N° d'ordre : ***03/2019***

THESE DE DOCTORAT

Présentée par

Mr: ABDESSAMAD EL ANSARI

Spécialité : Systèmes Embarqués

Sujet de la thèse :

Etude, conception et implémentation d'un système embarqué temps réel de compression vidéo haute définition sur une architecture dédiée multi-composants.

Formation Doctorale : Sciences de l'ingénieur Sciences Physiques, Mathématiques et Informatique.

Thèse présentée et soutenue le samedi 23 février 2019 à 09h30 à l'Amphi Al Khawarizmi devant le jury composé de :

Nom Prénom	Titre	Etablissement	
Mohammed JORIO	PES	Faculté des Sciences et Techniques de Fès	Président
Lhoussine BAHATTI	PES	Ecole Normale Supérieure de l'Enseignement Technique de Mohammedia	Rapporteur
Adnane ADDAIM	PES	Ecole Nationale des Sciences Appliquées de Kenitra	Rapporteur
Saad BENNANI DOSSE	PES	Ecole Nationale des Sciences Appliquées de Fès	Rapporteur
Anas MANSOURI	PH	Ecole Nationale des Sciences Appliquées de Fès	Examinateur
Hassane EL MOUSSAOUI	PH	Faculté des Sciences et Techniques de Fès	Examinateur
Ali AHAITOUF	PES	Faculté des Sciences et Techniques de Fès	Directeur de thèse

Laboratoire d'accueil : Laboratoire Energies Renouvelables et Systèmes Intelligents.

Etablissement : Faculté des Sciences et Techniques de Fès.

Table des matières

Introduction **1**

0.1 introduction 1

0.2 Objectif de cette thèse 2

0.3 Méthodologie de travail 3

0.4 Structure du rapport 4

1 Technologies des plateformes de l'implémentation **7**

1.1 Introduction 7

1.2 Technologies de l'implémentation logiciel 8

1.3 Technologies d'implémentation matériel 16

1.4 Technologies Co-désigne logiciel matériel 21

1.5 Le traitement en temps réel du codage vidéo 24

1.6 L'état de l'art des implémentations software 27

1.7 L'état de l'art des implémentations hardware 29

1.8 Résumé 33

2 le système de codage vidéo **35**

2.1 introduction au codage vidéo 35

2.2 Historique des normes d'encodage vidéo 36

2.3 La norme de codage vidéo HEVC 37

2.4 Structure du bloc 41

2.5 Les paramètres de norme HEVC 47

2.6 Résumé 48

3 Implémentation software du codeur HEVC **51**

3.1 Introduction 51
3.2 Implémentation 53
3.3 L'implémentation du codeur sur DSP 54
3.4 Conclusion 61
3.5 L'implémentation de Décodeur sur ARM 64
3.6 l'implementation de l'estimation du mouvement rapide sur DSP 70
3.7 Résumé 73

4 Contribution matériel sur la norme HEVC 75

4.1 Introduction 76
4.2 L'architecture de l'intra prédiction 76
4.3 Comparaison avec l'état de l'art du FPGA. 83
4.4 L'implémentation de la ICT sur la FPGA 88
4.5 L'implémentation de mode décision 94
4.6 Comparaison avec d'autres ouvrages 110
4.7 Co-désigne entre système logiciel et matériel 112
4.8 Résumé 117

5 Chapitre I. Le système de codage vidéo 119

5.1 Introduction au codage vidéo 119
5.2 Analyse aux limites 120
5.3 Vérification par simulation numérique 121

Conclusion 123

A Linux embarqué 125

A.1 linux embarqué 125

B codec vidéo 129

B.1 Pré-traitements 129

B.2 Encodage 130

B.3 Transmission 131

B.4 Décodage 132

B.5 Post-traitement 132

B.6 Le schéma de codage vidéo hybride 132

C profilage de codec **135**

Bibliographie **145**

Table des figures

1.1 Exemple de registre Q (128 bits) qui tronque en 32 bits. 11

1.2 L'architecture interne du Cortex-A8. 12

1.3 Architecture de base d'un DSP [Sha98]. 13

1.4 schéma de l'architecture interne du DSP C6678. 15

1.5 L'EVM TMS320C6678 utilisant dans ce travail et l'architecture interne de TMS320C6678.[Noal] 17

1.6 Evolution du prix, de la consommation, du coût et de la vitesse du FPGA 2011. 18

1.7 exemple d'un architecture interne FPGA avec des éléments de base. 19

1.8 FPGA Evolution. Les FPGA ont ajouté de l'IP durci depuis 2000 et sont maintenant plus SoC que les FPGA [Noai]. 20

1.9 diagramme interne de processeur Zynq avec tous les éléments de l'architecture [Noaj]. 23

2.1 Représentation chronologique de l'élaboration des principales normes de codage vidéo. 37

2.2 exemple d'un CTU devisé en multi CUs. 38

2.3 Exemple des Tus dans l'unité de codage (CU). 39

2.4 Schéma fonctionnel de l'encodeur HEVC avec les modules principaux. 40

2.5 le Schéma fonctionnel du décodeur du HEVC. 41

2.6 exemple de I, B et P frames. 41

2.7 les modes de partitionnement et les formes de PB Luma. 42

2.8 les différents modes de l'Intra prédiction utilisé par le standard HEVC. 43

2.9 (a) Partitionnement, (b) : Image prédite, (c) : Image résiduelle 44

2.10 Les outils de l'algorithme CABAC : binarisation, sélection du contexte et codage arithmétique. 46

2.11 Points d'entrée frame pour démarrer le décodage aux positions des octets indiquées dans l'en-tête de tranche. 47

2.12 partitionnement de la même séquence en des tiles et slices. 48

2.13 Traitement parallèle par front d'onde de Nc CTU par ligne. Les décodeurs peuvent commencer à décoder chaque ligne avec un retard de deux CTU par rapport à la ligne précédente. 48

3.1 BasketBalldrill,(b) :PartyScence, (c) :BQMall, (d) :RaceHorce, la valeur de QP=32. 56

3.2 Comparaison de la complexité des trois configurations AI, LD et RA pour la séquence QBMall avec QP=32. 59

3.3 La comparaison de l'implémentation de l'accélération de la technologie NEON entre ARM cortex-A8 et cortex-A9 pour des valeurs QP égales à 27 et 32. . . 63

3.4 L'algorithme d'EM proposé. 71

3.5 Modèle de recherche diamant. 71

3.6 Modèle de recherche SDSP. 72

3.7 le Modèle de recherche HDS 72

3.8 le Modèle de recherche LDPS 73

4.1 La direction différente pour 33 modes angulaires. 78

4.2 L'architecture globale pour le calcul de la prédiction intra. Algorithme dans le codeur HEVC. 80

4.3 L'architecture proposé pour le traitement DC Intra prédiction avec 64 échantillons d'entrée. 81

4.4 L'élément de traitement de l'accélérateur matériel (PE) proposé pour le calcul des valeurs planaire et angulaire de la prédiction intra-prévision. 82

4.5 Placer les pixels voisins dans la mémoire RAM. Px indique le pixel avec x = 0, ..nTbS * 2 - 1 et nTbS est égal à la longueur fois la largeur du bloc. 83

4.6 simulation du stockage des données sur RAM par le bus de l'écriture et lecteur. 84

4.7 Le plan montre l'emplacement du matériel IP sur la puce Zynq. La région reconfigurable est marquée par le rectangle bleu. 85

4.8 Analyse de puissance de la prédiction intra IP sur Zynq 7000 tous les SOC programmables. 86

4.9 Architecture matérielle de l'élément de traitement (PE). 92

4.10 Architecture matérielle pour la matrice B2. 93

4.11 Architecture matérielle de la matrice C2. 93

4.12 Accélérateur matériel de la commande-N 1-D DCT transformée. 94

4.13 Architecture proposée pour le calcul de la transformation DCT 2D. 95

4.14 la simulation de accélérateur hardware sur le logiciel modelsim pour le matrice 4x4. 96

4.15 simulation de accélérateur hardware sur le logiciel Modelsim pour le matrice 32x32. 96

4.16 Le nombre d'appels et le temps de chargement des fonctions SAD dans le codeur HEVC. 97

4.17 Architecture proposée pour le calcul de la transformation DCT 2D. 98

4.18 Comparateur à 1 bit utilisé dans la Comp. 101

4.19 Comparateur 1 bit utilisant dans la Comp. 102

4.20 Elément de traitement (PE) pour huit entrées et une sortie pixels. 103

4.21 L'architecture globale de l'algorithme SAD dans IME pour les codeurs HEVC. 104

4.22 Emplacement des pixels d'entrée des données Sub-CU 64×64. 105

4.23 Section proposée de la CLA 106

4.24 Proposed CLA n-bit adder based on architecture of CLA adder. 107

4.25 Construction des SAD 8×4 et 4×8 à partir de PE et aussi la construction de 8×8 à partir de deux 8×4. 108

4.26 la simulation de accélérateur hardware sur le logiciel Modelsim pour le bloc SAD 8x8. 109

4.27 la plateforme microzed 114

4.28 Proposed CLA n-bit adder based on architecture of CLA adder. 115

4.29 Architecture matérielle système sur puce proposée sur Zynq-7000. 116

5.1 Exemple de courbe TikZ. 120

A.1 Organigramme du processus de chargement d'un OS [RLN05]. 125

A.2 Charge Boot-Room Charge FSBL. 126

B.1 Espace couleur RVB. En haut à gauche : combinaison des couleurs primaires rouge, vert et bleu ; en bas : les composantes rouge, vert et bleu de l'image couleur en haut à droite. 130

B.2 Décomposition de l'image en composantes : luminance(Y) et chrominance (U et V) 130

B.3 schéma hybride de encodeur avec un décodeur à l'intérieur de l'encodeur . . . 133

C.1 présentation graphique du profilage de l'encodeur 136

C.2 présentation graphique du profilage de décodeur 137

Liste des tableaux

1.1 comparaison des caractéristiques principales deux processeurs ARM Cortex-A8 et Cortex-A9 13

3.1 Quatre séquences de test sont utilisées pendant l'expérimentation. 53

3.2 Le temps d'encodage obtenu pour les 150 images de quatre séquences qui sont choisies pour les tests en secondes. 54

3.3 Le profilage du codeur HEVC en pourcentage de blocs 57

3.4 CODING UNIT PARTITIONING OF PUS FOR EACH DIFFERENT CODING DEPTH LEVEL. 60

3.5 Comparaison des performances de notre exécution avec d'autres sur DSP pour 832x480 (480p) requises par fps. 60

3.6 Comparaison de la complexité de notre travail avec d'autres travaux de configuration RA. 61

3.7 FRÉQUENCE DE TRAME OU CODAGE DE TRAME PAR SECONDE (FPS) POUR CHAQUE CLASSE. 65

3.8 LA VITESSE ATTEINTE GRÂCE À LA TECHNOLOGIE DU NÉON. . . . 66

3.9 Résultats du profilage du décodeur avec la technologie Neon 68

3.10 comparaison de la répartition du temps entre les blocs de HEVC. 69

4.1 résultats de la mise en œuvre de l'architecture proposée par le FPGA. 84

4.2 Comparaison des résultats de synthèse de l'implémentation sur la FPGA. . . 87

4.3 COMPARAISON AVEC D'AUTRES TRAVAUX POUR UNE MISE EN ŒUVRE PLUS RAPIDE. 87

4.4 les nombres existants dans tous tailles de matrices 1-D DCT. 91

4.5 RÉSULTATS DE SYNTHÈSE DE L'ARCHITECTURE MATÉRIELLE. . . 95

4.6 COMPARAISON AVEC D'AUTRES TRAVAUX 97

4.7 L'UNITÉ DE CODAGE SÉPARANT DU PUs POUR CHAQUE DIFFÉRENT NIVEAU DE PROFONDEUR. 98

4.8 résultats de la mise en œuvre de l'architecture proposée par le FPGA. 110

4.9 Dissipation totale de la puissance sur puce (statique et dynamique) par conception pour la décision en mode SAD avec plus de statistiques. 111

4.10 LES RÉSULTATS DE SYNTHÈSE DE L'ARCHITECTURE PROPOSÉE . 111

4.11 comparaison de l'implémentation de l'algorithme de prédiction intra dans différentes plates-formes pour Basketballdrive (1920 × 1080 pixels) avec tous les intra et qp=32. 117

Introduction générale

0.1 introduction

Le trafic vidéo sur Internet augmente rapidement et devrait représenter environ 82% du trafic Internet total des consommateurs d'ici 2020. Pour y remédier, de nouvelles normes de codage vidéo telles que HEVC /H.265 *(High efficient video conding)* offrent une meilleure compression, en particulier en Utra HD en résolutions vidéo et supérieures. HEVC archives ceci par une variété de techniques algorithmiques telles que de plus grandes tailles de transformée et une prédiction inter-trame plus précise, cependant, ces techniques augmentent la complexité codage logiciels et matériels d'encodeur et décodeur.

L'évolution constante des normes de codage vidéo et les modifications successives qui y sont apportées rendent le temps disponible pour le développement de nouvelles versions d'encodeurs et décodeurs de plus en plus courts.

L'encodage et le décodage vidéo sont des applications très exigeantes qui nécessitent des supports technologiques avec une grande capacité de calcul. Actuellement, les différentes alternatives technologiques utilisées pour mettre en œuvre ce type d'applications peuvent être classées en trois grands groupes : celles basées sur des processeurs à usage général, celles basées sur des architectures spécifiques et celles basées sur le développement des puces numériques (sur FPGA) a une ligne de recherche dans les architectures matérielles orientées vers le codage vidéo numérique dans la FST de Fés. Dans le cadre de cette recherche, deux thèses de doctorat de Professeur Monsouri Anas [Noaa] et Monsieur Fahed Benboubker [Noab] ont été développées dans lesquelles des architectures avancées spécifiques ont été proposées pour la production de codeurs vidéo.

Cependant, à partir de 2004, certaines entreprises ont commencé à commercialiser des processeurs de signaux numériques (DSP) de très haute performance, dotés d'une grande capacité de calcul, qui permettent de mettre en œuvre des applications d'encodage et de décodage vidéo sur ceux-ci. De plus, ces processeurs intègrent une série de périphériques (ports vidéo d'entrée et de sortie, convertisseurs numérique-analogique ou accès au réseau local) qui facilitent les interfaces typiques de ce type d'applications, permettant d'implémenter des systèmes complets sur une seule puce (System on Chip ou SoC).

Les avantages de ces processeurs par rapport aux architectures matérielles spécifiques résident dans la facilité de modification de la norme qu'ils implémentent, puisqu'il s'agit uniquement du remplacement du logiciel qui exécute le processeur. Cette polyvalence permet des systèmes de codage/décodage avec des temps de développement plus courts que ceux obtenus avec des implémentations basées sur des architectures matérielles.

Un autre type de SoC sont ceux qui sont intégrés par des processeurs à usage général dédié pour l'embarqué (ARMs). Alors que les ARMS ont toujours été associés à des appareils

haute performance dans les combinés mobiles, les fabricants de puces commercialisent déjà des appareils moins performants que les ARMs classiques destinés à être intégrés dans les combinés mobiles.

Les méthodologies proposées dans la thèse précitée sont valables pour des architectures avec un multiple cœur DSP. Actuellement, deux facteurs empêchent la mise en œuvre de décodeurs vidéo fonctionnant en temps réel avec ce type d'architecture. D'une part, les nouvelles normes de codage telles que le codage vidéo à haute efficacité (HEVC) [ISO][UTI] exigent une capacité de calcul supérieure à celle de leurs prédécesseurs, et d'autre part, les résolutions spatiales et temporelles des images sont de plus en plus élevées (4K et 8K), ce qui implique également un coût informatique supérieur (plus de développements logiciels).

Pour tenir compte de l'implémentions de cette nouvelle norme et prendre en charge des résolutions spatiales élevées, il est nécessaire d'utiliser plus d'un processeur.

La charge de calcul du processus de codage peut être répartie entre plusieurs processeurs. En ce sens, début 2012, certains des fabricants les plus représentatifs ont commercialisé des appareils qui intègrent plusieurs DSP (désormais des processeurs Multi-DSP). Ces appareils ont une grande capacité de calcul pour une consommation réduite, ce qui les rend particulièrement adaptés aux applications qui doivent fonctionner de manière autonome avec des batteries.

Le partitionnement de la charge de calcul d'un algorithme fonctionnant sur un système multi-cœur est une question qui a été largement abordée dans les systèmes complexes intégrant des systèmes d'exploitation (OS) tels que Linux ou Windows. Des groupes de recherche travaillent déjà sur des méthodologies pour optimiser automatiquement la parallélisassions des algorithmes entre différents processeurs [Cha+01], [AD98], bien qu'à ce jour ces travaux se soient concentrés sur les processeurs à usage général. Cependant, les systèmes d'exploitation linux embarqué tels que uclinux [McC04] utilisés par les processeurs multi-DSP n'intègrent pas les facilités fournies par les SO haute performance, c'est donc au concepteur qu'il incombe de réaliser manuellement la division. Cela implique un travail laborieux et peu flexible puisque les changements dans la répartition de la charge de calcul entre les processeurs nécessitent des modifications importantes et un coût élevé en termes de temps de développement. Ces dernières années, les développeurs de standard ISO et UTI ont orienté ses recherches dans le domaine des architectures de codage vidéo vers des méthodologies des implémentations codeurs vidéo de dernière génération avec processeurs multi-cœurs. Dans ce contexte, cette thèse participe aux projets morocco-tunissien.

0.2 Objectif de cette thèse

Depuis leur introduction il y a trente ans, les réseaux de portes programmables sur site (FPGA) ont évolué, passant d'une simple logique de collage à une concurrence avec les circuits intégrés spécifiques à l'application (ASIC) conçus sur mesure. Les FPGA modernes comprennent des centaines de milliers de portes logiques programmables auxquelles s'ajoutent des

milliers de blocs de traitement numérique du signal (DSP) et de blocs de mémoire configurables, tous sur la même puce avec un tissu de routage flexible. La souplesse de routage et de configuration de ces nombreux blocs matériels permet aux FPGA de bénéficier de leur parallélisme intrinsèque ; les FPGA sont donc exploités pour le calcul massivement parallèle et peuvent être adaptés comme un accélérateur pour les modules de décodeur/encodeur vidéo.

Ces systèmes massivement parallèles exigent des structures de mémoire très parallèles pour suivre le rythme de leur nature concurrente puisque les mémoires sont généralement le goulot d'étranglement des performances de calcul. Dans cette thèse, nous proposons de nouvelles façons de construire de structures de mémoire parallèles dans les FPGA, en utilisant la logique régulière, les registres et les blocs de mémoire double accès que l'on trouve dans les FPGAs modernes.

0.3 Méthodologie de travail

Les recherches menées dans le cadre de cette thèse de doctorat sont de type documentaire et expérimental. En ce qui concerne la partie documentaire de la recherche, la méthodologie utilisée a consisté en une revue approfondie de l'état de l'art du point de vue théorique sur le codage vidéo et des propositions visant à organiser systématiquement les connaissances acquises et à identifier les problèmes existants dans l'implémentation de codage vidéo dans le système embarqué, ainsi que dans les propositions visant à les résoudre. Ce processus a permis de définir un ensemble d'objectifs spécifiques pour aborder les problèmes identifiés et qui représentent les contributions de la thèse de doctorat. Dans le processus de définition des objectifs, le FPGA et le DSP ont été considérés comme des technologies favorables, par conséquent, une compréhension approfondie de ces technologies a également été nécessaire.

En ce qui concerne la partie expérimentale de la recherche, un cas d'utilisation a été soulevé qui a nécessité l'élaboration d'une preuve de concept pour valider les propositions présentées dans la thèse de doctorat. Afin de développer la validation de principe, les différents modules complexes du codage la nouvelle norme vidéo HEVC ont été implémentés, y compris le schéma de l'encodeur et décodeur proposée le standard par un scénario de validation composé du système embarqué. Dans le scénario de validation, différentes expériences ont été réalisées afin d'observer le comportement de l'architecture proposée pour obtenir des résultats qui démontrent la faisabilité des propositions.

Il est important de mentionner que, compte tenu des résultats obtenus dans la partie expérimentale de la recherche, les questions en suspens ont été identifiées qui doivent être abordées ou qui nécessitent un plus grand degré d'analyse et qui constituent donc des axes de recherche futurs.

La méthodologie de travail utilisée comprend quatre phases : 1) Etude de l'état de l'art dans l'implémentation des algorithmes de traitement la vidéo dans les plates-formes processeurs et FPGAs. D'autres contributions publiées dans la littérature scientifique ont fait l'objet d'une étude détaillée sur les applications et les outils qui facilitent la répartition de la charge

de calcul entre les coeurs de traitement. Cette étude ne s'est pas limitée aux applications de traitement vidéo, ni aux systèmes basés sur des processeurs Multi-DSP ou FPGAs, mais a été généralisée à d'autres types d'applications et à l'utilisation d'autres plates-formes pour avoir une vision plus générale de l'implémentation des algorithmes basés sur des systèmes DSP et FPGA.

2) implémentation d'un codeur vidéo sur une plateformes Multi-DSP et ARM. Pour réaliser les expériences sur la complexité qui ont permis de mettre en évidence les différents problèmes, d'essayer des solutions, en définitive, d'en tirer les conclusions. Qui ont alimenté le développement de la méthodologie de travail proposée, un codeur open source de norme HEVC a été implémenté sur une plate-forme Multi-DSP et muti-ARM. En effet, une plate-forme de développement appropriée a été choisie parmi celles existant sur le marché [Noaj], [Noam]. A partir d'une implémentation d'open source HM du codeur, le processus de portage vers la technologie DSP et ARM et le temps de la charge de calcul des différents modules de traitement i.e. le profilage ont été effectués, parmi d'autres tâches nécessaires.

3) Synthèse de la méthodologie. L'expérience acquise lors de l'implémentation du codeur HEVC a permis de tirer des conclusions méthodologiques qui ont permis de définir une méthodologie de travail permettant de réduire les temps de développement de codeur vidéo et de flexibiliser l'implémentation dans les systèmes Multi-DSP et multi-ARM avec une analyse complète de la complexité associée.

4) Création d'un outil de conception. Les différentes étapes de la méthodologie de conception proposée ont été intégrées dans un outil qui permet de bien adopter notre proposition logicielle. L'outil a été vérifié avec la conception de plusieurs codeurs HEVC sur différentes plateformes basées sur des DSPs, mais aussi sur des ARMs d'une part, et d'autre part, pour avoir des meilleures architectures matérielles sur FPGA et aussi sur ASIC.

Ces quatre phases n'ont pas été développées de manière complètement séquentielle. L'effort principal de la première phase a été fait au début de la thèse, mais il a été nécessaire de continuer à mettre à jour les informations recueillies avec de nouvelles contributions qui ont continué à être publiées tout au long de l'élaboration de la thèse. Les deuxièmes et troisièmes phases se sont déroulées en parallèle et se sont reconstituées l'une l'autre pour une grande partie de l'élaboration de la thèse. La quatrième phase a été réalisée dans la dernière partie de la thèse, une fois les différentes étapes de la méthodologie de conception établies.

0.4 Structure du rapport

Le rapport présente la thèse de doctorat, dans lequel sont résumés les travaux développés au cours des dernières années est structuré en chapitres de la manière suivante :

- Chapitre I. Introduction générale La première partie est un chapitre qui se compose exclusivement l'introduction générale, puis les objectifs de la thèse de doctorat i.e. le cadre de cette thèse ou le cadre du travail de recherche développé dans le cadre des tâches de l'équipe

Micro-électronique et système embarqué de laboratoire ERSI de la FST de Fés à fin d'être bien est établi. Cette première partie présente également les objectifs de la thèse de doctorat. Ensuite, la méthodologie de travail dans cette thèse qui se divise en quatre partie essentielles, en fin, la structure du rapport de la thèse.

- Chapitre II, Technologies des plateformes de l'implémentation Le chapitre II présente des généralités sur les systèmes embarqués dédié pour le traitement de la vidéo tels que les FPGA et des DSP d'une part, et d'autre part, il contient des informations pertinentes sur les caractéristiques des plates-formes utilisées pour réaliser les tests et recueillir les résultats pendant la phase de développement. On traite des principales technologies utilisées pour l'implémentation des encodeurs et décodeurs de la norme vidéo HEVC. En ce sens, les principales caractéristiques des architectures des plates-formes utilisées pendant la phase de développement de ce travail sont analysées. L'état de l'art des mises en œuvre réalisées et publiées dans la littérature scientifique par d'autres groupes de recherche est également analysé.

- Chapitre III, Le système de codage vidéo Ce chapitre présente les dernières techniques de codage vidéo numérique et présente les principes du HEVC et des normes de codage vidéo [Sul+12], clés pour l'élaboration de la thèse.

- Chapitre IV, Implémentation software de codeur HEVC Le chapitre IV décrit en détail les différentes étapes de l'élaboration de la méthodologie proposée pour les implémentations logicielles sur des plateformes DSPs et ARMs. Nous présentons ici, en détail, tous les étapes et les résultats qui doivent être pris en compte lors de la conception de décodeur et encodeur vidéo sur les plates-formes Multi-DSP et multi-ARMs. Sont également incluses les améliorations incluses dans la phase de conception concernant les techniques de réduction de la consommation de temps de l'exécution et d'optimisation des performances à l'aide du logiciel proposé pour les estimateurs de mouvements. De plus, cette partie contient la phase de l'état de l'art des méthodologies de conception logiciels sur les plates-formes différents inclus qui sont utilisées pour l'embarqué.

Chapitre 1

Technologies des plateformes de l'implémentation

Sommaire

1.1 Introduction 7
1.2 Technologies de l'implémentation logiciel 8
1.2.1 Architecture de général processeur dédié pour le système embarqué 10
1.2.2 Architecture du processeur DSP 12
1.3 Technologies d'implémentation matériel 16
1.3.1 Technologie de la FPGA 18
1.3.2 Accélérateur matériel (Vivado) 20
1.3.3 Technologie de la ASIC 21
1.3.4 Langage de programmation matériel 21
1.4 Technologies Co-désigne logiciel matériel 21
1.4.1 L'architecture du processeur Zynq 22
1.4.2 Langage de programmation logiciel C/C++ 24
1.5 Le traitement en temps réel du codage vidéo 24
1.5.1 Notions de temps réel 24
1.5.2 Traitement embarquée du codage vidéo 25
1.5.3 Systèmes matériels de traitement de la vidéo 26
1.6 L'état de l'art des implémentations software 27
1.7 L'état de l'art des implémentations hardware 29
1.8 Résumé 33

1.1 Introduction

Récemment, il y a eu des propositions concernant l'évaluation comparative du traitement numérique du codage vidéo et l'obtention d'un meilleur rendement tout en utilisant des plate-forme conçues associée à une faible consommation d'énergie. L'utilisation d'architectures informatiques avec des accélérateurs développés nécessite une connaissance approfondie du système ainsi que des méthodes et des compétences de programmation irrégulières. Ces systèmes sont composés de processeurs de base intégrés à des réseaux de portes programmables sur le terrain (FPGA) et/ou à des unités de programmation graphique (GPU). L'architecture

DSP est l'un de ces types de processeurs. L'un des avantages du DSP est qu'il évite les problèmes de performance liés à l'intégration des accélérateurs dans les systèmes informatiques. Le TI TMS320C6678 multi-cœur DSP promettait d'avoir un bon compromis entre puissance, coût efficace et d'offrir le même paradigme de programmation que les processeurs multi-cœur tels que les processeurs ARMs [GB13].

L'un des facteurs major du développement des processeurs DSP est l'augmentation de la vitesse, la diminution de la consommation d'énergie, ainsi que la diminution de l'utilisation de la mémoire et des coûts [Ali+12].

Les recherches de certains chercheurs dans la littérature indiquent que les pipelines les moins profonds avec une exécution en ordre se sont avérés être les plus importants en termes de surface et d'énergie. Le traitement efficace tenu de ces considérations physiques et micro-architecturales, nous pensons que les éléments particuliers des futures architectures seront probablement des processeurs, des unités à virgule flottante, des vecteurs et des éléments de traitement SIMD simples et modestement pipelinés (plusieurs étages) [ZBG88]. Ce résultat de l'état d'art montre que le microprocesseur DSP de Texas Instruments incarne cette recommandation de nouveaux processeurs celle trouver chez AMD, International Business Machines (IBM), Intel, NVIDIA, NVIDIA, ARM etc.

Dans cette thèse, nous nous concentrons sur le développement, l'implémentation et l'optimisation d'un algorithme de décodeur et l'encodeur de la norme HEVC sur DSP et ARM. Les dispositifs DSP ont été principalement utilisés pour le codage audio/vidéo [McC94], le traitement de la vidéo (nécessite des opérations matricielles lourdes) [KMG06], ainsi que pour des applications en temps réel qui incluent le streaming vidéo [KMG06].

Des architectures matériels d'un ou plusieurs blocs de l'algorithmes l'encodeur HEVC ont été développés afin d'utiliser moins ressources et d'avoir le traitement en temps réel pour des très haute résolutions (4K et 8K en utilisant dans technologies de la FPGA et ASIC. La FPGA Artex-7 utilisé est celle de la société Xilinx dont on donnera le détail par la suite, elle a été embarqué dans une puce avec deux processeurs ARMs sur une seule puce nomé Zynq, donc, une partie de travail logiciel et matériel de cette thèse ont été élaboré sur cette carte.

1.2 Technologies de l'implémentation logiciel

L'approche évidente est d'augmenter la vitesse de l'horloge, et donc la vitesse à laquelle les instructions sont exécutées. D'énormes progrès ont été réalisés dans ce domaine, avec des vitesses d'horloge maximales de plusieurs GHz qui sont la norme pour les plates-formes informatiques. Ces augmentations sont principalement dues à la réduction des délais de propagation due aux progrès de la technologie des semi-conducteurs, qui a permis de réduire à la fois la taille des caractéristiques des transistors et la tension d'alimentation. Ce n'est pas sans problème, cependant, et l'une des conséquences d'une vitesse d'horloge plus élevée est une augmentation significative de la consommation d'énergie de l'unité centrale.

Bien que les vitesses de la mémoire de masse aient également augmenté (nand flache), elles n'ont pas suivi le rythme de l'augmentation des vitesses du DSP. Cela a causé des problèmes pour lire les instructions et les données de la mémoire du système à la vitesse requise. Des techniques de mise en cache ont été mises au point pour mettre en mémoire tampon les instructions et les données dans une mémoire plus petite à grande vitesse qui peut suivre le rythme du DSP. Le problème est alors comment maximiser la probabilité que les données requises seront dans la mémoire cache plutôt que dans la mémoire principale plus lente. Les oublis de cache (lorsque les données ne sont pas dans la mémoire cache) peuvent entraîner une dégradation significative des performances du processeur car le processeur est inactif en attendant que les données soient récupérées.

Les instructions et les données doivent être accessibles à partir de la mémoire principale. Une amélioration évidente est l'utilisation d'une architecture optimisée, qui double la bande passante effective de la mémoire en utilisant des mémoires séparées pour les instructions et les données. Cela peut donner une amélioration de la vitesse jusqu'à un facteur de deux. Une amélioration similaire peut être obtenue avec une seule mémoire principale, en utilisant des caches séparés pour les instructions et les données. Une autre approche consiste à augmenter la largeur de l'ALU, de sorte que plus de données soient traitées à chaque cycle d'horloge. Cela donne des améliorations évidentes pour des longueurs de mots de données plus importantes (par exemple des nombres à virgule flottante de double précision) car chaque nombre peut être chargé et traité en moins de cycles d'horloge. Cependant, lorsque la longueur naturelle des mots est plus étroite, comme c'est généralement le cas dans le traitement de la vidéo, l'amélioration des performances n'est pas aussi importante à moins que la largeur de bande de la mémoire de données ne soit le facteur limitant. Lorsque le chemin de données est plus large que la longueur du mot, il est également possible de concevoir l'Unité Arithmétique et Logique (UAL) pour fonctionner comme un processeur vectoriel. Plusieurs mots de données sont regroupés dans un seul mot processeur permettant à l'UAL d'effectuer la même opération simultanément sur plusieurs éléments de données (par exemple en utilisant les instructions DSP VLIW ; C6000 DSP [TMS00]). Ceci peut être particulièrement efficace pour les opérations de traitement de la vidéo pour les faibles résolutions où la même opération est effectuée sur chaque pixel.

La vitesse du cycle d'extraction/décodage/exécution peut être améliorée par la mise en pipeline des phases séparées. Ainsi, pendant qu'une instruction est exécutée, l'instruction suivante est décodée ou encodée et l'instruction suivante est récupérée. Selon la complexité des instructions, il peut également y avoir des phases (et des étapes de pipeline) pour lire les données de la mémoire et écrire les résultats dans la mémoire effectuer par LINUX embarqué. Ce type de traitement est efficace lors de l'exécution de longues séquences d'instructions (plusieurs séquences à traiter). Cependant, le traitement d'instruction devient plus complexe avec des boucles ou des branches qui cassent la séquence régulière. Pour ce faire, il est nécessaire de réduire les séquences traiter d'instructions afin d'enlever les instructions successives qui n'ont pas été correctement chargées, puis de charger les instructions correctes. Des techniques ont été mises au point pour tenter de réduire au minimum le temps perdu lors du chargement des séquences vidéo. La prédiction des séquences à traiter dans le fichier de configuration et pris de telle sorte qu'elle un chemin courte afin qu'on augmente la probabilité que les instructions

correctes seront chargées i.e. on adopte l'exécution codeur HEVC à l'environnement de la carte embarquée à partir de fichier de fichier de configuration de code source. Exécute les instructions qui ont déjà été chargées dans le pipeline, dans le but que le temps ne soit pas perdu si la branche est mal prévue, les résultats de l'exécution sont augmentés.

1.2.1 Architecture de général processeur dédié pour le système embarqué

1.2.1.1 Architecture de processeur embarqué ARM

Pour répondre aux besoins des systèmes embarqués, différents fabricants ont développé des processeurs basés sur des cœurs ARMs (Exynos, snapdragon, kirn ou bionic de apple) moins performants que les précédents. Ces processeurs sont généralement utilisés dans les terminaux portables, qu'il s'agisse de téléphonie mobile ou d'autres appareils multimédias dont la source d'alimentation est une batterie. L'utilisation de cette technologie alternative a connu une augmentation constante au cours de de dernière dizaine d'années, principalement en raison de son faible coût, de sa faible consommation d'énergie et de sa puissance de traitement toujours croissante. Aujourd'hui, le fabricant ARM Holdings [Noae] est un leader dans ce segment de marché, et ses processeurs Advanced RISC Machine (ARM) sont utilisés par les principaux fabricants d'appareils commerciaux tels que Samsung, Sony Ericsson, Apple ou Nokia, entre autres.

Ses principales caractéristiques sont décrites ci-dessous :

Ordinateur RISC (Reduced Instruction Set Computer). Contrairement aux processeurs basés sur la philosophie de conception CISC (Complex Instruction Set Computing), l'architecture RISC recherche une conception CPU qui exécute une série d'instructions réduites, de taille fixe et réduisant le nombre d'accès mémoire. Ce type d'architecture permet de simplifier la conception interne du processeur en réduisant le nombre de transistors qui l'intègrent. Cela réduit la consommation et la dissipation d'énergie.

Consommation et dissipation de chaleur. En raison de la faible consommation d'énergie de ce type de plates-formes, la plupart d'entre elles ne nécessitent pas de systèmes actifs de dissipation de la chaleur, et la dissipation passive au moyen d'éléments métalliques conducteurs est suffisante. Il en résulte une réduction de la consommation totale d'énergie, ainsi qu'une réduction de l'espace nécessaire à l'intérieur de l'appareil et du bruit produit par les ventilateurs. La section suivante présente la plate-forme Microzed, utilisée lors de la phase de développement logiciel de cette thèse de doctorat, qui intègre un processeur ARM multi-cœur.

1.2.1.2 L'architecture NEON

Roque2015 ; **Roque2015** ; **bossen_common_2013** L'architecture NEON est un SIMD pour la série ARM cortex-A qui a un très long registre configuré en 64 bit /128 bit. Cette technologie a été proposée pour accélérer l'algorithme de traitement du signal, elle fonctionne

décodeur HEVC car il offre les avantages d'une mise sur le marché plus rapide que le codeur HEVC. Dans [Bos+12], Brossen et al. Ont présenté une implémentation du processeur CPU x86 du décodeur, notamment Intel i7-2920XM qui appliquait un logiciel d'optimisation basé sur la technologie SIMD (single-instruction-multiple-data), qui est le parallélisme au niveau des données pour atteindre un temps réel en haute résolution comme en Full HD (1920x1080 pixels). Engelhardt et al in [Eng+14], ont publié une implémentation entièrement matérielle sur FPGA avec la technologie 65 nm, ils peuvent être décodés 30 images par seconde (fps) pour une résolution FULL HD utilisant une configuration principale à faible retard. Leur dispositif fonctionne à une fréquence d'horloge de 110 MHz et il faut 74K de modules logiques adaptatifs (ALMs). Souza et al [DSRS14] donnent une implémentation sur une plate-forme hétérogène basée sur CPU et Graphics Processing Unit (GPU). Ils ont utilisé OpenCL, qui est une API pour le calcul parallèle. Ils travaillent sur le logiciel de test Model software HM.11.0 pour exécuter deux blocs de déquantisation et de transformation inverse sur GPU et d'autres blocs sur CPU. Leurs facteurs d'accélération sont 18,9 et 16,5. Pescador et al [Pes+13] ont présenté un premier modèle de test d'implémentation HM du décodeur HEVC sur un TMS320DM6437 ancienne version DSP . Le résultat surligné montre un traitement de fréquence d'images jusqu'à 4.0 images/seconde pour une résolution de 832x480 pixels lorsqu'ils appliquaient une configuration à faible retard. Dans cette section, une implémentation totalement logicielle du décodeur logiciel HM15.0 avec toutes les configurations sur une évolution récente de la carte TSM32C66878 DSP est présentée avec les trois configurations possibles toutes intra, low-delay et random-access. Nous obtenons une bonne performance pour différentes résolutions de séquences par rapport à d'autres résultats de l'état de l'art en n'utilisant qu'un seul noyau fonctionnant à 1 GHz de fréquence d'horloge et sans utiliser les deux optimisations Enhanced Direct Memory Access (EDMA) et multicores. Cependant, le C6678 est un processeur VLIW (Very-Long Instruction Word), en fait le parallélisme par la technique de pipeline (VLIW) est exploité afin de tirer profit du parallélisme d'instruction, qui est le Parallélisme d'instruction (ILP) qui souhaite améliorer les performances d'exécution des instructions sur une opération par cycle horaire. Par exemple, l'exécution simultanée de plusieurs opérations permet de gagner du temps et d'obtenir de meilleures performances. Cependant, l'étude de la complexité de cette implémentation sur un cœur TMS320C6678 DSP conduit à augmenter le nombre de trames codées par seconde ou en temps réel pour la haute résolution pense à son implémentation en DSP multi- cœur (huit cœurs dans notre appareil) d'une part, et d'autre part, pour exécuter un décodeur HEVC en utilisant un accès direct amélioré (EDMA) pour réaliser une quantité énorme de données sans utiliser de processeur, il peut en effet être utilisé pour exécuter efficacement et flexible instructions, dont des données en mémoire [Ali+12].

1.7 L'état de l'art des implémentations hardware

De nombreux travaux de la recherche ont été publiés dans la littérature afin de proposer une architecture matérielle efficace pour le bloc de la prédiction intra du standard HEVC [Abe+16],[MXC16],[AB16],[Eng+14],[JMC12],[Kha+13],[HTC14],[KAH14],[KAM14], cependant, plusieurs travaux des implémentations matérielles ont proposé des conceptions hardware pour la prédiction intra, qui sont implémentées sur une technologie basée sur la

FPGA en [Abe+16], [Abe+16],[MXC16],[AB16],[JMC12],[Kha+13], ou sur une puce d'ASIC comme en [HTC14], ou à la fois sur les deux ASIC et FPGA en [KAH14], [KAM14]. Certaines de ces conceptions sont proposées pour supporter tous les modes de décision et toutes les tailles de blocs PUs.

Dans [Abe+16], l'implémentation matérielle du décodeur HEVC incluant la prédiction intra sur FPGA est présentée, il peut être utilisé pour décoder en temps réel en résolution 4K avec 30 images par seconde (fps). Les auteurs ont utilisé l'optimisation disponible sur FPGA sur les étapes de pipeline, par conséquent, ils utilisent un seul cycle de traitement de pixel de référence de référence dans la prédiction intra et d'autres techniques d'optimisations pour la transformation et les deux filtres dans le décodage HEVC, de sorte que conduire à un haut débit.

Dans [MXC16], de multiples techniques sont proposées afin de rendre l'accélérateur matériel développé pour la prédiction intra-prévision du fonctionnement du HEVC en pipeline complet. Les trois techniques adoptées sont une nouvelle structure mémoire pour les échantillons de séquences de la référence, un ordre de balayage dépendant du mode et une méthode inverse pour l'extension des échantillons de référence, qui permet de traiter 4 pixels par un seul cycle. Donc, le débit de cette architecture peut supporter le Quad Full HD (3840 × 2160) à 30 images/s. L'accélérateur peut également calculer tous les modes d'intra prédiction.

Dans [AB16]. Un matériel de prédiction intra est présenté pour le traitement de tous les modes, il est basé sur deux techniques pour réduire la complexité du calcul et pour accélérer le traitement. Le premier est adopté pour les modes angulaires et il s'appelle Élément de traitement pour les modes angulaires et le second est appelé Élément de traitement pour le mode planaire, puis le dessin est structuré en trajets répétés afin de calculer en parallèle. L'architecture peut donc traiter 1080p@100 ou 4K@24 en temps réel. Dans [Eng+14], les accélérateurs matériels pour chaque bloc de taille de 4×4 à 32×32 et un design sont présentés avec ces résultats de synthèse. Le matériel est proposé en langage VHDL et implémenté sur le Xilinx Virtex-6 qui est fabriqué au nœud technologique 40 nm. La conception prend 170K LUTs et 110K registres sans aucun bloc de mémoire à partir des ressources disponibles. Après la conception de synthèse, la fréquence maximale jusqu'à 219 MHz et est capable de calculer 24 images par seconde pour une résolution 4K.

Jiang et al [Kha+13] ont développé leur travail en pipeline et l'ont implémenté sur un appareil Xilinx Virtex 5. Il a consommé 69k LUTs, 64 DSPs internes et 148k bit mémoires internes type BRAM. Kahn et al. Dans [Kha+13] une conception en pipeline qui contient les éléments de traitement parallèle (PE) supportant toutes les tailles de PU et les modes de décision a été publiée, elle atteint 213 MHz et peut fonctionner en temps réel avec des contraintes atteignant 120 fps pour la résolution 1080p et 30 fps pour la résolution 4k. Dans le document [HTC14], une architecture VLSI est présentée pour la prédiction intra dans la norme HEVC. La conception a adopté trois techniques pour réduire systématiquement la complexité. Le premier est la mémoire hiérarchique intégrée, au lieu des portes comme registres utilisés pour stocker les échantillons voisins qui peuvent augmenter le débit. Le second est un schéma de programmation mode-adaptatif qui conduit à fournir au moins 2 échantillons/cycles. Les trois réduisent les multiplicateurs en les partageant dans l'architecture proposée. L'utilisation

parfaitement avec le pipeline et le parallélisme fonctionnant sur plusieurs données en utilisant la même instruction. Cette architecture a deux types de registres, Q et D. Seize registres Q de 128 bits, appelés Q0-Q15. La Figure B.1 donne un exemple du registre Q qui est tronqué en trois vecteurs composés de 4×32, 8×16 et 16×8 bits. Le nombre de registres D est de 32 sur 64 bits [22, p. 33].

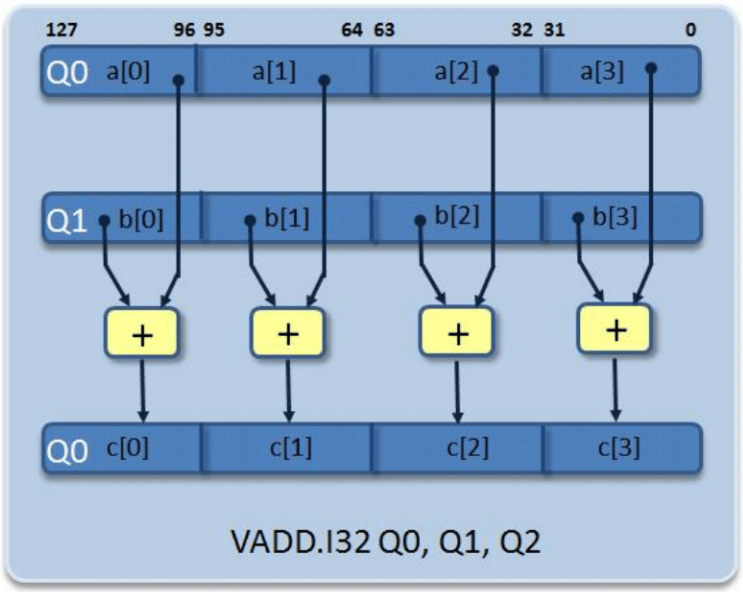


FIGURE 1.1 – Exemple de registre Q (128 bits) qui tronque en 32 bits.

La Figure B.2 montre le schéma fonctionnel de cette plate-forme. Il a le même ensemble d'instructions que l'ARMv7, et il est connu par la faible consommation d'énergie, ce qui fait, il adoptera pour divers appareils comme les smartphones, tablette et télévision numérique. L'architecture NEON est un SIMD ARM cortex A8 qui possède un très long registre configuré en 64 bit /128 bit. Cette technologie a proposé d'accélérer l'algorithme de traitement du signal, elle fonctionne parfaitement avec ce pipeline et le parallélisme fonctionnant sur plusieurs données en utilisant la même instruction.

1.2.1.3 Les plates-formes utilisées

Pour ce travail, nous avons utilisé la carte Beaglebone équipée par la Sitara AM3358 SOC [Noac]. L'AM3358 fait partie de la famille AM335x. Le processeur principal est un ARM cortex A8 cadencé à 1 GHz d'une part, et d'autre part, nous avons utilisé le plate-forme Zynq de Xilinx qu'est embarque un processeur ARM cortex a9 cadencé à un fréquence 667Mhz. Lors de ces expériences, deux plates-formes sont utilisées, la première est basée sur le cortex ARM A8 cadencé à 1 GHz et elle est intégrée sur la Sitara AM3358 SOC [Noac], qui est incluse sur la famille AM335x de Texas Instruments. La deuxième est embarque le cortex A9, elle a 667 MHz de fréquence d'horloge également intégrée sur Microzed Zynq-7000 SOC inclus proposé par Xilinx. Généralement, la plate-forme Microzed est utilisée pour des applications qui ont

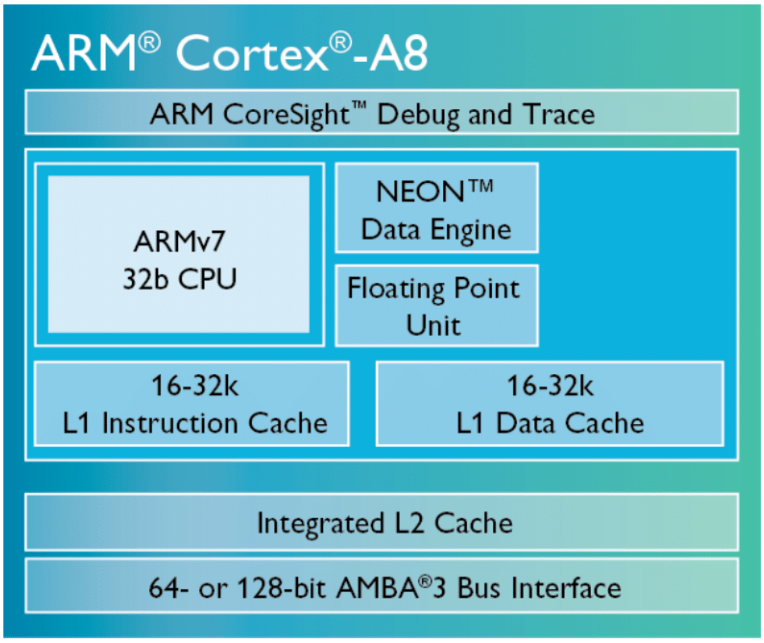


FIGURE 1.2 – L'architecture interne du Cortex-A8.

un traitement de propriété intellectuelle (IP) matériel lors de l'utilisation du FPGA et peut être contrôlée par le système logiciel implémenté sur les processeurs ARM. cependant, une comparaison des caractéristiques les plus importantes entre ces deux processeurs Cortex-A8 et Cortex-A9 est présentée dans le Tableau 4.7. Néanmoins, la série ARM cortex-A comme A8 et A9 avait le même ensemble d'instructions que l'architecture ARMv7. Par conséquent, elle présente une faible consommation d'énergie par rapport aux autres architectures, il est donc adopté pour différents dispositifs comme les smart-phones, tablettes et la télévision numérique haute définition.

L'architecture SIMD est nommée NEON pour le cas de la série ARM cortex-A. Le NEON dispose d'un registre très long allant jusqu'à 128 bits, l'objet de cette technologie est l'accélération de l'algorithme pour le traitement du signal. Pour cela, le traitement s'effectue de manière transparente avec ce pipeline et le parallélisme fonctionnant à plusieurs dates en utilisant la même instruction.

1.2.2 Architecture du processeur DSP

1.2.2.1 Introduction

Les dispositifs dits DSP sont des microprocesseurs à architecture très spécialisée dans les applications de traitement du signal en temps réel, généralement caractérisés par l'application répétitive d'opérations de type multiplication et accumulation (MAC, Multiplier et Accumuler) sur de grandes quantités de données, parfois en mode streaming (séquence continue des données via un buffer).

TABLE 1.1 – comparaison des caractéristiques principales deux processeurs ARM Cortex-A8 et Cortex-A9

	Cortex-A8	**Cortex-A9**
Architecture version	ARMv7	ARMv7
Multicore	No	Yes (1,2,4)
Pipeline type	In-order, dual Issue, superscalar	Out-of-order, variable length, superscalar
Pipeline stages	13	08-11
MMU	Yes	Yes
Core to L1 interface	64 bit ; NEON 128 bit	64 bit ; NEON 128 bit
Floating-point unit	VFP lite and NEON SIMD	VFPv3 and NEON SIMD
DMIPS/MHz	2.01	2.5

Ces processeurs se composent principalement de blocs interconnectés par un bus de données interne. De plus, ils intègrent des structures de mémoire et des DMA (Direct Memory Access) qui permettent l'exécution de la même instruction sur plusieurs données, ce qui est communément appelé SIMD (Single Instruction Multiple Data). La Figure B.3 montre l'architecture de base d'un DSP avec un bus de données 32 bits et des accumulateurs 58 bits.

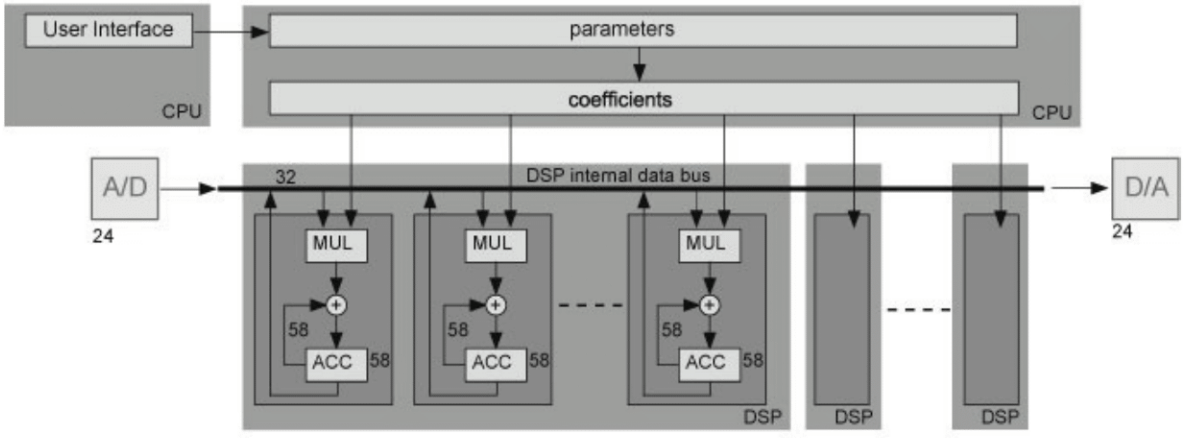


FIGURE 1.3 – Architecture de base d'un DSP [Sha98].

Les DSP sont couramment utilisés pour effectuer tous les types de traitement (encodage/décodage, cryptage, compression, etc.) des signaux audio et/ou vidéo. Les langages les plus utilisés pour sa programmation sont le langage assembleur (principalement utilisé lorsqu'une partie spécifique du code doit être optimisée), spécifique à chaque périphérique, et C/C++ (dans la plupart des cas).

Les processeurs de signaux numériques (DSP) sont des microprocesseurs spécialisés avec une architecture optimisée pour une faible consommation d'énergie. Les DSP sont utilisés pour le traitement audio/vidéo, les appareils portables et mobiles ainsi que de nombreux systèmes embarqués. Ce type de microprocesseurs à une approche différente pour la conception matérielle et nécessite une optimisation logicielle irrégulière pour obtenir de bonnes performances de mise à l'échelle. Par conséquent, il est essentiel pour le développeur de logiciels d'optimiser les algorithmes à la main dans les routines de code d'assemblage. De nos jours, les processeurs n'ont pas besoin des efforts supplémentaire pour optimiser le code, car les compilateurs avancés et sophistiqués gèrent l'optimisation pour eux. Mais les processeurs modernes ont besoin d'utiliser plus de puces ou de micro schémas qui consomment plus d'énergie.

Le DSP offre une solution à moindre coût avec de meilleures performances et une faible latence (pas de prise en charge de la mémoire virtuelle, la mémoire virtuelle nécessite plus de temps pour passer d'un processus à l'autre, ce qui entraîne une latence accrue). L'utilisation du DSP dans le calcul haute performance a été proposée par la communauté scientifique en raison de sa puissance efficiente et de ses bonnes performances informatiques. Dans cette thèse, nous testons l'implémentation d'encodeur/décodeur de la vidéo HEVC qu'a des algorithmes qui se basent sur le traitement matricielles fin de mesurer la performance de l'exécution et la complexité du codeur. Le DSP c6678 comprend à la fois la conception des puces et les processus de fabrication. A titre d'exemple, la consommation d'énergie sur TI DSP C6678 pour la double précision 60 Flops nécessite 10 watts sur puce, 50 watts à bord, ce qui peut également donner 160 Flops pour la simple précision [Noam].

1.2.2.2 Architecture de processeur de signal numérique (DSP) c6678

Le TMS320C6678 est un DSP Multi-cœur Fixe/Flottent point se référant à l'architecture multi-cœur KeyStone de TI, il contient huit c66x CorePac DSPs avec 1 GHz pour chaque cœur, le C66x corePac a huit unités fonctionnelles, deux fichiers de registres, et deux chemins de données. Les fichiers de registre sont limités par les noms A et B qui ont chacun trente-deux (de A0 à A31 pour 1 et de B0 à B31 pour B) avec un large 32-bit, les 64 registres sont utilisés pour les données ou les pointeurs d'adresse de données. Les huit unités fonctionnelles sont .M1, L1, D1, S1, .M2, L2, D2 et. S2, les premières unités .M dédiées à toutes les opérations de multiplication. Les deuxièmes et troisièmes unités. S etc. exécutent un ensemble général de fonctions arithmétiques, logiques et de branchement. Les quatrièmes unités. D utilisées pour le chargement et le stockage des données. Les chemins de données nommés A et B. En outre, les registres et les unités fonctionnelles présentés dans les chemins de données, l'unité centrale de traitement (CPU) contient l'unité de récupération de programme, l'unité de répartition des instructions 16/32 bits, l'emballage des instructions avancées, l'unité de décodage des instructions, les registres de commande Test logique, émulation et DMA interne (IDMA) pour les transferts entre mémoires internes illustrés à la Figure 2.1.

Le DSP peut être chargé et stocker le mot double avec 32 bits entre les caches de mémoire et les fichiers de registre de but général, cette fonction est réalisée par les unités fonctionnelles D, les fichiers de registre sont limités dans deux qui ont nommé A et B que chacun d'eux a

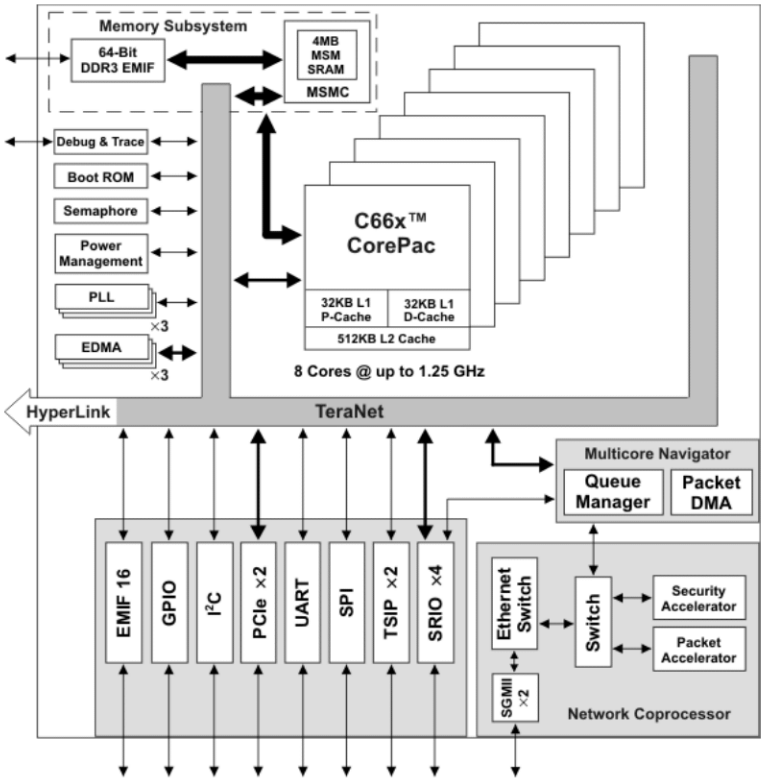


FIGURE 1.4 – schéma de l'architecture interne du DSP C6678.

trente-deux (de A0 à A31 pour 1 et de B0 à B31 pour B) avec 32 bits larges, tous 64 registres sont employés pour des données ou des indicateurs d'adresses de données.

- Mémoire système Le C6678 a huit cœurs, pour chaque cœur, la mémoire système sur puce constituée par les niveaux L1 et L2 de la mémoire statique à accès aléatoire (SRAM), la mémoire cache L1 a 32 Ko de programme et 32 Ko de données, la mémoire cache L2 a 512 Ko, pour tous les cœurs, la mémoire cache L2 et L3 SRAM avec 4096 Ko peut également être utilisée Multi-cœur mémoire partagée, le dispositif interface à 512 Mo 64 bits DDR3 avec 1333 M qui est partagé mémoire. L'appareil réalisé par le contrôleur EDMA3 (Enhanced Direct Memory Access) avec trois contrôleurs de canaux qui permettaient de transférer des données entre deux mémoires, par exemple les données de transfert entre la mémoire externe DDR3 et la mémoire cache interne.

1.2.2.3 La technologie VLIW pour le DSP C6678

D'autres architectures de jeux d'instructions ont été conçues pour augmenter le débit du CPU. Les architectures RISC (reduced instruction set computer) le font en simplifiant les instructions, ce qui permet une vitesse d'horloge plus élevée. Les architectures VLIW (très gros mot d'instruction) permettent d'exécuter plusieurs instructions indépendantes en parallèle, dans un effort pour maximiser l'utilisation de toutes les parties du CPU. Le C6678 peut

exécuter des instructions avec SIMD qui est un parallélisme de niveau de données (DLP) et il est utilisé uniquement avec des opérations en virgule flottante, les instructions fonctionnent sur des vecteurs avec 128 bits, par exemple, il est capable d'exécuter quatre instructions 32 bits simultanément note que l'exécution de jusqu'à 8 instructions par cycle.

Le c6678 a également VLIW qui est le niveau d'instruction Parallélisée (ILP). Cette technique de parallélisme repose sur huit unités fonctionnelles qui sont .M1, .L1, .D1, .S1, .M2, .L2, .D2, et .S2. Chacune de ces fonctions peut être exécutée une instruction pour chaque cycle d'horloge, le .D1&2 utilisant pour les opérations de transfert entre les registres généraux (A et B).

1.2.2.4 La plateforme EVM DSP TMDXEVM6678L

La Figure 2.2 montre le EVM DSP TMDXEVM6678L , le huit sous-systèmes de base TMS320C66x DSP (CorePacs C66x Core) fonctionnant chacun à 1 GHz, architecture VLIW (mot d'instruction très long), mémoire sur puce de 8,5 Mo (Mo), données multiples (SIMD) à instruction unique pour 64000 MIPS de performance. Chaque noyau C66x intègre une grande quantité de mémoire sur puce. En plus des 32 Ko de mémoire cache de programme et de données L1, il y a 512 Ko de mémoire interne par noyau qui peut être configuré comme mémoire vive ou cache mappée. La plate-forme intègre également 4 Mo de mémoire partagée multi-cœur. Pour prendre en charge les applications qui nécessitent une grande quantité de mémoire, telles que les applications vidéo ultra HD, le TMS320C6678L inclut une mémoire externe DDR3-1333 de 512 Mo. La plate-forme EVM est livrée avec le kit de développement logiciel multi-cœur de TI (MCSDK) pour SYS/BIOS Real Time Operating System (RTOS). Ce périphérique inclut également 64 Mo de mémoire NAND Flash pour faire fonctionner le noyau Linux-c6x. Les performances sont également améliorées par un contrôleur EDMA capable de gérer les transferts de mémoire indépendamment du CPU. TMS320C6678 prend en charge une multitude d'interfaces standard haut débit, notamment RapidIO pour les communications DSP-to-DSP, PCI Express Gen2 et Gigabit Ethernet pour les réseaux IP (Internet Protocol). Il inclut également I2C, UART et le port d'interface série de télécommunication (TSIP).

1.3 Technologies d'implémentation matériel

Les applications des FPGAs et des ASICs sont similaires. Bien que les FPGAs soient plus lents et consomment plus d'énergie, ils ont le grand avantage d'être reprogrammables. Cette caractéristique rend le flux de conception très flexible, minimisant ainsi les coûts de développement et d'acquisition pour de petites quantités d'appareils. Les FPGA peuvent implémenter n'importe quel circuit, des simples fonctions de portes logiques aux systèmes combinés complexes, grâce à leur logique programmable, à condition que vous ayez les ressources nécessaires. Son utilisation se répand de plus en plus pour des applications qui nécessitent des systèmes temps réel avec un haut degré de parallélisme, comme l'automobile, le traitement d'images

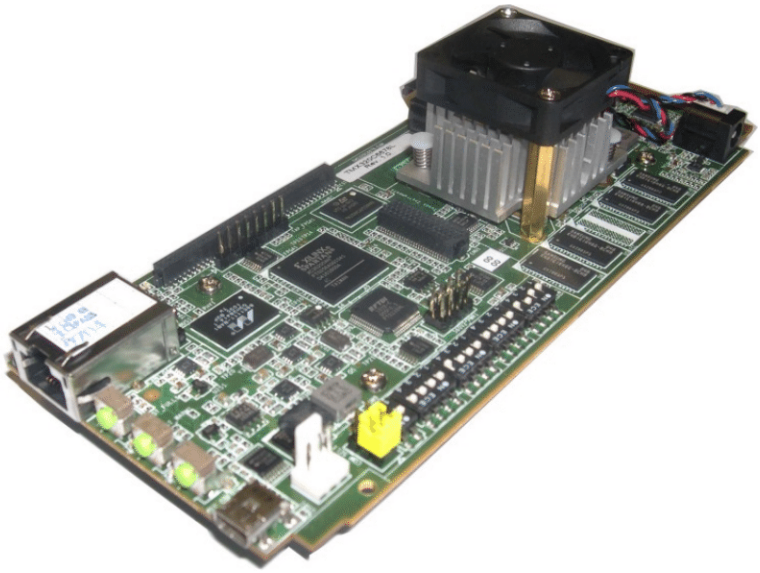


FIGURE 1.5 – L'EVM TMS320C6678 utilisant dans ce travail et l'architecture interne de TMS320C6678.[Noal]

et la vision par ordinateur, la médecine, la bio-informatique, les systèmes aérospatiaux, le prototypage des ASIC, etc.

Dans ce travail, nous allons implémenter une partie de l'encodeur vidéo HEVC détailler en bas dans le chapitre V dans un FPGA. Plus précisément, nous allons mettre en œuvre l'extraction des caractéristiques, qui bien que n'étant pas un algorithme en soi est une partie fondamentale et doit toujours se faire indépendamment de l'algorithme qui est finalement choisi. De plus, comme nous l'avons vu, l'implémentation d'algorithmes dans les FPGA est largement décrite dans la littérature, alors que l'extraction des caractéristiques est une question moins abordée.

D'autre part, le calcul des images intégrales et celui des différentes caractéristiques peuvent être effectués simultanément, car chacune d'entre elles est indépendante de la suivante. Il est donc très intéressant de l'implémenter dans un FPGA dont la structure est intrinsèquement parallèle. Cela réduirait considérablement le temps d'exécution de cette phase, ce qui permettrait par la suite l'utilisation d'algorithmes plus précis, mais qui pourrait nécessiter un temps de prédiction plus long.

Nous pouvons définir un FPGA comme un dispositif qui possède un grand nombre d'éléments matériels reconfigurables. Cela permet de modéliser une grande variété de circuits matériels, adaptés aux besoins spécifiques de chaque application, dans un seul appareil. Les FPGAs sont nés dans les années 90 comme une évolution d'autres dispositifs reconfigurables moins complexes appelés PLDs (Programmable Logic Device) ou CPLDs (Complex Programmable Logic Device), qui ont été construits à l'aide de portes logiques et de multiplexeurs.

1.3.1 Technologie de la FPGA

Nous pouvons définir un FPGA comme un dispositif qui possède un grand nombre d'éléments matériels reconfigurables. Cela permet de modéliser une grande variété de circuits matériels, adaptés aux besoins spécifiques de chaque application, dans un seul appareil. Les FPGAs sont nés dans les années 90 (figure 2.3)comme une évolution d'autres dispositifs reconfigurables moins complexes appelés PLDs (Programmable Logic Device) ou CPLDs (Complex Programmable Logic Device), qui ont été construits à l'aide de portes logiques et de multiplexeurs.

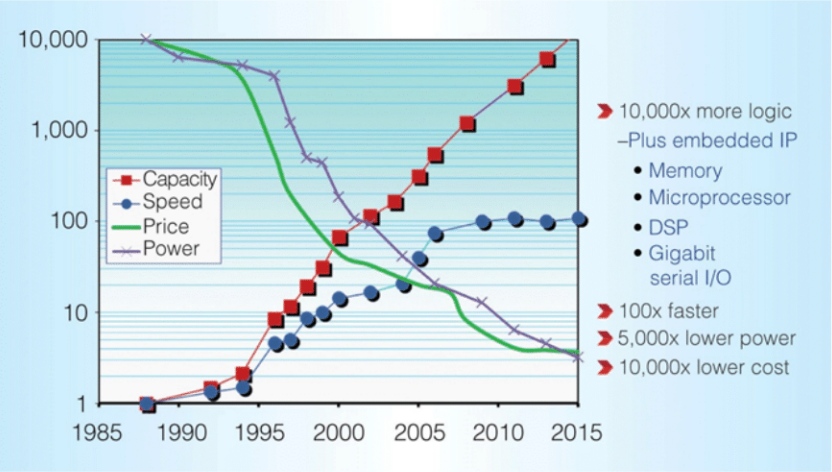


FIGURE 1.6 – Evolution du prix, de la consommation, du coût et de la vitesse du FPGA 2011.

Aujourd'hui, ce sont des appareils beaucoup plus complexes, avec une forte densité d'intégration, à l'intérieur desquels vous pouvez trouver beaucoup de ressources, telles que (Figure 2.4) :

- Blocs logiques : ce sont les principaux éléments reconfigurables du FPGA et permettent la matérialisation des fonctions logiques. Leur structure peut changer en fonction du fabricant, mais elles sont généralement composées de petits blocs fonctionnels configurables appelés LUTs (Look up Tables), qui permettent d'implémenter des fonctions logiques, ainsi que de fonctionner comme mémoire distribuée ; des registres ou bascules pour enregistrer des données ; et des ports d'entrée/sortie. De plus, ces blocs peuvent être combinés pour implémenter des fonctions plus complexes ou des banques de mémoire. - Blocs de RAM : qui peuvent avoir des capacités différentes, des configurations différentes de leurs ports d'entrée/sortie et être organisés en banques de tailles différentes.

- Blocs DSP : sont des structures matérielles spécialement optimisées pour effectuer des opérations telles que des ajouts et des multiplications, qui sont largement utilisées dans le traitement du signal. De cette façon, en n'utilisant pas des éléments reconfigurables mais des éléments fixes, on évite l'utilisation de blocs logiques pour mettre en œuvre des fonctions simples, en améliorant la fréquence de traitement et la consommation énergétique. La grande majorité des FPGA d'aujourd'hui intègrent des DSP capables de supporter des opérations en

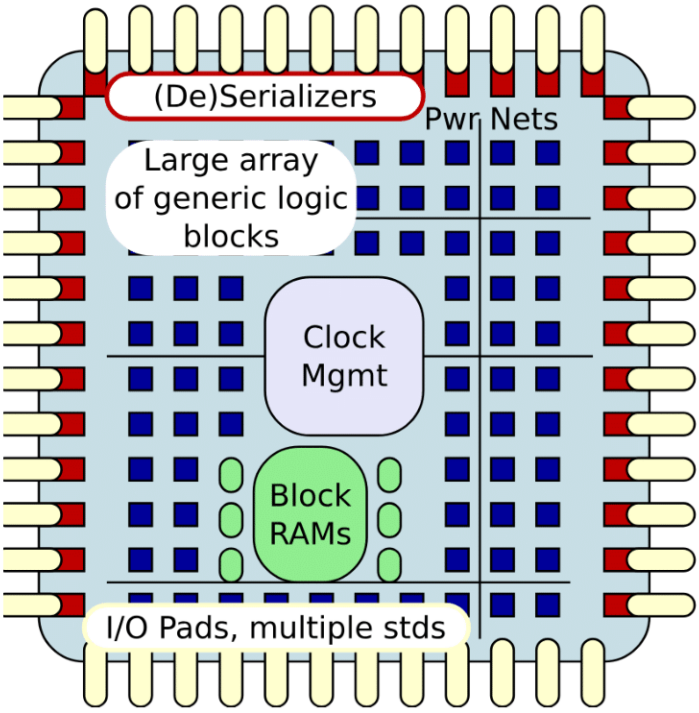


FIGURE 1.7 – exemple d'un architecture interne FPGA avec des éléments de base.

virgule flottante.

- PLL (Phase-Looked Loops) et DLL (Delay-Looked Loops) : circuits en boucle fermée dont la fonction principale est de trouver et modifier la phase de l'horloge de référence du système,

- Blocs d'entrée/sortie : Ce sont des ressources de connexion associées aux broches de sortie qui permettent différentes configurations. Grâce à eux, il est possible de programmer la vitesse de rotation, le retard des broches d'entrée/sortie, les résistances d'arrachement, etc.

- Les ressources d'interconnexion : elles peuvent présenter des topologies différentes et leur fonction principale est l'interconnexion de tous les éléments du FPGA. On peut distinguer les ressources d'interconnexion locale, qui relient des ensembles contigus d'unités logiques, et les ressources d'interconnexion globale, qui permettent l'interconnexion de celles-ci et de toutes les autres ressources FPGA [Sha98].

Tous ces éléments se retrouvent, totalement en fonction de la gamme à laquelle appartient le FPGA. En outre, ces dispositifs peuvent disposer de ressources supplémentaires telles que des émetteurs-récepteurs pour l'envoi de données à large bande passante, des DSP de précision variable et, ces dernières années, ont également tendance à intégrer un microprocesseur, ayant ainsi un SoC (on Chip) hétérogène, comme on le verra dans la section suivant consacrée à la plate-forme de développement Zynq.

En général, la vitesse du FPGA est comparativement plus lente que celle de l'ASIC spécifique à l'application. De même, il n'est pas aussi efficace de mettre en œuvre une logique gourmande en ressources puisque le FPGA consomme plus d'énergie. Néanmoins, le FPGA surpasse les performances lorsqu'il se comporte comme un bloc logique, comme une interface ou une logique de collage, un contrôleur de bus, un processeur de protocole simple. Heureusement, l'amélioration technologique permet d'étendre le domaine d'application aux régions ASIC qui ont des exigences rigoureuses en matière de calcul, de débit, de fréquence et de puissance. Le FPGA est également largement utilisé comme plate-forme de vérification pour la conception ASIC à l'étape préliminaire, pour héberger le prototype avant d'être transplanté en fabrication ASIC. Pour différents fournisseurs de FPGA, les structures varient dans la configuration des ressources, mais partagent trois caractéristiques communes, à savoir le bloc logique configurable (CLB), le bloc entrée-sortie (IOB) et le canal de routage interne (Interconnect).

1.3.2 Accélérateur matériel (Vivado)

La Figure 2.5 illustre cette évolution au cours des 15 dernières années. A la fin des années 1990, les FPGA étaient pour la plupart des matrices de portes avec de la RAM intégrée. Au cours de la dernière décennie, les FPGA ont ajouté davantage de fonctions IP durcies, y compris les émetteurs-récepteurs, les DSP, les interfaces de mémoire DDR, les PLL et la technologie de puissance programmable (PPT). Au cours de cette décennie, les fournisseurs de FPGA ont ajouté des sous-couches de codage physique (PCS) telles que les MAC Ethernet, les DSP à virgule flottante et les processeurs ARM. Ces puces qui contiennent des processeurs ARM embarqués sont appelées "SoC FPGAs"

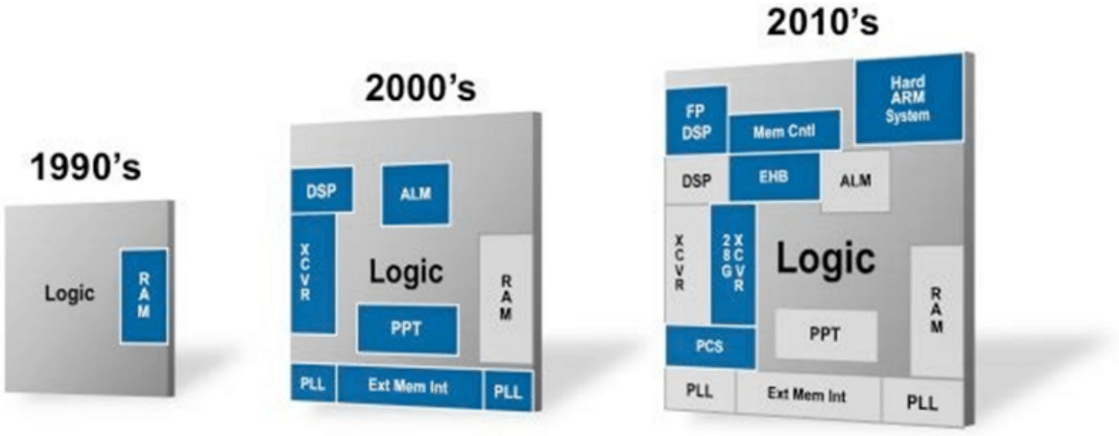


FIGURE 1.8 – FPGA Evolution. Les FPGA ont ajouté de l'IP durci depuis 2000 et sont maintenant plus SoC que les FPGA [Noai].

La conception ou "programmation" d'un FPGA se fait à l'aide de langages de description matérielle, les plus utilisés étant VHDL et Verilog. D'autre part, les principaux fabricants de ces appareils fournissent des outils qui simplifient le processus de conception. De plus, il existe également des IP (Intellectual Property Core), des composants prédéfinis décrits dans

ces langages, qui simplifient l'implémentation des algorithmes et des fonctions. Les logiciels fournis par la société Xilinx® seront utilisés pour réaliser ce travail, plus particulièrement les programmes Vivado HLS, Vivado et Vivado SDK.

1.3.3 Technologie de la ASIC

Un ASIC (*Application-Specific Integrated Circuit*) est circuit intégré propre à une application, il est un type de circuit intégré spécialement conçu pour une application ou un usage spécifique. Comparé à un dispositif logique programmable ou à un circuit intégré logique standard, un ASIC peut améliorer la vitesse parce qu'il est spécifiquement conçu pour faire une chose et il fait bien cette chose. Il peut aussi être plus petit et consommer moins d'électricité. L'inconvénient de ce circuit est qu'il peut être plus coûteux à concevoir et à fabriquer, en particulier si seulement quelques unités sont nécessaires.

1.3.4 Langage de programmation matériel

Le Hardware Design Langage (VHDL / Verilog) est l'approche dominante pour la conception et la vérification de systèmes numériques. Les cellules programmables peuvent être utilisées pour mettre en œuvre de simples logiques (portes de base) à des algorithmes complexes (traitement vidéo, algorithmes cryptographiques, etc.), ou à de très grands projets, comme un système de contrôle technique.

1.4 Technologies Co-désigne logiciel matériel

Ces dernières années, les avancées technologiques, réalisées par la société Xilinx®, ont réduit les différences entre les modèles de programmation des processeurs et des FPGA. Tout comme il existe des compilateurs pour C et d'autres langages de haut niveau pour les différentes architectures de processeurs, le compilateur Vivado Hihg-Level Synthesis (HLS) Xilinx® possède les mêmes fonctionnalités, traduisant les programmes écrits en C/C+ en langage descriptif matériel, avec lequel les FPGAs fonctionnent. Ce chapitre sera divisé en trois sections, qui coïncident avec les étapes qui ont été réalisées pour la mise en œuvre en FPGA de l'extraction des caractéristiques : - Conception d'un accélérateur matériel pour l'extraction des fonctionnalités, avec Vivado HLS. - Construction du schéma fonctionnel du système, qui comprendra l'accélérateur matériel, l'ensemble des mémoires nécessaires au stockage des données et un processeur, avec Vivado IP Integrator. - Programmation du processeur, pour le contrôle du système complet. De plus, dans cette section, ils effectueront des mesures du temps d'exécution, pour sa comparaison ultérieure avec un CPU.

Dans le chapitre V, un noyau IP ou accélérateur matériel a été conçu pour en extraire les caractéristiques. Un noyau IP est un bloc de logique ou de données qui peut être réutilisé lors de la conception d'un FPGA. Pour la conception de notre accélérateur, nous allons utiliser le

compilateur Vivado HLS, décrit ci-dessus.

Bien que ce logiciel VIVADO facilite et accélère énormément la tâche de conception, puisqu'il évite d'avoir à connaître le langage descriptif du matériel et ses particularités, il est essentiel de prendre en compte certaines considérations. Voici des difficultés rencontrées dans l'exécution de ce travail. Tout d'abord, il est important de garder à l'esprit que l'algorithme à développer doit être adapté à l'architecture du matériel à utiliser. Dans notre cas, par exemple, nous avons dû adapter le prototype réalisé en C/C++, qui utilisait des matrices comme type de données. Dans un FPGA, cela n'est pas possible, car les mémoires ont des adresses consécutives et fixes qui doivent être configurées au moment de la programmation, et non pendant l'exécution du programme, comme c'est le cas avec le logiciel. Comme le compilateur HLS n'est pas conçu pour utiliser ce type de données, et donc pas pour effectuer une transformation cohérente, l'image d'entrée a dû être sérialisée, adaptant toutes les fonctions pour travailler avec des données de type vectoriel, au lieu de tableaux. De cette façon, il est possible de maintenir la structure du reste du programme, sans varier la dynamique des boucles, ni les conditionnels

1.4.1 L'architecture du processeur Zynq

Dans la Figure 2.6 on trouve le processeur ARM, le protocole AXI permet le contrôle et la lecture-écriture de multiples entrées et sorties à usage général via des interfaces multiplexées utilisant des registres 32 bits. Par conséquent, il permet un grand flux d'informations sans utiliser un nombre excessif de broches. De plus, étant donné qu'il est standardisé et qu'il reçoit beaucoup de support dans les outils de développement Xilinx, ce sera une excellente option pour servir de lien commun entre ce projet et les améliorations futures.

La caractéristique principale de Zynq est qu'il combine un processeur ARM Cortex-A9 double cœur [Cro+14] avec un réseau de portes programmables sur site (FPGA) traditionnel. Bien que les processeurs dédiés aient été couplés avec des FPGA auparavant, cela n'a jamais été tout à fait la même proposition. À Zynq, l'ARM Cortex-A9 est un processeur de qualité applicative, capable d'exécuter des systèmes d'exploitation complets tels que Linux, tandis que la logique programmable est basée sur l'architecture FPGA Xilinx série 7 [Noan], [Noao]. L'architecture est complétée par des interfaces AXI standard, qui fournissent des connexions à large bande passante et à faible latence entre les deux parties du dispositif [Noaj]. Cela signifie que le processeur et la logique peuvent être utilisés chacun pour ce qu'ils font le mieux, sans la surcharge d'interfaçage entre deux dispositifs physiquement séparés. Entretemps, les avantages découlant de la simplification du système à une seule puce comprennent la réduction de la taille physique et du coût global.

L'un des principaux points architecturaux à noter avec les périphériques Zynq-7000 est que la méthode de démarrage est limitée à une source unique - le démarrage du périphérique doit être piloté par le processeur. Lorsque l'appareil est mis sous tension ou réinitialisé, le premier noyau du PS démarre à partir de mémoires externes avant de configurer le PL [Cro+14]. En limitant la méthode de démarrage à une source unique, il garantit qu'il n'existe aucun moyen

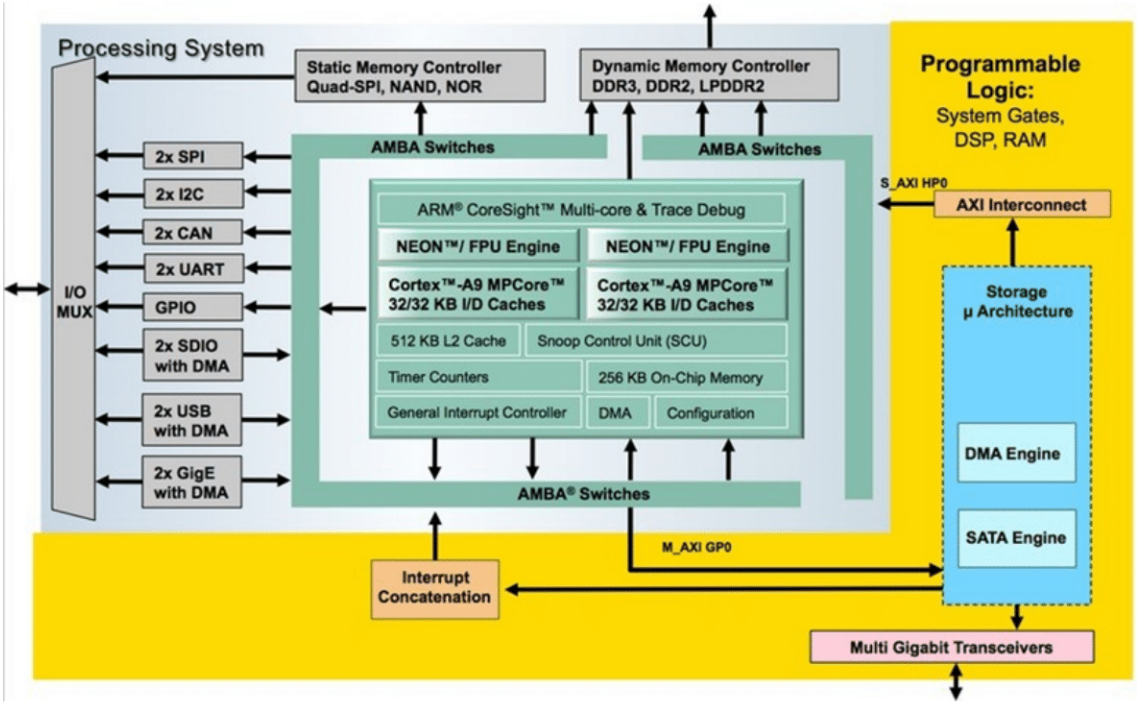


FIGURE 1.9 – diagramme interne de processeur Zynq avec tous les éléments de l'architecture [Noaj].

manuel de charger un logiciel malveillant après la configuration du PL, ni aucun moyen de charger une image malveillante sur le PL une fois que le processeur a été initialisé.

Un certain nombre de fonctionnalités ont été incorporées dans les périphériques Zynq-7000 qui facilitent le démarrage sécurisé. L'une de ces caractéristiques est la ROM de démarrage, qui a été conçue pour gérer diverses formes de sécurité. L'authentification asymétrique et symétrique du FSBL (First- Stage Boot Loader), du U-Boot, du flux binaire PL et du logiciel utilisateur (OS et applications utilisateur) est supportée.

1.4.1.1 Connexion PL vers PS : conception du pilote

Une fois le matériel développé et le système d'exploitation installé, nous devons mettre en œuvre un système qui permette à ces deux composants de communiquer entre eux, le pilote.

Pour comprendre le fonctionnement et la nécessité pour un pilote d'établir cette communication, il est nécessaire de distinguer deux concepts : mémoire virtuelle et mémoire physique. Deux termes qui font référence au système que Linux utilise pour gérer la mémoire du système d'exploitation et son en interaction avec lui. Comme expliqué dans la partie matérielle, le VDMA extrait les informations directement de la mémoire. Cette mémoire à laquelle le VDMA est connecté est la mémoire physique connue habituellement par le nom RAM.

La mémoire virtuelle est l'abstraction de cette mémoire physique pour utilisation par le

système d'exploitation. Cette distinction est faite pour permettre au système d'exploitation d'avoir un meilleur contrôle sur la mémoire. Par exemple, nous pouvons avoir la moitié de la mémoire physique comme mémoire virtuelle utilisée par un utilisateur particulier, si une partie de cette mémoire n'est pas utilisée dans une période de temps donnée, le système d'exploitation peut sélectionner ce segment de mémoire virtuelle et l'enregistrer sur le disque dur, rendant l'accès plus lent quand vous avez besoin des informations à nouveau mais libérant une grande quantité de mémoire. Cet exemple, connu sous le nom de pagination est l'un des principaux avantages de ce système de division de mémoire et permet au système d'exploitation, payant un prix en vitesse d'accès, de simuler une mémoire beaucoup plus grande que les possédés physiques. Cette distinction est importante pour le développement du projet car l'utilisateur n'a pas accès à la mémoire physique, le système d'exploitation bloquera toute tentative d'accès.

1.4.2 Langage de programmation logiciel C/C++

Le principal inconvénient dans le choix d'utiliser une plate-forme matérielle, par opposition à son approche purement logicielle, est généralement le modèle de programmation. Les algorithmes logiciels sont généralement écrits en C/C++ ou dans d'autres langages de haut niveau, qui sont indépendants des plates-formes sur lesquelles ils doivent être exécutés, du moins d'une manière générale. Ces langages permettent des itérations rapides, des améliorations incrémentales et rendent le code portable.

1.5 Le traitement en temps réel du codage vidéo

1.5.1 Notions de temps réel

Un système en temps réel est un système dans lequel la réaction à un événement doit se produire dans un délai précis, sinon le système est considéré comme ayant échoué [KG06]. Du point de vue du traitement des images ou vidéos, un système d'imagerie en temps réel est un système qui capture régulièrement des images, les analyse pour obtenir des données, puis les utilise pour contrôler certaines activités. Tout le traitement doit être effectué dans un délai prédéfini (souvent, mais pas toujours, la cadence de capture d'image). Les exemples de systèmes de traitement d'images en temps réel abondent. Dans les systèmes de vision industrielle, l'algorithme de traitement d'image est utilisé pour l'inspection ou le contrôle de processus. Certains systèmes de vision robotique utilisent la vision pour la planification du trajet ou pour contrôler le robot d'une manière ou d'une autre où le temps est critique. Le contrôle autonome du véhicule nécessite une vision ou une autre forme de détection pour la navigation du véhicule ou la prévention des collisions dans un environnement dynamique. Dans les systèmes de transmission vidéo, les images successives doivent être transmises et affichées dans le bon ordre et avec une gigue minimale pour éviter une perte de qualité de la vidéo obtenue.

Les systèmes en temps réel sont classés en deux catégories : le temps réel dur et le temps réel souple. Un système temps réel dur est un système dans lequel le système complet est considéré comme ayant échoué si la sortie n'est pas produite dans les délais requis. Un exemple est l'utilisation de la vision pour classer des articles sur une bande transporteuse. La décision de classement doit être prise avant que l'article n'atteigne le point d'actionnement, où l'article est dirigé d'une manière ou d'une autre selon le résultat du classement. Si le résultat n'est pas disponible à ce moment-là, le système a échoué. D'autre part, un système temps réel souple est un système dans lequel le système complet ne tombe pas en panne si l'échéance n'est pas respectée, mais la performance se détériore. La transmission vidéo via Internet en est un exemple. Si l'image suivante est retardée ou ne peut pas être décodée à temps, la qualité de la vidéo résultante se détériore. Un tel système est souple en temps réel, car même si l'échéance n'a pas été respectée, un résultat peut encore être produit, et le système complet n'a pas échoué.

Du point de vue du traitement du signal, le temps réel peut signifier que le traitement d'un échantillon doit être terminé avant l'arrivée du prochain échantillon [KG06]. Pour le traitement vidéo, cela signifie que le traitement total par pixel doit être terminé dans un délai d'échantillonnage de pixels. Bien sûr, tout le traitement d'un pixel ne peut pas être terminé avant l'arrivée du pixel suivant, car de nombreuses opérations de traitement d'image nécessitent des données de plusieurs pixels pour chaque pixel de sortie. Cependant, cela limite le taux de traitement moyen, y compris les frais généraux associés au stockage temporaire des valeurs de pixels qui seront utilisées plus tard [KG06]. L'exigence que l'algorithme de traitement de la vidéo entière ait un temps d'exécution limité implique que chaque opération doit également avoir un temps d'exécution limité. Cette caractéristique exclut certaines classes d'algorithmes de niveau d'opération du traitement en temps réel. En particulier, les opérations basées sur des algorithmes itératifs ou récursifs ne peuvent être utilisées que si elles peuvent converger de manière satisfaisante dans un nombre prédéfini d'itérations, pour l'ensemble des entrées qui peuvent être rencontrées.

1.5.2 Traitement embarquée du codage vidéo

Un système embarqué est un système informatique qui est intégré dans un produit ou un composant électronique. Par conséquent, un système embarqué est généralement conçu pour exécuter une tâche spécifique ou un petit nombre de tâches spécifiques, souvent avec des contraintes en temps réel. Un exemple évident d'un système de traitement d'image intégré est un appareil photo numérique. Les fonctions d'imagerie comprennent le contrôle de l'exposition et de la mise au point, l'affichage d'un aperçu et la gestion de la compression et de la décompression des images sous forme de la vidéo [Cat05].

La compression vidéo pour les hautes résolutions est également utile pour les caméras intelligentes, où la caméra capture non seulement l'image, mais la traite et compresse également pour en extraire ou envoie les informations requises par l'application. Les systèmes de surveillance "intelligents", l'inspection ou le contrôle industriel, la vision robotique, le diffusion vidéo, etc. sont des exemples d'applications utiles. L'une des exigences de nombreux

systèmes embarqués est qu'ils doivent être de petite taille et légers. Un grand nombre d'entre eux épuisent leurs batteries et doivent donc fonctionner avec une faible puissance. Même ceux qui ne sont pas alimentés par batterie ont généralement une puissance limitée.

1.5.3 Systèmes matériels de traitement de la vidéo

Dans la section précédente, il n'a pas été précisé si les processeurs qui constituaient les systèmes embarqués étaient un ensemble de processeurs série standard ou du matériel dédié mettant en œuvre les fonctions embarquées. Théoriquement, cela fait peu de différence, tant que le système électronique embarqué est mis en œuvre efficacement et que les frais généraux de communication sont maintenus bas.

Les premiers systèmes informatiques de série étaient trop lents pour traiter des vidéos de toute taille ou de tout volume. La structure régulière des images a conduit à la conception de systèmes matériels pour exploiter plusieurs processeurs embarqués, car les systèmes matériels embarqués inclus homogènes ou/et hétérogène processeur (multi-DSPs, multi-ARMs ou ARM plus DSP comme l'exemple du puce OMAP de Texas Instruments [Cum03]. Cependant, des premiers systèmes étaient basés sur un élément de traitement (ET) relativement simple à chaque pixel [Ung58], où les autours ont proposé en 1958 un système basé sur une grille carrée à 4 connexions pour le traitement des images binaires. Bien que le matériel de chaque ET soit relativement simple, il a démontré que de nombreuses opérations de traitement d'image de bas niveau pouvaient être effectuées. Bien que ce premier système n'ait pas été construit [DL00], il a inspiré d'autres architectures similaires.

Pour la capture et l'affichage de vidéos ou d'images, même les ordinateurs en série nécessitaient un système matériel pour s'interfacer avec la caméra ou l'écran. Ces cartes d'acquisition d'images sont interfacées avec le système de l'ordinateur hôte soit par accès direct à la mémoire (DMA), soit par mémoire partagée qui a été mappée dans l'espace d'adresse de l'hôte. Une carte d'acquisition d'image de base se compose d'un convertisseur A/N, d'une banque de la mémoire capable de contenir au moins une image et d'un convertisseur numérique/analogique pour l'affichage des images. Nombre d'entre eux permettaient d'appliquer des opérations ponctuelles de base à l'aide de tables de consultation sur l'image capturée ou affichée. Ces simples pipelines ont été étendu à des opérations plus sophistiquées et à des systèmes de traitement d'images entièrement pipelines, dont le plus remarquable. Le pipeline est moins adapté au traitement d'images de haut niveau ; des systèmes hybrides ont été développés, composés de matériel en pipeline pour le prétraitement et d'un ensemble de processeurs série pour le traitement de haut niveau, comme dans le système dans [CV94] proposé en 1994.

Les premiers systèmes matériels ont été mis en œuvre avec des circuits intégrés à petite et moyenne échelle. L'avènement du VLSI (intégration à très grande échelle) a permis de réduire le coût du matériel et d'augmenter considérablement la vitesse et les performances des systèmes résultants. Cela a conduit à l'utilisation d'un plus large éventail d'architectures matérielles et à la mise en œuvre d'opérations de traitement d'images [HM85]. Il est intéressant de noter qu'à mesure que la technologie s'est améliorée, la taille des éléments de

réduction a fait augmenter le coût de production des masques, ce qui a fait augmenter considérablement la proportion des coûts uniques par rapport au coût des dispositifs individuels. Par conséquent, l'économie de la production de VLSI signifie que seule une gamme limitée de circuits (suffisamment générale pour justifier de grandes séries de production) a été largement utilisée commercialement. Les systèmes de traitement d'images basés sur le matériel sont très rapides, mais leur plus gros problème est leur relative rigidité. Une fois configurés, ils s'acquittent très bien de leur tâche, mais il est difficile, voire impossible, de reconfigurer ces systèmes si la nature de la tâche change. Dans les années 1980, l'introduction de la technologie FPGA (field programmable gate array) a ouvert de nouvelles possibilités pour la conception logique numérique. Les FPGA combinent la nature parallèle inhérente au matériel et la flexibilité du logiciel en ce sens que leurs fonctionnalités peuvent être reprogrammées ou reconfigurées. Les premiers FPGA étaient assez petits par rapport au nombre équivalent de portes, de sorte qu'ils avaient tendance à être utilisés principalement pour fournir une logique d'interconnexion et d'interface flexible (parfois appelée logique de la colle), et pour la mise en œuvre de contrôleurs basés sur des machines à états finis. Plusieurs FPGA ont été nécessaires pour mettre en œuvre tout système de traitement d'image significatif. Au fur et à mesure que les FPGA devenaient de plus en plus puissants, des systèmes hybrides ont été développés où les FPGA étaient utilisés comme coprocesseurs ou accélérateurs de vision dans un ordinateur hôte [AA95]. Les FPGA modernes disposent désormais de ressources suffisantes pour permettre la mise en œuvre d'applications complètes sur un FPGA unique, ce qui en fait un choix idéal pour les systèmes de vision temps réel embarqués.

1.6 L'état de l'art des implémentations software

Il existe plusieurs travaux connexes dans la littérature sur l'accélération des codeurs HEVC utilisant l'optimisation à une seule instruction multiple de données sur un processeur ARM et Intel x86 [Ahn+14],[Jun+16],[Bos+12]. Bossen et al.[Bos+12] étant donné l'analyse de la complexité de la norme de codeur HEVC sur les deux plates-formes basée sur les processeurs ARM cortex-A9 et Intel x86 utilisant les instructions de l'ensemble SIMD, cette étude inclut un profilage et une comparaison avec la norme précédente H.264/AVC de faible complexité. Ahn et al dans [Ahn+14] ont proposé une optimisation du codeur HEVC en utilisant SIMD et le parallélisme de niveau de coupe sur le multi-cœur sur la plate-forme x86, ils se sont concentrés sur certains modules comme les calculs de coûts RD et les transformations pour des tailles de blocs variables, et ils ont ajouté un traitement efficace avec un cadre basé sur un traitement afin de réduire le nombre de filtrage redondant. Ces techniques permettent une optimisation logicielle appliquée à HM9.0 jusqu'à 33,56% du gain de temps moyen. Jun et al In [Jun+16] a optimisé le codeur HEVC à l'aide du SIMD et du traitement parallèle sur la plate-forme X86. Ils ont un rapport d'accélération jusqu'à 192 fois supérieur à celui des temps d'exécution complets, mais ils ont une perte de codage acceptable. Leurs tests sont appliqués en séquence Ultra HD et peuvent être compressés à raison d'environ 1,4 images par seconde, en référence à la version logicielle HM 14.0.

La plupart des implémentations sur les processeurs embarqués se concentrent sur le déco-

deur HEVC [Bos+12], [Yan+12] car il est moins complexe que le codeur où il y a quelques travaux sur l'état de l'art et certains travaux portent sur l'optimisation logicielle de ses différents modules comme la prédiction intra [Bos+12], l'estimation de mouvement [Yan+12]. Yong-Jo & Tae-Jin [Ahn+14] présentent une optimisation jusqu'à 31 fois supérieure à celle du logiciel original HM.9.0 qui a été implémenté sur l'unité centrale de traitement (CPU) x86 en utilisant des opérations SIMD (single instruction multiple data) et un parallélisme au niveau des données. Sung-Fang & Cheng-Han [TTC14] présente une implémentation sur le FPGA (Field Programmable Gate Array) qui est basé sur une puce de test HM4.0 et HEVC capable de coder la résolution 8K (8192x4320) avec un débit jusqu'à 1062Mpixels/s, cette puce utilisant la technologie TSMC 28nm HPM et utilisant 8350K gate count. Gabriel et José [CM+15] conçoivent une architecture parallèle hétérogène composée d'une unité de traitement graphique (GPU) et d'une unité centrale multi-cœur, il a tiré parti de ces techniques basées sur le traitement parallèle de front d'onde (WPP), celles-ci peuvent être accélérées en moyenne à 3,41 fois. L'inconvénient de ce travail est qu'il consomme plus d'énergie en utilisant le CPU et le GPU par rapport à notre DSP basse consommation.

Dans [SJL16] les auteurs mis en œuvre l'estimation de mouvement du HEVC dans la dernière série de DSP TMS320C6678. Ils ont présenté un algorithme basé sur le modèle de chaîne de Markov (MCMDP) afin d'améliorer la mémoire d'accès. De plus, ils ont proposé une nouvelle méthode de calcul de la prédiction des vecteurs de mouvement (MVP). Ils ne peuvent pas atteindre le traitement en temps réel, mais, ils ont considérablement amélioré les performances du ME du HEVC sur ce DSP multi-cœur.

Dans [Ju+16], les auteurs ont proposé une puce pour la norme décodeur/codeur vidéo HEVC sur la technologie de traitement CMOS. Les auteurs utilisent plusieurs technologies afin d'accélérer le traitement. Premièrement, ils utilisent des algorithmes rapides au lieu de l'algorithme original du HEVC afin de réduire la complexité, en particulier pour les modes Intra et inter. Ensuite, ils ont proposé une architecture matérielle basée sur une architecture entièrement parallèle par élément de traitement (PE) qui est adoptée pour le filtre SAO IP et l'estimation du mouvement qui peut réduire l'accès à la mémoire jusqu'à 78,4%. En outre, ils ont appliqué une unité de gestion de mémoire (MMU) à la fois dans le pool SRAM (LOOP) et dans la traduction de bus de données (DBT) qui permet de réduire la bande passante externe des pixels voisins jusqu'à 71,6%. Leur puce réalise le traitement en temps réel pour la résolution 4096x2160 (4K@30 fps).

Dans [LCS08]. L'implémentation partielle de l'ancienne norme H.264/AVC est implémentée en DSP en utilisant à la fois des optimisations de mots d'instruction très longs (VLIW) et des données multiples à instruction unique (SIMD). Ils ont appliqué cette implémentation à la méthode détectée pour l'algorithme le plus critique Sam of absolute difference (SAD) qui est basée à la fois sur le parallélisme de niveau d'instruction optimal (ILP) et le parallélisme de niveau de données (DLP), en plus, ils utilisent un algorithme de décision rapide pour optimiser la distorsion de débit (RDO) afin de réduire le nombre de la comparaison et la mise à jour causée par les différences de vecteur de mouvement pendant le traitement. De plus, ils incluent le logiciel de la technique de pipeline pour optimiser spatialement l'accès mémoire. Actuellement, la majorité des implémentations embarquées se concentrent sur le

de cette technique peut réduire la surface de consommation, mais ces trois techniques ont plus de puissance grâce à la mémoire SRAM utilisée et la simulation après l'implémentation montre la consommation d'énergie aller jusqu'à 2,11 mW. La conception est synthétisée à 200 MHz en utilisant la technologie ASIC 40nm et peut supporter la résolution 3840×2160@30 en temps réel.

Dans [KAH14], on utilise l'égalité des pixels et la similarité des pixels pour réduire la quantité du calcul effectué par l'algorithme de prédiction intra dans le HEVC. Par conséquent, la consommation d'énergie est également réduite. Les conceptions matérielles n'ont exécuté que des modes de prédiction angulaire 4×4 et 8×8 et ont été implémentées sur les deux FPGA Xilinx Virtex 6 avec 150 MHz de fréquence, et l'implémentation sur la technologie ASIC se base sur la bibliothèque de cellules standard 90nm de Synopsys, ils peuvent donc traiter 30 images par seconde en résolution Full HD (1920 × 1080) pour les deux cas.

Dans [KAM14], ils exploitent une propriété symétrique des équations intra-perdition des directions horizontale et verticale, qui conduit à une étude limitée dans une direction et qui change l'ordre des échantillons de référence sélectionnés dans l'autre direction. La conception matérielle est implémentée à la fois sur le Virtex 6 FPGA de Xilinx qui fonctionne à 234MHz et sur la technologie de processus CMOS TSMC 180 nm Certaines architectures matérielles sont développées dans la littérature pour traiter le DCT entier appliqué dans le block de transformation du standard HEVC [Cha+16]. En outre, la mise en œuvre de ces propositions se concentre en 2D-DCT ou est limitée à 1D-DCT. Certains de ces travaux n'ont pas traité toutes les tailles DCT entières. Alors que d'autres œuvres calculent des blocs de toutes tailles. Chaque conception d'entre eux ont été caractérisés par les mêmes techniques pour accélérer l'algorithme DCT. Ruhan et al in [Ram+14] ont présenté une architecture matérielle rapide pour une transformation DCT inverse à 4 points pour le standard HEVC. Leur conception s'est concentrée sur l'occurrence de cas spéciaux où ils ont une augmentation du taux de BD-Rate de 0,4%, mais elle a réduit d'environ 87,5% le nombre de calculs DCT 1-D dans le processus IDCT 2-D. Ils ont implémenté leur accélérateur matériel dans le Cyclone V FPGA sous forme de dispositif ALTERA, il prend 152 ALMs, 96 registres avec une fréquence de fonctionnement jusqu'à 78.31 MHz, il atteint 100 images par seconde pour 4k (3840 × 2160 pixels) de résolution. Darji et Makwana [DM15] ont proposé une architecture VLSI pour le DCT 1-D utilisant un DCT sans multiplicateur. L'algorithme CSD-CSE a été utilisé pour réduire les opérations d'addition. Ils ont atteint une réduction de 36% et 28% de la surface après synthèses dans le FPGA et l'ASIC respectivement par rapport aux travaux précédents utilisant des conditions similaires. Son architecture de conception jusqu'au bloc 32×32 qui est synthétisé dans la technologie Spartan3 EXC3S500E FPGA et qui consomme 5785 de tranches avec une fréquence de fonctionnement de 23.727 MHz, alors qu'il a atteint 100 MHz pour la mise en œuvre ASIC Wenjun Zhao et al [64] a présenté une architecture VLSI en pipeline sans multiplicateur pour un DCT bi-dimensionnel entier en 2-D pour toutes les tailles de HEVC. Ils ont utilisé des opérations d'addition et de décalage au lieu de multiplications. La conception répond aux exigences de traitement en temps réel des séquences vidéo 4K et a été synthétisée à la fois dans un FPGA Cyclone IV et dans une technologie ASIC 45nm. Sur le FPGA il y a 5446 éléments logiques (LEs) avec une fréquence de fonctionnement jusqu'à 125 MHz avec un débit de 1707.91 Msamples/s environ pour 32 points. Pour l'ASIC, il faut compter 205,5K portes avec

333 MHz et jusqu'à 4546,56 Msamples/s pour 32 points. Paris Kitsos et al [65] conçus, dans le FPGA Virtex7 de Xilinx, en utilisant VHDL une architecture pour DCT 2D, ils ont remplacé les multiplicateurs traditionnels par l'Arithmétique Distribuée pour effectuer la multiplication et proposé deux architectures distinctes en utilisant plusieurs ROMs pour augmenter les performances. La première conception utilisant deux ROMs nécessite 105 cycles pour traiter une transformée 8 points avec une fréquence de fonctionnement de 338.5 MHz et la seconde avec quatre ROMs nécessite seulement 65 cycles avec une fréquence de 256MHz. Dans un autre ouvrage [66], les auteurs ont proposé un accélérateur matériel pour calculer l'entier inverse DCT pour les normes H.264/AVC et HEVC. Ils ont traité toutes les tailles de blocs et leurs résultats expérimentaux ont été obtenus sur un FPGA Xilinx Virtex-7. Dans les blocs d'inter prédictions, le SAD pour l'algorithme d'estimation du mouvement est l'étape la plus importante et la plus lourd en traitement. Son implémentation matérielle peut être utile pour l'encodeur vidéo. Dans la littérature, de nombreux travaux sont proposés pour soutenir l'architecture SAD pour plusieurs domaines d'application, tels que la vision par ordinateur, comme la détection de mouvement pour le traitement d'images [HHW17], sur le système de détection et de reconnaissance de mouvement basé sur la vidéosurveillance [MBD18], qu'il implémente sur carte embarquée basé sur XC2V1000 FPGA et estimation de mouvement pour la compression vidéo normes [WDB12],[Reh+09],[KL09],[LGI08],[NAN13],[Yua+13],[Med+14] [69]–[75]. Toutes les architectures proposées visent à atteindre un traitement en temps réel pour des séquences à plus haute résolution avec la fréquence de fonctionnement la plus élevée possible et à calculer les blocs d'inter prédiction maximum quelle que soit leur taille. Walter et al [WDB12] ont parlé d'une architecture matérielle avec un haut débit pour un calcul SAD de faible puissance fonctionnant jusqu'au processus d'encodage en temps réel pour une résolution de 720×480 pixels. Leur conception a été développée et synthétisée en utilisant deux technologies CMOS, 180nm et 65nm. Ils ont utilisé deux paramètres pour comparer les résultats obtenus, soit le débit maximal et l'énergie minimale consommée par opération. Ils ont indiqué que la moyenne de la puissance consommée à l'aide de la technologie 65 nm diminue de 53 % par rapport à la technologie 180 nm, compte tenu de la baisse nécessaire de la tension d'alimentation. De plus, la nouvelle technologie 65 nm a atteint la haute fréquence en raison du débit plus élevé. Dans [Reh+09], Rehman et al. Ont proposé une architecture matérielle efficace pour l'algorithme SAD, conçu pour le traitement des images. L'avantage de cette architecture est l'ajout de différence absolue décroissante pour les tailles de blocs de 4×4 pixels. Utilisant un FPGA avec Xilinx XC2V1000, ils ont atteint une fréquence de 133 MHz et utilisé 657 LUTs avec un seul bloc 4×4, et ensuite, ils utilisent le moteur développé 4×4 pour traiter les plus gros blocs. Une autre architecture matérielle dédiée à la SAD a été rapportée par Kalamiros et Lygouras [KL09] pour l'accélération de la vision stéréo, notamment pour le calcul de la méthode de corrélation locale. L'architecture proposée a été implémentée sur le FPGA Cyclone II EP2C35, fonctionnant à 100 MHz, et leur accélérateur SAD proposé prend 23900 éléments logiques pour atteindre 162 images par seconde de résolution VGA (640 × 480) codées sur 8 bits de profondeur.

Zhenyu et al [LGI08] ont conçu deux architectures matérielles pour l'algorithme SAD, la norme H.264/AVC prenant en charge la technique de taille de bloc variable. La première architecture, basée sur le SAD partiel propagé, atteint 231,6 MHz- fréquence de fonctionnement à un coût de 84,1K portes, et la seconde est une architecture basée sur l'arbre SAD

atteindre 204,8 MHz avec 88,5k portes. Les deux conceptions ont été synthétisées sur une technologie CMOS TSMC 180 nm 1P6M CMOS, et les approches au niveau architecture ont été utilisées pour améliorer les performances. De plus, Purnachand et ses collaborateurs [NAN13] ont élaboré une architecture matérielle pour la norme de codage vidéo du HEVC. La synthèse a été effectuée sur un FPGA Xilinx Virtex-5 FPGA et prend 291,27K de comptes de portes et atteint 171,9 MHz comme fréquence de fonctionnement. Leur accélérateur est basé sur un traitement parallèle et conçu pour supporter toutes les tailles de blocs PU, y compris les blocs pour le Partitionnement asymétrique du mouvement (AMP). Un autre accélérateur matériel SAD a été proposé par Xu et al [Yua+13] et synthétisé dans le Xilinx Virtex-6 XC6VLX-550T FPGA. Leur architecture contient deux sous-architectures parallèles pour traiter 1080p avec 30 images par seconde (fps) atteignant le codage vidéo en temps réel. Cependant, l'accélérateur ne peut calculer que 16×16 PU et les sous-blocs correspondants. La synthèse montre une fréquence de fonctionnement de 110 MHz avec 55346 LUTs, 19744 registres et 148kB de BRAM. Plus récemment, Medhat et al [Med+14] ont proposé un accélérateur SAD matériel parallèle pour l'estimation du mouvement, synthétisé sur un FPGA Xilinx Virtix-7 XC7VX550T. Une fréquence de fonctionnement de 458 MHz a été signalée avec 39901 LUTs et 24957 registres. Avec cette architecture, une résolution de 2K avec 30 images/seconde peut être traitée, ce qui est la clé du succès de cette architecture. D'autres solutions personnalisées ont également été publiées telles que le VLSI Pipeline moins de multiplieurs point fixe 8×8 DCT Désigne de rapide pipeline 8×8 des autres blocs [ZAR17]. Les auteurs ont mis en œuvre et comparé le DCT à l'aide de deux architectures, non liées au pipeline et d'un pipeline en deux étapes, et les résultats des simulations montrent que le débit peut être amélioré d'environ 1 Giga pixels/s à 1,8 Giga pixels/s, pour un faible coût d'environ 14% de ressources supplémentaires (c'est-à-dire les registres de pipeline).

Dans ce travail, nous proposons une nouvelle architecture VLSI très efficace pour le mode de décision SAD, mise en œuvre dans le HM du HEVC. Cette architecture peut traiter toutes les tailles de PUs, y compris l'AMP, qui est utilisé pour augmenter l'efficacité du codage des séquences vidéo ayant des zones qui sont plus irrégulières. La nouvelle architecture est conçue pour charger tous les pixels d'entrée des blocs de référence et actuels dans un premier temps, en les rendant disponibles pendant le processus de traitement. Ensuite, la conception proposée réduit fortement les échanges avec la mémoire et les accès mémoire et conduit finalement à un calcul parallèle de tous les PUs simultanément, réduisant le temps de latence.

1.8 Résumé

Cette partie a donnée en revue l'architecture générale du dispositif Zynq et détaillé ses deux composants, le PS et le PL, ainsi que les ressources d'interface entre eux. Il a été démontré que le système de traitement comprenait un processeur ARM Cortex-A9 double cœur avec des extensions associées pour le traitement SIMD et le traitement en virgule flottante, regroupés dans une unité APU avec des ressources mémoire. Les capacités de l'APU et son interface avec le reste du système de traitement ont été soulignées. La structure interne de la fonction interne dans son ensemble et ses liens avec la logique programmable ont également été expliqués.

La structure et les ressources de la section de la logique programmable ont été décrites, y compris la logique du tissu, les ressources de la RAM Block et du DSPs, et les ressources d'interface, et l'importante question de l'interface entre le PL et le PS en utilisant AXI a été examinée. Enfin, une comparaison a été faite entre les appareils de la famille Zynq-7000.

Tout au long de ce chapitre, les supports technologiques programmables sur lesquels les décodeurs vidéo sont développés ont été présentés. Les particularités de chaque architecture ont également été présentées d'un point de vue général, en contrastant leurs avantages et inconvénients respectifs. Actuellement, les implémentations sur des plates-formes DSP et ARM hautes performances sont principalement utilisées à des fins d'essais expérimentaux et fonctionnels, tandis que les CPU de type ARM sont devenues de plus en plus pertinentes avec le développement de systèmes de plus en plus puissants tout en maintenant une consommation énergétique réduite. Dans les deux cas, l'utilisation de processeurs multi-cœurs est pleinement en place et leur planification efficace continue d'être l'un des défis à relever.

Chapitre 2

le système de codage vidéo

Sommaire

2.1 introduction au codage vidéo 35
2.2 Historique des normes d'encodage vidéo 36
2.3 La norme de codage vidéo HEVC 37
2.3.1 Structure de bloc de la norme HEVC 37
2.3.2 Schéma de l'encodeur 39
2.3.3 Schéma de décodeur 40
2.3.4 Les tranches Slices et tiles 40
2.4 Structure du bloc 41
2.4.1 La transformation 42
2.4.2 La prédiction intra 42
2.4.3 La prédiction inter 44
2.4.4 La quantification 45
2.4.5 Les filtres 45
2.4.6 Le codage entropique 46
2.5 Les paramètres de norme HEVC 47
2.5.1 Traitement parallèle par front d'onde (WPP) 47
2.5.2 Configurations temporelles 48
2.6 Résumé 48

2.1 introduction au codage vidéo

L'encodage et le décodage vidéo sont des applications très exigeantes qui nécessitent des supports technologiques avec une grande capacité de calcul. Actuellement, les différentes alternatives technologiques utilisées pour mettre en œuvre ce type d'applications peuvent être classées en trois groupes principaux : celles basées sur des processeurs à usage général, celles basées sur des architectures spécifiques et celles basées sur des processeurs de signaux numériques.

L'évolution constante des normes de codage vidéo et les modifications successives qui y sont apportées rendent le temps disponible pour le développement de nouvelles versions de codeurs et décodeurs de plus en plus court. Le but du codage vidéo est de compresser les flux

vidéo, qui consistent en une séquence d'images qui, à un moment donné, seront transmises ou stockées. Compresser signifie réduire la quantité d'information de sorte que le nombre de bits requis pour représenter cette information soit suffisamment faible pour permettre l'utilisation d'applications vidéo. Par exemple, une vidéo au format Haute Définition (HD) (1920×1080) à une cadence de 25 images par seconde et 8 bits pour représenter chacun des canaux Rouge, Vert et Bleu (RGB), nécessite : (192×x1080)× (25)× (3) × (8)=1.20Gbit/s A travers cet exemple simple, il est évident que le codage vidéo est obligatoire pour le diffusion ou même le stockage de fichiers vidéo : la quantité de débit binaire rapportée est que cet exemple est au-delà des limites des architectures et réseaux informatiques et de services actuels. Selon la qualité cible, il est courant d'avoir des taux de compression allant de 10 à 1000. Pour les fournisseurs de contenu, le fait de pouvoir réduire la taille du contenu qu'ils diffusent leur permet d'augmenter le nombre de contenus qu'ils peuvent stocker, ainsi que le nombre d'abonnés qu'ils peuvent atteindre en utilisant les mêmes ressources de stockage et de capacité réseau. En 2013, les deux tiers du trafic du protocole Internet (IP) étaient dus au streaming vidéo, et cette tendance devrait continuer d'augmenter pour atteindre 84% du trafic IP d'ici 2020. Ces prévisions soulignent la nécessité de poursuivre les recherches sur les nouvelles techniques de codage vidéo. En tant qu'opérateur de réseau, étant également un fournisseur d'accès Internet (FAI) qui fournit des services vidéo tels que le streaming TV en direct ou les services de vidéo à la demande.

Cependant, à partir de 2004, certaines entreprises ont commencé à commercialiser des processeurs de signaux numériques (DSP) de très haute performance, dotés d'une grande capacité de calcul, qui permettent de mettre en œuvre des applications d'encodage et de décodage vidéo sur ceux-ci. De plus, ces processeurs intègrent une série de périphériques (ports vidéo d'entrée et de sortie, convertisseurs numérique-analogique ou accès au réseau local) qui facilitent les interfaces typiques de ce type d'applications, permettant d'implémenter des systèmes complets sur une seule puce SOC *(System On Chip ou SoC).*

2.2 Historique des normes d'encodage vidéo

Depuis l'élaboration de la première norme de codage au début des années 1980 et jusqu'à aujourd'hui, deux organisations internationales ont dirigé le développement et la normalisation de nouveaux systèmes de codage vidéo. D'une part, l'UIT (Union internationale des télécommunications) a développé les normes H.261 et H.263 entre 1984 et 1998. L'ISO (Organisation internationale de normalisation) a élaboré les normes MPEG-1, MPEG-2 et MPEG-4 en 1990, 1995 et 1999 respectivement. Toutefois, en 2003, les deux organisations, l'UIT et l'ISO, ont uni leurs forces pour élaborer conjointement la norme H.264 dans la nomenclature UIT ou MPEG-4 partie dans la nomenclature ISO. Il s'agit de l'une des normes les plus répandues et elle est largement utilisée dans les systèmes de télévision numérique ou de diffusion vidéo aux États-Unis. Enfin, vers 2010, les deux organisations ont commencé à travailler sur une nouvelle norme appelée H.265 (selon l'UIT) ou HEVC *(High Efficiency Video Coding selon l'ISO).* Le processus de normalisation du HEVC s'est achevé en 2012 ; depuis lors, le HEVC est considéré comme le successeur naturel du H.264. La Figure 2.1 résume graphiquement

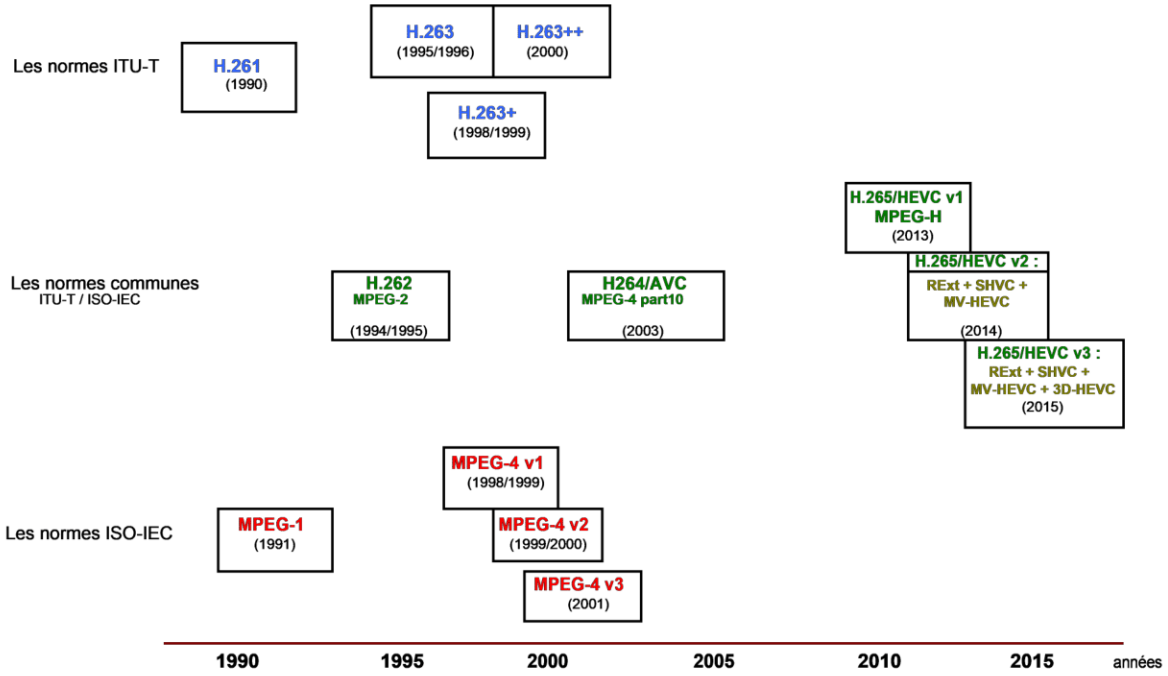


FIGURE 2.1 – Représentation chronologique de l'élaboration des principales normes de codage vidéo.

l'évolution des diverses normes de codage au cours des 30 dernières années par l'UIT et l'ISO.

La section suivante sera consacrée à la présentation en détail de la norme HEVC qui nous concerne dans ce travail, en mettant l'accent sur les détails les plus pertinents pour cette thèse de doctorat. Des informations sur les normes MPEG se trouvent dans [Noag], et des documents sur les différentes normes [Noaf] sont disponibles sur le site web de l'UIT-T.

2.3 La norme de codage vidéo HEVC

2.3.1 Structure de bloc de la norme HEVC

Les outils importants mis en œuvre dans le HEVC sont : premièrement, la nouvelle unité de prédiction de blocs de taille variable (PU) jusqu'à 64×64 qui est proposée pour supporter différentes vidéos haute résolution contrairement au Macrobloc fixe 16×16 existant dans son prédécesseur, le H.264/AVC. Deuxièmement, la prédiction intra, qui est l'un des blocs les plus importants et les plus longs de la norme de codage vidéo du HEVC, utilisée pour la prédiction des pixels afin de former l'unité de prédiction (PU). Dans la nouvelle norme, il peut atteindre une meilleure efficacité que celle définie dans le codage vidéo précédent, puisqu'il dispose de plusieurs modèles de prédiction (jusqu'à 35 modes) et qu'il calcule également le grand bloc PU (jusqu'à 64×64 pixels). Il exploite la redondance spatiale qui n'existe que dans une seule

trame. Tout ça rend son traitement beaucoup plus complexe [Lai+12]. À l'heure actuelle, le HEVC travaille dans une structure de partitionnement de codage par blocs très souple et efficace qui est divisée en quatre niveaux, comme suit : CTU, CU, PU et TU. Tout d'abord, chaque frame est divisée en blocs appelés unités d'arbre de codage (CTU), de différentes tailles : 16×16, 32×32 ou 64×64. Un CTU comprend un bloc d'arbre de codage (CTB) pour Luma et deux CTBs pour le composant Chroma. La CTU est similaire au Macroblock 16×16 fixe existant dans la norme antérieure H.264/AVC. Ensuite, l'unité de codage (CU) est sous CTU et peut être divisée en quatre profondeurs, elle est dimensionnée en carré de 8×8, 16×16, 32×32 et 64×64 pixels selon la résolution du codage des séquences. Ensuite, la CU elle-même peut être divisée en Unité de Prédiction (PU). Ce bloc peut être utilisé pour la prédiction intra et inter blocs avec des tailles allant de 4×4 à 64×64 selon le mode de décision. Enfin, l'unité de transformation (TU) est déterminée à partir du PU et convient à la transformation et à la quantification.

2.3.1.1 Unité de codage CU

La décision de codage ou une zone d'image est prise au niveau de CU (unité de codage). Il y a 3 niveaux de prédiction : intra, inter ou « saute » (skip) prédiction. Une structure de partitionnement PU (prediction unit) a sa racine au niveau CU. La norme HEVC contient des tailles variables de PBs de 4×4 jusqu'à 64×64 échantillons.

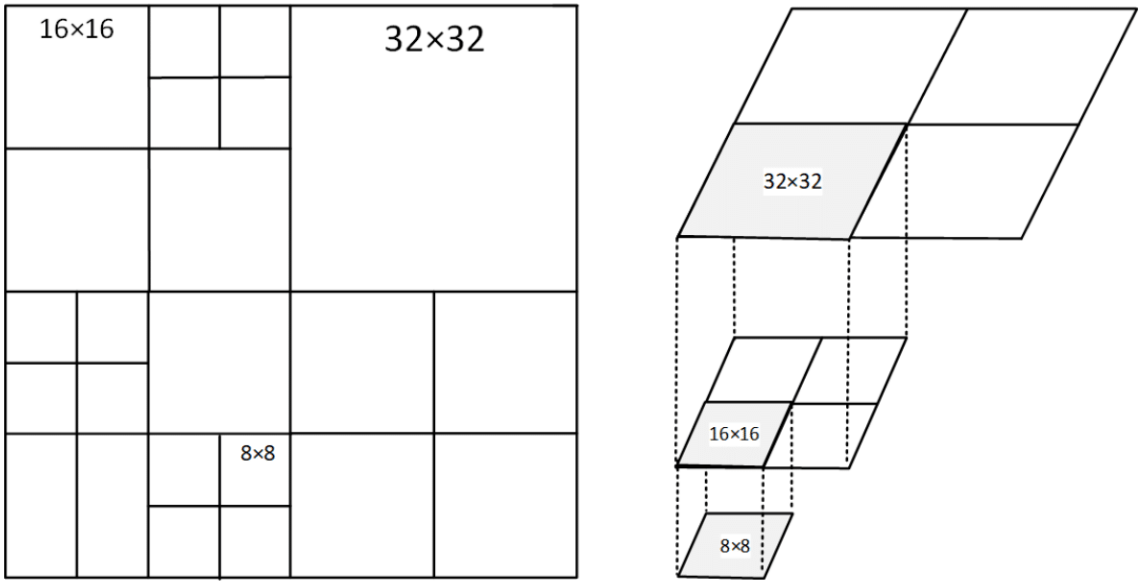


FIGURE 2.2 – exemple d'un CTU devisé en multi CUs.

Une CU non divisée est une unité de prédiction (PU) qui peut être subdivisée en 8 modes différents présentés sur la Figure 2.2, Il existe des modes symétriques (première ligne dans l'image), pour la prédiction intra et inter, et des modes antisymétriques (deuxième ligne dans l'image), uniquement pour la prédiction inter [Wie15].

2.3.1.2 Unité de prédiction PU

Une unité de prédiction (PU) est composée de blocs de prédiction de Luma (et de Chroma) adjacents qui partagent le même processus de prédiction avec l'information de prédiction identique qui se trouve dans le flux binaire. L'arbre de prédiction du HEVC permet de diviser les CB Luma/Chroma en un, deux ou quatre PB de forme rectangulaire. La partition de mouvement symétrique (SMP) et la cloison de mouvement carrée (Square) sont adoptées de l'AVC et étendues pour supporter de plus grandes tailles de CB. Les modes de partition de mouvement asymétrique (AMP) sont introduits dans le HEVC pour les formes irrégulières qui nécessiteraient plus de bits pour être codées avec les modes Square et SMP.

2.3.1.3 Unité codage TU

Les TUs sont employées sur les résidus provenant des blocs de prédiction intra ou inter et sont basées sur l'algorithme DCT et la quantification. La taille des TU varie de 4×4 à 32×32 surfaces carrées. De plus, la TU peux être divisée récursivement en blocs plus petits selon la structure en quadri arbres [Wie15]. La Figure 2.3 montre TUs dans l'unité de codage CU.

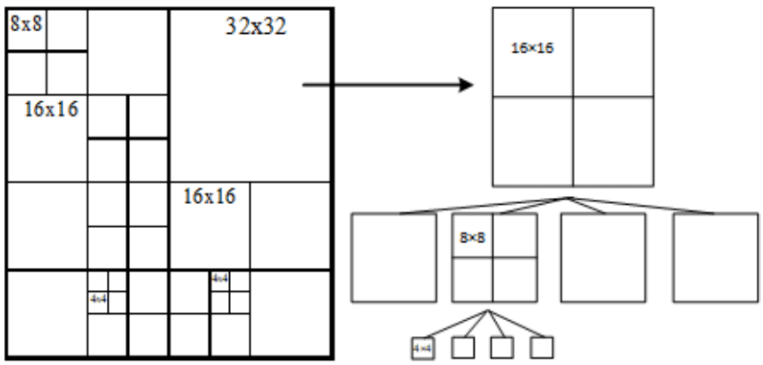


FIGURE 2.3 – Exemple des Tus dans l'unité de codage (CU).

2.3.2 Schéma de l'encodeur

Le schéma fonctionnel de l'encodeur HEVC est illustré à la Figure 2.4. Normalement, il reçoit une trame d'entrée de type YUV à encoder et génère dans ses faces de sortie un fichier

de données nommé bit-Stream. Principalement, il inclut tous les modules nécessaires afin de réduire la redondance de l'information existante sur ces trames et d'assurer une meilleure compression du débit binaire généré.

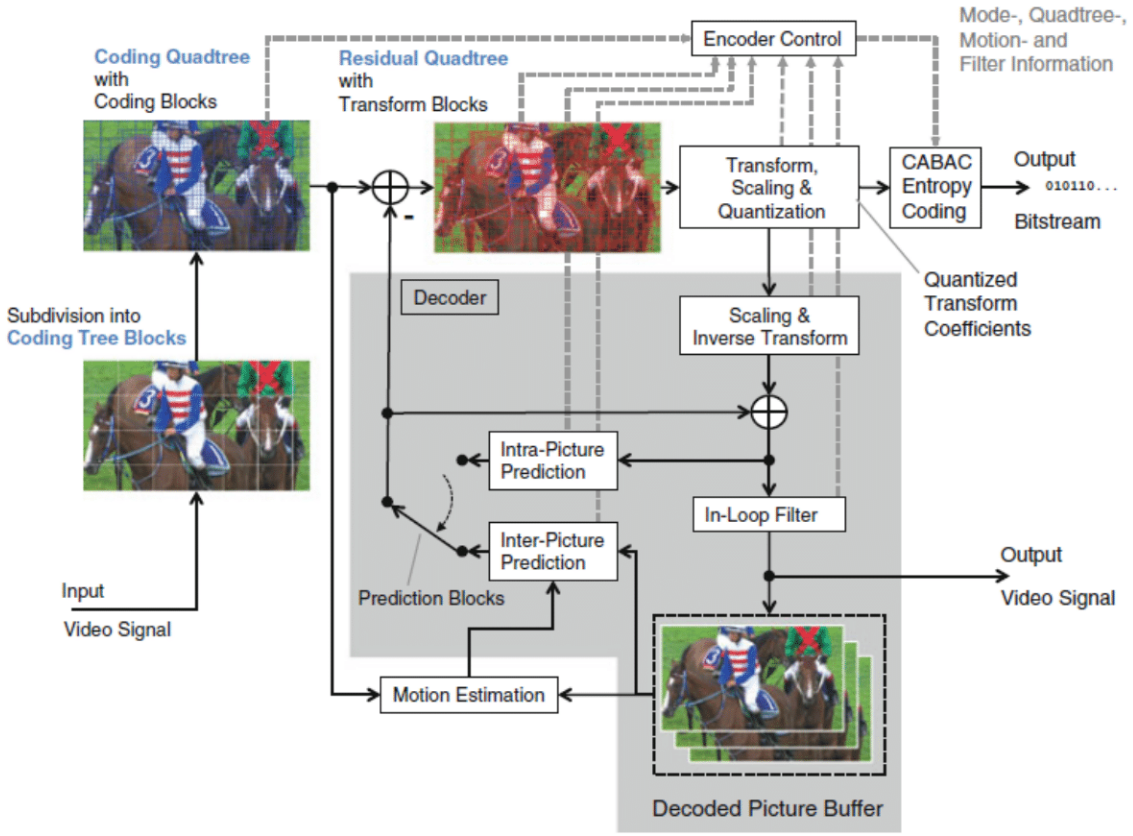


FIGURE 2.4 – Schéma fonctionnel de l'encodeur HEVC avec les modules principaux.

2.3.3 Schéma de décodeur

Le modèle proposé pour le décodeur HEVC est présenté à la Figure 2.5. Il reçoit un flux binaire d'entrée et génère une trame de décodage YUV dans ses côtés de sortie à plusieurs blocs, qui sont selon le processus de l'encodeur HEVC. Dans la Figure 2.5, tous les blocs du décodeur HEVC avec les données de flux binaire en entrée et les trames YUV générées en sortie sont présentés.

2.3.4 Les tranches Slices et tiles

Comme le montre la Figure 2.6, une image peut être divisée en une ou plusieurs tranches de sorte que l'image devienne une collection d'une ou plusieurs tranches. Deux types de segmentations d'images sont définis par la norme HEVC. i. Les CTUs sont des blocs qui permettent le balayage de trame pendant l'encodage. Chaque image peut être divisée en une

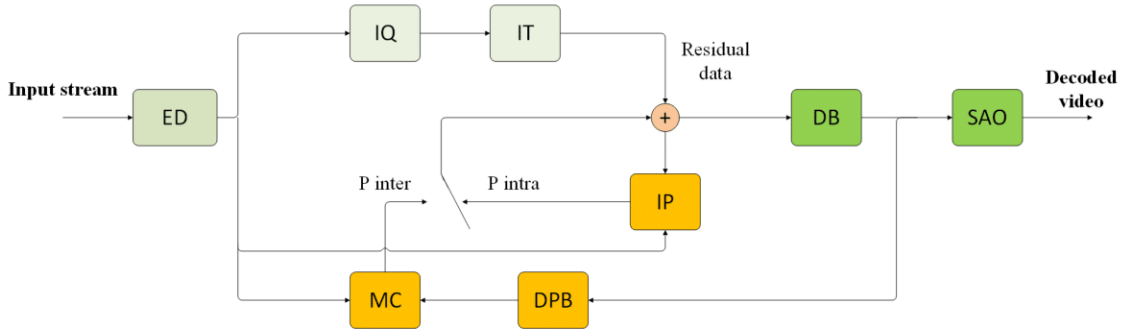


FIGURE 2.5 – le Schéma fonctionnel du décodeur du HEVC.

ou plusieurs tranches. Les tranches sont autonomes et peuvent être codées séparément à l'aide d'un de ces types :

1. I tranches - tous les CUs sont codés en utilisant uniquement la prédiction Intra.
2. P slice -CUs peut être codé en utilisant Intra prédiction on inter prédiction avec compensation de mouvement uni-prédiction.
3. Tranche B - Le CUs peut être codé en utilisant la prédiction Intra ou la prédiction Inter avec compensation de mouvement uni-prédiction ou bi-prédiction. ii. Les tiles sont aussi une séquence de CTU, mais I, des régions de forme rectangulaire. Ils sont employés dans le HEVC afin d'utiliser une architecture de traitement parallèle pour le codage.

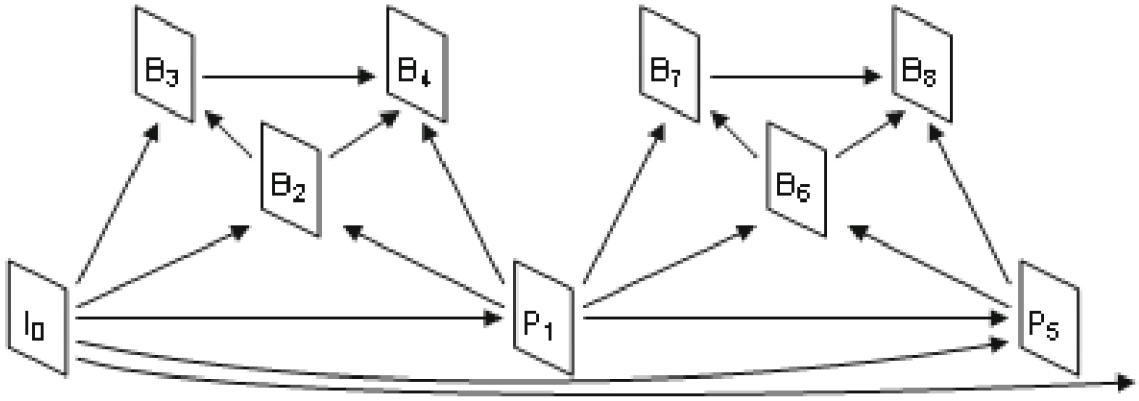


FIGURE 2.6 – exemple de I, B et P frames.

2.4 Structure du bloc

La Figure A.1 illustre les modes de partitionnement et les formes de PB Luma associées spécifiés pour les disjoncteurs codés de taille 2N × 2N. Le jeu de modes pris en charge comprend deux modes carrés (PART_2N×2N et PART_N×2N et PART_N×N),

deux modes SMP (PART_2N×N et PART_N×2N) et quatre modes AMP (PART_2N×nU, PART_2N×nD, PART_nL×2N et PART_nR×2N). Avec les modes AMP, n représente une plus petite des partitions, alors que U, D, L et R indiquent respectivement haut, bas, gauche et droite. Par exemple nU indique que la plus petite partition se trouve dans la partie supérieure du bloc. L'utilisation de CU comme racine pour la prédiction garantit que la taille d'un PB est toujours la même ou plus petite que celle du CB.

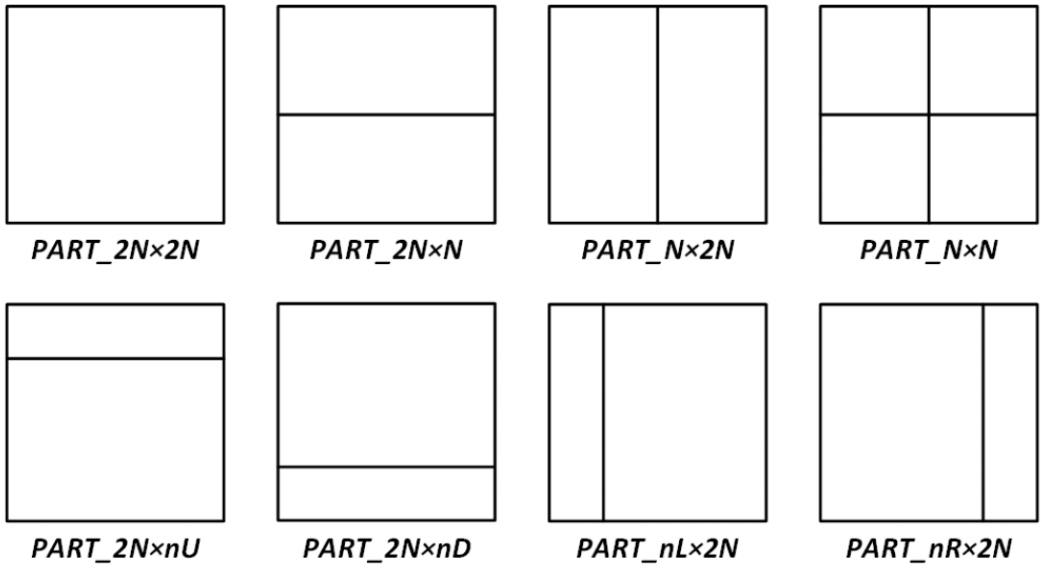


FIGURE 2.7 – les modes de partitionnement et les formes de PB Luma.

2.4.1 La transformation

- Transform and Quantization (TQ) :La transformée est une estimation de la transformée cosinus discrète (DCT) avec des valeurs entières de haute précision de quatre matrices de transformée entières, qui traite des matrices de transformée en deux dimensions et a des tailles de 4×4, 8×8, 16×16 et 32×32, en outre, le HEVC a utilisé des valeurs approximatives de la transformée discrète en sinus (DST) seulement pour bloc 4×4 de la matrice transformée en entier, qui ne s'est appliquée que dans un bloc Luma de l'intra prédiction 4×4 résiduel. La proposition de quantification dans le HEVC est semblable à celle qui existe dans le H.264/AVC où le paramètre de quantification (QP) choisit de 0 à 51 pour 8 bits, deux paramètres de quantification (Qstep) et l'arrondi subséquent sont utilisés pour la quantification par division et par multiplication pour la quantification inverse.

2.4.2 La prédiction intra

La prédiction intra (IP) est calculée sur la base des valeurs de la simple sortie de référence dans les UT déjà en cours de traitement, on suit la taille de l'encodage PU, qui a quatre tailles de blocs PU intra modes différents [Lai+12]. Le nombre de modes est en augmentation

par rapport au H.264/AVC, il peut atteindre 35 modes (A.2) et peut être classé selon trois catégories, qui sont angulaires, DC et planaires. Les modes 0 et 1 sont respectivement DC et planaire. Pour les modes 2 à 18 sont la première moitié horizontale des modes de prédiction angulaire, pour la deuxième partie verticale les modes sont de 19 à 34. Les modes de prédiction angulaire fournissent une texture complexe et une composante haute fréquence, il est basé sur un calcul géant appliqué dans 33 directions différentes qui utilise les pixels voisins. Les modes DC et Planar dédient pour une image qui a un changement de banc similaire entre les pixels voisins, ces modes sont bons pour générer un bloc de prédiction neutre. Le mode DC est basé sur tous les pixels voisins juste à côté des pixels verticaux et horizontaux, à l'exception des pixels d'angle.

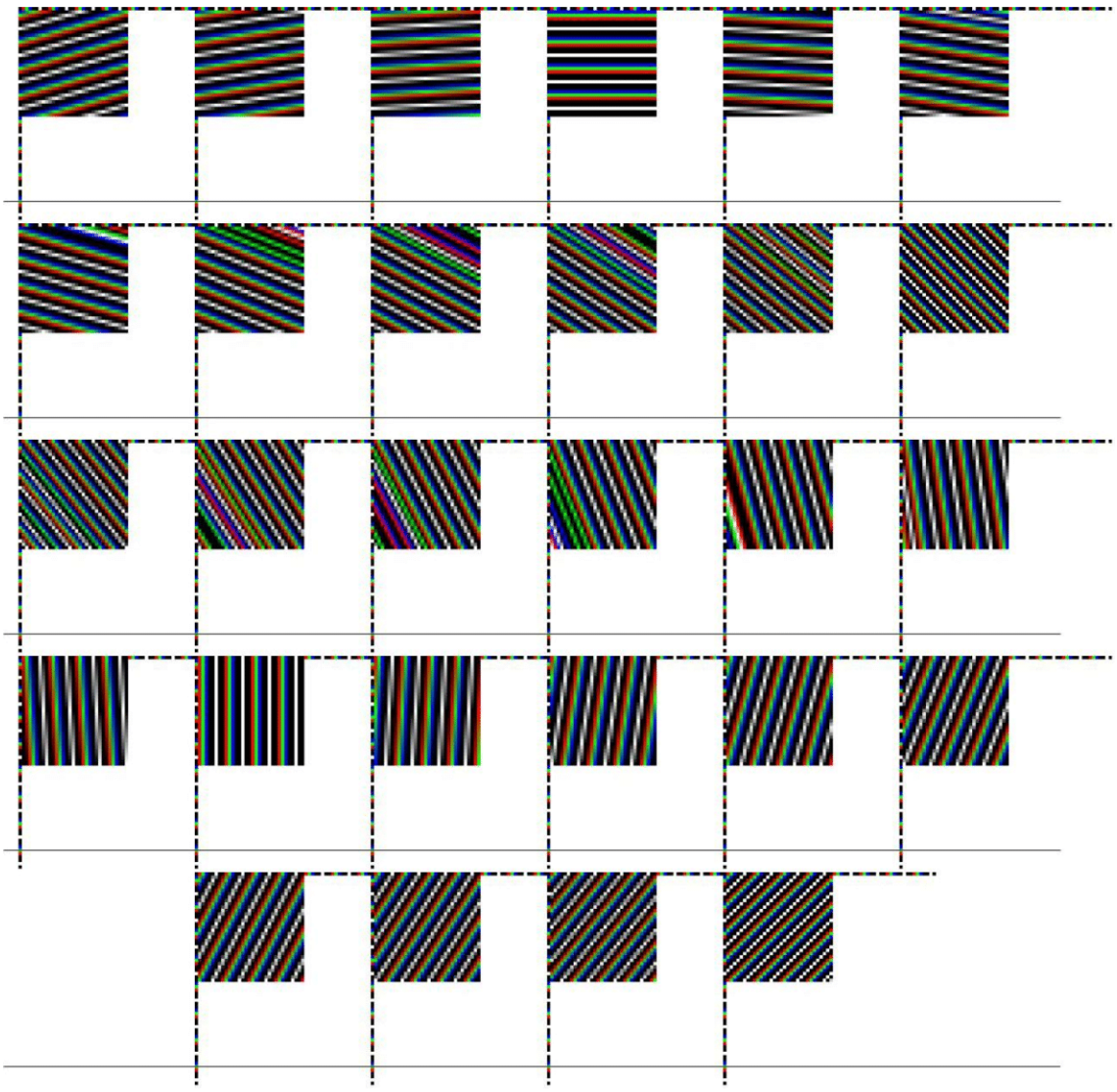
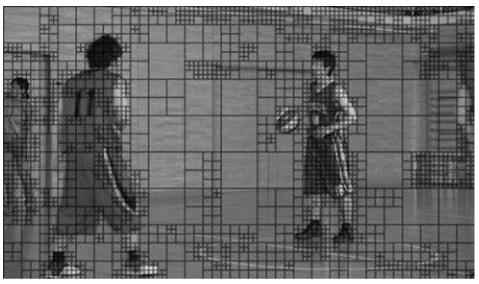


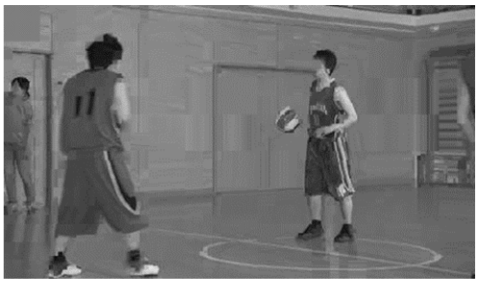
FIGURE 2.8 – les différents modes de l'Intra prédiction utilisé par le standard HEVC.

2.4.3 La prédiction inter

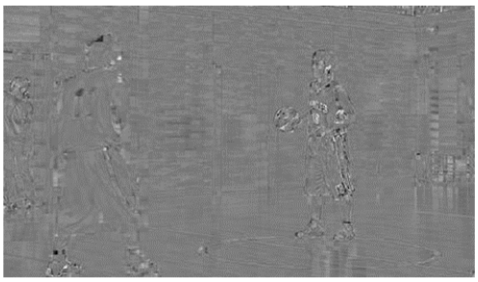
La prédiction inter, ou la prédiction temporelle, exploite le partage de nombreuses ressemblances et que certaines de leurs régions constituantes se déplaceront dans leur ensemble comme l'exemple de la 2.9 Comme l'ordre d'encodage n'est pas nécessairement le même que l'ordre de visualisation, les inter prédictions peuvent avoir leur origine dans des images passées ou futures, et combiner les deux origines. Cette fonction facilite le suivi des mouvements entre les images. Les images qui sont prédites à l'aide d'une seule image du passé ou du futur sont appelées images prédites ou tranches P. Les images de type B ont des origines de prédiction dans deux images différentes.



(a) Partitionnement



(b) Image prédite



(c) Image prédite

FIGURE 2.9 – (a) Partitionnement, (b) : Image prédite, (c) : Image résiduelle

Inter prédiction (InterP) : la prédiction Inter est utilisée pour réduire la redondance temporelle entre les frames des séquences YUV par bloc de détermination prédit pour l'image courante lorsqu'elle est disponible dans le tampon de l'image décodée (i.e. dans le mémoire déjà réservée pour les images décodées). Il est traité en premier par prédiction à compensation de mouvement basée sur des vecteurs de mouvement et s'écoule par filtrage d'interpolation de sous-échantillon. La prédiction par la compensation de mouvement exploite la redondance temporelle entre les séquences vidéo où le bloc prédit est présent sur un ou deux blocs déplacés dans le mémoire des images interne avant le décodage. Les vecteurs du mouvement indiquant la dislocation du bloc de prédiction, de plus, les indices de référence présentant toute l'image en pouvant être utilisés pour trouver le bloc de prédiction, en se basant sur ces informations, la prédiction avancée du vecteur de mouvement (AMVP) fournit le meilleur bloc de prédiction dans la liste des prédicteurs de vecteur de mouvement qui sont sélectionnés parmi trois blocs voisins dans le temps et dans l'espace. 4) La prédiction ou compensation de mouvement

(MC) suit la structure principale des CUs et aussi des PUs, elle est utilisée pour calculer le vecteur du mouvement en utilisant la prédiction unidirectionnelle pour un cadre de référence et la bidirectionnelle pour deux cadres de référence, dans le cas de Luma selon le découpage symétrique ou asymétrique des Pus et de chaque PU, les vecteurs de mouvement ont une résolution $\frac{1}{4}$ de pixel, les positions des pixels sont non-entiers celles obtenues par interpolation avec filtre (FIR) à 8-tap, on détermine la direction horizontale puis la verticale. Pour le mode Chroma à une résolution de 1/8ème de pixel que l'utilisation d'un filtre d'interpolation 4-tap.

2.4.4 La quantification

La quantification est appliquée dans le domaine de la transformation, est utilisée comme opérateur d'approximation, réduisant ainsi le nombre de valeurs de sortie possibles. Dans des normes comme le HEVC, la quantification est scalaire : chaque coefficient est approximé indépendamment de ses valeurs voisines. Dans ces schémas de codage, l'étape de quantification est contrôlée par un paramètre de quantification (QP) qui élimine tout coefficient dont le niveau d'énergie est inférieur à un certain seuil. Les coefficients énergétiques élevés sont également affectés par la quantification. Il convient de noter que la quantification est la seule étape irréversible de l'ensemble du système de codage vidéo hybride, qui induit un codage vidéo avec perte. Le codage vidéo sans perte (ou presque sans perte) ne peut être atteint en n'utilisant pas la quantification dans le processus. La quantification inverse (QI) est similaire aussi à celle H.264/AVC. Cette opération permet d'augmenter le coefficient décodé juste après le décodage d'entropie.

2.4.5 Les filtres

La norme HEVC spécifie deux algorithmes de filtrage distincts. Le filtre de déblocage est issu des normes précédentes. Il filtre les pixels de bordure du CB par des règles simples basées sur les types de prédiction des blocs de bordure. Filtre de déblocage (DF) et filtre d'adaptation d'échantillon (Sample-Adaptive Offset SAO) : Le filtrage en boucle est une étape avant le décodage et le stockage de l'image sur tamponne mémoire pour l'utilisation après la prédiction, le HEVC a adopté deux filtres en boucle consécutifs, le premier est le filtre de déblocage qui est dédié à réduire les bords des blocs afin d'améliorer la qualité visuelle sur la structure des blocs, il est reconnu par le traitement élevé car il applique un filtre vertical après un filtre horizontal sur les bords pour chaque bloc codant par CU, PU et TU. Le deuxième filtre est le décalage adaptatif d'échantillon (SAO) qui est fonctionnel avec la sortie de données du filtre de déblocage lorsqu'il est probablement appliqué avec des transformations (TUs) de plus grande taille, ce qui réduit la distorsion d'échantillon. Les deux filtres sont exécutés séquentiellement afin de reconstruire d'images et améliorer considérablement la qualité visuelle de ces images. Il y a plusieurs conditions à suivre pour l'appliquer, la plupart du temps, il a d'une part une taille de 8×8 échantillons pour le composant Luma et d'autre part un 4×4 échantillons pour le composant Chroma. Car SAO n'est appliqué que pour reconstruire des échantillons qui ont des distorsions régulières causées par des processus de transformation, de

quantification ou de prédiction.

2.4.6 Le codage entropique

Entropy Encoding (ED) :la dernière étape de l'encodage vidéo est le codage entropique, qui est la compression de données sans perte. La méthode adoptée dans le HEVC est le Context-Based Adaptive Binary Arithmetic Coding (CABAC), il est utilisé pour la première fois par H.264/AVC avec Context Adaptive VLCs (CAVLC), mais CABAC est mis en œuvre uniquement dans HEVC, car il est connu par l'efficacité de codage élevé qui améliore le débit pour leurs techniques qui peuvent être réduire le contexte codé, l'intervalle de dépendances dans le traitement augmente le résidu pour la parallélisassions de l'encodeur. Pour le décodage d'entropie (ED) est utilisé pour décoder le flux binaire d'entrée dans le décodeur. Le HEVC n'utilise aussi que le codage arithmétique binaire adaptatif au contexte (CABAC) comme outil de décodage d'entropie, ce dernier est connu pour ses bonnes performances en termes de taux de compression et son efficacité élevée pour le décodage [SB12], mais il a permis une consommation grande de la batterie des appareils mobiles. La configuration du schéma interne de CABAC est organisée séquentiellement en des étapes de base donnant dans la 2.10 :

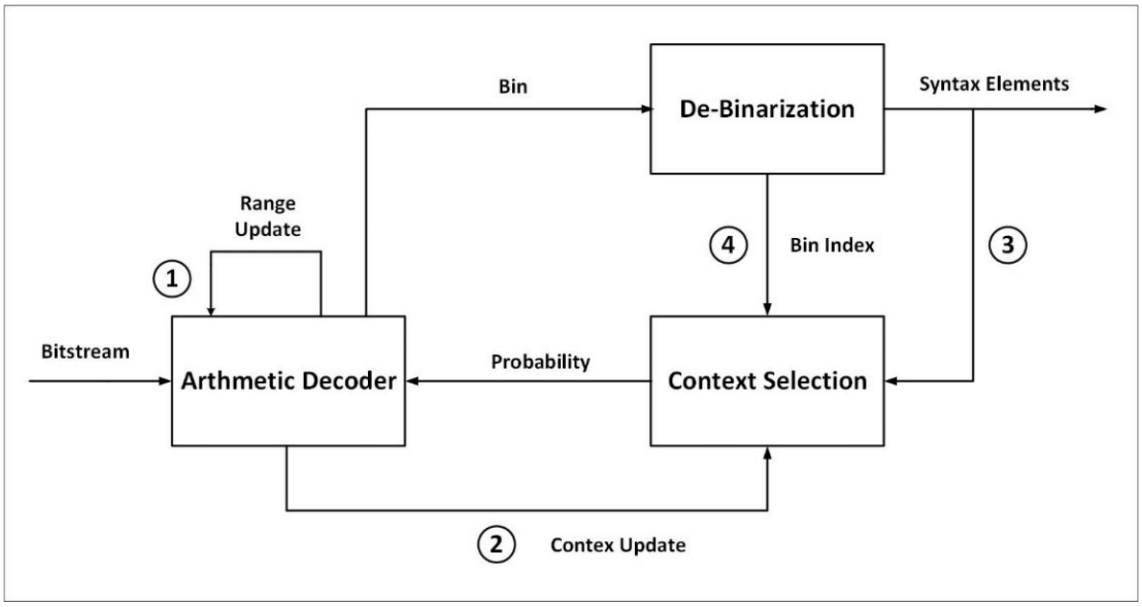


FIGURE 2.10 – Les outils de l'algorithme CABAC : binarisation, sélection du contexte et codage arithmétique.

(1) La modélisation du contexte (y compris la mémoire de contexte et la sélection du contexte) offre un modèle de contexte pour (2) le décodeur arithmétique binaire, la bin décodeur actuel et aussi (2) selon la bin précédente de (1), car (3) la dé-binarisation est utilisée à la fin pour compresser les bins en bits utilisant plusieurs différents processus de dé-Binaire qui sont appliqués la probabilité.

2.5 Les paramètres de norme HEVC

Slice Tranche : Une tranche est l'ensemble des données codées correspondant à une partie, ou à la totalité, d'une image. Chaque tranche peut être décodée indépendamment l'une de l'autre en termes de codage entropique, de prédiction ou de reconstruction du signal. Cela réduit les erreurs de décodage en cas de perte de transmission. Chaque tranche est composée d'un nombre variable de CTU. Tiles Carreaux : dans le cadre du standard HEVC, un sous-niveau supplémentaire a été défini pour faciliter le traitement parallèle de différentes régions d'image. Les tuiles correspondent à des régions rectangulaires d'une image et peuvent être décodées indépendamment, elles peuvent aussi partager les mêmes informations d'un en-tête de tranche.

2.5.1 Traitement parallèle par front d'onde (WPP)

Étant donné que le HEVC est conçu pour fonctionner avec la vidéo 4K, deux outils de parallélisassions au niveau des données ont été inclus dans la norme : Dans les Figure 2.11 et Figure 2.12, Les tiles permettent de découper la séquence en petits morceaux rectangulaires qui peuvent être traités en parallèle. La prédiction sur différentes tuiles peut être gérée en limitant les prédictions pour pointer vers la tuile courante, mais cette approche réduit l'efficacité du codage. Une autre approche consiste à ajouter de la logique pour forcer toutes les tuiles voisines dans les cadres de référence à être complétées avant de traiter la tuile, mais cela ajoute des dépendances qui limitent la parallélisassions.

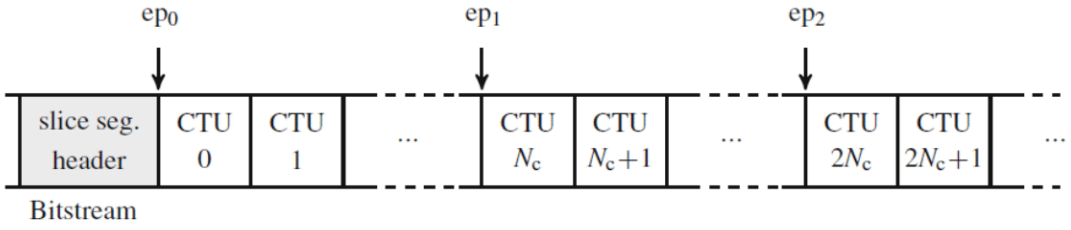


FIGURE 2.11 – Points d'entrée frame pour démarrer le décodage aux positions des octets indiquées dans l'en-tête de tranche.

Dans la Figure 2.13, Le WPP utilise des rangées séparées de CTU dans le codage, avec synchronisation du contexte à la rangée suivante après deux CTU [CHP11]. Cela signifie que la ligne suivante peut être démarrée lorsque deux CTU de la ligne précédente ont été codées. L'efficacité de la compression n'est que très peu affectée et l'encodage et le décodage peuvent être mis en parallèle assez efficacement. La synchronisation de contexte requise entraîne certaines limitations aux parallélisassions. Il faut coder deux CTUs avant de pouvoir lancer la ligne suivante de CTU, de sorte que le traitement peut être lancé avec un seul fil, en augmentant au fur et à mesure que le nombre de lignes de CTU codées augmente. De plus, la parallélisassions peut-être encore augmentée en encodant plusieurs images en parallèle.

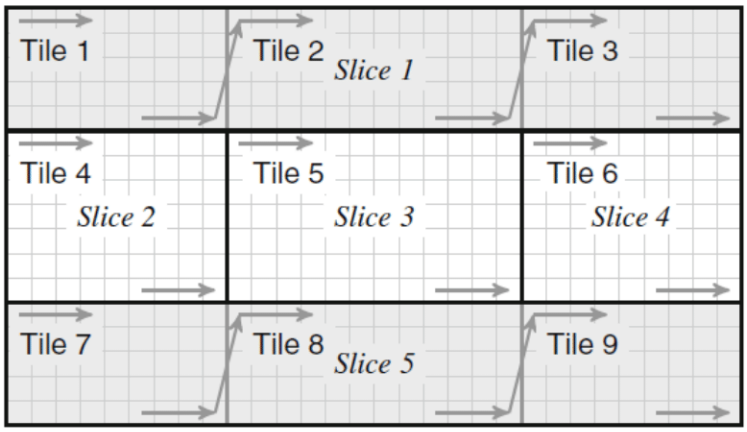


FIGURE 2.12 – partitionnement de la même séquence en des tiles et slices.

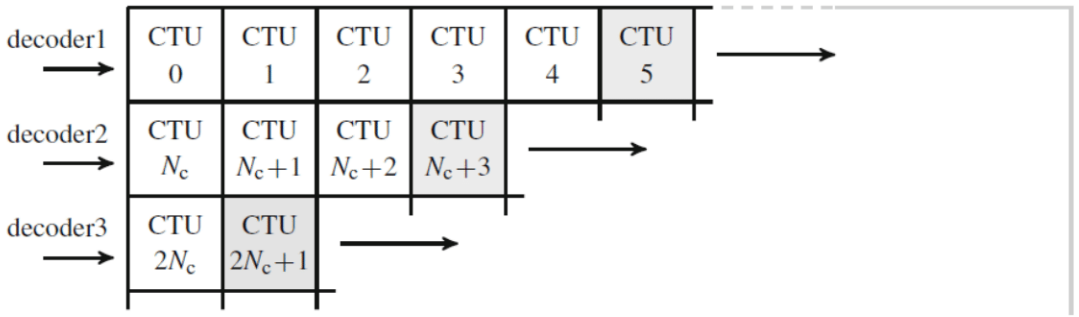


FIGURE 2.13 – Traitement parallèle par front d'onde de Nc CTU par ligne. Les décodeurs peuvent commencer à décoder chaque ligne avec un retard de deux CTU par rapport à la ligne précédente.

2.5.2 Configurations temporelles

Les trois configurations suivantes sont les plus importantes.

— 1) All intra (AI) : la prédiction intra est utilisée seule.
— 2) Low delay (LD) : la première image est encodée en utilisant des tranches I.
— 3) Accès aléatoire (RA) : image à accès aléatoire environ toutes les 1s.

2.6 Résumé

Tout au long de ce chapitre, les principes de l'encodage vidéo numérique ont été présentés, y compris un bref examen de l'évolution des normes d'encodage au cours des dernières années. La norme H.265/HEVC a été introduite. La caractéristique la plus remarquable de cette norme est l'amélioration de jusqu'à 50% du débit binaire requis pour transmettre une

séquence vidéo par rapport à son prédécesseur H.264/AVC pour le même niveau de qualité. Les principales caractéristiques de la norme, les profils et les niveaux ont été présentés, ainsi que les principaux blocs fonctionnels qui composent les codeurs. Enfin, la norme vidéo HEVC a été introduite. HEVC permet des descriptions de haut niveau des décodeurs/encodeur vidéo et intègre un ensemble de blocs avec diverses fonctionnalités nécessaires pour implémenter ces types d'applications. Cette norme représente une révolution dans le développement des codeurs vidéo, puisqu'elle permet de réaliser des conceptions d'une grande modularité tout en permettant la réutilisation de la compression de hautes résolutions avec moindre de débits.

Chapitre 3

Implémentation software du codeur HEVC

Sommaire

3.1 Introduction 51
3.2 Implémentation 53
3.3 L'implémentation du codeur sur DSP 54
3.3.1 L'exécution de l'encodeur 54
3.3.2 L'exécution du décodeur 55
3.3.3 Performances de l'exécution 58
3.3.4 Profilage des résultats avec analyse 58
3.3.5 Comparaison avec d'autres travaux 60
3.4 Conclusion 61
3.4.1 CONCLUSION 64
3.5 L'implémentation de Décodeur sur ARM 64
3.5.1 Performances de l'exécution 64
3.5.2 Résultats du profilage 67
3.5.3 Conclusion 69
3.5.4 Comparaison avec d'autre travaux 69
3.6 l'implementation de l'estimation du mouvement rapide sur DSP . . 70
3.6.1 les estimateurs rapides du mouvment 70
3.6.2 Diamond search 71
3.6.3 SDSP 71
3.6.4 HDS 72
3.6.5 LDPS 73
3.7 Résumé 73

3.1 Introduction

Lorsque la première version logicielle du standard HEVC a été proposée, de nombreuses implémentations logicielles ont vu le jour pour offrir des analyses statistiques concernant le temps d'exécution, la consommation électrique et les analyses de complexité pour le codeur et le décodeur. Cependant, le décodeur est encore plus facile à étudier en raison de sa moindre

complexité par rapport au codeur. Par conséquent, on peut trouver de nombreux travaux se concentrant sur le décodeur dans la littérature scientifique. Cette étude se concentre sur l'implémentation logicielle de l'encodeur sur le processeur embarqué TMS320C6678. Il s'agit de prendre en considération le parallélisme au niveau instruction, de l'optimisation de l'architecture VLIW et le codage logiciel rapide proposé [Cha+01]. Ces implémentations logicielles nous permettent d'avoir une base de référence des évaluations auxquelles on peut généralement s'attendre sur le dernier processeur embarqué DSP de Texas Instruments.

Nous avons choisi d'implémenter également le codeur HEVC sur le DSP multi-cœur TMS320C6678 pour bénéficier d'une fréquence de traitement élevée et d'une architecture optimisée. Le DSP appartient à la dernière génération de DSP multi- cœur de Texas Instrument. Il s'agit d'un DSP à virgule fixe / flottante haute performance basé sur l'architecture multicœur KeyStone de TI. Cette architecture fournit une structure haute performance pour l'intégration des cœurs RISC (Reduced Instruction Set Computer) et DSP avec des co-processeurs et des E/S spécifiques aux applications. MulticœurNavigator, TeraNet HyperLink et Multi-cœur Shared Memory Controller fournissent une bande passante interne adéquate pour un accès non bloquant à tous les cœurs de traitement, coprocesseurs, périphériques et E/S.

Dans cet section, nous présentons une implémentation entièrement logicielle d'une dernière version du HEVC Test Module (HM.15.0) avec ses trois configurations intra, low-delay et random-access de l'encodeur HEVC sur les deux processeurs mobiles (processeur de signal numérique DSP et processeur ARM), qui atteint une bonne performance pour une séquence vidéo 480p (720 x 480 pixels) en utilisant un seul noyau fonctionnant à 1 GHz à une fréquence de fréquence des deux processeurs sans optimisation avec accès direct amélioré (EDMA) et le multi-cœur d'exécution. Ensuite, une implémentation d'un logiciel de comparaison dans deux processeurs basse consommation pour appareils mobiles est proposée. Afin d'améliorer cette implémentation puis la comparaison, en exploitant principalement l'optimisation offerte par ce type de processeur embarqué, qui est une extension de l'architecture NEON pour ARM qui est un cas spécifique de parallélisme de données au niveau des données SIMD (Single Instruction Multiple Data). Et VLIW pour DSP. L'étude de la complexité de cette implémentation sur les processeurs TMS320C6678 et ARM conduit à l"exécution du codeur HEVC en temps réel pour une haute résolution au moyen de diverses optimisations telles que EDMA fortement recommandées sur la base de cette analyse de complexité, ainsi que la parallélisassions sur huit cœurs existants dans cette plate-forme DSP ou l'exploitation des autres plateformes dont les processeurs peuvent atteindre 64cœurs.

Dans la littérature, nous pouvons trouver de nombreuses façons d'optimiser le standard des codeurs HEVC afin d'augmenter le décodage de trames par seconde, ce qui est un objectif similaire de notre travail dans les appareils à faible consommation d'énergie. Une façon de minimiser le temps de traitement sur le processeur mobile ARM est de tirer profit de la technologie NEON, connue sous le nom de SIMD. En fait, actuellement, les processeurs les plus disponibles sur le marché incluent les extensions de jeux d'instructions SIMD équivalents. Ce traitement de données multiples dans les mêmes opérations a commencé avec la plate-forme Intel x86 et a été nommé MMX. Depuis 1995, les instructions SIMD ont été utilisées dans un premier travail avec le vidéo standard MPEG-1 afin d'obtenir le traitement en temps

réel pour la basse résolution 352×288 pixels, par la suite, SIMD ont été appliquées dans plusieurs versions de décodeurs vidéo comme H.263, H.264. Dans [Yan+12] Yan et al, ont implémenté le HEVC sur des processeurs x86 et exploité la technique SIMD en tant que fonctions arithmétiques des Extensions 2 (SSE2) de l'architecture x86 pour paralléliser chaque bloc du décodeur afin d'augmenter le décodage de trame par seconde, que leurs performances sont 5 fois supérieures pour une résolution FHD. Dans [Chi+15] Chi et al présente le SIMD optimisation pour HEVC décodeur sur plusieurs architectures mobiles et également sur les plateformes PC. Leur plus haute performance a atteint plus de 11 fois pour le bloc Sample-Adaptive Offset sur architecture x86 en utilisant les Extensions Vectorielles Avancées 2.

3.2 Implémentation

Dans ce travail, nous utilisons le logiciel de référence HM.15.0 [TMS00] proposé par la norme HEVC. La version du logiciel HM est codée en langage de programmation C++, nous l'avons développée sur Code Composer Studio (CCS) qui est un environnement de développement de Texas Instruments où l'on peut trouver ce DSP intégré avec d'autres familles de processeurs intégrés. Ainsi, nous profitons de cet environnement pour bien adopter le logiciel sur les propriétés de cette version de DSP telles que les déclarations de variables et les configurations de mémoires etc. aussi, nous adoptons le logiciel pour l'implémentation ARM.

Pour la simulation, nous utilisons quatre séquences de classe C (832×480 pixels) qui sont BasketballDrill (BSK), PartyScence (PTS), BQMall (BQM), et RaceHorese (RAC) présentées dans le tableau 4.8 avec leurs caractéristiques comme la résolution, la profondeur de bit, l'image par seconde (fps) et le nombre de vues, que le document JCT-VC a choisi des conditions communes aux essais dans [Noaf]. D'autres configurations sont les paramètres de quantification (QP) avec les valeurs 22, 27, 32 et 37. Les trois configurations utilisées sont Intra (AI), Low delay (LD) et Random Access (RA), la première étant basée sur des images intra uniquement et sans référence temporelle, c'est-à-dire sans compensation de mouvement. Le deuxième faible retard n'a que la première image d'une séquence vidéo qui est codée en tant que prédiction Intra et les autres images sont codées en prédiction P ou B. Cette configuration est surtout utilisée sur les vidéos dédiées à la communication en temps réel. La troisième accès aléatoire (Random Access) est un premier et chaque image est codée à intervalles d'environ une seconde dans la prédiction Intra qui est généralement utilisée dans la vidéo de divertissement diffusée.

Table 3.1 – Quatre séquences de test sont utilisées pendant l'expérimentation.

Name	resolution	Bit depth	Fps	Frame count
BasketballDrill	832 × 480	8	50	150
PartyScence	832 × 480	8	50	150
BQMall	832 × 480	8	60	150
RaceHorese	832 × 480	8	30	150

3.3 L'implémentation du codeur sur DSP

3.3.1 L'exécution de l'encodeur

Le codeur HEVC est implémenté sur un seul noyau DSP c66x, en fait, nous n'implémentons pas le codeur dans les huit noyaux car c'est la première implémentation du codeur HEVC sur DSP dans la littérature. Notre travail s'intéresse aux statistiques sur un seul noyau, de plus, si nous utilisons le multi-cœur, la communication entre les noyaux peut augmenter le temps d'exécution. Le premier travail consiste à exécuter l'encodeur sur un DSP simple cœur pour déterminer le temps nécessaire pour les 150 premières trames de chaque séquence, le temps exprimé (en seconde) est donné dans le tableau 4.10 pour les différentes séquences choisies.

TABLE 3.2 – Le temps d'encodage obtenu pour les 150 images de quatre séquences qui sont choisies pour les tests en secondes.

config	**intra**				**lowdelay_P**				**randomacces**			
Séq	**22**	**27**	**32**	**37**	**22**	**27**	**32**	**37**	**22**	**27**	**32**	**37**
BDrill	11385	8905	7802	7277	26454	22471	20949	18 901	26475	20401	16600	14777
PScence	14842	11949	10138	9058	31781	30775	23933	20170	25092	23717	18991	15302
BQMall	11661	9181	8499	7603	34271	25412	20918	18888	23028	19929	16581	14728
RHorese	12187	9821	8518	7828	34502	37209	29139	23779	30483	27507	22507	18444

Les remarques qui reprennent les résultats du tableau 4.10 ci-dessus sont les suivantes : premièrement, si QP augmente le temps d'exécution augmente aussi, deuxièmement, la configuration intra est plus rapide en suivant RA puis LD, troisièmement, les résultats de cette simulation ne sont pas en temps réel. Enfin, les différences entre les résultats de ces séquences sont dues aux caractéristiques de mouvement et de texture dans chaque séquence et dans chaque scène. Dans cette section, nous étudions la complexité de chaque bloc à partir du schéma de l'encodeur donné en Figure 2.4. Pour cela, nous limitons notre travail à deux valeurs moyennes de QP 27 et 32, pour être plus précis, nous calculons le temps nécessaire pour chaque méthode lors de l'exécution. Le tableau 4.10 donne un exemple pour les méthodes (fonctions) plus complexes appliquées sur la séquence BasketballDrill avec QP égale à 32. Nous utilisons 150 premières trames pour le profilage du codeur et toutes les exécutions sont sans optimisation. Dans le tableau 4.10, la méthode xt_xquant est plus complexe ; les complexités ne sont pas dans son algorithme, mais elle avait beaucoup de fonctions déduites pour la copie comme mem_copy, dans l'ensemble, DSP est connu par lentement pour copier les données entre la mémoire contre elle est plus rapide dans le traitement des données. A part cela, pour la prédiction intra configuration predIntraLumaAng qui est dédiée au mode de prédiction intra Luma. Pour les configurations RA et LD, l'estimation des mouvements est plus complexe et elle est déduite pour le calcul de l'estimation du mouvement.

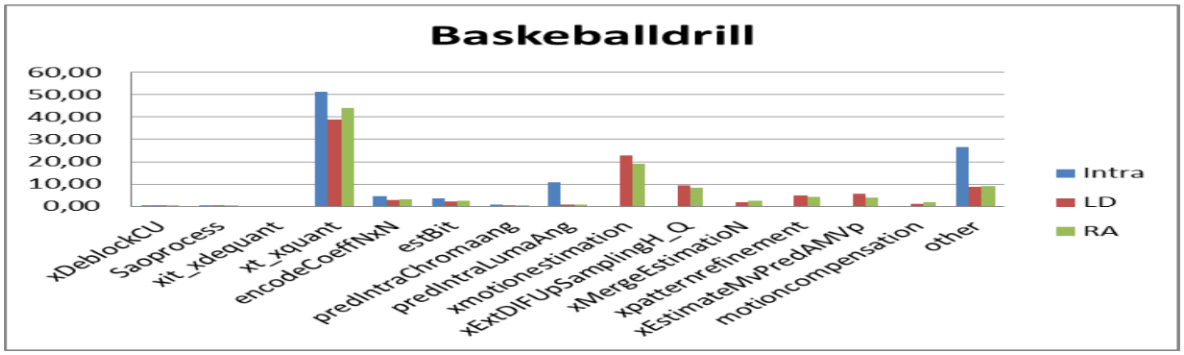
Dans la figure 3.1, nous donnons le résultat de quatre séquences avec ces trois configurations (AI, RA et LD), où il y a une comparaison individuelle des variations de ces méthodes

(fonctions).

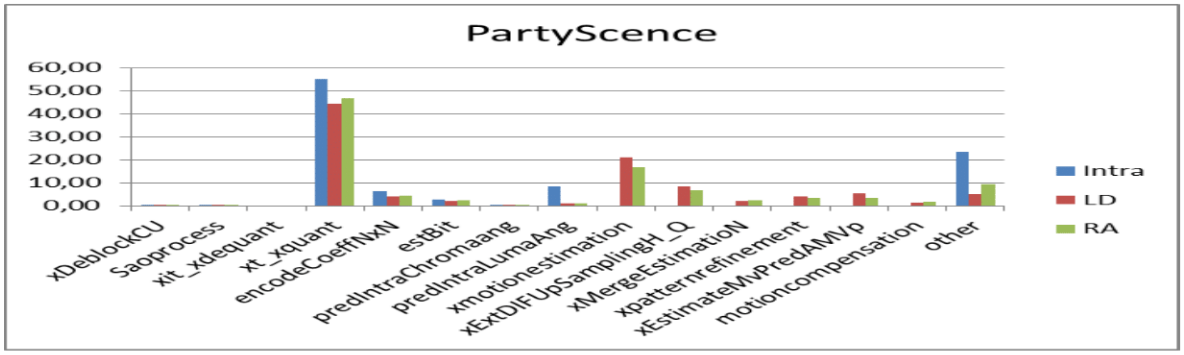
Le tableau 3.3 montre les pourcentages du temps de calcul de toutes les méthodes dans ces blocs, pour trois configurations, intra, LD et RA avec deux valeurs de QP, 27 et 32. La transformation et la quantification prennent environ 50% du temps total pour ces traitements et spécifiquement pour les fonctions de copie. Cependant, pour les deux configurations, LD et RA, l'inter prédiction est plus complexe et prend environ 21,2% à 46 %. Par rapport à la configuration intra avait la prédiction intra bloc plus complexe, qu'il faut autour de 10,11% à 11,95%. Pour les trois configurations, nous avons constaté que le codage par entropie prend de 5,33 % à 10,46 %. Pour les codeurs HEVC, les deux filtres SAO et DF sont très négligeables par rapport aux autres blocs, ils ne dépassent pas 0,19%.

3.3.2 L'exécution du décodeur

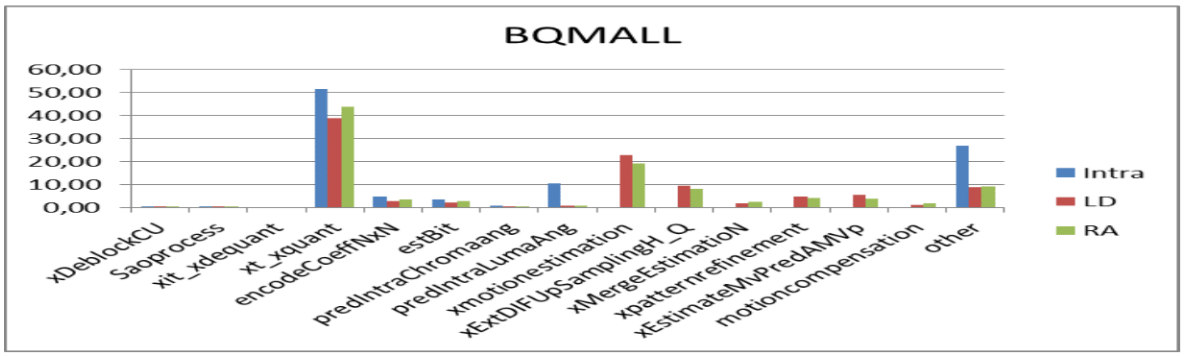
Dans notre simulation, nous avons sélectionné parmi la liste proposée par le standard dans le fichier [Bo11], les séquences utilisant dans les simulations suivantes données en bas, ces séquences se distinguent les unes des autres par leur mouvement haut ou bas et leur texture complexe ou lisse. Les caractéristiques de ces séquences sont les suivantes : la première est la résolution, qui est la somme des pixels contenus dans chaque image et présentée comme le nombre de pixels en longueur multiplié par le nombre de pixels en largeur. La second est le nombre d'images, qui est le nombre d'images dans l'ensemble de la séquence. La troisième est le nombre d'images à décoder en temps réel. La dernière est la profondeur de bit, qui est le bit pris par chaque pixel. De plus, la norme limite la valeur de la PQ à quatre valeurs comme suit : 22, 27, 32 et 37. En fait, les trois configurations utilisées pour les exécutions sont Intra, Low delay et Random access. La première est basée sur des images intra seulement et sans référence temporelle, c'est-à-dire sans algorithme de compensation de mouvement. La second est le retard minimal, qui revient à trouver la première image dans une séquence vidéo est décodée comme une prédiction Intra seulement et les autres images sont décodées en intra et/ou inter prédiction avec les deux configurations P et B prédiction. Cette configuration est surtout utilisée sur les vidéos dédiées à la communication en temps réel. La troisième est l'accès aléatoire, qui est une première et chaque image environ un second intervalle sont décodés sur Intra prédiction et les autres images sont décodées en inter et/ou intra prédiction, qui est généralement utilisé dans la vidéo de divertissement diffusée. Toutes ces caractéristiques et encore plus d'informations sont mentionnées dans le fichier de configuration proposé par la norme [Bo11]. Afin de suivre une norme commune, nous avons choisi les configurations à partir des conditions d'essai communes des documents JCT-VC [Bo11]. Nous avons sélectionné des séquences des classes E (1280 × 720), C (832 × 480) et D (416×240), les séquences ont une profondeur de 8 bits et un sous-échantillonnage couleur 4 :2 :0 pour les composantes Luma et Chroma qui les nomme et leur numéro de trame comme suit : Classe E : KristenAndSara(600), Johnny(600), FourPeople(600). Classe C : BasketballDrill(500), PartyScence(500), BQMall(600), RaceHorese(300). Classe D : BlowBubbles(500), BQSquare (600), BasketballPass (500), RaceHorese (300).



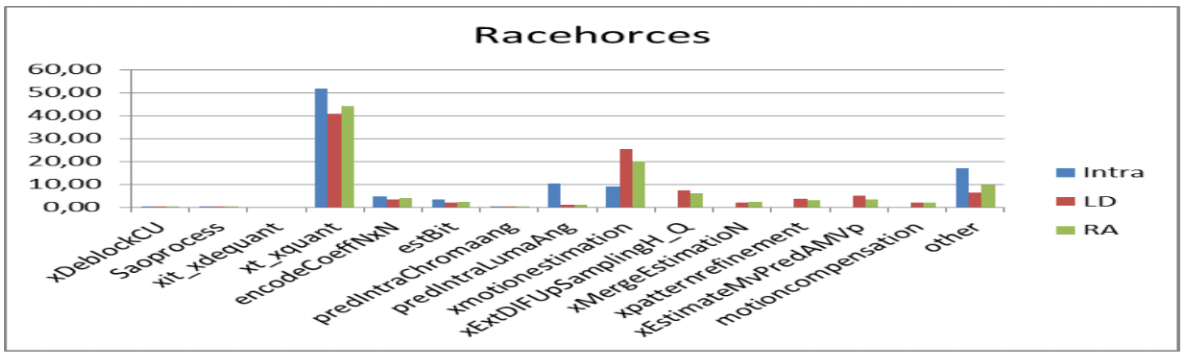
(a) BasketBalldrill.



(b) PartyScence.



(c) BQMall.



(d) RaceHorce.

FIGURE 3.1 – BasketBalldrill,(b) :PartyScence, (c) :BQMall, (d) :RaceHorce, la valeur de QP=32.

TABLE 3.3 – Le profilage du codeur HEVC en pourcentage de blocs

Seq	Cfg	QP	EP	QT	INTRA	INTER	DF	SAO	OT
FourBeaple	**AI**	27	8,36	23,94	37,41	0,00	12,95	4,06	13,28
		32	6,45	51,45	13,46	0,00	13,29	3,93	11,42
	LD	27	4,37	8,77	0,45	40,98	9,38	1,00	35,06
		32	3,29	6,02	0,00	46,57	7,97	0,44	43,78
	RA	27	3,49	5,03	1,92	50,39	6,66	0,94	32,38
		32	2,83	4,03	1,72	57,42	6,42	0,73	26,86
	AI	27	6,29	25,42	41,10	0,00	11,87	3,44	11,87
		32	4,83	26,48	41,23	0,00	12,38	3,30	11,78
Johanne	**LD**	27	2,70	4,74	0,12	50,45	6,52	0,41	35,10
		32	1,59	2,15	0,09	33,12	4,73	0,15	13,76
	RA	27	2,35	3,20	0,96	58,97	5,46	0,55	28,49
		32	1,95	2,40	0,90	59,55	5,06	0,34	30,59
BasketBallDrill	**AI**	27	12,39	20,27	34,44	0,00	14,41	4,66	13,83
		32	0,91	32,00	35,16	0,00	15,20	4,57	13,25
	LD	27	6,30	30,47	2,54	32,78	9,39	2,40	16,11
		32	6,85	22,48	2,28	35,62	10,49	1,49	20,80
	RA	27	5,76	16,77	4,50	44,17	8,93	1,74	18,13
		32	4,69	14,72	4,22	44,50	10,06	1,22	20,59
BQMall	**AI**	27	11,67	22,61	35,35	0,00	12,82	4,10	13,46
		32	9,33	24,34	34,93	0,00	14,31	3,98	13,12
	LD	27	7,11	12,47	1,45	47,15	9,65	2,38	19,79
		32	5,23	9,88	1,35	52,29	9,59	1,34	20,33
	RA	27	5,80	7,42	3,13	56,41	8,36	1,30	17,58
		32	4,56	6,38	2,92	57,29	8,76	0,91	19,16
BasketBallPass	**AI**	27	11,88	26,43	30,40	0,00	11,35	3,95	15,98
		32	9,05	28,34	30,56	0,00	12,79	3,94	15,31
	LD	27	9,65	14,98	3,79	34,41	10,54	3,15	23,49
		32	7,54	13,51	3,47	36,37	11,62	2,32	25,17
	RA	27	8,42	10,79	5,88	40,52	10,24	1,92	22,23
		32	5,94	9,35	5,32	43,93	10,66	1,32	23,48
BQSqaure	**AI**	27	17,00	16,55	32,20	0,00	11,11	3,03	20,11
		32	14,96	15,19	34,25	0,00	12,76	3,21	19,63
	LD	27	10,08	4,89	0,15	51,04	9,42	2,37	22,04
		32	6,46	3,22	0,19	56,40	8,81	1,11	23,81
	RA	27	7,17	4,81	1,29	56,84	7,05	1,62	21,22
		32	4,61	2,24	1,26	63,32	6,33	0,78	21,45

3.3.3 Performances de l'exécution

La première étape consiste à exécuter le décodeur logiciel HM sur le processeur DSP unique afin de calculer le nombre de trames décodées par seconde. Dans ce travail, nous nous concentrons sur l'implémentation en noyau unique parce que c'est la première implémentation rapportée dans la littérature dans cette dernière génération de DSP, et à partir de ces statistiques dans un noyau unique, nous pouvons faire une bonne remontée en temps réel en exploitant la ressource minimale dans ce navire. La première mesure est de déterminer le nombre d'images par seconde (fps) pour toutes les séquences sélectionnées en utilisant l'optimisation VLIW, pour cela on calcule le temps de chaque exécution, puis on le divise par le nombre d'images contenues dans chaque séquence, les résultats sont présentés au Tableau 3.3. Le résultat pour chaque valeur du Tableau 3.3 et des tableaux suivants est une moyenne de 60 exécutions.

Les résultats du Tableau 3.3 jusqu'à 25 fps pour la séquence BQSqure avec qp=37 en configuration LD. Pour toute expérience. Nous remarquons que lorsque la valeur de QP augmente, le nombre d'images décodées augmente également. De plus, les résultats de l'image décodée par seconde pour l'IA sont inférieurs à ceux de la RA et de la LD, en raison de l'utilisation de l'intra seulement. Pour RA et LD, nous voyons que ce sont à peu près les mêmes résultats avec une légère différence en raison du contenu des frames. Généralement, nos résultats sur un seul noyau indiquent qu'en utilisant seulement les autres noyaux parallèles, nous pouvons être en temps réel (30 images par seconde) pour deux classes C et D, si on exploitait l'EMDA pour diminuer encore plus le temps, on pourra atteindre le temps réel pour la classe E.

Dans le Tableau 3.4, nous avons limité la simulation à deux séquences de chaque classe, parce que nous obtenons presque les mêmes résultats dans chaque classe. Par conséquent, la performance obtenue par l'optimisation VLIW s'appliquait à chaque séquence de classe deux (elles sont connues sous le nom de mouvement haut et bas). Cependant, l'accélération est traitée en fonction du temps d'exécution avec et sans VLIW. Les résultats de l'optimisation se situent entre 3,10 et 8,94. De plus, ces différences entre les résultats dus aux trois configurations (AI, LD et RA), à la PQ et aux caractéristiques de contenu des séquences.

3.3.4 Profilage des résultats avec analyse

Dans cette section, nous présentons l'analyse des performances du profilage sur le DSP. La complexité de chaque bloc à l'intérieur du diagramme du décodeur HEVC est déterminée en incluant dans chaque méthode (fonction) du Tableau 3.4 une fonction spécifique pour calculer le temps de consommation en fonctionnement complet. Le pourcentage d'allocation de chaque méthode (fonctions) est calculé en fonction du temps total.

Le Tableau 3.4 présente les fonctions les plus complexes en utilisant la séquence BQMall avec valeur fixe QP en 32. Les résultats montrent que les fonctions les plus complexes pour trois configurations comme suit : pour l'IA les méthodes les plus complexes sont xReconIntraQT utilise pour la prédiction intra qui prend 28%, suivie par la fonction invtransformNxN

utilise pour traiter le bloc de transformation inverse et la fonction loopFilterPic du filtre en boucle 25% et 13,68% respectivement. Pour LD et RA les valeurs de profilage sont les mêmes, la fonction la plus complexe est la compensation de mouvement utilise pour calculer la compensation de mouvement que jusqu'à 37%. Ensuite, la fonction xGetNewPicBuffer a des valeurs importantes jusqu'à 20% que l'utilisation pour la prédiction inter et il a les fonctions pour copier ou échanger les données entre la mémoire. En outre, il est connu les fonctions utilisant la mémoire prend charge importante de DSP. A cet égard, l'optimisation par EMDA peut être aidée en évitant le transfert de données par DSP. Enfin, nous pouvons déduire de ces statistiques de ces analyses de performance. Les fonctions xReconIntraQT et invtransformNxN pour la configuration AI, et la fonction motioncompensation pour les configurations LD et RA peuvent être implémentées sur des multicœurs pour réduire sa complexité. La Figure B.2 montre un exemple de comparaison du pourcentage des trois configurations AI, LD et RA de chaque fonction présentées dans le Tableau 7 Le pourcentage des différentes classes de logiciels par rapport à la charge totale pour BQMall avec QP=32 en utilisant la séquence QBMall avec QP=32. On peut voir la variation de pourcentage entre eux.

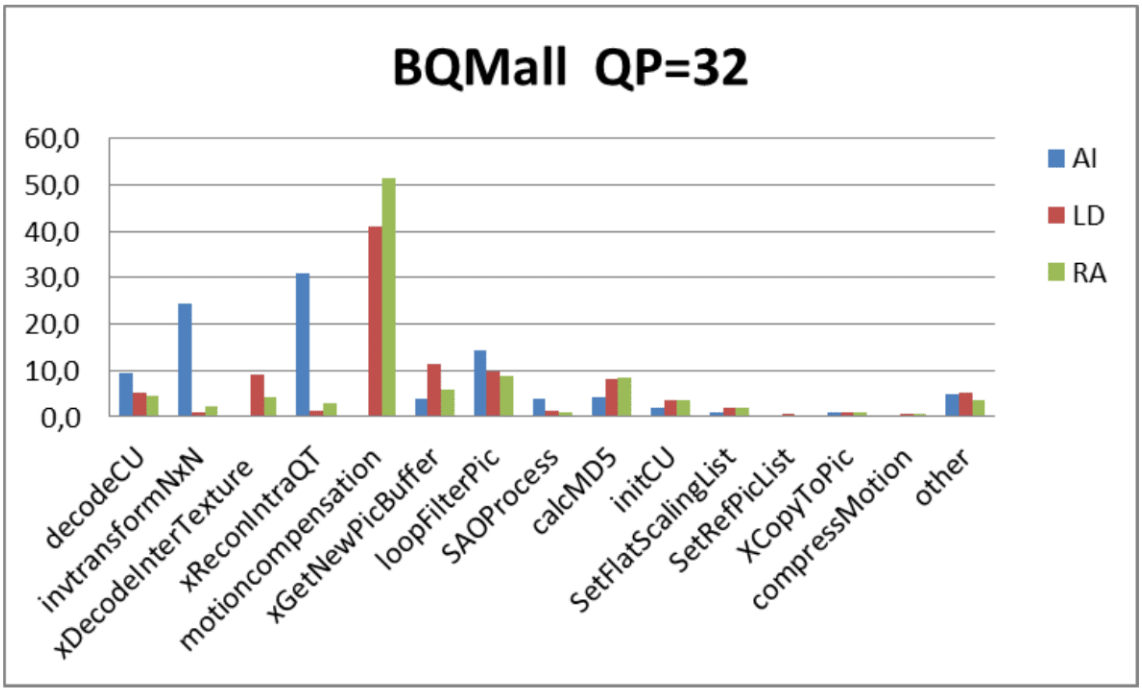


FIGURE 3.2 – Comparaison de la complexité des trois configurations AI, LD et RA pour la séquence QBMall avec QP=32.

Dans le Tableau **??**. Nous avons regroupé les fonctions dédiées à chaque bloc pour les utiliser dans les étapes suivantes comme le profilage par blocs présenté dans le schéma du décodeur de la figure B.1. Et aussi pour comparer nos résultats avec d'autres travaux.

Après avoir regroupé les fonctions, en donnant dans le Tableau 7 en bloc dans le Tableau 8, nous avons présenté le profilage de deux séquences pour chaque résolution pour la PQ égale à 27 et 32 dans le Tableau 3.4. Pour la configuration AI, nous pouvons voir qu'environ 30% du décodage chargé pour la prédiction bock intra se consume par la quantification inverse de

bloc et la transformation inverse d'environ 25%, par rapport aux configurations LD et RA, le bloc le plus complexe est la prédiction inter, qui est jusqu'à 63% et convient à l'optimisation du parallélisme, s'écoule par la quantification inverse de module et la transformation inverse avec le pourcentage environ 10%. Sauf que les autres blocs EP, DF et SAO ont un faible pourcentage et c'est un bloc différent les uns des autres selon les séquences et les paramètres des configurations. Enfin, le module TQ est adapté à l'optimisation EDMA car il dispose d'une fonction dédiée au fonctionnement entre la mémoire, telle que la copie de fonction qui permet d'obtenir un temps de décodage important.

TABLE 3.4 – CODING UNIT PARTITIONING OF PUS FOR EACH DIFFERENT CODING DEPTH LEVEL.

coding Depth level	**block partitioning**
0	64×64 ; 64×32 ; 32×64 ; 32×32 ; 16×64 ; 48×64 ; 64×16 ; 64×48
1	32×32 ; 32×16 ; 16×32 ; 16×16 ; 8×32 ; 24×32 ; 32×8 ; 32×24
2	16×16 ; 16×8 ; 8×16 ; 8×8 ; 4×16 ; 12×16 ; 16×4 ; 16×12
3	8×8 ; 8×4 ; 4×8

3.3.5 Comparaison avec d'autres travaux

Dans cette sous-section, les deux résultats dans la section ci-dessus, tels que la performance de l'exécution et le profil, qui sont comparés avec d'autres réalisations présentées dans un travail précédent. Parmi les travaux sur le DSP présentés dans [Pes+13], ils ont implémenté le logiciel HM.9.0 sur un DSP mono coeur, les résultats sont présentés dans le Tableau **??**.

TABLE 3.5 – Comparaison des performances de notre exécution avec d'autres sur DSP pour 832x480 (480p) requises par fps.

	configuration	IA		LD		RA	
	séquence	Min	max	Min	max	min	max
notre	480p	1.97	3.25	5	7.07	5.09	6.94
DSP[Pes+13]		1.4	1.9	2.8	4	2.7	3.6

Dans le Tableau **??**, nous comparons leurs séquences minimum (min) et maximum (max) afin de les appliquées de la résolution 480p. Ensuite, nous comparons avec d'autres profils des

modules décodeurs HEVC sur différents processeurs comme sur DSP [Pes+13], CPU[Vii+12], et sur processeur mobile intégré ARM [Bos+12].

TABLE 3.6 – Comparaison de la complexité de notre travail avec d'autres travaux de configuration RA.

QP	config	**EP**	**QT**	**INTRA**	**INTER**	**DF**	**SAO**	**OT**
27	notre	5,80	7,42	3,13	56,41	8,36	1,30	17,58
	[Bos+12]	4,8	3,7	1,8	52,7	13		24
	[Bos+12]	23	4	5	46	16	3	3
	[Pes+13]	14,5	10,7	11,9	29,2	13,9	4,8	15
32	notre	4,56	6,38	2,92	57,29	8,76	0,91	19,16
	[Bos+12]	3,5	3,8	1,5	55,1	13		22,9
	[Bos+12]	16	3	4	53	17	2	5
	[Pes+13]	11,1	10,7	12,7	32	14,8	4,7	14,1

Par rapport à ce travail, l'accélération d'obtenir jusqu'à 2 fois pour la configuration à faible délai, en outre, ils travaillent sur l'architecture DSP avec un seul noyau contre notre EVM avec multi-coeurs qui peuvent être exploités dans le travail suivant.

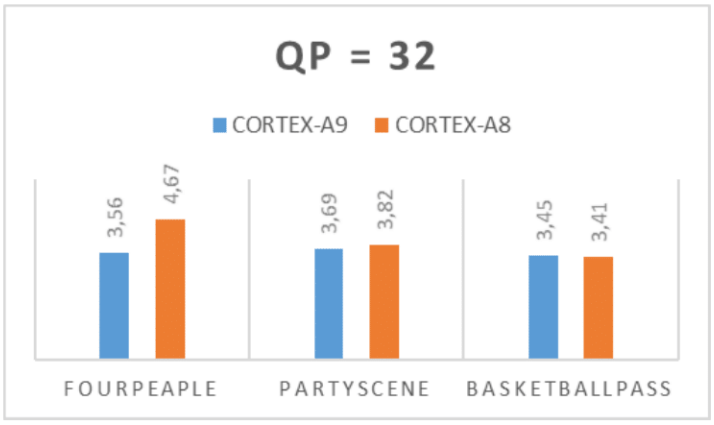
Dans le Tableau **??**, nous sélectionnons les travaux antérieurs utilisés la séquence QBMall, QP égal sont 27 et 32 et la configuration Random accès, qui est le plus utilisé leteratury, qui est généralement, nos résultats sont similaires à d'autres résultats signalés sur [Pes+13], [Vii+12] et [Bos+12], le bloc plus complexe pour nos et autres implémentations est permet inter prédiction présentée par compentation de mouvement avec valeur environ à 50%. Pour le bloc moins complexe, nous avons trouvé le filtre SAO pour toutes les architectures. Pour d'autres blocs, nous en avons trouvé peu différents, par exemple le QT fantaisie est plus complexe car notre travail contre le PE est un pourcentage élevé pour [Bos+12].

3.4 Conclusion

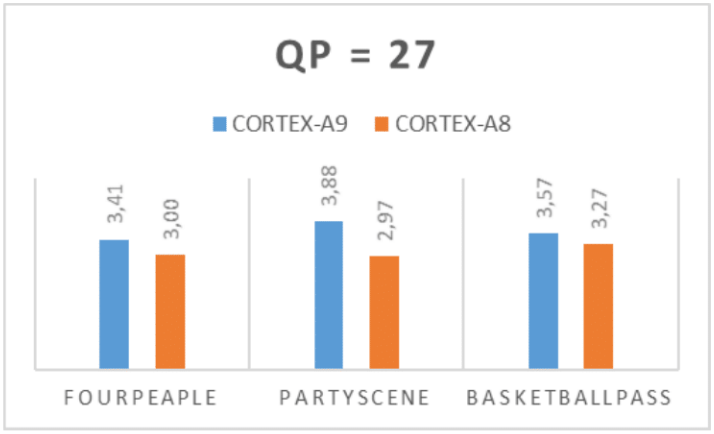
Le but de cette section est d'implémenter le décodeur HEVC sur le processeur embarqué DSP, une approche est définie pour construire le logiciel de décodage HEVC sur la dernière famille de version du DSP TMDSEVM6678, le résultat sur simple coeur donne une bonne performance par rapport aux mêmes autres implémentations sur processeur embarqué, en outre, la complexité de chaque bloc du décodeur est déterminée, en plus, la performance de l'optimisation avec VLIW est déterminée. Sur la base de ces résultats, nous pourrons définir une stratégie d'exploitation de tous les autres coeurs dans le but d'augmenter la fréquence d'images par seconde. Le temps réel pour la haute résolution a pu être obtenu en se basant sur l'implémentation sur plusieurs coeurs (huit coeurs) et en utilisant l'optimisation récente de l'IDMA comme QDMA et EDMA, qui sont intégrés dans cette puce qui peut être utilisée dans l'échange de données du plus grand mémoire. III.4. L'implémentation du codeur processeur embarqué ARM cortex Dans ce travail, une implémentation du dernier logiciel HM du

décodeur HEVC sur deux processeurs embarqués, ARM cortex-A8 et cortex-A9 avec technologie NEON qui accélère le temps de traitement à quatre fois est présentée. Ensuite, une analyse détaillée de la complexité est effectuée et détaillée, par conséquent, il sera utile tant pour la meilleure co-conception du décodeur sur la PFGA avec ARM qui existent sur un seul appareil, que pour un bon parallélisme sur les multicoeurs embarqués. III.5. L'implémentation d'encodeur sur ARM Il existe plusieurs travaux connexes dans la littérature sur l'accélération des codeurs HEVC utilisant l'optimisation à une seule instruction et à plusieurs données sur un processeur ARM et Intel x86 [Ahn+14],[Jun+16], [Bos+12]. les résultats de la comparaison sont réalisés sur deux plates-formes embarquées, la première est l'ordinateur monocarte BeagleBone Black à faible coût, équipé du SOC Sitara AM3358 de Texas Instruments avec ARM Cortex A8 fonctionnant jusqu'à 1 GHz d'horloge [Col13] et 512 MB de mémoire SRAM intégrée. Le second est Microzed basé sur le SOC Zynq-7000 SOC incluant le Cortex-A9 ARM à deux coeurs capable d'atteindre 667 MHz [Raj+11] et il possède une mémoire SRAM DDR3 de 1 Go. Les deux processeurs sont de la série ARM Cortex et ont les mêmes exécutions de mode ARMv7 Instruction Set Architecture (ISA) qui sont utilisées de manière intensive pour les grands appareils mobiles tels que les tablettes et les smartphones. Le système d'exploitation embarqué fonctionne sur les deux plates-formes est la version GNU/Linux, qui démarre à partir de la mémoire Flash-NAND pour deux cas et ses utilisations pour ces avantages pour les développeurs logiciels. La compilation est effectuée sur l'outil de synthèse Xilinx Software Development Kit (SDK) Eclipse-based IDE. Le logiciel de référence HM est écrit en langage C++ et compilé pour les deux architectures ARM. La chaîne d'outils G+++ fournie par Xilinx est utilisée, et l'optimisation de vectorisation NEON est automatiquement activée dans le jeu de niveaux -O3. Nous avons sélectionné une seule séquence parmi trois classes différentes E (1280 × 720) avec la séquence Fourpeople, C (832 × 480) avec PartyScence et D (416 × 240) avec BasketballPass, car les résultats obtenus après plusieurs tests sont presque les mêmes. Les deux valeurs sélectionnées de QP parmi les quatre valeurs proposées par la norme dans [Bo11], qui sont 27 et 32. Et nous avons utilisé le profil principal et la configuration à accès aléatoire. Toutes ces caractéristiques et encore plus d'informations sont déclarées dans le fichier de configuration JCT-VC documentent les conditions de test communes dans [Bos+12]. L'évaluation expérimentale de cette mise en oeuvre de la comparaison s'appuie sur le logiciel de référence HM version 15.0 [Noah]. Tous les tests ont été effectués en utilisant les 200 premières trames pour chaque séquence où sont stockés sur carte SD pour les deux cas. Tous les résultats des tests sont la moyenne de 20 fois le nombre d'exécutions. Comme le montre la figure **??**, les résultats pour deux valeurs de la PQ (27 et 32) et en utilisant uniquement un accès aléatoire. Les valeurs présentées dans chaque colonne sont la différence entre les résultats avant et après l'optimisation NEON appliquée. Les résultats de la comparaison sont présentés pour trois séquences décrites ci-dessus et sont regroupés dans deux cas renvoyés aux deux processeurs ARM cortex a8 et A9. Généralement, les résultats montrent une petite différence, on peut dire qu'ils sont presque identiques dans les deux cas si on se base sur les valeurs obtenues. La plupart du temps, ces comparaisons ont donné des résultats similaires mais non identiques en raison de ces processeurs cortex-A8 est le prédécesseur du cortex A9 d'une part, et de divers environnements de test.

Dans le but d'améliorer les résultats obtenus en monocoeur en n'utilisant qu'une optimisation partielle NEON par par parallélisme des données applicatives au niveau de chaque



(a) QP=32



(b) QP=27

FIGURE 3.3 – La comparaison de l'implémentation de l'accélération de la technologie NEON entre ARM cortex-A8 et cortex-A9 pour des valeurs QP égales à 27 et 32.

module après une étude détaillée de l'ensemble du codeur HEVC, les travaux suivants seront réalisés. Et nous allons concentrer notre travail sur le processeur Zynq pense seulement à cette architecture, qui a deux processeurs physiques ARM cortex a9. Cela peut permettre d'optimiser le parallélisme au niveau du fil parce que le HEVC est facile à exécuter sur des multicoeurs. Le Zynq a également intégré un FPGA derrière le duel de ces cœurs qui peuvent être offerts une co-conception entre un logiciel et du matériel. Les résultats expérimentaux de la comparaison d'implémentation proposée du codeur HEVC sur les deux processeurs basés sur l'optimisation au néon indiquent que le temps d'exécution a été en moyenne 4 fois plus rapide que le temps total d'encodage sans optimisation, alors que la qualité est la même. En outre, ces résultats ont montré que l'accélération obtenue est pratiquement la même dans toutes les résolutions des différentes séquences d'essai.

3.4.1 CONCLUSION

Dans cette section, nous avons présenté une comparaison d'implémentation entre deux logiciels d'exécution de codeur HEVC dans deux différents processeurs embarqués cortex de bras A8 et A9 utilisant la technologie Neon. Ces résultats montrent que l'accélération donnée dans les deux cas est presque identique et jusqu'à 4 fois. Dans les travaux futurs, nous pourrons améliorer ces résultats en utilisant des pièces matérielles sortant de Zynq d'une part, et d'autre part, nous nous concentrerons sur le parallélisme sur le multicoeur de faible puissance, afin de répondre aux besoins de traitement en temps réel pour les hautes résolutions comme Ultra HD et 4K. afin de réduire la complexité et de réaliser le traitement en temps réel, nous étudions les caractéristiques de chaque bloc HEVC, et la manière la plus appropriée d'exploiter toutes les ressources disponibles sur l'appareil telles que le parallélisme sur le multicoeur. Nous avons également proposé en hardware le bloc du HEVC afin d'exploiter le FPGA déjà embarqué avec processeur ARM sur puce unique.

3.5 L'implémentation de Décodeur sur ARM

3.5.1 Performances de l'exécution

Dans un premier temps, l'exécution du décodeur logiciel HM sur le seul processeur ARM (cortex A8 et A9) est effectuée, ce qui permet de déterminer le nombre de décodage de trames par seconde. Dans ce travail, nous nous intéressons à la comparaison de l'accélération de l'entrée dans le noyau unique, car c'est la première comparaison de mise en oeuvre rapportée dans la littérature. L'optimisation SIMD est prise en compte lors de l'exécution du logiciel et la mesure est indiquée en images par seconde (fps). C'est pourquoi, nous avons mesuré le temps de chaque exécution une première fois, puis nous avons divisé ce temps par le nombre d'images contenues dans chaque séquence indiquée dans le Tableau 4 ci-dessus. Enfin, les résultats sont présentés dans le Tableau **??**.

Afin de calculer le nombre de trames décodées par seconde sur les deux processeurs qui peuvent être obtenues sur ces plates-formes lorsque la technologie Neon est utilisée pour toutes les séquences. Cependant, les quatre valeurs 22, 27, 32, 37 du paramètre Quantification sont améliorées. dans le Tableau 12,. Notez que ces performances n'incluent pas le temps nécessaire au chargement et au stockage des données. Les résultats obtenus sont présentés dans le Tableau 9 qui représente la moyenne de 20 temps d'exécution. Le résultat le plus élevé est pour la séquence BasketballPass, mais encore loin des exigences en temps réel (25 images par seconde). Le nombre d'images décodées par le cortex-A8 est supérieur à celui traité par le cortex-A9. En effet, la fréquence d'horloge peut être augmentée jusqu'à 1 Ghz en cortex-A8 contrairement à 667 Mhz en cortex-A9. De plus, la différence entre ces deux processeurs est encore faible si l'on tient compte de la consommation d'énergie.

Le rapport de temps entre le temps d'exécution avec et sans optimisation du néon sur les deux processeurs cortex-A9 et cortex-A8 est présenté dans le Tableau **??**, qu'il a appliqué

TABLE 3.7 – FRÉQUENCE DE TRAME OU CODAGE DE TRAME PAR SECONDE (FPS) POUR CHAQUE CLASSE.

			QP			
config	**Séq**	**processeur**	***22***	***27***	***32***	***37***
Intra	**BQT**	***Cortex-A8***	0.40	0.52	0.61	0.69
		Cortex-A9	0,36	0,46	0,56	0,65
	BM	***Cortex-A8***	2.15	2.78	2.45	3.18
		Cortex-A9	1,93	2,12	2,51	2,79
	BP	***Cortex-A8***	7.56	8.45	9.48	10.59
		Cortex-A9	6,58	7,25	8,33	9,43
LD	**BQT**	***Cortex-A8***	0.53	0.87	1.11	1.28
		Cortex-A9	0,29	0,29	0,30	0,30
	BM	***Cortex-A8***	3.41	4.13	4.96	5.71
		Cortex-A9	3,26	3,85	4,76	5,26
	BP	***Cortex-A8***	9.61	10.97	12.71	14.43
		Cortex-A9	8,93	10,00	11,63	13,51
RA	**BQT**	***Cortex-A8***	0.59	0.93	1.12	1.20
		Cortex-A9	0,55	0,89	1,04	1,08
	BM	***Cortex-A8***	3.76	4.47	5.03	5.49
		Cortex-A9	3,66	4,17	4,84	5,13
	BP	***Cortex-A8***	10.19	11.64	13.14	14.43
		Cortex-A9	9,62	10,87	12,20	12,82

afin de calculer l'accélération obtenue lorsque l'optimisation du néon est utilisée pour les deux processeurs, cependant, Les résultats de l'accélération avaient 2,77 comme minimum et 4,22 comme maximum. Néanmoins, ces différences sont principalement dues d'une part à la vitesse de déplacement des séquences utilisées, et d'autre part aux caractéristiques différentes entre les deux processeurs proposés dans le Tableau 1. De manière générale, pour tous les cas, les résultats obtenus sur le cortex-A8 sont lents par rapport au cortex-A9, malgré une fréquence supérieure au cortex-A9, mais le cortex A8 a une architecture améliorée car il a annoncé après le cortex A8.

TABLE 3.8 – LA VITESSE ATTEINTE GRÂCE À LA TECHNOLOGIE DU NÉON.

			QP			
Config	**séquence**	**processeur**	***22***	***27***	***32***	***37***
Intra	**BQT**	***Cortex-A8***	3.61	3.76	3.74	3.71
		Cortex-A9	4,05	4,06	4,19	4,22
		Difference	**0,44**	**0,3**	**0,45**	**0,51**
	BM	***Cortex-A8***	3.66	4.16	3.26	3.72
		Cortex-A9	4,05	4,03	4,07	4,09
		Différence	**0,39**	**-0,13**	**0,81**	**0,37**
	BP	***Cortex-A8***	3.41	3.39	3.41	3.36
		Cortex-A9	3,78	3,64	3,78	3,64
		Différence	**0,37**	**0,25**	**0,37**	**0,28**
LD	**BQT**	***Cortex-A8***	3.64	3.62	3.64	3.65
		Cortex-A9	3,95	3,89	3,91	3,84
		Difference	**0,31**	**0,27**	**0,27**	**0,19**
	BM	***Cortex-A8***	3.38	3.31	3.27	3.20
		Cortex-A9	3,71	3,51	3,68	3,46
		Difference	**0,33**	**0,2**	**0,41**	**0,26**
	BP	***Cortex-A8***	3.03	2.94	2.85	2.77
		Cortex-A9	3,25	3,16	3,07	3,05
		Difference	**0,22**	**0,22**	**0,22**	**0,28**
RA	**BQT**	***Cortex-A8***	3.59	3.60	3.72	3.76
		Cortex-A9	3,85	3,85	3,90	3,80
		Difference	**0,26**	**0,25**	**0,18**	**0,04**
	BM	***Cortex-A8***	3.35	3.36	3.36	3.34
		Cortex-A9	3,66	3,53	3,63	3,56
		Différence	**0,31**	**0,17**	**0,27**	**0,22**
	BP	***Cortex-A8***	2.96	2.91	2.86	2.83
		Cortex-A9	3,21	3,20	3,02	2,85
		Différence	**0,25**	**0,29**	**0,16**	**0,02**

La compilation est effectuée sur l'outil TI Code Composer Studio Software Development Kit (SDK). Le logiciel de référence HM est écrit en langage C++ et compilé pour les deux architectures ARM. La chaîne d'outils G+++ fournie par TI est utilisée, et l'optimisation de la vectorisation NEON est automatiquement activée dans le jeu de niveaux -O3. Cette comparaison doit tenir compte de l'avantage des instructions SIMD qui convient dans le processus

d'encodage vidéo pour ce type de processus en raison d'un grand avantage en terme de gain de temps. Le jeu d'extension de l'architecture SIMD est appelé NEON pour les processeurs ARM Cortex-Ax de cette catégorie. Le NEON utilise un registre très long allant jusqu'à 64 bits / 128 bits. Et il est basé sur deux types de registres Q et D. Nous n'avons sélectionné que quatre séquences de la classe C (832×480), car les résultats obtenus après plusieurs tests sont presque les mêmes pour une autre classe. Les deux valeurs sélectionnées de QP parmi les quatre valeurs proposées par la norme dans [Noac], qui sont 27 et 32. Et nous avons utilisé le profil principal et la configuration à accès aléatoire. Toutes ces caractéristiques et encore plus d'informations sont déclarées dans le fichier de configuration JCT-VC documentent les conditions de test communes dans [Noac]. L'évaluation expérimentale de cette mise en oeuvre de la comparaison s'appuie sur le logiciel de référence HM version 15.0 [Noam]. Tous les tests ont été effectués en utilisant les 150 premières trames pour chaque séquence où sont stockés sur carte SD pour les deux cas. Tous les résultats des tests sont la moyenne de 20 fois le nombre d'exécutions. Tous les tests de performance pour l'architecture x86 sont exécutés sur un processeur Intel Coeur(TM) i3-370M cadencé à 2,4 GHz avec 4 de mémoire RAM DDR3 et il a Ubuntu version 14.04 comme système d'exploitation.

3.5.2 Résultats du profilage

L'analyse de complexité est limitée sur le cortex A8 uniquement, parce que les résultats sur les deux processeurs sont presque identiques, par conséquent, le profilage de chaque fonction à l'intérieur du décodeur logiciel HEVC sur les deux plates-formes reçoit une valeur avec quelques différences qui ne peuvent pas avoir une influence sur les résultats finals. Le tableau xx montre les résultats de profilage de tous les blocs du décodeur logiciel HM sur les cortex ARM A8 et A9. Tout d'abord, la fonction analyse NAL analyse le flux d'entrée. Ensuite, le décodage d'entropie traite les résultats de la fonction NAL via la fonction TDecEntropy incluse dans TDecTop. Ensuite, la fonction TComTrQuant a appelé au décodage IT (transformée inverse) et IQ (quantification inverse) en utilisant successivement les deux fonctions InvTransformNxN et setQPforQuant. Après la prédiction Intra et Inter calculée par la fonction TDecCU, pour l'Intra la fonction xRecontra est utilisée pour traiter le composant Luma (par la fonction xIntraLumaReQT) et Chroma (par la fonction xIntraReChromaBLK), pour l'inter, la fonction xReconInter est appelée pour décoder la fonction motionCompensation et la texture par fonction xDecodeInterTexture. Enfin, la fonction TDecTop contient deux fonctions pour décoder le filtre, la fonction Tcomloopfilter for Debloking Filter et la fonction adaptiveOffset pour SAO.

L'étude de complexité du profilage sur le cortex ARM-A8 est calculée comme suit : la complexité de chaque bloc référencé dans le schéma du décodeur HEVC est présentée par un certain nombre de fonctions, cependant, le temps de calcul est mesuré à partir de toutes les fonctions chargées pendant le fonctionnement du décodeur, puis il est regroupé pour que le temps des fonctions consacrées à chaque bloc soit présenté, afin de faire le profil par blocs et non pas à plusieurs.

les résultats du profilage sont présentés sur le Tableau **??**, l'exécution se fait avec l'optimi-

sation Néon, et la simulation est limitée à trois séquences pour chaque classe où est BQTerrace, BQMall et BasketballPass, car les résultats des autres séquences donnent presque les mêmes valeurs. également, deux valeurs moyennes de QP 27 et 32 sont sélectionnées, car elles ont été adoptées dans la majorité des travaux associés. en raison du profilage de chaque bloc décodeur, les blocs plus complexes restent les mêmes pour chaque valeur de QP et pour les trois classes (B, C et D). pour TOUS les modes Intra (intra), les blocs QT et INTRA sont plus complexes, mais en mode low-delay (LD) et Random Access (RA) les blocs QT et INTER sont plus longs à réaliser.

TABLE 3.9 – Résultats du profilage du décodeur avec la technologie Neon

séq	config	QP	EP	QT	INTRA	INTER	DF	SAO
BQT	INTRA	27	17,9	20,02	42,66	0	13,3	6,11
		32	14,07	17,79	45,7	0	15,69	6,76
	LD	27	15,14	9,15	0,59	57,63	10,87	6,62
		32	11,32	6,36	0,59	67,12	10,48	4,12
	RA	27	13,96	6,98	2,63	60,49	9,86	6,08
		32	10,58	4,88	2,4	70,52	9,22	2,4
BM	INTRA	27	16,74	20,1	43,57	0	13,04	6,19
		32	15,25	21,78	47,16	0	15,81	7,02
	LD	27	18,57	12,05	3,41	46,77	12,64	6,57
		32	16,23	10,45	3,27	52,46	13,56	4,04
	RA	27	15,19	8,06	5,79	59,91	11,05	3,5
		32	14,05	7,04	5,42	59,42	11,77	2,3
BP	INTRA	27	19,23	23,21	40,64	0	11,35	5,57
		32	14,62	24,18	41,6	0	13,36	6,21
	LD	27	21,23	14,37	7,37	37,19	12,91	6,93
		32	18,71	12,82	6,55	41,8	14,1	6,02
	RA	27	19,17	11,64	10,94	40,9	12,58	4,77
		32	16,67	9,93	8,99	48,08	13	3,32

Comme discussion, ce travail sur le décodeur logiciel HM sur ARM donne un aperçu du temps d'exécution sur deux générations différentes de processeurs. La version Cortex A9 est plus performante que Cortex A8, sachant que les deux plates-formes ont démarré le Linux embarqué en tant que système d'exploitation. Néanmoins, le profilage du décodeur a montré que les fonctions plus d'exécution sont transformées et intra environ 46% pour la configuration Intra Only, et pour les deux configurations Random Access et Low Delay, le bloc ont plus de temps est l'inter prédiction que jusqu'à environ 42% pour la plupart des cas. Par conséquent, dans les travaux futurs, nous nous concentrerons sur ces blocs plus consommer le temps afin d'accélérer le traitement.

3.5.3 Conclusion

Le but de cette section est d'implémenter le décodeur HEVC sur les processeurs embarqués ARM avec une comparaison, une approche est définie pour exécuter le logiciel HEVC pour décodeur sur les deux familles de versions ARM cortex-A8 et cortex-A9 avec optimisation NEON qui peut être accélérée de 4 fois dans la plupart des cas. De plus les résultats peuvent démontrer que le cortex a9 est meilleur que cortex a8, Hawevor, les résultats sont améliorables avec une stratégie pour exploiter toutes les autres coeurs afin de maximiser la fréquence de trame par seconde. Le temps réel pour la haute résolution a pu être obtenu en se basant sur l'implémentation sur des multicoeurs (huit coeurs) et en utilisant l'optimisation récente de l'IDMA comme QDMA et EDMA, qui sont intégrés dans cette puce qui peut être utilisée dans le plus grand échange de données en mémoire. Dans le travail suivant, nous nous basons sur les résultats de thèse présentés ci-dessus afin d'exécuter le décodeur HEVC sur un processeur ARM multicoeur qui peut être exécuté en temps réel pour la résolution supérieure comme 4k et 8K.

3.5.4 Comparaison avec d'autre travaux

Dans le tableau **??**, nous avons comparé nos travaux d'implémentation du codeur sur DSP et aussi sur ARM avec d'autres travaux existants dans la littérature sur le Tableau 15. Nous donnons une comparaison de profilage avec le travail [Van+12], qu'il a réalisé sur le CPU.

TABLE 3.10 – comparaison de la répartition du temps entre les blocs de HEVC.

Config	**Travaux**	Intra	Inter	TQ	DBK	SAO	EP	Other
AI	notre (DSP)	11,2	0	51,44	0,06	0,19	8,01	26,69
	notre (ARM)	45,1	0	22,1	1,2	0,6	17	14
	[Van+12]	35	0	40	0	0	10	16
LD	notre (DSP)	0,74	45,7	38,81	0,12	0,07	5,09	8,75
	Notre (ARM)	7	59,3	11	0,51	0,7	9,3	12,19
	[Van+12]	1	81	9	0	0	3	9
RA	notre (DSP)	0,81	39,46	43,92	0,01	0,05	5,91	9,09
	notre (ARM)	6,3	57,4	13,1	0,61	0,87	9,1	12,19
	[Van+12]	1	78	11	0	0	2	8

Pour toutes les configurations, les filtres sont négligeables, pour l'entropie de codage, nos résultats sont similaires à ceux présentés dans [Van+12], mais la QT est plus complexe et l'inter prédiction est moins complexe que les travaux rapportés dans [Van+12].

3.6 l'implementation de l'estimation du mouvement rapide sur DSP

3.6.1 les estimateurs rapides du mouvment

Le vecteur de mouvement pour chaque bock détermine par estimation de mouvement (EM) qui indique le mouvement entre le bloc courant et la correspondance trouvée. Le HEVC adopte deux méthodes pour trouver la meilleure correspondance. La première est la recherche complète (FS) où tous les taux de distorsion (RD) sont déterminés à choisir son minimum en sachant qu'optimal en performance, comprend un coût plus élevé, il est désactivé dans le logiciel de référence, la deuxième méthode est la recherche de zone de test (TZ search) qui est un algorithme de recherche rapide avec intègre pixel, il est activé et mis en oeuvre dans le logiciel de référence HM. Bien que la recherche TZ soit très rapide que FS avec une bonne performance.

Les CU inter incluent les blocs de codage équivalents qui sont partitionnés en un ou plusieurs blocs de prédiction, la figure 3.4 donne tous les types de partitionnement disponibles pour inter, ce partitionnement se réfère au partitionnement asymétrique de mouvement (AMP). Les plus grandes tailles de blocs utilisées pour la prédiction à compensation de mouvement sont 64×64, et les plus petites sont 8×4 et 4×8.

Le HEVC améliore l'inter-codage par une nouvelle interpolation : interpolation d'échantillons fractionnaires des images de référence avec la précision d'un quart de pixel, en standard H.264/AVC, six prises pour le filtrage Luma des positions de demi-points, interpolation linéaire des positions de quarts de pixel et calcul des échantillons Chroma par interpolation de quatre échantillons pel entiers voisins. Bien que le HEVC utilise des filtres à sept et huit points pour l'interpolation Luma pour les positions des demi-points et des quarts de points, il en va de même. Les échantillons Chroma sont calculés à l'aide de filtres à quatre points d'écoute. L'interconnexion des trois algorithmes (SDSP, HDS et LDPS) est qu'ils sont tous basés sur le modèle de recherche des petits diamants. Le LDPS et le HDS ont été modifiés afin de se conformer à la nature de la vidéo, améliorant ainsi leur efficacité. La figure 3.4 illustre le schéma d'algorithme ME proposé.

La méthode proposée consiste à améliorer l'algorithme de recherche TZ. En fait, la prédiction et les étapes initiales de recherche par grille ont été conservées telles quelles. Les modifications proposées commencent à partir de l'étape de recherche matricielle. En fait, cette étape a été supprimée parce que la valeur de la distance du point minimum global n'est plus pertinente, il n'est pas nécessaire de faire un autre balayage puisque l'étape de prédiction est maintenant effective et donne une idée assez précise de la position finale du meilleur point. Par conséquent, nous passons directement à l'étape du raffinement. Cette étape sera effectuée par nos trois algorithmes différents à savoir Line Diamond Pattern Search (LDPS) [Wer+10], Small Diamond Search Pattern [CJJ03] (SDSP) ou Horizontal Diamond Search (HDS) [Sam+06] au lieu d'une simple recherche par losange ou par carré. En fait, en adaptant l'un des trois modèles de recherche proposés, nous pouvons surmonter l'effet d'omettre

Chapitre 3

Implémentation software du codeur HEVC

Sommaire

3.1 Introduction 51
3.2 Implémentation 53
3.3 L'implémentation du codeur sur DSP 54
3.3.1 L'exécution de l'encodeur 54
3.3.2 L'exécution du décodeur 55
3.3.3 Performances de l'exécution 58
3.3.4 Profilage des résultats avec analyse 58
3.3.5 Comparaison avec d'autres travaux 60
3.4 Conclusion 61
3.4.1 CONCLUSION 64
3.5 L'implémentation de Décodeur sur ARM 64
3.5.1 Performances de l'exécution 64
3.5.2 Résultats du profilage 67
3.5.3 Conclusion 69
3.5.4 Comparaison avec d'autre travaux 69
3.6 l'implementation de l'estimation du mouvement rapide sur DSP . . 70
3.6.1 les estimateurs rapides du mouvment 70
3.6.2 Diamond search 71
3.6.3 SDSP 71
3.6.4 HDS 72
3.6.5 LDPS 73
3.7 Résumé 73

3.1 Introduction

Lorsque la première version logicielle du standard HEVC a été proposée, de nombreuses implémentations logicielles ont vu le jour pour offrir des analyses statistiques concernant le temps d'exécution, la consommation électrique et les analyses de complexité pour le codeur et le décodeur. Cependant, le décodeur est encore plus facile à étudier en raison de sa moindre

complexité par rapport au codeur. Par conséquent, on peut trouver de nombreux travaux se concentrant sur le décodeur dans la littérature scientifique. Cette étude se concentre sur l'implémentation logicielle de l'encodeur sur le processeur embarqué TMS320C6678. Il s'agit de prendre en considération le parallélisme au niveau instruction, de l'optimisation de l'architecture VLIW et le codage logiciel rapide proposé [Cha+01]. Ces implémentations logicielles nous permettent d'avoir une base de référence des évaluations auxquelles on peut généralement s'attendre sur le dernier processeur embarqué DSP de Texas Instruments.

Nous avons choisi d'implémenter également le codeur HEVC sur le DSP multi-cœur TMS320C6678 pour bénéficier d'une fréquence de traitement élevée et d'une architecture optimisée. Le DSP appartient à la dernière génération de DSP multi- cœur de Texas Instrument. Il s'agit d'un DSP à virgule fixe / flottante haute performance basé sur l'architecture multicœur KeyStone de TI. Cette architecture fournit une structure haute performance pour l'intégration des cœurs RISC (Reduced Instruction Set Computer) et DSP avec des co-processeurs et des E/S spécifiques aux applications. MulticœurNavigator, TeraNet HyperLink et Multi-cœur Shared Memory Controller fournissent une bande passante interne adéquate pour un accès non bloquant à tous les cœurs de traitement, coprocesseurs, périphériques et E/S.

Dans cet section, nous présentons une implémentation entièrement logicielle d'une dernière version du HEVC Test Module (HM.15.0) avec ses trois configurations intra, low-delay et random-access de l'encodeur HEVC sur les deux processeurs mobiles (processeur de signal numérique DSP et processeur ARM), qui atteint une bonne performance pour une séquence vidéo 480p (720 x 480 pixels) en utilisant un seul noyau fonctionnant à 1 GHz à une fréquence de fréquence des deux processeurs sans optimisation avec accès direct amélioré (EDMA) et le multi-cœur d'exécution. Ensuite, une implémentation d'un logiciel de comparaison dans deux processeurs basse consommation pour appareils mobiles est proposée. Afin d'améliorer cette implémentation puis la comparaison, en exploitant principalement l'optimisation offerte par ce type de processeur embarqué, qui est une extension de l'architecture NEON pour ARM qui est un cas spécifique de parallélisme de données au niveau des données SIMD (Single Instruction Multiple Data). Et VLIW pour DSP. L'étude de la complexité de cette implémentation sur les processeurs TMS320C6678 et ARM conduit à l"exécution du codeur HEVC en temps réel pour une haute résolution au moyen de diverses optimisations telles que EDMA fortement recommandées sur la base de cette analyse de complexité, ainsi que la parallélisassions sur huit cœurs existants dans cette plate-forme DSP ou l'exploitation des autres plateformes dont les processeurs peuvent atteindre 64cœurs.

Dans la littérature, nous pouvons trouver de nombreuses façons d'optimiser le standard des codeurs HEVC afin d'augmenter le décodage de trames par seconde, ce qui est un objectif similaire de notre travail dans les appareils à faible consommation d'énergie. Une façon de minimiser le temps de traitement sur le processeur mobile ARM est de tirer profit de la technologie NEON, connue sous le nom de SIMD. En fait, actuellement, les processeurs les plus disponibles sur le marché incluent les extensions de jeux d'instructions SIMD équivalents. Ce traitement de données multiples dans les mêmes opérations a commencé avec la plate-forme Intel x86 et a été nommé MMX. Depuis 1995, les instructions SIMD ont été utilisées dans un premier travail avec le vidéo standard MPEG-1 afin d'obtenir le traitement en temps

réel pour la basse résolution 352×288 pixels, par la suite, SIMD ont été appliquées dans plusieurs versions de décodeurs vidéo comme H.263, H.264. Dans [Yan+12] Yan et al, ont implémenté le HEVC sur des processeurs x86 et exploité la technique SIMD en tant que fonctions arithmétiques des Extensions 2 (SSE2) de l'architecture x86 pour paralléliser chaque bloc du décodeur afin d'augmenter le décodage de trame par seconde, que leurs performances sont 5 fois supérieures pour une résolution FHD. Dans [Chi+15] Chi et al présente le SIMD optimisation pour HEVC décodeur sur plusieurs architectures mobiles et également sur les plateformes PC. Leur plus haute performance a atteint plus de 11 fois pour le bloc Sample-Adaptive Offset sur architecture x86 en utilisant les Extensions Vectorielles Avancées 2.

3.2 Implémentation

Dans ce travail, nous utilisons le logiciel de référence HM.15.0 [TMS00] proposé par la norme HEVC. La version du logiciel HM est codée en langage de programmation C++, nous l'avons développée sur Code Composer Studio (CCS) qui est un environnement de développement de Texas Instruments où l'on peut trouver ce DSP intégré avec d'autres familles de processeurs intégrés. Ainsi, nous profitons de cet environnement pour bien adopter le logiciel sur les propriétés de cette version de DSP telles que les déclarations de variables et les configurations de mémoires etc. aussi, nous adoptons le logiciel pour l'implémentation ARM.

Pour la simulation, nous utilisons quatre séquences de classe C (832×480 pixels) qui sont BasketballDrill (BSK), PartyScence (PTS), BQMall (BQM), et RaceHorese (RAC) présentées dans le tableau 4.8 avec leurs caractéristiques comme la résolution, la profondeur de bit, l'image par seconde (fps) et le nombre de vues, que le document JCT-VC a choisi des conditions communes aux essais dans [Noaf]. D'autres configurations sont les paramètres de quantification (QP) avec les valeurs 22, 27, 32 et 37. Les trois configurations utilisées sont Intra (AI), Low delay (LD) et Random Access (RA), la première étant basée sur des images intra uniquement et sans référence temporelle, c'est-à-dire sans compensation de mouvement. Le deuxième faible retard n'a que la première image d'une séquence vidéo qui est codée en tant que prédiction Intra et les autres images sont codées en prédiction P ou B. Cette configuration est surtout utilisée sur les vidéos dédiées à la communication en temps réel. La troisième accès aléatoire (Random Access) est un premier et chaque image est codée à intervalles d'environ une seconde dans la prédiction Intra qui est généralement utilisée dans la vidéo de divertissement diffusée.

Table 3.1 – Quatre séquences de test sont utilisées pendant l'expérimentation.

Name	resolution	Bit depth	Fps	Frame count
BasketballDrill	832 × 480	8	50	150
PartyScence	832 × 480	8	50	150
BQMall	832 × 480	8	60	150
RaceHorese	832 × 480	8	30	150

3.3 L'implémentation du codeur sur DSP

3.3.1 L'exécution de l'encodeur

Le codeur HEVC est implémenté sur un seul noyau DSP c66x, en fait, nous n'implémentons pas le codeur dans les huit noyaux car c'est la première implémentation du codeur HEVC sur DSP dans la littérature. Notre travail s'intéresse aux statistiques sur un seul noyau, de plus, si nous utilisons le multi-cœur, la communication entre les noyaux peut augmenter le temps d'exécution. Le premier travail consiste à exécuter l'encodeur sur un DSP simple cœur pour déterminer le temps nécessaire pour les 150 premières trames de chaque séquence, le temps exprimé (en seconde) est donné dans le tableau 4.10 pour les différentes séquences choisies.

TABLE 3.2 – Le temps d'encodage obtenu pour les 150 images de quatre séquences qui sont choisies pour les tests en secondes.

config	**intra**				**lowdelay_P**				**randomacces**			
Séq	**22**	**27**	**32**	**37**	**22**	**27**	**32**	**37**	**22**	**27**	**32**	**37**
BDrill	11385	8905	7802	7277	26454	22471	20949	18 901	26475	20401	16600	14777
PScence	14842	11949	10138	9058	31781	30775	23933	20170	25092	23717	18991	15302
BQMall	11661	9181	8499	7603	34271	25412	20918	18888	23028	19929	16581	14728
RHorese	12187	9821	8518	7828	34502	37209	29139	23779	30483	27507	22507	18444

Les remarques qui reprennent les résultats du tableau 4.10 ci-dessus sont les suivantes : premièrement, si QP augmente le temps d'exécution augmente aussi, deuxièmement, la configuration intra est plus rapide en suivant RA puis LD, troisièmement, les résultats de cette simulation ne sont pas en temps réel. Enfin, les différences entre les résultats de ces séquences sont dues aux caractéristiques de mouvement et de texture dans chaque séquence et dans chaque scène. Dans cette section, nous étudions la complexité de chaque bloc à partir du schéma de l'encodeur donné en Figure 2.4. Pour cela, nous limitons notre travail à deux valeurs moyennes de QP 27 et 32, pour être plus précis, nous calculons le temps nécessaire pour chaque méthode lors de l'exécution. Le tableau 4.10 donne un exemple pour les méthodes (fonctions) plus complexes appliquées sur la séquence BasketballDrill avec QP égale à 32. Nous utilisons 150 premières trames pour le profilage du codeur et toutes les exécutions sont sans optimisation. Dans le tableau 4.10, la méthode xt_xquant est plus complexe ; les complexités ne sont pas dans son algorithme, mais elle avait beaucoup de fonctions déduites pour la copie comme mem_copy, dans l'ensemble, DSP est connu par lentement pour copier les données entre la mémoire contre elle est plus rapide dans le traitement des données. A part cela, pour la prédiction intra configuration predIntraLumaAng qui est dédiée au mode de prédiction intra Luma. Pour les configurations RA et LD, l'estimation des mouvements est plus complexe et elle est déduite pour le calcul de l'estimation du mouvement.

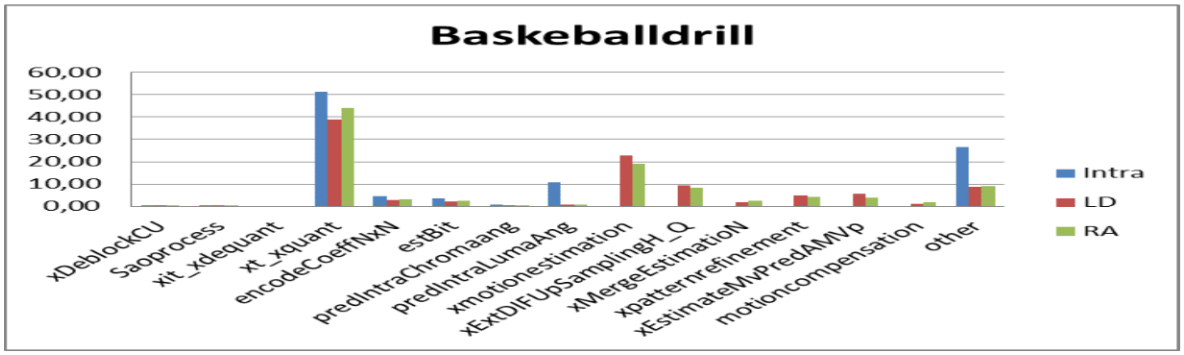
Dans la figure 3.1, nous donnons le résultat de quatre séquences avec ces trois configurations (AI, RA et LD), où il y a une comparaison individuelle des variations de ces méthodes

(fonctions).

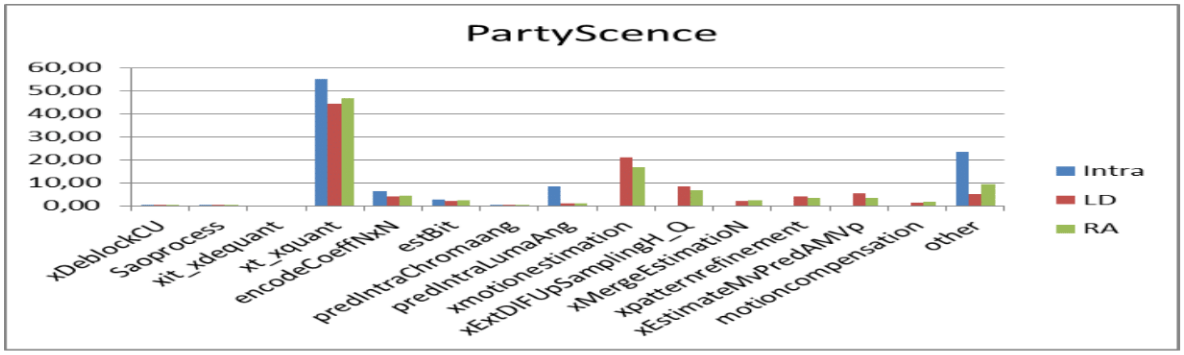
Le tableau 3.3 montre les pourcentages du temps de calcul de toutes les méthodes dans ces blocs, pour trois configurations, intra, LD et RA avec deux valeurs de QP, 27 et 32. La transformation et la quantification prennent environ 50% du temps total pour ces traitements et spécifiquement pour les fonctions de copie. Cependant, pour les deux configurations, LD et RA, l'inter prédiction est plus complexe et prend environ 21,2% à 46 %. Par rapport à la configuration intra avait la prédiction intra bloc plus complexe, qu'il faut autour de 10,11% à 11,95%. Pour les trois configurations, nous avons constaté que le codage par entropie prend de 5,33 % à 10,46 %. Pour les codeurs HEVC, les deux filtres SAO et DF sont très négligeables par rapport aux autres blocs, ils ne dépassent pas 0,19%.

3.3.2 L'exécution du décodeur

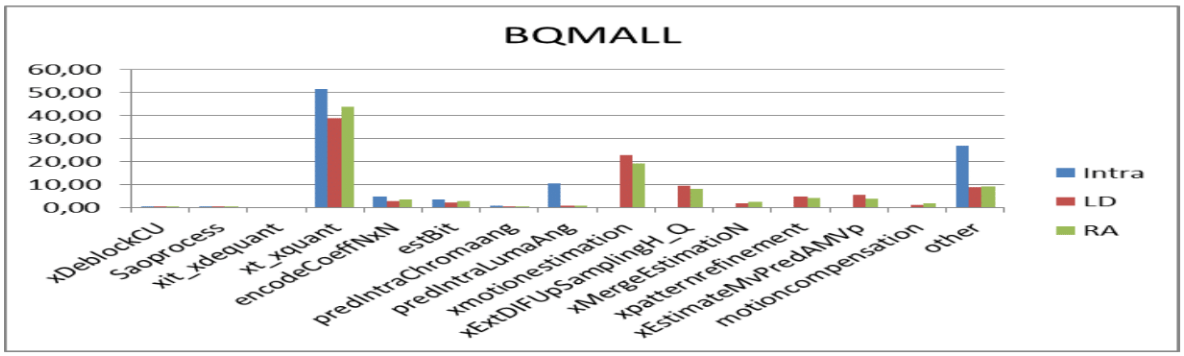
Dans notre simulation, nous avons sélectionné parmi la liste proposée par le standard dans le fichier [Bo11], les séquences utilisant dans les simulations suivantes données en bas, ces séquences se distinguent les unes des autres par leur mouvement haut ou bas et leur texture complexe ou lisse. Les caractéristiques de ces séquences sont les suivantes : la première est la résolution, qui est la somme des pixels contenus dans chaque image et présentée comme le nombre de pixels en longueur multiplié par le nombre de pixels en largeur. La second est le nombre d'images, qui est le nombre d'images dans l'ensemble de la séquence. La troisième est le nombre d'images à décoder en temps réel. La dernière est la profondeur de bit, qui est le bit pris par chaque pixel. De plus, la norme limite la valeur de la PQ à quatre valeurs comme suit : 22, 27, 32 et 37. En fait, les trois configurations utilisées pour les exécutions sont Intra, Low delay et Random access. La première est basée sur des images intra seulement et sans référence temporelle, c'est-à-dire sans algorithme de compensation de mouvement. La second est le retard minimal, qui revient à trouver la première image dans une séquence vidéo est décodée comme une prédiction Intra seulement et les autres images sont décodées en intra et/ou inter prédiction avec les deux configurations P et B prédiction. Cette configuration est surtout utilisée sur les vidéos dédiées à la communication en temps réel. La troisième est l'accès aléatoire, qui est une première et chaque image environ un second intervalle sont décodés sur Intra prédiction et les autres images sont décodées en inter et/ou intra prédiction, qui est généralement utilisé dans la vidéo de divertissement diffusée. Toutes ces caractéristiques et encore plus d'informations sont mentionnées dans le fichier de configuration proposé par la norme [Bo11]. Afin de suivre une norme commune, nous avons choisi les configurations à partir des conditions d'essai communes des documents JCT-VC [Bo11]. Nous avons sélectionné des séquences des classes E (1280 × 720), C (832 × 480) et D (416×240), les séquences ont une profondeur de 8 bits et un sous-échantillonnage couleur 4 :2 :0 pour les composantes Luma et Chroma qui les nomme et leur numéro de trame comme suit : Classe E : KristenAndSara(600), Johnny(600), FourPeople(600). Classe C : BasketballDrill(500), PartyScence(500), BQMall(600), RaceHorese(300). Classe D : BlowBubbles(500), BQSquare (600), BasketballPass (500), RaceHorese (300).



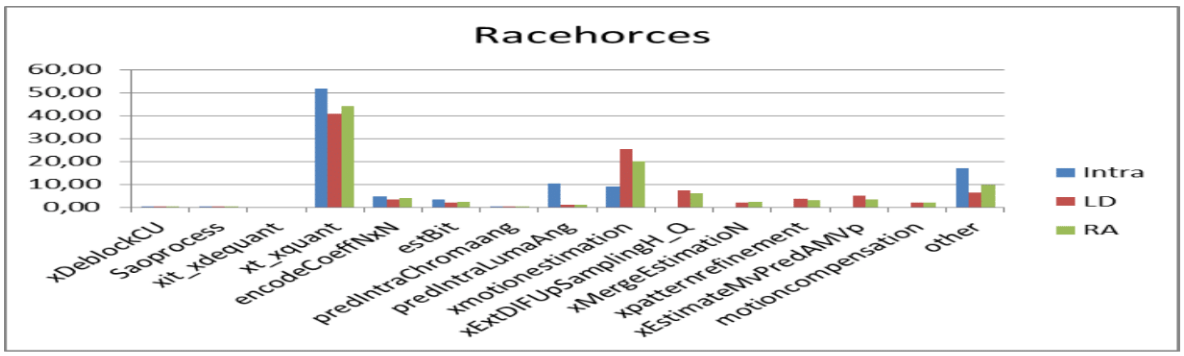
(a) BasketBalldrill.



(b) PartyScence.



(c) BQMall.



(d) RaceHorce.

FIGURE 3.1 – BasketBalldrill,(b) :PartyScence, (c) :BQMall, (d) :RaceHorce, la valeur de QP=32.

TABLE 3.3 – Le profilage du codeur HEVC en pourcentage de blocs

Seq	Cfg	QP	EP	QT	INTRA	INTER	DF	SAO	OT
FourBeaple	**AI**	27	8,36	23,94	37,41	0,00	12,95	4,06	13,28
		32	6,45	51,45	13,46	0,00	13,29	3,93	11,42
	LD	27	4,37	8,77	0,45	40,98	9,38	1,00	35,06
		32	3,29	6,02	0,00	46,57	7,97	0,44	43,78
	RA	27	3,49	5,03	1,92	50,39	6,66	0,94	32,38
		32	2,83	4,03	1,72	57,42	6,42	0,73	26,86
	AI	27	6,29	25,42	41,10	0,00	11,87	3,44	11,87
		32	4,83	26,48	41,23	0,00	12,38	3,30	11,78
Johanne	**LD**	27	2,70	4,74	0,12	50,45	6,52	0,41	35,10
		32	1,59	2,15	0,09	33,12	4,73	0,15	13,76
	RA	27	2,35	3,20	0,96	58,97	5,46	0,55	28,49
		32	1,95	2,40	0,90	59,55	5,06	0,34	30,59
BasketBallDrill	**AI**	27	12,39	20,27	34,44	0,00	14,41	4,66	13,83
		32	0,91	32,00	35,16	0,00	15,20	4,57	13,25
	LD	27	6,30	30,47	2,54	32,78	9,39	2,40	16,11
		32	6,85	22,48	2,28	35,62	10,49	1,49	20,80
	RA	27	5,76	16,77	4,50	44,17	8,93	1,74	18,13
		32	4,69	14,72	4,22	44,50	10,06	1,22	20,59
BQMall	**AI**	27	11,67	22,61	35,35	0,00	12,82	4,10	13,46
		32	9,33	24,34	34,93	0,00	14,31	3,98	13,12
	LD	27	7,11	12,47	1,45	47,15	9,65	2,38	19,79
		32	5,23	9,88	1,35	52,29	9,59	1,34	20,33
	RA	27	5,80	7,42	3,13	56,41	8,36	1,30	17,58
		32	4,56	6,38	2,92	57,29	8,76	0,91	19,16
BasketBallPass	**AI**	27	11,88	26,43	30,40	0,00	11,35	3,95	15,98
		32	9,05	28,34	30,56	0,00	12,79	3,94	15,31
	LD	27	9,65	14,98	3,79	34,41	10,54	3,15	23,49
		32	7,54	13,51	3,47	36,37	11,62	2,32	25,17
	RA	27	8,42	10,79	5,88	40,52	10,24	1,92	22,23
		32	5,94	9,35	5,32	43,93	10,66	1,32	23,48
BQSqaure	**AI**	27	17,00	16,55	32,20	0,00	11,11	3,03	20,11
		32	14,96	15,19	34,25	0,00	12,76	3,21	19,63
	LD	27	10,08	4,89	0,15	51,04	9,42	2,37	22,04
		32	6,46	3,22	0,19	56,40	8,81	1,11	23,81
	RA	27	7,17	4,81	1,29	56,84	7,05	1,62	21,22
		32	4,61	2,24	1,26	63,32	6,33	0,78	21,45

3.3.3 Performances de l'exécution

La première étape consiste à exécuter le décodeur logiciel HM sur le processeur DSP unique afin de calculer le nombre de trames décodées par seconde. Dans ce travail, nous nous concentrons sur l'implémentation en noyau unique parce que c'est la première implémentation rapportée dans la littérature dans cette dernière génération de DSP, et à partir de ces statistiques dans un noyau unique, nous pouvons faire une bonne remontée en temps réel en exploitant la ressource minimale dans ce navire. La première mesure est de déterminer le nombre d'images par seconde (fps) pour toutes les séquences sélectionnées en utilisant l'optimisation VLIW, pour cela on calcule le temps de chaque exécution, puis on le divise par le nombre d'images contenues dans chaque séquence, les résultats sont présentés au Tableau 3.3. Le résultat pour chaque valeur du Tableau 3.3 et des tableaux suivants est une moyenne de 60 exécutions.

Les résultats du Tableau 3.3 jusqu'à 25 fps pour la séquence BQSqure avec qp=37 en configuration LD. Pour toute expérience. Nous remarquons que lorsque la valeur de QP augmente, le nombre d'images décodées augmente également. De plus, les résultats de l'image décodée par seconde pour l'IA sont inférieurs à ceux de la RA et de la LD, en raison de l'utilisation de l'intra seulement. Pour RA et LD, nous voyons que ce sont à peu près les mêmes résultats avec une légère différence en raison du contenu des frames. Généralement, nos résultats sur un seul noyau indiquent qu'en utilisant seulement les autres noyaux parallèles, nous pouvons être en temps réel (30 images par seconde) pour deux classes C et D, si on exploitait l'EMDA pour diminuer encore plus le temps, on pourra atteindre le temps réel pour la classe E.

Dans le Tableau 3.4, nous avons limité la simulation à deux séquences de chaque classe, parce que nous obtenons presque les mêmes résultats dans chaque classe. Par conséquent, la performance obtenue par l'optimisation VLIW s'appliquait à chaque séquence de classe deux (elles sont connues sous le nom de mouvement haut et bas). Cependant, l'accélération est traitée en fonction du temps d'exécution avec et sans VLIW. Les résultats de l'optimisation se situent entre 3,10 et 8,94. De plus, ces différences entre les résultats dus aux trois configurations (AI, LD et RA), à la PQ et aux caractéristiques de contenu des séquences.

3.3.4 Profilage des résultats avec analyse

Dans cette section, nous présentons l'analyse des performances du profilage sur le DSP. La complexité de chaque bloc à l'intérieur du diagramme du décodeur HEVC est déterminée en incluant dans chaque méthode (fonction) du Tableau 3.4 une fonction spécifique pour calculer le temps de consommation en fonctionnement complet. Le pourcentage d'allocation de chaque méthode (fonctions) est calculé en fonction du temps total.

Le Tableau 3.4 présente les fonctions les plus complexes en utilisant la séquence BQMall avec valeur fixe QP en 32. Les résultats montrent que les fonctions les plus complexes pour trois configurations comme suit : pour l'IA les méthodes les plus complexes sont xReconIntraQT utilise pour la prédiction intra qui prend 28%, suivie par la fonction invtransformNxN

utilise pour traiter le bloc de transformation inverse et la fonction loopFilterPic du filtre en boucle 25% et 13,68% respectivement. Pour LD et RA les valeurs de profilage sont les mêmes, la fonction la plus complexe est la compensation de mouvement utilise pour calculer la compensation de mouvement que jusqu'à 37%. Ensuite, la fonction xGetNewPicBuffer a des valeurs importantes jusqu'à 20% que l'utilisation pour la prédiction inter et il a les fonctions pour copier ou échanger les données entre la mémoire. En outre, il est connu les fonctions utilisant la mémoire prend charge importante de DSP. A cet égard, l'optimisation par EMDA peut être aidée en évitant le transfert de données par DSP. Enfin, nous pouvons déduire de ces statistiques de ces analyses de performance. Les fonctions xReconIntraQT et invtransformNxN pour la configuration AI, et la fonction motioncompensation pour les configurations LD et RA peuvent être implémentées sur des multicœurs pour réduire sa complexité. La Figure B.2 montre un exemple de comparaison du pourcentage des trois configurations AI, LD et RA de chaque fonction présentées dans le Tableau 7 Le pourcentage des différentes classes de logiciels par rapport à la charge totale pour BQMall avec QP=32 en utilisant la séquence QBMall avec QP=32. On peut voir la variation de pourcentage entre eux.

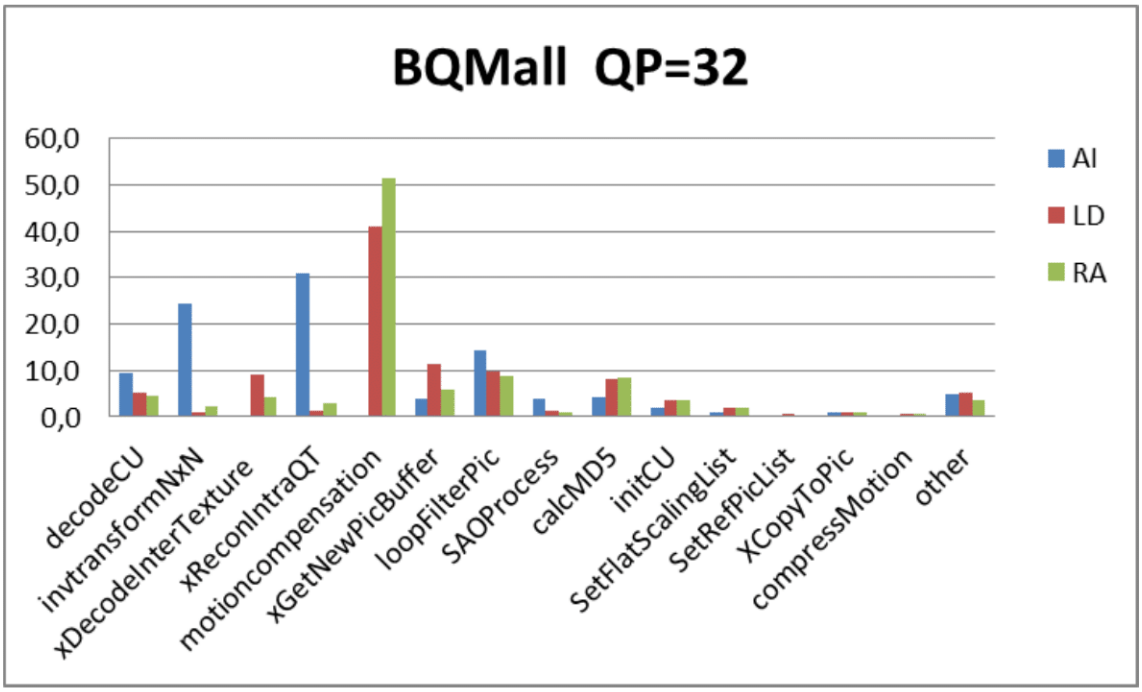


FIGURE 3.2 – Comparaison de la complexité des trois configurations AI, LD et RA pour la séquence QBMall avec QP=32.

Dans le Tableau **??**. Nous avons regroupé les fonctions dédiées à chaque bloc pour les utiliser dans les étapes suivantes comme le profilage par blocs présenté dans le schéma du décodeur de la figure B.1. Et aussi pour comparer nos résultats avec d'autres travaux.

Après avoir regroupé les fonctions, en donnant dans le Tableau 7 en bloc dans le Tableau 8, nous avons présenté le profilage de deux séquences pour chaque résolution pour la PQ égale à 27 et 32 dans le Tableau 3.4. Pour la configuration AI, nous pouvons voir qu'environ 30% du décodage chargé pour la prédiction bock intra se consume par la quantification inverse de

bloc et la transformation inverse d'environ 25%, par rapport aux configurations LD et RA, le bloc le plus complexe est la prédiction inter, qui est jusqu'à 63% et convient à l'optimisation du parallélisme, s'écoule par la quantification inverse de module et la transformation inverse avec le pourcentage environ 10%. Sauf que les autres blocs EP, DF et SAO ont un faible pourcentage et c'est un bloc différent les uns des autres selon les séquences et les paramètres des configurations. Enfin, le module TQ est adapté à l'optimisation EDMA car il dispose d'une fonction dédiée au fonctionnement entre la mémoire, telle que la copie de fonction qui permet d'obtenir un temps de décodage important.

TABLE 3.4 – CODING UNIT PARTITIONING OF PUS FOR EACH DIFFERENT CODING DEPTH LEVEL.

coding Depth level	**block partitioning**
0	64×64 ; 64×32 ; 32×64 ; 32×32 ; 16×64 ; 48×64 ; 64×16 ; 64×48
1	32×32 ; 32×16 ; 16×32 ; 16×16 ; 8×32 ; 24×32 ; 32×8 ; 32×24
2	16×16 ; 16×8 ; 8×16 ; 8×8 ; 4×16 ; 12×16 ; 16×4 ; 16×12
3	8×8 ; 8×4 ; 4×8

3.3.5 Comparaison avec d'autres travaux

Dans cette sous-section, les deux résultats dans la section ci-dessus, tels que la performance de l'exécution et le profil, qui sont comparés avec d'autres réalisations présentées dans un travail précédent. Parmi les travaux sur le DSP présentés dans [Pes+13], ils ont implémenté le logiciel HM.9.0 sur un DSP mono coeur, les résultats sont présentés dans le Tableau **??**.

TABLE 3.5 – Comparaison des performances de notre exécution avec d'autres sur DSP pour 832x480 (480p) requises par fps.

	configuration	IA		LD		RA	
	séquence	Min	max	Min	max	min	max
notre	480p	1.97	3.25	5	7.07	5.09	6.94
DSP[Pes+13]		1.4	1.9	2.8	4	2.7	3.6

Dans le Tableau **??**, nous comparons leurs séquences minimum (min) et maximum (max) afin de les appliquées de la résolution 480p. Ensuite, nous comparons avec d'autres profils des

modules décodeurs HEVC sur différents processeurs comme sur DSP [Pes+13], CPU[Vii+12], et sur processeur mobile intégré ARM [Bos+12].

TABLE 3.6 – Comparaison de la complexité de notre travail avec d'autres travaux de configuration RA.

QP	config	**EP**	**QT**	**INTRA**	**INTER**	**DF**	**SAO**	**OT**
27	notre	5,80	7,42	3,13	56,41	8,36	1,30	17,58
	[Bos+12]	4,8	3,7	1,8	52,7	13		24
	[Bos+12]	23	4	5	46	16	3	3
	[Pes+13]	14,5	10,7	11,9	29,2	13,9	4,8	15
32	notre	4,56	6,38	2,92	57,29	8,76	0,91	19,16
	[Bos+12]	3,5	3,8	1,5	55,1	13		22,9
	[Bos+12]	16	3	4	53	17	2	5
	[Pes+13]	11,1	10,7	12,7	32	14,8	4,7	14,1

Par rapport à ce travail, l'accélération d'obtenir jusqu'à 2 fois pour la configuration à faible délai, en outre, ils travaillent sur l'architecture DSP avec un seul noyau contre notre EVM avec multi-coeurs qui peuvent être exploités dans le travail suivant.

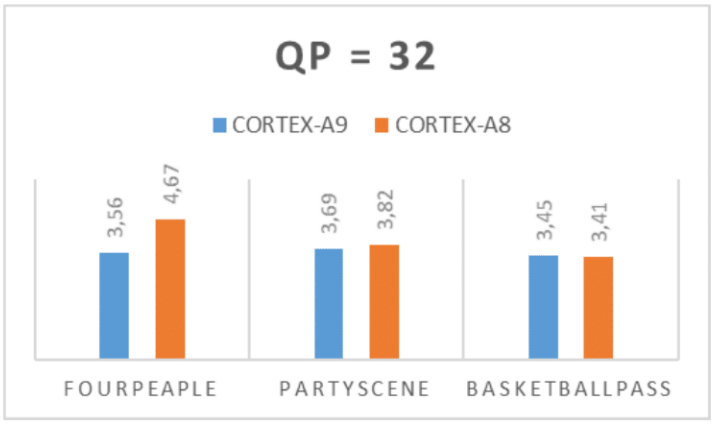
Dans le Tableau **??**, nous sélectionnons les travaux antérieurs utilisés la séquence QBMall, QP égal sont 27 et 32 et la configuration Random accès, qui est le plus utilisé leteratury, qui est généralement, nos résultats sont similaires à d'autres résultats signalés sur [Pes+13], [Vii+12] et [Bos+12], le bloc plus complexe pour nos et autres implémentations est permet inter prédiction présentée par compentation de mouvement avec valeur environ à 50%. Pour le bloc moins complexe, nous avons trouvé le filtre SAO pour toutes les architectures. Pour d'autres blocs, nous en avons trouvé peu différents, par exemple le QT fantaisie est plus complexe car notre travail contre le PE est un pourcentage élevé pour [Bos+12].

3.4 Conclusion

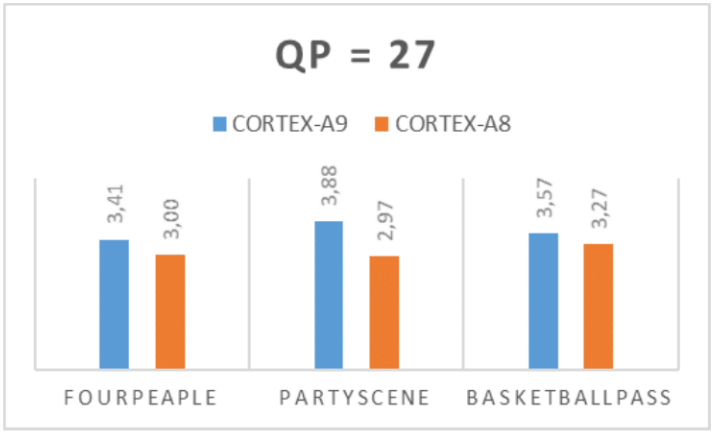
Le but de cette section est d'implémenter le décodeur HEVC sur le processeur embarqué DSP, une approche est définie pour construire le logiciel de décodage HEVC sur la dernière famille de version du DSP TMDSEVM6678, le résultat sur simple coeur donne une bonne performance par rapport aux mêmes autres implémentations sur processeur embarqué, en outre, la complexité de chaque bloc du décodeur est déterminée, en plus, la performance de l'optimisation avec VLIW est déterminée. Sur la base de ces résultats, nous pourrons définir une stratégie d'exploitation de tous les autres coeurs dans le but d'augmenter la fréquence d'images par seconde. Le temps réel pour la haute résolution a pu être obtenu en se basant sur l'implémentation sur plusieurs coeurs (huit coeurs) et en utilisant l'optimisation récente de l'IDMA comme QDMA et EDMA, qui sont intégrés dans cette puce qui peut être utilisée dans l'échange de données du plus grand mémoire. III.4. L'implémentation du codeur processeur embarqué ARM cortex Dans ce travail, une implémentation du dernier logiciel HM du

décodeur HEVC sur deux processeurs embarqués, ARM cortex-A8 et cortex-A9 avec technologie NEON qui accélère le temps de traitement à quatre fois est présentée. Ensuite, une analyse détaillée de la complexité est effectuée et détaillée, par conséquent, il sera utile tant pour la meilleure co-conception du décodeur sur la PFGA avec ARM qui existent sur un seul appareil, que pour un bon parallélisme sur les multicoeurs embarqués. III.5. L'implémentation d'encodeur sur ARM Il existe plusieurs travaux connexes dans la littérature sur l'accélération des codeurs HEVC utilisant l'optimisation à une seule instruction et à plusieurs données sur un processeur ARM et Intel x86 [Ahn+14],[Jun+16], [Bos+12]. les résultats de la comparaison sont réalisés sur deux plates-formes embarquées, la première est l'ordinateur monocarte BeagleBone Black à faible coût, équipé du SOC Sitara AM3358 de Texas Instruments avec ARM Cortex A8 fonctionnant jusqu'à 1 GHz d'horloge [Col13] et 512 MB de mémoire SRAM intégrée. Le second est Microzed basé sur le SOC Zynq-7000 SOC incluant le Cortex-A9 ARM à deux coeurs capable d'atteindre 667 MHz [Raj+11] et il possède une mémoire SRAM DDR3 de 1 Go. Les deux processeurs sont de la série ARM Cortex et ont les mêmes exécutions de mode ARMv7 Instruction Set Architecture (ISA) qui sont utilisées de manière intensive pour les grands appareils mobiles tels que les tablettes et les smartphones. Le système d'exploitation embarqué fonctionne sur les deux plates-formes est la version GNU/Linux, qui démarre à partir de la mémoire Flash-NAND pour deux cas et ses utilisations pour ces avantages pour les développeurs logiciels. La compilation est effectuée sur l'outil de synthèse Xilinx Software Development Kit (SDK) Eclipse-based IDE. Le logiciel de référence HM est écrit en langage C++ et compilé pour les deux architectures ARM. La chaîne d'outils G+++ fournie par Xilinx est utilisée, et l'optimisation de vectorisation NEON est automatiquement activée dans le jeu de niveaux -O3. Nous avons sélectionné une seule séquence parmi trois classes différentes E (1280 × 720) avec la séquence Fourpeople, C (832 × 480) avec PartyScence et D (416 × 240) avec BasketballPass, car les résultats obtenus après plusieurs tests sont presque les mêmes. Les deux valeurs sélectionnées de QP parmi les quatre valeurs proposées par la norme dans [Bo11], qui sont 27 et 32. Et nous avons utilisé le profil principal et la configuration à accès aléatoire. Toutes ces caractéristiques et encore plus d'informations sont déclarées dans le fichier de configuration JCT-VC documentent les conditions de test communes dans [Bos+12]. L'évaluation expérimentale de cette mise en oeuvre de la comparaison s'appuie sur le logiciel de référence HM version 15.0 [Noah]. Tous les tests ont été effectués en utilisant les 200 premières trames pour chaque séquence où sont stockés sur carte SD pour les deux cas. Tous les résultats des tests sont la moyenne de 20 fois le nombre d'exécutions. Comme le montre la figure **??**, les résultats pour deux valeurs de la PQ (27 et 32) et en utilisant uniquement un accès aléatoire. Les valeurs présentées dans chaque colonne sont la différence entre les résultats avant et après l'optimisation NEON appliquée. Les résultats de la comparaison sont présentés pour trois séquences décrites ci-dessus et sont regroupés dans deux cas renvoyés aux deux processeurs ARM cortex a8 et A9. Généralement, les résultats montrent une petite différence, on peut dire qu'ils sont presque identiques dans les deux cas si on se base sur les valeurs obtenues. La plupart du temps, ces comparaisons ont donné des résultats similaires mais non identiques en raison de ces processeurs cortex-A8 est le prédécesseur du cortex A9 d'une part, et de divers environnements de test.

Dans le but d'améliorer les résultats obtenus en monocoeur en n'utilisant qu'une optimisation partielle NEON par par parallélisme des données applicatives au niveau de chaque



(a) QP=32



(b) QP=27

FIGURE 3.3 – La comparaison de l'implémentation de l'accélération de la technologie NEON entre ARM cortex-A8 et cortex-A9 pour des valeurs QP égales à 27 et 32.

module après une étude détaillée de l'ensemble du codeur HEVC, les travaux suivants seront réalisés. Et nous allons concentrer notre travail sur le processeur Zynq pense seulement à cette architecture, qui a deux processeurs physiques ARM cortex a9. Cela peut permettre d'optimiser le parallélisme au niveau du fil parce que le HEVC est facile à exécuter sur des multicoeurs. Le Zynq a également intégré un FPGA derrière le duel de ces cœurs qui peuvent être offerts une co-conception entre un logiciel et du matériel. Les résultats expérimentaux de la comparaison d'implémentation proposée du codeur HEVC sur les deux processeurs basés sur l'optimisation au néon indiquent que le temps d'exécution a été en moyenne 4 fois plus rapide que le temps total d'encodage sans optimisation, alors que la qualité est la même. En outre, ces résultats ont montré que l'accélération obtenue est pratiquement la même dans toutes les résolutions des différentes séquences d'essai.

3.4.1 CONCLUSION

Dans cette section, nous avons présenté une comparaison d'implémentation entre deux logiciels d'exécution de codeur HEVC dans deux différents processeurs embarqués cortex de bras A8 et A9 utilisant la technologie Neon. Ces résultats montrent que l'accélération donnée dans les deux cas est presque identique et jusqu'à 4 fois. Dans les travaux futurs, nous pourrons améliorer ces résultats en utilisant des pièces matérielles sortant de Zynq d'une part, et d'autre part, nous nous concentrerons sur le parallélisme sur le multicoeur de faible puissance, afin de répondre aux besoins de traitement en temps réel pour les hautes résolutions comme Ultra HD et 4K. afin de réduire la complexité et de réaliser le traitement en temps réel, nous étudions les caractéristiques de chaque bloc HEVC, et la manière la plus appropriée d'exploiter toutes les ressources disponibles sur l'appareil telles que le parallélisme sur le multicoeur. Nous avons également proposé en hardware le bloc du HEVC afin d'exploiter le FPGA déjà embarqué avec processeur ARM sur puce unique.

3.5 L'implémentation de Décodeur sur ARM

3.5.1 Performances de l'exécution

Dans un premier temps, l'exécution du décodeur logiciel HM sur le seul processeur ARM (cortex A8 et A9) est effectuée, ce qui permet de déterminer le nombre de décodage de trames par seconde. Dans ce travail, nous nous intéressons à la comparaison de l'accélération de l'entrée dans le noyau unique, car c'est la première comparaison de mise en oeuvre rapportée dans la littérature. L'optimisation SIMD est prise en compte lors de l'exécution du logiciel et la mesure est indiquée en images par seconde (fps). C'est pourquoi, nous avons mesuré le temps de chaque exécution une première fois, puis nous avons divisé ce temps par le nombre d'images contenues dans chaque séquence indiquée dans le Tableau 4 ci-dessus. Enfin, les résultats sont présentés dans le Tableau **??**.

Afin de calculer le nombre de trames décodées par seconde sur les deux processeurs qui peuvent être obtenues sur ces plates-formes lorsque la technologie Neon est utilisée pour toutes les séquences. Cependant, les quatre valeurs 22, 27, 32, 37 du paramètre Quantification sont améliorées. dans le Tableau 12,. Notez que ces performances n'incluent pas le temps nécessaire au chargement et au stockage des données. Les résultats obtenus sont présentés dans le Tableau 9 qui représente la moyenne de 20 temps d'exécution. Le résultat le plus élevé est pour la séquence BasketballPass, mais encore loin des exigences en temps réel (25 images par seconde). Le nombre d'images décodées par le cortex-A8 est supérieur à celui traité par le cortex-A9. En effet, la fréquence d'horloge peut être augmentée jusqu'à 1 Ghz en cortex-A8 contrairement à 667 Mhz en cortex-A9. De plus, la différence entre ces deux processeurs est encore faible si l'on tient compte de la consommation d'énergie.

Le rapport de temps entre le temps d'exécution avec et sans optimisation du néon sur les deux processeurs cortex-A9 et cortex-A8 est présenté dans le Tableau **??**, qu'il a appliqué

TABLE 3.7 – FRÉQUENCE DE TRAME OU CODAGE DE TRAME PAR SECONDE (FPS) POUR CHAQUE CLASSE.

			QP			
config	**Séq**	**processeur**	***22***	***27***	***32***	***37***
Intra	**BQT**	***Cortex-A8***	0.40	0.52	0.61	0.69
		Cortex-A9	0,36	0,46	0,56	0,65
	BM	***Cortex-A8***	2.15	2.78	2.45	3.18
		Cortex-A9	1,93	2,12	2,51	2,79
	BP	***Cortex-A8***	7.56	8.45	9.48	10.59
		Cortex-A9	6,58	7,25	8,33	9,43
LD	**BQT**	***Cortex-A8***	0.53	0.87	1.11	1.28
		Cortex-A9	0,29	0,29	0,30	0,30
	BM	***Cortex-A8***	3.41	4.13	4.96	5.71
		Cortex-A9	3,26	3,85	4,76	5,26
	BP	***Cortex-A8***	9.61	10.97	12.71	14.43
		Cortex-A9	8,93	10,00	11,63	13,51
RA	**BQT**	***Cortex-A8***	0.59	0.93	1.12	1.20
		Cortex-A9	0,55	0,89	1,04	1,08
	BM	***Cortex-A8***	3.76	4.47	5.03	5.49
		Cortex-A9	3,66	4,17	4,84	5,13
	BP	***Cortex-A8***	10.19	11.64	13.14	14.43
		Cortex-A9	9,62	10,87	12,20	12,82

afin de calculer l'accélération obtenue lorsque l'optimisation du néon est utilisée pour les deux processeurs, cependant, Les résultats de l'accélération avaient 2,77 comme minimum et 4,22 comme maximum. Néanmoins, ces différences sont principalement dues d'une part à la vitesse de déplacement des séquences utilisées, et d'autre part aux caractéristiques différentes entre les deux processeurs proposés dans le Tableau 1. De manière générale, pour tous les cas, les résultats obtenus sur le cortex-A8 sont lents par rapport au cortex-A9, malgré une fréquence supérieure au cortex-A9, mais le cortex A8 a une architecture améliorée car il a annoncé après le cortex A8.

TABLE 3.8 – LA VITESSE ATTEINTE GRÂCE À LA TECHNOLOGIE DU NÉON.

			QP			
Config	**séquence**	**processeur**	***22***	***27***	***32***	***37***
Intra	**BQT**	***Cortex-A8***	3.61	3.76	3.74	3.71
		Cortex-A9	4,05	4,06	4,19	4,22
		Difference	**0,44**	**0,3**	**0,45**	**0,51**
	BM	***Cortex-A8***	3.66	4.16	3.26	3.72
		Cortex-A9	4,05	4,03	4,07	4,09
		Différence	**0,39**	**-0,13**	**0,81**	**0,37**
	BP	***Cortex-A8***	3.41	3.39	3.41	3.36
		Cortex-A9	3,78	3,64	3,78	3,64
		Différence	**0,37**	**0,25**	**0,37**	**0,28**
LD	**BQT**	***Cortex-A8***	3.64	3.62	3.64	3.65
		Cortex-A9	3,95	3,89	3,91	3,84
		Difference	**0,31**	**0,27**	**0,27**	**0,19**
	BM	***Cortex-A8***	3.38	3.31	3.27	3.20
		Cortex-A9	3,71	3,51	3,68	3,46
		Difference	**0,33**	**0,2**	**0,41**	**0,26**
	BP	***Cortex-A8***	3.03	2.94	2.85	2.77
		Cortex-A9	3,25	3,16	3,07	3,05
		Difference	**0,22**	**0,22**	**0,22**	**0,28**
RA	**BQT**	***Cortex-A8***	3.59	3.60	3.72	3.76
		Cortex-A9	3,85	3,85	3,90	3,80
		Difference	**0,26**	**0,25**	**0,18**	**0,04**
	BM	***Cortex-A8***	3.35	3.36	3.36	3.34
		Cortex-A9	3,66	3,53	3,63	3,56
		Différence	**0,31**	**0,17**	**0,27**	**0,22**
	BP	***Cortex-A8***	2.96	2.91	2.86	2.83
		Cortex-A9	3,21	3,20	3,02	2,85
		Différence	**0,25**	**0,29**	**0,16**	**0,02**

La compilation est effectuée sur l'outil TI Code Composer Studio Software Development Kit (SDK). Le logiciel de référence HM est écrit en langage C++ et compilé pour les deux architectures ARM. La chaîne d'outils G+++ fournie par TI est utilisée, et l'optimisation de la vectorisation NEON est automatiquement activée dans le jeu de niveaux -O3. Cette comparaison doit tenir compte de l'avantage des instructions SIMD qui convient dans le processus

d'encodage vidéo pour ce type de processus en raison d'un grand avantage en terme de gain de temps. Le jeu d'extension de l'architecture SIMD est appelé NEON pour les processeurs ARM Cortex-Ax de cette catégorie. Le NEON utilise un registre très long allant jusqu'à 64 bits / 128 bits. Et il est basé sur deux types de registres Q et D. Nous n'avons sélectionné que quatre séquences de la classe C (832×480), car les résultats obtenus après plusieurs tests sont presque les mêmes pour une autre classe. Les deux valeurs sélectionnées de QP parmi les quatre valeurs proposées par la norme dans [Noac], qui sont 27 et 32. Et nous avons utilisé le profil principal et la configuration à accès aléatoire. Toutes ces caractéristiques et encore plus d'informations sont déclarées dans le fichier de configuration JCT-VC documentent les conditions de test communes dans [Noac]. L'évaluation expérimentale de cette mise en oeuvre de la comparaison s'appuie sur le logiciel de référence HM version 15.0 [Noam]. Tous les tests ont été effectués en utilisant les 150 premières trames pour chaque séquence où sont stockés sur carte SD pour les deux cas. Tous les résultats des tests sont la moyenne de 20 fois le nombre d'exécutions. Tous les tests de performance pour l'architecture x86 sont exécutés sur un processeur Intel Coeur(TM) i3-370M cadencé à 2,4 GHz avec 4 de mémoire RAM DDR3 et il a Ubuntu version 14.04 comme système d'exploitation.

3.5.2 Résultats du profilage

L'analyse de complexité est limitée sur le cortex A8 uniquement, parce que les résultats sur les deux processeurs sont presque identiques, par conséquent, le profilage de chaque fonction à l'intérieur du décodeur logiciel HEVC sur les deux plates-formes reçoit une valeur avec quelques différences qui ne peuvent pas avoir une influence sur les résultats finals. Le tableau xx montre les résultats de profilage de tous les blocs du décodeur logiciel HM sur les cortex ARM A8 et A9. Tout d'abord, la fonction analyse NAL analyse le flux d'entrée. Ensuite, le décodage d'entropie traite les résultats de la fonction NAL via la fonction TDecEntropy incluse dans TDecTop. Ensuite, la fonction TComTrQuant a appelé au décodage IT (transformée inverse) et IQ (quantification inverse) en utilisant successivement les deux fonctions InvTransformNxN et setQPforQuant. Après la prédiction Intra et Inter calculée par la fonction TDecCU, pour l'Intra la fonction xRecontra est utilisée pour traiter le composant Luma (par la fonction xIntraLumaReQT) et Chroma (par la fonction xIntraReChromaBLK), pour l'inter, la fonction xReconInter est appelée pour décoder la fonction motionCompensation et la texture par fonction xDecodeInterTexture. Enfin, la fonction TDecTop contient deux fonctions pour décoder le filtre, la fonction Tcomloopfilter for Debloking Filter et la fonction adaptiveOffset pour SAO.

L'étude de complexité du profilage sur le cortex ARM-A8 est calculée comme suit : la complexité de chaque bloc référencé dans le schéma du décodeur HEVC est présentée par un certain nombre de fonctions, cependant, le temps de calcul est mesuré à partir de toutes les fonctions chargées pendant le fonctionnement du décodeur, puis il est regroupé pour que le temps des fonctions consacrées à chaque bloc soit présenté, afin de faire le profil par blocs et non pas à plusieurs.

les résultats du profilage sont présentés sur le Tableau **??**, l'exécution se fait avec l'optimi-

sation Néon, et la simulation est limitée à trois séquences pour chaque classe où est BQTerrace, BQMall et BasketballPass, car les résultats des autres séquences donnent presque les mêmes valeurs. également, deux valeurs moyennes de QP 27 et 32 sont sélectionnées, car elles ont été adoptées dans la majorité des travaux associés. en raison du profilage de chaque bloc décodeur, les blocs plus complexes restent les mêmes pour chaque valeur de QP et pour les trois classes (B, C et D). pour TOUS les modes Intra (intra), les blocs QT et INTRA sont plus complexes, mais en mode low-delay (LD) et Random Access (RA) les blocs QT et INTER sont plus longs à réaliser.

TABLE 3.9 – Résultats du profilage du décodeur avec la technologie Neon

séq	config	QP	EP	QT	INTRA	INTER	DF	SAO
BQT	INTRA	27	17,9	20,02	42,66	0	13,3	6,11
		32	14,07	17,79	45,7	0	15,69	6,76
	LD	27	15,14	9,15	0,59	57,63	10,87	6,62
		32	11,32	6,36	0,59	67,12	10,48	4,12
	RA	27	13,96	6,98	2,63	60,49	9,86	6,08
		32	10,58	4,88	2,4	70,52	9,22	2,4
BM	INTRA	27	16,74	20,1	43,57	0	13,04	6,19
		32	15,25	21,78	47,16	0	15,81	7,02
	LD	27	18,57	12,05	3,41	46,77	12,64	6,57
		32	16,23	10,45	3,27	52,46	13,56	4,04
	RA	27	15,19	8,06	5,79	59,91	11,05	3,5
		32	14,05	7,04	5,42	59,42	11,77	2,3
BP	INTRA	27	19,23	23,21	40,64	0	11,35	5,57
		32	14,62	24,18	41,6	0	13,36	6,21
	LD	27	21,23	14,37	7,37	37,19	12,91	6,93
		32	18,71	12,82	6,55	41,8	14,1	6,02
	RA	27	19,17	11,64	10,94	40,9	12,58	4,77
		32	16,67	9,93	8,99	48,08	13	3,32

Comme discussion, ce travail sur le décodeur logiciel HM sur ARM donne un aperçu du temps d'exécution sur deux générations différentes de processeurs. La version Cortex A9 est plus performante que Cortex A8, sachant que les deux plates-formes ont démarré le Linux embarqué en tant que système d'exploitation. Néanmoins, le profilage du décodeur a montré que les fonctions plus d'exécution sont transformées et intra environ 46% pour la configuration Intra Only, et pour les deux configurations Random Access et Low Delay, le bloc ont plus de temps est l'inter prédiction que jusqu'à environ 42% pour la plupart des cas. Par conséquent, dans les travaux futurs, nous nous concentrerons sur ces blocs plus consommer le temps afin d'accélérer le traitement.

3.5.3 Conclusion

Le but de cette section est d'implémenter le décodeur HEVC sur les processeurs embarqués ARM avec une comparaison, une approche est définie pour exécuter le logiciel HEVC pour décodeur sur les deux familles de versions ARM cortex-A8 et cortex-A9 avec optimisation NEON qui peut être accélérée de 4 fois dans la plupart des cas. De plus les résultats peuvent démontrer que le cortex a9 est meilleur que cortex a8, Hawevor, les résultats sont améliorables avec une stratégie pour exploiter toutes les autres coeurs afin de maximiser la fréquence de trame par seconde. Le temps réel pour la haute résolution a pu être obtenu en se basant sur l'implémentation sur des multicoeurs (huit coeurs) et en utilisant l'optimisation récente de l'IDMA comme QDMA et EDMA, qui sont intégrés dans cette puce qui peut être utilisée dans le plus grand échange de données en mémoire. Dans le travail suivant, nous nous basons sur les résultats de thèse présentés ci-dessus afin d'exécuter le décodeur HEVC sur un processeur ARM multicoeur qui peut être exécuté en temps réel pour la résolution supérieure comme 4k et 8K.

3.5.4 Comparaison avec d'autre travaux

Dans le tableau **??**, nous avons comparé nos travaux d'implémentation du codeur sur DSP et aussi sur ARM avec d'autres travaux existants dans la littérature sur le Tableau 15. Nous donnons une comparaison de profilage avec le travail [Van+12], qu'il a réalisé sur le CPU.

TABLE 3.10 – comparaison de la répartition du temps entre les blocs de HEVC.

Config	**Travaux**	Intra	Inter	TQ	DBK	SAO	EP	Other
AI	notre (DSP)	11,2	0	51,44	0,06	0,19	8,01	26,69
	notre (ARM)	45,1	0	22,1	1,2	0,6	17	14
	[Van+12]	35	0	40	0	0	10	16
LD	notre (DSP)	0,74	45,7	38,81	0,12	0,07	5,09	8,75
	Notre (ARM)	7	59,3	11	0,51	0,7	9,3	12,19
	[Van+12]	1	81	9	0	0	3	9
RA	notre (DSP)	0,81	39,46	43,92	0,01	0,05	5,91	9,09
	notre (ARM)	6,3	57,4	13,1	0,61	0,87	9,1	12,19
	[Van+12]	1	78	11	0	0	2	8

Pour toutes les configurations, les filtres sont négligeables, pour l'entropie de codage, nos résultats sont similaires à ceux présentés dans [Van+12], mais la QT est plus complexe et l'inter prédiction est moins complexe que les travaux rapportés dans [Van+12].

pour remplacer la multiplication par élément de traitement (PE), qui supporte tous les coefficients existant à l'intérieur du DCT entier.

4.4.2.1 Architecture proposée pour l'élément de traitement (PE)

Tous les coefficients possibles de la transformation du noyau A sont présentés dans le Tableau **??**, la multiplication est remplacée par un décalage et une addition. Par exemple, dans l'équation IV, cette valeur est appliquée au coefficient 89.

TABLE 4.4 – les nombres existants dans tous tailles de matrices 1-D DCT.

Size 1-D DCT	**les coefficients des matrices**
32×32	90,88,85,82,78,73,67,61,54,46,38,31,22,13,4
16×16	90,87,80,70,57,43,25,9
8×8	89,75,50,18
4×4	83,36

La Figure 4.9 L'élément de traitement de l'accélérateur matériel (PE) proposé pour le calcul des valeurs planaire et angulaire de la prédiction intra-prévision. montre l'architecture PE qui calcule la multiplication par les coefficients de la matrice de transformation du noyau. Il contient un multiplexeur, un accumulateur, un compteur 3 bits et l'unité de contrôle en plus de deux registres. L'unité de commande est utilisée pour réinitialiser l'accumulateur, le compteur et pour commander le multiplexeur de coefficients.

L'accumulateur est utilisé pour stocker les résultats des étapes intermédiaires dans chaque opération afin d'être utilisé dans l'horloge suivante. La sortie du PE est donnée tous les six cycles : un cycle pour charger chaque valeur et 5 cycles aux étapes nécessaires pour calculer les coefficients les plus élevés du Tableau 19. Par exemple, le coefficient 87[0101010111b], prend 5 cycles pour appliquer le décalage et l'addition.

Le PE fonctionne comme suit, lorsque le coefficient est défini, sur la base de sa décomposition déterminée précédemment. L'opération commence par un passage de l'ordre supérieur à l'ordre inférieur, il faut un cycle d'horloge pour chaque décalage, l'accumulateur est utilisé pour ajouter la valeur courante dans le résultat avant, et le compteur est utilisé pour contrôler le nombre de cycles pour l'accumulateur et le fonctionnement par décalage. Le compteur est limité à 3 bits car le PE nécessite 6 cycles. Après le PE pour le calcul de l'élément, l'étape suivante consiste à proposer une conception pour chaque matrice basée sur l'architecture PE.

Les Figure **??** et Figure **??**. Montrent l'architecture matérielle pour le calcul des matrices B2 et C2, respectivement. Chacun d'eux possède deux PE, des multiplexeurs et registres arborescents, ainsi qu'un compteur et un additionneur. Le compteur 1 bit est utilisé pour sélectionner les multiplexeurs. Il compte tous les cinq cycles après l'enclenchement des PEs qui travaillent indépendamment. La sortie est donnée après six cycles d'horloge. La Figure **??** affiche l'accélérateur dans l'ordre N pour la transformation DCT 1-D qui est appliqué pour

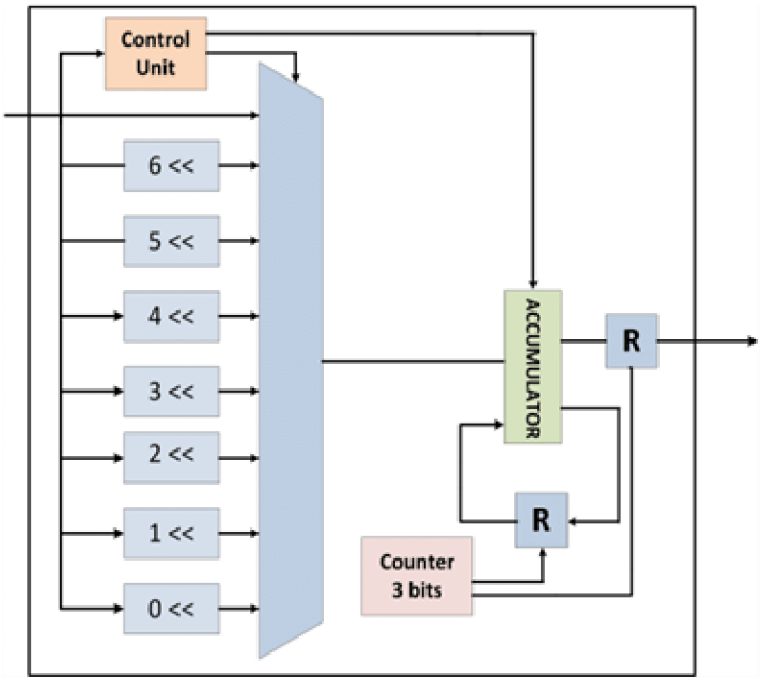


FIGURE 4.9 – Architecture matérielle de l'élément de traitement (PE).

calculer toutes les tailles de TU de 4×4 à 32×32 en même temps car, chaque petit ordre N-1D est contenu dans l'ordre N-higher.

La Figure 4.13 présente l'architecture proposée pour le traitement de la transformation DCT 2D. Il possède une transformée 1-D DCT suivie d'une mémoire transposée dont la sortie est utilisée pour la partie suivante de la transformée DCT 2D.

4.4.3 Résultats de l'implémentation et comparaisons

D'une part, la transformation des entiers du bloc 2-D DCT est implémentée en langage C afin de valider et réduire la complexité de l'architecture proposée ; d'autre part, le matériel accélérateur est décrit en langage VHDL et prototypé sur FPGA Artex-7, intégré sur processeur ZYNQ-7000. Les résultats de sortie obtenus après le traitement dans l'architecture matérielle proposée sont d'abord comparés aux résultats de sortie générés par le logiciel de référence de la version HEVC HM.15.0. Ensuite, ils sont également comparés aux logiciels implémentés en langage C pour valider l'architecture proposée. Toutes ces comparaisons sont un bon accord.

Les résultats de synthèse de l'architecture proposée sont présentés dans le Tableau **??**, cette architecture fonctionne jusqu'à 289.77 MHz comme fréquence d'horloge maximale et consomme 34421 LUTs qui représentent 64.70 % des ressources pour notre appareil (53200 LUTs) et 8729 registres bits qui représentent 8.2% des registres ressources. Enfin, il n'a pas utilisé de DSP parce qu'il n'a pas d'opération de multiplication.

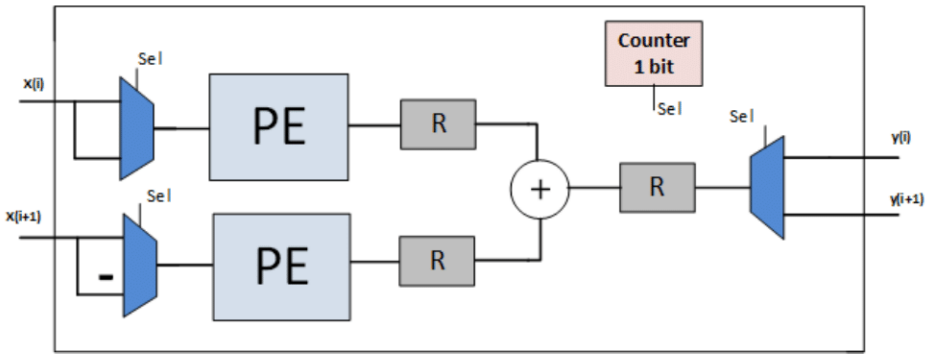


FIGURE 4.10 – Architecture matérielle pour la matrice B2.

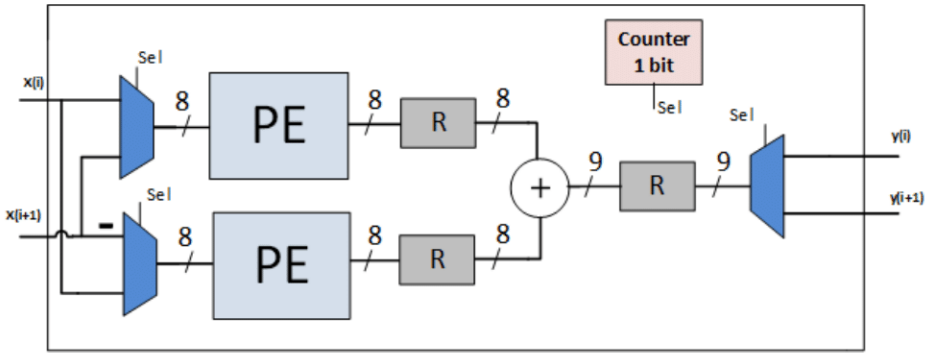


FIGURE 4.11 – Architecture matérielle de la matrice C2.

La simulation dans ModelSim (Figure **??** et Figure **??**) démontre que notre architecture peut être calculée en bloc 32×32 sur 192 cycles d'horloge et qu'il faut 6 cycles pour traiter un élément de traitement (PE). L'architecture proposée peut calculer 30 images par seconde (fps) pour la résolution 8K et 120 fps pour la résolution Ultra HD. En fait, c'est une bonne chose. Le 8K (7680×4320 pixels) et pour 30 images par seconde avec le format d'échantillonnage 4 :2 :0 requis (7680×4320×1.5×30) 1492.99 MSample/s comme débit, notre débit est donc supérieur à cette valeur. De plus, l'architecture matérielle proposée peut être améliorée pour traiter plus d'images par seconde dans un périphérique plus riche en ressources.

Dans le Tableau **??**, nous comparons l'accélérateur 2D-DCT matériel proposé avec certains travaux rapportés dans la littérature. La conception proposée dans [Zho+15] ne prend en charge que deux cas 16 et 32 points, successivement

La plus petite zone rapportée pour [64], [91] peut être justifiée étant donné que ces auteurs n'ont traité qu'un seul bloc dans chaque cas. Par conséquent, ils présentent une fréquence basse avec une fréquence plutôt acceptable dans l'ensemble. Rigoureusement, notre travail ne peut être comparé qu'à celui rapporté dans [66]. En fait, ils ont utilisé la même série d'appareils FPGA (série 7 de Xilinx). Ils ont utilisé moins de ressources avec un débit plus élevé, mais leur architecture ne peut pas exécuter le DCT 2D à l'opposé du nôtre.

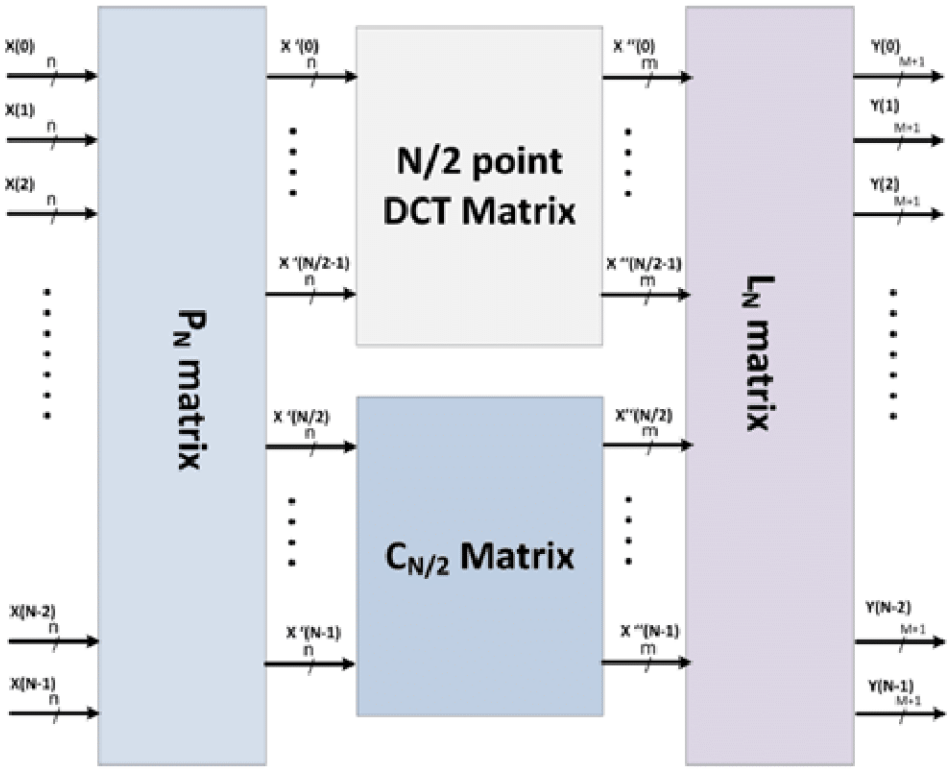


FIGURE 4.12 – Accélérateur matériel de la commande-N 1-D DCT transformée.

4.4.4 Conclusion

Une nouvelle architecture matérielle efficace pour la conception d'un accélérateur de haute performance de transformée DCT 2D pour HEVC est présentée dans cette section. Nous avons décomposé la matrice de transformation en multiplications de différentes matrices. Ces matrices ont été implémentées sur la base d'un matériel d'éléments de traitement, ne contenant que des éléments shift et adder sans aucune multiplication pour réduire la surface et obtenir une solution d'exploitation matérielle rapide. Les résultats expérimentaux obtenus avec le FPGA Xilinx Zynq-7000 Artix-7000 Artix-7 après synthèse montrent que notre architecture peut traiter des trames de haute résolution jusqu'à 8K en temps réel avec un débit jusqu'à 30 images par seconde.

4.5 L'implémentation de mode décision

4.5.1 Introduction

De nombreux efforts sont nécessaires pour déterminer laquelle des implémentations matérielles ou logicielles peut être la plus avantageuse pour chaque tâche. Par exemple, il est

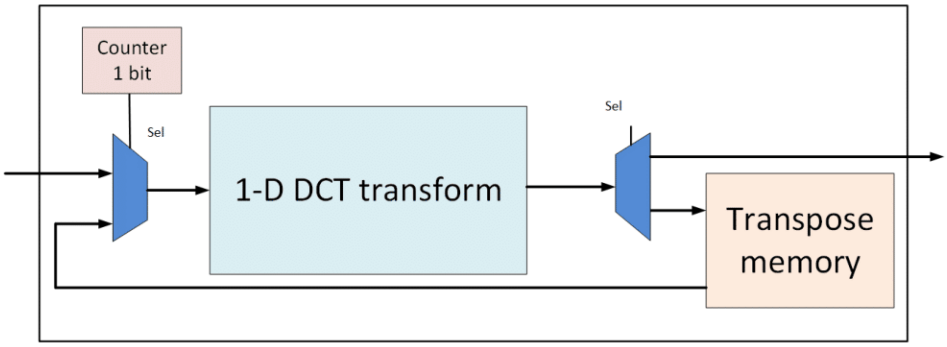


FIGURE 4.13 – Architecture proposée pour le calcul de la transformation DCT 2D.

TABLE 4.5 – RÉSULTATS DE SYNTHÈSE DE L'ARCHITECTURE MATÉRIELLE.

Available ICT	4/8/16/32
Slice LUTs	34421
Slice registers	8729
Maximum frequency	289.77
DSP	0
Memory	0

préférable de convertir la partie la plus consommatrice de temps en partie matérielle, comme dans le cas de tâches utilisant des échanges de communication entre blocs. Parfois, mélanger les avantages du logiciel et du matériel peut être plus bénéfique dans l'ensemble du processus de codage vidéo. Il s'agit d'abord de profiler le projet logiciel pour identifier les modules les plus lourdes et décider quel type de conception est le plus adapté. Heureusement, de nombreuses cartes de développement embarquées offrent cette possibilité de conception de mélange, parmi lesquelles le Chip Zynq 7000 de la famille Xilinx, basé sur la technologie CMOS 28 nm. L'inter prédiction est le bloc de temps de chargement le plus important de l'encodeur, car il contient les algorithmes d'estimation et de compensation de mouvement [Bos+12]. L'algorithme d'estimation de mouvement reste le même que pour le H.264/AVC, basé sur le coût SAD avec la possibilité de traiter les plus grandes tailles de PUs annoncer dans le HEVC.

Dans la plupart des cas, le logiciel de référence du modèle d'essai standard HEVC (HM) est disponible en ligne [HHW17]. La classe TComRDCost prend le temps d'exécution le plus long du codeur HM, comparé aux autres classes. Cela prend environ 40% du temps total d'encodage [Bos+12]. Il comprend le calcul de la somme absolue de la différence (SAD) et d'autres mesures de distorsion. La fonction GetSAD, qui calcule et donne le SAD de chaque comparaison entre deux blocs, comme sous fonctions on trouve : xGet-SAD4, xGetSAD8, xGetSAD16, xGetSAD16, xGetSAD32 et xGetSAD64 ; chacun est sélectionné selon la taille du bloc correspondant, appelé côté entrée, respectivement 4×4, 8×8, 16×16, 32×32 et 64×64.

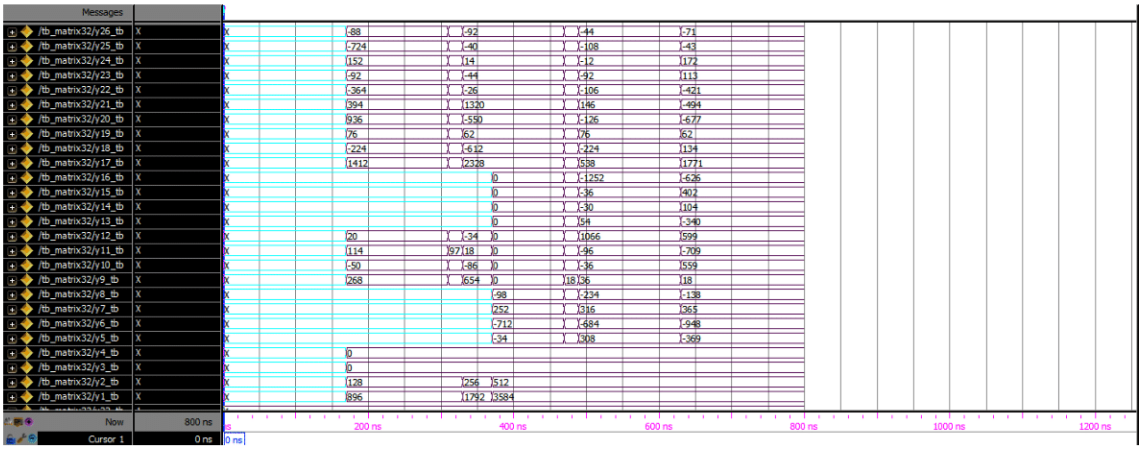


FIGURE 4.14 – la simulation de accélérateur hardware sur le logiciel modelsim pour le matrice 4x4.

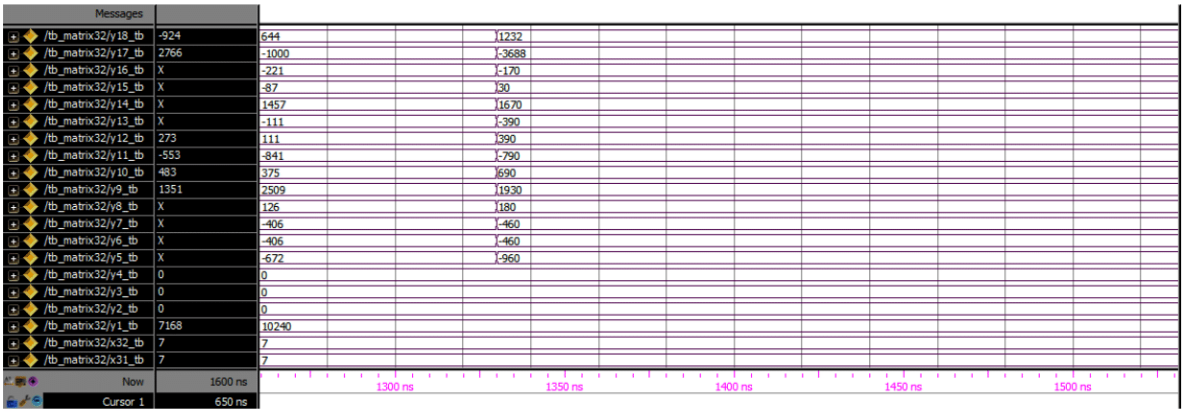


FIGURE 4.15 – simulation de accélérateur hardware sur le logiciel Modelsim pour le matrice 32x32.

De plus, xGetSAD12, xGetSAD24 et xGetSAD48 sont utilisés pour le partitionnement asymétrique du mouvement (AMP). Pour calculer le temps de chargement de chaque fonction SAD à l'intérieur du logiciel de référence HM15.0 [Noak], le profilage est utilisé à partir de Gprof, un outil d'analyse des performances pour les applications Unix. L'encodeur est exécuté sur un ordinateur portable qui dispose de 4 Go de RAM et d'un noyau Intel duel cœur cadencé à 2,53 GHz et démarrer sous Linux Ubuntu 14.04 LTS. De plus, les configurations utilisées dans cette simulation sont l'accès aléatoire avec QP égal à 32 et la séquence BasktBallDrive (1920 × 1080 pixels) avec 500 frames. Les résultats de cette comparaison après un moyenne de 20 tests sont présentés dans la Figure **??**.

Dans un travail précédent [EAAM17], nous avons comparé l'implémentation d'un codeur HEVC complet dans les deux processeurs, ARM et Intel afin d'étudier la complexité du logiciel, de sorte que, dans le processeur Intel est environ 10 fois plus rapide que le processeur intégré ARM, mais les résultats d'exécution trop loin des conditions de traitement temps réel, notamment pour la haute résolution. Pour surmonter ce problème, nous nous concentrons sur

TABLE 4.6 – COMPARAISON AVEC D'AUTRES TRAVAUX

	Our work	**Dias[66]**	**Jeske [91]**	**Zhao [64]**
FPGA	Artix-7	Virtex-7	Startix-III	Cyclone-IV
resources	8729/34421 R/LUTs	6,420/21,452 R/LUTs	5,168 LEs	5,446 LEs
frequency	289MHz	280Mhz	87Mhz	125Mhz
Throughput	1,541M	2.2G	1,404M	780M
DCT Type	2-D DCT	1-D DCT	1-D DCT	2-D DCT
TUs	4/8/16/32	4/8/16/32	16	32

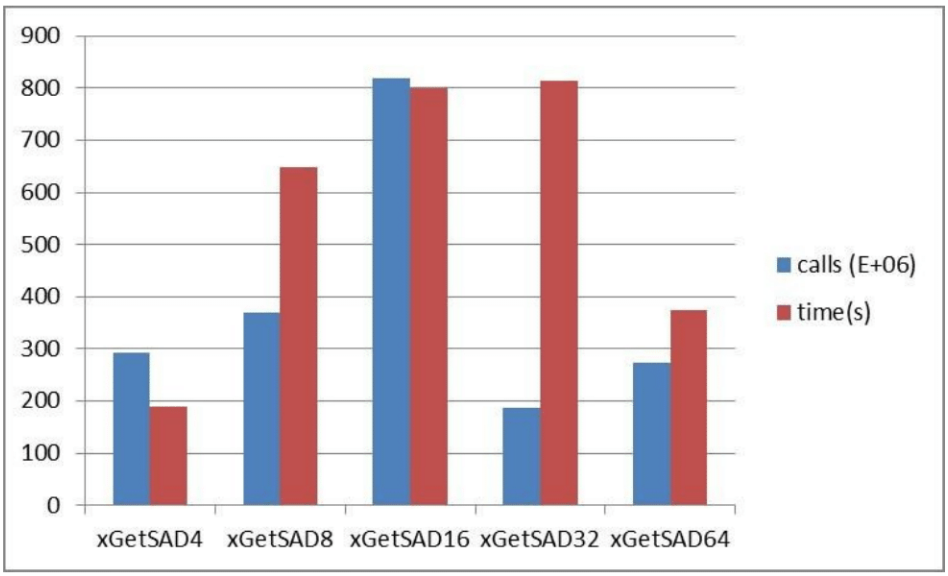


FIGURE 4.16 – Le nombre d'appels et le temps de chargement des fonctions SAD dans le codeur HEVC.

l'implémentation matérielle dans ce travail. Le logiciel de référence du modèle d'essai standard du HEVC (HM) est disponible en ligne [Noak]. La classe TComRDCost, qui est utilisée pour le calcul des coûts RD et qui inclut toutes les fonctions pour le calcul SAD. Le TComRDCost prend le temps d'exécution le plus long du codeur HM, comparé aux autres classes. Cela prend environ 40% du temps total d'encodage [Bos+12].

4.5.2 HEVC Mode Décision de l'algorithme d'estimation du mouvement

4.5.2.1 Estimation du mouvement entier (IME) dans l'encodeur HEVC

Dans la Figure 4.17, l'inter prédiction dans le codeur HEVC est traitée en trois étapes. La première consiste à fixer le point de départ de l'EMI à l'aide du processus des prédicteurs avancés de vecteurs de mouvement (AMVP), ce qui permet de trouver la meilleure prédiction de vecteurs de mouvement (MVP) parmi les candidats des vecteurs de mouvement temporels et spatiaux. Ensuite, le processus IME, qui est un niveau entier, est utilisé pour trouver le meilleur vecteur de mouvement dans la région délimitée. Ensuite, l'estimation fractionnaire du mouvement (FME), qui est le niveau fractionnaire, est appliquée pour affiner la recherche autour de l'IME en utilisant le filtrage d'interpolation. Cette étape se déroule en deux étapes, la première pour trouver le vecteur de mouvement quart de pixel et la seconde pour trouver le vecteur de mouvement hal-pel pixel pour obtenir le meilleur emplacement.

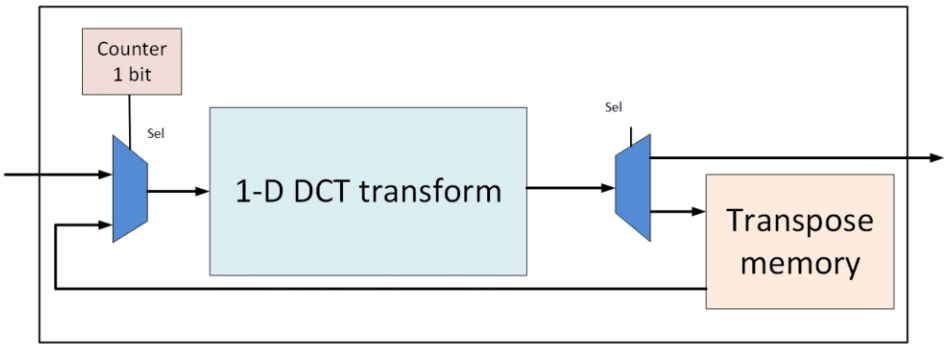


FIGURE 4.17 – Architecture proposée pour le calcul de la transformation DCT 2D.

Tableau 22 PARTITIONNEMENT DE L'UNITÉ DE CODAGE DU PUS POUR CHAQUE NIVEAU DE PROFONDEUR DE CODAGE DIFFÉRENT Codage Niveau de profondeur partitionnement en blocs

TABLE 4.7 – L'UNITÉ DE CODAGE SÉPARANT DU PUs POUR CHAQUE DIFFÉRENT NIVEAU DE PROFONDEUR.

Niveau de profondeur	**Le bloc de partitionnement**
0	64×64 ; 64×32 ; 32×64 ; 32×32 ; 16×64 ; 48×64 ; 64×16 ; 64×48
1	32×32 ; 32×16 ; 16×32 ; 16×16 ; 8×32 ; 24×32 ; 32×8 ; 32×24
2	16×16 ; 16×8 ; 8×16 ; 8×8 ; 4×16 ; 12×16 ; 16×4 ; 16×12
3	8×8 ; 8×4 ; 4×8

Le HEVC adopte la recherche par zone d'essai (TZS) pour l'EMI, un regroupement entre la recherche au diamant (DS) (ou recherche par carré) et la recherche par séquence, présentée à la Figure 4.17 après la fixation de la position de départ par le processus MVP, la première

étape consiste à appliquer la grille de recherche du diamant pour chaque itération afin de trouver le point avec la distorsion de coût minimale pour l'itération. Ce point sera le centre de la prochaine itération. La deuxième étape est la recherche matricielle, qui se poursuit si la distance entre le centre de l'itération en cours, et le meilleur point est supérieur à la constante iRaster (égale à 5 en HM). Dans la recherche en diamant, pour chaque itération, on prend huit points, sauf pour la quatrième, qui ne nécessite que quatre points. Pour 64×64 PU, le nombre d'itérations nécessaires est 7 pour scanner toute la fenêtre de recherche 64 ; au total, 52 ((68) + 4) points de recherche doivent être considérés.

Figure 4.17 TZS avec un modèle de recherche de diamants et de trame : (a) Recherche de diamants avec quatre itérations. b) trame Pour atteindre le traitement en temps réel pour l'application 4k, dans le sous-échantillonnage couleur au format Chromatique 4 :2 :0, pour un bloc de codage 64×64, on distingue le composant Luma 64×64 et deux blocs 32×32 Chroma composants Cr et Cb. Le nombre total possible d'unités de puissance est détaillé dans le Tableau 22. Selon le principe du Coding Tree Block (CTB) [YXQ07], 593 PU pour 64×64 et 145 pour 32×32 ; par conséquent, un CTB 64×64 a 883 PU candidates. Donnant ces statistiques, pour qu'une séquence 4K (3840x2160) atteigne le traitement en temps réel (30 images par seconde), elle doit traiter 60,75K CTB/s ((3840×2160×30)/(64×64)) en une seconde ou 53,642 millions (60,75K883) points de recherche par seconde lorsque la recherche par balayage est inhibée ; cependant, si elle est active, ce nombre augmente à 53,663 millions par CTU. Donner cette recherche matricielle est une recherche complète (Figure 4.17 (b)), dans le pire des cas, lorsque toutes les PUs d'une CTU ont une distance entre le centre et le point le mieux trouvé supérieure à cinq, cinq pixels en largeur et en longueur, 25 points sont ajoutés dans chaque PU correspondant à 20825 (883x25) points ajoutés pour une CTU. Le SAD est appliqué pour calculer le coût de chaque point.

4.5.2.2 Algorithme du SAD dans le HEVC

La partie critique des normes des codeurs vidéo exploitant la redondance temporelle est l'estimation du mouvement (ME). Le récent HEVC a un algorithme d'estimation de mouvement plus complexe que le H.264/AVC, en raison de la densité des pixels dans le traitement haute résolution de l'image (8K et 4K) et le nouveau partitionnement de blocs PU. Le traitement SAD est effectué pour l'entier ME (IME), et il est calculé pour toutes les tailles pour les candidats PU. Il est calculé sur la base des informations provenant du bloc courant ou original (Bo) de l'UP et du bloc prédit ou de référence (Br) obtenu par le processus de dérivation des prédicteurs du vecteur de mouvement. Le calcul du SAD est effectué à l'aide de l'équation 4.19.

$$SAD = \sum_{i=1}^{k} \sum_{j=1}^{L} |Br(i,j) - Bo(i,j)| \tag{4.19}$$

Où i et j sont les pixels correspondants dans les blocs Bo et Br. K et L sont la largeur et la hauteur de chaque fenêtre de recherche, respectivement. Indique le pixel du bloc de

référence et Bo(i,j) indique le pixel du bloc original (courant). La complexité du calcul réside dans le nombre d'appels et la taille différente du SAD pour l'IME de l'inter prédiction. Par conséquent, un accélérateur matériel de SAD est nécessaire.

4.5.3 Proposition de l'architecture matériel de mode décision

Le SAD est une fonction de coût utilisée pour calculer la distorsion entre deux blocs. Plusieurs architectures pour le calcul du SAD ont été proposées dans la littérature. Ils sont principalement basés sur l'un des trois types de conception suivants : parallèle, pipeline et séquentiel. La conception séquentielle utilise un accumulateur pour stocker les résultats de la valeur absolue dans chaque cycle d'horloge. Il utilise un minimum de matériel, mais il faut plus d'un cycle d'horloge pour obtenir les résultats ; par la suite, il est impossible d'obtenir un traitement en temps réel en raison de la résolution plus élevée dans nos cas. L'architecture en pipeline, puisqu'elle utilise une série d'étapes connectées, prend en entrée un vecteur de pixels au lieu d'une entrée de pixels et génère plusieurs résultats sur une ligne ; par exemple, dans [WDB12], les auteurs utilisent un vecteur d'entrée de 128 bits présente 16 pixels échantillon 8 bits de 4×4 bloc échantillon. L'architecture du pipeline est donc plus rapide que l'architecture séquentielle, mais ce n'est pas suffisant pour atteindre le temps réel en haute résolution. Dans [WDB12], l'architecture proposée est conçue par étapes en pipeline ; cependant, elle n'atteint un taux de 30 fps que pour de petites séquences de 720×480 pixels avec le niveau de configuration 3 du codeur H.264/AVC. La conception parallèle est plus rapide que la conception séquentielle et la conception en pipeline, car elle est construite sur plusieurs tranches, qui sont traitées indépendamment l'une de l'autre ; par conséquent, la consommation de surface est accrue. De plus, il est possible de mélanger ces types ; par exemple, l'architecture totale fonctionne en parallèle, et le design des silices fonctionne en mode séquentiel ou en pipeline comme dans la réf. [LGI08]. Pour l'utilisation d'une architecture entièrement parallèle, une conception hautement optimisée est nécessaire pour réduire la surface consommée et maintenir des performances supérieures. Le parallélisme dans l'architecture matérielle est exploré pour augmenter le débit de l'accélérateur afin de calculer toutes les opérations SAD sur quatre cycles d'horloge, mais le nombre de pixels d'échantillons d'entrée augmente et le nombre d'additionneurs dans l'arbre de profondeur augmente. Une optimisation est nécessaire pour réduire la surface consommée et le délai d'accès au système de codage en temps réel, notamment pour les applications 4K. L'implémentation de l'algorithme SAD se fait en trois étapes principales ; d'abord la différence absolue entre deux pixels est calculée par l'unité de traitement (PrU) conçue dans la Figure **??**. Ensuite, la somme de ces différences est calculée en une seule étape pour quatre pixels d'entrée en utilisant l'élément de traitement conçu (PE) présenté dans la Figure 4.21. Enfin, pour l'ensemble de l'algorithme SAD, le PE est implémenté 1024 fois pour permettre le traitement de toutes les tailles de blocs de 4×8 & 8×4 à 64×64.

4.5.3.1 Architecture de la différence absolue

Le PrU, étant donné la différence absolue (abs), fonctionne par comparaison entre les deux pixels provenant de l'image de référence et de l'image actuelle. Si la différence est positive ou égale à zéro, l'abs est directement obtenu ; sinon, les deux compléments du terme négatif sont utilisés pour rendre le résultat positif. Cette opération de comparaison de pixels peut être décrite par l'équation suivante 4.20 :

$$abs = \begin{cases} Br - Bo, & \text{if } Br - Bo \geq 0. \\ not(Br - Bo) + 1, & Br - Bo < 0. \end{cases} \tag{4.20}$$

L'architecture matérielle proposée pour PrU est présentée à la Figure 4.19. Il s'agit d'un circuit optimal utilisant un multiplexeur (MUX) dont l'entrée de sélection correspond à l'un des deux états des sorties du comparateur (Comp) (c'est-à-dire valeur positive ou négative). Selon le signe de la comparaison, le MUX donne dans chaque cas la différence absolue provenant de la sortie du soustracteur (SUB). L'équation not(Br-Bo)+1 est réalisée en utilisant un onduleur suivi d'un additionneur 1 bit. Les résultats sont toujours codés sur 8 bits. Les registres (R) sont utilisés pour stocker les valeurs des pixels traités. Pour diminuer le coût du matériel, la conception du PrU est optimisée par l'optimisation du comparateur.

La fonction principale de Comp est de comparer deux signaux d'entrée et de renvoyer les résultats sur trois sorties, chacune codée sur 1 bit. Le Comp est un comparateur 8 bits. Pour réduire la surface utilisée, il contient les portes logiques du comparateur 1 bit (Figure **??**) et utilise le compresseur principal 2 :1 décrit dans la réf [Kim+12a]. Dans notre architecture [YXQ07], la construction Com est basée sur un arbre de comparateur 1 bit utilisant la méthode du compresseur 2 :1 pour comparer deux pixels d'entrée A et B codés sur 8 bits et délivrer les résultats de la comparaison .

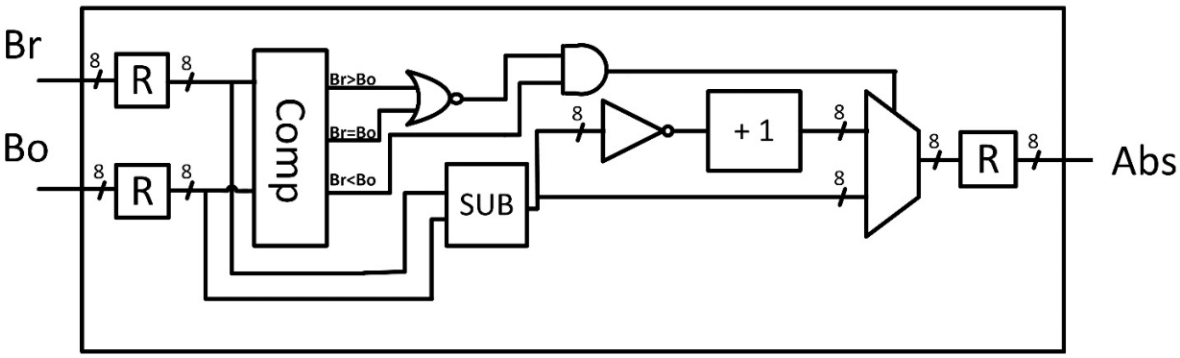


FIGURE 4.18 – Comparateur à 1 bit utilisé dans la Comp.

Le PE, comme le montre la figure 4.21, est composé de quatre PrU, sept registres et trois additionneurs pour calculer la somme de la différence absolue de huit pixels, quatre du bloc

de référence et quatre de l'original.

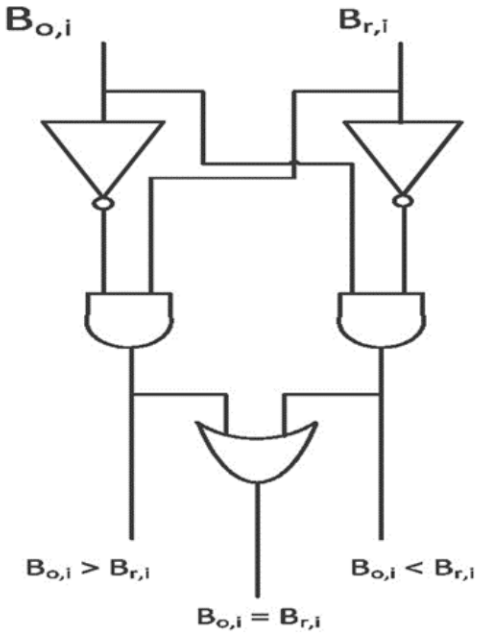


FIGURE 4.19 – Comparateur 1 bit utilisant dans la Comp.

Le résultat est donné par le signal de sortie S4x1, qui est maintenant codé sur 10 bits et obtenu en un cycle d'horloge.

L'architecture globale de calcul du SAD est présentée à la Figure **??**. Il contient :

— Trois blocs RAM 4K à port unique, les deux premiers (2×1024 ×8 bits) sont dédiés aux pixels des blocs de stockage actuels et originaux, et le troisième est utilisé pour stocker les résultats des SADs calculés.
— 1024 PE disposés en parallèle pour commencer à fonctionner en même temps, et calcul ligne par ligne de la somme partielle des différences absolues (PSAD) pour les huit pixels sélectionnés.
— Un additionneur de trois blocs pour délivrer le SAD en additionnant le PSAD deux par deux jusqu'au résultat final. Cette partie est appelée addition de combinaison dans la Figure 4.22.

Dans l'architecture matérielle, les PrU sont organisées ligne par ligne de telle sorte que, à partir de huit PrU, deux petites PU 8×4 et 4×8 peuvent être construits.

De plus, l'architecture proposée est conçue pour le plus grand bloc de taille 64×64 et peut facilement être adaptée pour les autres blocs de plus petites tailles avec un stockage mémoire adéquat des pixels de départ donnant dans la Figure 4.22.

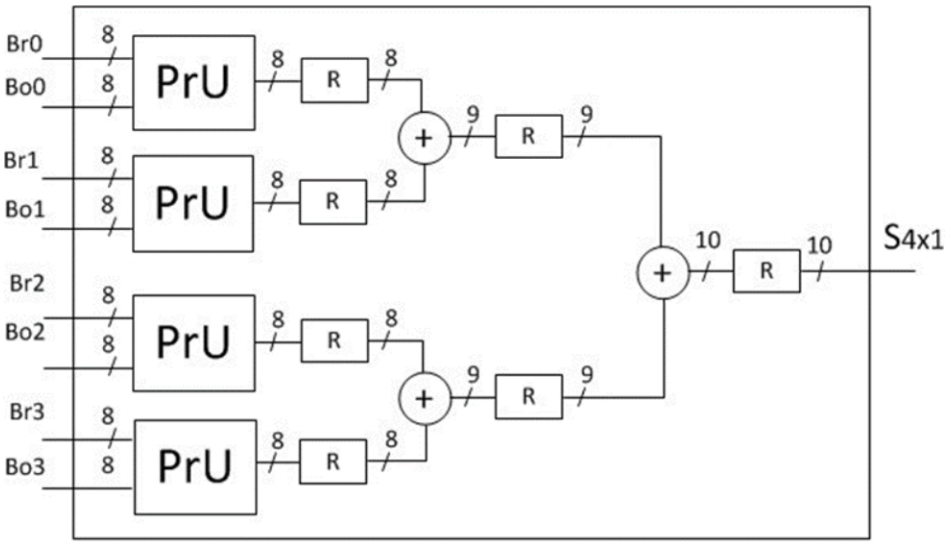


FIGURE 4.20 – Elément de traitement (PE) pour huit entrées et une sortie pixels.

4.5.3.2 Additionneur haute vitesse RCA

L'algorithme SAD est basé sur une opération arithmétique de base, puisqu'il comporte des sommations doubles. Pour le rendre plus rapide, plusieurs architectures proposées utilisent la propagation de carry pour l'addition binaire connue par ses hautes performances. Par conséquent, sa mise en œuvre est adéquate pour l'intégration à très grande échelle (VLSI).

Dans les références [YXQ07], [Kim+12a], les auteurs ont utilisé deux additionneurs parallèles plus rapides, Ripple Carry Adders (RCA) et Carry Lookahead Adders (CLA), et ils font une comparaison entre eux basée sur le retard et la consommation de surface. Premièrement, RCA prend plus de temps que le CLA parce qu'il a une propagation longue (à travers tous les additionneurs) ; il commence à partir du premier additionneur et il finit au dernier. Le CLA n'a pas de propagation de porteuse parce que la sortie de porteuse de l'additionneur de courant est connectée au bit d'entrée de porteuse de l'additionneur suivant. Pour la zone, la RCA nécessitait moins de surface que la CLA, puisque la dernière a une architecture scalaire et qu'elle avait besoin de plus de composants. Étant donné que nous sommes intéressés par un système rapide, le CLA est plus adopté dans notre accélérateur matériel pour concevoir tous les additionneurs, même s'il augmente la surface nécessaire mais sans affecter la surface totale ; de plus, il est simple à concevoir, étant donné que l'architecture CLA utilise la logique des portes de base telles que AND, OR et XOR.

Notre CLA comprend deux niveaux. Le premier niveau est utilisé pour calculer les valeurs partielles (intermédiaires), Gi et Pi, et générer les termes SUM(i) et Ci+1 comme sortie, définis par l'équation présentée dans la Figure 4.23, qui propose une section du plan de la CLA. Le deuxième niveau est la conception totale de la CLA (Figure 4.24, qui calcule chaque porte, la somme totale et indique le débordement de l'addition en sortie. Lorsque chaque report est disponible, la somme partielle est calculée simultanément, puis le trop-plein de

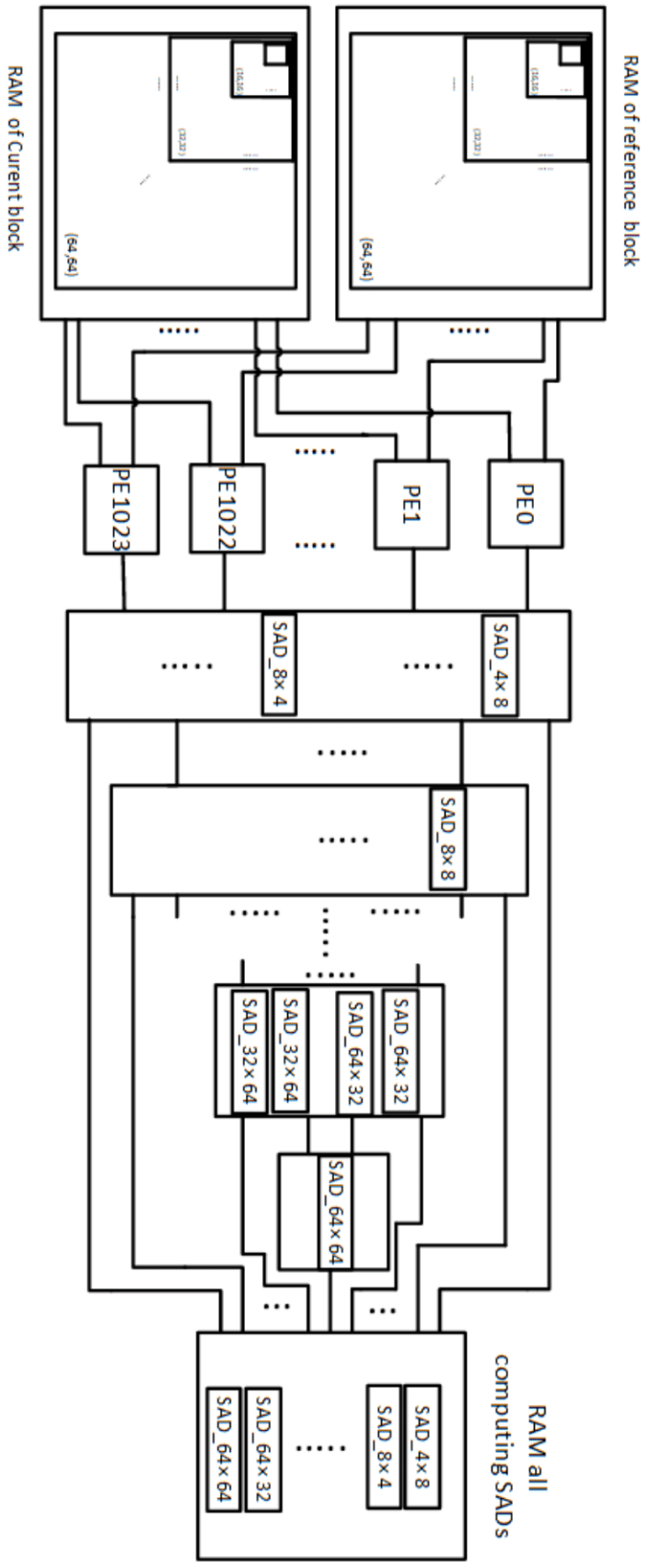


FIGURE 4.21 – L'architecture globale de l'algorithme SAD dans IME pour les codeurs HEVC.

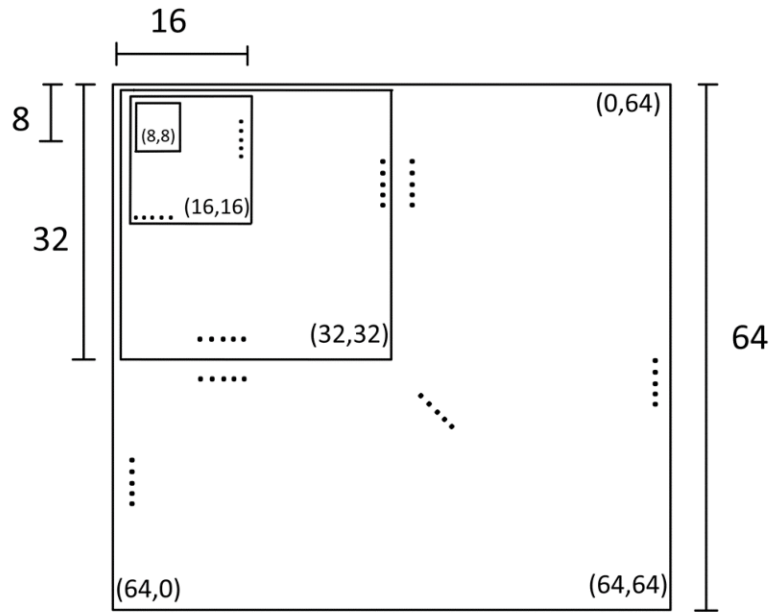


FIGURE 4.22 – Emplacement des pixels d'entrée des données Sub-CU 64×64.

l'addition est calculé à la fin.

4.5.3.3 Additionneur haute vitesse CLA

L'algorithme SAD est basé sur une opération arithmétique de base, puisqu'il comporte des sommations doubles. Pour le rendre plus rapide, plusieurs architectures proposées utilisent la propagation de carry pour l'addition binaire connue par ses hautes performances. Par conséquent, sa mise en œuvre est adéquate pour l'intégration à très grande échelle (VLSI).

Dans les références [YXQ07], [Kim+12a], les auteurs ont utilisé deux additionneurs parallèles plus rapides, Ripple Carry Adders (RCA) et Carry Lookahead Adders (CLA), et ils font une comparaison entre eux basée sur le retard et la consommation de surface. Premièrement, RCA prend plus de temps que le CLA parce qu'il a une propagation longue (à travers tous les additionneurs) ; il commence à partir du premier additionneur et il finit au dernier. Le CLA n'a pas de propagation de porteuse parce que la sortie de porteuse de l'additionneur de courant est connectée au bit d'entrée de porteuse de l'additionneur suivant. Pour la zone, la RCA nécessitait moins de surface que la CLA, puisque la dernière a une architecture scalaire et qu'elle avait besoin de plus de composants. Étant donné que nous sommes intéressés par un système rapide, le CLA est plus adopté dans notre accélérateur matériel pour concevoir tous les additionneurs, même s'il augmente la surface nécessaire mais sans affecter la surface totale ; de plus, il est simple à concevoir, étant donné que l'architecture CLA utilise la logique des portes de base telles que AND, OR et XOR.

Notre CLA comprend deux niveaux. Le premier niveau est utilisé pour calculer les valeurs partielles (intermédiaires), Gi et Pi, et générer les termes SUM(i) et Ci+1 comme sortie, définis

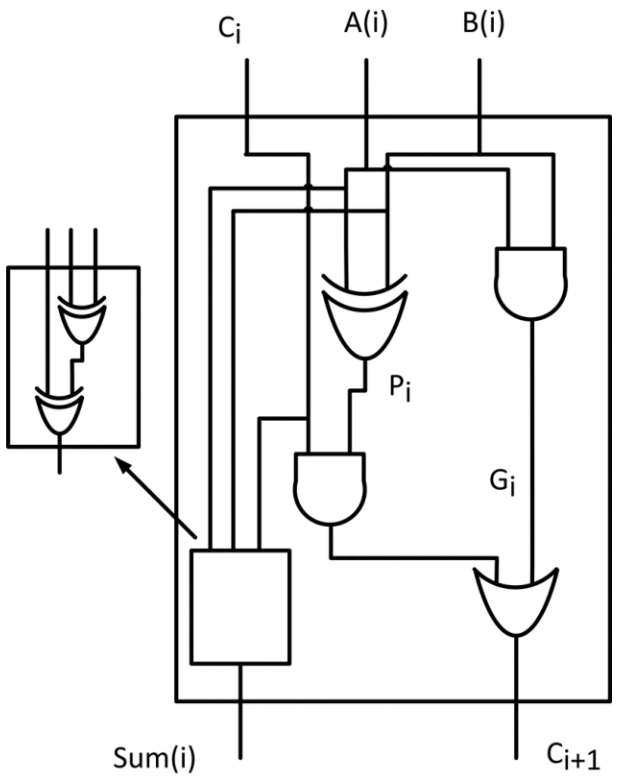


FIGURE 4.23 – Section proposée de la CLA

par l'équation présentée dans la Figure 4.23, qui propose une section du plan de la CLA. Le deuxième niveau est la conception totale de la CLA (figure 2.6), qui calcule chaque porte, la somme totale et indique le débordement de l'addition en sortie. Lorsque chaque report est disponible, la somme partielle est calculée simultanément, puis le trop-plein de l'addition est calculé à la fin. Le CLA n'a pas de propagation de porteuse parce que la sortie de porteuse de l'additionneur de courant est connectée au bit d'entrée de porteuse de l'additionneur suivant. Pour la zone, la RCA nécessitait moins de surface que la CLA, puisque la dernière a une architecture scalaire et qu'elle avait besoin de plus de composants. Étant donné que nous sommes intéressés par un système rapide, le CLA est plus adopté dans notre accélérateur matériel pour concevoir tous les additionneurs, même s'il augmente la surface nécessaire mais sans affecter la surface totale ; de plus, il est simple à concevoir, étant donné que l'architecture CLA utilise la logique des portes de base telles que AND, OR et XOR.

Notre CLA comprend deux niveaux. Le premier niveau est utilisé pour calculer les valeurs partielles (intermédiaires), Gi et Pi, et générer les termes SUM(i) et Ci+1 comme sortie, définis par l'équations 4.21, 4.22, 4.23, et 4.24 qui proposent une section du plan de la CLA. Le deuxième niveau est la conception totale de la CLA (Figure 60), qui calcule chaque porte, la somme totale et indique le débordement de l'addition en sorti.

$$G_i = A(i).B(i). \tag{4.21}$$

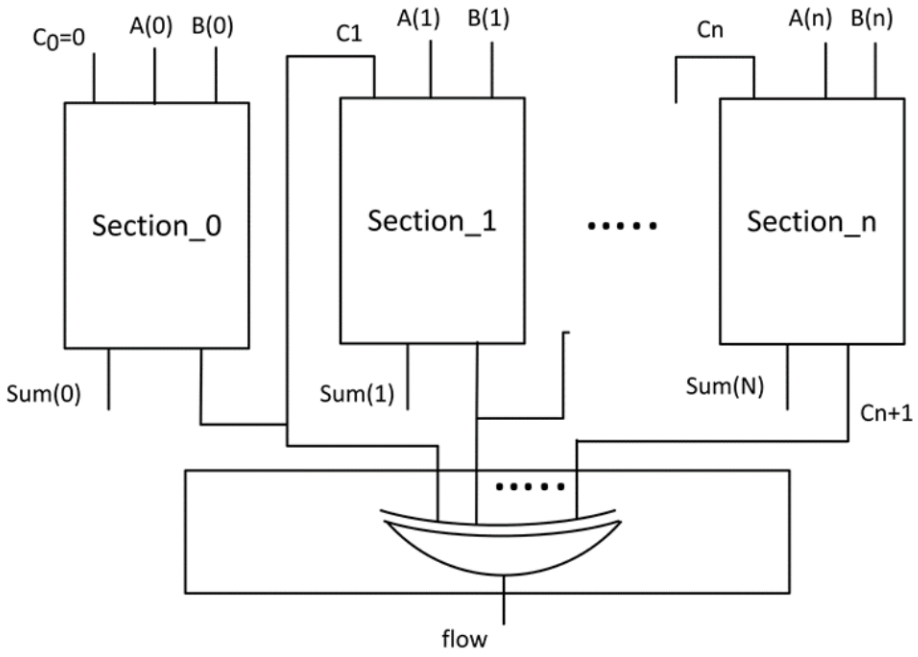


FIGURE 4.24 – Proposed CLA n-bit adder based on architecture of CLA adder.

$$P_i = A(i) \oplus B(i). \tag{4.22}$$

$$C_out_i = G_i + (P_i.C_in_i). \tag{4.23}$$

$$Sum(i) = P_i \oplus C_i. \tag{4.24}$$

Cout_i (ou Cn+1) et Cin_i (ou Cn) correspondent respectivement à la sortie de maintien et à l'entrée de maintien de la section N. Comme l'illustre la figure 4.24, chaque Cout d'une section donnée est connecté à Cin de la suivante. Gn et Pn sont respectivement les porteurs générés et propagés.

Dans la Figure 4.25, notre architecture hautement parallèle commence le traitement à partir de 1024 PEs et arrive au SAD_64×64, différent du cas de [Kulkarni], qui peut calculer différemment la taille des fenêtres. Le SAD_64×64 est la valeur du SAD correspondant au grand bock 64×64, et génère tous ses petits SAD correspondants simultanément. Pour résumer notre approche, on calcule à la fois des blocs plus petits 8×4 et 4×8, présentés par les valeurs SAD_4×8 et SAD_8×4 respectivement. Ils sont calculés par l'addition des huit PE correspondant à chaque bloc. Après avoir calculé tous les blocs plus petits à l'intérieur de 64×64, l'étape suivante calcule la valeur SAD (SAD_8×8) de 8×8 blocs en ajoutant deux blocs plus petits déjà traités. Suivant la même méthode, le bloc est traité à partir des valeurs SAD précédentes pour arriver à la valeur SAD_64×64 du bloc le plus grand. Enfin, le dessin

est disposé dans l'arbre. Un exemple est donné dans la Figure 4.25 montrant que la valeur SAD_8×8 est traitée en ajoutant deux SAD_8×4.

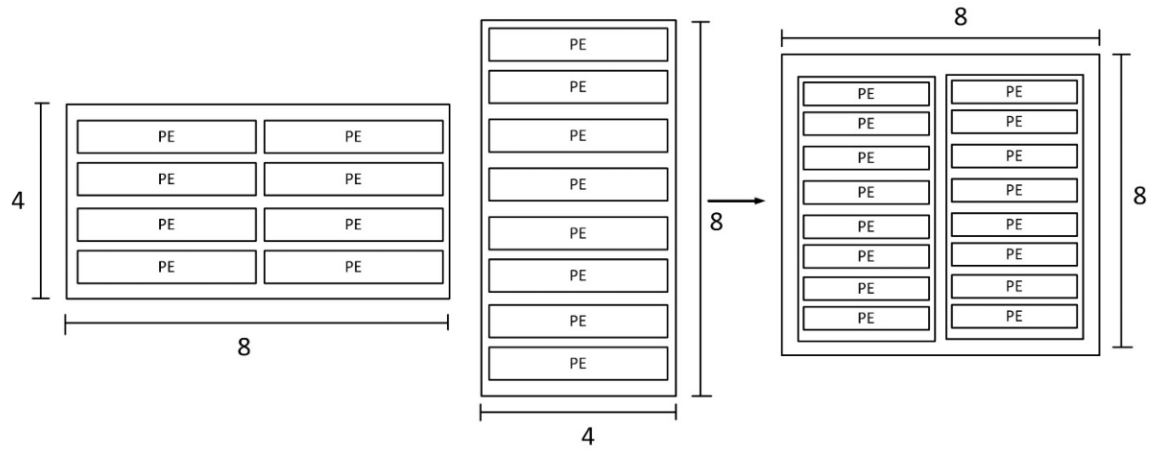


FIGURE 4.25 – Construction des SAD 8×4 et 4×8 à partir de PE et aussi la construction de 8×8 à partir de deux 8×4.

4.5.4 Résultats de l'implémentation

Dans un premier temps, l'architecture proposée pour l'algorithme SAD a été implémentée en langage C++, puis compilée pour être validée et réduire sa complexité. Ensuite, la conception matérielle de l'accélérateur est développée à l'aide du langage de description matérielle VHDL, testée avec Mentor Graphics ModelSim PE (Figure 4.26) avant d'être synthétisée dans le FPGA Artex-7.

Le premier test a été réalisé sur des séquences vidéo d'essai proposées par la norme. Le SAD a été validé à l'aide de trois techniques, la première par le logiciel de référence HEVC HM15.0 version [Noak], la seconde par le logiciel C++ exécuté sous Linux Ubuntu, et la dernière par la conception hardware proposée. Les trois résultats obtenus sont en assez bon accord.

Les trames de la séquence Basketballdrive (1080p) et autres sont utilisées dans ce test, et la valeur de QP est fixée à 32. Pour d'autres configurations de l'exécution, nous avons été inspirés par le travail présenté dans la Réf. [JL15], qui fournit une évaluation des performances du codage distribué du HEVC, et s'est concentré sur les effets de la méthode de partitionnement vidéo sur les performances globales du codage distribué en utilisant un schéma de partitionnement basé sur le GOP afin de mesurer l'efficacité du codeur. Ensuite, le fichier de configuration Random-access only est sélectionné parmi trois configurations (Random-access, low-delay et intra only), car il est le plus adapté à notre cas. Toutes ces caractéristiques et encore plus d'informations sont mentionnées dans le fichier de configuration proposé par la norme en [Bo11].

Le bloc de conception proposé pour l'architecture de l'algorithme SAD dans le HEVC

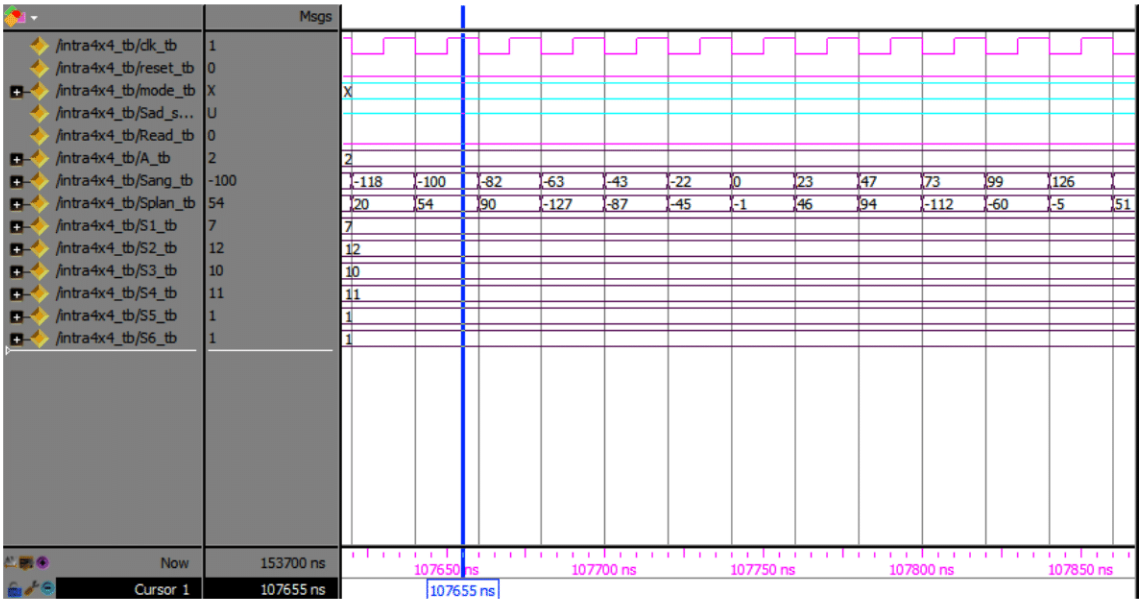


FIGURE 4.26 – la simulation de accélérateur hardware sur le logiciel Modelsim pour le bloc SAD 8x8.

est d'abord codé dans un langage C++ de haut niveau et il est implémenté sur les deux processeurs ARM cortex-a9 intégré dans le dispositif Zynq et Intel Cœuri3 CPU cadencé à 667 MHz et 2,40 GHz, respectivement, pour être validé et réduire la complexité informatique. Deuxièmement, cette conception est également codée en langage VHDL et testée avec Mentor Graphics ModelSim PE, puis l'architecture matérielle prototypée sur Artex-7 FPGA (xc7z020-1clg400) de Xilinx, et le matériel d'accélération a été vérifié sur carte Microzed.

4.5.4.1 Résultats de synthèse et analyse

L'architecture proposée a été synthétisée en utilisant l'environnement de conception récent (Xilinx Vivado 2016.3) comme dans les travaux précédents sur l'algorithme ICT à l'intérieur du codeur HEVC [Noac]. Les résultats de synthèse de notre architecture sont présentés dans le Tableau 23 résultats de la mise en œuvre de l'architecture proposée par le FPGA. La conception fonctionne avec une fréquence maximale jusqu'à 115.15MHz, et elle consomme 32752 LUTs, correspondant à 61.56% des ressources du dispositif (53200 LUTs), et elle utilise 5077 registres (4.77%) sur les 106400 disponibles sur la carte. Il prend 108Ko dans la mémoire embarquée BRAM, et en plus, notre conception exploite les deux multiplexeurs embarqués sur les FPGA, F7 MUX et F8 MUX en 1577 et 481, respectivement. Cependant, il n'a pas besoin de DSPs parmi les 220 DSPs existants dans le FPGA.

La conception proposée a été simulée à l'aide de ModelSim. Il montre que l'architecture peut, en un cycle d'horloge, calculer le SAD pour toutes les tailles de blocs jusqu'à 64×64, incluant l'AMP. Avec la fréquence atteinte de 115,15 MHz, notre conception matérielle peut traiter 30 images par seconde (fps) pour une résolution de 4K calculée à partir du nombre de

TABLE 4.8 – résultats de la mise en œuvre de l'architecture proposée par le FPGA.

Available SAD block	8×4 & 4×8 to 64×64
Slice LUTs	32752
Slice registers	5077
Maximum frequency	115.15Mhz
DSP	0
F7 MUX	1577
F8 MUX	481
Memory	108Kb RAM

points de recherche estimé pour 30fps@4k dans la section II divisé par la fréquence de fonctionnement. Et il peut atteindre jusqu'à 120 images/seconde pour la haute résolution complète calculée de la même manière. La conception peut être améliorée pour atteindre plus que ce nombre d'images par seconde avec un coût de consommation de ressources supplémentaires de l'appareil.

Les résultats de la planification de l'étage de sortie après synthèse sont montrés dans la Figure **??**. Il explique l'emplacement du chemin logique. Les ressources de la zone préciseront les exigences du matériel de l'accélérateur, comme le Block Ram et les LUT. Les résultats de la planification du plancher ont indiqué que la distribution est bien co-localisée sur le fond du FPGA. Par conséquent, il donne les meilleures performances lors de l'exécution des liaisons critiques de synchronisation avec le placement du chemin critique.

Figure Le plan montre l'emplacement du matériel IP sur la puce Zynq. La zone reconfigurable est marquée d'un rectangle bleu.**??**

4.5.4.2 Analyse de puissance

Dans le Tableau 24, la dissipation totale de la puissance sur puce par le SAD de décision en mode IP pour l'estimation du mouvement sur Zynq 7000 Tous les SOC programmables sont de 0,184 W et la température de jonction est de 28,5 °C. La contribution de la puissance dynamique est de 60% de la puissance dissipée totale, et la contribution de la puissance E/S à la puissance dynamique est de 0,008 W.

4.6 Comparaison avec d'autres ouvrages

Dans le Tableau 25, nous présentons les résultats publiés [Reh+09], [NAN13],[Yua+13],[Med+14],[JAS18] pour comparer nos performances. Nous avons sélectionné des travaux connexes sur le FPGA mentionnant les mêmes caractéristiques, telles que les LUT de tranche, les registres de tranche, la fréquence maximale, la fréquence d'images par seconde et le bloc de taille PU utiliser pour le calcul SAD dans chaque travail.

Résumé — Le codage vidéo à haute efficacité (HEVC) est le dernier standard de compression vidéo, qui est appliqué dans de nombreux domaines multimédia. Actuellement, il est principalement utilisé pour l'ultra haute résolution, ce qui conduit à des exigences de calcul élevées du point de vue du temps d'exécution, de l'énergie et de la consommation de la mémoire. Plusieurs travaux ont été entrepris pour optimiser le codec HEVC de façon logicielle et/ou matériel afin de l'adopter dans les systèmes embarqués. Une implémentation détaillée des derniers encodeurs et décodeurs vidéo HEVC sur les processeurs embarqués TMS320C6678 et ARM du processeur de signal numérique (DSP) est proposée dans ce mémoire de thèse. Cette implémentation inclut une étude de l'analyse de complexité en particulier le profilage. Une optimisation basée sur la technologie VLIW (Very Long Instruction Word) et Neon est présentée. Les résultats expérimentaux ainsi mis en évidence sur un seul noyau atteignent une vitesse plus rapide de 4 fois celle mesurée sans optimisation. Pour l'implémentation matériel, trois implémentations sont proposées. Une nouvelle architecture VLSI efficace pour la transformation de cosinus discret entier (DCT) incluse dans le dernier nouveau standard HEVC. Ensuite une architecture matérielle parallèle pour le mode de décision correspondant à la somme absolue de la différence (SAD) de l'estimation du mouvement est proposée. L'estimateur de mouvement étant l'algorithme le plus critique dans la récente norme de codage vidéo HEVC. Enfin, un nouvel accélérateur matériel parallèle et très efficace pour le bloc de prévision intra est validée. Ces trois architectures permettent d'atteindre les exigences temps réel pour la résolution 4K (30 frames/s). De plus ce critère temps réel est validé pour le bloc DCT avec la résolution 8K.

Mots clés : HEVC, Embedded system, FPGA, DSP et ARM.

Abstract — The high efficient video coding (HEVC) is the last standard video compression, which is applied in many multimedia application domains. Currently, it is used mainly for ultra-high resolution that is leaded to high computational requirements from the perspective of the execution time, energy and memory consumption. Several works are appearing that optimize HEVC coding for software and hardware in order to adopt it for the embedded systems. A detailed implementation of the last video encoding and decoding HEVC on both embedded processor Digital Signal processor (DSP) TMS320C6678 and ARM is proposed in this thesis. This implementation includes a study of complexity analysis like profiling. An optimization based on Very Long Instruction Word (VLIW) and Neon technology is adopted. The highlighted experimental results of VLIW and Neon on single core attain a good speedup 4 times compared to the whole execution time without optimization. For the hardware proposition, three implementations are achieved. A new efficient VLSI architecture for Integer Discrete Cosine Transform (DCT) included in the last new HEVC standard is proposed. The second is a new parallel hardware architecture for mode decision corresponding to the Sum Absolute of the Difference (SAD) for motion estimation, which is the most critical algorithm in the recent video encoding standard HEVC. The final one concerns a novel parallel and high efficient hardware accelerator for the intra prediction block. All these architectures work in the real time requests (30 frames/s) for the 4K video resolution. Moreover, the DCT block reaches the real time requirements event at 8K resolution.

Keywords : : HEVC, Embedded system, FPGA, DSP and ARM.

employée. Le codeur doit générer un flux binaire conforme aux exigences de l'application. Il s'agit par exemple de la sélection des outils utilisés pour l'encodage (qui correspondent à un certain profil dans un schéma de codage vidéo standard) ou des contraintes concernant les exigences de tampon côté décodeur et le débit binaire résultant pour la transmission de la vidéo codée. L'optimisation des décisions de commande du codeur est un aspect crucial de la conception du codeur. Au niveau de l'image, la sélection des outils de codage ainsi que l'allocation des bits sur les régions de l'image doivent être prises en compte. Deux implémentations d'encodeurs indépendants conformes à la même spécification de codage vidéo peuvent produire une qualité vidéo reconstruite très différente. Selon le temps disponible pour encoder la vidéo, la quantité et l'effort consacrés à l'optimisation varient. En application temps réel, le codeur ne peut qu'utiliser les informations passées de la vidéo qu'il est en train d'encoder pour l'adapter au contenu. De plus, les exigences de transmission et d'autres événements imprévisibles influencent le processus de codage. Dans les systèmes de production qui préparent la vidéo pour le stockage et la distribution multimédia, la vidéo est traitée pour une distribution à grande échelle à un nombre potentiellement très élevé de clients. Ici, le traitement hors ligne de la vidéo peut être appliqué et un effort sérieux pour l'optimisation de l'encodeur et l'ajustement des paramètres peut être investi. En vue de la transmission vers le côté décodeur, les bits codés sont encapsulés dans un format de transport en fonction des exigences de l'application.

B.3 Transmission

Le bloc de construction de la transmission représente la transmission de la séquence vidéo codée vers le côté récepteur. Cette transmission peut être en temps réel et même bidirectionnelle, par exemple dans des applications conversationnelles. Ici, le délai entre l'émetteur et le destinataire est crucial. Dans les applications de diffusion en direct (comme les événements sportifs), un certain délai peut être tolérable, car aucun retour direct entre le récepteur et l'émetteur n'est généralement nécessaire. Dans les applications de stockage et de distribution vidéo, la transmission peut être totalement découplée dans le temps du processus de production. Ces applications comprennent les systèmes de vidéo à la demande ou le stockage sur disques Dur comme les SD cartes ou les flaches. Les aspects de conception du réseau de transport comprennent l'accessibilité du signal désiré. Ceci est par exemple important dans les applications multiple-canaux comme la télévision où la possibilité d'entrer ou de passer d'un canal à l'autre est une caractéristique importante. Dans les applications conversationnelles où plus de deux participants sont connectés lors d'un appel vidéo commun, l'organisation et la redirection des signaux vidéo vers les bons côtés du récepteur sont nécessaires. Dans les applications de streaming et de stockage, il est souhaitable de disposer de fonctions telles que la recherche rapide en avant ou en arrière. Un autre aspect important est la robustesse du signal de transmission face aux perturbations, distorsions et pertes. Dans les scénarios de transmission par paquets, une transmission sans erreur des paquets est souvent nécessaire. Des distorsions dans le signal de transmission peuvent induire la perte des paquets affectés dans de tels systèmes. Par conséquent, des mécanismes de protection contre les pertes et, éventuellement, la récupération des paquets perdus sont souhaitables.

B.4 Décodage

Le décodeur stocke le flux binaire reçu dans un tampon et reconstruit les données codées en une séquence vidéo dans le format indiqué par le codeur. Dans un système de codage vidéo standard, le processus de décodage suit les procédures normatives définies dans la spécification de codage vidéo sous-adjacente. S'il n'y a pas de pertes, le à la sortie du décodeur correspond exactement à la vidéo telle qu'elle est disponible pour référence côté encodeur. Le flux binaire peut contenir des informations indiquant que seules certaines parties de la vidéo décodée sont affichées. Il peut même être indiqué que des images décodées complètes sont suspendues de l'affichage (alors qu'elles sont potentiellement utilisées pour la prédiction). Un aspect important du contrôle du décodeur est la gestion du tampon pour les images décodées ainsi que pour le flux binaire entrant. Les images décodées dans le tampon d'images décodées peuvent être utilisées comme référence dans le processus de décodage. De plus, dans les systèmes qui fonctionnent à une fréquence d'image définie pour la présentation, il faut affirmer que le processus de décodage rend les images décodées disponibles pour affichage selon les contraintes de temps imposées par le format utilisé.

B.5 Post-traitement

Les images produites par le décodeur peuvent être introduites dans une étape de post-traitement où des opérations d'amélioration de l'image et d'adaptation de l'affichage peuvent être réalisées. Des méthodes de dissimulation des erreurs post-décodeur peuvent également être appliquées. Si le format du matériel vidéo décodé ne correspond pas au format requis pour l'affichage, l'application d'opérations de conversion similaires à celles énumérées pour le pré-traitement est indiquée. Il s'agit par exemple du format couleur, de la résolution de l'image ou de la fréquence d'image.

B.6 Le schéma de codage vidéo hybride

Dans le codage vidéo le schéma hybride est donné sous forme des blocs inter-connectés (la Figure B.3, chaque image est partitionnée en des blocs d'échantillons et des blocs multiples au sein d'une image sont agrégés pour former des tranches en tant qu'entités décodables indépendamment les unes des autres. Tout en adhérant à ce principe de base, la nouvelle norme de codage vidéo à haute efficacité (HEVC) offre un certain nombre de caractéristiques novatrices tant en ce qui concerne l'agrégation d'échantillons pour le partitionnement de blocs que pour le partitionnement d'images par blocs. Cette section décrit d'abord le concept de partitionnement par blocs quadri-blocs du HEVC pour améliorer la prédiction et la transformation, y compris les parties intégrantes des blocs d'arbres de codage (CTB), les blocs de codage (CB), les blocs de prédiction (PB) et les blocs de transformation (TB). De plus, le l'amélioration de l'efficacité du codage pour les différentes configurations du HEVC en ce qui concerne au choix de différentes profondeurs d'arbres et de tailles de blocs pour la prédiction et l'analyse sont

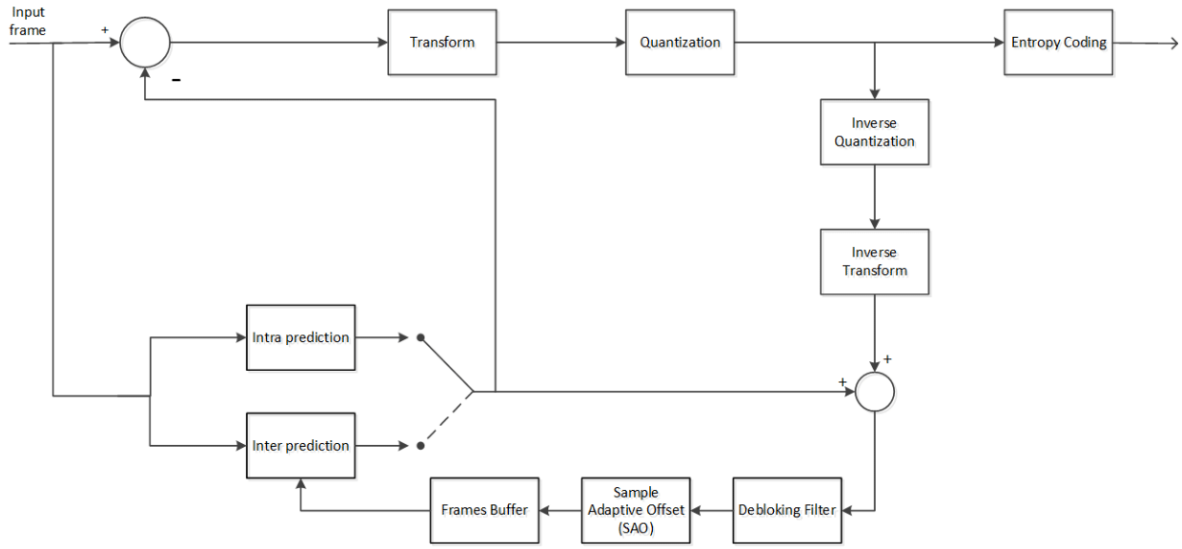


FIGURE B.3 – schéma hybride de encodeur avec un décodeur à l'intérieur de l'encodeur

évaluées. L'un des résultats de cette évaluation expérimentale a été la suivante a observé que plus de la moitié des économies de débit binaire moyennes réalisées par le HEVC par rapport au son prédécesseur H.264/AVC peut être attribué à sa flexibilité accrue de partitionnement de blocs pour la prédiction et le codage par transformation. La deuxième partie de ce chapitre se concentre sur l'amélioration des concepts de partitionnement d'images pour la mise en paquets. À des fins de traitement parallèle dans le HEVC. Cela comprend la discussion sur les nouveaux outils pour la prise en charge du parallélisme de haut niveau, tel que les tuiles et le traitement parallèle de front d'onde (WPP). En outre, le nouveau concept de fragmentation des tranches en segments de tranches dépendants pour l'accès à un flux binaire parallèle et le traitement à très faible retard est présenté ainsi qu'un résumé des avantages et des inconvénients du WPP et des tuiles.

Annexe C

profilage de codec

Plusieurs essais expérimentaux sont menés pour valider le profilage de codec HEVC en se basant sur les packages de Linux afin de présenter les résultats graphiquement. la figure C.1 présente les résultats de l'encodeur, cependant la figure **??** présente de décodeur.

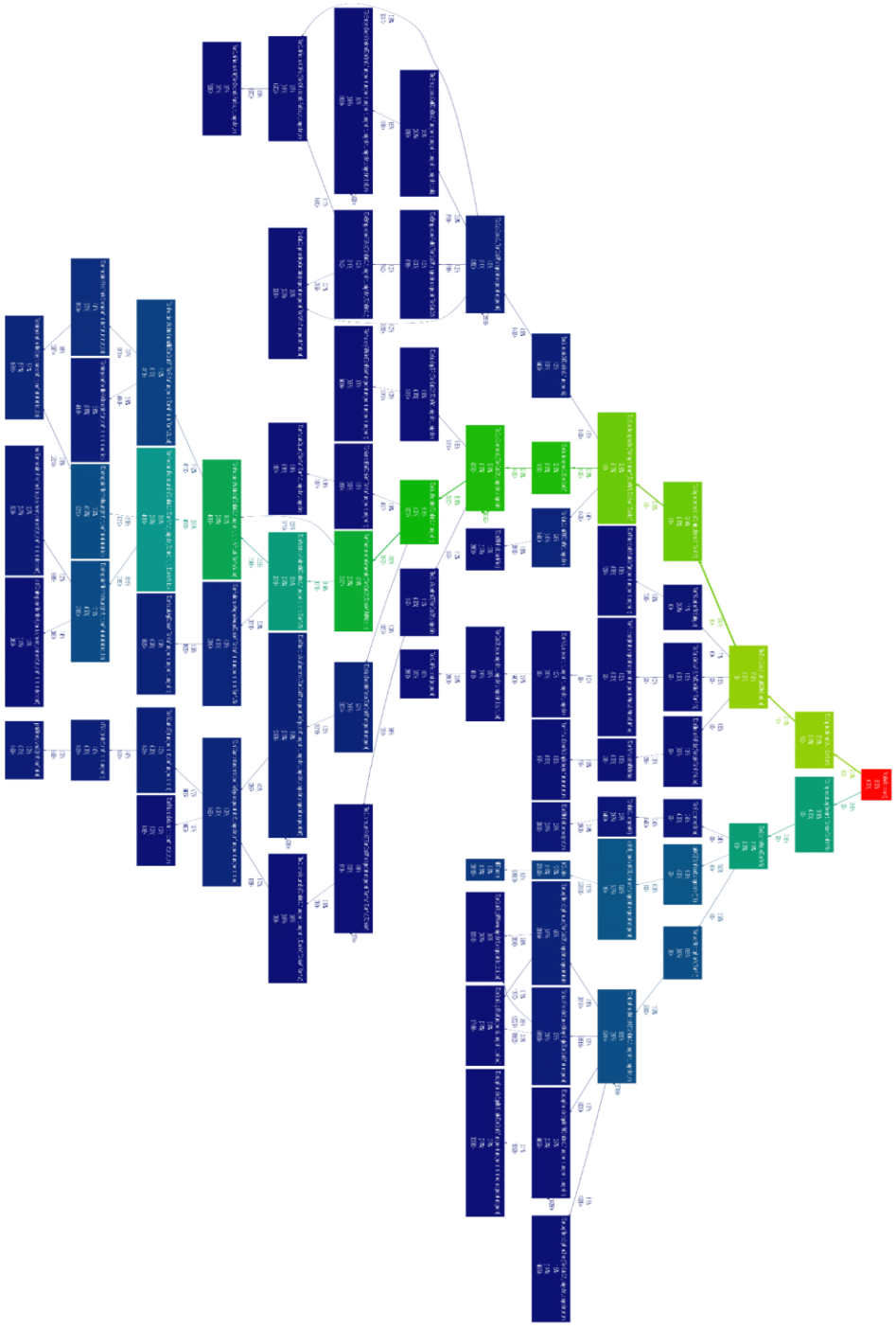


FIGURE C.1 – présentation graphique du profilage de l'encodeur

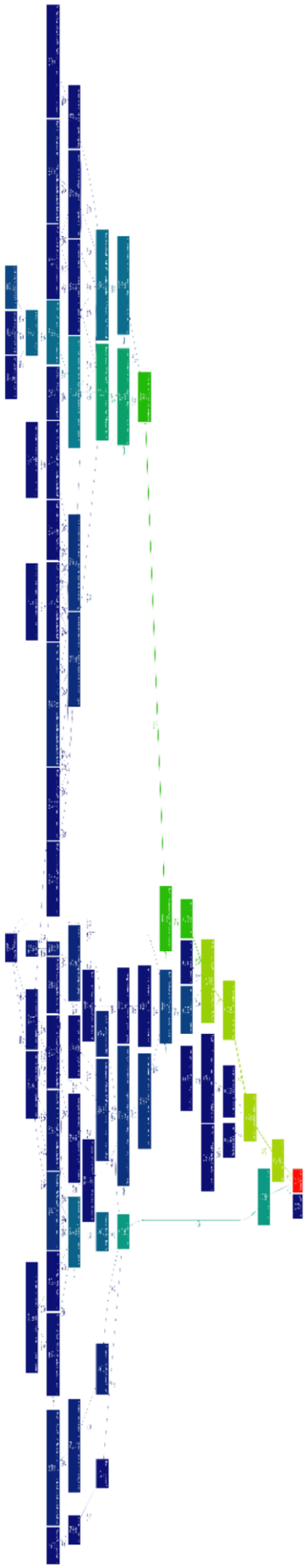


FIGURE C.2 – présentation graphique du profilage de décodeur

Bibliographie

[AA95] Peter M. Athanas et A. Lynn Abbott. « Real-time image processing on a custom computing platform ». In : *Computer* 28.2 (1995), p. 16-25 (cf. p. 27).

[AB16] Farouk Amish et El-Bay Bourennane. « Fully pipelined real time hardware solution for High Efficiency Video Coding (HEVC) intra prediction ». In : *Journal of Systems Architecture* 64 (2016), p. 133-147 (cf. p. 29, 30, 86, 87).

[Abe+16] Maleen Abeydeera et al. « 4K real-time HEVC decoder on an FPGA ». In : *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (2016), p. 236-249 (cf. p. 29, 30, 83, 86, 87).

[AD98] Haitham Akkary et Michael A. Driscoll. « A dynamic multithreading processor ». In : *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society Press, 1998, p. 226-236 (cf. p. 2).

[Ahn+14] Yong-Jo Ahn et al. « Implementation of fast HEVC encoder based on SIMD and data-level parallelism ». In : *EURASIP Journal on Image and Video Processing* 2014.1 (2014), p. 1-19 (cf. p. 27, 28, 62).

[Ali+12] Murtaza Ali et al. « Level-3 BLAS on the TI C6678 multi-core DSP ». In : *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on*. IEEE, 2012, p. 179-186 (cf. p. 8, 29).

[Bo11] Frank Bossen et others. « Common test conditions and software reference configurations ». In : *Joint Collaborative Team on Video Coding (JCT-VC), JCTVC-F900* (2011) (cf. p. 55, 62, 108, 116).

[Bos+12] Frank Bossen et al. « HEVC complexity and implementation analysis ». In : *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), p. 1685-1696 (cf. p. 27-29, 61, 62, 95, 97).

[Cat05] John Catsoulis. *Designing Embedded Hardware : Create New Computers and Devices*. " O'Reilly Media, Inc.", 2005 (cf. p. 25).

[Cha+01] Rohit Chandra et al. *Parallel programming in OpenMP*. Morgan kaufmann, 2001 (cf. p. 2, 52).

[Cha+16] Chia-Wei Chang et al. « A Fast Algorithm-Based Cost-Effective and Hardware-Efficient Unified Architecture Design of 4$\textbackslashtimes$ 4, 8$\textbackslashtimes$ 8, 16$\textbackslashtimes$ 16, and 32$\textbackslashtimes$ 32 Inverse Core Transforms for HEVC ». In : *Journal of Signal Processing Systems* 82.1 (2016), p. 69-89 (cf. p. 31).

[Chi+15] Chi Ching Chi et al. « SIMD acceleration for HEVC decoding ». In : *IEEE Transactions on Circuits and Systems for Video Technology* 25.5 (2015), p. 841-855 (cf. p. 53).

[CHP11] Gordon Clare, F. Henry et S. Pateux. « Wavefront parallel processing for HEVC encoding and decoding ». In : *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-F274, Ver* 2 (2011), p. 14-22 (cf. p. 47).

[CJJ03] Woong IL Choi, Byeungwoo Jeon et Jechang Jeong. « Fast motion estimation with modified diamond search for variable motion block sizes ». In : *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on.* T. 2. IEEE, 2003, p. II-371 (cf. p. 70, 71).

[CM+15] Gabriel Cebrián-Márquez et al. « Accelerating HEVC using heterogeneous platforms ». In : *The Journal of Supercomputing* 71.2 (2015), p. 613-628 (cf. p. 28).

[Col13] Gerald Coley. « Beaglebone black system reference manual ». In : *Texas Instruments, Dallas* (2013) (cf. p. 62).

[Cro+14] Louise H. Crockett et al. *The Zynq Book : Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc.* Strathclyde Academic Media, 2014 (cf. p. 22).

[Cum03] Peter Cumming. « The TI OMAP^{TM} Platform Approach to SoC ». In : *Winning the SOC Revolution.* Springer, 2003, p. 97-118 (cf. p. 26).

[CV94] Roger S. Clist et Robert J. Valkenburg. « Close coupling of a MIMD system to a MAXbus pipeline–the Kiwivision MTM multi-transputer architecture ». In : *Proc. SPIE Machine Vision Applications, Architectures and Systems Integration III* (1994) (cf. p. 26).

[DL00] M. J. Duff et James T. Liu. « Complementarity of the Maldacena and Randall-Sundrum pictures ». In : *Physical Review Letters* 85.10 (2000), p. 2052 (cf. p. 26).

[DM15] Anand D. Darji et Raviraj P. Makwana. « High-performance multiplierless DCT architecture for HEVC ». In : *VLSI Design and Test (VDAT), 2015 19th International Symposium on.* IEEE, 2015, p. 1-5 (cf. p. 31).

[DSRS14] Diego F. De Souza, Nuno Roma et Leonel Sousa. « Cooperative CPU+ GPU deblocking filter parallelization for high performance HEVC video codecs ». In : *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, p. 4993-4997 (cf. p. 29).

[EAAM17] Abdessamad El Ansari, Ali Ahaitouf et Anass Mansouri. « An implementation comparison of the HEVC encoder on two embedded processors ». In : *Wireless Technologies, Embedded and Intelligent Systems (WITS), 2017 International Conference on.* IEEE, 2017, p. 1-3 (cf. p. 96).

[Eng+14] Denis Engelhardt et al. « FPGA implementation of a full HD real-time HEVC main profile decoder ». In : *IEEE Transactions on Consumer Electronics* 60.3 (2014), p. 476-484 (cf. p. 29, 30, 85-87).

[GB13] Yang Gao et Jason D. Bakos. « Sparse matrix-vector multiply on the texas instruments c6678 digital signal processor ». In : *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*. IEEE, 2013, p. 168-174 (cf. p. 8).

[HHW17] Jui-Hung Hsieh, Jian-Hao Huang et Hung-Ren Wang. « DVFS-aware motion estimation design scheme based on bandwidth–rate–distortion optimization in application processor systems ». In : *Integration, the VLSI Journal* 57 (2017), p. 74-80 (cf. p. 32, 95).

[HM85] Dorit S. Hochbaum et Wolfgang Maass. « Approximation schemes for covering and packing problems in image processing and VLSI ». In : *Journal of the ACM (JACM)* 32.1 (1985), p. 130-136 (cf. p. 26).

[HTC14] Chao-Tsung Huang, Mehul Tikekar et Anantha P. Chandrakasan. « Memory-hierarchical and mode-adaptive HEVC intra prediction architecture for quad full HD video decoding ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.7 (2014), p. 1515-1525 (cf. p. 29, 30, 87).

[Ins11] Texas Instruments. *TMS320C66X DSP CorePac User Guide*. 2011 (cf. p. 18).

[JAS18] Amit M. Joshi, Mohd Samar Ansari et Chitrakant Sahu. « VLSI Architecture of High Speed SAD for High Efficiency Video Coding (HEVC) Encoder ». In : *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, p. 1-4 (cf. p. 110, 112).

[JL15] Myunghoon Jeon et Byoung-Dai Lee. « Toward Content-Aware Video Partitioning Methods for Distributed HEVC Video Encoding ». In : *International Journal of Electrical and Computer Engineering (IJECE)* 5.3 (2015), p. 569-578 (cf. p. 108).

[JMC12] Wei Jiang, Hanjie Ma et Yaowu Chen. « Gradient based fast mode decision algorithm for intra prediction in HEVC ». In : *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*. IEEE, 2012, p. 1836-1840 (cf. p. 29, 30, 85-87).

[Ju+16] Chi-Cheng Ju et al. « A 0.5 nJ/pixel 4 K H. 265/HEVC codec LSI for multi-format smartphone applications ». In : *IEEE Journal of Solid-State Circuits* 51.1 (2016), p. 56-67 (cf. p. 28).

[Jun+16] Dongsan Jun et al. « Development of an ultra-HD HEVC encoder using SIMD implementation and fast encoding schemes for smart surveillance system ». In : *The Journal of Supercomputing* (2016), p. 1-21 (cf. p. 27, 62).

[KAH14] Ercan Kalali, Yusuf Adibelli et Ilker Hamzaoglu. « A low energy intra prediction hardware for high efficiency video coding ». In : *Journal of Real-Time Image Processing* (2014), p. 1-14 (cf. p. 29-31, 87).

[KAM14] Manel Kammoun, Ahmed Ben Atitallah et Nouri Masmoudi. « An optimized hardware architecture for intra prediction for HEVC ». In : *Image Processing, Applications and Systems Conference (IPAS), 2014 First International*. IEEE, 2014, p. 1-5 (cf. p. 29-31, 87).

[KG06] Nasser Kehtarnavaz et Mark Gamadia. « Real-time image and video processing : from research to reality ». In : *Synthesis Lectures on Image, Video & Multimedia Processing* 2.1 (2006), p. 1-108 (cf. p. 24, 25).

[Kha+13] Muhammad Usman Karim Khan et al. « Hardware-software collaborative complexity reduction scheme for the emerging HEVC intra encoder ». In : *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 2013, p. 125-128 (cf. p. 29, 30, 85-87).

[Kim+12a] Il-Koo Kim et al. « Block partitioning structure in the HEVC standard ». In : *IEEE transactions on circuits and systems for video technology* 22.12 (2012), p. 1697-1706 (cf. p. 101, 103, 105).

[Kim+12b] Il-Koo Kim et al. « Hm9 : High efficiency video coding (HEVC) test model 9 encoder description ». In : *Proc. 9th JCT-VC Meeting*. 2012, p. 6-11 (cf. p. 76).

[KL09] John Kalomiros et John Lygouras. « Comparative study of local SAD and dynamic programming for stereo processing using dedicated hardware ». In : *EURASIP Journal on advances in Signal Processing* (2009) (cf. p. 32).

[KMG06] Shashi Kant, U. Mithun et PSSBK Gupta. « Real time H. 264 video encoder implementation on a programmable DSP processor for videophone applications ». In : *Consumer Electronics, 2006. ICCE'06. 2006 Digest of Technical Papers. International Conference on*. IEEE, 2006, p. 93-94 (cf. p. 8).

[Lai+12] Jani Lainema et al. « Intra coding of the HEVC standard ». In : *Circuits and Systems for Video Technology, IEEE Transactions on* 22.12 (2012), p. 1792-1801 (cf. p. 38, 42, 76).

[LCS08] Wonchul Lee, Hyojin Choi et Wonyong Sung. « Algorithm and software optimization of variable block size motion estimation for H. 264/AVC on a VLIW–SIMD DSP ». In : *Journal of Signal Processing Systems* 51.3 (2008), p. 289-302 (cf. p. 28).

[LGI08] Zhenyu Liu, Satoshi Goto et Takeshi Ikenaga. « Optimization of Propagate Partial SAD and SAD tree motion estimation hardwired engine for H. 264 ». In : *Computer Design, 2008. ICCD 2008. IEEE International Conference on*. IEEE, 2008, p. 328-333 (cf. p. 32, 100).

[MBD18] Mourad Moussa, Nesrine Bdioui et Ali Douik. « Motion Detection and Clustering Using PCA and NN in Color Image Sequence ». In : Tel*KOMNIKA* 16.2 (2018), p. 747-754 (cf. p. 32).

[McC04] David McCullough. « uCLinux for Linux programmers ». In : *Linux Journal* 2004.123 (2004), p. 7 (cf. p. 2).

[McC94] Charles L. McCarthy. « A Low-Cost Audio/Video Decoder Solution for MPEG System Streams ». In : *Consumer Electronics, 1994. Digest of Technical Papers., IEEE 1994 International Conference on*. IEEE, 1994, p. 312-313 (cf. p. 8).

[Med+14] Ahmed Medhat et al. « A highly parallel SAD architecture for motion estimation in HEVC encoder ». In : *Circuits and Systems (APCCAS), 2014 IEEE Asia Pacific Conference on*. IEEE, 2014, p. 280-283 (cf. p. 32, 33, 110, 111).

[MR16] P. Muralidhar et C. B. Ramarao. « Efficient architecture for global elimination algorithm for H. 264 motion estimation ». In : *Sadhana* 41.1 (2016), p. 47-54 (cf. p. 111, 112).

[MXC16] Biao Min, Zhe Xu et Ray CC Cheung. « A Fully Pipelined Hardware Architecture for Intra Prediction of HEVC ». In : *IEEE Transactions on Circuits and Systems for Video Technology* (2016) (cf. p. 29, 30, 84, 86, 87).

[NAN13] Purnachand Nalluri, Luis Nero Alves et Antonio Navarro. « A novel SAD architecture for variable block size motion estimation in HEVC video coding ». In : *System on Chip (SoC), 2013 International Symposium on*. IEEE, 2013, p. 1-4 (cf. p. 32, 33, 110, 111).

[Noaa] *.. : : calibre bibliothèque : :.. Livres correspondants* (cf. p. 1).

[Noab] *.. : : calibre bibliothèque : :.. Livres correspondants* (cf. p. 1).

[Noac] *AM3358 Sitara Processor \textbar TI.com*. Mar. 2017 (cf. p. 11, 67, 109).

[Noad] *AR# 46913 : Zynq-7000 Example Design - Program the PL using the Linux driver for DEVCFG* (cf. p. 125).

[Noae] *Architecting a Smarter World – Arm* (cf. p. 10).

[Noaf] *H.265 : Codage vidéo à haute efficacité* (cf. p. 37, 53).

[Noag] *High Efficiency Video Coding | MPEG* (cf. p. 37).

[Noah] *HM-15.0 dans tags. – JCT-VC HEVC*. Jan. 2017 (cf. p. 62).

[Noai] *Intel® FPGAs and Programmable Devices - Intel® FPGA*. en (cf. p. 20).

[Noaj] *MicroZed | Zedboard* (cf. p. 4, 22, 23).

[Noak] *svn_HEVCSoftware - Revision 4598 : /tags/HM-15.0*. Sept. 2015 (cf. p. 96, 97, 108, 116).

[Noal] *TMDSEVM6678 TMS320C6678 Evaluation Modules \textbar TI.com*. Mar. 2017 (cf. p. 17).

[Noam] *TMS320C6678 Evaluation Modules - TMDSEVM6678 - TI Tool Folder*. Sept. 2015 (cf. p. 4, 14, 67).

[Noan] *Zynq-7000 All Programmable SoC*. Jan. 2017 (cf. p. 22, 114).

[Noao] *Zynq Linux - Xilinx Wiki - Confluence* (cf. p. 22, 125).

[Pes+13] Fernando Pescador et al. « Complexity analysis of an HEVC decoder based on a digital signal processor ». In : *Consumer Electronics, IEEE Transactions on* 59.2 (2013), p. 391-399 (cf. p. 29, 60, 61).

[Raj+11] Vidya Rajagopalan et al. « Xilinx zynq-7000 epp : An extensible processing platform family ». In : *Hot Chips 23 Symposium (HCS), 2011 IEEE*. IEEE, 2011, p. 1-24 (cf. p. 62, 114).

[Ram+14] Barath Ramesh et al. « Optimization and evaluation of image-and signal-processing kernels on the TI C6678 multi-core DSP ». In : *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 2014, p. 1-6 (cf. p. 31).

[Reh+09] Saad Rehman et al. « An FPGA based generic framework for high speed sum of absolute difference implementation ». In : *European Journal of Scientific Research* 33.1 (2009), p. 629 (cf. p. 32, 110, 111).

[RLN05] Pichai Raghavan, Amol Lad et Sriram Neelakandan. *Embedded Linux system design and development*. Auerbach Publications, 2005 (cf. p. 125).

[Sam+06] Amine Samet et al. « New horizontal diamond search motion estimation algorithm for H. 264/AVC ». In : *Second Symposium on Communication, Control and Signal Processing, ISCCSP*. 2006, p. 13-15 (cf. p. 70, 72).

[SB12] Vivienne Sze et Madhukar Budagavi. « High throughput CABAC entropy coding in HEVC ». In : *Circuits and Systems for Video Technology, IEEE Transactions on* 22.12 (2012), p. 1778-1791 (cf. p. 46).

[Sha98] Ashok K. Sharma. *Programmable Logic Handbook : PLDs, CPLDs, and FPGAs*. McGraw-Hill New York, 1998 (cf. p. 13, 19).

[SJL16] Shiming Sun, Hongxu Jiang et Bo Li. « An efficient Markov chain-based data prefetching for motion estimation of HEVC on multi-core DSPs ». In : *Multimedia Tools and Applications* 75.13 (2016), p. 8019-8043 (cf. p. 28).

[SKS09] Raminder Preet Pal Singh, Parveen Kumar et Balwinder Singh. « Performance analysis of 32-bit array multiplier with a carry save adder and with a carry-look-ahead adder ». In : *International Journal of Recent Trends in Engineering* 2.6 (2009), p. 83-86 (cf. p. 79).

[Sul+12] Gary J. Sullivan et al. « Overview of the high efficiency video coding (HEVC) standard ». In : *IEEE Transactions on circuits and systems for video technology* 22.12 (2012), p. 1649-1668 (cf. p. 5).

[TMS00] CPU TMS320C6000. « Instruction Set Reference Guide ». In : *Texas Instruments Inc., Dallas, TX* (2000) (cf. p. 9, 53).

[TTC14] Sung-Fang Tsai, Cheng-Han Tsai et Liang-Gee Chen. « Encoder Hardware Architecture for HEVC ». In : *High Efficiency Video Coding (HEVC)*. Springer, 2014, p. 343-375 (cf. p. 28).

[Ung58] Stephen H. Unger. « A computer oriented toward spatial problems ». In : *Proceedings of the May 6-8, 1958, western joint computer conference : contrasts in computers*. ACM, 1958, p. 234-239 (cf. p. 26).

[Van+12] Jarno Vanne et al. « Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs ». In : *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), p. 1885-1898 (cf. p. 69).

[Vii+12] Marko Viitanen et al. « Complexity analysis of next-generation HEVC decoder ». In : *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012, p. 882-885 (cf. p. 61).

[WDB12] Fábio Leandro Walter, Cláudio Machado Diniz et Sergio Bampi. « Synthesis and comparison of low-power high-throughput architectures for SAD calculation ». In : *Analog Integrated Circuits and Signal Processing* 73.3 (2012), p. 873-884 (cf. p. 32, 100).

[Wer+10] Imen Werda et al. « Optimal DSP Based Integer Motion Estimation Implementation for H. 264/AVC Baseline Encoder. » In : *Int. Arab J. Inf. Technol.* 7.1 (2010), p. 96-104 (cf. p. 70, 73).

[Wie15] Mathias Wien. « High efficiency video coding ». In : *Coding Tools and specification* (2015), p. 133-160 (cf. p. 38, 39).

[Yan+12] Leju Yan et al. « Implementation of HEVC decoder on x86 processors with SIMD optimization ». In : *Visual Communications and Image Processing (VCIP), 2012 IEEE*. IEEE, 2012, p. 1-6 (cf. p. 28, 53).

[Yua+13] Xu Yuan et al. « A high performance VLSI architecture for integer motion estimation in HEVC ». In : *ASIC (ASICON), 2013 IEEE 10th International Conference on*. IEEE, 2013, p. 1-4 (cf. p. 32, 33, 110, 111).

[YXQ07] Li Yufei, Feng Xiubo et Wang Qin. « A high-performance low cost SAD architecture for video coding ». In : *IEEE Transactions on Consumer Electronics* 53.2 (2007) (cf. p. 99, 101, 103, 105).

[ZAR17] Nurulnajah Mohd Zabidi et Ab Al-Hadi Ab Rahman. « VLSI Design of a Fast Pipelined 8x8 Discrete Cosine Transform ». In : *International Journal of Electrical and Computer Engineering (IJECE)* 7.3 (2017), p. 1430-1435 (cf. p. 33).

[ZBG88] Hans P. Zima, Heinz-J. Bast et Michael Gerndt. « SUPERB : A tool for semi-automatic MIMD/SIMD parallelization ». In : *Parallel computing* 6.1 (1988), p. 1-18 (cf. p. 8).

[Zho+15] Dajiang Zhou et al. « Ultra-high-throughput VLSI architecture of H. 265/HEVC CABAC encoder for UHDTV applications ». In : *IEEE Transactions on Circuits and Systems for Video Technology* 25.3 (2015), p. 497-507 (cf. p. 93).

rouge, vert et bleu ; en bas : les composantes rouge, vert et bleu de l'image couleur en haut à droite. Cependant, dans la Figure B.2 l'espace couleur YCbCr. En haut à gauche : le plan de couleur Cb Cr au milieu de la luminance ; en bas : les composantes de luminance et de chrominance de l'image couleur en haut à droite.



FIGURE B.1 – Espace couleur RVB. En haut à gauche : combinaison des couleurs primaires rouge, vert et bleu ; en bas : les composantes rouge, vert et bleu de l'image couleur en haut à droite.



FIGURE B.2 – Décomposition de l'image en composantes : luminance(Y) et chrominance (U et V)

B.2 Encodage

L'étage codeur transforme la séquence vidéo d'entrée en représentation codée d'un flux binaire. Le flux binaire dans les normes de codage vidéo contemporaines à une structure orientée paquets où l'information codée est organisée hiérarchiquement en fonction de la spécification

TABLE 4.9 – Dissipation totale de la puissance sur puce (statique et dynamique) par conception pour la décision en mode SAD avec plus de statistiques.

6***Dynamic (60%)**	Clocks	0.004 W	2%
	Signal	0.097 W	53%
	logic	0.068 W	37%
	BRAM	0.007 W	4%
	I/O	0.008 W	4%
	Total	**0184 W**	100%
Static (40%)	PL Static	0.122 W	100%
Total on chip power (100%)		0.306 W	100%

TABLE 4.10 – LES RÉSULTATS DE SYNTHÈSE DE L'ARCHITECTURE PROPOSÉE

	notre	**nalluri [NAN13]**	**yuan [Yua+13]**	**medhat [Med+14]**	**muralidhar [MR16]**	**rehman [Reh+09]**
Technologie	28 nm Artix-7	65 nm Virtex-5	40 nm Virtex-6	28 nm Virtex-7	90 nm Virtex-4	120 nm Virtex 2
LUTs	33K	15K	55k	39K	12K	657
Registers	5K	20K	19K	24K	7K	-
Max fréquence	115.15Mhz	171.94Mhz	110Mhz	458.7Mhz	32Mhz	133Mhz
Serach window	±64	-	±24	±20	±16	-
Frames rate	4K@30fps	-	FHD@30fps	2K@30fps	HD@5.34	1024 x 1024@127fps
PUs	ALL	ALL	4×8 to 32×32	ALL	32×32	4×4

Dans la réf. [NAN13], toute l'architecture matérielle de l'algorithme SAD est présentée, afin de comparer nos performances avec ce travail, nous avons limité la comparaison à la surface consommée obtenue dans le Virtex-5 65 nm car les auteurs n'indiquent pas comment dans leur conception les nombres de trames peuvent être traités. Leur fréquence maximale est donc environ 1,5 fois la nôtre. Pour les LUTs, la solution qu'ils proposent est 18K de moins que notre travail, mais les résultats pour les registres montrent qu'ils ont 15K de plus que nos résultats [Yua+13], Yuan et al. présentent une architecture SAD parallèle synthétisée sur Virtex-6 40nm. Ils ont utilisé plus de ressources et de registres de LUT que dans notre cas, mais ils ont rapporté que une fréquence maximale basse, seulement 110 MHz. Le nombre d'images par seconde peut aller jusqu'à 30 images/seconde en Full HD. (1080p). Medhat el al. [Med+14] sur un FPGA Virtex-7 avec 28nm, la même technologie utilisée dans ce travail, a rapporté une autre conception parallèle. Même si leurs résultats semblent comparables aux nôtres, notre conception prend 118% de LUTs en moins que leur travail, cependant, le résultat concernant les registres consommant montre qu'ils ont 380% de plus que nos résultats avec

une fréquence maximale inférieure (398% de plus que notre fréquence). Ils exploitent cinq fois plus de registres de tranches, et ils peuvent traiter 30 images/seconde pour une résolution de 2K, en utilisant seulement une petite fenêtre de recherche, 104x104 pixels. Par rapport à notre architecture, Joshi et al [JAS18] ont présenté une architecture pour le SAD en standard HEVC. il utilise une conception en pipeline avec traitement parallèle avec 16 unités de traitement afin de réduire le temps de calcul de l'exécution et d'augmenter les performances. La conception matérielle est prototypée et simulée sur Xilinx Virtex-5 FPGA pour la famille XC5VLX20T. Elle est également synthétisée sur la technologie 65 nm et atteint une fréquence maximale de 475,21 MHz. Ils n'ont pas donné le nombre de trames codées, et la fréquence la plus élevée obtenue a conduit à une plus grande consommation d'énergie. Ils n'ont atteint que 127 images/seconde pour une résolution d'image de 1024x1024, mais leur conception est toujours limitée à la résolution moyenne. Muralidhar et Ramaro [MR16] font état d'une zone moins consommée (LUT et registre) sur le FPGA Virtex-4, mais ils sont restés si loin du traitement en temps réel, avec un taux de seulement 5,32 fps pour la résolution HD ; leur circuit est encore bien adopté pour la norme H.264/AVC de codage vidéo. Par rapport aux œuvres existantes, notre conception présente une bonne performance qui explore le parallélisme de l'architecture matérielle, ce qui permet d'atteindre un haut débit pour un encodage 4K en temps réel. Par conséquent, notre circuit présente un bon codage et une bonne efficacité en ce qui concerne la consommation de ressources.

4.6.1 Conclusion

Dans cette section, nous avons conçu un nouvel accélérateur haute parallèle pour l'implémentation matérielle de l'algorithme SAD. Il s'agit du nouveau standard HEVC, qui est implémenté sur un FPGA Artix-7 Xilinx Zynq-7000. Le design est parallèle et permet l'accélération de l'estimation du mouvement entier sans modification de l'estimation du mouvement algorithme. L'architecture proposée a une fréquence de fonctionnement maximale jusqu'à 115,15 MHz et peut traiter les données de l des séquences vidéo 4K bien connues avec une haute résolution 38402160 dans le cadre des exigences du temps réel. Par rapport aux travaux précédents, cette conception présente une architecture à grande vitesse et à faible coût de surface adaptée aux encodeurs vidéo sur une large gamme d'applications vidéo. La conception est simple et parfaitement adaptée au codeur HEVC. Dans le prochain travail, notre architecture proposée servira de moteur pour le prochain circuit proposé pour le bloc d'inter prédiction d'un codeur HEVC, en conservant le traitement en temps réel pour des résolutions supérieures, 8K et 4K.

4.7 Co-désigne entre système logiciel et matériel

4.7.1 Introduction

Au cours des dernières années, une large gamme de systèmes multiprocesseurs a été développée pour les systèmes embarqués avec un grand succès sur le marché en raison de leur

capacité de calcul relativement élevée, leur faible coût et leur consommation d'énergie réduite. Ce type de systèmes s'ajoute à ceux qui ont un parcours historique plus long, comme ceux qui intègrent des systèmes d'ordinateurs personnels ou de PC, et qui offrent une plus grande gamme de caractéristiques, mais avec un coût plus élevé et une consommation d'énergie élevée pour être utilisés dans des terminaux portables.

D'autre part, la tendance commerciale dominante au cours de la dernière décennie a été le lancement continu de nouveaux terminaux multimédias, tels que les smart-phones ou les tablettes, intégrés par des systèmes intégrés. De plus, la conception d'applications et d'interfaces utilisateurs pour ce type de plate-forme a vu ses délais de développement raccourcis en raison des délais commerciaux exigeants.

Concevoir des applications pour fonctionner sur des plates-formes (ARM + FPGA ou ARM +DSP) multi-cœurs pose un certain nombre de défis. Dans de nombreux cas, la complexité des implémentations à franchir dépend à la fois des difficultés posées par le(s) système(s) à travailler et des applications à soutenir.

Dans ce contexte, la recherche de méthodologies fournissant des solutions pour le développement de logiciels compatibles avec les systèmes multiprocesseurs dans des délais de plus en plus courts est devenue l'objectif à atteindre tant par l'industrie que par les universités.

4.7.2 Accélérateur matériel sur SOC Zynq

Les travaux de fin d'année en cours portent sur la conception, le développement et la mise à l'essai d'un système de capture simultanée d'images provenant de plusieurs caméras, avec sortie HDMI, sur un SoPC ("System on Programmable Chip") avec Linux intégré. Son objectif final est de servir de base au développement des futurs travaux de capacités au traitement des codages vidéos.

Les travaux s'inscrivent dans un contexte de forte croissance de la vision par ordinateur et de développement de systèmes sur du matériel reconfigurable (FPGA). Traditionnellement, les algorithmes de vision par ordinateur, d'une grande complexité informatique et donc nécessitant une grande puissance de calcul, ont été développés en utilisant des solutions purement logicielles. Cependant, la demande croissante pour les projets informatiques vision qui nécessitent des plates-formes de développement plus rapides et plus robustes a conduit ce domaine à rechercher des solutions matérielles, et dans notre cas, la technologie SoPC de Xilinx (Zynq).

La technologie SoPC Zynq unifie le meilleur du logiciel et du matériel, en fournissant un microprocesseur connecté à un FPGA, permettant de développer le contrôle de l'application en logiciel, de moindre exigence technique et prix, et que ce logiciel interagisse avec le matériel pour le traitement des parties critiques du projet.

L'implémentation qui a été utilisée de SoPC Zynq dans ce projet est microzed de Diligent. La carte microzed est basée sur la puce Xilinx ZYNQ ZYNQ XC7Z020-1CLG400C, qui contient un processeur Cortex-A9 double cœur et un FPGA de la famille Artix-7 ainsi que le

système de communication entre la connexion périphérique et la mémoire RAM.

Récemment, le Xilinx a développé une nouvelle série de SOC (System-on-a-Chip) de puce programmable appelée Zynq- 7000 [Raj+11] qui est incluse dans un processeur FPGA et ARM. La microZed est une plate-forme Zynq équipée du SOC Zynq-7020 [Raj+11]. Cette carte est utilisée pour les applications qui ont en général un traitement matériel de propriété intellectuelle (IP) contrôlé par le système logiciel. Le processeur ARM intégré dans cette plate-forme est le cortex-A9 avec une fréquence allant jusqu'à 667 MHz.

Le Zynq-7000 est un SOC programmable de Xilinx. Il comprend un système de programmation dédié à la programmation logicielle basé sur un double processeur ARM, bien qu'il dispose d'un matériel de programmation sur FPGA de fabrication sur la technologie de processus 28 nm que les deux ont une communication système avec eux basée sur un bus AXI. Le FPGA dispose de 53200 LUT, 106400 FF, 220 DSP, 140 BRAM et 125 IO comme indiqué sur la figure 2.6 .

4.7.3 La plateforme microzed

Dans le projet en cours, une carte Diligent microzed **e**st utilisée. Cette carte a dans son noyau la puce ZYNQ XC7Z020-1CLG400C **d**e Xilinx, appartenant à la gamme des dispositifs basés sur PcSOC de Xilinx et plus spécifiquement à la famille des Zynq-700.

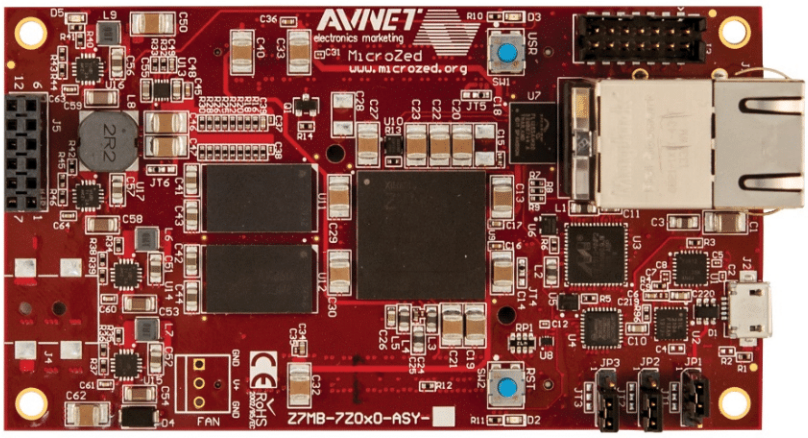


FIGURE 4.27 – la plateforme microzed .

$$89\,[01011001_b]\,.x_i = (x_i \ll 6) + (x_i << 4) + (x_i << 3 + (x_i)) \tag{4.25}$$

Pour terminer cette section, il faut noter que la carte Microzed a été choisie principalement en raison de son faible coût, 300 dollars, et des plusieurs périphériques disponibles, notamment la disponibilité des ports USB et Ethernet.

Le cortex-A9 de l'ARM dispose d'un ensemble d'instructions comme l'ARMv7. Par exemple,

il est connu pour sa faible consommation d'énergie, ce qui le rend bien adapté à divers appareils comme les smartphones, les tablettes et la télévision numérique. L'architecture NEON est une SIMD pour ARM cortex-A9 qui a un très long registre configuré en 64 bit /128 bit. Cette technologie a été proposée pour accélérer l'algorithme de traitement du signal, elle fonctionne parfaitement avec ce pipeline et le parallélisme fonctionnant sur plusieurs données en utilisant la même instruction. Cette architecture a deux types de registres, Q et D. Seize registres Q de 128 bits, appelés Q0-Q15.

Comme le montre la Figure 4.28, le matériel IP a été intégré dans le FPGA et testé sur une carte Microzed comprenant un matériel système et un logiciel système, comme le montre la figure 2.6, un double processeur dur ARM cortex a9 (logiciel partiel) couplé au FPGA (matériel partiel) a été programmé avec le logiciel HM moins que la prédiction intra pour configurer l'architecture du matériel test. En raison de son logiciel de référence n'est pas en mesure d'encoder les trames de la résolution 4K et d'avoir une étude en temps réel en FPGA. Le port d'accès de test du matériel de l'accélérateur a été exploité les mires de test de l'HM à partir de la BRAM existant sur la carte et de taille 1 Go.

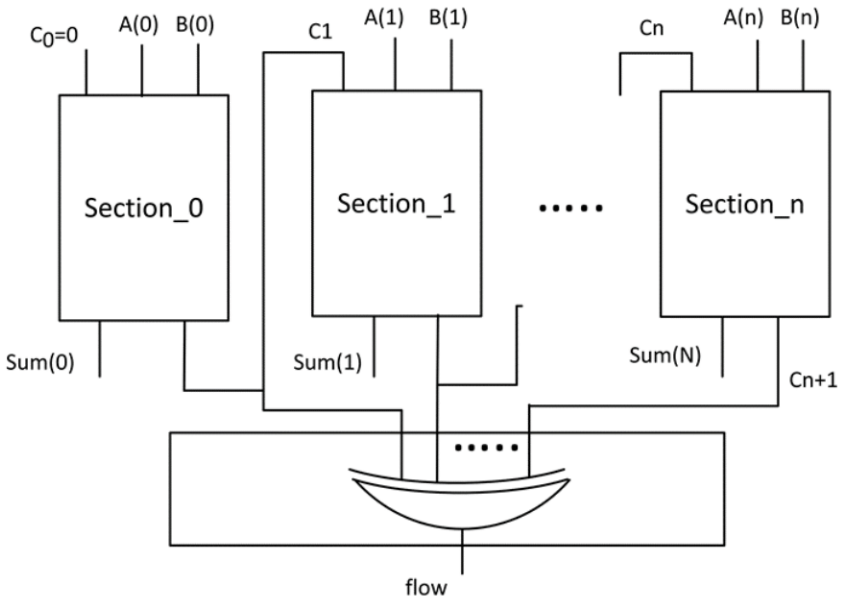


FIGURE 4.28 – Proposed CLA n-bit adder based on architecture of CLA adder.

Le système général de validation comprend une mémoire interne et une mémoire externe. La mémoire interne est une mémoire intégrée dans le FPGA qui peut être de la RAM bloquée d'une part, ou dans des registres discrets et des registres à décalage d'autre part. Par conséquent, la sélection entre eux fait dans la plupart des cas au cours de la langue de programmation de la conception du circuit et en moins par les outils de forme Xilinx, qui est mappé chaque type de mémoire dans la conception. La mémoire externe est DDR connectée via une interface bus AXI comme accès à la mémoire système.

4.7.4 Proposition du système co-désigne logiciel matériel

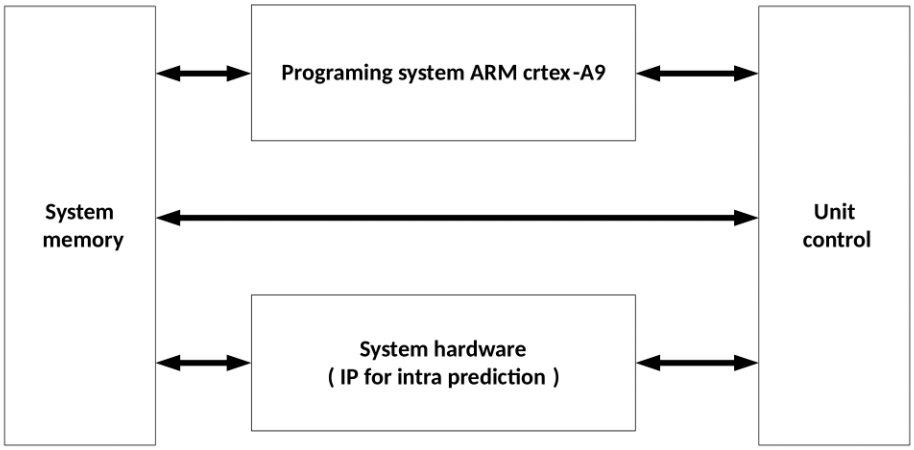


FIGURE 4.29 – Architecture matérielle système sur puce proposée sur Zynq-7000.

4.7.5 Résultats et la co-désigne

Afin de comparer les performances du système matériel du processeur avec l'implémentation logicielle sur les deux processeurs ARM cortex-a9 intégré dans le dispositif Zynq et Intel Cœur i3 CPU. Les étapes pour mesurer le temps d'exécution sont les suivantes ; d'abord les deux processeurs sont exécutés le système d'exploitation Linux, l'ensemble HM version 15.0 [Noak] est exécuté sur eux, ensuite, la fonction est d'inclure à l'intérieur du HM en particulier les fonctions dédiées à la prédiction intra du codeur, qui est calculé le nombre de cycles lorsque les fonctions sont appelées (du début à la fin de traitement) et une boucle pour accumuler le nombre de cycles de chaque appel, donc, le temps est déterminé par multiplication les résultats des cycles par la fréquence des processeurs (667MHz pour ARM et 2,40 GHz pour Intel).

Les 150 images de la séquences Basketballdrive (1080p) est utilisé dans ce test, et la valeur de QP est fixée sur 32, puis le fichier de configuration intra seulement est sélectionné parmi trois configurations (Random-access, low-delay et intra seulement) car il convient le mieux à ce cas. Toutes ces caractéristiques et encore plus d'informations sont mentionnées dans le fichier de configuration proposé par la norme en [Bo11].

Afin de comparer le temps de traitement de l'implémentation matérielle de l'accélérateur avec le logiciel d'exécution de l'algorithme pour le bloc de prédiction intra dans le HM15.0 implémenté en C++. De plus, l'implémentation de logiciels sur les deux processeurs ARM cortex-A9 et Intel cœur i3 permet l'optimisation NEON et SIMD respectivement sur le fichier de configuration avec -O3. La compilation de l'ensemble du logiciel d'algorithme pour la prédiction intra du codeur prend 0,341 ms. Cependant, 5,72 ms est le temps chargé alors que l'exécution sur le système de programmation ARM cortex-A9 processeur démarrant sur Linux Pentalinux version embarqué. Le matériel IP proposé permet d'atteindre une accélération 13,2

fois supérieure à celle du logiciel, comme indiqué dans le Tableau 26.

TABLE 4.11 – comparaison de l'implémentation de l'algorithme de prédiction intra dans différentes plates-formes pour Basketballdrive (1920 × 1080 pixels) avec tous les intra et qp=32.

Implémentation	fréquence	Le temps de traitement (ms)
dual ARM cortex-A9	667 MHz	5.72
intel cœur I3	2.40 GHz	0.341
Zynq-7020 (FPGA)	143MHz	0.0258

Comme le montre le Tableau 26, le temps de mesure sur trois systèmes de mise en œuvre de logiciels et de matériel. Les résultats de l'implémentation du logiciel sur les deux processeurs ont montré que le processeur Intel est 16,8 fois plus rapide que le processeur intégré ARM cortex-A9 en raison de la fréquence plus élevée et des différentes architectures. Ainsi, les deux processeurs ne sont pas adaptés au traitement en temps réel pour cette résolution et ses estimations sous des images vidéo 1080p telles que 480p et 720p.

4.8 Résumé

Dans cette chapitre, Pour la proposition hardware, trois architecture matériels sont présentés. Premièrement, nous développons une nouvelle architecture VLSI efficace pour la transformation de cosinus discret entier (DCT) incluse dans le dernier nouveau standard HEVC. L'architecture proposée dispose de 4x4/8x8/16x16/32x32 Transform Units (TU) fonctionnant en parallèle pour calculer le DCT entier bidimensionnel (2-D) en utilisant un bloc 1-D DCT et une mémoire de transposition. La transformation 1D est d'abord calculée et la transformation correspondante est stockée dans une mémoire. Ces données stockées sont ensuite utilisées pour réaliser la transformation DCT 2D, en utilisant le même bloc pour le premier calcul pour 1D. En outre, cette architecture est basée sur le décalage et l'additionneur au lieu de la multiplication. L'implémentation a été réalisée sur le FPGA Xilinx Artix-7 (Zynq-7000), l'accélérateur matériel proposé DCT entier 2-D prend 34421 LUTs, 8729 registres et la fréquence d'horloge maximale pour l'implémentation proposée est 289 Mhz.

Enfin, cette conception peut être utilisée pour traiter les 30 images de la séquence vidéo 8K et jusqu'à 120 images par seconde pour le 4K. Deuxièmement, il s'agit d'une nouvelle architecture matérielle parallèle pour la mode de décision, la somme absolue de la différence (SAD) de l'estimation du mouvement, qui est l'algorithme le plus critique dans la récente norme de codage vidéo HEVC. En fait, cette norme a introduit de nouvelles grandes tailles de blocs variables pour l'algorithme d'estimation de mouvement et, par conséquent, le SAD nécessite un traitement en temps plus réduit afin d'obtenir le traitement en temps réel même pour les séquences à ultra-haute résolution. L'accélérateur proposé exécute l'algorithme SAD en parallèle pour toutes les unités de prédiction de sous-bloc (PU) et de codage (CU) quelle

que soit leur taille, ce qui entraîne une amélioration considérable des performances étant donné que toutes les tailles de bloc, PU dans chaque CU, sont prises en charge et traitées en même temps. Le même FPGA Artix-7 de Xilinx est utilisé pour le prototypage et la synthèse de l'accélérateur proposé. Les résultats montrent que l'architecture matérielle proposée fonctionne à une fréquence d'horloge de 115MHz, et qu'elle atteint les exigences en temps réel 30 images par seconde (fps) pour la résolution 4k (3840×2160) et consomme 32752 LUTs. Finalement, il s'agit d'un nouvel accélérateur matériel parallèle et très efficace pour le bloc intra-prévision. Cet accélérateur permet d'obtenir un traitement à grande vitesse grâce à des unités de traitement en pipeline et une architecture de forme parallèle. La complexité de l'accès à la mémoire est également réduite grâce à la conception proposée, avec une consommation d'énergie encore plus faible. Les ressources consommées par l'architecture proposée sur la FPGA (28 nm) atteint 16520 LUTs et avoir 143,65 MHz de fréquence maximale et est capable de supporter le débit de la séquence 3840×2160 à 30 images par seconde et les expériences montrent également que le matériel peut atteindre 16,8 fois plus vite que l'implémentation software. Le schéma hardware est également implémenté avec un nombre de portes de 293K dans la technologie CMOS TSMC 180 nm sous la fréquence de 125 MHz.

Chapitre VI. Conclusion générale et les futurs travaux

Chapitre 5

Chapitre I. Le système de codage vidéo

Sommaire

5.1 Introduction au codage vidéo 119
5.2 Analyse aux limites 120
5.2.1 Calculer en cent leçons 121
5.2.2 Le monde conique 121
5.3 Vérification par simulation numérique 121

Lorem ipsum dolor sit amet, consectetur adipiscing elit **Bello1963**. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin.

5.1 Introduction au codage vidéo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam

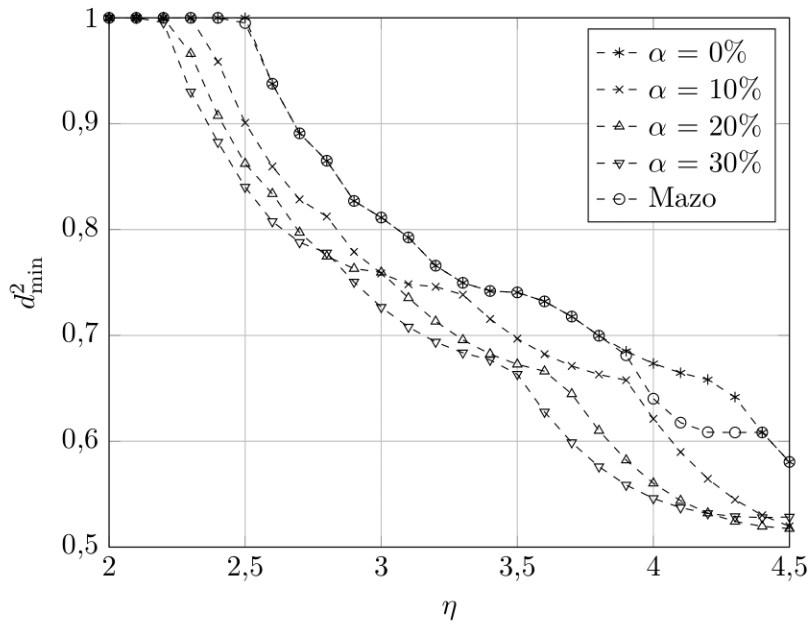


FIGURE 5.1 – Exemple de courbe TikZ.

sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula.

Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

5.2 Analyse aux limites

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in

risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum.

5.2.1 Calculer en cent leçons

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

5.2.2 Le monde conique

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

5.3 Vérification par simulation numérique

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam

sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar.

Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

Conclusion et perspectives

De nos jours, les systèmes embarqués jouent un rôle important dans la vie quotidienne de la plupart des gens, qu'il s'agisse de produits clients tels que les smartphones et les télévisions, ou des caméras de surveillances. En conséquence, il est nécessaire de travailler en temps réel tout en garantissant une très bonne qualité des données vidéos compressées et transmises. Grâce à une gamme étendue d'objectifs vidéos comme l'exemple de streaming, les systèmes embarqués peuvent être utilisés dans ces domaines en basant sur des systèmes avec une grande puissance de calcul, qui peuvent supporter des logiciels complexes et d'énormes ressources. On trouve également des systèmes embarqués à petite taille et aux ressources limitées qui sont destinés à des dispositifs simples et peu coûteux.

La norme HEVC a été standardisé en 2013 pour suivre l'évolution des systèmes embarquées surtout les systèmes intégrants la compression vidéo et/ou des écrans d'affichages de très haute résolution 4K/8K fréquemment utilisés, même si ce nouveau standard ait démontré son efficience par rapport aux plusieurs normes de compression telles que JPEG2000, H.263 et H.264/AVC, il montre aussi un coût de calcul très élevé, ce qui demande plus des travaux de recherche pour l'exécuté en temps real.

L'objectif de cette thèse est une contribution au développement et à l'implémentation sur un système embarqué dont on trouve circuit de type ARM, DSP et FPGA, des implémentation logiciels de l'encodeur et aussi de décodeur sur les deux processeurs embarqués ARM et DSP avec des optimisation algorithmique ont été arboré d'une part, et d'autre part, des accélérateurs matériels du bloc de l'intra, la transformée ICT et le bloc de décision SAD de la chaine de compression de l'encodeur vidéo, qui sont généralement essentiel coupable implémentation logiciels en temps réelle et aussi adéquate sur des plateformes matérielles embarqué. À la fin, nous avons pris l'exemple de l'accélérateur de l'INTRA pour désigne un système de l'exécution de la norme HEVC sur un enveniment logiciel/matériel en partageant les données entre les deux systèmes afin de diminuer encore plus le temps de l'exécution. la première partie est conçue au de codage vidéo et les plateformes de systèmes embarquées avec l'état d'art des différents travaux des littératures sur les implémentations de codage vidéo dans deux premières chapitres, la première aborde un détail des principes générales de système de compression vidéo dont état art de la compression vidéo et la description du standard HEVC, et le deuxième chapitre sur les différents connaissant les plateformes de systèmes embarqués dédies dont on trouves des FPGAs, ARMs et DSPs, et la fin de cette chapitre un état d'art sur des implémentations de la norme HEVC a été présenté.

La deuxième partie est été abordée à notre contribution sur la norme HEVC en trois chapitres (4, 5 et 6), le chapitre 4 est présente des implémentations logicielles sur les plateformes embarquées contenant séparément deux processeurs ARM et DSP, le temps de l'exécution et aussi le profilage de l'encodeur et de décodeur ont été détaillé. Les architectures matérielles proposées pour les trois blocs (INTRA, ICT, le bloc de décision SAD) avec une comparaison aux différents travaux de littératures ont été présentés. A la fin, Le chapitre 6 présente les

statiques de l'implémentation de bloc de l'Intra prédiction avec l'encodeur sur un système hétérogène ARM+FPGA.

De plus en plus de systèmes embarqués utilisent des processeurs dotés de telles caractéristiques, notamment ARMv7-A (par ex. Cortex-A15) et ARMv8 (par ex. Cortex-A53). Les fournisseurs de FPGA tels que Xilinx propose également une carte FPGA avec Cortex-A15. Dans ce cas, nous aimerions notre approche sur une architecture ARM plus avancée pour exploiter pleinement l'extension matérielle de processeur ARMv8, et aussi les implémentations hardware sur FPGA qui sont composent une bonne base pour les futures implémentations aux autres blocs de la norme HEVC ou bien de la nouvelle norme Cost HEVC qu'est dans la phase de l'évaluation en gardant toujours la notion de temps real pour séquences vidéo de formats 4K (4096x2160) et l'ultra HDTV ou 8K (7680x4320).

Annexe A

Linux embarqué

A.1 linux embarqué

Au cours de l'élaboration du notre thèse, il a été conclu que l'utilisation d'un système d'exploitation allait être nécessaire. Les principales raisons de cette décision sont : Requis par les pilotes de la caméra, le système d'exploitation permet l'utilisation directe des exécutables Linux qui nous sont fournis par le fabricant. Intégration native du protocole TCP/IP.

Plus grande polyvalence pour ajouter de nouveaux algorithmes de vision à l'avenir. En raison de son utilisation, il est jugé nécessaire de faire une introduction aux méthodes et aux exigences qu'un SoPC a pour le chargement et l'utilisation d'un système d'exploitation.

Dans ce projet, il a été décidé d'utiliser une distribution du noyau linux spécifique pour les périphériques Xilinx, le linux-xilinx 8 [Noao]. De plus, un système de fichiers connu sous le nom de Linaro [Noad] est utilisé, une version réduite d'Ubuntu, spécialement conçue pour les systèmes embarqués. Le processus de démarrage du système d'exploitation peut être résumé selon la Figure A.1 suivante :

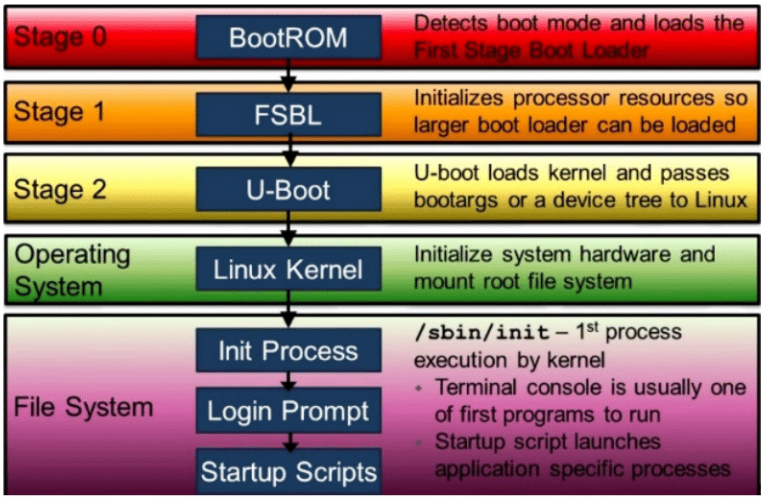


Figure A.1 – Organigramme du processus de chargement d'un OS [RLN05].

A.1.1 Boot-ROM

La Boot-ROM est installée par défaut dans le FPGA SoPC et sa fonction principale est de détecter la méthode de démarrage requise (NAND, QSPI, JTAG ou SD) et charge le code du Bootloader de premier niveau dans la mémoire. A cette fin, le Boot-Room est capable, par lui-même, d'accéder aux données éventuellement stockées dans le NAND ou le SD ou de lire la transmission des données du JTAG ou du QSPI. Après avoir vérifié ces quatre formulaires de démarrage autorisés, il détecte lequel d'entre eux contient le FSBL (First Stage Bootloader) et démarre.

A.1.2 Bootloader

Bootloader est le chargeur de démarrage de première étape, Avant d'installer et de charger le système d'exploitation, vous devez démarrer et configurer PS et PL. Ce travail est effectué par la FSBL. La FSBL sera chargée, dans ce projet, sur la carte SD avec le reste des composants nécessaires au démarrage du système. La fonction du FSBL est d'initialiser la mémoire (DDR3) et les horloges de bord essentielles pour exécuter des composants plus complexes.

A.1.3 U-Boot

Une fois que le FSBL a terminé la configuration de la carte, le SSBL (Second Stage Bootloader, SSBL) est chargé en RAM (figure A.2), cette classification générique des programmes est chargée d'installer et de configurer les systèmes d'exploitation et est même utilisée par la plupart des ordinateurs à usage général.

Dans notre cas, nous utiliserons U-Boot, un SSBL spécialement conçu pour les microprocesseurs comme l'ARM inclus dans le Zynq bien qu'il supporte une grande variété de plateformes.

Il sera également responsable de la transmission de l'arborescence des périphériques, des informations des périphériques disponibles vers le PS, vers le noyau du système d'exploitation.

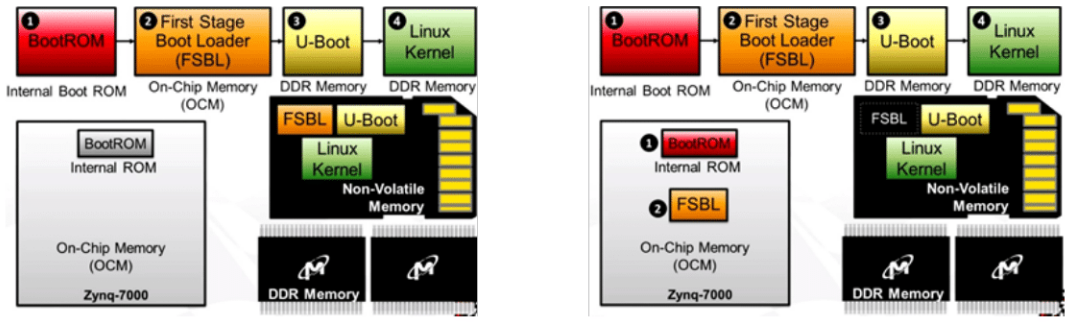


FIGURE A.2 – Charge Boot-Room Charge FSBL.

A.1.4 Linux

Enfin, le SSBL (U-Boot) localise le noyau Linux dans le SD, le charge en mémoire et procède au démarrage en appelant le programme d'initialisation, ce programme sera responsable de localiser toutes les fonctions du noyau et de les appeler correctement pour générer la bonne initialisation du système d'exploitation. Ceci chargera le terminal qui s'affichera à l'écran pour que l'utilisateur puisse s'enregistrer et commencer à utiliser le système d'exploitation

A.1.5 Connexion PL vers PS : conception du pilote

Une fois le matériel développé et le système d'exploitation installé, nous devons mettre en œuvre un système qui permette à ces deux composants de communiquer entre eux, le pilote.

Pour comprendre le fonctionnement et la nécessité pour un pilote d'établir cette communication, il est nécessaire de distinguer deux concepts : mémoire virtuelle et mémoire physique. Deux termes qui font référence au système que Linux utilise pour gérer la mémoire du système d'exploitation et son en interaction avec lui. Comme expliqué dans la partie matérielle, le VDMA extrait les informations directement de la mémoire. Cette mémoire à laquelle le VDMA est connecté est la mémoire physique connue habituellement par le nom RAM.

La mémoire virtuelle est l'abstraction de cette mémoire physique pour utilisation par le système d'exploitation. Cette distinction est faite pour permettre au système d'exploitation d'avoir un meilleur contrôle sur la mémoire. Par exemple, nous pouvons avoir la moitié de la mémoire physique comme mémoire virtuelle utilisée par un utilisateur particulier, si une partie de cette mémoire n'est pas utilisée dans une période de temps donnée, le système d'exploitation peut sélectionner ce segment de mémoire virtuelle et l'enregistrer sur le disque dur, rendant l'accès plus lent quand vous avez besoin des informations à nouveau mais libérant une grande quantité de mémoire. Cet exemple, connu sous le nom de pagination est l'un des principaux avantages de ce système de division de mémoire et permet au système d'exploitation, payant un prix en vitesse d'accès, de simuler une mémoire beaucoup plus grande que les possédés physiques. Cette distinction est importante pour le développement du projet car l'utilisateur n'a pas accès à la mémoire physique, le système d'exploitation bloquera toute tentative d'accès.

A.1.6 Langage de programmation logiciel C/C++

Le principal inconvénient dans le choix d'utiliser une plate-forme matérielle, par opposition à son approche purement logicielle, est généralement le modèle de programmation. Les algorithmes logiciels sont généralement écrits en C/C++ ou dans d'autres langages de haut niveau, qui sont indépendants des plates-formes sur lesquelles ils doivent être exécutés, du moins d'une manière générale. Ces langages permettent des itérations rapides, des améliorations incrémentales et rendent le code portable.

Annexe B

codec vidéo

B.1 Pré-traitements

Le matériel vidéo entrant peut présenter diverses dégradations et montrer des artefacts indésirables, qui doivent être enlevés. La conversion du format géométrique et la conversion du format des couleurs. De plus, la vitesse d'image à laquelle la séquence vidéo est capturée peut faire l'objet de modifications. Étant donné que différentes méthodes de distribution fonctionnent avec différentes vitesses d'image, la conversion de la vitesse d'image est une tâche importante. Au stade du prétraitement, le matériel vidéo brut entrant peut donc subir de multiples opérations de traitement du signal et d'amélioration du signal en vue des étapes suivantes de la chaîne d'outils vidéo. Pour le codage vidéo, un prétraitement spécifique peut être appliqué afin d'adapter le signal aux exigences de l'application cible. Cela comprend par exemple la fréquence d'image, la résolution de l'image et le format de couleur. D'autres prétraitements dédiés peuvent être appliqués au matériau source pour soutenir le codeur afin d'améliorer l'efficacité de la compression ou d'ajuster l'impression visuelle. Les opérations applicables comprennent, par exemple, le filtrage passe-bas local ou le filtrage antibruit. Une fois que la vidéo numérique a été capturée à la source, qui peut être de différentes sortes, comme des scènes naturelles ou du contenu synthétique généré par ordinateur, elle peut avoir besoin d'être traitée pour être encodée. Habituellement, l'étape de prétraitement peut inclure un certain filtrage, une mise à l'échelle et des conversions d'espace couleur sur la séquence vidéo brute (non compressée). Un espace colorimétrique est un modèle mathématique abstrait spécifié par les couleurs primaires qui est utilisé pour décrire une représentation d'une image, généralement sous forme d'un ensemble de valeurs des composantes trichromatiques X, Y, Z. Les conversions d'espace colorimétrique sont utilisées pour modifier la représentation des couleurs du contenu afin de mieux l'adapter au système visuel humain (HVS) et pour décorer les composants pour une meilleure efficacité du codage. Souvent, l'espace colorimétrique à la source est au format RVB, mais comme le HVS est plus sensible aux variations de lumière qu'aux variations de couleur, la version à échelle de gris de l'image, qui contient l'information lumineuse (Luma), est séparée de l'information couleur (Chroma). Les espaces colorimétriques de la famille où le Luma est séparé du Chroma sont généralement appelés espaces colorimétriques YUV. De cette façon, l'information de couleur peut être sous-échantillonnée sans aucune dégradation visuelle. Par conséquent, les améliorations apportées à la qualité du codage vidéo porteront sur la composante Luma, puisqu'elle joue un rôle plus important dans la qualité perceptuelle et représente une part importante dans le flux binaire final. La Figure B.1 présente l'espace de couleur RVB, En haut à gauche : combinaison des couleurs primaires

Résumé de la thèse

Le codage vidéo à haute efficacité (HEVC) est le dernier standard de compression vidéo, qui est appliqué dans de nombreux domaines multimédia. Actuellement, il est principalement utilisé pour l'ultra haute résolution, ce qui conduit à des exigences de calcul élevées du point de vue du temps d'exécution, de l'énergie et de la consommation de la mémoire. Plusieurs travaux ont été entrepris pour optimiser le codec HEVC de façon logicielle et/ou matériel afin de l'adopter dans les systèmes embarqués. Une implémentation détaillée des derniers encodeurs et décodeurs vidéo HEVC sur les processeurs embarqués TMS320C6678 et ARM du processeur de signal numérique (DSP) est proposée dans ce mémoire de thèse. Cette implémentation inclut une étude de l'analyse de complexité en particulier le profilage. Une optimisation basée sur la technologie VLIW (Very Long Instruction Word) et Neon est présentée. Les résultats expérimentaux ainsi mis en évidence sur un seul noyau atteignent une vitesse plus rapide de 4 fois celle mesurée sans optimisation. Pour l'implémentation matériel, trois implémentations sur la FPGA et ASIC sont proposées. Une nouvelle architecture VLSI efficace pour la transformation de cosinus discret entier (DCT) incluse dans le dernier nouveau standard HEVC. Ensuite une architecture matérielle parallèle pour le mode de décision correspondant à la somme absolue de la différence (SAD) de l'estimation du mouvement est proposée. L'estimateur de mouvement étant l'algorithme le plus critique dans la récente norme de codage vidéo HEVC. Enfin, un nouvel accélérateur matériel parallèle et très efficace pour le bloc de prévision intra est validée. Ces trois architectures permettent d'atteindre les exigences temps réel pour la résolution 4K (30 frames/s). De plus ce critère temps réel est validé pour le bloc DCT avec la résolution 8K.

Mots-clés : HEVC, Embedded system, FPGA, DSP and ARM