcodage. Le résultat est en plus évolutif dans le sens où rien n'empêche de futurs utilisateurs d'intégrer de nouveaux blocs fonctionnels respectant les interfaces standardisées des zones reconfigurables.

10 Perspectives

La dernière architecture n'est pour l'instant fournie qu'avec peu de modules, notamment au niveau des modules de communication du type UDP, RTP ... Il serait intéressant d'enrichir la bibliothèque proposée et de continuer les tests de performances et fonctionnalités.

Annexe A

Liste des abréviations utilisées

FPGA : Field Programmable Gate Array

MR : Module Reconfigurable

MPEG : Moving Picture Experts Group

ICAP : Internal Configuration Access Port

MAC : Media Access Control

CABAC : Context-based Adaptive Binary Arithmetic Coding

CAVLC : Context-based Adaptive Variable Length Coding

XML : eXtensible Markup Language

VLC : Variable Length Coding

image-I : image dite Intra construite avec une prédiction ne dépendant que d'elle-même. Elle sert d'image de référence.

image-P : image prédite grâce à une estimation de vecteur de mouvement. Ce champ de vecteur occasionnant des erreurs, des résidus (différence entre l'image aux mouvements compensés grâce aux vecteurs et l'image originale) sont également encodés en parallèle afin d'obtenir une reconstruction parfaite.

image-B : image prédite de manière bidirectionelle. Des vecteurs de mouvement sont calculés à la fois depuis une image précédente et depuis une image future.

PSNR : *Peak Signal to Noise Ratio*, mesure la distortion entre une image compressée (puis décompressée) et l'image originale. Des images de bonne qualité obtiennent un PSNR entre 30 et 40 dB.

DSLAM : le *Digital subscriber line access multiplexer* est un multiplexeur qui permet de faire circuler à la fois les données de téléphonie classique et les données utilisant les technologies d'ADSL.

RTP : le *Real-time Transfer Protocol* permet l'insertion sur un protocole de transfert de données classique, tel que UDP, de marqueurs temporels permettant la lecture d'un flux vidéo en temps réel.

UDP : *User Datagram Protocol*

RTSP : Le (*Real Time Streaming Protocol*) est utilisé pour produire une intéraction entre le client et le serveur de contenu audiovisuel. L'utilisateur, à partir de la machine cliente peut alors lire le flux ou l'interrompre via une interface graphique.

Publications personnelles

M.GUARISCO, H. RABAH, Y. BERVILLER, S. WEBER, "An efficient VLSI implementation of H.264/AVC intra-frame transcoder", IEEE International Conference on Electronics, Circuits, and Systems (ICECS), December, 11-14, 2011.

[Soumis, en attente de review] M.GUARISCO, H. RABAH, Y. BERVILLER, S. WEBER, "FPGA implementation of embedded system for real time adaptation of H264/AVC-SVC video streams", Revue Elsevier, Signal Processing : Image Communication. 2010.

M.GUARISCO, H. RABAH, Y. BERVILLER, S. WEBER, "Dynamically reconfigurable architecture for real time adaptation of H264/AVC-SVC video streams", Computer Vision and Pattern Recognition, 2010. CVPR'10. IEEE, Conference on, 13-18 juin 2010.

M. GUARISCO, H. RABAH, Y. BERVILLER, S. WEBER, S. BELKOUCH, "FPGA-Based SoC For Transcoding H264/AVC-SVC With Low Latency And High Bitrate Entropy Coding", IEEE, System on Chip Conference 2009 (SOCC'09), 9-11 Sept 2009.

M. GUARISCO, H. RABAH, S. WEBER, A. SENOUSSAOUI, E. RENAN, "TOSCANE :TOwards SCalable Audiovisual Communication Networks", IEEE GIIS 09, Hammamet, Tunisia, 22-26 June, 2009, pp. 1-6.

L. CARMINATI, A.S. BACQUET, C. DEKNUDT, P. CORLAY, M. GHARBI, M. ZWINGELSTEIN - COLIN, M. GAZALET, F.X. COUDOUX, Y. BERVILLER, M. GUARISCO,

H. RABAH, S. WEBER, O. SALEM, A. SENOUSSAOUI, A. MEHAOUA, "TOwards SCalable Audiovisual broadcasting NEtworks : Overview of the TOSCANE Project", Int. Broadcasting Conf., IBC 2009, 10-14 Sept. 2009, Amsterdam.

C. DEKNUDT, P. CORLAY, A.S. BACQUET, F.X. COUDOUX, M. GUARISCO, H. RABAH, Y. BERVILLER, S. WEBER, "Transrating by frequencies selectivity for H.264/AVC intra pictures" Broadband Multimedia Systems and Broadcasting, 2009. BMSB '09. IEEE International Symposium on 13-15 May 2009 Page(s) :1 - 7 2009.

M. GUARISCO, H.RABAH, Y.BERVILLER, S.WEBER, "An Efficient Architecture Dedicated for Transcoding H.264/AVC Video Standard", SAME 2008, University Booth. 2008.

M. GUARISCO, H. RABAH, S. WEBER, "Architecture matérielle très haut débit pour l'encodage - décodage CAVLC pour H.264/AVC", GDR SoCSip, colloque 2008.

M. GUARISCO, X. ZHANG, H. RABAH, S. WEBER, "An Efficient Implementation of Scalable Architecture for Discrete Wavelet Transform On FPGA", System-on-Chip, 2007. DCAS 2007. 6th IEEE Dallas Circuits and Systems Workshop on Volume , Issue , 15-16 Nov. 2007 Page(s) :1 - 3 2007.

X.ZHANG, M.GUARISCO, H.RABAH, S.WEBER, "Architecture hybride reconfigurable pour l'application multimédia", C2I Nancy 17-19 octobre 2007.

Résumé

De nos jours, un nombre considérable d'appareil de haute technologie (PC, PDA, téléphone mobile...) permettent un accès à une multitude de fichiers multimédia par l'intermédiaire d'un réseau de distribution (Internet, réseau 3G ...). L'hétérogénéité des possibilités d'affichage et de transport implique diverses adaptations de ces contenus multimédias afin de les rendre diffusables. Le projet TOSCANE entend développer un système de diffusion qui s'appuie sur un codage dit conjoint de source et de canal afin d'améliorer la couverture de la diffusion des contenus vidéo. Deux stratégies seront mises en oeuvre : l'optimisation de la diffusion sur le réseau ADSL et chez l'abonné (diffusion WiFi, par exemple).

Nous avons réalisé, au cours de ces travaux de thèse un transcodeur par requantification ainsi qu'un transcodeur par troncature (ou transmission sélective). Ces deux méthodes ont été comparées et il apparait qu'en termes de qualité d'image l'une ou l'autre de ces méthodes est plus efficace selon le contexte. La suite de nos travaux consiste en l'étude du standard scalable dérivé de H.264 AVC, le standard SVC (pour scalable video coding). Nous avons souhaité étudier un transcodeur en qualité, mais aussi en résolution spatiale qui permettra de réécrire le flux SVC en un flux AVC décodable par les décodeurs du marché actuel. Cette transposition est réalisée grâce à une architecture reconfigurable permettant de s'adapter aux nombreux types de flux pouvant être conformes au standard SVC d' H.264. L'étude proposée a aboutie à une implémentation partielle d'un transcodeur du type SVC vers AVC. Nous proposons dans cette thèse une description des implémentations de transcodage concernant les formats AVC puis SVC.

Mots-clés : Transcodage, Compression vidéo, Structure logique, Systèmes adaptatifs

Abstract

Nowadays, a considerable number of high-tech device (PC, PDA, mobile phone ...) allow access to a multitude of multimedia files via a distribution network (Internet, 3G network .. .). The diversity of display capabilities and transport involves adaptations of multimedia content to make it releasable. The TOSCANE project aims to develop a distribution system based on said coding joint source and channel to improve the coverage of the broadcast video content. Two strategies will be implemented : the optimization of the distribution network and customer premises ADSL (WiFi broadcast, for example).

We have achieved, in this thesis work, a transcoder using requantization or troncature (selective transmission). This two methods have been compared and evaluated and the results show that one or the other of these methods is more efficient according to the context. Our work consists also in the study of the video standard derived from H.264/AVC, the standard H.264/SVC (for scalable video coding). We wanted to study a quality transcoder, but also in spatial resolution which will allow a rewriting of the SVC stream towards a AVC stream usable with current market decoder. This transposition is achieved through a reconfigurable architecture to adapt to many types of flow which may conform to standard SVC to H.264. The proposed study has led to a partial implementation of a transcoder type of SVC to AVC. We propose in this thesis describes the implementations of AVC formats for transcoding and SVC.

Key-words : Transcoding, Video compression, Logic structure, Adaptative systems

Bibliographie

[AG96] P. ASSUNÇNO et M. GHANBARI : Post-processing of mpeg-2 coded video for transmission at lower bit-rate. *In Proceeding of IEEE International Conference Acoustics, Speech and Signal Processing, Atlanta*, 1996.

[AWSZ05] I. AHMAD, Xiaohui WEI, Yu SUN et Ya-Qin ZHANG : Video transcoding : an overview of various techniques and research issues. *Multimedia, IEEE Transactions on In Multimedia, IEEE Transactions on*, 7:793–804, 2005.

[BC98] N. BJÖRK et C. CHRISTOPOULOS : Transcoder architectures for video coding. *IEEE Transaction on Consummers Electronic*, 44:88–98, 1998.

[BC00] N. BJÖRK et C. CHRISTOPOULOS : Transcoder architectures for networked multimedia. *Proc. IEEE International Symposium Circuits and Systems*, 4:25–28, 2000.

[BDG+07] K. BABIONITAKIS, G. DOUMENIS, G. GEORGAKARAKOS, G. LENTARIS, K. NAKOS, D. REISIS, I. SIFNAIOS et N. VLASSOPOULOS : A real-time h.264/avc vlsi encoder architecture. *Journal of Real-Time Image Processing*, 3:43–59, 2007.

[BKI04] Jens BIALKOWSKI, André KAUP et Klaus ILLGNER : Fast transcoding of intra frames between h.263 and h.264. *International Conference on Image Processing*, 4:2785–2788, 2004.

[CAK+09] Ngai-Man CHEUNG, Oscar C. AU, Man-Cheung KUNG, Peter H.W. WONG et Chun Hung LIU : Highly parallel rate-distortion optimized

intra-mode decision on multicore graphics processors. *IEEE Transaction on Circuits and Systems for Video Technology*, 19:1692–1703, 2009.

[CFAK10] Nagai-Man CHEUNG, Xiaopeng FAN, O.C. AU et Man-Cheung KUNG : Video coding on multicore graphics processors. *Signal Processing Magazine, IEEE*, 27:79–89, 2010.

[CHT+06] Tung-Chien CHEN, Yu-Wen HUANG, Chuan-Yung TSAI, Bing-Yu HSIEH et Liang-Gee CHEN : Architecture design of context-based adaptative variable-length coding for h.264/avc. *IEEE Transaction on Circuits and Systems-II : Express Briefs*, 53:832–836, 2006.

[CLSG06] Chih-Da CHIEN, Keng-Po LU, Yi-Hung SHIH et Jiun-In GUO : A high performance cavlc encoder design for mpeg-4 avc/h.264 video coding applications. *In Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–3841, May 2006.

[CM95] S.F. CHANG et D.G. MESSERSCHMIDT : Manipulation and compositing of mc-dct compressed video. *IEEE Journal on Selected Areas in Communications*, 13:1–11, 1995.

[CRG+09] Vinay CHANDER, Aravind REDDY, Shriprakash GAURAV, Nishant KHANWALKAR, Manish KAKHANI et Shashikala TAPASWI : Fast and high quality temporal transcoding architecture in the dct domain for adaptive video content delivery. *In ICCET '09 : Proceedings of the 2009 International Conference on Computer Engineering and Technology*, pages 91–97, Washington, DC, USA, 2009. IEEE Computer Society.

[CWLY09] Yi-Chih CHAO, Shih-Tse WEI, Bin-Da LIU et Jar-Ferr YANG : Combined cavlc decoder, inverse quantizer, and transform kernel in compact h.264/avc decoder. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19:53–62, 2009.

[CWYL06] Yi-Chih CHAO, Shih-Tse WEI, Jar-Ferr YANG et Bin-Da LIU : Combined cavlc decoder and inverse quantizer for efficient h.264/avc de-

coding. *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, .:259–262, 2006.

[DBKW05] M. DICK, J. BRANDT, V. KAHMANN et L. WOLF : Adaptive transcoding proxy architecture for video streaming in mobile networks. *In Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III – 700–3, sep. 2005.

[DCB+09] Christophe DEKNUDT, Patrick CORLAY, Anne-Sophie BACQUET, Françoix-Xavier COUDOUX, Michael GUARISCO, Hassan RABAH, Yves BERVILLER et Serge WEBER : Transrating by frequencies selectivity for h.264/avc intra pictures. *In Broadband Multimedia Systems and Broadcasting, 2009. BMSB '09. IEEE International Symposium on*, pages 1–7, 2009.

[DCNLVdW09] J. DE COCK, S. NOTEBAERT, P. LAMBERT et R. Van de WALLE : Architectures for fast transcoding of h.264/avc to quality-scalable svc streams. *Multimedia, IEEE Transactions on*, 11(7):1209 –1224, nov. 2009.

[Dia09] DIALOGIC : An introduction to dialogic software video transcoder (svt). Rapport technique, Dialogic, 2009.

[dSsKhJY10] Kwang deok SEO, Jin soo KIM, Soon heung JUNG et JeongJu YOO : A practical rtp packetization scheme for svc video transport over ip networks. *ETRI Journal*, 32:281–291, 2010.

[DWMZ03] Wu DI, Gao WEN, Hu MINGZENG et Ji ZHENZHOU : An exp-golomb encoder and decoder architecture for jvt/avs. *In ASIC, 2003. Proceedings. 5th International Conference on*, volume 2, pages 910 – 913 Vol.2, oct. 2003.

[Ele95] A. ELEFTHERIADIS : *Dynamic Rate Shaping of Compressed Digital Video*. Thèse de doctorat, Department of Electrical Engineering, Columbia University, New York, 1995.

[EV62] G. ESTRIN et C.R. VISWANATHAN : Organization of a "fixed-plus-

variable" structure computer for computation of eigenvalues and eigenvectors of real symmetric matrices. *Journal of ACM*, 9:41 – 60, 1962.

[Fuj09] FUJITSU : Full hd h.264/mpeg-2 encoder-transcoder ics. Rapport technique, Fujitsu, 2009.

[GM07] Tony Gladvin GEORGE et N. MALMURUGAN : The architecture of fast h.264 cavlc decoder and its fpga implementation. *Proceedings of the Third International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, 2:3§89–392, 2007.

[H26] Joint draft itu-t rec. h.264 | iso/iec 14496-10 / amd.3 scalable video coding.

[Hen08] Jean-Pierre HENOT : Powerful video compression for professional hd applications - mustang asic. Rapport technique, Thomson, 2008.

[HL09] Jian HUANG et Jooheung LEE : Efficient vlsi architecture for video transcoding. *Consumer Electronics, IEEE Transactions on*, 55(3):1462 – 1470, aug. 2009.

[hMD02] Wei hsiu MA et David H. C. DU : Reducing bandwidth requirement for delivering video over wide area networks with proxy server. *IEEE Transactions on Multimedia*, 4:539–550, 2002.

[HWL98] J.N. HWANG, T.D. WU et C.W. LIN : Dynamic frame-skipping in video transcoding. *In Proceeding of IEEE Workshop Multimedia Signal Processing*, 1998.

[ISOa] ISO/IEC : Coding of audio-visual objects - part 2 : Visual.

[ISOb] ISO/IEC : Information technology, generic coding of moving pictures and associated audio information : Video, 2nd edition.

[IT] ITU-T : Video coding for low bit-rate communication.

[JZy09] Zhang JIAN et Deng ZHI-YONG : Low complexity transcoder for mpeg-2 to h.264. *In Second International Workshop on Computer Science and Engineering*, 2009.

[Kes10] N. KESHAVENI : Design and fpga implementation of integer transform and quantization processos and thier inverses for h.264 video encoder. *International Journal of Computer Science and Communication*, 2010.

[KHHH96] G. KESSMAN, R. HELLINGHUIZEN, F. HOEKSMA et G. HEIDMAN : Transcoding of mpeg bitstream. *Signal Processing : Image Communication*, 8:481–500, 1996.

[KKH09] I. KOFLER, R. KUSCHNIG et H. HELLWAGNER : In-network real-time adaptation of scalable video content on a wifi router. *In Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1 –2, jan. 2009.

[Kor08] Reeba KORAH : Fpga implementation of integer transform and quantizer for h.264. *Journal of Signal Processing Systems*, 2008.

[Kow08] Cyril KOWALISKI : Badaboom 1.0 uses nvidia gpus to transcode video. Rapport technique, techreport.com, 2008.

[KPKH08] Ingo KOFLER, Martin PRANGL, Robert KUSCHNIG et Hermann HELLWAGNER : An h.264/svc-based adaptation proxy on a wifi router. *In NOSSDAV '08 : Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 63–68, New York, NY, USA, 2008. ACM.

[LBH04] Oussama LAYAIDA, Slim BENATALLAH et Daniel HAGIMONT : A framework for dynamically configurable and reconfigurable network-based multimedia adaptations. *In Journal of Internet Technology*, 2004.

[LC06] Yi-Ming LIN et Pei-Yin CHEN : An efficient implementation of cavlc for h.264/avc. *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, 3:601–604, 2006.

[Leo08] Alexis M. LEONTARIS, Athanasios ; Tourapis : Drift characterization of intra prediction and quantization in h.264. *Data Compression Conference*, pages 212–221, 2008.

[LH97] A. LAN et J.N. HWANG : Context dependant reference frame placement for mpeg video coding. *In Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing*, 1997.

[Li06] Haiyan LI : A streaming implementation of transform and quantization in h.264. *Lecture notes in computer science*, 2006.

[LM07] F. LONETTI et F. MARTELLI : Motion vector composition algorithm in h.264 transcoding. *In Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on*, pages 401–404, June 2007.

[MFA+07] Sandro MOIRON, Sergio FARIA, Pedro ASSUNÇÃO, Vitor SILVA et Antonio NAVARRO : H.264/avc to mpeg-2 video transcoding architecture. *Proc Conf. on Telecommunications*, 1:449–452, 2007.

[MHKK03] Henrique MALVAR, HALLAPURO, Marta KARCZEWICZ et Louis KEROFSKY : Low-complexity transform and quantization in h.264/avc. *IEEE transactions on circuits and systems for video technology ISSN 1051-8215*, 13:598–603, 2003.

[MKF+09] J. L. MARTINEZ, H. KALVA, W. A. C. FERNANDO, P. CUENCA et F. J. QUILES : Efficient wz-to-h.264 transcoding using motion vector information sharing. *In ICME'09 : Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, pages 1394–1397, Piscataway, NJ, USA, 2009. IEEE Press.

[NLCC07] Toan Dinh NGUYEN, Gueesang LEE, June-Young CHANG et Han-Jin CHO : Efficient mpeg-4 to h.264.avc transcoding with spatial downscaling. *ETRI Journal*, 29:826–828, 2007.

[PB06] Eric C. PEARSON et Harminder S. BANWAIT : Method and apparatus for transcoding between h.264 cabac and cavlc entropy coding modes, 2006.

[PK06] KY PARK et Hyuk KIM : Remote fpga reconfiguration using microblaze or powerpc processors. Rapport technique, Xilinx, 2006.

[PSG+08] F. PESCADOR, C. SANZ, M.J. GARRIDO, E. JUAREZ et D. SAMPER : A dsp based h.264 decoder for a multi-format ip set-top box. *Consumer Electronics, IEEE Transactions on*, 54(1):145 –153, feb. 2008.

[RB07] C.A. RAHMAN et W. BADAWY : Cavlc encoder design for real-time mobile video applications. *Circuits and Systems II : Express Briefs, IEEE Transactions on*, 54(10):873–877, Oct. 2007.

[Ric03] Iain E. G. RICHARDSON : *H.264 and MPEG-4 Video Compression*. WILEY, 2003.

[SC98] A.N. SKODRAS et C. CHRISTOPOULOS : Downsampling of compressed images in the dct domain. *In Proceeding of European Signal Processing Conference*, 1998.

[SEG06] Mohammad SHORFUZZAMAN, Rasit ESKICIOGLU et Peter GRAHAM : Cideo transcoding using network processor to support dynamically adaptive video multicast. *In Proceedings of the 20th International Conference on Advanced Information Networking and Application (AINA'06)*, 2006.

[SEG09] Mohammad SHORFUZZAMAN, Rasit ESKICIOGLU et Peter GRAHAM : In-network adaptation of video streams using network processors. *Hindawi Publishing Corporation Advances in Multimedia*, 2009:20, 2009.

[SG00] T. SHANABLEH et M. GHANBARI : Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding format. *IEEE Transaction Multimedia*, 2:101–110, 2000.

[SKZ96] H. SUN, W. KWOK et J. ZDEPSKI : Architecture for mpeg compressed bitstream scaling. *IEEE trans. Circuits Syst. Video Technology*, 6:191–199, 1996.

[SLB04] Bo SHEN, Sung-Ju LEE et Sujoy BASU : Caching strategies in

transcoding-enabled proxy systems for streaming media distribution networks. *IEEE Transaction On Multimedia*, 6:375–386, 2004.

[SPC05] Kibum SUH, Seongmo PARK et Hanjin CHO : An efficient hardware architecture of intra prediction and tq.iqit module for h.264 encoder. *In ETRI Journal*, 2005.

[SRS+04] Sriram SETHURAMAN, Arvind RAMAN, Kismat SINGH, Manisha Agrawal MOHAN, Neelakanth SHIGIHALLI et B.S. SUPREETH : Implementing an mpeg-2-to-h.264 transcoder on the dm642. Rapport technique, Embedded Edge, 2004.

[SSV97] B. SHEN, I.K. SETHI et B. VASUDEV : Adaptive motion vector resampling for compressed video downscaling. *In IEEE Proceeding of International Conference of Image Processing*, 1997.

[SVK01] Patrick SCHAUMONT, Ingrid VERBAUWHEDE et Kurt KEUTZER : A quick safari through the reconfiguration jungle. *In In Design Automation Conference*, pages 172–177. ACM Press, 2001.

[VCS03] A. VETRO, C. CHRISTOPOULOS et Huifang SUN : Video transcoding architectures and techniques : An overview. *Signal Processing Magazine, IEEE*, 20:18–29, 2003.

[VSDP98] A. VETRO, H. SUN, P. DAGRACA et T. POON : Minimum drift architecture for three layer scalable dtv decoding. *IEEE Transaction on Consummer Electronic*, 44:1012–1032, 1998.

[Vun07] Nicholas VUN : Development of h.264 encoder for a dsp based embedded system. *In Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, 2007.

[VYLS02] A. VETRO, P. YIN, B. LIU et H. SUN : Reduced spatio-temporal transcoding using an intra-refresh technique. *In Proceeding of IEEE International Symposium of Circuits and Systems*, 2002.

[WHZ+01] Dapeng WU, Yiwei Thomas HOU, Wenwu ZHU, Ya-Qin ZHANG et Jon M. PEHA : Streaming video over the internet : Approaches and

directions. *IEEE Transaction on Circuits and Systems for Video Technology*, 11:282–300, 2001.

[Wil09] Derek WILSON : Avivo video converter redux and ati stream quick look. Rapport technique, Anandtech, 2009.

[WWL+01] Q. WANG, F. WU, S. LI, Z. XIONG, Y.Q. ZHANG et Y. ZHONG : A new rate allocation scheme for prodressive fine granular scalable coding. *In Proceeding of IEEE International Symposium on Circuits and Systems*, 2001.

[YSL99] J. YOUN, M.T. SUN et C.W. LIN : Motion vector refinement for high performance transcoding. *IEEE Transaction on Multimedia*, 1:30–40, 1999.

[YWL00] P. YIN, M. WU et B. LUI : Video transcoding by reducing spatial resolution. *In Proceeding of IEEE International conference on Image Processing*, 2000.

[ZYB98] W. ZHU, K.H. YANG et M.J. BEACKEN : Cif to qcif video bitstream down-conversion in the dct domain. *Bell Labs Tech. Journal*, 3:21–29, 1998.