Efficient Hardware implementation of transform module for the Versatile Video Coding standard on Intel SoC FPGA

Ahmed Kammoun

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THESE DE DOCTORAT DE

L'INSA RENNES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Signal, Image, Vision

Par

Ahmed KAMMOUN

Implémentation matérielle efficace du module de transformée pour le futur standard vidéo Versatile Video Coding en technologie SoC FPGA

Thèse présentée et soutenue à Rennes, le 12 / 12 / 2019
Unité de recherche : IETR
Thèse N° : 19ISAR 33 / D19 - 33

Rapporteurs avant soutenance :

Marco CAGNAZZO
Professeur, Telecom-Paris Tech
Chiheb REBAI
Professeur, Sup'Com de Tunis, Tunisie

Composition du Jury :

Frédéric DUFAUX
Directeur de Recherche, CNRS laboratoire L2SI / Président du jury
Marco CAGNAZZO
Professeur, Telecom-Paris Tech / Rapporteur
Chiheb REBAI
Professeur, Sup'Com de Tunis, Tunisie / Rapporteur
Mohamed ATRI
Professeur, Faculté de sciences Monastir / Examinateur
Jean-François NEZAN
Professeur, INSA Rennes / Directeur de thèse
Nouri MASMOUDI
Professeur, ENIS Sfax / Directeur de thèse
Wassim HAMIDOUCHE
Maître de Conférences, INSA Rennes / Encadrant
Fatma BELGHITH
Maître assistante, Fac de sciences Sidibouzid / Encadrante
Pierrick PHILIPPE
Ingénieur de recherche, Orange Rennes / Invité

Titre : **Implémentation matérielle efficace du module de transformée pour le futur standard vidéo Versatile Video Coding en technologie SoC FPGA**

Mot clés : Versatile Video Coding, Transformée MTS, LFNST, DCT-II, Arria10, Implémentation hardware

Résumé : La future norme de codage vidéo MPEG / ITU nommée Versatile Video Coding (VVC) est attendue d'ici à la fin de 2020. VVC permettra une meilleure efficacité de codage par rapport à la norme actuelle de codage High Efficiency Video Coding (HEVC). Ce gain de codage est apporté par plusieurs outils de codage. La Multiple Transform Selection (MTS) est l'un des principaux outils de codage introduits dans VVC. Le concept MTS implique trois types de transformée, à savoir la transformée en cosinus discrète (DCT)-II, la transformée en sinus discret (DST)-VII et la DCT-VIII.

La première contribution repose sur la proposition de la première implémentation matérielle dans la littérature du module de transformée en 2D, incorporé dans les premiers projets de VVC. L'architecture comprend cinq types de transformation et prend en charge toutes les tailles de transformation de 4 à 32, tenant compte de toutes les combinaisons de taille asymétriques. L'architecture proposée bénéficie des multiplieurs LPM internes et des blocs DSP offerts par la plateforme cible SoC-FPGA Arria 10.

Ensuite, en raison de la conception très complexe de MTS, une approche d'approximation est proposée pour réduire la complexité de calcul du DST-VII et du DCT-VIII. L'approximation consiste à appliquer des étapes d'ajustement à une variante de la famille DCT-II, principalement la DCT-II et son inverse. Les étapes d'ajustement sont des matrices diagonales à faible complexité (maximum 5 coefficients par ligne) dérivées d'un algorithme d'optimisation génétique. De plus, une implémentation matérielle efficace du MTS approximé est proposée, valable pour les deux sens direct et inverse. L'architecture proposée est totalement pipelinée et supporte les trois types de transformation impliqués dans la dernière version VVC avec une consommation des ressources hardware très modérée.

Enfin, dans la dernière version de la norme VVC, un autre outil complexe de transformée nommé Low Frequency Non Separable Transform (LFNST) est également intégré. Dans ce travail, nous proposons aussi une implémentation matérielle efficace du LFNST utilisant les multiplications à valeur non constante (basé sur les mémoires ROMs) afin de réduire la complexité de calcul en nombre d'opérations et les besoins en ressources matérielles.

Title : **Efficient Hardware implementation of transform module for the Versatile Video Coding standard on Intel SoC FPGA**

Keywords: Versatile Video Coding**,** MTS transform, LFNST, DCT-II, Arria10, Hardware Implementation

Abstract: The future MPEG/ITU video coding standard named Versatile Video Coding (VVC) is expected by the end of 2020. VVC will enable better coding efficiency than the current High Efficiency Video Coding (HEVC) standard. This coding gain is brought by several coding tools. The Multiple Transform Selection (MTS) is one of the key coding tools that have been introduced in VVC. The MTS concept involves three transform types including Discrete Cosine Transform (DCT)-II, Discrete Sine Transform (DST)-VII and DCT-VIII.

The first contribution relies on proposing the first hardware implementation in literature of the 2D Transform Module incorporated in the first VVC drafts. The architecture involves five transform types and supports all transform sizes from 4 to 32 taking into accounts all asymmetric size combinations. The proposed architecture benefits from internal LPM multipliers and DSP blocks offered by the target SoC-FPGA device Arria 10.

Then, due to high complex design of MTS, an approximation approach is proposed to reduce the computational cost of the DST-VII and DCT-VIII.
The approximation consists in applying adjustment stages to a variant of DCT-II family mainly DCT-II and its inverse. Adjustment stages are sparse block-band matrices derived using a genetic optimization algorithm. In addition, an efficient hardware implementation of the forward and inverse approximate transform module is proposed. The pipelined architecture design supports all three transform types involved in the latest VVC draft using very moderate hardware requirements.

Last but not least, in the latest VVC standard, other complex transform tool Low Frequency Non Separable Transform (LFNST) is also incorporated. In this work, we propose an efficient hardware implementation of LFNST through non constant value multiplications to reduce the computational complexity and hardware resource requirements.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Implémentation matérielle efficace du module de transformée pour le nouveau standard video Coding en technologie FPGA

Nom Prénom de l'auteur : KAMMOUN AHMED

Membres du jury :
- Monsieur REBAI Chiheb
- Monsieur ATRI Mohamed
- Monsieur MASMOUDI Nouri
- Madame BELGHITH Fatma
- Monsieur DUFAUX Frédéric
- Monsieur NEZAN Jean-François
- Monsieur HAMIDOUCHE Wassim
- Monsieur CAGNAZZO Marco

Président du jury : Frédéric Dufaux

Date de la soutenance : 12 Décembre 2019

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 12 Décembre 2019

Signature du président de jury

Le Directeur,

M'hamed DRISSI

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Intitulé de la thèse :

Implémentation matérielle efficace du module de transformée pour le futur standard vidéo Versatile Video Coding en technologie SoC FPGA

Ahmed KAMMOUN

En partenariat avec :

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

UNIVERSITE BRETAGNE LOIRE MATHSTIC

N° d'ordre : 2018-08-TH

Année 2019

THÈSE DE DOCTORAT

Domaine : STIC

Spécialité : Electronique

École Doctorale MATHSTIC

présentée par

Ahmed KAMMOUN

Sujet :

Efficient hardware implementation of Transform module for the Versatile Video Coding standard on Intel SoC FPGA

Soutenue à Rennes le 12 décembre 2019 devant le jury composé de :

Présidente du jury :	**–**
	Professeur, Université –
Directeur de Thèse :	**Jean-Francois NEZAN**
	Professeur, IETR, Rennes
	Nouri MASMOUDI
	Professeur, ENIS, Sfax
Rapporteurs :	**–**
	Maître de conférences, –
	–
	–
Co-Encadrants :	**Wassim HAMIDOUCHE**
	Maître de conférences, INSA, Rennes
	Fatma BELGHITH
	Maître de conférences, ENIS, Sfax

I would like to dedicate this thesis to my loving parents, all my family members and my friends ...

Acknowledgments

First of all, i would like to thank my *family and friends* for supporting me during all stages of my education to whom I dedicate every success in my life.

I would like to express my sincere and deep gratitude to my PhD supervisor at LETI lab of ENIS, professor Si *Nouri Masmoudi*, for his continuous support, guidance, encouragement and for being always available whenever I needed him. I really appreciate all his contributions of time, support, and ideas.
I would also like to thank my PhD supervisor at IETR lab of INSA Rennes, professor *Jean François Nezan* for his advice, dedication and support during these years of work. I will always be grateful for the opportunity to work with him and integrate the VAADER team.

I would like to thank my supervisor, *Wassim Hamidouche* from IETR Lab at INSA Rennes. He has transmitted me the enthusiasm in the daily work by flooding me with new ideas and challenges every day. In addition to being an excellent supervisor, he is an extremely hardworking researcher. I tried my best to follow his example but I could not meet half of his dedication.

A special mention to *Fatma Belghith*, my supervisor at LETI in Sfax, for her contributions of time, support, and ideas. She was one of the reasons that gave me the desire to work in the field of image processing and encouraging me to continue my education with a PhD.

During these years, i was very lucky to work with well known international laboratories such as *Pervasive Computing* at TUT Finland and *GDEM* research group at UPM Spain. I am very honored to co-work, learn and most importantly spend really good times with them. So i have to express my gratitude to *Jarno*, Marco, Arttu, *Fernando*, Mathias, Miguel, Raquel, Dani, Jaime and Ruben...

I would like the thank *Pierrick Philippe* from Orange labs. I feel that I am very lucky and honored to have been able to work with him and I hope that our paths will cross again during our professional and personal lives.

I strongly believe that without all these efforts, the results of my work during the last three years would be very different.

I also want to thank *Naty Sidaty* for being there when i need him the most and for all his guidance and advice with the interesting discussions we had about almost any topic. He will be always considered as one of my best friends and i wish him all the best. A special thanks to *Olivier Deforges, Luce Morin, Myriam, Guillaume and Muhammed Abdel Wahab* who made my stay a lot more pleasant. I am very thankful to the *rest of my colleagues of both the Vaader team at IETR laboratory and Circuits and Systems at LETI laboratory*, who made working there so enjoyable.

Last but not least, i would like to thank my dear *Asma* and my sister *Emna* for their infinite support in these years of work and dedication.

The members of the jury *Frédéric DUFAUX, Marco CAGNAZZO, Hicham REBAI, Mohamed ATRI and Pierrick PHILIPPE* deserve a distinctive mention as well for having accepted to review, assisted to my PhD defence and given constructive feedback.

Résumé

La future norme de codage vidéo MPEG / ITU nommée *Versatile Video Coding* (VVC) est attendue d'ici à la fin de 2020. VVC permettra une meilleure efficacité de codage par rapport à la norme actuelle de codage *High Efficiency Video Coding* (HEVC). Ce gain de codage est apporté par plusieurs outils de codage. La *Multiple Transform Selection* (MTS) est l'un des principaux outils de codage introduits dans VVC. Le concept MTS implique trois types de transformée, à savoir la transformée en cosinus discrète *(DCT)-II*, la transformée en sinus discret *(DST)-VII* et la *DCT-VIII*.

La première contribution repose sur la proposition de la première implémentation matérielle dans la littérature du module de transformée en 2D, incorporé dans les premiers projets de VVC. L'architecture comprend cinq types de transformation et prend en charge toutes les tailles de transformation de 4 à 32, tenant compte de toutes les combinaisons de taille asymétriques. L'architecture proposée bénéficie des multiplieurs LPM internes et des blocs DSP offerts par la platforme cible *SoC-FPGA Arria 10*. Ensuite, en raison de la conception très complexe de MTS, une approche d'approximation est proposée pour réduire la complexité de calcul du *DST-VII* et du *DCT-VIII*. L'approximation consiste à appliquer des étapes d'ajustement à une variante de la famille *DCT-II*, principalement la *DCT-II* et son inverse. Les étapes d'ajustement sont des matrices diagonales à faible complexité (maximum 5 coefficients par ligne) dérivées d'un algorithme d'optimisation génétique. De plus, une implémentation matérielle efficace du module d'approximation de transformée MTS est proposée dans valable pour les deux sens direct et inverse. L'architecture proposée est totalement pipelinée et supporte les trois types de transformation impliqués dans la dernière version VVC avec une consommation des ressources hardware très modérée.
Enfin, dans la dernière version de la norme VVC, un autre outil complexe de transformée nommé *Low Frequency Non Separable Transform* (LFNST) est également intégré. Dans ce travail, nous proposons aussi une implémentation matérielle efficace du LFNST utilisant les multiplications à valeur non constante (basé sur les mémoires ROMs) afin de réduire la complexité de calcul en nombre d'opérations et les besoins en ressources matérielles.

Abstract

The future MPEG/ITU video coding standard named *Versatile Video Coding* (VVC) is expected by the end of 2020. VVC will enable better coding efficiency than the current *High Efficiency Video Coding* (HEVC) standard. This coding gain is brought by several coding tools. The *Multiple Transform Selection* (MTS) is one of the key coding tools that have been introduced in VVC. The MTS concept involves three transform types including Discrete Cosine Transform *(DCT)-II*,Discrete Sine Transform *(DST)-VII* and *DCT-VIII*.

The first contribution relies on proposing the first hardware implementation in literature of the 2D Transform Module incorporated in the first VVC drafts. The architecture involves five transform types and supports all transform sizes from 4 to 32 taking into accounts all asymmetric size combinations. The proposed architecture benefits from internal LPM multipliers and DSP blocks offered by the target *SoC-FPGA device Arria 10*. Then, due to high complex design of MTS, an approximation approach is proposed to reduce the computational cost of the *DST-VII* and *DCT-VIII*. The approximation consists in applying adjustment stages to a variant of *DCT-II* family mainly *DCT-II* and its inverse. Adjustment stages are sparse block-band matrices derived using a genetic optimization algorithm. In addition, an efficient hardware implementation of the forward and inverse approximate transform module is proposed. The pipelined architecture design supports all three transform types involved in the latest VVC draft using very moderate hardware requirements.
Last but not least, in the latest VVC standard, other complex transform tool *Low Frequency Non Separable Transform* (LFNST) is also incorporated. In this work, we propose an efficient hardware implementation of LFNST through non constant value multiplications to reduce the computational complexity and hardware resource requirements.

Résumé étendu

0.1 Préambule

Notre environnement est analogique depuis des décennies. Cependant, la demande excessive de confort, de qualité et d'évolution continue de l'être humain nous a obligé à passer au format numérique. De nos jours, ce dernier et en particulier le monde du multimédia envahissent tout l'environnement quotidien. Ce vaste monde connaît actuellement un développement technologique remarquable, basé essentiellement sur le contenu vidéo et ses processus de stockage et de transmission.
Selon une récente étude de la société Cisco dans [1], le trafic vidéo sur IP global comptera pour 82% du trafic IP total d'ici 2022, étant 61% en 2016. Une autre étude de la société Ericsson a annoncé une croissance du trafic vidéo sur les réseaux mobiles d'environ 35% par an au cours des 6 prochaines années [2]. Ces chiffres s'avèrent très fiables pour les raisons suivantes: L'accès au contenu vidéo via Internet et les applications mobiles est en train de devenir "open source" ou uniquement conditionné par un processus d'authentification d'une plate-forme privée; Un concept adopté par des fournisseurs de contenu renommés tels que Netflix, YouTube ou Amazon. (Les grandes retransmissions d'événements sportifs et les retransmissions en direct de jeux sont des expériences visuelles de plus en plus populaires. En outre, les jeunes générations (12 à 16 ans) sont également des fournisseurs actifs et des consommateurs de vidéos via des services de médias sociaux tels que Facebook, Twitter, Snapchat et plus récemment TikTok. Cet environnement dynamique est à l'origine de changements radicaux dans la manière de consommer des vidéos et explique le trafic croissant dans le monde. Parallèlement au contenu considérable et à la diversité des moyens d'accessibilité, la croissance du volume de données vidéo s'explique également par l'augmentation des exigences de qualité et d'immersion de la part des utilisateurs finaux. La résolution est passée de la définition standard (SD) à la haute définition (HD) et nous observons actuellement une popularité croissante du contenu UHD (Ultra High Definition), ce qui correspond au fait que tous les téléviseurs actuels prennent en charge la résolution UHD. Bien que le développement des industries des fabricants de téléviseurs et de caméras permette une plus grande fluidité dans l'affichage des contenus, les nouveaux formats vidéo émergents, notamment les résolutions 4K et 8K, High Frame Rate (HFR), High Dynamic Range (HDR) [3] contribuent considérablement à augmenter la part

du trafic vidéo, car elles impliquent inévitablement une augmentation physique de la quantité de données à afficher afin d'améliorer la qualité de l'expérience (QoE).
Pour résumer, les données vidéo sont de plus en plus présentes dans la vie quotidienne des utilisateurs et leur taille numérique se multiplie. Même si les réseaux et les capacités de stockage sont nettement plus élevés qu'au passé, la conclusion est simple. La compression vidéo est non seulement pertinente: la compression vidéo est une nécessité et, plus loin encore, le codage vidéo efficace est extrêmement important.

0.2 Contexte et motivations

Au cours des 30 dernières années, le codage vidéo a incité les chercheurs universitaires et industriels à créer des produits et des services attrayants. Les principes de base du codage vidéo n'ont pas changé au cours de ces dernières années, car presque toutes les générations de normes vidéo reposent sur les mêmes principes de codage vidéo hybrides utilisés depuis la Recommandation H.261 en 1988. Néanmoins, les améliorations des outils de codage vidéo ont conduit à des réductions impressionnantes du débit pour une même qualité vidéo. Afin de satisfaire aux exigences croissantes en matière de stockage et de transmission vidéo, l'efficacité du codage vidéo doit être améliorée soit en optimisant les outils de codage existants au moyen d'ajustements mineurs ou majeurs sans modifier la syntaxe du décodeur, soit en développant et en incorporant des outils de codage plus efficaces pour aboutir à une nouvelle génération norme vidéo. L'objectif primordial d'une nouvelle norme est une réduction de 50% du débit pour une qualité d'image équivalente. Cet objectif est atteint par la plus récente norme de codage vidéo High Efficiency Video Coding (HEVC) [4, 5] (publiée officiellement début 2013 [6]) par rapport à son prédécesseur [7, 8] Advanced Video Coding AVC/ H.264 [9].
Dans ce contexte, les travaux menés dans le cadre de cette thèse ont débuté en octobre 2016 durant laquelle un potentiel nouveau standard nommé Versatile Video Coding VVC, est dans sa première phase de normalisation. En fait, JVET (Joint Video Exploration Team) [10] a lancé un "Call for Proposals" (CFP) sur la compression vidéo afin de développer la norme VVC, avec des performances de codage allant au-delà de HEVC. La norme VVC est attendue d'ici la fin de l'année 2020 [11]. Les outils de codage, développés en VVC, permettent d'augmenter l'efficacité du codage de 30% par rapport à HEVC [12]. Ce gain est la somme de plusieurs améliorations apportées aux modules de la chaîne de codage, y compris le processus de transformée, qui est l'un des outils clés du codec hybride ainsi que le sujet principal de ce travail. La "Multiple Transform Selection" (MTS) en 2D est l'un des nouveaux concepts introduits dans VVC [13]. La version antérieure de MTS (nommée Adaptive Multiple Transform (AMT)), intégrée dans le code de référence "Joint Exploration Model" (JEM) [14], implique cinq types de transformée notamment les Transformées en Cosinus Discrète (DCT) II, V et VIII et la Transformée en Sinus Discrète (DST) de types

VII et I [15]. Dans VVC, la MTS ne repose que sur trois transformées trigonométriques, à savoir les DCT-II et VIII et DST-VII.
Néanmoins, parallèlement à la réduction du débit, les nouveaux outils introduisent une complexité de calcul très élevée, que ce soit dans l'encodeur ou décodeur. Il est logique d'obtenir une meilleure efficacité de codage accompagnée d'une complexité accrue et d'une plus grande puissance de traitement que les codecs précédents. Par exemple, HEVC est 3 à 5 fois plus complexe qu'AVC. Cependant, la complexité de la VVC devrait être 7 à 10 fois supérieure à la norme HEVC. Nous soulignons que ces chiffres varient souvent, car la conception du modèle de référence est encore en phase de développement.
Cette complexité significative est l'un des principaux défis pour le développement du standard VVC, en particulier pour les implémentations en temps réel sur des plates-formes intégrées. Par ailleurs, les implémentations matérielles ont pour objectif d'accélérer les performances et d'essayer de réduire la complexité de calcul, mais sous la contrainte de la disponibilité des ressources (mémoire, logique, etc.). Dans ce contexte, les plates-formes intégrées connaissent également de grands progrès. Récemment, les FPGA (Field-Programmable Gate Array), nouvellement créées, permettent la mise en œuvre de conceptions de systèmes sur puce (SoC). Ces périphériques sont disponibles pour les applications Low End (LE) [16], Middle End (ME) [17] et High End (HE) [18]. Ils sont équipés de nombreuses améliorations sur les plans materiel et logiciel qui les rendent plus adaptés aux applications très complexes nécessitant une mémoire et des ressources de calcul élevées, telles que le traitement vidéo à haute résolution.

0.3 Contributions

Ce document présente un ensemble de propositions ayant pour objectif final de réduire le niveau de complexité élevé introduit par le module de transformée, essentiellement pour la norme VVC. Il aborde le problème de l'implémentation efficace des différents types de transformées impliquées tout au long de l'évolution du standard VVC, partant des AMT (5 types) dans ses premières versions à MTS (3 types) dans la version actuelle. Les principaux objectifs sont de réduire la complexité de calcul et d'atteindre un traitement à haute fréquence d'image traitées pour supporter le codage temps réel des vidéos 4K tenant compte des contraintes et des limites des ressources matérielles (mémoires, utilisation de la logique, fréquence de fonctionnement, etc.).
Les principales contributions de cette thèse sont les suivantes:

1. Une implémentation matérielle efficace du module de l'AMT en 2D et avec pipeline comprenant les cinq types de transformés de taille 4x4, 8x8, 16x16 et 32x32 est proposée. C'est la première proposition d'implémentation pour VVC dans la littérature. La méthode de conception proposée profite bien des ressources logicielles/matérielles

de la plateforme FPGA cible *Arria 10*, telles que les LPMs (Library of Parameterized Modules) [19], IP internes et les blocs DSP (Digital Signal Processing) afin de réduire l'utilisation de la logique. Une architecture 2D unifiée intègre tous les modules de transformée 1D de taille 4x4, 8x8, 16x16 et 32x32 tenant compte de toutes les combinaisons 2D possibles (symétriques et asymétriques). L'architecture proposée est capable de supporter le codage vidéo 2K à 50 images par seconde avec une fréquence de fonctionnement pouvant atteindre 147 MHz.

2. En raison de la conception très complexe de MTS, une approche d'approximation est proposée pour réduire le coût de calcul du DST-VII et du DCT-VIII. L'approximation consiste à appliquer des étapes d'ajustement, basées sur des matrices diagonales à faible complexité de calcul à une variante de la famille DCT-II, principalement la DCT-II et son inverse. Les coefficients optimaux des matrices d'ajustement sont dérivés à partir d'un algorithme génétique d'optimisation. De plus, une implémentation matérielle efficace de l'approximation du module de transformée MTS est proposée pour les deux sens direct et inverse pouvant être intégrée à la fois au codeur et au décodeur VVC. La conception de l'architecture comprend deux étages: un module en pipeline, unifié et reconfigurable pour la transformée DCT-II direct et inverse puisque c'est l'élément principal pour la méthode d'approximation des DST-VII et DCT-VIII. Le deuxième élement est l'étage d'ajustement à faible complexité de calcul et une allocation de ressources logiques très modérée.

   En termes d'efficacité de codage, les approximations DST-VII et DCT-VIII préservent le gain de codage du MTS. D'autre part, l'architecture globale proposée permet de traiter respectivement les vidéos 2K et 4K à 386 et 96 images par seconde tout en gardant une faible consommation des ressources logiques de la plateforme FPGA *Arria10*.
3. Un nouvel outil de codage appelé Low Frequency Non Separable Transforms (LFNST) est récemment incorporé dans la norme VVC, qui est étroitement lié au processus de transformation pour améliorer davantage l'efficacité du codage. Dans ce document, une étude du LFNST est tout d'abord fournie. Elle porte sur son évolution tout au long du processus de normalisation jusqu'à l'ébauche actuelle de VVC. Une implémentation matérielle efficace et optimisée de LFNST est proposée sur la base d'une multiplication à coefficients non constants utilisant des blocs de mémoire ROM et des multiplieurs LPM. L'architecture matérielle proposée permet une consommation très modérée des ressources logiques pour des différentes configurations de conception. Une approche d'optimisation du temps d'exécution est également proposée, qui permet de réduire considérablement le temps de traitement avec une augmentation négligeable des besoins en ressources matérielles. Au meilleur de nos connaissances, il s'agit de la première implémentation matérielle de la littérature du module LFNST pour la norme VVC.

0.4 Organisation du manuscrit

Cette section présente l'organisation générale de cette thèse.

Le chapitre 2 commence par une introduction aux bases du codage vidéo et à quelques concepts essentiels pour cette thèse, partant de l'historique de la compression vidéo aux principaux standards et codecs, en passant par le schéma de compression vidéo hybride classique. Il présente un bref aperçu des principales spécificités de la norme HEVC à la pointe de la technologie, ainsi que de la future norme VVC, en se concentrant principalement sur les nouveaux outils de codage et les améliorations responsables du gain de codage attendu.

Chapitre 3 Dans ce chapitre, l'accent est rapidement mis sur le module de transformée. Ce dernier sera étudié bien expliqué é pour les normes HEVC et VVC. Nous présentons ensuite d'importantes propositions de pointe en matière d'optimisation de la transformation, d'approximation et de mise en œuvre matérielle, essentiellement pour le standard VVC, qui constituent la motivation principale de cette thèse. Une Présentation de la plate-forme cible, Intel SoC FPGA Arria 10, est également fournie, en se concentrant principalement sur les outils matériels et logiciels utilisés pour ce travail.

Le chapitre 4 s'intéresse à l'implémentation du module AMT, y compris cinq types de transformation. Tout d'abord, une conception détaillée éliminant toutes les multiplications est proposée pour l'ordre 4. Ensuite, ce travail est étendu à toutes les tailles de blocs de 4 à 32 afin de présenter la première proposition d'implémentation dans la littérature du module AMT en 2D pour VVC. Les résultats expérimentaux et de synthèse sont discutés et comparés avec d'autres travaux dans la littérature.

Le chapitre 5 décrit l'approche d'approximation adoptée pour le module MTS. Il est d'abord modélisé comme un problème d'optimisation à contraintes entiers. L'algorithme génétique est ensuite utilisé pour résoudre le problème et calculer les matrices d'ajustement pour les grandes tailles de transformée $N \in \{16, 32\}$. Une implémentation matérielle de l'approximation du module MTS en 2D. Les architectures sont entièrement pipelinés et unifiés pour les deux sens de transformée direct et inverse comprenant l'étage d'ajustement qui est basé sur des matrices orthogonales diagonales à faible complexité de calcul. Les résultats expérimentaux et de synthèse des implémentations 1D et 2D sont également présentés et discutés par rapport aux travaux existants dans la littérature.

Le chapitre 6 présente une application co-design de l'implémentation DCT-II inverse pour HEVC sur la plateforme FPGA Arria 10 SoPC. Tout d'abord, une vue d'ensemble du périphérique FPGA cible est fournie portant sur les principaux caractéritiques et les critères de choix. Ensuite, les différents outils logiciels et matériels utilisés dans cette application sont détaillés. Les interfaces de communication utilisant les FIFOs Avalon Memory Mapped sont utilisés pour créer la communication entre le code software C (driver) responsable de l'envoi

et la éception des données et la partie IP matérielle représentée par le circuit de l'architecture IDCT en VHDL. Toutes les étapes d'implémentation réalisées sont décrites dans ce chapitre. De plus, les résultats expérimentaux en termes de temps d'exécution sont fournis et discutés pour différentes configurations utilisant des bus de communication à 32 et 64 bits.

Le chapitre 7 commence par une description détaillée du nouvel outil de codage associé à l'opération de transformation, Low Frequency Non Separable Transforms (LFNST). Un bref historique de son intégration tout au long de la normalisation VVC est également présenté. Ensuite, la solution proposée pour fournir une implémentation matérielle efficace est détaillée en se basant d'une approche de multiplication à coefficients non constants via l'utilisation des mémoires ROMs et d'un très faible nombre de multiplicateurs et blocs DSP. Les résultats expérimentaux et de synthèse sont présentés et discutés pour différentes configurations de conception.

Enfin, le chapitre 8 conclut ce travail en résumant les contributions proposées et en fournissant des perspectives pour les prochains travaux.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, context and motivations for this work are described. Then, the main contributions of the work are listed followed by the whole manuscript outline.

Contents

1.1 Preamble

Our environment has been analog for decades. However, the excessive demand for comfort, quality and continual optimization of the human being has arisen to switch to digital format. Nowadays, this latter and especially the world of multimedia invade the entire everyday environment. This vast world is undergoing a remarkable technological development aimed essentially at video content and its storage and transmission processes.
According to a recent study conducted by Cisco company in [1], the video traffic over Global IP will account for 82% of all IP traffic by 2022, up from 61% in 2016. Another study of Ericsson company announced a growth of video traffic in mobile networks around 35% annually for the next 6 years [2]. These numbers are proving to be strongly reliable for the following reasons: Access to video content using Internet and mobile applications is becoming open source or conditioned only by an authentication process for a private platform display; a concept which is adopted by famous content providers such as Netflix, YouTube or Amazon. Big sports events broadcasts and game live streams are visual experiences that also attract a growing popularity. In addition, young generations (12-16years) are also active providers and consumers of videos through social media services as Facebook, Twitter, Snapchat and more

recently TikTok. This dynamic environment is the reason behind drastic changes in the way of consuming videos and the explanation of the growing worldwide video traffic. In parallel to the tremendous content and the diversity of accessibility means, the video data volume growth is also explained by the increase of end-user demands for quality and immersion. The resolution rose from Standard Definition (SD) to High Definition (HD) and currently, we are observing a growing popularity of Ultra High Definition (UHD) contents, matching the fact that all current TVs support UHD. Although the development of television and cameras manufacturers industries allowed more fluidity in the display of contents, the new emerging video formats including 4K, 8K resolutions, High Frame Rate (HFR), High Dynamic Range (HDR) [3] and omnidirectional videos considerably contribute to increase the part of video traffic as they inevitably imply a physical growth in the quantity of data to display in order to improve the Quality of Experience (QoE).
To sum up, video data are more and more present in users' everyday life and numerical size of video data is multiplying. Even if networks and storage capacities are significantly higher than in the past, the conclusion is simple. Video compression is not only relevant: Video compression is a necessity and even further, an efficient video coding is extremely important.

1.2 Context and Motivations

Over the past 30 years, video encoding has motivated academic and industrial researches to produce appealing products and services. The fundamentals of video encoding have not changed over these years, as almost every video standard generation is based on the same block-based hybrid video coding principles that have been used since Recommendation H.261 in 1988. Nevertheless, improvements of video coding tools have led to impressive reductions in bitrate for a similar objective video quality. In order to satisfy the increasing demands of storage and video transmission, video coding efficiency should be enhanced either by optimizing existing encoding tools with minor or major adjustments without changing the syntax of the decoder, or by developing and incorporating more efficient coding tools to build the next generation video standard. The prime objective of a new standard is 50% bitrate reduction for equivalent perceived quality. And that is what the latest standard High Efficiency Video Coding (HEVC) [4, 5] (officially released in early 2013 [6]) has achieved [7, 8] with respect to its predecessor H.264/Advanced Video Coding (AVC) [9].

In this context, the work carried out in this thesis started in October 2016 where a new potential standard Versatile Video Coding was in its first phase of standardization. In fact, the Joint Video Exploration Team (JVET) [10] has launched a Call for Proposals (CFP) on video compression in order to develop the VVC standard, with coding performance beyond HEVC. The VVC standard is expected by the end of 2020 [11]. The coding tools, developed in VVC enable to increase the coding efficiency by 30% compared to HEVC [12]. This gain

is the sum of several improvements in the coding chain modules including the transformation process which is one of the key tools of the hybrid codec and the main subject of this work. The separable two dimensional Multiple Transform Selection (MTS) is one of the new concepts introduced in VVC. The earlier version of the MTS (named Adaptive Multiple Transform (AMT)) [13], integrated in the Joint Exploration Model (JEM) [14], consists of five transform kernels including Discrete Cosine Transform (DCT) types II, V and VIII, and Discrete Sine Transform (DST) types VII and I [15]. In VVC, the MTS relies only on three trigonometric transforms including DCT-II and VIII, and DST-VII.

Nevertheless, parallel to the bitrate reduction, the new tools introduce a very heavy computational complexity increase on both decoding and (even more) encoding side. It is logical that a better coding efficiency is reached at the expense of higher complexity and requires more processing power as former codecs. For instance, HEVC is 3 to 5x more complex than AVC. However, VVC complexity is expected to be 7 to 10x higher than HEVC standard. We point out that these numbers are often varying, since the reference model design is still under development phase.

This coarse complexity is one of the main challenges for the development of the VVC standard, especially for real time implementations on embedded platforms. On the other hand, hardware implementations are meant to provide some performance accelerations and try to compensate the substantial increase of computational complexity but under the constraints of their resources availability (memory, logic...). In this context, the embedded platforms are also witnessing great progress. Recently, the new created advanced Field-Programmable Gate Array (FPGA) chips enable the implementation of Systems on Chips (SoC) designs. These devices are available for Low End (LE) [16], Middle End (ME) [17] and High End (HE) [18] applications. They are equipped with many soft and hard improvements to make them more adequate for applications requiring high memory and computation resources, such as high resolution video processing. The hybrid platform is expected to perform the sequential video encoding/decoding operations, mainly the entropy engine on the software part, while transforms are accelerated on the FPGA part.

1.3 Manuscript contributions

This document presents a set of proposals that has the final purpose of reducing the high complexity level introduced by the transform module, essentially for the VVC standard. It tackles the real time implementation problem of the different Discret Cosine/Sinus Transforms involved throughout the evolution of VVC standard, from AMT (5 types) in his first drafts to MTS (3 types) in the current version. The main objectives are reducing the computational complexity and achieving high frame rate processing for 4K videos taking into account the

hardware resource constraints and limitations (memories, logic use, operational frequency...). The main contributions of this thesis are the following:

1. Propose an efficient pipelined hardware implementation of the 2D AMT including the five transforms types of sizes 4x4, 8x8, 16x16 and 32x32. This implementation is considered as the first proposal of 2D AMT implementation for VVC in the literature. The proposed design methodology takes advantage of the internal software/hardware resources of the target FPGA device *Arria 10* such as Library of Parameterized Modules (LPM) core IPs [19] and Digital Signal Processing (DSP) blocks, aiming to reduce the logic utilization.A unified 2D architecture embeds all 1D 4x4, 8x8, 16x16 and 32x32 transform modules by taking into account the all asymmetric 2D block size combinations and is able to sustain 2K video coding at 50 frames per second with an operational frequency up to 147 MHz.
2. Due to MTS high complex design, an approximation approach is proposed to reduce the computational cost of the DST-VII and DCT-VIII. The approximation consists in applying adjustment stages, based on sparse block-band matrices, to a variant of DCT-II family mainly DCT-II and its inverse. The optimal coefficients of the adjustment matrices are derived using a genetic optimization algorithm. Moreover, an efficient hardware implementation of the forward and inverse approximate transform module that can be integrated in both hardware VVC encoder and decoder is proposed. The architecture design includes a pipelined and reconfigurable forward-inverse DCT-II core transform as it is the main core for DST-VII and DCT-VIII computation along with additional adjustment stage at low computational complexity and logic resource allocation.

   In terms of coding efficiency, the approximate DST-VII and DCT-VIII preserve the coding gain of the MTS. On the other hand, the proposed unified hardware architecture enables to reach a high frame rate while using a moderate hardware and logic resource of the *Arria10* FPGA device. It enables to process a video in HD and 4K resolutions at 386 and 96 fps, respectively.
3. A new coding tool named as Low Frequency Non Separable Transforms (LFNST) is recently incorporated in the VVC standard which is strongly related to the transform process to further improve the coding efficiency. In this document, first a study of the LFNST is provided focusing on its evolution throughout the standardization process until the current VVC draft. An efficient and optimized hardware implementation of LFNST is proposed based on non constant coefficient multiplication using ROM memory blocks and LPM Core IP multipliers. The proposed hardware architecture requires very moderate logic cost for the different design configurations. An alternative execution time optimization approach is also presented that enables to significantly reduce processing time with negligible increase in hardware resource requirements. Up

to the best of our knowledge, it is the first hardware implementation in the literature for VVC LFNST module.

1.4 Manuscript organization

This section presents the overall organization of this thesis.

Chapter 2 starts with an introduction to the basics of video coding and some essential concepts for this thesis, from the history of video compression across the main standards and codecs, to the classical hybrid video compression scheme. It presents a brief overview of the main tools of the state-of-the art HEVC and also then next potential VVC focusing mainly on the new coding tools and improvements enabling the coding gain expected.

In Chapter 3, the focus is quickly put on the transform stage where a background of transform process is discussed for HEVC and VVC standards. Then, important existing state-of-the-art proposals in transform optimization, approximation and hardware implementation essentially for VVC standard are presented as it is the main motivation and focus of this thesis. Finally, an overview of the target platform device: Intel SoC FPGA Arria 10 is also provided, focusing mainly on its hardware and software tools used for this work.

Chapter 4 interests in the implementation of the AMT process including five transform types. Firstly, a multiplierless design of 4-point VVC transform is described and detailed. Then, this work is extended to consider all the block sizes from 4 to 32 to propose an efficient architecture of the 2D AMT module as the first proposal in the literature. Experimental and synthesis results are discussed and compared with related works.

Chapter 5 describes the approximation approach adopted for the MTS concept. The approximation is first modeled as a constrained integer optimization problem. The genetic algorithm is then used to solve the problem and compute the adjustment matrices for large transform sizes $N \in \{16, 32\}$. A hardware implementation of the 2D forward- inverse approximate transform design is proposed where the architecture and the use of low-cost orthogonal adjustment matrices are detailed. The experimental and synthesis results of 1D and 2D implementations are also presented and discussed with respect to the related works.

Chapter 6 presents a Co design application of HEVC inverse DCT-II implementation on Arria 10 FPGA SoPC. The different software and hardware tools used in the co-design application are detailed. Communication bridges using Avalon Memory Mapped FiFOs are used to create the communication between software driver, responsible for sending/receiving data, and the hardware IP of the IDCT architecture described in VHDL programming language. All the implementation process and the state machine design adopted are described in this chapter.

Moreover, the experimental results in terms of execution time are provided and discussed for different configurations using 32-bit and 64-bit com-bus.

Chapter 7 starts with a detailed description of the new coding tool related to the transform operation, Low Frequency Non Separable Transform. A brief background of his integration throughout the VVC standardization is also presented. Then, the proposed solution to provide an efficient hardware implementation is illustrated using a non-constant coefficient multiplication approach through the use of ROMs and very low number of multipliers using LPM Core IPs. The experimental and synthesis results are provided and discussed for different design configurations.

Finally, chapter 8 concludes this work by summarizing the proposed contributions and providing future perspectives.

Chapter 2

Video Compression and Standards

This chapter presents the video coding history and fundamental concepts of video coding useful for this work. It also gives a background of the state of the art modern video coding standards HEVC and VVC with an emphasis on their specificities .

Contents

2.1 Introduction

This chapter introduces the video compression and its standardization history. Section 2.3 describes some fundamental concepts that are useful for video applications such as color space, different coding structures and the common test conditions applied to evaluate and compare video coding contributions. The most important and used metrics in video coding applications are described in section 2.4. Section 2.5 is dedicated to present a brief overview on the state-of-the-art video coding standard, High efficiency Video Coding explaining his coding scheme and the main blocks of the coding chain. In section 2.6, a brief overview of the future video coding standard VVC is provided focusing on the new introduced features and coding tools responsible for the achieved coding gain. Finally, Section 2.7 summarizes this chapter.

2.2 Video Compression history

The most internationally active bodies responsible for the standardization of video compression systems are ISO-IEC and ITU-T. The technical work of ISO-IEC is carried out within the Motion Picture Experts Group (MPEG) group which has defined the MPEG-1, MPEG-2, MPEG-4 standards for several applications including multimedia or television. In parallel with the MPEG activities, the Video Coding Experts Group (VCEG) of the ITU-T is particularly interested in the definition of technical recommendations for video conference and video telephony applications, this group has developed the H.26x standards as shown in Figure 2.1.

H.261 is the first video standard approved in 1990 [20]. It targets video phone applications. The processed image formats are CIF (288x352 pixels) and QCIF (144x176 pixels). The Mpeg-2/H.262 [21, 22] standard was jointly developed and approved in 1993 by the ISO and ITU standardization organizations across the MPEG and VCEG groups. This standard encountered success and massive adoption through DVD, broadcast/broadband industry such as high-definition Television DTV. This joint collaboration did not prevent these two groups from developing their own standards.

H.263 is a video coding standard for very low bit rate video communication, adopted in 1995 [23]. It targets video conferencing and video telephony applications. The H.263 standard

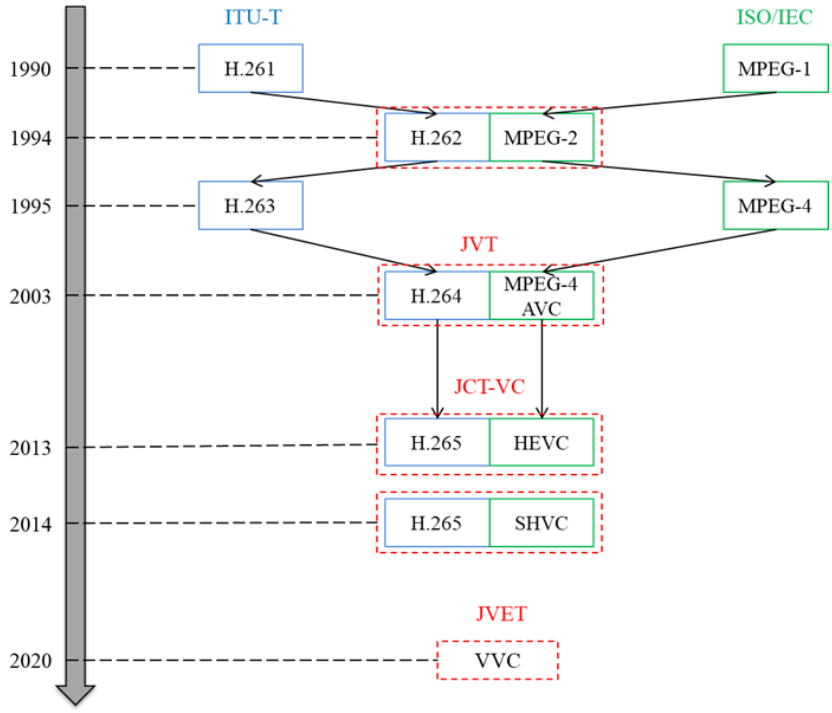

Figure 2.1 – Diagram of video coding standardization history

was then modified to give rise to two new versions, called respectively H.263+ finalized in 1998 and H.263++ finalized in 2000. This latter improved compression efficiency by 15- 25% over the first version and supported custom and flexible video formats.

In the same way, MPEG independently developed MPEG-4 [24] standard containing the Rec. ITU-T H.263 baseline design. It addresses a wide variety of audiovisual applications ranging from videoconferencing to audiovisual production. By considering the coding of traditional video, MPEG-4 combines the tools of MPEG-2 with novelties resulting in more efficient in compression and more resilient to transmission errors. Both ITU-T and ISO have created the Joint Video Team (JVT) in 2001 to develop to finalize a more efficient standard than MPEG-2, called H.264 [25]. The H.264 / AVC standard (AVC for Advanced Video Coding), still known as MPEG-4 Part 10, was first approved in May 2003 and is the most widely used codec especially in telephone companies and internet video providers. The H.264 / MPEG-4 AVC standard halves the transmission or storage rate for equivalent visual quality compared to previous standards. In January 2005, both VCEG and MPEG groups agreed to finalize the Scalable Video Coding (SVC) project as an amendment to their H.264 / MPEG-4 AVC standard.

Over the next few years, the H.264 / AVC standard has delivered the application needs of multimedia systems, offering higher performance compared to previous generation video encoders. However, its effectiveness is obviously insufficient for the storage and the trans-

mission of new video format with higher resolutions (QFHD and beyond). Subsequently, in 2010, the Joint Collaborative Team on Video Coding (JCT-VC) group, gathering both MPEG and VCEG experts, was created with the mandate of reducing by 50% the bitrate for the same visual quality compared to H.264/AVC. The HEVC / H.265 codec [26] was finalized in January 2013 enabling to double the rate for the same subjective video quality compared to H.264/AVC. In 2014, MPEG announced that the second edition of HEVC will be accompanied by three extensions that are the multiview extensions (MV-HEVC), range extensions (REext), and scalable HEVC extension (SHVC).

Finally, Joint Video Expert Team (JVET), established in 2015 gathering experts from MPEG and VCEG, is working on the successor of HEVC standard [11]. This new video coding standard is named Versatile Video Coding (VVC) [27] and is expected to be released by the end of 2020. In the same way, the main objective is to provide 50% bitrate reduction for the same perceived video quality with respect to HEVC standard.

2.3 Video formats and related applications

Digital video consist of a succession of frames at a certain speed of display named frame rate. It is known that displaying 25 frames per second (fps) is the minimum frame rate to trick the human eye to mimic an animated video. In addition, digital video adopts the characteristics of the frames such as color space, the resolution and the depth of data. Since the digital image is sampled along the two spatial axes, it can also be represented by a grid of elementary points called pixels.

2.3.1 Color space

In order to represent a frame in the RGB space (Red, Green, Blue) [28], each pixel consists of three components: red, green and blue. The addition of these 3 monochromatic colors allows to obtain all the other colors. There are other color spaces such as YUV or YCbCr (luminance, blue chrominance, red chrominance) that exploit the eye sensitivity to light more than to color. The migration from the RGB space to the YUV space is applied using an irreversible linear transformation given by the following equation (2.1).

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B, \\ Cb &= -0.169R - 0.331G + 0.5B, \\ Cr &= 0.5R - 0.419G - 0.081B \end{aligned} \tag{2.1}$$

Thus, the frame is defined by the three images having the Y, U and V components. Indeed, the luminance component Y refers to the gray level of a pixel while the two chrominance components U and V define the colors. Since chrominance has only a small impact on the human visual system (SVH) [29], it is clever to reduce the size of color images using sub-sampling, keeping only half of information (4: 2: 2) or only a quarter of the information (4: 2: 0).

- 4:4:4 In this format, there is no sub-sampling of the two chrominance components. Thus, the three luminance and chrominance components have the same size.
- 4:2:2 This format uses for each four luminance pixels (Y) two blue chrominance pixels (Cb) and two red chrominance pixels (Cr). Thus, the width of the two chrominance images is halved. This format allows to reduce the size of the image in RGB format.
- 4:2:0 This format is used to halve the height and width of the two chrominance images, and then the YUV image size is reduced with respect to the RGB one.

2.3.2 Configurations and coding structures

Digital video consists of a set of frames that follow one another at a certain speed namely I (for Intra), P (for Predicted) or B (for Bi-directionally predicted) frames. This set is called Group Of Pictures (GOP). There are mainly four configurations used in current applications of video coding:

2.3.2.1 All Intra

All Intra (AI) configuration, all GOPs are formed by I-frames as shown in Figure 2.2. There is no temporal dependency between the frames and it is usually less complex than others by avoiding the Motion Estimation (ME). This configuration allows fast coding but at low compression rates.

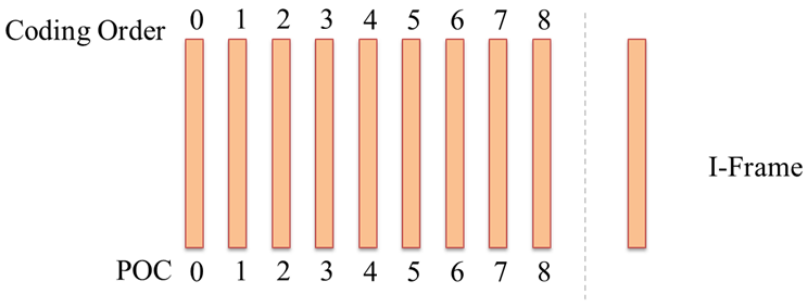

Figure 2.2 – All Intra structure

2.3.2.2 Random Access

In this configuration, a hierarchical structure of B frames is used as shown in Figure 2.3. Random Access configuration presents the highest coding efficiency among the others thanks to the bi-prediction. However, it generates a bitstream with higher latency because of the reordering of frames during encoding.

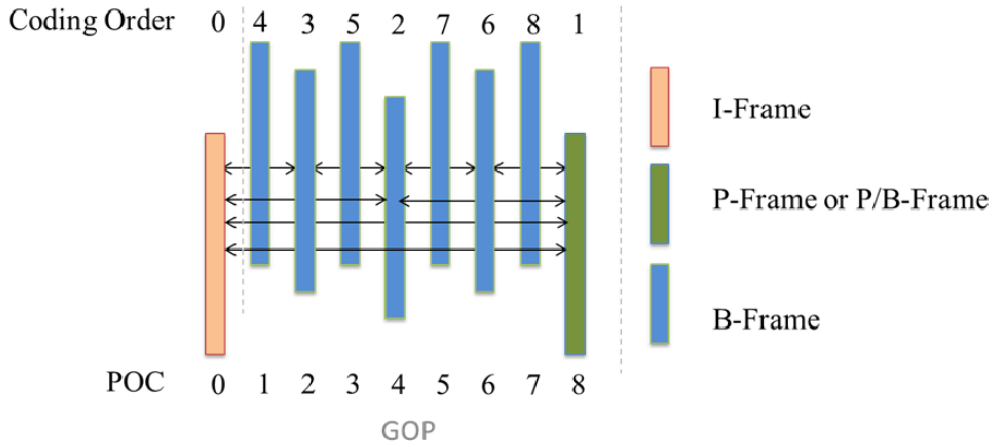

Figure 2.3 – Random Access structure

2.3.2.3 Low Delay P

Two types of frames are used in the LD-P configuration; I and P. The first frame of a GOP is an I frame followed by a large number of P-frames. In contrast to the RA configuration, frames are not reordered and the coding order is identical to the display one. This configuration presents a higher efficiency than the AI structure and a lower delay compared to RA configuration.

2.3.2.4 Low Delay B

Similar to the LD-P configuration, the first frame is encoded as an I frame followed by B-frames and the coding order remain identical. Moreover, the LD-B configuration achieves a better coding efficiency than LD-P thanks to the bi-prediction and offers a low delay coding. However, encoding process may be more complex for each frame, due to the additional coding options.

2.3.3 Common test Conditions

In order to allow fair comparison between video coding contributions, in the context of standardization, some Common Test Conditions (CTC) are provided along with each standard.

The CTC are a set of requirements to meet, so that different coding tools can be compared. It is also recommended to follow CTC recommendations and rules for fair comparison which the main important are the following:

- a defined number of Sequences spread into different. Each class represents a particular resolution (classA-E) or content type (classF is "screen content"). For instance, there are 6 classes (A, B, C, D, E and F) fixed in HEVC test conditions [30] and for VVC two additional classes of sequences related to higher resolutions A1, A2 are adopted [31].
- Resolution, framerate, bitdepth and number of frames to encode are fixed for each sequence.
- The Intra Period, i.e. the frequency of I-frame (Frame only composed of I-Slices) is also fixed and defined as a function of the framerate.
- Base QP are set to 22, 27, 32 and 37 for R-D curves achievement. Offsets between frames are also set based on the coding structure.

2.4 Video coding metrics

In order to achieve higher reduction of the data size, lossy coding techniques are used. They introduce a measurable difference between the source signal and the reconstructed one. The measure of this difference is called the Distortion and is noted by D. In order to qualify the efficiency of video coding, metrics for objective and subjective evaluations are used. Subjective metrics qualify a video based on scores given by a set of human testers. They are also used to define psycho-visual tools qualifying the video encoding based on human observations. Among the most commonly used objective metrics are Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) metrics that measure the distortion between two images using the peak-to-peak signal-to-noise ratio. There is also the Bjontegaard metric that calculates the average difference of PSNRs and the average difference in rates using rate/distortion curves.

2.4.1 PSNR

The most used and known distortion objective metric is the Peak Signal-to-Noise Ratio (PSNR) [32]. It is based on the Mean-Squared Error (MSE) that is also a distortion metric by itself quantifying the difference between the samples from two images. The MSE is defined by equation(2.2) where $O_{i,j}$ and $R_{i,j}$ respectively the original and reconstructed samples at the coordinates (i, j), M and N are the spatial dimensions of the 2D video signal and B is the number of bits per sample.

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (R_{i,j} - O_{i,j})^2 , \tag{2.2}$$

The PSNR is computed from the MSE using the equation(2.3) where $2^B - 1$ represents the maximum value of a sample with B the number of bits per sample. The PSNR is expressed in a logarithmic scale (dB) to cope with the wide range that signals might have. The higher its value, the better is the quality. The PSNR can be calculated on an image or on a video sequence.

$$PSNR = 20log_{10}(\frac{2^B - 1}{\sqrt{MSE}}). \tag{2.3}$$

2.4.2 SSIM

The SSIM metric consists in measuring the structural similarity between two images [33]. In fact, it measures the difference by structural areas and not by pixel unlike PSNR. The SSIM between two area x (source) and y (reconstructed) of samples is computed using equation(2.4) where μ_x is the average of x, μ_y is the average of y, σ_x is the variance of x, σ_y is variance of y, σ_{xy} is the covariance of x and y, c1 and c2 are two stabilization variables given by equation(2.5) with B the number of bits per sample and k_1 and k_2 two coefficients (set at 0.01 and 0.03 by default).

$$SSIM_{x,y} = \frac{(2\mu_x\mu_y + c1)(2\sigma_{xy} + c2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c2)}. \tag{2.4}$$

$$c1 = (k_1 2^B - 1)^2; c2 = (k_2 2^B - 1)^2. \tag{2.5}$$

It is considered more related to the perceived quality than the PSNR. This can be explained by the fact that the human eye is attracted to specific areas of the screen such as faces and would neglect other zones that might have a lower PSNR. The SSIM is a value between 0 and 1 indicating the correlation with respect to the source. The more SSIM is closer to 1, the better correlation is.

2.4.3 Bjeontegard measures

Comparing two video coding techniques objectively might be complicated, as both the distortion and the bitrate savings have to be taken into account jointly. Metrics introduced by Gisle Bjøntegaard, known as Bjøntegaard Delta (BD) measurements are the most widely

used metrics to compare the performance of encoding tools [34, 35].They measure the average difference between two Rate-Distortion (RD) curves interpolated through four bit-rate points.

- The Bjøntegaard Delta PSNR (BD-PSNR) measures the average PSNR difference in decibels (dB) for two different encoding algorithms considering the same bit rate.
- The Bjøntegaard Delta Bit Rate (BD-BR) reports the average bit rate difference in percent for two encodings at the same quality.

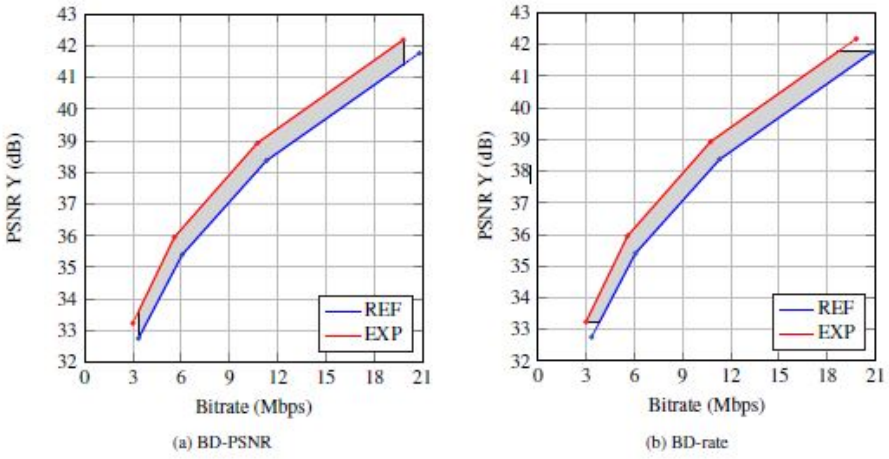

Figure 2.4 – BD-BR and BD-PSNR curves

Figure 2.4 depicts an example of BD-BR and BD-PSNR comparison of reference and experiment encoding labeled as REF and EXP, respectively.

2.4.4 Rate distortion optimazation

The decision of an encoder aim to estimate the coding parameters such as particular block size for the partitioning or a prediction mode that achieve the best rate distortion options. That is to say it checks the distortion that decision might cause as well as an estimation of the bitrate needed exploring different coding possibilities. Finally, it selects the one that provides the best score in terms of rate and distortion. This constrained problem of distortion minimization is named rate distortion optimization (RDO) [36] and can be modeled by Lagrangian method in equation(2.6) as follows:

$$J_\lambda = D + \lambda R. \tag{2.6}$$

Where J is the RD-cost that refers to the trade off between the distortion D and the rate R and λ is the Lagrangian weighting factor that control the RD-cost J. RDO process exhaustively tests all possible coding parameters to select the ones that minimize the RD-cost

J and compute them to the bitstream. That is why it is considered as the main source of high complexity.

2.5 Hybrid video coding Standard HEVC/H.265

The video coding scheme is called hybrid due to the combination of temporal prediction between frames of the video sequence and 2D spatial prediction within the frame. Inter/Intra Prediction, transformation, quantization and entropy coding are the basic processing steps within the video encoder to generate a compressed bit stream. Figure 2.5 shows a simplified structure of the hybrid video coding scheme.

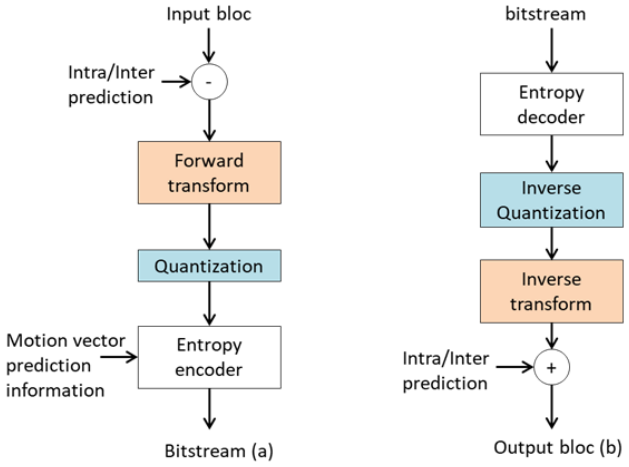

Figure 2.5 – Structure of the hybrid video coding scheme

The bitstearm will be the input of the decoder chain in order to regenerate the image once the inverse quantization and inverse transform are applied to reconstruct the residue.
HEVC is a block based hybrid video coding scheme as like as its predecessors [4, 37]. HEVC encoder starts with the division of an image into block-shaped regions using a QuadTree procedure [38]: the first image is always intra-coded as it will be used as reference. Otherwise, a predicted signal is generated for each block, using either the Intra-frame prediction or the Inter-frame prediction, named the Motion Compensated Prediction (MCP).
The difference between the current or the source image and the predicted one is called the residual signal and is further encoded by transform and quantization. Quantized transform coefficient plus the prediction information will subsequently be coded with Context-Adaptive Binary Arithmetic Coding (CABAC) [39] to generate the bitstream to ease the storage and transmission. Figure 2.6 depicts the HEVC codec structure.

The encoder uses a decoding algorithm (represented inside a blue box) to reconstruct the compressed images for use in temporal prediction. To do this, inverse quantization and

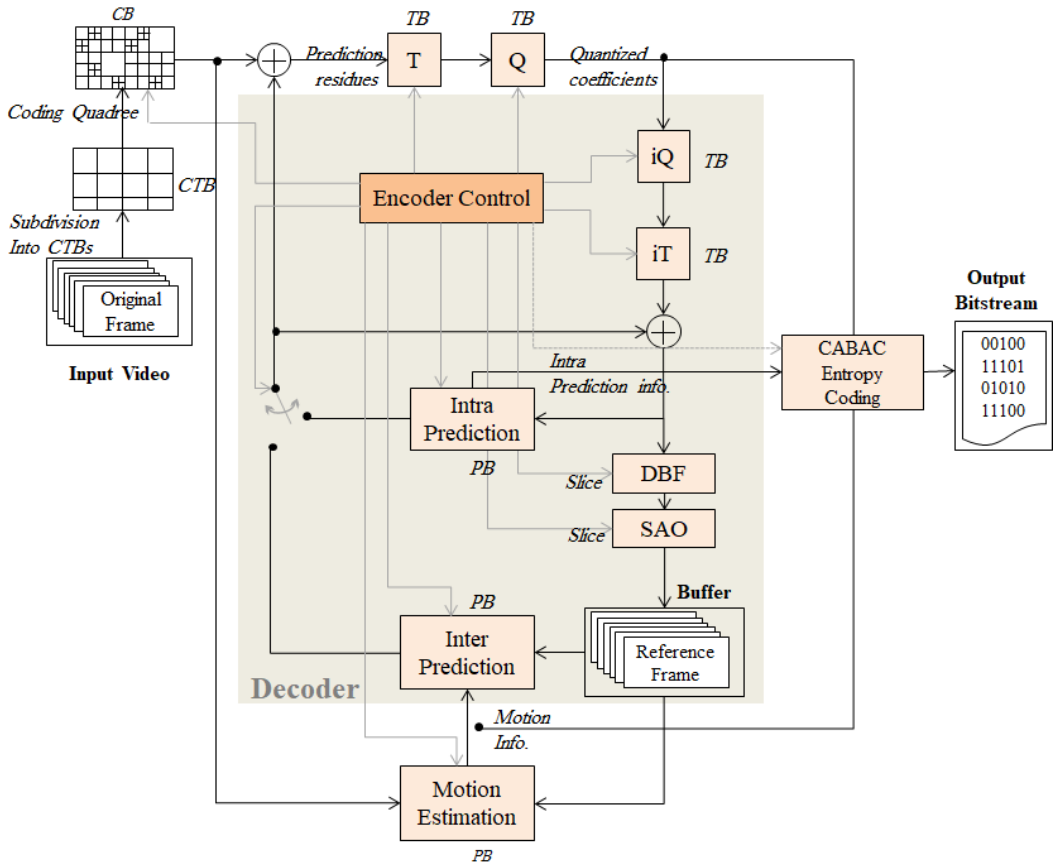

Figure 2.6 – Structure of the HEVC video coding scheme

inverse transform are applied to reconstruct the residue. This latter added to the predicted samples to generate the decoded frame. The result undergoes the filtering operation; the two in-loop filters in HEVC are the deblocking filter [40] used to smooth the distortions at the block boundaries and the Sample Adaptive Offset (SAO) [41] designed to lessen the ringing artifact. The final representation of the image is then stored in a "buffer" of references used in inter coding of the other images.

2.5.1 Partitioning

As shown in Figure 2.7, a GOP can be split to slices. A slice is a part of the frame that can be decoded independently of other slices constituting the frame.

For HEVC, each slice is divided into LxL square block sizes blocks (L = 16, 32 or 64) named Coding Tree Unit (CTU) itself could be divided in Coding Units (CU). This is included in the new partitioning method of the HEVC standard Quad Tree structure that introduces a larger and more flexible block structure with three essential units [42] :

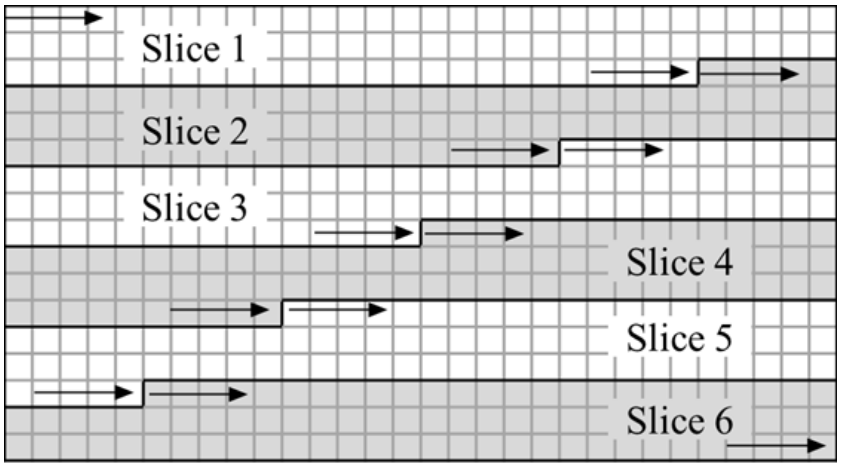

Figure 2.7 – Example of frame partitioning to slices

2.5.1.1 Coding Unit CU

The coding unit (CU) is a square region, represented as the leaf node of a quadtree partitioning of the CTU, which shares the same prediction mode. The quadtree partitioning structure allows recursive splitting into four equally sized nodes, starting from the CTU. The possible size of a CU is 8x8, 16x16, 32x32 or 64x64 depending on the hierarchical depth p as illustrated Figure 2.8.

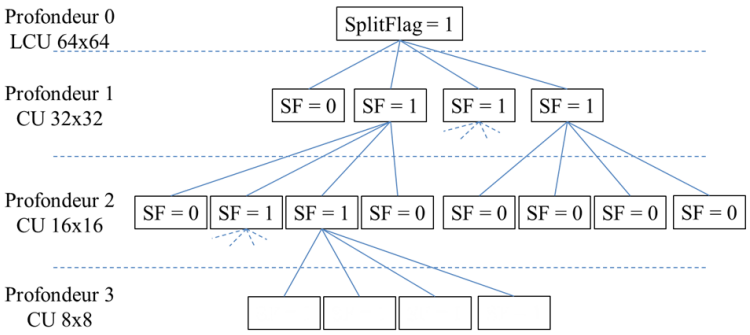

Figure 2.8 – Different Coding Unit depths

If the Split Flag (SF) is equal to '0', the CU of size 2Nx2N is coded in its depthp. Otherwise, the coding unit is subdivided into 4 independent CUs of depth p+1 and size NxN. This representation allows a better flexibility of the best CU size based on well-defined criteria and parameters such as RDcost.

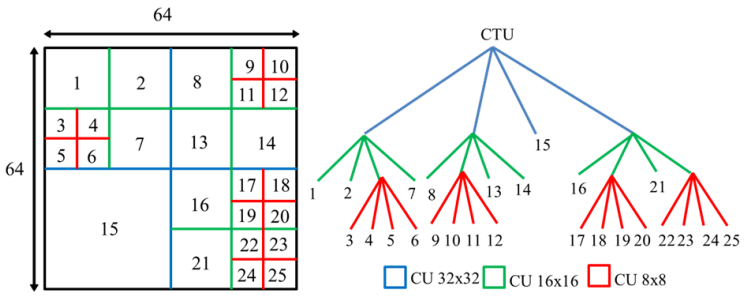

Figure 2.9 – Exmaple of CTU partition to CUs

2.5.1.2 Prediction Unit PU

Each CU at a given depth of the QuadTree can be predicted in one, two or four partitions, named PU. In general, the PU is not restricted to being square in shape, in order to facilitate partitioning which matches the boundaries of real objects in the picture. The prediction is applied on each PU independently, whatever the number of PUs within the CU. However, all PUs that belong to the same CU use the same type of prediction (Intra or Inter).
For inter coding modes, there are four symmetrical PU partitions (2Nx2N, 2NxN, Nx2N, NxN) and four asymmetrical partitions (2NxnU, 2NxnD, nLx2N and nRx2N) as presented in Figure 2.10. Intra coding modes only support the squared PUs, hence 2Nx2N or NxN.

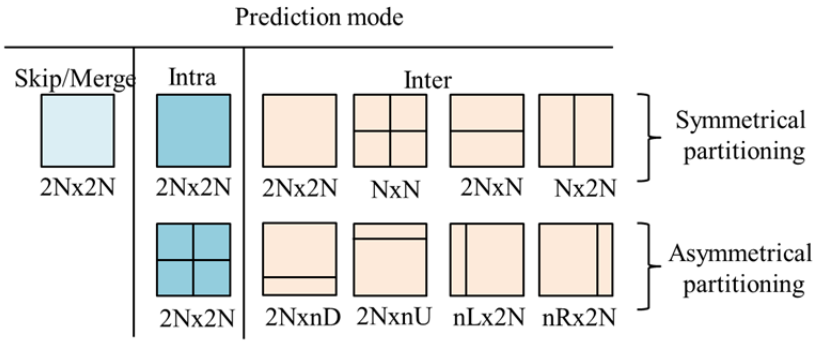

Figure 2.10 – Possible partition modes for PUs

2.5.1.3 Transform Unit TU

This is the basic unit for the Transform and Quantification processes. The size of a TU depends on that of its CU, it can exceed the size of the PU but never the size of its corresponding CU and can be from 4x4 up to 32x32.

2.5.2 Inter/Intra prediction

2.5.2.1 Intra frame prediction

The Intra prediction, sometimes referred to as spatial prediction, consists in reducing the spatial redundancies in the frame. The basic principle of intra prediction is based on the fact that the texture of a picture region is similar to the texture of its neighborhood and can be predicted from there. HEVC provides more flexibility for intra prediction as it apolied to PUs of size 4x4 to 32x32 and with 35 Intra prediction modes; DC mode that predicts one sample from an average of the neighboring samples, Planar mode where a weighted average of four reference samples generate the prediction of one sample and 33 directional modes [43, 4].

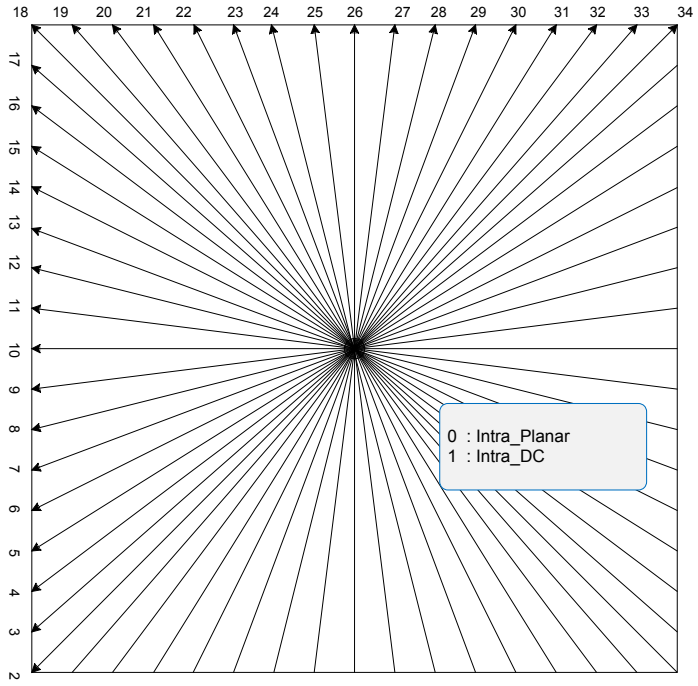

Figure 2.11 – Mapping between intra prediction direction and intra prediction mode

For directional modes, the pixels of the predicted block are interpolated at a defined angle as shown in Figure 2.11. It should be noted that the modes 10 and 26 which respectively correspond to the horizontal and vertical modes are the most used. The Intra predictor can be coded in two ways: First is the 3 Most Probable Modes (MPMs) method using the surrounding PUs coding modes, i.e. left and above. In case the considered Intra predictor matches one of the predictors included in the MPMs, only the index within this reduced set is transmitted. The second option, in the case MPMs are not used, is using a fixed length code of 5 bits. More details are presented in [44].

2.5.2.2 Inter frame prediction

Inter prediction reduces redundant information using Motion Compensation (MC), which is based on the fact that the difference between adjacent frames in the video sequence results from camera and object motions. In the encoder side, The Motion Estimation (ME) stage is carried out. It searches the best matching area in the reference picture for the current prediction block. It is one of the most complex parts of video coders in terms of computational requirements. Once a good prediction has been found, a motion vector is created, indicating the offset that has to be applied in the block from the reference picture.
In HEVC, ME process uses multiple reference frames where the reconstructed frames are split into two reference lists: List 0 and List 1. List 0 is composed of the indices of past encoded frames in display order whereas List 1 is composed of the indices of future encoded frames in display order. For inter prediction, P slices are predicted using only list 0 and B slices are predicted one or two reference frames of both list 0 and list 1. In the case of bi-prediction of B slices, the predictions performed from List 0 and from List 1 are averaged. There are several block-matching ME algorithms to search for the best matching leading to the optimal solution. The primitive algorithm Full Search (FS) is an exhaustive algorithm that consists in testing all possible block candidates of the search window. Faster approaches have been developed to reduce the computational complexity of this process such as Three Step Search (TSS) [45], 2-D Logarithmic Search [46], Four Step Search (FSS) [47]... These algorithms provide good motion estimation performance for relatively small search windows with more or less regular motion. The algorithm TZ Search is the fast search algorithm adopted in the standard HEVC. The block matching method aims to minimize a certain distortion criterion used in the RD cost calculation such as (SAD) and Sum Square Differences (SSD) computed by the following equations:

$$\begin{aligned} SAD &= \sum_{i,j} |P_A(i,j) - P_B(i,j)| \\ SSD &= \sum_{i,j} (P_A(i,j) - P_B(i,j))^2 \end{aligned} \tag{2.7}$$

Where$P_A(i,j)$ and $P_B(i,j)$ are the pixels of same sized blocks A and B at the position (i, j), respectively

2.5.3 Transform

The goal of the transform is to convert the values from the spatial to the frequency domain. In the spatial domain, the residual signal is spread among the samples of the blocks, while

the objective of transform domains is to concentrate as much as possible the residual signal in the top left corner (low frequencies) carrying the most significant information. This idea of the energy compaction is the main property of the transform stage.
Most of the transforms used in standardized video coding schemes belong to the Discrete Trigonometric Transform (DTT) family. HEVC uses an integer Discrete Cosine Transform (DCT)-based transform according to the TU size ranging from 4x4 to 32x32 [48]. An alternative 4x4 integer Discrete Sine Transform (DST)-based transform is applied to 4x4 residue blocks resulting from Intra prediction. Due to the statistical property of the residual signal, the use of DST-based transform to encode Intra predicted blocks of size 4x4 obtains better results.

2.5.4 Quantization

After transformation, the transformed coefficients are quantized using a non-linear discrete mapping of the coefficient values into integer quantization indices reducing the amount of possible output values [49]. The quantization is scalar: each coefficient is approximated independently of its neighboring values. In HEVC, the quantization step is controlled by a Quantization Parameter (QP) that discards any coefficient whose energy level is below a certain threshold.

2.5.5 In loop filters

HEVC includes two processing stages in the in-loop filter: a deblocking filter [40] and then a Sample Adaptive Offset (SAO) filter [41].
The deblocking filter aims to reduce the visibility of blocking artefacts caused by the block-based approach in HEVC partitioning. It is applied only to samples located at block boundaries. A deblocking filter process is performed for each CU in the same order as the decoding process. First vertical edges are filtered (horizontal filtering) then horizontal edges are filtered (vertical filtering).
The SAO filter is applied after the deblocking filter and aims to improve the accuracy of the reconstruction of the original signal amplitudes. The main function of SAO filtering is to reduce visible ringing artifact effects. SAO filter classifies the reconstructed samples in two categories: Band Offset (BO) or Edges Offset (EO). The reconstructed samples are classified in EO categories based on sample value differences of the current sample and its neighboring samples. According to edge direction on the block boundary, an estimated offset is applied to the reconstructed sample to avoid the artifacts located at the block boundaries. The BO classified samples are only evaluated according to the band samples and an offset is also applied to the reconstructed sample.

2.5.6 Entropy coding

Entropy coding is the last step in the coding process (the first in the decoding process) that converts the video signal to a series of syntax elements. These latter describe how to reconstruct the input video signal, including the prediction information. In HEVC, the entropy coding consists of the following steps; each syntax element is firstly converted into binary symbols. Then, context modeling estimates the probability of each symbol, coded according to its specific context [39]. Finally, whatever the probability, arithmetic coding is used in order to obtain the bit-stream. More information can be found in [4, 50, 51].

2.6 Future Video Standard: Versatile Video Coding VVC specificities

Since 2016, JVET has continued research and development toward the next generation of MPEG-based video codec beyond HEVC under the JEM software [52, 53, 54, 55, 56]. JEM is built on-top of the HM software and includes several new normative, i.e. that implies decoder modification, coding tools which are candidates for VVC standard adoption [13]. It provides reference decoder and encoder, and is identical to the HM in its design and working principle. Although it presented a significant coding gain in terms of bit rate reduction, it introduces as well high computation and run time cost with respect to HEVC.
The development of VVC standard has been officially started in April 2018 based on both HM and JEM references as anchors in response to the CfP issued by JVET [11]. A new reference software, Versatile Test Model (VTM) has been released along with a first draft of VVC specification [57]. VTM version 1.0 consists in a reduced version of HEVC, removing elements that are unnecessary, inefficient, or inappropriate, but using new elements common to many proposals in response to CfP [58]. This new software code base has been motivated by having a clean-slate design as compared to HEVC. As the standardization process is in progress through enhanced coding tools investigations, the VVC software, Versatile Test Model VTM (which is currently under his 6th draft version [59]) is becoming as competitive as the JEM with reduced computation cost and encoder/decoder run times as shown in Figure 2.12.

2.6.1 Block partitioning improvements

HEVC introduced the flexible quad-tree partitioning of the coding tree units (CTUs) into coding units (CUs), prediction units (PUs) and transform units (TUs) of different sizes. This partitioning scheme is replaced in JEM by a quad-tree plus binary-tree (QTBT) block structure. CTUs are partitioned using a quad-tree followed by a binary tree. The binary

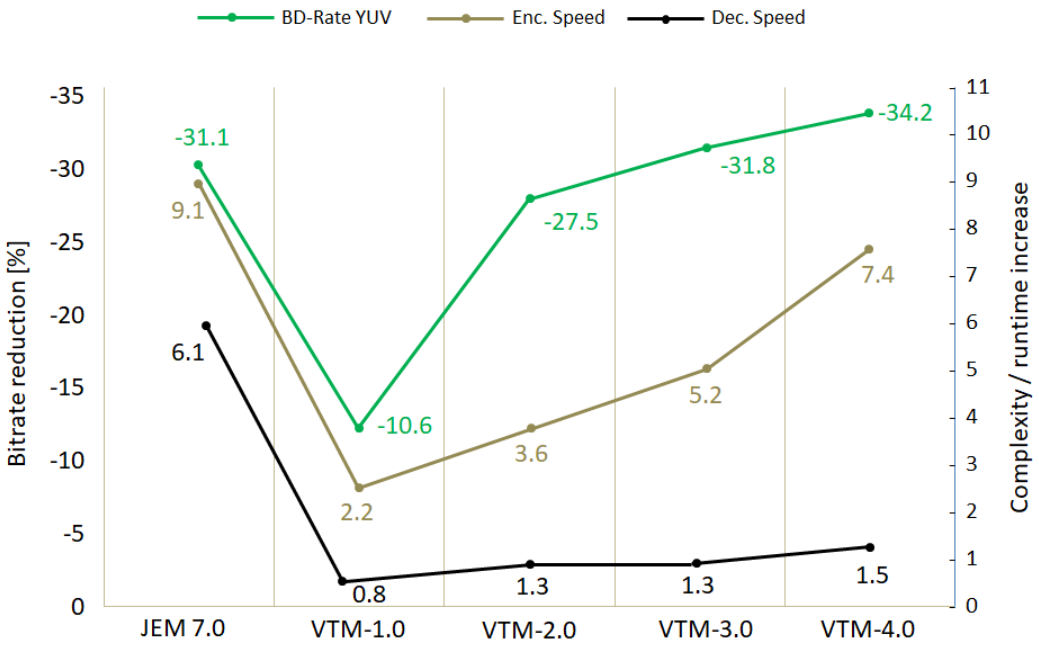

Figure 2.12 – Comparaison of coding efficiency and complexity throughout the VVC stadardization process [60]

tree allows further dividing the square blocks (i.e. CUs in HEVC) into rectangles. Hence, the HEVC CUs, PUs and TUs paradigm disappears in JEM as the prediction and transform areas are of same size. In addition to a more flexible partitioning for prediction, QTBT allows saving overhead signaling (i.e. bits) of having three independent coding structures; only the CU structure remains. Moreover, JEM enables bigger CTUs, whose maximal size is increased from 64x64 to 128x128 to account for bigger frame resolution (e.g. 8K) [13].
On the other hand, in VVC (draft6), the quadtree is enhanced with nested multi-type tree (MTT) using binary and ternary splits segmentation structure that replaces the concepts of multiple partition unit types (CU, PU and TU) to support more flexibility for CU partition shapes. In the coding tree structure, a CU can have either a square or rectangular shape. A coding tree unit (CTU) is first partitioned by a quaternary tree (a.k.a. quadtree) structure. Then the quaternary tree leaf nodes can be further partitioned by a multi-type tree structure [59]. The multi-type tree leaf nodes are called coding units (CUs), and unless the CU is too large for the maximum transform length, this segmentation is used for prediction and transform processing without any further partitioning. Figure 2.13 shows a CTU divided into multiple CUs with a quadtree and nested multi-type tree coding block structure, where the bold block edges represent quadtree partitioning and the remaining edges represent multi-type tree partitioning.

The quadtree with nested multi-type tree partition provides a content-adaptive coding tree structure comprised of CUs. The size of the CU may be as large as the CTU or as small as

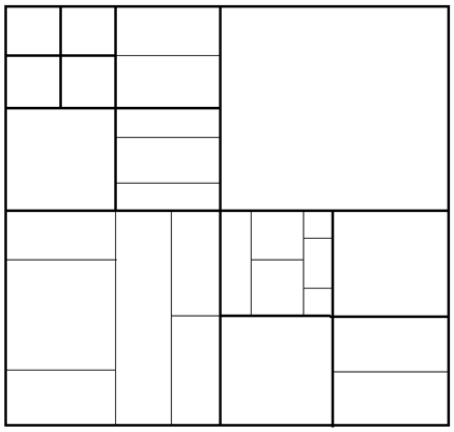

Figure 2.13 – Example of quadtree with nested multi-type tree coding block structure

4x4 in units of luma samples. For the case of the 4:2:0 chroma format, the maximum chroma CB size is 64x64 and the minimum chroma CB size is 2x2.

2.6.2 Intra prediction improvements

For the new video coding standard VVC, the number of angular modes is extended from 32 to 65, for a total of 67 intra prediction modes, including the Planar and DC predictors as shown in Figure 2.14.

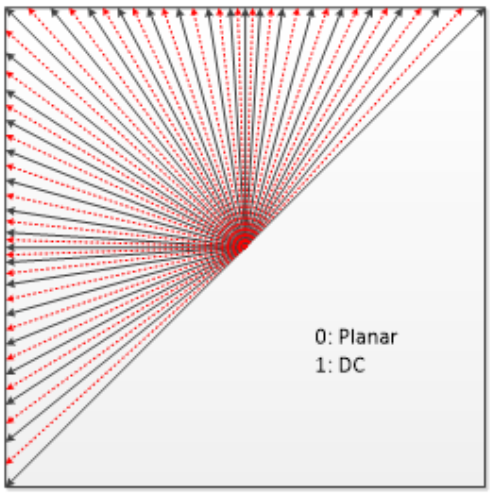

Figure 2.14 – Extended intra prediciton modes to 67

To keep the complexity of the most probable mode (MPM) list generation low, an intra-mode index coding method with 6 Most Probable Modes (MPMs) is used instead of 3 for HEVC [59]. In addition, the precision of the interpolation filters used to generate the intra prediction block with angular prediction modes is increased by using 4-tap in VTM (as for

VVC) instead of 2-tap filters (in HEVC) to improve boundary smoothing with previous reconstructed samples. The results of intra prediction of planar mode are further modified by a position dependent intra prediction combination (PDPC) method. PDPC is an intra-prediction method which invokes a combination of the un-filtered boundary reference samples and HEVC style intra prediction with filtered boundary reference samples. To reduce the cross-component redundancy, a Cross-Component Linear Model (CCLM) prediction mode is used, for which the chroma samples are predicted based on the reconstructed luma samples of the same CU by using a linear model.
Additional new features dedicated to intra prediction are adopted on top of VTM reference where detailed explanations are provided in [59] as the following:

- Wide-angle intra prediction for non-square blocks while keeping the total number of intra prediction modes (67) and intra mode coding method unchanged.
- Multiple reference line (MRL) intra prediction.
- Intra sub-partitions enabling 1D intra prediction and transform blocks.
- Matrix weighted Intra Prediction (MIP).

2.6.3 Inter prediction improvements

In HEVC, motion vector differences (MVDs), between the motion vector and predicted motion vector of a PU, are signaled in units of quarter luma samples. In the JEM, a Locally Adaptive Motion Vector Resolution (LAMVR) is introduced. Thereby, MVD can be coded in units of quarter luma samples, integer luma samples or four luma samples. It helps to reduced motion information coding cost. Additionally, the precision of internal motion vector storage is increased to 1/16 pel for Luma, and 1/32 pel for Chroma. Overlapped Block Motion Compensation (OBMC) is implemented in JEM [13]. Moreover, two Sub-CU motion vector prediction methods are considered: Alternative Temporal Motion Vector Prediction (ATMVP), and Spatial-Temporal Motion Vector Prediction (STMVP). These methods allow splitting larger CUs into smaller Sub-CUs and predicting a more accurate motion vector field for these sub-CUs, fetching additional motion predictors from past coded data.
For VVC standard (VTM), features introduced in JEM are whether modified or removed. As a result, beyond inter coding features in HEVC, the VTM software in his 6th draft includes the following inter prediction coding tools list [59]:

- Extended merge prediction.
- Merge mode with MVD (MMVD).
- AMVP mode with symmetric MVD signalling.
- Affine motion compensated prediction.
- Subblock-based temporal motion vector prediction (SbTMVP).
- Adaptive motion vector resolution (AMVR).

- Motion field storage: 1/16th luma sample MV storage and 8x8 motion field compression.
- Bi-prediction with CU-level weight (BCW).
- Bi-directional optical flow (BDOF).
- Decoder side motion vector refinement (DMVR).
- Triangle partition prediction.
- Combined inter and intra prediction (CIIP).

2.6.4 Transform improvements

Transform module witnessed many changes throughout the standardization process. HEVC transform is based on a Discrete Cosine Transform (DCT-II), apart for Intra-coded 4x4 TUs for which a Discrete Sine Transform (DST-VII) is used. In comparison, in JEM, four additional core transforms from the DCT and DST families (DCT-V, DCT-VII, DST-I and DST-VII) are introduced and put in competition to select the best transform type [13]. The transform types list is reduced from 5 types to only three (DCT-II, DCT-VII and DST-VII) in the VTM due to efficiency/complexity causes [59].
In addition, an alternative transform module interested in non-separable transforms is introduced in both JEM and VTM references under the name of Mode-Dependent Non-Separable Secondary Transform (MDNSST) and Low Frequency Non Separable Transforms (LFNST), respectively. It is applied between the core transform and the quantization (at encoder side), with the motivation to reduce remaining dependencies after the separable core transforms which only address horizontal and vertical signal dependencies. The transform process is thoroughly discussed and detailed in the next chapters.

2.6.5 Filtering improvements

In addition to Deblocking and Sample Adaptive Offset (SAO) filters in HEVC, two more in-loop filters are applied in the JEM: Bilateral filter (BLF) and Adaptive Loop filter (ALF). They are processed in the following order: bilateral and debloking filters, then SAO and ALF. However, in VTM, as the current official VVC reference, only ALF filter is incorporated and BLF is no longer considered.
ALF consists in the derivation of Wiener filters to minimize distortion between reconstructed and original signals. Two diamond filter shapes are used where one among 25 filters is selected for each 4x4 block, based on the direction and activity of local gradients. Figure 2.15 illustrates the ALF filter shapes used in VVC. The 7x7 diamond shape is applied for luma component and the 5x5 diamond shape is applied for chroma components [59].

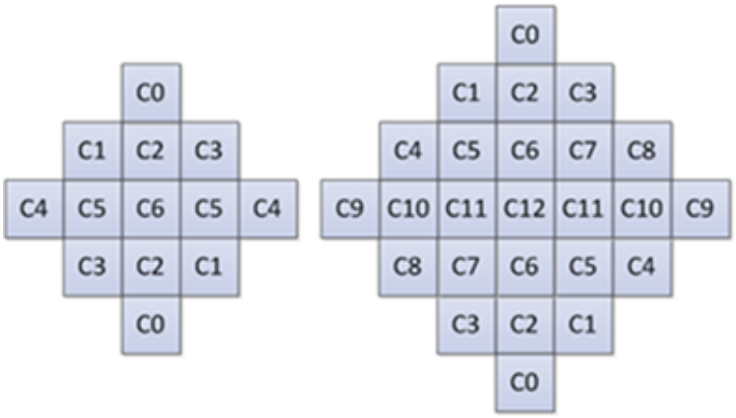

Figure 2.15 – ALF filter shapes: 5x5 for chroma and 7x7 for luma

2.6.6 Entropy coding improvements

For VVC standard (VTM), the CABAC technique is improved in comparison to the HEVC design. The major modifications involved the CABAC core engine and modified context modeling for transform coefficients as well as transform skip block.

2.7 Conclusion

This section presented a general overview of video compression environment. In the last three decades, ISO and ITU-T are the most known standardization organizations starting from H.261 and MPEG. The history of compression and fundamental notions for video applications are described in the first sections of this chapter. Indeed, ISO and ITU-T have united forces to develop standards through the years until the HEVC release as the latest official video coding standard, and even further currently as they are working on the emerging Versatile Video Coding standard which is expected by the end of 2020. Sections 2.5 and 2.6 present a background of the state-of-the-art modern video coding standards HEVC and VVC with an emphasis on their specifications caracterizing the basic processing blocks of the encoding chain: block partitioning, inter/intra prediction, transformation, quantization and entropy coding.

Chapter 3

Background of transform module in HEVC and VVC standards

This chapter focuses on the transform coding tool in the modern video coding standard and the different state of the art hardware implementations in the literature.

Contents

3.1 Introduction

The coding tools, developed in VVC, enable to increase the coding efficiency by 35% compared to HEVC. This gain is the sum of several improvements in the coding chain modules including the transformation process which is one of the key tools of the hybrid codecs from AVC to

VVC video coding standards. However, much higher complexity and resource requirements are introduced in parallel to the coding gain challenging especially real time implementations. One of the ways to face the aforementioned computational cost of the new codecs is to provide hardware accelerations for the high computational processes such as transform module which requires a significant operational and resource cost. Nowadays, the new created Soc FPGAs such as Arria 10 Soc with their enhanced features are adequate for the implementation of complex System On Chip (SOC) designs used in video processing applications.
In this chapter, the focus is put on the transform coding block. Section 3.2 presents firstly a brief overview of the transform and its characteristics. Next, a background on the transform process adopted in different modern video standards, from the DCT-II used in HEVC to the multiple transform competition concepts in VVC, is described. Section 3.3 is dedicated to the related works in the literature on hardware implementation of the transform, as well as its approximation approaches, for HEVC and VVC standards. Section 3.4 presents an overview of the target FPGA platform Arria 10 SoC used in the proposals of this work. Finally, section 3.5 concludes this chapter.

3.2 Transform block in modern video Codecs

Modern video coding standards, from AVC to the current VVC, apply transform coding on the prediction error residual in a similar manner as in hybrid codec scheme where the residual block is partitioned into multiple Transform Blocks (TB)s. Most of the transforms used in standardized video coding schemes belong to the Discrete Trigonometric Transform (DTT) family which are derived originally from the Karhunen-Loève Transform (KLT). The KLT is defined as the linear orthogonal transform that reduces the redundancy by a maximum decorrelation of the data, so that the signal can be stored more efficiently [61].
The Discrete Cosine Transform (DCT) approximates the KLT for image signals and provides an efficient implementation. As a result, it is considered as the preferred in image and video coding algorithms to decorrelate the signals and provide optimal bit allocation [47, 4].

3.2.1 Orthogonality

Transforms used in image processing and video coding systems are orthogonal. Orthogonal matrices H are square matrices whose rows and columns are orthogonal unit vectors, also known as orthonormal vectors, with:

$$H \cdot H^T = H^T \cdot H = I. \tag{3.1}$$

Subsequently, considering the fact that the inverse matrix of an orthogonal matrix is its transposed version, this property does offer some interesting benefits:

- Better energy compaction of the residual.
- Fast computation of inverse transform with no need to store it separately.
- Re-use of fast algorithms for both direct and inverse transform applications.
- Energy preservation.

3.2.2 Separability

Transform blocks used in video compression are two dimensional signals. The straightforward approach to work with signals is to use non-separable transforms. These transforms take the residual samples from a block previously reshaped into a single-dimensional signal. The main disadvantage of this approach is the number of calculations required to obtain the transformed signal. Indeed, for an NxN block, the number of operations required to transform it in a non-separable way is N^4 multiplications and $N^2(N^2\text{-}1)$ additions.
Due to the high amount of operations needed to transform a block using non-separable transforms, separable transforms are widely used in video coding reducing the number of required operations to $2N^3$ multiplications and $2N^2$(N-1) additions. Subsequently, two-dimensional transforms are computed by applying 1-D transforms in the horizontal and vertical directions separably.
For the MxN input block B, the 1D horizontal transform of the M rows of B is computed as given in equation (3.2)

$$Y_{int} = H_H \cdot B^T, \tag{3.2}$$

where H_H is the NxN matrix of the horizontal transform coefficients and $\cdot$ is the matrix multiplication.
The 1D vertical transform of the N columns of Y_{int} is performed by a matrix multiplication between the intermediate output coefficients (Y_{int}) and the matrix of the vertical transform coefficients H_V of size MxM, as given in equation (3.3) .

$$Y = H_V \cdot Y_{int}^T. \tag{3.3}$$

Equation (3.4) describes the 2D transform operation by computing the transformed coefficients Y of the input residuals block B.

$$Y = H_V \cdot (H_H \cdot B^T)^T. \tag{3.4}$$

3.2.3 HEVC Transform module

The elements of the core transform matrices used in HEVC were derived by approximating scaled DCT basis functions according to the TB size ranging from 4x4 to 32x32, under considerations such as limiting the necessary dynamic range for transform computation and maximizing the precision and closeness to orthogonality when the matrix entries are specified as integer values.

For simplicity, only one integer matrix for the length of 32 points is specified, and subsampled versions are used for other sizes. For example, the matrix for the length-16 transform is as shown in the following equation [4].

$$C_2^{16} = \begin{pmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & 90 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\ 87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\ 75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\ 70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\ 43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 & 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\ 9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 & 90 & -87 & 80 & -70 & 57 & -43 & 25 & -9 \end{pmatrix} \tag{3.5}$$

The matrices for the length-4 and length-8 transforms can be derived by using the first eight entries of rows 0, 2, 4, ...,7 and using the first four entries of rows 0, 1, 2, 3, respectively. Although the standard specifies the transform simply in terms of the value of a matrix, the values of the entries in the matrix were selected to have key symmetry properties that enable fast partially factored implementations with far fewer mathematical operations than an ordinary matrix multiplication.

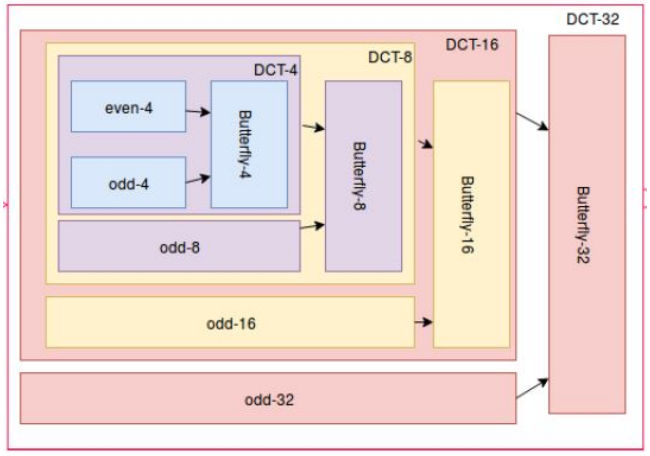

Figure 3.1 – 32-point DCT-II scheme using symmetry and recursion properties

With its recursion property, the larger transforms can be constructed by using the smaller transforms as building blocks which is beneficial especially for hardware implementation. Figure 3.1 depicts a simplified scheme of 32-point DCT-II using symmetry (butterfly decomposition structures) and recursion properties.
Due to the increased size of the supported transforms, limiting the dynamic range of the intermediate results from the first stage of the transformation is quite important. HEVC explicitly inserts a 7-bit right shift and 16-bit clipping operation after the first 1-D inverse transform stage of the transform (the vertical inverse transform stage) to ensure that all intermediate values can be stored in 16-bit memory (for 8-bit video decoding).
For a 4x4 transform block size, an alternative integer transform derived from a DST is applied to the luma residual blocks for intra-picture prediction modes, with the transform matrix. For some contents, DST is the optimal transform with performance close to KLT [4]. It is proved that DST has the similar form with the KLT for the directional intra prediction residues, whenever the correlation matrix is modelled by a separable, directional, and an isotropic image correlation model [4]. Indeed, the basis functions of the DST better fit the statistical property that the residual amplitudes tend to increase as the distance from the boundary samples that are used as reference for prediction becomes larger. In terms of complexity, the 4x4 DST-style transform is not much more computationally demanding than the 4x4 DCT-style transform, and it provides approximately 1% bit-rate reduction in intra-picture predictive coding. However, the usage of the DST type of transform is restricted to only 4x4 luma transform blocks, since for other cases the additional coding efficiency improvement for including the additional transform type was found to be marginal [4].
In response to JVET call for proposals to develop new coding standard outperforming the HEVC standard, the transform module is one of the coding tools that witnessed many modification proposals to test in order to boost it coding efficiency. The most interesting ideas are extending the CTU size to 256x256 with larger transform block size up to 64x64 instead of 32x32 in HEVC. This was supported by a the new partitioning structure QTBT (quad-tree + Binary-tree). Furthermore, in addition to the classic DCT-II transform used in HEVC, additional transform types of DCT/DST family were suggested to put in competition along with the DCT-II to select the best type enabling better efficiency in terms of RD-cost. This new concept is called Adaptive Multiple Transform (AMT).

3.2.4 Adaptive multiple Transform

Although the HEVC standard is based on the the well-known DCT type II (DCT-II) as the main transform function and the DST type VII (DST-VII) for Intra blocks of size 4x4, the use of trigonometric transforms has been extended In the early versions of the VVC standard development (JEM software). New approach of transform called Adaptive Multiple Transform is introduced and includes a total of five transform types: DCT-II, DCT-V, DCT-VII, DST-I

and DST-VII. The different transform basis functions of the DCT/DST types are computed as follows [13].

$$C_{2\,i,j}^{N} = \gamma_i \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(i-1)(2j-1)}{2N}\right), \tag{3.6}$$

with $\gamma_i = \begin{cases} \sqrt{\frac{1}{2}} & i = 1 \\ 1 & i \in \{2, \ldots, N\} \end{cases}$.

$$C_{5\,i,j}^{N} = \gamma_i \gamma_j \sqrt{\frac{4}{2N-1}} \cos\left(\frac{2\pi ij}{2N-1}\right), \tag{3.7}$$

with $\gamma_i = \begin{cases} \sqrt{\frac{1}{2}} & i = 1 \\ 1 & i \in \{2, \ldots, N\} \end{cases}$,

$\gamma_j = \begin{cases} \sqrt{\frac{1}{2}} & j = 1 \\ 1 & j \in \{2, \ldots, N\} \end{cases}$

$$S_{1\,i,j}^{N} = \sqrt{\frac{2}{N+1}} \sin\left(\frac{\pi(i+1)(j+1)}{N+1}\right). \tag{3.8}$$

$$S_{7\,i,j}^{N} = \sqrt{\frac{4}{2N+1}} \sin\left(\frac{\pi(2i-1)j}{2N+1}\right). \tag{3.9}$$

$$C_{8\,i,j}^{N} = \sqrt{\frac{4}{2N+1}} \cos\left(\frac{\pi(2i-1)(2j-1)}{2(2N+1)}\right), \tag{3.10}$$

with $(i, j) \in \{1, 2, \ldots, N\}^2$ and N is the transform size.
The AMT algorithm is applied at the block level on intra and inter prediction residuals. A specific *CU-level* flag is added in the bitstream to signal whether single or multiple transforms are used. If the *CU-level* flag is equal to 0, the classic HEVC transforms (DCT-II and DST-VII) are applied, otherwise two additional flags are added for signaling the horizontal and vertical transforms, used for the current CU [13].
For Intra prediction mode, an intra mode-dependent transform candidate selection is applied. According to the selected intra mode, a transform subset is identified as presented in Table3.2 and 3.1.

For inter prediction, DST-VII and DCT-VIII can be used for horizontal and vertical transforms. For both Inter and Intra CU blocks, the JEM encoder encodes with all transforms within the corresponding subset and then selects the one that minimizes the rate distortion cost. Related to their magnitude characteristics, the combinations of these transform types improve, in significant manner, the flexibility of the transform design [15]. However, the fact that five

Table 3.1 – Selected Horizontal (H) and Vertical (V) transform sets for each intra mode

Intra Mode	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
V	2	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0
H	2	1	0	1	0	1	0	1	0	1	0	1	0	1	2	2	2	2
Intra Mode	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	–
V	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	–
H	2	2	2	2	2	1	0	1	0	1	0	1	0	1	0	1	0	–
Intra Mode	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**
V	1	0	1	0	1	0	1	0	1	0	1	2	2	2	2	2	2	2
H	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0
Intra Mode	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	–	–	–	–
V	2	2	1	0	1	0	1	0	1	0	1	0	1	0	–	–	–	–
H	0	0	1	0	1	0	1	0	1	0	1	0	1	0	–	–	–	–

transform types will be excessively evaluated, for each CU, comes with the cost of higher computation complexity. This can be an issue for real time implementation.

Table 3.2 – Pre-defined transform candidate subsets

Transform Set	Transform Candidates
0	DST-VII, DCT-VIII
1	DST-VII, DST-I
2	DST-VII, DCT-V

The JEM tried to incorporate all possible new coding tools able to provide a significant coding efficiency with respect to the HEVC performance. Indeed it presented up to 30% bit rate reduction. However, this was equipped with a huge complexity level that can no longer be neglected especially for industrial companies and real time implementations. Therefore, for the next phase of the new standard establishment, computational complexity and resource requirements are taken into consideration along with the bitrate gain. In fact, transform module was the main subject of discussion in the 12th and 13th JVET meetings. Statistical analysis showed that DCT-II, DST-VII and DCT-VIII are the most used in transform process and brought more than 90% of the coding gain [62]. Therefore, regarding complexity and coding efficiency, DCT-V and DSTI are no longer considered in VVC due to their high computation requirements especially that they do not have efficient implementation. As a

result, the new transform process in VVC is called Multiple Transform Selection including only three transform types.

3.2.5 Multiple Transform Selection in VVC

The concept of separable transforms competition has been widely investigated for HEVC [63, 64] which considers only DCT-II along with DST-VII for Intra luma blocks of size $4x4$ [65], and then integrated in the JEM software [66]. This latter enables five trigonometrical transform types including DCT-II, V and VIII, and DST-I and VII. This concept enables a significant increase of the coding efficiency estimated aound 3% of bitrate reduction [66]. However, this coding gain comes at the expense of both memory increase, used to store the coefficients of those transforms, and complexity overhead required to test the transform candidates at the encoder side.
The MTS concept in VVC defines only three transform types including DCT-II, VIII and DST-VII. As illustrated in Figure 3.2, the MTS concept selects, for Luma blocks of size lower than 64, the set of transforms that minimizes the rate distortion cost among five transform sets and the skip configuration. However, only DCT-II is considered for chroma components and Luma blocks of size 64. The MTS solution brings a significant coding gain of respectively 2% and 0.9% in AI and RA coding configurations [67] compared to the HEVC transform process.

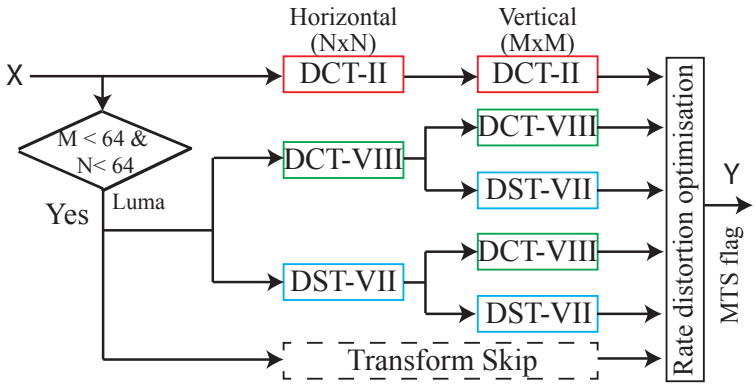

Figure 3.2 – The concept of 2D separable transforms selection in VVC. X is the input block of residuals, Y is the output transformed block and MTS flag is the index of the selected set of transforms

In order to keep the orthogonality of the transform matrix, the transform matrices are quantized more accurately than the transform matrices in HEVC. In order to control MTS scheme, separate enabling flags are specified at SPS level for intra and inter, respectively. When MTS is enabled at SPS, a CU level flag is signalled to indicate whether MTS is applied or not. Here, MTS is applied only for luma. The MTS CU level flag is signalled when the following conditions are satisfied: 1) Both width and height smaller than or equal to 32. 2)

CBF flag is equal to one.
If MTS CU flag is equal to zero, then DCT2 is applied in both directions. However, if MTS CU flag is equal to one, then two other flags are additionally signalled to indicate the transform type for the horizontal and vertical directions, respectively [59]. Transform and signalling mapping table as shown in Table 3.3.

Table 3.3 – Transform and signaling mapping table [59]

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/Inter	
–	–	–	Horizontal	Vertical
0	–	–	DCT-II	
1	0	0	DST-VII	DST-VII
1	0	1	DCT-VIII	DST-VII
1	1	0	DST-VII	DCT-VIII
1	1	1	DCT-VIII	DCT-VIII

When it comes to transform matrix precision, 8-bit primary transform cores are used. Therefore, all the transform cores used in HEVC are kept as the same, including 4-point DCT-2 and DST-7, 8-point, 16-point and 32-point DCT-2. Also, other transform cores including 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7 and DCT-8, use 8-bit primary transform cores.
To reduce the complexity of large size DST-7 and DCT-8, High frequency transform coefficients are zeroed out for the DST-7 and DCT-8 blocks with size (width or height, or both width and height) equal to 32. Only the coefficients within the 16x16 lower-frequency region are retained [59]. As in HEVC, the residual of a block can be coded with transform skip mode. To avoid the redundancy of syntax coding, the transform skip flag is not signalled when the CU level MTS_CU_flag is not equal to zero.

3.3 Related works

3.3.1 Hardware Implementation of HEVC transform

The DCT-II, as it is the common transform used in video coding standards AVC and HEVC, have been well studied and investigated in the literature. Several hardware implementations of the DCT-II have been proposed. Chang et al. [68] proposed a fast algorithm based on hardware-sharing architecture for 4x4;8x8;16x16, and 32x32 inverse core transforms. It presented a highly hardware efficient design with an effective cost by using the symmetrical characteristics of the elements in inverse core transform matrices. The proposed 1-D hardware sharing scheme required 115.7 Kgate counts to achieve an operational frequency of up to200 MHz. Shen et al. [69] presented a unified VLSI architecture for 4, 8, 16, and 32-point

integer IICTs. The architecture supported MPEG-2/4, H.264, AVS, VC-1, and HEVC video standards. A multiplierless technique was applied to the 4 and 8-point IDCTs. However, regular multipliers with hardware sharing were applied to the 16- and 32-point IICTs. To reduce hardware overheads, the memory was transposed using the SRAM module. The architecture supported 4 Kx2 K (4096x2048 pixels) at 30 fps real-time decoding at 191 MHz with 93 K gate counts and 18944-bit SRAM.

The work of Kammoun et al. [70] described a unified hardware architecture for 4x4; 8x8; 16x16, and 32x32 inverse 2D core transform IDCT in HEVC standard. It eliminated multiplications through addition and shift operations and was based on reusing some coefficients with most occurrences as 2, 4, 9, 18, 36, and 64 to further optimize area consumption. The operating frequency of the hardware design is about 130 MHz. Kalali et al. [71] proposed a hardware implementation of the 2D Inverse Core transform of the HEVC using High Level Synthesis (HLS)tools: Xilinx Vivado HLS, LegUp and MATLAB SimulinkHDL Coder. The proposed design used 4 different cores for each TU size and then all duplicated to perform the 2D approach which affected the occupied FPGA area. The maximum operational frequencies through these HLS toolswere respectively 208 Mhz, 143 Mhz and 110 Mhz. Sun et al. [72] interested in a reordered parallel-inserial-out (RPISO) scheme for the 2D IDCT core transform hardware implementation in order to reduce the required calculations by minimizing the redundant inputs of the butterfly structures. They also tried to reduce the memory buffer area by adopting a cyclic data mapping scheme and a pipelining schedule.

Shen *et al.* [73] presented a unified VLSI architecture for 4, 8, 16, and 32 point IICT. Regular multipliers and hardware sharing (recursion) are applied to the 16- and 32-point IICT. To reduce the required hardware resources, the intermediate 1D results are transposed using the SRAM module. Meher *et al.* [65] presented an efficient and reusable architectures for DCT-II implementation supporting different sizes using constant matrix multiplications. This architecture can be pruned to reduce the complexity of implementation substantially for both folded and full-parallel 2D DCT-II implementations with only a marginal effect on the coding performance (from 0.8% to 1% BDR loss in coding efficiency when both DCT-II and inverse DCT-II are pruned). Chen *et al.* [74] proposed a 2D hardware implementation of the HEVC DCT transform. The reconfigurable architecture supports all block sizes from $4x4$ to $32x32$. To reduce the logic utilization, this implementation benefits from several hardware resources, such as DSP blocks, multipliers and memory blocks. The proposed architecture has been synthesized for various FPGA platforms showing that the design enables to sustain 4Kp30 video encoding with reduced hardware cost. Ahmed *et al.* [75] proposed a dynamic N-point DCT-II hardware implementation for HEVC inverse transform of sizes $4x4$, $8x8$, $16x16$ and $32x32$. The hardware architecture is partially folded in order to save the area and improve the speed up of the design. This architecture reaches a maximum operational frequency of 150 MHz which enables to support real time processing of 1080p30 video.

3.3.2 Hardware Implementation of MTS

Recently, several works [76, 77, 78, 79, 80] have investigated the hardware implementation of the initial version of MTS including the five transform types. Mert *et al.* [76] proposed a 2D implementation including all transform types for 4x4 and 8x8 sizes using adders and shifts instead of multiplications. Two hardware methods have been provided. The first ones uses separate datapaths and the second method considers two reconfigurable datapaths for all 1D transforms. Although this work presented a 2D hardware implementation of all transform types, it only supports 4x4 and 8x8 block sizes, while the transform of larger block sizes (16x16 and 32x32) are more complex and would require more hardware resources. In [77], Garrido *et al.* proposed a pipelined 1D hardware implementation for all block sizes from 4x4 to 32x32. The design has been synthesized for different FPGA chips using multiple ROM blocks to store the matrices of transform coefficients. The synthesis results show that the design can support 2K and 4K video processing with low hardware resources. However, this solution only considers 1D design, while the transform process consists in 2D operations which could normally be more complex. Moreover, this design does not consider asymmetric block size combinations. This work has been then extended in [78] to support 2D design using Dual port RAMs for the transpose memory. They proposed to pipeline the 2D process placing two separate 1D processors in parallel for horizontal and vertical transforms. In [79], Kammoun *et al.* presented a multiplierless implementation of the MTS 4-point transform module. This has been extended to 2D hardware implementation of all block sizes (including rectangular ones), with using the IP Cores multipliers [19] to leverage the DSP blocks of the *Arria 10* platform [80]. This solution supports all transform types and enables a 2D transform process with efficient pipeline architecture. However, it requires high logic utilization compared to solutions proposed in [76, 77].

3.3.3 Approximations of Transforms

Several contributions have been proposed by the JVET to overcome the complexity/resource allocations issues of the MTS [81, 82, 83, 84]. These solutions proposed to reduce the computational complexity and required number of multiplications per pixel required to process the DST-VII and DCT-VIII. In fact, approximation of transform module is not new in the literature, and it has been widely investigated for DCT-II [85, 86, 87, 88, 89, 90]. Jridi *et al* [85] presented a generalized approximation algorithm for the 8-point DCT-II. It relies on factorizing the DCT matrix into even-odd decomposition and then replacing the odd part with the even one to further reduce the operation count. The approximate 8-point DCT architecture is used to generate a reconfigurable implementation of larger block sizes based on the same principle. However, the rough approximation of the 8-point core and using it for larger sizes resulted in more than 5% coding loss in terms of rate distortion performance

(BDR). Renda *et al* [86] proposed to approximate 8-point DCT-II by an exact low-complexity factorization of the 8-point DCT [91] to be used as core module in the generalized algorithm proposed in [85]. The $8x8$ matrix multiplication is reduced to only 5 multiplications and 29 additions. The 8-point scheme is used then to generate larger transform sizes of 16 and 32. This enables a better coding performance compared to the work in [85], but it still achieves a poor rate distortion performance with an average of 4% bitrate loss. Work in [87] proposed a three processing levels to approximate DCT-II. This approach consists in replacing all multiplication operations with shifts and additions, high frequency coefficient filtering and then using inexact additions to compute the DCT-II transform. Chen *et al.* [88] presented an approximate DCT-II supporting block sizes from 8 to 64. This solution enables a factorizable structure for both even and approximate odd parts to further reduce its implementation complexity while preserving similar rate distortion coding performance compared to the original. Jridi *et al.* [89] proposed an approximation method of the HEVC-DCT-II that leverages the even-odd butterfly architecture. The matrix coefficients of the even part are replaced by two coefficient values requiring only shift operations to perform the multiplication. The second step of the approximation replaces odd part by even one in order to reduce the computational complexity of the design, especially benefiting from the recursion property. As a result, all multiplications are removed and larger block sizes implementations are optimized. Work in [90] proposed to approximate the HEVC DCT-II transform design by using a similar approach than the one developed in [89], while the difference lay in odd part coefficients, which are approximated according to their distance with respect to the extremum two values (max and min) of the original ones.

3.4 Description of the target platform Arria 10 SoPC

For the work developed in this thesis, following the great progress that embedded platforms have witnessed, we have used one of the new created advanced SoC FPGA platforms, Arria 10, in the proposed hardware implementation for transform coding tools of VVC standards.

3.4.0.1 Intel-Altera Soc FPGA

The 10^{th} generation of Intel-Altera SoPC, which includes Stratix 10, Arria 10 and Max 10, is almost two times more efficient than its predecessors, mainly the 5th generation including Cyclone V, Arria V and Stratix V. (see Figure 3.3).

New software and hardware features have been introduced to make them more adequate for applications requiring high memory and computation resources, such as high resolution video processing. They mainly impacted technology (level of transistor integration), operational frequency, bandwidth, power dissipation etc [92].

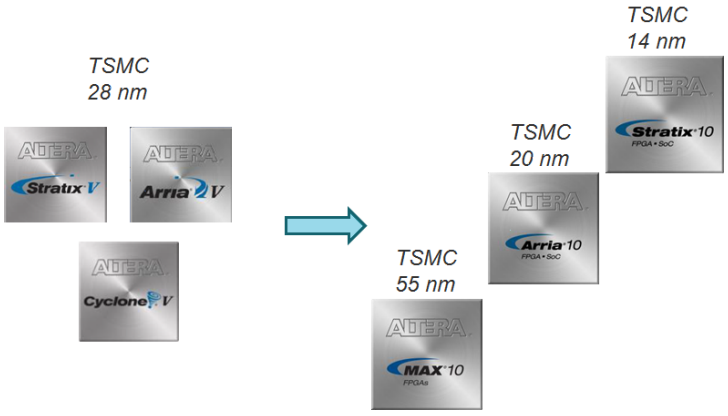

Figure 3.3 – 5th and 10^{th} Altera SoC generations

Table 3.4 shows a comparison of two product features from 5th and 10^{th} generation SoPC devices, Arria 5 and Arria 10, respectively.

Table 3.4 – Comparaison betwwen Arria V and Arria 10 features

	Arria V	Arria 10	
Technology	28 nm	20 nm	
Logic	370-450 K LE	160-660 K LE	1.5x
Multipliers	2312	3376	1.5x
Frequency (FPGA)	300 Mhz+	500 Mhz+	1.6x
Memory interfaces	1333 Mbps	2666 Mbps	2x
Power dissipation	1.0	0.6	40 lower

Within the great progress witnessed for embedded platforms, new devices are meant to provide the desirable performance under very strict constraints such as size, logic resource availability (memory, logic...), cost, time to market etc. Figure 3.4 describes the position of the different Arria SoC FPGAs with respect to performance-consumption ratio.

It can be noticed that the fifth generation of SoPCs are the least energy consumers since they are not adapted to very complex applications. The more applications require high performance the more the logic consumption increases as well as the cost and the time to market.

Arria 10 is included in the Middle End (ME) products which is able to provide the desired performance while keeping a minimized logic resource consumption and an acceptable cost. Combined with its development kit, Arria 10 presents a reconfigurable hardware/software platform that guarantees a faster path to commercialization. It can be a good fit for high resolution video processing, especially hardware accelerations of high computational complexity applications related to the new potential video coding standard VVC. Figure 3.5 shows an overview of the Arria 10 SoPC development kit.

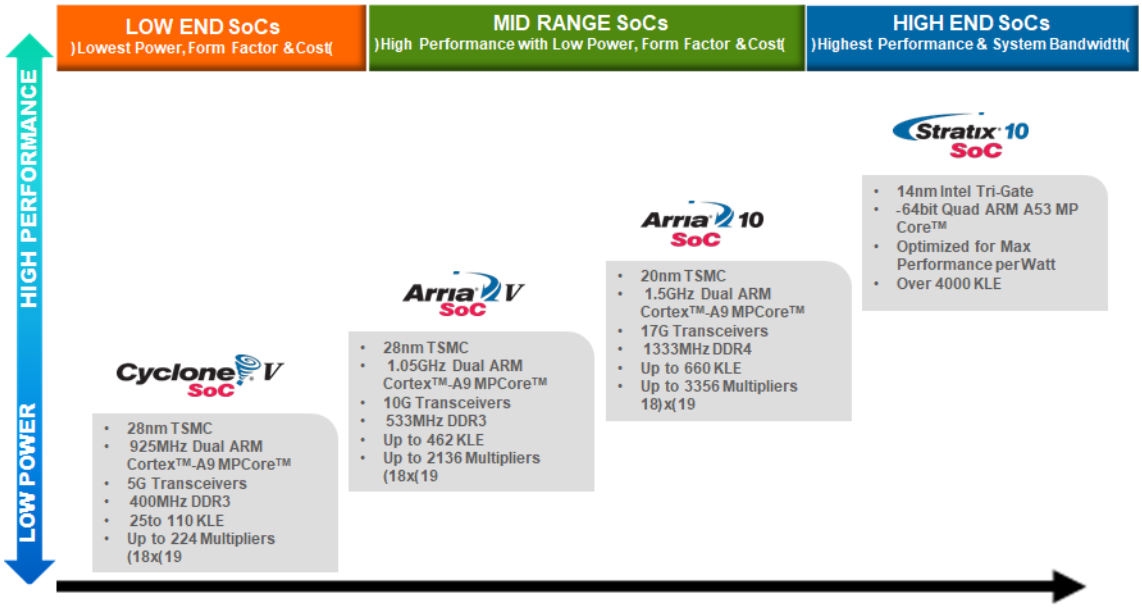

Figure 3.4 – Evaluation scale of different Altera SoPC families

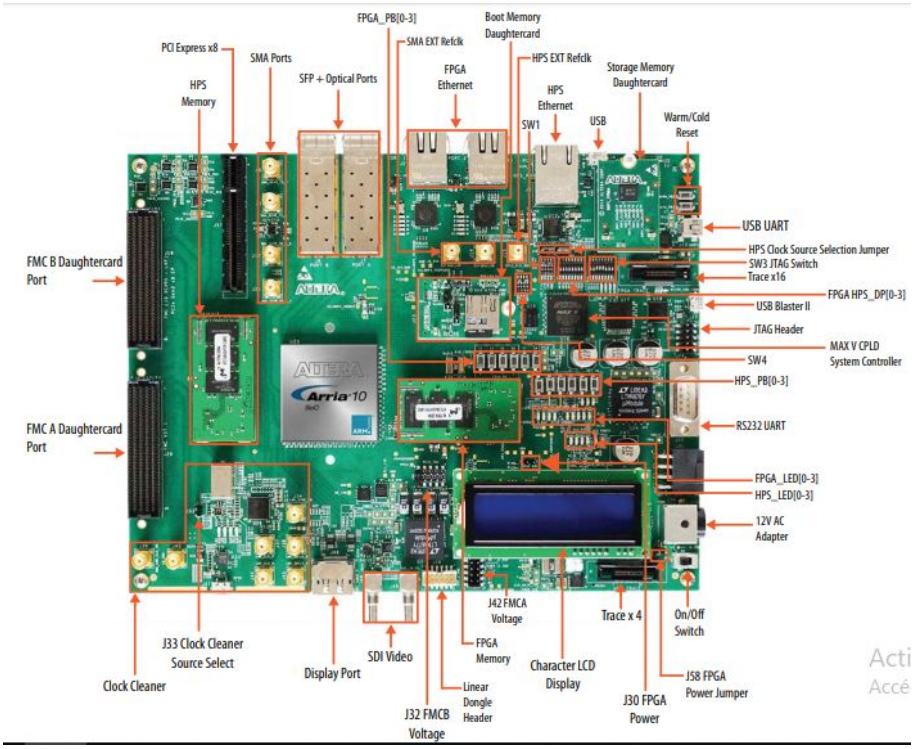

Figure 3.5 – Arria 10 SoC development Kit

Indeed, Arria 10 SoC family is divided into three other families: GX, TX and SX. We will focus on the SX range since it is the only one that integrates software processors with FPGA

blocks for software/hardware co design applications. Table 3.5 presents the different tools and characteristics of Arria 10 SX. The most important features and optimizations offered

Table 3.5 – Important tools and characteristics of Arria 10 SX SoPC

Device	Arria10 (SX 160- 660)
Technology	Dual Core Cortex A9
Frequency	1.5 Ghz
Trancievers	12-48 (x2)
Max data rate	17,4 Gbps
Memories	DDR4 / DDR3 / DDR2/ LPDDR3 / LPDDR2 / RLDRAM 3/ RLDRAMII / QDR IV / QDR II+ / QDR II SRAM
Memory interface	2666 Mbps
I/O Pins	288-696
Logic	160-660 K LE
ALMs (Adaptive logic modules)	61,510-250,540
Registers	246,040-1,002,160
DSP	156-1688
Multipliers	3,376
Performance (FPGA)	500 MHz+
Power dissipation	40 to 60 % lower
Ethernet	2x 10/100/1000 SGMII Ethernet ports

by the Arria 10 platform are listed as follows:

- Dual Core Processor ARM Cortex A9: two ARM processors of 20nm technology with 1.5 Ghz clock frequency each. They are equipped with a relatively large and shared cache memory.
- Large number of DSP blocks (up to 1687) and multipliers (up to 3376). These blocks can perform several constant multiplications between proper constant values as inputs. With a computing capacity of up to 1.5 G FLOPS, they are dedicated to intensive computational applications.
- A low power dissipation with up to 40% lower than previous generation devices. Thanks to Smart Voltage ID tool, Arria 10 is able to run at lower than nominal Voltage Common Collector (Vcc) while retaining same performance level reducing static and dynamic power. On the other hand, low power transistors can be used for noncritical logic paths. In fact, at the routing level, it is possible to accelerate the critical paths requiring speed by reducing the power supply of the other paths as shown in Figure 3.6 (a). Moreover, additional software optimizations can be applied with Quartus configuration to provide high speed where needed and reduce it elsewhere (b, c)
- HPS/FPGA: Arria 10 enables a flexible interaction between processors and FPGA block. This is ensured by the different interfaces connecting the processors with not only the

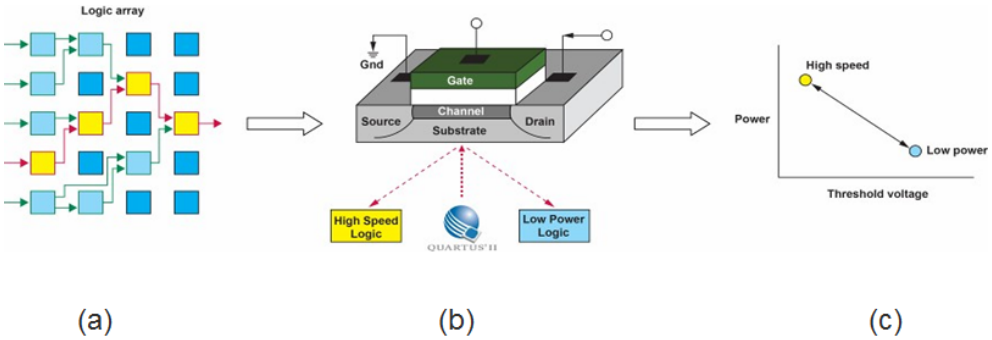

Figure 3.6 – Routing optimizations for Arria 10

remaining peripherals but also with the FPGA block. In addition, a 64-bit Axi bus and a connector with DDR interfaces are available for high-speed communication. Figure 3.7 presents a simplified model that describes the connections between the software part (processors) and the FPGA blocks in Arria 10.

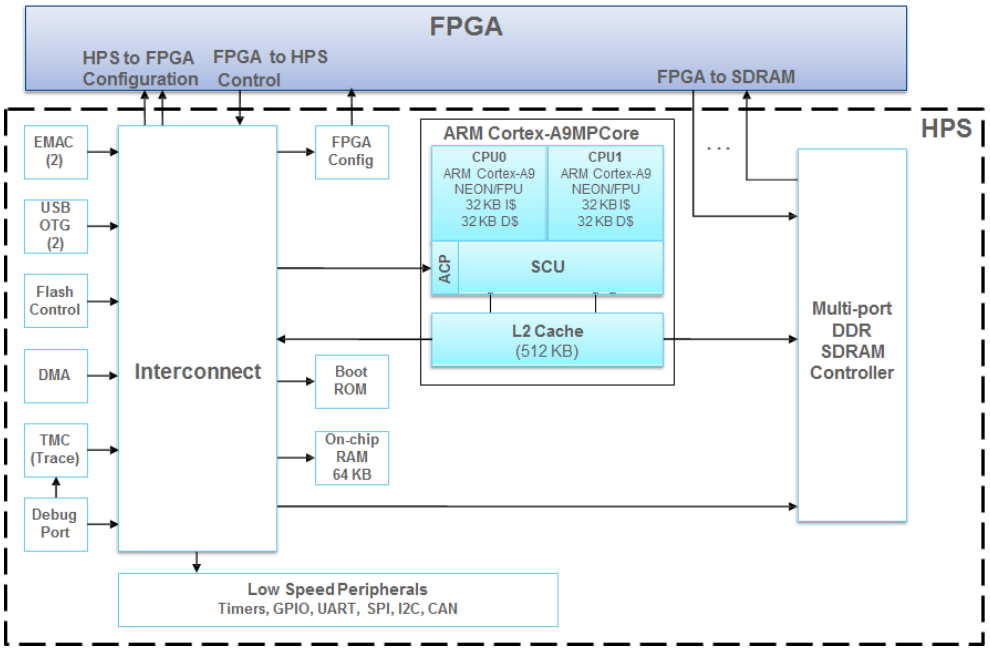

Figure 3.7 – Different processor and FPGA interactions in Arria 10 SoPC

This flexibility is further enhanced during all designing steps of applications or circuits. There will always be a mutual coherence between the software and hardware development (design, simulation, debugging, control ...) provided by the associated software that is delivered by Intel with the user guides such as "ARM DS -5 Toolkit "and" Quartus Prime ".

3.5 Conclusion

This chapter presented a background on the discrete cosine transform DCT-II used in HEVC showing its important properties (symmetry, orthogonality, recursion...) in order to reduce its computational complexity. For the future video coding standard, other transform types of DCT/DST family are put in competition along with DCT-II to select the best choice that provides better coding efficiency according to the RD cost computation. A description of the multiple transform concept throughout the standardization process, known as AMT in an early stage and currently named MTS, is also described. As mentioned earlier, hardware implementations are meant to provide some performance accelerations and try to compensate the substantial increase of computational complexity but under the constraints of their resources availability (memory, logic...). In this context, different related works in literature focusing on hardware implementation of the transform module are listed and described. Finally, the end of this chapter presents an overview of the target FPGA platform Arria 10 SoC used in the main contributions of this thesis (detailed in the next chapters) to provide an efficient hardware implementation for the transform coding tools in VVC standard.

Chapter 4

Hardware Design and Implementation of Adaptive Multiple Transforms for the VVC Standard

This chapter details one of the main contributions of the thesis as it proposes the first 2D hardware implementation in the literature of the Adaptive Multiple Transform including 5 transform types and supporting all block sizes from 4x4 to 32x32 considering all possible block size combination (symmetric and asymmetric).

Contents

4.1 Introduction

In the early stage of VVC standardization process (JEM reference), transform module has included a total of 5 transform types. This has introduced a significant complexity level especially for real time implementations targeting high frame rate performance considering the logic resource limitation and constraints. In this chapter, we propose an efficient 2D hardware implementation of the adaptive multiple transform. Section 4.2 presents a multiplierless 4-point implementation of all transform types included in AMT. Next, this solution is extended and optimized to propose the first 2D implementation of AMT on FPGA SoPC. It includes 5 transform types and supports all block sizes from 4x4 to 32x32 considering all possible block size combination (symmetric and asymmetric). In Section 4.3, a brief description of the FPGA target device is given focusing on its beneficial features for this work, followed by the detailed hardware implementation approaches for the 1D and 2D architecture designs. The experimental and synthesis results of 1D and 2D implementations are presented and discussed in Section 6.6. A comparison with other proposed works is also investigated in this section. Finally, Section 4.5 concludes this chapter.

4.2 Multiplierless 4-point hardware architecture of AMT transform module

4.2.1 Hardware architectures of the AMT transform types

The JEM codec is based on transform basis functions to calculate multiplication matrices coefficients. Hence as a first step, we extracted these matrices to be used as an input for the hardware implementation. In the following, we will present the proposed decomposition algorithms of each transform type for size 4x4 with their associated architectures benefiting from the matrices correlation and symmetry. The general equation for all transform types is:

$$Y_{int} = H^4 \cdot B^T, \tag{4.1}$$

where B [B0...B3] represents the residual vector which is the input of the 1D-transform unit and Y_{int} [Y0...Y3] is the corresponding output one.

4.2.1.1 DCT-II transform

The only difference, with respect to HEVC DCT-II transform, consists in the coefficients values while the symmetric property of the matrix remains unchanged. Therefore, proceeding with the same decomposition method proposed in [93], the size 4 of DCT-II matrix will be computed as follows: In equation (4.1), H^4 (C_2^4 in this case) can be described as:

$$C_2^4 = Pr^4 \cdot C1_2^4 \cdot P1^4, \tag{4.2}$$

where $C_2^4 = \begin{pmatrix} 256 & 256 & 256 & 256 \\ 334 & 139 & -139 & -334 \\ 256 & -256 & -256 & 256 \\ 139 & -334 & 334 & -139 \end{pmatrix}$

$P1^4 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$ and $Pr^4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ in order to obtain the bloc-diagonal matrix as follows: $C1_2^4 = \begin{pmatrix} 256 & 256 & 0 & 0 \\ 256 & -256 & 0 & 0 \\ 0 & 0 & -139 & -334 \\ 0 & 0 & 334 & -139 \end{pmatrix} =$

$\begin{pmatrix} 256 & 256 \\ 256 & -256 \end{pmatrix} \oplus \begin{pmatrix} -139 & -334 \\ 334 & -139 \end{pmatrix}$,

where $\oplus$ is direct sum operator.

$C11_2^4 = \begin{pmatrix} 256 & 256 \\ 256 & -256 \end{pmatrix}$ and $C22_2^4 = \begin{pmatrix} -139 & -334 \\ 334 & -139 \end{pmatrix}$

$C11_2^4$ requires 2 additions and 2 shifts operations. However

$C22_2^4 = 128 \begin{pmatrix} -1 & -3 \\ 3 & -1 \end{pmatrix} + \begin{pmatrix} -11 & 50 \\ -50 & -11 \end{pmatrix}$ and of course with replacing these coefficients by their equivalences it will require then 16 additions and 14 shifts. Figure 4.1 illustrates the 4 point DCT-II hardware architecture.

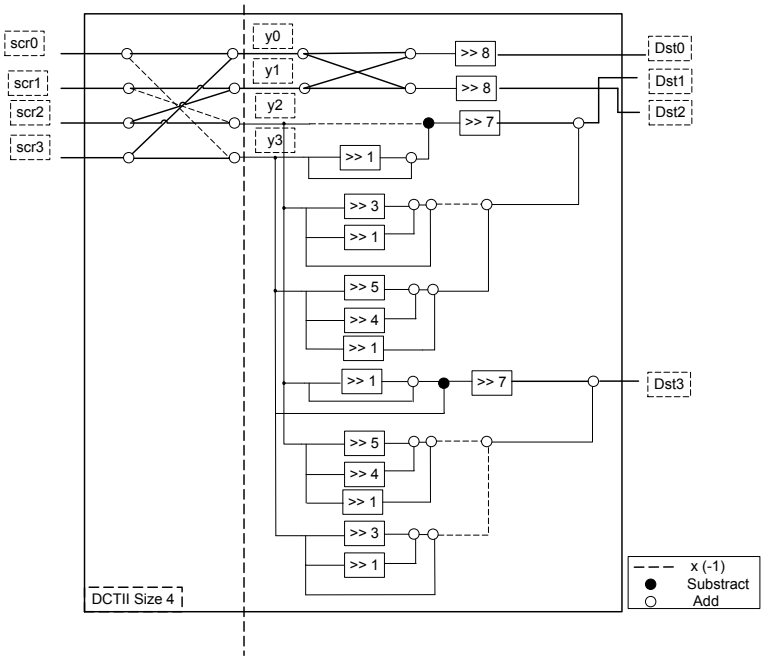

Figure 4.1 – Proposed hardware architecture for DCT-II size 4

4.2.1.2 DST-I transform

The previous decomposition principle can be applied since the DST-I matrix coefficients present the same symmetric properties than DCT-II.

$$S_1^4 = \begin{pmatrix} 190 & 308 & 308 & 190 \\ 308 & 190 & -190 & -308 \\ 308 & -190 & -190 & 308 \\ 190 & -308 & 308 & -190 \end{pmatrix}$$

Through the same sparse matrices we obtain the following bloc-diagonal matrix:

$$S1_1^4 = \begin{pmatrix} 190 & 308 & 0 & 0 \\ 308 & -190 & 0 & 0 \\ 0 & 0 & -190 & -308 \\ 0 & 0 & 308 & -190 \end{pmatrix}$$

We can notice that coefficients are similar in both submatrices, that's why it would be the same block architecture with appropriate signs.

$$\begin{pmatrix} 190 & 308 \\ 308 & -190 \end{pmatrix} = 190 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} + 118 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

This equation requires 12 additions and 12 shifts as number of operations . Figure 4.2 presents the proposed architecture of DST-I.

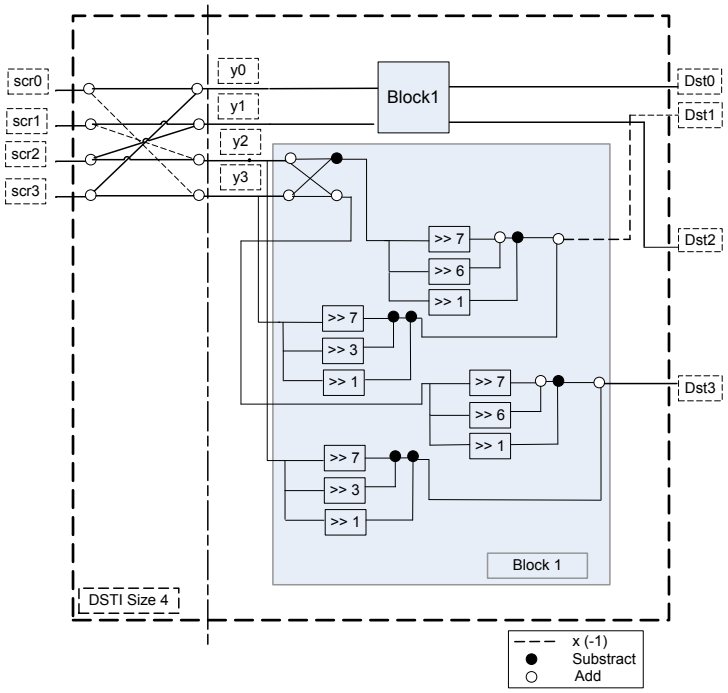

Figure 4.2 – Proposed hardware architecture for DST-I size 4

4.2.1.3 DST-VII transform

$$S_7^4 = \begin{pmatrix} 117 & 219 & 296 & 336 \\ 296 & 296 & 0 & -296 \\ 336 & -117 & -296 & 219 \\ 219 & -336 & 296 & -117 \end{pmatrix}$$

Noticing that the 1^{st}, 3^{rd} and 4^{th} rows have the same coefficients but in different order and signs, the idea consists in designing one block that can be used three times in parallel with the appropriate coefficients order and signs. This equation is an example of the decomposition adopted for the first output:

$$\begin{aligned} Y0 = 128\,[B0 + 2(B1 + B2 + B3) + B3] \\ + 8\,[B0 - 5\,(B1 - B2) - 6\,B3] \\ - 3\,[B0 - B1]\,. \end{aligned} \tag{4.3}$$

The DST-VII transform architecture presented in Figure 4.3 details how additions and shift operations are used to compute the DST-VII transform while eliminating all multiplications.

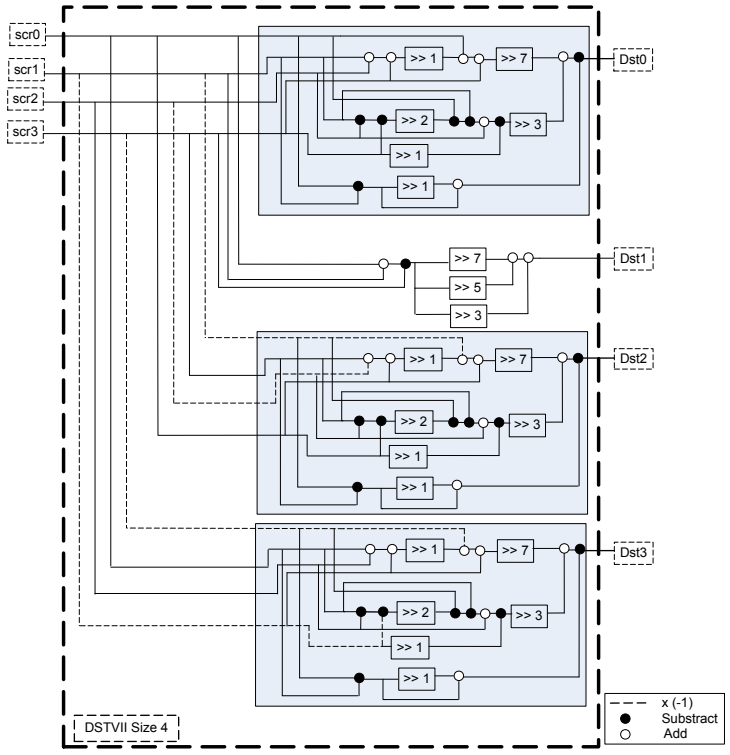

Figure 4.3 – Proposed hardware architecture for DST-VII size 4

4.2.1.4 DCT-VIII transform

$$C_8^4 = \begin{pmatrix} 336 & 296 & 219 & 117 \\ 296 & 0 & -296 & -296 \\ 219 & -296 & -117 & 336 \\ 117 & -296 & 336 & -219 \end{pmatrix}$$

Compared to the DST-VII matrix, DCT-VIII one has the same coefficients but in inverse order for each row. Then, with only inversing the inputs order and assigning the appropriate coefficients signs we can easily benefit from DST-VII (size 4) architecture to implement the DCT-VIII transform type with no additional computational complexity.

4.2.1.5 DCT-V transform

$$C_5^4 = \begin{pmatrix} 194 & 274 & 274 & 274 \\ 274 & 241 & -86 & -349 \\ 274 & -86 & -349 & 241 \\ 274 & -349 & 241 & -86 \end{pmatrix}$$

Coefficients redundancies in C_5^4 allowed us to propose an architecture, presented in Figure 4.4. This architecture consists in one block charged to provide the 1^{st} output. This block requires

7 additions and 6 shifts. A second block (*block2-DCT5*) is used three times in parallel modifying only the inputs order to obtain the other outputs using 14 additions and 7 shifts.

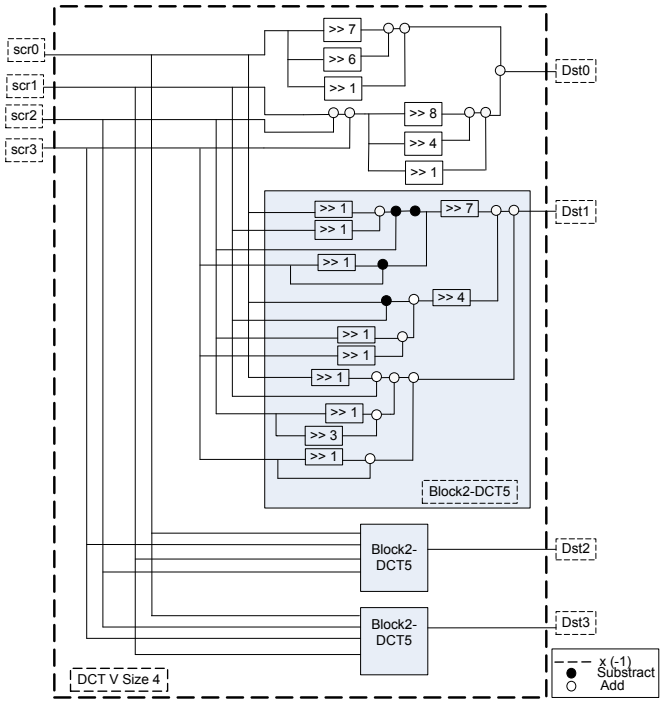

Figure 4.4 – Proposed hardware architecture for DCT-V size 4

4.2.2 Experimental and synthesis results

The five transform types architectures were specified in VHDL language using *Modelsim* software tool.

4.2.2.1 Simulation results

4.2.2.2 State-machine based architecture

A unified circuit that encompasses the different types for size 4 is designed. In DST-VII (Figure 4.3), DCT-VIII- and DCT-V (Fig. 4.4) architectures, there was a component that has been used three times at once. Taking advantage of this property, we noticed that introducing a state-machine managing those blocks operation sequentially, according to a definite control unit, would provide a considerable computational complexity reduction compared to the first proposed implementation method. Table 4.1 shows the performance of both solutions (no state-machine 1^{st} method and with state-machine 2^{nd} method) in terms of required operations (adders and shifts) and clock cycles. We can notice that the number of operations

is mainly related to the values of the coefficients matrices. It also gives us a clear idea how using state machines (2^{nd}) enables about 45% reduction in computational complexity at the expense of higher clock cycles.

Table 4.1 – Performance of the 1^{st} (no state machine) and 2^{nd} (with state machine) implementation methods

	Adders		Shifts		Clock cycles	
	1^{st} meth.	2^{nd} meth.	1^{st}	2^{nd}	1^{st}	2^{nd}
DCT-II	24	24	16	16	4	4
DST-I	28	16	24	12	5	7
DST-VII	46	18	21	9	5	9
DCT-VIII	0	0	0	0	6	10
DCT-V	49	21	27	13	5	9
Total	147	79	88	50	-	-
Reduction	–	46 %	–	44%	–	–

Table 4.2 presents a comparison between the original design using multiplication operations and the multiplierless proposed architectures under Stratix-III EP3SL340F1760C4 FPGA device using the software tool *Quartus II 9.0*. The synthesis results show that the proposed methods offer a wide logic elements optimization. The low reserved registers number of the original solution is due to the lack of intermediate computations by shift and addition operations. On the other hand, the decrease in the number of operations (Table 4.1, 2^{nd} method) was reflected automatically to the hardware cost due to the reuse of the hardware resources offered through state machines. Table 4.2 shows also that the two proposed designs have reached respectively 318 MHz and 285 MHz as maximum processing frequencies.

Table 4.2 – Synthesis results of the proposed architectures

	original	**1^{st} method**	**2^{nd} method**
Pins	303 (41%)	295 (40%)	295 (40%)
ALUTS	10116 (5%)	6842 (3%)	3802 (1%)
Registers	994 (< 1%)	3603 (1%)	2219 (< 1%)
Frequence	356 MHz	318 MHz	285 MHz

4.2.2.3 Discussion

As the design architecture (1^{st} method) provides parallel output generation, we can reach further optimization by adding the pipelining operation. This latter is more interesting when it concerns computing of multiple rows of size 4 (4x4 blocks size as an example). Table 4.5 shows the clock cycles required for computing 1D 4x4 blocks size of each transform type. Of course, the more rows are computed the more interesting pipeline becomes, especially for

future works (other transform sizes) as the AMT adopts asymmetric block sizes (4x8 , 4x16, 4x32, 4x64..).

Table 4.3 – Clock cycles of pipelined 4x4 blocks

	DCT-II	DST-I	DST-VII	DCT-VIII	DCT-V
Clock cycles 1st	11	12	12	13	12
Clock cycles 2nd	11	22	30	30	30

The required frequency to compute 4K videos at 30 frames per second (fps) is 202 Mhz (13x3840x2160x30 / 4x4). Therefore, the first proposed design, reaching an operational frequency up to 318 Mhz, can easily support 4K coding assuming only 4x4 block sizes. With involving other larger sizes and more computational complexity we may have different results. On the other hand, although the state machine architecture provided computational and area optimization, it can't really benefit from the pipelining operation because of data dependencies and lack of total parallel output generation. As an example, DST-I requires 22 clock cycles to compute 4x4 block instead of 28 (7x4) without pipeline. This can support 2K resolution video processing at 60 fps.

4.3 The proposed hardware implementation of 2D AMT

4.3.1 1D-AMT Hardware implementation

4.3.1.1 4-point AMT implementation

- Logic Model

The 4-point 1D-AMT design is summarized in Table 4.4. A *start* positive pulse launches the operation while the transform type is selected by the *selection* input. The input data

Table 4.4 – 4-point 1D interface description design

Signal	I/O	Bits	Description
clk	I	1	Clock system
reset	I	1	Active low
start	I	1	Positive pulse
selection	I	3	Transform types: 0: DCT-II, 1:DST-I, 2: DST-VII, 3: DCT-VIII, 4:DCT-V
src0 .. src3	I	64	Input vector, 4 16 bit inputs
dst0 .. dst3	O	104	Output vector, 4 26 bit outputs
done	O	1	Qualifies output, active high

(vector of residual blocks as the difference between the input image and the predicted one) is provided at a column basis with the start pulse. Four 16-bit inputs must be provided

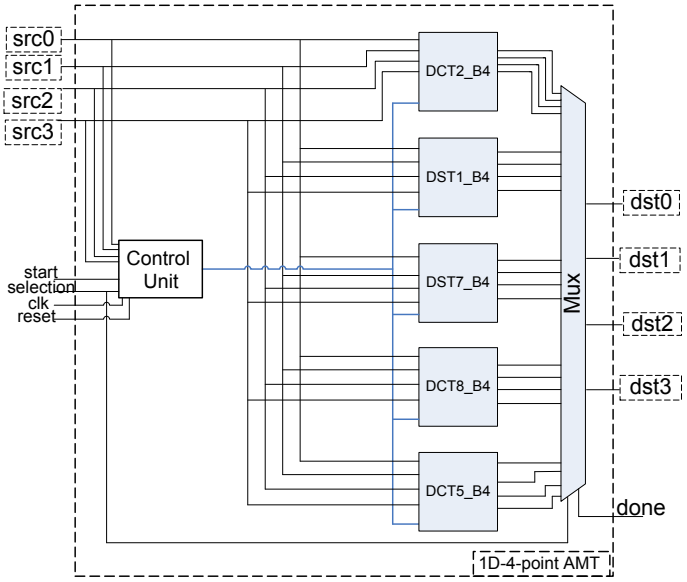

Figure 4.5 – Proposed 1D 4-point architecture design

simultaneously. After the transform process, the output values are assigned to *dst0 ..dst3* as shown in Figure 4.5. Finally, the *done* signal indicates that the outputs are available.

- Proposed 4-point AMT architecture

For the DCT-II and DST-I transform types, some preliminary decompositions using efficient butterfly structure are applied in order to reduce the computational complexity of their design as shown in Figure 4.6 and Figure 4.7.

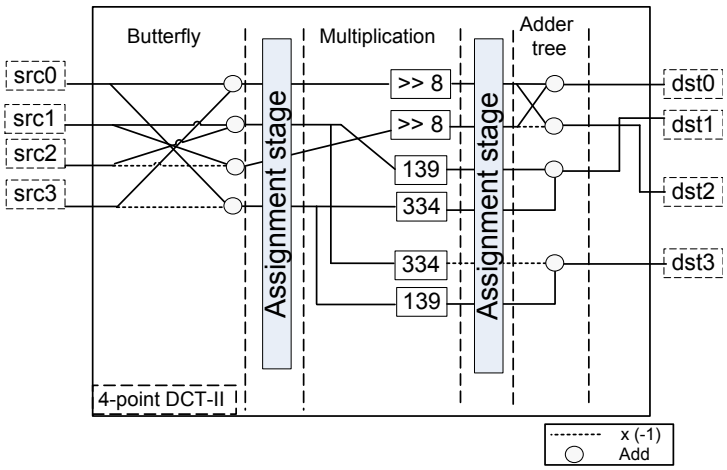

Figure 4.6 – Proposed 1D 4-point DCT-II architecture (dotted line refers to inverse sign value and add to addition operation)

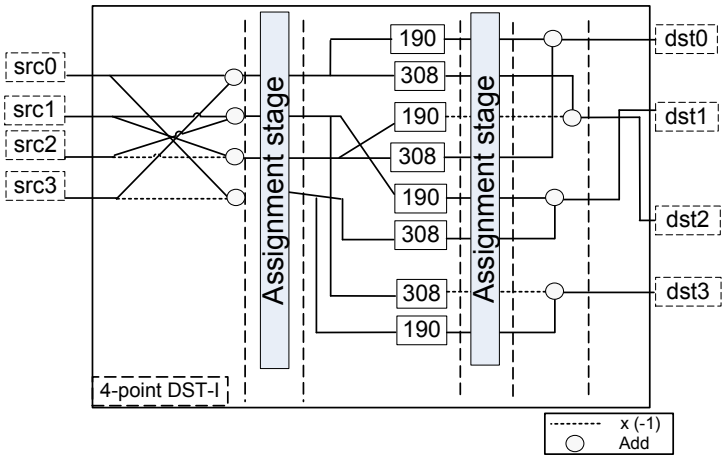

Figure 4.7 – Proposed 1D 4-point DST-I architecture

After the butterfly stage, all multiplication operations required are performed in parallel at once using the LPM multipliers [19] of the target platform. The constant values mentioned in Figure 4.6 - 4.10 refer to the coefficients of the transform matrix involved in the AMT.

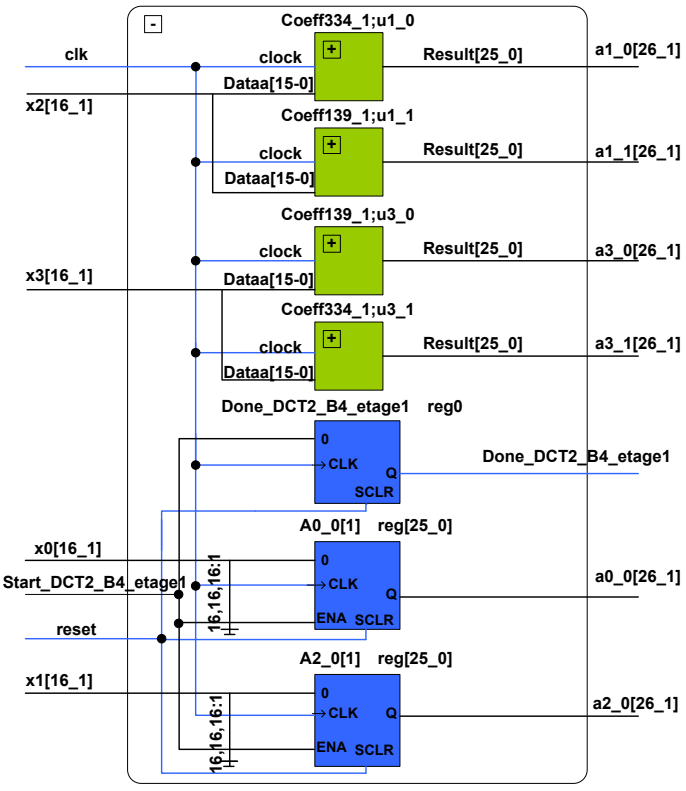

Figure 4.8 – RTL scheme of the DCT-II multiplication stage; green for LPMs and blue for Shift operators

Figure 4.8 presents a RTL scheme of the DCT-II (Figure 4.6) multiplication stage. LPM instances (green blocks) and shift gates (blue blocks) with appropriate coefficients are placed in parallel to perform multiplication operations. Finally, an adder tree is applied to provide the 1D four outputs. The dotted vertical line separating two stages is equivalent to a clock cycle in the processing operation. Butterfly decomposition structures can not be applied for the other transform types. Thus, they are computed as forward matrices multiplications. Figure 4.9 and Figure 4.10 illustrate the proposed architectures for DST-VII and DCT-V, respectively.

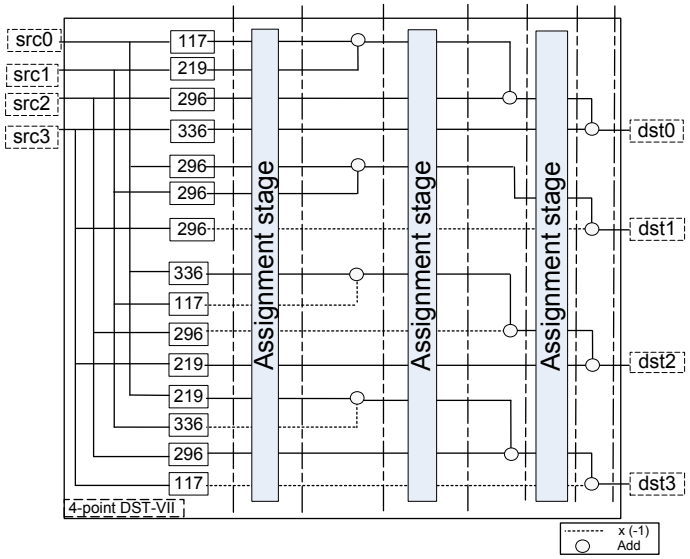

Figure 4.9 – Proposed 1D 4-point DST-VII architecture

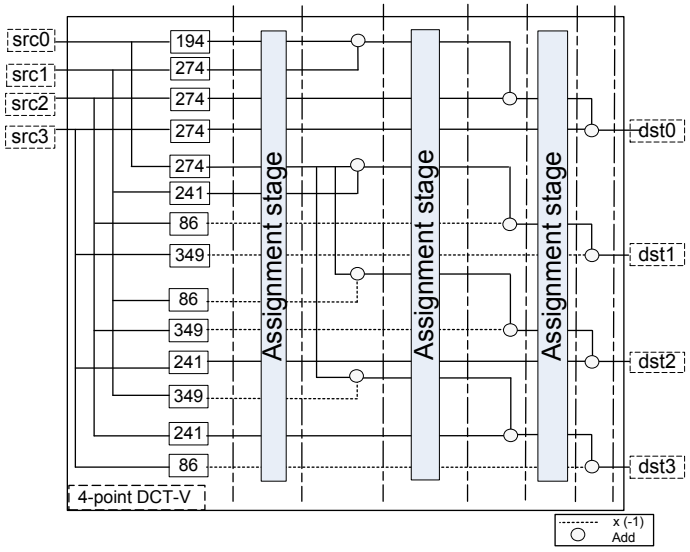

Figure 4.10 – Proposed 1D 4-point DCT-V architecture

Internal LPMs are used for all required multiplications in parallel. Then, three adder tree stages are placed successively in order to obtain the final outputs.
Compared to the DST-VII matrix, DCT-VIII one has the same coefficients but in reverse order for each row. Therefore, we only inverse the inputs order and assign the appropriate coefficients signs to easily benefit from DST-VII architecture, illustrated in Figure 4.9, to implement the DCT-VIII transform type.

- Pipelined architecture design

In order to increase the design performance, the different architectures have been pipelined. The assignment stage components, as shown in Figures 4.6, 4.7, 4.9 and 4.10, are added after multiplication stage and between two successive adder tree stages .They are based on registers and have basically two roles: storing the current results and transferring the appropriate data and intermediate signals to the next stage. These components are responsible for the pipeline operation avoiding data conflicts or loss which may occur in the next clock cycles as inputs are refreshing. Figure 4.11 shows a timeline presenting a 4x4 block pipeline processing.

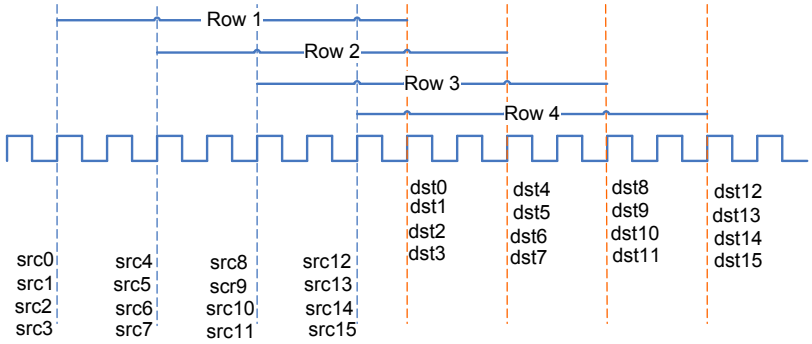

Figure 4.11 – Timeline for 4x4 block pipeline processing

Each assignment stage introduces one additional cycle to the latency providing the first four outputs. From that, within every two cycles, another four outputs are provided. Table 4.5 gives the latency (L) in clock cycles required to compute the first outputs of each transform type. Of course, computing more rows in parallel would increase the performance enabled by the pipeline. In general, we can calculate the clock cycles (C_{Cycles}) required to compute M inputs rows by equation (4.4).

$$C_{Cycles} = L + (M - 1)\,\Delta. \tag{4.4}$$

where L is the number of cycles required to provide the first outputs (latency) and Δ is the pipeline level which refers to the number of cycles required between two outputs. In the example illustrated in Figure 4.10, $N = M = 4$, $L = 7$, $\Delta = 2$ and $C_{Cycles} = 13$.

Table 4.5 – Latency (L) in clock cycles required to provide the first outputs for 4-point transforms

	DCT-II	DST-I	DST-VII	DCT-VIII	DCT-V
Latency (L)	5	5	7	7	7

4.3.1.2 N-point AMT implementation

For DCT-II and DST-I, as their operations are recursive, an N point 1D transform can be performed by applying two $N/2$-point 1D transforms with additional preprocessing. For the DST-I, the applied $N/2$-point is of type DST-VII as illustrated in Figure 4.12. DCT-V and DST-VII do not have the recursivity property. Therefore, they are implemented with matrices multiplications using the LPM multipliers IP Cores as for the 4-point case. DCT-VIII transform type is always implemented using the DST-VII with appropriate changes of inputs order and signs.

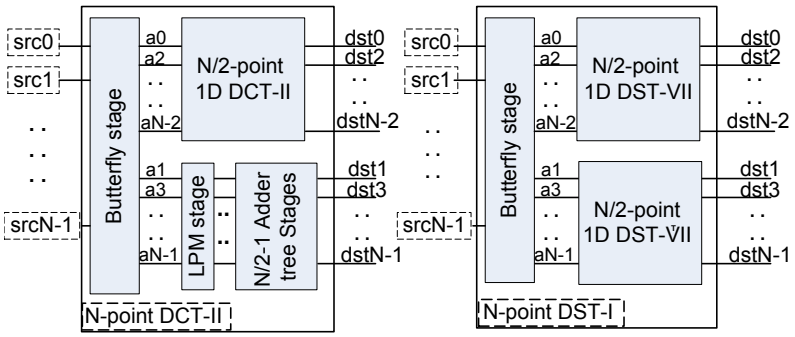

Figure 4.12 – Architectures of N-point DCT-II and DST-I

It is worth noting that for the 32-point implementation, pipeline is not adopted. This is justified by the fact that using the registers to ensure the pipeline stages for all the 32-point transform types together would require more logic utilization than the available one in the target platform. Instead, in order to preserve the clock cycles for 1D and 2D processes, adder trees were modified to operate two addition operations in one cycle. As a result, clock cycles required to provide 32-point outputs are reduced by half.

To summarize, the clock cycles required to implement one 1D outputs column considering the worst case type are 7, 15, 31 and 15 cycles for 4, 8, 16 and 32-point transforms, respectively. Considering MxN blocks, to calculate the required clock cycles, equation (4.4) is applied for 4x4, 8x8 and 16x16. For 32-point implementation it is equal to 15x32= 480 cycles since the 32-point transforms are not pipelined.

4.3.2 2D-AMT implementation approach

Using its separable property, an (MxN)-point 2D AMT could be computed by the row-column decomposition technique in two distinct stages:

1. STAGE-1: N-point 1D AMT is computed for each column of the input matrix to generate an intermediate output (Y_{int}).
2. STAGE-2: M-point 1D AMT is computed for each row of the intermediate output matrix to generate desired 2D output.

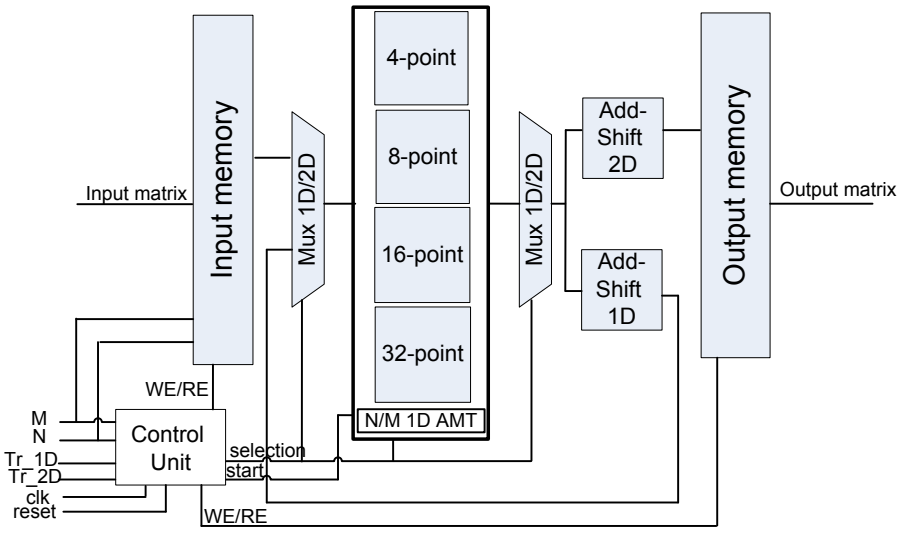

Figure 4.13 – Proposed 2D AMT architecture

Figure 4.13 illustrates the proposed architecture for the 2D AMT approach. Depending on the two block size parameters MxN, the control unit uses the input memory to store the input data. A *start* signal is given to begin the 1D transform. If N = 4, 8 or 16, input columns are read from memory each two cycles within M *start* signals.
N-point transform module operates to provide the 1D outputs. The first output values are available after the required latency according to the transform order (N) and type as explained earlier in Table 4.4 and Table 4.5. At the next clock cycle, they are stored in temporary registers after the corresponding Add and Shift operations to be rounded and saturated to 16 bits. Once the first N outputs are available, within every two cycles, new outputs are obtained until reaching M rows. When N is equal to 32, *start* signal is given only if the corresponding outputs are available and stored due to the absence of pipeline for the 32-point case.
The final *done-N* signal indicates that 1D intermediate outputs are available and stored in the corresponding registers. Subsequently, the 2D transform process can begin. The 2D transform type is assigned and M-point transform module will operate. The 1D temporary outputs, transposed, will be the inputs of 2D process. The same 1D transform principle

explained above is applied only with reversing M and N as block sizes may have asymmetric combinations. Finally, every 2D M-outputs are stored and displayed two by two via FIFO memory blocks. Delivering and managing the WE/RE signals for the used memories and assigning the appropriate modules, all are guaranteed by a control unit according to a state machine.

4.4 Experimental and synthesis results

4.4.1 Experimental setup

The proposed 2D transform design is implemented using the Verilog HDL description language. The architectures of 1D and 2D processes of different orders have been tested with state of the art simulation and synthesis software tools [94],[95]. Test bench files and JEM4.0 reference vectors were used to validate the output results.

4.4.2 Synthesis results of 1D-AMT implementation

The objective is to implement the five AMT transform types with sizes up to 32. Therefore, even if the used platform offers a large number of DSP blocks, it will not obviously cover all the multiplication operations. The LPM multiplier cores IP [19] are characterized to be configurable either to use the default implementation via registers and Aluts or use dedicated circuitry i.e DSP blocks to preserve the logic utilization. With this property we can manage to customize the number of DSPs and avoid exceeding the available resources. All the synthesis are realized with the corresponding software tool [94] under the FPGA target device. Table 4.6 shows the synthesis results of 4 point module implementation and the DSPs usage of the design. Using only 3% of DSPs (42), logic utilization is reduced by about 30% (Alms & registers). The AMT module of larger size would increase the DSPs usage.

Table 4.6 – Synthesis results of the proposed 1D 4-point AMT design

	without DSPs	with DSPs
Pins	175	175
Alms	1915	1156
Registers	4597	3222
DSPs	0	42 (3%)
Frequence	550 MHz	532 MHz

Since the 32-point module is the most complex, the LPM multipliers required for the five transform types implementation are configured to use DSPs. However, 4, 8 and 16-

point modules are implemented using the default implementation resources (without DSPs). Synthesis results of the 8 and 16 point modules are given in Table 4.7.

Table 4.7 – Synthesis results of 1D 8 and 16-point AMT designs

	1D 8-point	1D 16-point
Pins	343	679
Alms	9558	48982
Registers	25525	156328
DSPs	0	0
Frequency	537 MHz	414 MHz

The high number of used registers shown in Table 4.7 is mainly due to two reasons: the first one is the use of registers enabling the pipeline through the assignment stages and the second one is the use of the default logic resources through LPMs multipliers. On the other hand, as shown in Table 4.8, the absence of assignment stages i.e pipeline (as explained in section 4.3) and benefiting from DSP blocks for the 32-point AMT module reduce the usage of logic and register resources.

Table 4.8 – Synthesis results of the proposed 1D 32-point AMT design

Design	Pins	Alms	Registers	DSPs	Frequency
1D-AMT	72	45865	72425	1561	254 Mhz

The 32-point design is adjusted using FIFO memories to provide two by two 16-bit inputs and outputs in order to avoid pin assignment problem. As the DCT-II and DST-I have recursivity property, LPM multipliers of components from lower order modules are reconfigured to use the DSPs blocks in the 32-point implementation. To more evaluate all 1D implementation design performance, Table 4.9 summarizes the frame rate in fps that can be processed for 2K and 4K video resolutions.

Table 4.9 – Performance of 1D 4, 8, 16 and 32-point designs

1D-AMT size	Cycles	Frequency	2K fps	4K fps
4-point	13	550 Mhz	217	54
8-point	29	537 Mhz	381	95
16-point	61	414 Mhz	559	140
32-point	480	254 Mhz	174	44

Square block sizes and worst cases are considered for all 1D AMT implementations to compute the frame rate in fps by equation (4.5).

$$framerate(fps) = (Freq\, M\, N) \,/\, (C_{Cycles}\, Res\, \frac{3}{2}), \tag{4.5}$$

where *Freq* is the required operational frequency, $M\,N$ the size of the processed block, C_{Cycles} the clock cycles required for processing the block, *Res* the target video resolution and the term $\frac{3}{2}$ is a factor related to the image color sampling in 4:2:0.
We can notice from Table 4.9 that the efficiency of 1D AMT implementation increases with larger block sizes. This is due to the proposed pipeline architecture that enables clock cycles preservation when higher rows are computed. The 16-point AMT design can support 2K and 4K videos at 559 and 140 fps, respectively.
On the other hand, even if the 1D 32-point module is not pipelined, it is still efficient enough to sustain real time coding with 174 and 44 fps for 2K and 4K video resolutions, respectively. This is justified by reducing the adder tree stages and using the internal LPM Cores and DSP blocks offered by the target device.
The proposed architecture offers better performancein terms of processed frame rate with respect to state of the art 1D AMT implementation [77]. For large block sizes 16x16 and 32x32, the proposed design is able to perform more than twice frame rate for 2K and 4K resolutions video as shown in Table 4.10.

Table 4.10 – Comparison of proposed 1D AMT transform designs with solution in [77]

	4-point		8-point		16-point		32-point	
	[77]	Prop	[77]	Prop	[77]	Prop	[77]	Prop
Alms	501	1915	501	9558	501	48982	501	45865
DSPs	16	0	16	0	16	0	16	1561
RAM	640 Kb	0	640 Kb	0	640 Kb	0	640 Kb	0
Freq	458	550	458	537	458	414	458	254
2K fps	585	217	294	381	146	559	72	174
4K fps	146	54	73	95	36	140	18	44

However, it is worth noting that in terms of logic utilization, the proposed design have higher resource consumption. The work in [77] benefits from RAM memory of 640 Kbit to preserve the logic cost. This would be an objective for our future works. Reducing the number of reserved registers and Aluts can allow the pipeline of the 32-AMT module and further enhance the speed performance.

4.4.3 Synthesis results of 2D- AMT implementation

The synthesis results of the unified 2D implementation (Section 4.3-C) are presented in Table 4.11. The design reaches an operational frequency of up to 147 Mhz using about 53% of the device logic resources and 93% of the available DSPs.

Table 4.11 – Synthesis results of the unified 2D 4, 8, 16 and 32-point AMT design

Design	Pins	Alms	Registers	DSPs	Frequency
2D-AMT	72	133017 (53%)	274902	1561 (93 %)	147 Mhz

The performance of the unified design is evaluated in Table 4.12. This table presents the frame rate in fps that can be processed for different 2D block size combinations computed using equation (4.5). Cycles involved in transform types selection and in intermediate 1D outputs transposition are taken into account in the 2D clock cycles calculation. However, cycles reserved to store the input data and display final 2D output data are not considered.

Table 4.12 – Performance of unified 2D design

2D-AMT size	Cycles	2K fps	4K fps
4x4	30	25	7
8x8	62	49	12
16x16	126	96	24
32x32	964	50	13
32x16	337	72	18
16x8	94	64	16
8x32	201	60	15

The proposed design enables high frame rate performance. It should be noted that the larger block size is, the better the results are as long as the pipeline is going deeper with more rows to compute. These numbers are obtained supposing the same size for all transforms. However, in real applications, each frame is encoded with a mix of transform block sizes. Regarding this, the 2D design may have better performance. In addition, in future works, as we intend to reduce the high register number reserved for the pipeline process, the 32-point module can also be pipelined and the 2D design may work at higher operational frequency with less clock cycles.

A fair comparison with other works in literature is quite difficult. Most of works are focusing on the 2D-HEVC DCT-II. Works related to AMT hardware implementation adopt either 2D implementation up to only 8x8 block size [76] or only 1D implementation supporting square block sizes up to 32x32 [77]. Table 4.13 summarizes the key parameters to compare the proposed unified design performance with state of the art works.
The proposal presents the union of 4, 8, 16 and 32-point transform modules. It also controls all possible combinations of not only block sizes which can be asymmetric but also transform types which differ from 1D and 2D processes. Furthermore, it manages the Input/Output memory blocks delivering the appropriate WE and RE signals depending on the block sizes. Finally, the whole process is managed by a definite state machine.

Table 4.13 – Comparison of different 2D hardware transform designs

Solutions	[96]	[75]	[74]	[76]	[77]	Proposed
Technology	ASIC 90 nm	ASIC 90 nm	28 nm FPGA	40 nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA
ALMs	–	–	–	5292	999	133017
DSPs	0	0	128	–	32	1561
Frequency (Mhz)	187	150	222	167	458	147
Frames/sec	7680x4320p60	1080x720p30	3840x2160p30	3840x2160p30	3840x2160p18	1920x1080p50
Max bit length	25	25	25	27	–	26
Transf unit	4x4, 8x8, 16x16, 32x32	4x4, 8x8, 16x16, 32x32	4x4, 8x8, 16x16, 32x32	4x4, 8x8	4x4, 8x8, 16x16, 32x32	4x4, 8x4, 16x4, 32x4,4x8, 8x8, 16x8, 32x8,4x16, 8x16, 16x16, 32x16,4x32, 8x32, 16x32, 32x32
Transf type	DCT-II	DCT-II	DCT-II	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V
Dimension	2D	2D	2D	2D	1D	2D

The 2D constraints obviously increase the complexity level and the critical paths for the synthesis results adding some internal delays. This may affect the performance in terms of area and frame rate and operational frequency. It is not the case for the 1D process where almost all these constraints do not interfere.
The first purpose of designing a unified circuit involving all 4, 8, 16 and 32-point transform types is preserving the area consumption on the target device. The second one which is more interesting is satisfying the asymmetric combinations of the processed unit size as one of the transform core improvements provided by the VVC. Up to the best of our knowledge, this is the first 2D hardware implementation of AMT core supporting 4 up to 32-point transforms and that supports all 2D block sizes combinations.

4.5 Conclusion

The hardware implementation AMT design involving DCT-II, DST-I, DST-VII, DCT-VIII and DCT-V transform types has been investigated in this chapter. A multiplierless implementation of these transforms for size 4 is presented. Concerning the hardware architectures, two aspects have been introduced and compared in this study. The first one is based on eliminating the multiplication operations (with adders and shifters) using symetric and sparse matrices. The second one provides a more optimized computational complexity using state-machines to preserve the hardware resources, while presenting some additional clock cycles.
This work is extended for the other larger sizes where we proposed an unified 2D implementation of the AMT for the VVC standard. A hardware implementation of 1D 4, 8, 16 and 32-point AMT modules using LPM multiplier core IPs and DSP blocks is presented. The 1D architecture design allows to perform 4K video coding at 44 frames per second. To the best of our knowledge, this is the first 2D implementation design that takes into account all asymmetric block size combinations, from 4 to 32. With an operational frequency of up to 147 Mhz, the unified 2D AMT design is able to sustain 2K video coding at 50 frames per second.
As future work, in order to reach higher performance, logic resources involved in pipeline process can be reduced to allow the pipeline of the 32-point design. As a result, higher operational frequency with less clock cycles can be achieved. This can also be motivated by the fact that transform module complexity is reduced in the next VVC standard including only three transform types named as Multiple Transform Selection (MTS).

Chapter 5

Forward-Inverse 2D Hardware Implementation ofApproximate Transform Core for the VVC Standard

Contents

5.1 Introduction

MTS is one of the new concepts introduced in VVC [13]. The earlier version of the MTS, integrated in the JEM, consists of five transform kernels including DCT types II, V and VIII, and DST types VII and I [15]. In VVC, the MTS relies only on three trigonometric

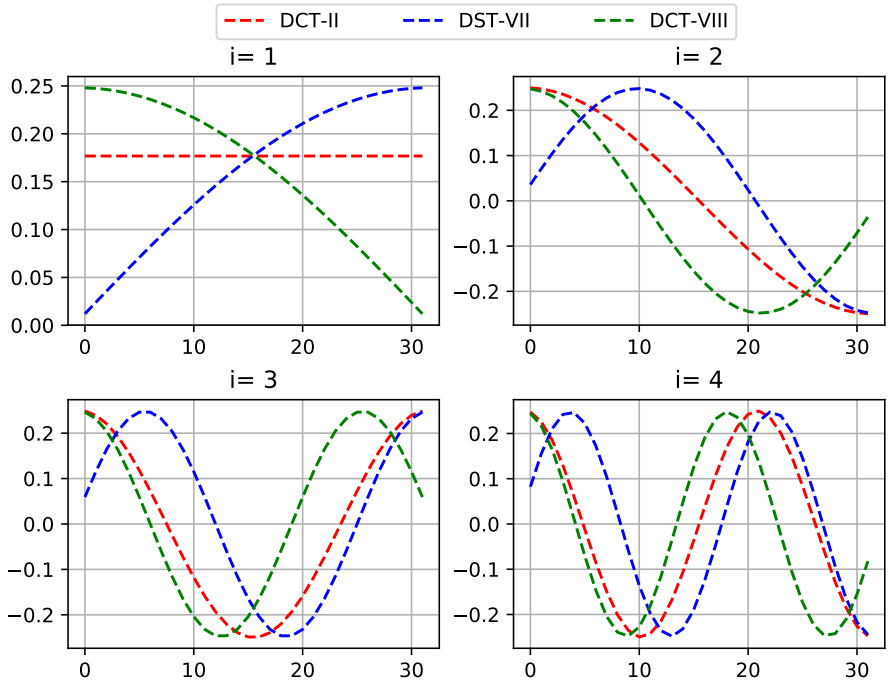

Figure 5.1 – Illustration of the first four basis functions of DCT-II, DCT-VIII and DST-VII

transforms including DCT-II and VIII, and DST-VII. These latter leverage the most of coding gain achieved in the JEM by the five transform types [62].

The basis functions of DCT-II C_2, DST-VII S_7 and DCT-VIII C_8 are given in chapter 3 section 3.2.4 [97], while their first four basis functions ($i = 1, 2, 3, 4$) are drown in Figure 5.1. Besides the usual DCT-II used in video coding standards, VVC encoder selects combinations of DCT-VIII and DST-VII, for the horizontal and vertical transforms, to optimize the RD cost J, a trade-off between distortion D and rate R [66]

$$J = D + \lambda R, \tag{5.1}$$

While the DCT-II has been well studied and optimized with fast implementations [98, 99, 100], the DST-VII/DCT-VIII do not have efficient fast implementation algorithms [101, 102], and rely on classical matrix multiplications. In this chapter we focus on approximating the DST-VII based on the inverse DCT-II and an adjustment band matrix A as initially proposed in [81]

$$\hat{S}_7 = \Gamma \cdot C_2^T \cdot \Lambda \cdot A, \tag{5.2}$$

where $\hat{S}_7$ is the approximation of S_7, $\Gamma \cdot C_2^T \cdot \Lambda$ is equivalent to the DST-III transform S_3, Λ and Γ matrices are computed by Equations (5.3) and (5.4), respectively.

$$\Lambda_{i,j} = \begin{cases} 1, & \textit{if } j = N + 1 - i \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

$$\Gamma_{i,j} = \begin{cases} (-1)^{i-1}, & \textit{if } j = i \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

The DCT-VIII C_8 can be derived from DST-VII, due to their duality property, without additional complexity involving only permutation Λ and sign change Γ matrices

$$C_8 = \Gamma \cdot S_7 \cdot \Lambda. \tag{5.5}$$

This chapter tackles the problem of hardware implementation of the three transform types used in VVC on the target *Arria 10* FPGA platform. The approximation of DST-VII through adjustment band matrix A and inverse DCT-II is first modelled as a constrained integer optimisation problem. The genetic algorithm is then used to solve the problem and compute the adjustment matrices for large transform sizes $N \in \{16, 32\}$.

The approximation methods of DCT-II listed in subsection 3.3.3 of chapter 3 certainly decrease the computational complexity at the expense of some coding loss in terms of image quality (PSNR) and BDR performance. However, unlike the HEVC, MTS involves three transform types. Therefore, in the proposed solution, the DCT-II is not approximated and used instead as the main core to approximate the DST-VII and DCT-VIII. Otherwise, the coding loss would no longer be neglected to preserve the MTS coding gain estimated between 1 to 2% in VVC. The computational complexity, in terms of number of multiplications, additions and shifts of the different DCT-II implementations are summarized in Table 5.1 for different block sizes $N \in \{4, 8, 16, 32\}$.

Table 5.1 – Computational complexity of DCT-II implementations

Transforms	4-point			8-point			16-point			32-point		
	+	×	>>	+	×	>>	+	×	>>	+	×	>>
DCT-II butterfly [73, 65, 74]	8	4	–	28	20	–	100	84	–	372	340	–
DCT-II [75]	17	0	5	74	0	39	232	0	132	548	0	249
DCT-II [76]	88	0	80	784	0	608	–	–	–	–	–	–
Approximate DCT-II[88]	–	–	–	26	20	–	78	64	–	202	152	–
Approximate DCT-II [89, 90]	8	0	2	24	0	6	64	0	12	160	0	24
Forward multiplication	12	16	–	56	64	–	240	256	–	992	1024	–

In this work, we propose an efficient unified and pipelined hardware architecture for both forward and inverse DCT-II. This latter is used to approximate forward and inverse DST-VII and DCT-VIII along with additional adjustment stage at low computational complexity and logic resource allocation. This architecture supports a reconfigurable 2D implementation of approximate DST-VII and DCT-VIII design that can be integrated in both hardware VVC encoder and decoder. In terms of coding efficiency, the approximate DST-VII and DCT-VIII preserve the coding gain brought by the MTS. On the other hand, the proposed unified hardware architecture enables to reach a high frame rate while using a moderate hardware and logic resource of the *Arria10* FPGA device. It enables to process a video in HD and 4K resolutions at 386 and 96 fps, respectively.

The following paragraphs are organized as follows. The approximation of DST-VII, expressed as a constrained integer optimization problem, is described in Section 5.2. Section 5.3 presents the proposed hardware implementation of the 2D approximate transform design. The experimental and synthesis results of 1D and 2D implementations are presented and discussed in Section 5.4. Finally, Section 5.5 concludes the paper.

5.2 Approximation Method of the MTS Transforms

5.2.1 Problem Formulation

In order to reduce the computational complexity and the resource allocation of the transform block, the approximation approach originally proposed in [81] presents an efficient alternative that approximates several DCT/DST types. It consists in applying adjustment stages of low complexity to DCT-II family transforms. The relations between these DCT-II variants transform matrices are expressed as follow

$$\begin{aligned} C_3 &= C_2^T, \\ S_2 &= \Lambda \cdot C_2 \cdot \Gamma, \\ S_3 &= \Gamma \cdot C_2^T \cdot \Lambda, \end{aligned} \tag{5.6}$$

where Λ and Γ are defined in Equations (5.3) and (5.4) with $N \in \{4, 8, 16, 32\}$.
In fact, Λ and Γ matrices can be interpreted by vector reflection and sign changes, respectively, which are computationally trivial. Using the transforms of Equation (5.6), different types of DCT and DST can be approximated by applying adjustment stages (pre-processing and post-processing) to the DCT-II family transforms.

In this paper, we focus on the approximation of the DST-VII based on the inverse DCT-II and then DCT-VIII can be derived from the approximate DST-VII $\hat{S}_7$ as expressed in

Equation (5.5). Table 5.2 gives the DCT-II family used to approximate forward and inverse DST-VII and DCT-VIII.

Table 5.2 – DCT-II variants used to approximate DST-VII and DCT-VIII

Transform type	DCT-II	DST-VII	DCT-VIII
Forward	DCT-II	DST-III	DCT-III
Inverse	DCT-III	DST-II	DCT-II

5.2.2 Approximation through Adjustment Stage

The proposed DST-VII approximation ($\hat{S}_7$) enables to reduce the DST-VII computational complexity since it only involves the DCT-II transform and a multiplication by a band matrix A. Therefore, the complexity of this approximation is equal to the complexity of the DCT-II plus the complexity related to the multiplication by the band matrix A which depends on the maximum number of non-zero coefficients by row θ. The complexity of the multiplication by the matrix A in terms of numbers of multiplications and additions are given by $\theta\, N$ and $(\theta - 1)\, N$, respectively.
The error between the DST-VII S_7 and its approximation $\hat{S}_7$ is expressed by a weighted least-squares error:

$$E(A) = \sum_{i=1}^{N} \omega_i \sum_{j=1}^{N} \left(S_{7\,i,j} - \hat{S}_{7\,i,j} \right)^2, \tag{5.7}$$

where ω_i, $i \in \{1, \ldots, N\}$ is a weight vector of size N which might account for the relative importance of the frequency components. When the ω_i is constant equal to 1, the error function corresponds to the squared Frobenius norm.
Orthogonality has to be taken in consideration for the adjustment matrix A. This property is required since it enables the use of transpose matrix instead of its inverse to recover the original signal without introducing losses at compression stage. The orthogonality of the adjustment matrix A can be expressed by Equation (5.8):

$$O(A) = ||A \cdot A^T - I||_2^2, \tag{5.8}$$

where I is the identity matrix and $||\cdot||_2$ stands for the Euclidean norm. The objective function of this constrained optimization problem is expressed with a Lagrangian multiplier λ as follows:

$$\min_A E + \lambda\, O(A). \tag{5.9}$$

Equation (5.9) aims to minimize the trade-off between error $E(A)$ and orthogonality $O(A)$ of the approximate transform $\hat{S}_7$. The trade-off between approximation and orthogonality can be tuned by the Lagrangian parameter λ. The second constraint on the adjustment stage is

to be sparse block-band matrix, which is easy to compute with small number of taps. The optimal solution of the optimization problem of Equation (5.9) consists in the $A^\star$ matrix that leads to the the original DST-VII S_7 expressed as follows

$$A^\star = \Lambda \cdot C_2 \cdot \Gamma \cdot S_7, \tag{5.10}$$

with $E(A^\star)$ and $||A^\star \cdot A^{\star\,T} - I||_2^2$ terms are both equal to zero. We applied Equation (5.10) to compute $A^\star$ matrix for $N = 8$ with values multiplied by 2^β (with β the bit-depth set to 7 bits) and rounded to the nearest integer $\tilde{A}_{8,8}$

$$\tilde{A}^\star_{8,8} = \begin{pmatrix} 127 & 11 & -6 & 4 & -2 & 2 & -1 & 0 \\ -10 & 125 & 20 & -10 & 6 & -4 & 2 & -1 \\ 6 & -16 & 122 & 30 & -13 & 7 & -4 & 1 \\ -4 & 10 & -22 & 118 & 39 & -15 & 7 & -2 \\ 4 & -8 & 14 & -27 & 114 & 47 & -15 & 4 \\ -3 & 7 & -11 & 18 & -32 & 110 & 53 & -10 \\ 3 & -6 & 10 & -15 & 23 & -38 & 109 & 47 \\ -2 & 4 & -7 & 10 & -15 & 22 & -39 & 118 \end{pmatrix}$$

However, the A^* solution is not appropriate as it does not provide integer values, as required for video coding applications, and does not reveal a sparse property, leading to fewer arithmetic operations. $\tilde{A}^\star_{8,8}$ has its most significant absolute values around the diagonal and lower absolute values are located at lower-left and upper-right parts of the matrix. This property of the adjustment matrix A is stronger for adjustment matrices of higher sizes $N \in \{16, 32\}$.

In this paper, adjustment band matrix that minimizes the trade-off between error and orthogonality is sought with the constraint of A to include few integer values different from zero. This discrete constrained optimization problem is expressed as follows

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & E(A) + \lambda\, O(A) \\ \text{subject to} \quad & A_{i,j} = 0, \quad \forall\, j > i + \lfloor \theta/2 \rfloor \\ & A_{i,j} = 0, \quad \forall\, j \leq i - \lceil \theta/2 \rceil, \\ & i, j \in \{1, \ldots, N\}^2, \\ & A_{i,j} \in \mathbb{Z} \cap [\, -2^\beta + 1,\, 2^\beta \,], \\ & \lambda \in \mathbb{R}^+. \end{aligned} \tag{5.11}$$

It has been shown in [103] that the DST-VII is optimal in terms of energy packing for image intra-predicted residuals. Indeed, those residuals have an auto-correlation matrix which is

tri-diagonal matrix R_x of size NxN expressed by Equation (5.12).

$$R_{x\,i,i} = b,\ R_{x\,i,i+1} = c,\ R_{x\,j-1,j} = a,\ R_{x\,N,N} = b - \alpha, \tag{5.12}$$

with $(a, b, c, \alpha) = (-1, 2, -1, 1)$ and $1 \leq i < N$, $1 < j \leq N$ The eigen-vectors of the matrix R_x are the basis of the DST-VII transform [97]. Therefore, for the approximation of the DST-VII, we propose to weight the relative importance of the approximation basis with the eigen-values of the of the auto-correlation matrix R_x. This gives more importance to the lower frequency range where an important part of the signal energy stands. According to [104] the eigen-values are computed as follows

$$\omega_i = b + 2\sqrt{a\,c}\,\cos\left(\frac{2\,i\,\pi}{2N+1}\right),\ \ i = 1, \ldots, N. \tag{5.13}$$

5.2.3 Genetic Search Algorithm

To provide an approximation of the DST-VII, the adjustment matrix, which consists of a selected number θ of integer values around the diagonal, need to be determined for a desired level of orthogonality $O(A)$ expressed in Equation (5.8). To solve this problem in the integer domain, continuous optimization methods such as gradient descent are not appropriate. Also, an exhaustive search would result in evaluating $\left(2^{\beta+1}+1\right)^{\theta\,N}$ combinations (β is the bit-depth set to 7 bits). Techniques such Integer Programming [105] can provide helpful techniques in that context. However, in this study, a genetic algorithm approach was preferred as it provided satisfactory results and appeared to converge well.

Genetic algorithms, are easily re-configurable to address various scenarios such that the adjustment matrix with different number of coefficients per row. Indeed, this optimization algorithm solves Equation (5.8) with $\theta\,N$ parameters with the same strategy. Basically, it consists in changing individual elements of the adjustment matrix in the *mutation* process. Although convergence is not guaranteed with the Genetic Algorithm approach, it appears in practice that it converges in a consistent fashion with different initialization points.
The principle of the genetic search is the following:

- From a set of N_p selected adjustment matrices, called parents, N_c children are created by individual changes in the close-to-diagonal values. One among the children's values, randomly selected, is changed by the addition of +/-1 while ensuring that the value remain in the adjustment matrix bit-depth range.
- The resulting $N_p\,N_c$ adjustment candidate matrices are evaluated with Equation (5.9), this can be done in parallel, e.g. using OpenMP programming interface [106].

- From the candidate matrices, $N_p - 1$ are randomly retained, and the best performing matrix that minimizes the trade-off between error and orthogonality is kept. From these N_p matrices the three steps are re-iterated until convergence of the algorithm.

As the A matrices have coefficients around the diagonal, the number of parameters depends on the the matrix size and the number of coefficients per row θ. It is in the range of $\theta\, N$, each coefficient is to be expressed on β bits to allow implementation on fixed-point devices. The convergence is measured in terms of stabilization of the algorithm, when there is no further reduction of the optimized metric after many iterations. The λ value is modified in order to provide different solutions in the approximation / orthogonality space. It is essential in video coding to provide transforms with a reconstruction level, sufficiently low, to avoid the introduction of distortion in the transform process. Subsequently, λ needs to be chosen in a way that the orthogonality measure $O(A)$ is in the range of -60 dB, the same orthogonality level than the discrete DCT-II used in VVC.
For this work, coefficients of 5 tap sparse block-band adjustment matrices $\theta = 5$ are used with an additional constraint of symmetry across the diagonal between non-zero coefficients

$$\begin{aligned} A_{j,i} &= -A_{i,j}, \ \ \forall\, j = i+1, \ \ \ i \in \{1, 2, \ldots, N-1\}, \\ A_{j,i} &= A_{i,j}, \ \ \forall\, j = i+2, \ \ \ i \in \{1, 2, \ldots, N-2\}. \end{aligned} \tag{5.14}$$

The value $\theta = 5$ is selected since it achieves a good trade-off between complexity and approximation of the original DST-VII transform.

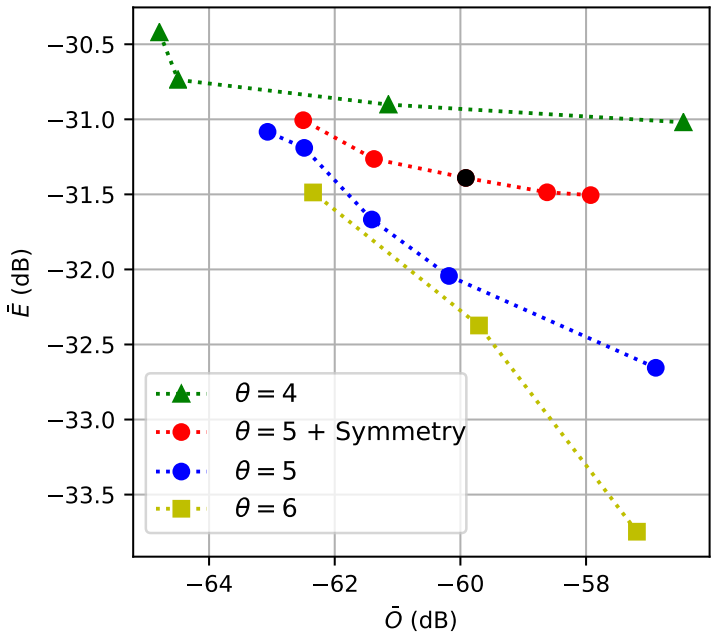

Figure 5.2 – Performance of approximate DST-VII transform $N = 32$.

The symmetric property reduces the memory storage of the adjustment matrix, and enables a faster convergence of the Genetic algorithm. Figure 5.2 illustrates the error and orthogonality performance of the proposed solution for different θ values. The configuration highlighted in black, enabling the desired level of orthogonality around -60 dB and symmetry of coefficients with $\theta = 5$, is selected for hardware implementation and its coding performance is assessed under the VTM 3.0 software.

5.3 2D Hardware Implementation of Transform Module

As expressed in Equation (5.6) giving the relations between DCT-II types, the approximations of forward and inverse DST-VII are performed by applying adjustment stages to inverse DCT-II C_2^T and the forward DCT-II C_2, respectively. In the following we detail the implementation of the main DCT-II forward and inverse transforms, which are then used to approximate 2D forward and inverse DST-VII and DCT-VIII transforms.

5.3.1 Unified Forward and Inverse DCT-II Core Transform

In this section the C_2^N corresponds to the N-point DCT-II matrix with $N \in \{4, 8, 16, 32\}$. The DCT-II and IDCT-II N-point kernels are computed by Equations (5.15) and (5.16), respectively

$$C_2^N = P^N \cdot \begin{pmatrix} C_2^{N/2} & 0 \\ 0 & O^{N/2} \end{pmatrix} \cdot \begin{pmatrix} I^{N/2} & J^{N/2} \\ -J^{N/2} & I^{N/2} \end{pmatrix}, \tag{5.15}$$

$$[C_2^N]^T = \begin{pmatrix} I^{N/2} & -J^{N/2} \\ J^{N/2} & I^{N/2} \end{pmatrix} \cdot \begin{pmatrix} [C_2^{N/2}]^T & 0 \\ 0 & O'^{N/2} \end{pmatrix} \cdot P^N, \tag{5.16}$$

where P^N is a permutation matrix to reorder the output data in appropriate form, $C_2^{N/2}$ is the DCT-2 of size $N/2$, $O^{N/2}$ is a matrix of size N/2 x N/2 consisting of odd rows of the first $N/2$ columns of the C_2^N matrix. $I^{N/2}$ and $J^{N/2}$ are, respectively, the identity and the cross-identity (reflection) matrices of size N/2 x N/2. Finally, $O'^{N/2}$ is a matrix of size $N/2xN/2$ consisting of odd rows of the first $N/2$ columns of the $[C_2^N]^T$ matrix.
Comparing $O^{N/2}$ and $O'^{N/2}$, we notice that for i from 1 to $N/2$, $O^{N/2}$ i^{th} column has the same coefficients than the $N/2$- i^{th} column of $O'^{N/2}$ but in inverse order. Subsequently, $O'^{N/2}$ can be implemented using the same architecture than $O^{N/2}$. This can be achieved with computationally trivial steps, by inverting the inputs and outputs orders. As a result, a unified architecture design is proposed to embed forward and inverse DCT-II sharing the same Nx N odd part of the C_2^N matrix, which is the most complex part.
Therefore, benefiting from recursion property as presented in Figure 5.3, the same principle is applied for lower block sizes to deepen the hardware sharing in the unified circuit.

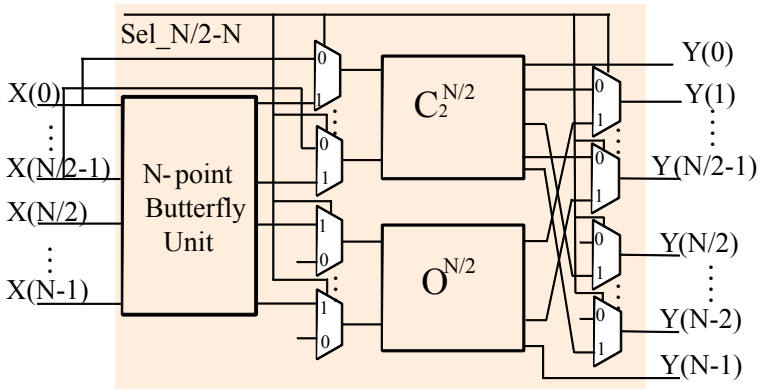

Figure 5.3 – Proposed architecture of recursive C_2^N implementation with N=8, 16 and 32.

In terms of required number of operations, the state of the art 32-point butterfly forward and inverse DCT-II implementation requires 680 multiplication operations according to Table 5.1 for both DCT-II and Inverse DCT-II. The proposed architecture of the unified DCT-II and IDCT-II requires only 344 multiplication operations: 256 (odd part) plus 88 (even part) multiplication operations of $C_2^{16}/[C_2^{16}]^T$).

Recursion property and reusing the same architecture of different odd-part matrices in a unified DCT-II/IDCT-II scheme enable considerable reduction in logic resource and allow to preserve 256, 64 and 16 multiplication operations respectively for 32, 16 and 8-point designs. Table 5.3 details the computational complexity of the proposed architecture design for different block sizes from 4 to 32 considering forward and inverse processes.

Table 5.3 – Computational complexity of the proposed forward and inverse DCT-II and approximate DCT-VIII and DST-VII implementations

	4-point			8-point			16-point			32-point		
	+	×	>>	+	×	>>	+	×	>>	+	×	>>
DCT-II butterfly [73, 65, 74]	16	8	–	56	40	−−	200	168	−−	744	680	−−
Forward matrix multiplication	24	32	−−	112	128	−−	480	512	−−	1984	2048	−−
Proposed DCT-II	16	8	−−	36	24	−−	92	88	−−	332	344	−−
Proposed Approx DST-VII	24	32	−−	112	128	−−	51	58	9	112	114	30

Moreover, multiplication operations are performed using the LPM IP Cores multipliers offered by DSP blocks of the Arria 10 FPGA device. Figure 5.4 illustrates the proposed architecture of the unified DCT-II/IDCT-II core transform.

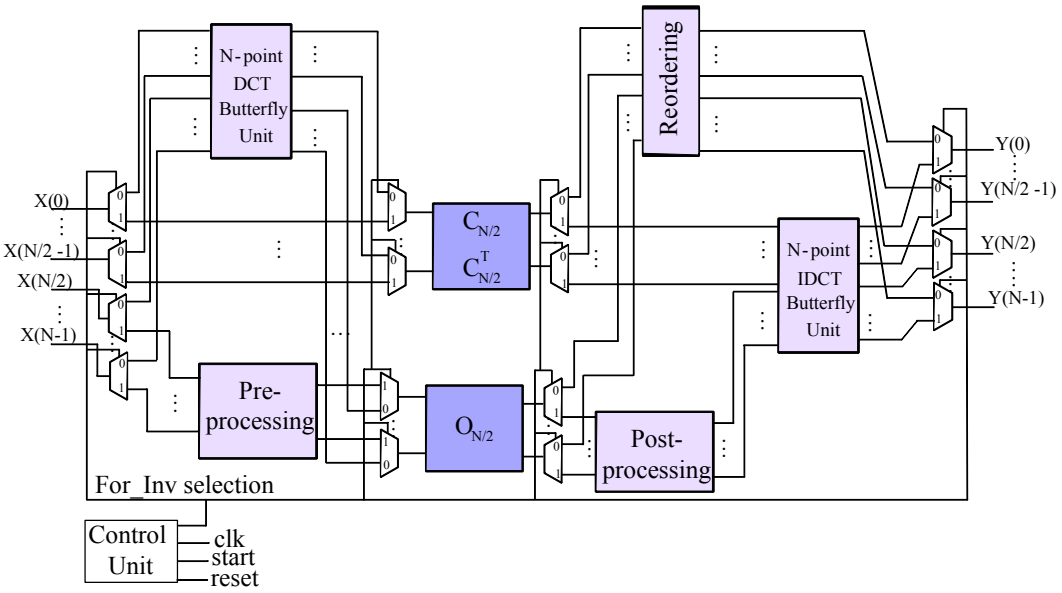

Figure 5.4 – Proposed architecture of unified 32-point DCT-II and IDCT-II core transforms

From equations (5.15) and (5.16), and benefiting from butterfly decomposition architecture, the difference between DCT-II and IDCT-II is the hierarchical application of the associate butterfly block; as a first or last stage for forward and inverse processes, respectively, depending on *Forward-Inverse* selection signal.

For the IDCT $[C_2^{32}]^T$ computation, the 32-odd part is computed as O'^{16}. Trivial pre-processing and post-processing steps on its associated inputs and outputs are applied with no additional computing complexity. The obtained results of the $[C_2^{16}]^T$ implementation (16-point IDCT-II) outputs go through IDCT-II butterfly stage in order to provide the final IDCT-II 32-point outputs.

O'^{16} implementation requires 16 clock cycles where all multiplications are performed at one clock cycle using LPM multipliers, then adder trees (with two addition operations) are placed sequentially to generate the output. The pipeline installed consists in introducing assignment stages. They are based on registers and have basically two roles: storing the current results and transferring the appropriate data and intermediate signals to the next stage. These components are responsible for the pipeline operation avoiding data conflicts or loss which may occur in the next clock cycles as inputs are refreshing [80].

Figure 5.5 presents a timing diagram of 1D IDCT-II computation of 32x32 input block. It details the different steps and latency required to generate 1D output results.

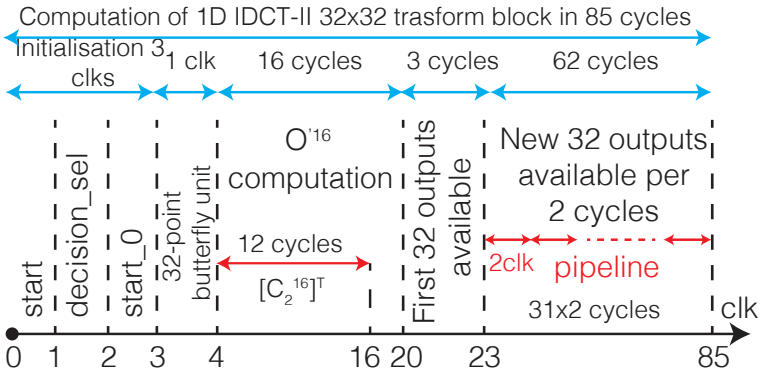

Figure 5.5 – Timing diagram of 1D IDCT-II 32x32 block computation

In the case forward DCT-II (*Forward-Inverse* is equal to 0), the 32-point odd part is computed as O^{16}. Then the obtained results together with the C_2^{16} multiplication (16-point DCT-II) ones form the final outputs of 32-point DCT-II. The design is not only unified for forward and inverse DCT, but also for all block sizes from 4 to 32 through a size dependent selection process.

5.3.2 Hardware Architecture of Adjustment Stages

As explained in Section 5.2-A, the approximation method is based on DCT-II architecture and diagonal-sparse orthogonal adjustment matrices with low computational complexity. These latter are generated using the genetic algorithm detailed in Section 5.2-C. In this work, we consider 16 and 32 approximation orders as they are the most complex cases. 16 and 32-point adjustment matrices of DST-VII are 5 tap sparse block-band matrices. As

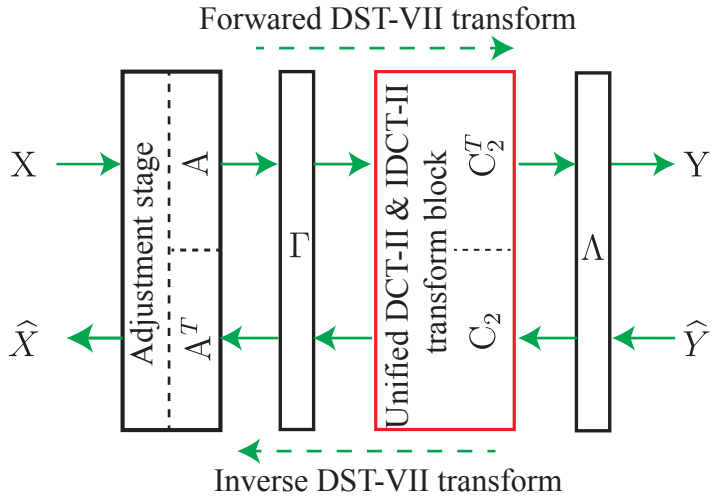

Figure 5.6 – 1 D approximate DST-VII transform scheme using the pre/post processing stages

illustrated in Figure 5.6, the adjustment matrices are placed and used as a pre-processing stage in the forward transform process, and a post-processing stage in the inverse one.
With a maximum of 5 coefficients per row, it would require 80 and 160 multiplications for 16 and 32-point orders, respectively. However, it is worth noting that not all adjustment matrix rows include five coefficients, and coefficients with power-of-two values are implemented using shift operations, which would further reduce the number of required multipliers.
The symmetry property of the adjustment matrix A expressed by Equation (5.14) enables to use the same coefficients to perform the multiplication by its inverse A^T in the post processing stage of the inverse DST-VII. Therefore, we can use the same implementation of the adjustment matrices in both forward and inverse transform processes and half of associated computational complexity is preserved. Then, as the approximation approach consists in using the DCT-II architecture, DST-VII implementation requires only the number of operations included by the adjustment matrices implementation over the DCT-II ones. The approximate DCT-VIII $\hat{C}_8$ is obtained easily using approximate DST-VII $\hat{S}_7$ architecture with only some changes in input and output order and signs as expressed in Equation (5.5). Therefore, the approximate DCT-VIII transform requires almost no hardware resource (except one multiplexer) and does not introduce additional computational complexity. Table 5.3 shows the computational complexity required for the proposed DCT-II, and approximate DCT-VIII and DST-VII implementations for both forward and inverse operations. Approximation through adjustment stages is used for 16 and 32-point orders because they are the most complex cases. For 4 and 8-point DST-VII, straightforward matrix multiplication is used as presented in Table 5.3.

5.3.3 Proposed 2D Implementation of VVC Transform Approximation

Using property of separable transforms, the 2D process could be computed by the row-column decomposition technique in two distinct stages. First, a 1-dimensional unit is performed for each column of the input matrix to generate the intermediate output. This unified circuit enables to compute DCT-II, approximate DCT-VIII or DST-VII depending on the selected transform type as illustrated in Figure 5.7.

Once the first N intermediate 1D outputs are available, they are scaled and stored in N Dual-Port RAM (16x512 i.e 16x32x16) at a column order ($IntermOu\_0\_0$ .. $IntermOu\_0\_31$). Figure 5.8 shows the structure of the transposition memory. After storing the outputs of the N columns, the first block memory ($RAM0$) will contain the first results computed from each column. For $N = 32$, data of $RAM0$ output port buffer will be the concatenation of all first output values from every processed column (32 values of 16 bits each: $IntermOu\_0\_0$ .. $IntermOu\_31\_0$). The advantage of using dual port RAMs enables to read the 32 values with a single reading signal. The same principle is used to secure the storage process in $RAM1$

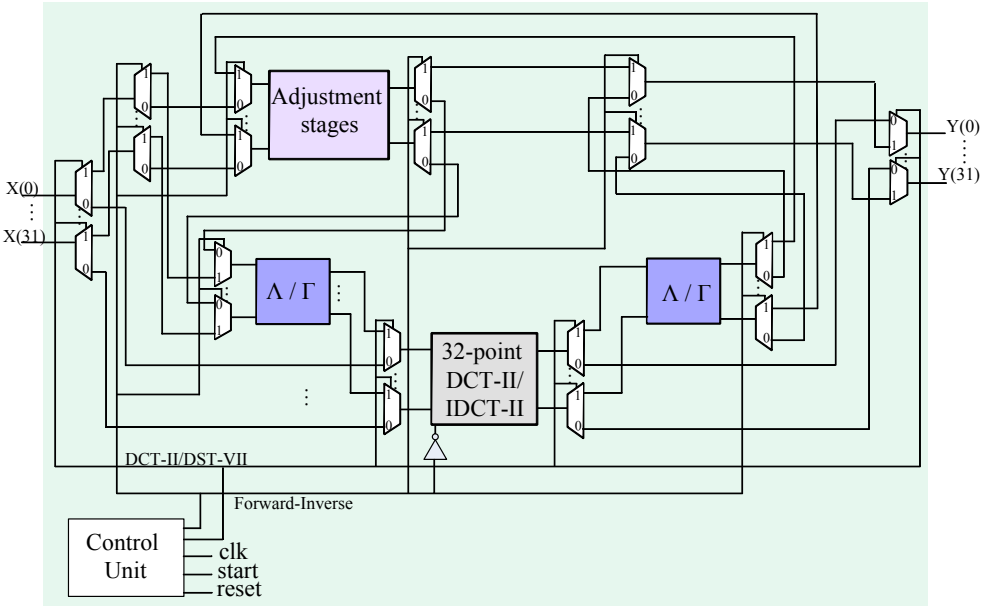

Figure 5.7 – Proposed unified architecture of approximate forward-inverse transform design

to $RAM31$. As a result, considering 32x32 1D intermediate output matrix, assigning consecutive r_ena signals for the 32 RAMs sequentially, leads to automatically transpose the results and fed them as inputs to the same 1D architecture in order to generate the desired 2D output.

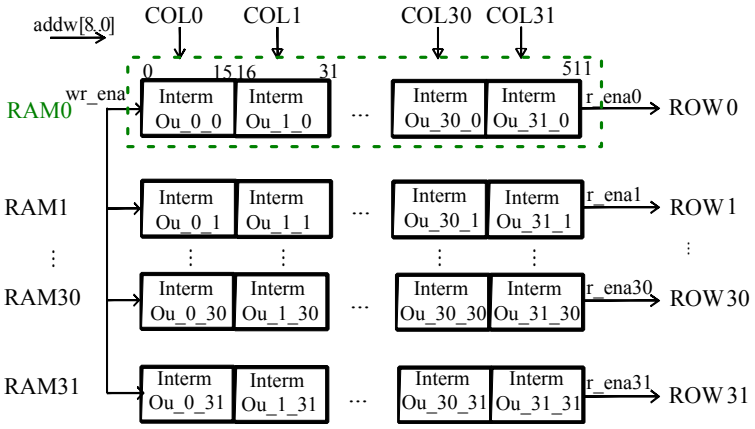

Figure 5.8 – Architecture of the transposition memory with 32 Dual-Port RAMs. Data can be written at a column basis and read at a row basis

The proposed 2D circuit is able to efficiently compute approximate DST-VII and DCT-VIII transforms using a unified 1D forward-inverse DCT-II core transform and adjustment stages circuit. Moreover, it is unified for both 16 and 32 block sizes and reconfigurable to perform either forward or inverse transform processes. Input and output First In First Out (FIFO) memory blocks are added in both ends of the design, each of size 16 Kbits (16x32x16), to

store and display input and output vectors. Moreover, a control unit according to a state machine is defined. It enables to assign the appropriate signals and blocks, and control reconfiguration aspects. In addition, it manages the different steps of 2D pipeline process.

5.4 Experimental and Synthesis Results

5.4.1 Experimental setup

The coding and complexity performance of the proposed approximate solution are investigated in this section under the VVC CTC. Those experiments are tested among mandatory video classes, where each class corresponds to a specific resolution (up to 4K video) and video content characteristic (with computer generated and visio-conference materials). The proposed approximate DST-VII and DCT-VIII have been integrated in the VTM draft 3.0 reference software [107]. The BD-R metric is used to assess the coding performance over four bitrates between two coding configurations giving the bitrate gain/loss ($-/+$) in percentage for similar PSNR quality. The encoder and decoder run times are also drown to assess the complexity of the proposed approximations.
On the other hand, for this work, hardware implementation of the transforms is also done using the Verilog HDL description language. The architectures of 1D and 2D processes of different orders have been tested with state of the art simulation and synthesis software tools [94],[95] under Arria 10 SoC FPGA device [17]. Test bench files were used to validate the output results.

5.4.2 Rate Distortion Coding Performance

Table 5.4 gives the coding and complexity performance is terms of both BD-R and run time of the proposed solution. The encoding ($EncT$) and decoding ($DecT$) run times are also compared in percentage to the anchor VTM3.0 [108]. This latter uses the HEVC DCT-II up to size 64 together with DST-VII and DCT-VIII core transforms for MTS, up to size 32, implemented as matrix multiplications. From Table 5.4 it is shown that the proposed approximations of the DST-VII and DCT-VIII introduce slight coding loss of 0.15% in average for the luminance component (Y), and 0.09% for the two chrominance components (U and V) in AI coding configuration. Overall, we can conclude that the coding performance remains similar to the anchor for RA and LD-B Inter coding configurations.

Table 5.4 – Performance (%) in terms of BD-R and run time complexity of approximate DST-VII and DCT-VIII

Class	All Intra Main 10					Random Access Main 10					Low Delay B Main 10				
	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*
A1	0.15	0.11	0.14	93	80	0.12	0.31	0.27	99	97	–	–	–	–	–
A2	0.22	0.12	0.08	95	84	0.09	0.21	0.23	99	98	–	–	–	–	–
B	0.14	0.14	0.20	94	84	0.10	0.31	0.17	99	98	0.07	-0.22	0.11	100	101
C	0.06	0.00	-0.05	95	89	0.07	-0.06	0.35	99	100	0.06	0.15	0.35	100	100
E	0.18	0.10	0.06	94	86	–	–	–	–	–	0.06	0.81	-0.11	100	97
Av.	0.15	0.09	0.09	94	85	0.09	0.19	0.25	99	98	0.06	0.16	0.14	100	100

The encoding and decoding run times slightly decrease with the approximate DST-VII and DCT-VIII in AI configuration, while they remain constant in RA and LD-B configurations. These results can only support the effectiveness of the proposed VVC transform approximation method. In fact, the gain in number of multiplications and additions enabled by the approximate transforms through adjustment matrices is low in the context of the VTM software, which includes other time consuming operations. However, this gain in number of operations as well as in memory usage has a significant impact in the context of hardware implementation on FPGA platforms with limited logic and memory resources.

5.4.3 Synthesis Results and Discussion

Since MTS approximation is based on DCT-II architecture, Table 5.5 presents the area consumption of some related works for 1D 32-point forward DCT-II implementation on different platforms.

Table 5.5 – Area consumption of some 1D 32-point DCT-II implementations in different platforms

ButterflyDCT-II	Dimention	Technology	Area consumption
[89]	1D forward	Xilinx Sparta	18772 (LUT)
[65]	1D forward	TSMC 90nm	253 (Kgate)
[80]	1D forward	Arria 10 Soc	11231 (Alm)

In this paper a unified forward and inverse DCT-II design is proposed. Thus, regarding information given in Table 5.5 it would consider twice the required hardware cost. In addition, for further fair evaluation, we will focus more on [80] work which provided both DCT-II and DST-VII implementations without approximation on the same FPGA target device with similar pipelining approach as used in the proposed work.

Table 5.6 provides more detailed hardware synthesis results of 16 and 32-point DCT-II and DST-VII implementations proposed in [80].

Table 5.6 – Synthesis results of the 1D 16 and 32-point DCT-II and DST-VII [80]

	16-point		32-point	
	DCT-II	DST-VII	DCT-II	DST-VII
Alms	2428(1%)	5981(2.5%)	11231(4.5%)	22794(9%)
Registers	14041(4%)	50135(15%)	76711(22.5%)	186418(55%)
DSPs	84(5%)	186(11%)	276(16%)	681(40%)
Frequency	401 MHz		268 MHz	
Cycles	61		61	
Fps (2K)	541		1440	
Fps (4K)	135		361	

Results presented in Table 5.6 show that the proposed design provides good performance in terms of processed frames per second up to 135 and 361 of 4K videos for 16 and 32-point modules, respectively. It can also be noticed that 32-point module implementation requires about 3 to 5x hardware resources than 16-point one. Moreover, it is worth noting that for 32-point implementation, internal architectures are slightly modified in a way to reduce logic utilization by more than half and also the required clock cycles to compute a 32x32 block (61 for 32-point) [80]. Otherwise, logic resource would be 6x or more and then exceed the target device range. This technique is used in the proposed work without affecting the computational design performance. Furthermore, information given in Table 5.6 refers only to requirements for forward transform configuration. This is only to have an idea on the complexity and required resources of hardware implementation of the MTS.
On the other hand, the implementation of the approximation method, aims to maintain the desirable high performance while keeping minimal logic utilization. Table 5.7 presents the synthesis results of the proposed unified forward/inverse DCT-II core transform. This latter, configured to operate as Forward or Inverse DCT-II, will be used in DST-VII and inverse DST-VII implementations using adjustment stages.

Table 5.7 – Synthesis results of the unified 1D 32-point DCT core transform and the proposed architecture of approximate forward-inverse DST-VII and DCT-VIII

	DCT-II / IDCT-II		Approximation design	
	16-point	32-point	16-point	32-point
Alms	16505 (7%)		23199 (9%)	
Registers	51862 (15%)		69226 (20%)	
DSPs	328 (20%)		500 (30%)	
Frequency	308 MHz		316 MHz	
Cycles	46	85	55	95
Fps (2K)	551	1205	472	1095
Fps (4K)	137	300	118	273

The second part (right) of Table 5.7 gives the synthesis results of the 1D DST-VII approximation implementation. It embeds the DCT-II core transform and then the additional complexity introduced by adjustment stages can be interpreted or deducted as the difference between DCT-II transform core and DST-VII approximation results.
Finally, the synthesis results of the unified 2D approximation circuit are summarized in Table 5.8. The low computational complexity introduced by adjustment stages will have a high impact on the design performance.

Table 5.8 – Synthesis results of the unified 2D implementation design of 32-point forward-inverse DCT-II and approximate DST-VII and DCT-VIII

2D process	DCT-II / IDCT-II		Approximation design	
	16-point	32-point	16-point	32-point
Alms	26130 (10%)		31421 (12%)	
Registers	62109 (18%)		75553 (22.5%)	
DSPs	328 (20%)		500 (30%)	
Memory	64 Kbits (<1%)		64 Kbits (<1%)	
Frequency	225 MHz		228 MHz	
Cycles	95	175	115	196
Fps (2K)	194	423	163	386
Fps (4K)	49	105	41	96

We can notice that the larger block size is, the higher frame rate performance is as long as the pipeline is going deeper with more rows to compute. Thus, the proposal is able to sustain 2K and 4K video processing at 386 and 96 frames per second, respectively. Moreover, it requires only 12% of Alms, 22% of registers and 30% of DSP blocks offered by the target device.
It should be noted that the proposed design is configured to compute one transform type at a time in both sides (encoder and decoder). At the encoder, pixels are processed many times through the rate distortion optimization process. As a result, computing two different transform types in parallel would be an alternative way of further optimization, especially that the presented solution is a low hardware area consuming. Then, the actual throughput in the encoder could be increased.
A fair comparison with other works in literature is quite difficult. Most of works are focusing on the 2D-HEVC DCT-II. There are only few works related to MTS hardware implementation. Table 5.9 summarizes the key parameters to compare the proposed unified design performance with state of the art works. Work in [76] involves 5 transform types but is restricted to only 4x4 and 8x8 block sizes reaching 30 fps for 4K video coding. Work in [78] presents also an interesting 2D implementation of MTS module regarding hardware cost. It is unified for all block sizes from 4x4 to 32x32 but is not able to support real time coding for 4K videos. Work in [80] is considered as the first 2D MTS implementation supporting 5 transform types and all block sizes (including rectangular ones) from 4 to 32. However, it is drawback is the high usage of logic resource. Finally, all these works consider only forward transform process.

Table 5.9 – Comparison of different 2D hardware transform designs

Solutions	[65]	[74]	[76]	[78]	[80]	Proposed
Technology	ASIC 90 nm	28 nm FPGA	40 nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA
Area cons. (Alms)	–	–	5292	3654	133017	36766
DSPs	0	128	–	32	1561	738
Frequency (Mhz)	187	222	167	458	147	228
Throughput (fps)	7680x4320p60	3840x2160p30	3840x2160p30	3840x2160p18	1920x1080p50	3840x2160p96
Transf unit	4x4, 8x8 16x16, 32x32	4x4, 8x8, 16x16, 32x32	4x4, 8x8	4x4, 8x8, 16x16, 32x32	4x4, 8x4, 16x4, 32x4, 4x8, 8x8, 16x8, 32x8,4x16, 8x16, 16x16, 32x16,4x32, 8x32, 16x32, 32x32	4x4, 8x8, 16x16, 32x32
Transf type	DCT-II	DCT-II	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-VII, DCT-VIII	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-VII, DCT-VIII
Dimension	2D	2D	2D	2D	2D	2D
Process	Forward	Forward	Forward	Forward	Forward	Forward + Inverse

On the other hand, it is worth noting that the proposed design enables both forward and inverse transform processes. In fact, associated with the DCT/ IDCT-core transform, the unified circuit is able to compute 2D forward and inverse implementation for DCT-II, approximate DST-VII and DCT-VIII transform types supporting all sizes from 4x4 to 32x32 unlike the other works presented in Table 5.9. As a result, considering this fact would require twice their results. Moreover, as it is mentioned above, the low additional hardware requirements of forward and inverse DST-VII architectures (for 4x4 and 8x8 sizes through matrix multiplication) can be noticed in area consumption and DSP blocks used for the proposed work (Table 5.8 and Table 5.9). Furthermore, the proposed solution is able to sustain 4K video processing at 96 frames per second requiring only moderate hardware cost of the target device.

5.5 Conclusion

In this paper we have proposed the approximation method adopted for hardware implementation of forward and inverse MTS concept of VVC standard. It consists in applying low cost adjustment stages to a DCT-II variant in order to approximate DST-VII and DCT-VIII transform types. An efficient hardware implementation of approximate VVC transform process is also proposed. The 32-point 1D architecture allows to process 4K videos at 273 frames per seconds. It embeds a reconfigurable and pipelined DCT-II core transform to compute forward and inverse DCT-II sharing the most logic consuming part. The proposed unified 2D implementation design can compute forward and inverse DCT-II, DST-VII and DCT-VIII approximation while using only moderate hardware resource of the target device. The unified circuit is able to sustain 2K and 4K video processing at 386 and 96 frames per second, respectively. Future works will aim to include 64 transform order for DCT-II. Moreover, rectangular block sizes would be considered with hopefully similar performance results.

Chapter 6

Co design application of transform module on Arria 10 SoC

This chapter presents a Co design application of HEVC inverse DCT-II implementation on Arria 10 FPGA SoPC using software and hardware tools offered by the platform to enable communication between software driver responsible for sending receiving data and the hardware IP of the IDCT computation.

Contents

6.1 Introduction

Semiconductor technology and design methodologies have recognized a great evolution that allows for the design and the development of complex digital System on Chip (SoC). This latter is a chip integrating a large number of heterogeneous components selected to reach the requirements of specific system supporting complex arithmetic treatments. Components can be processor cores, DSP cores, hardware accelerators, programmable logic and so on. Latest FPGA are good example of SoPCs, which are Programmable SoCs including programmable logic and processor cores to design high performance systems in a die. The design of SoPCs is complex due to its heterogeneity. The control and low complexity computations are mapped and are executed on the processors and high complexity computations are executed on programmable logic resources. Following a hardware/software methodology, the developer must be able to choose which part of its algorithm should be executed on the logic resource of its FPGA. The design of a hardware accelerator on programmable logic resource is mandatory. But then, the developer will have to setup synchronizations and data transfers between the processor and the logic resource.

This chapter presents a co-design application of HEVC inverse DCT-II implementation on Arria 10 FPGA SoC. An overview of the target FPGA device is provided in section 6.2. Section 6.3 presents the different software and hardware tools used in a co-design methodology in general and in the driver development in particular. The driver takes over synchronizations and data transfers and provides a simple software Application Programming Interface (API) to the developer. It benefits from communication bridges using Avalon Memory Mapped FiFOs, used to create the communication with the hardware IP of the IDCT architecture. The co-design application on Arria 10 board is detailed in sections 6.4 and 6.5. Finally, this chapter presents and discusses the performance of the driver in the context of hardware/software co design developments in terms of execution time for different configurations using 32-bit and 64-bit com-bus is section 6.6 followed by a conclusion section 6.8.

6.2 System on Chip and FPGA

A complex digital system is an assembly of different discrete components representing each a particular less complex function such as adder, memory, interface component, processor, etc. Increasing the complexity of a digital system results in an increase in the number of used components, hence the device cost.
Taking into account Moore's empirical law which states that for a given silicon surface, the number of transistors integrated is doubled every 18 months, the design way of complex digital systems is changed; we then move from logic gates level to system level (or functionality).

Several improvements have affected the manufacturing processes of electronic components. The evolution of circuits offers the advantage of being able to integrate a complex digital system on the same component: it is the concept of the single chip.
Recently, the use of a "schematic" approach at the level of logic gates and basic functions Register Transfer Logic (RTL) for designing complex systems is abandoned in favor of a "textual" approach. However, the schematic approach remains valid and used for the design of small systems.
Textual approaches such as High Level Syntax (HLS) allows rapid creation and integrating sub-functions and simplify the designer's work. For FPGAs, the system is designed using a hardware description language such as VHDL or Verilog that offers the possibility of synthesizing a numerical function. They are actually programming languages that use a well-defined and standardized syntax with an associated compiler or simulator. The hardware description languages are used to facilitate the transformation design methodologies. The development of complex digital systems must adapt to different requirements that come up frequently. That is why some companies offer other features known as Intellectual Property (IP) modules (mathematical functions: adders, FIR, bus interfaces...) that can be bought or downloaded freely on the Internet. Thus, designing a complex digital system becomes easier with assembling the IP modules.

6.2.1 SoC / SoPC

A SoC is composed of several functional modules such as Digital Signal Processing (DSP), embedded memories, buses, I/O controller etc... integrated in a single chip or a single piece of silicon with at least one processor.
Early SoCs were mono-processor architectures that controlled devices through master/slave communications where the processor is dedicated to the calculation and control of the entire system. Systems on single chip are dedicated to specific applications in order to satisfy only the intended application needs and could only provide simple functions requiring little computing power.
If the features of the embedded system are implemented in Application-Specific Integrated Circuit (ASIC) components, it is referred then to Systems on Chip or SoC. If they are implemented in logic programmable components of FPGA type, it is called SoPC systems acronym for System on a Programmable Chip. It is an embedded system in which resources can be modified adaptively to the target application by programming or reprogramming. This technology corresponds to the integration of software and hardware resources of a SoC on a programmable logic circuit chip.
The programmable logic circuits are of various natures. The choice of the Look Up Table (LUT)-based FPGA family is mainly justified by its high integration capacity and reconfigurable ability. FPGAs are dedicated to the design of real-time embedded systems. One of

the main advantages is their infinite reconfigurability allowing more maximal flexibility to the designers developing simple or complex applications such as video processing.

- Logic elements
  They present the basic modules of any FPGA circuit which contain all the combinational logic. These structures consist of one or more LUT tables responsible for computing a logical function. They often have the same composition regardless of manufacturer and architecture.
- Memories
  Knowing that FPGAs are more and more used for complex applications requiring large storage resource such as video processing applications it is crucial to extend the architecture of FPGAs with memory modules. Thus, it is no longer necessary to communicate with elements outside the circuit which allows a significant gain in terms of access time. Two types of memory can be associated with an FPGA: Read-Only Memory (ROM) for immutable information and Random Access Memory (RAM) for variable information. These memories have different access modes (sequential or random) and access response characteristics (synchronous or asynchronous).
- Routing
  Routing elements are considered as the most important elements in FPGAs as they present the largest portion of the silicon consumed on the circuit chip. They are composed of different length segments that ensure the connection between different logic blocks. Their importance lies mainly in their ability to determine the critical paths and logic density of the system. That is why the routing presents a critical step in the development of an application on an FPGA.
- I/O pins
  These input/output elements are used to ensure communication between a circuit and its external environment.
- Clock
  The clock is responsible for the proper functioning of an electronic system. An FPGA circuit can receive one or more clocks and routing resources are suitable for transporting clocks over long paths. In order to have the same clock in the whole circuit, the FPGA circuit can create other clocks associated to the main clock at different frequencies.

SoC FPGA devices integrate both processor and FPGA architectures into a single device. Consequently, they provide higher integration, lower power, smaller board size, and higher bandwidth communication between the processor and the FPGA. They also include a rich set of peripherals, on-chip memory, an FPGA-style logic array, and high speed transceivers.

Integrating these technologies on the same die as shown in Figure 6.1 eliminates the cost and saves board space. If both the CPU and FPGA use separate external memories, it may also be possible to consolidate both into one memory device, for further savings. In

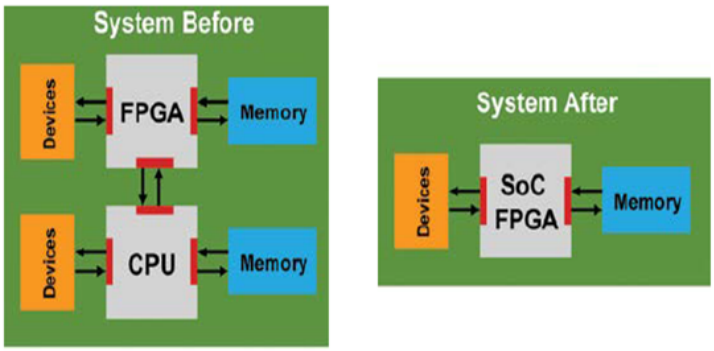

Figure 6.1 – Soc FPGA system

addition, communication between the processor and the FPGA consumes substantially less power compared to using separate chips. The integration of thousands of internal connections between the processor and the FPGA leads to substantially higher bandwidth and lower latency compared to a two-chip solution.
The most famous and experienced companies in SoC FPGA manufacturing are Xilinx [109] and Altera. Nowadays Intel has bought Altera to unite their work [110] and launch more sophisticated products such as the target platform device used in this work Arria 10 SoC FPGA.

6.3 Experimental setup and co-design tools

6.3.1 Hardware

Quartus prime is provided by Intel/Altera as a hardware designing tool that supports Arria 10 devices [94]. It allows the complete management of an FPGA design flow, from the architecture description to the implementation. Quartus adopts VHDL and Verilog as hardware description languages, visual editing of logic circuits, and vector waveform simulation.
To create a SoPC system that involves various I/O devices such as SRAM and SDRAM controllers, Direct Memory Access (DMA) controller, Quartus prime integrates a platform designer called Qsys [111]. This latter saves significant time and effort in the FPGA design process by automatically generating logic interconnections between intellectual property (IP) functions and subsystems. Qsys also provides a high abstraction layer to manage the different required interfaces. Moreover, Altera/Intel offers a specific tool for SoPC FPGAs named Embedded Development Suit (EDS) that provide further development tools, utility programs, and design examples.

6.3.2 Software

Embedded software is a computer program buried in an electronic system. It is an operating system/kernel that is specific to embedded devices and characterized by the following criteria:

- Targeted to the application field.
- Reliable and secure for completely autonomous operation.
- With almost infinite lifespan.
- Specific to the target architecture.
- Optimized and customized (size, execution time, etc.).

Angstrom is an embedded Linux that can be adopted for a large number of platforms and devices. In this chapter, we explain how the software driver is designed to establish the communication between Hard Processor System HPS (ARM) running the embedded software (Angstrom) and the hardware components of the Arria 10 SoC FPGA.

6.4 Co-design implementation of HEVC Inverse DCT-II

6.4.1 32 point IDCT hardware implementation

Figure 6.2 illustrates the unified 2D-IDCT Core transform architecture for all block sizes from 4x4 to 32x32. The main element is the unified 1D IDCT used for both horizontal and

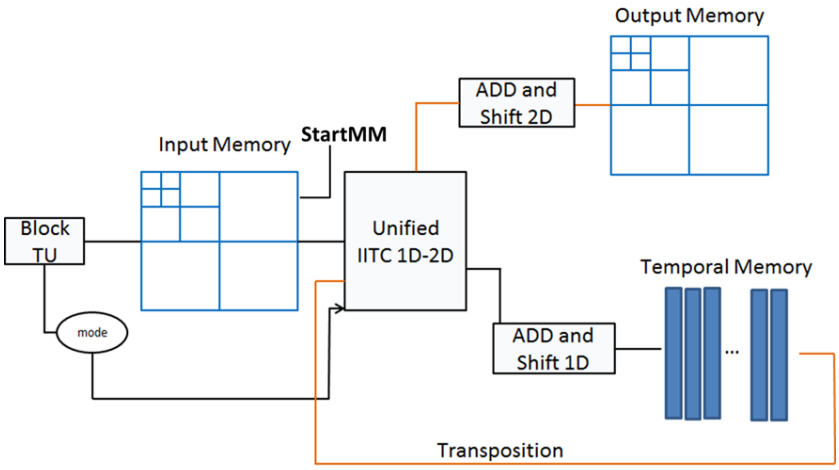

Figure 6.2 – General architecture of the 2D IDCT design

vertical transforms separated by several memory blocks. A control module is responsible for interconnecting all these components to ensure the process properly. As the decomposition of different matrices is recursive, a selection phase indicates weather if 32-point 1D or 16-point 1D transform is used. If the second condition is set (16-point), only the first 16 inputs

are considered and transferred to the even part of the 32-point architecture. The same control path is applied to select 8 or 4-point orders. This process is provided by multiplexers according to a definite selection referring to the desired transform block size. Else, all the 32 inputs are available and 32-point 1D transform is executed through even and odd parts. The whole architecture is pipelined in order to improve the performance of the design in terms of execution time. Figure 6.3 depicts the designed multiplexer adopted in the selection process between 4 and 8-point orders.

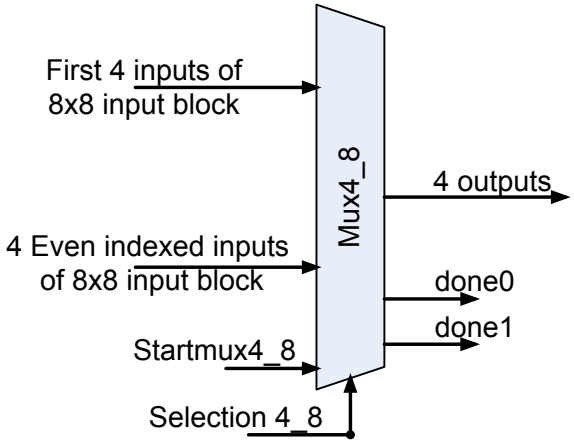

Figure 6.3 – Proposed architecture of the selection process multiplexer

If "selection4_8" = 0 then order 4 is set, otherwise, order 8 is chosen. The same principle is applied for 8/16 and 16-32 control processes. For the unified circuit, according to a “mode” decision input, the size of the transform is first chosen as the following:

- “00” refers to IDCT4 as all selection signals "selection4_8", "selection8_16" and "selection16_32" are set to 0.
- ”01” refers to IDCT8. As a result, only "selection4_8" =1 and the others are set to 0.
- “10” is for IDCT16. "selection4_8" and "selection8_16" are set to 1 and "selection16_32"=0 .
- “11” for IDCT32 where all selection signals are set to 1.

After that, data which have been already stored in the input FIFO are read 2 by 2. The number of read clock cycles signaled by "startMM" signal depends on the inverse transform size (Figure 6.2) . At each “start” signal given, the first dimension is calculated for each row of the IDCT block of an appropriate size. Once the outputs are obtained, they will be stored in intermediate FIFOs whose number varies according to the size of the desired transform. Next, to start the 2D processing, 1D outputs are read from the FIFOs while scanning the rows if the columns were first scanned and vice versa: that is to say the first inputs vector of the IICT contains the first values of each FIFO and so on. Indeed, the transposition of the values from the FIFOs is done by acting appropriately on the read signals of each FIFO.

Once the 2D-outputs are obtained and shifted, they are stored in the output FIFO 2 by 2 and finally, we proceed with the display. For the IDCT block, either for the 1^{st} dimension or the 2^{nd}, the process is the same. The difference lies in the size of the transform.

6.4.2 On chip Avalon Memory Mapped FIFOs

This FIFO is a good fit for our application as we intend to send and receive data to/from the main FPGA component (IDCT in our case). Memory Mapped (MM) FIFO consists of four main interfaces: "in", "insignal", "out" and "outsignal". Indeed, two FIFOs (input and output) will be used and placed on both sides of the IDCT component. "in" and "out" are data buses interfaces which will feed or receive data to/from the main FPGA IP (IDCT). Their respective width buses have to be compatible. "Insignal" and "outsignal" are reserved for the interface status. Set to 1, data transaction is enabled. Avalon Memory Mapped (Avalon MM) interfaces are dedicated to streaming high-speed data, reading and writing registers and memory, and controlling off-chip devices. These standard interfaces are included into the components available in the Platform Designer (Qsys). Avalon-MM is an address-based read/write interface typical for master-slave connections.

Qsys provides a certain flexibility for designers to adapt the width size of Avalon MM interfaces to their architecture needs. The target Arria 10 SoC device supports up to 256-bit width Avalon MM FIFO ports width. In this application, we test two different configurations of Avalon MM and analyse their impact in terms of time transfer. The input/output interfaces of the FIFO are Avalon slave interfaces that can be mastered by other master interfaces. ARM AXI is the master interface used in this application and explained in the next paragraph.

6.4.3 H2F bridge and H2F LW bridge

For data exchange between a hardware component and software drivers, Altera/Intel Qsys provides com-interfaces to carry data. H2Fbridge (HPS to FPGA bridge) and H2FLWbridge (HPS to FPGA lightweight bridge) are interfaces of type ARM AXI. AXI is part of ARM AMBA, a family of micro controller buses. The protocol simply sets up the rules for how different modules on a chip communicate with each other, requiring a handshake-like procedure. Such protocol is flexible to be configured in the Qsys communication environment and provides an effective medium to transfer data between the existing components on the chip.
As previously mentioned, Qsys manages in-between different interface connection by providing a high abstraction designing level. The interfaces are dynamic and can be adapted to the designer's needs [111]. The "H2Fbridge" supports up to 128-bit port width with a minimal

of 32-bit while the H2FLWbridge can only be set to 32-bit size. Figure 6.4 illustrates the system interconnect interface provided by Qsys tool.

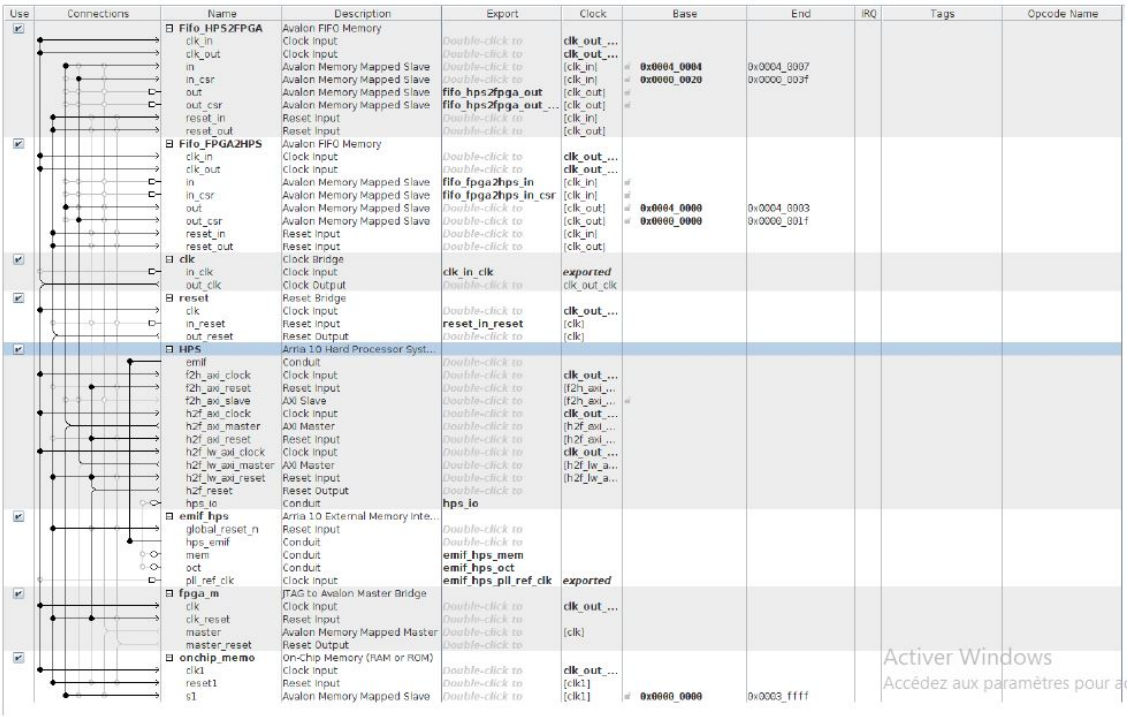

Figure 6.4 – Qsys System interconnect for the proposed design

Master-slave configuration of AXI and Avalon MM interfaces is necessary to ensure the hardware/software communication. H2Fbridge and H2FLWbridge are connected to Avalon-MM FIFO input/output and status signal (insignal/outsignal) interfaces, respectively. The bridges are used for a single task at a time. That is why the software driver is used to manage and control reading/writing processes separately.

6.4.4 State machine design

We propose a state machine design illustrated in Figure 6.5 to encapsulate all hardware components (both Avalon MM FIFOs and the hardware IP of IDCT). It is described in VHDL programming language and details all the processing steps.

The width of an Avalon MM FIFO depends on the com-bus size and its bus size can reach up to 128-bit with a minimal size of 32-bits. The depth and width of the MM FIFO are the reconfigurable parameters to adjust the communication. Changing the H2Fbridge bus forces the change of the FIFO's width. For this configuration, 32-bit wide com-bus is used. IICT32 is a unified architecture supporting transform block sizes from 4x4 to 32x32. Transform size is designed by a selection signal of 2bits indicating the desired size. It is considered as an

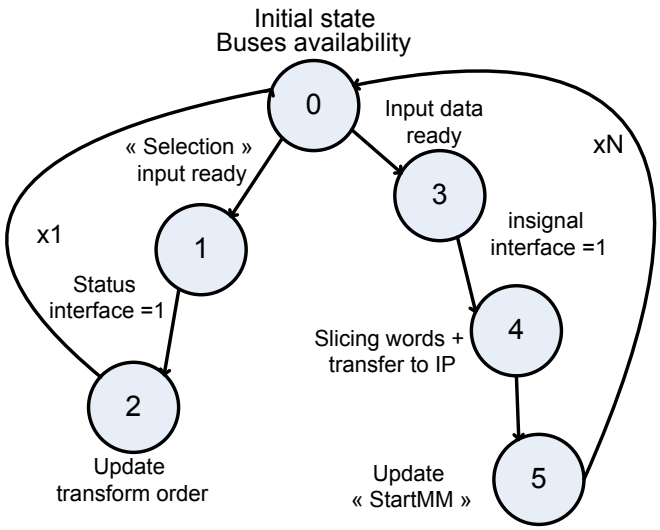

Figure 6.5 – State machine design for the proposed hardware component

input alongside with different matrix input values. Therefore, a third Avalon-MM FIFO, addition to the input/output FIFOs already mentioned above, is needed for the selection signal in. The different states are chronologically described as the following:

- State 0: checks if there is data in the FIFOs or not. If so, we move to state 1 for process. Furthermore, it checks if there is data inside the input FIFO. If so, we move on to the state three. If neither, we wait until new data comes to one of the FIFOs.
- State 1: sets "one" to the status interface of the selection FIFOs signaling that the bus is ready to carry more data. Then, we move on to the second state.
- State 2: The “selection” signal referring to the desired transform order (4, 8, 16 or 32) is updated. Knowing that there is no more data in the selection FIFO, the system goes back to state 0 and then moves to state 3.
- State 3: informs that input FIFO is ready to receive data (input block) by setting the “insignal” interface to “1”. Then we move on to state 4.
- State 4: As mentioned earlier, 32 bit com-bus is used. Note that IICT32 is also adjusted with two FIFOs placed in both sides with a 32 bit port width to exchange inputs and outputs 2 by 2 values (each on 16 bits). This state ensures first the slicing of every concatenated 32-bit word in two 16-bit registers and then feeding them to the hardware IP moving to the final state 5.
- State 5: For data to be completely used by the IICT32, its "startMM" signal (Figure 6.2) needs to be updated each time new input data (32 bits= 2 input values) is available. Therefore, state 5 updates the "startmm" signal and returns to state 0. Depending on the transform order selection, the number of words in MM FIFOs as well as “startmm” pulses are fixed. For instance if the selection is "00", IICT4 is executed. Thus, 8 words of 32-bit are required in the FIFO. To sum up, to perform N-point IICT, NxN input

block is required. As a result, regarding 32-bit com-bus, this process has to be executed NxN/2 times to complete the communication.

Arria 10 platform supports a larger com-bus configuration of 64 bits. This could evidently optimize the communication transfer-time. As mentioned above changing the com-bus width breed to a modified state machine design. Indeed, the same approach as 32-bit communication is adopted. The state zero is for verification, states one and two are for extracting data from the selection FIFO. All FIFOs are connected to a 64-bit wide com-bus. MM FIFOS are reconfigured to have 64 bit port width to enable storing 4 concatenated input values at a time (16 x 4 = 64 bits). As a result, states 4 and 5 witnessed some changes with respect to the previous structure as follows:

- State 4: remains always a state of slicing but to four 16-bit registers in this case.
- State 5: The first two inputs are fed to the IICT IP updating the “startmm” signal. In the same way, the remaining two inputs are sent to the IP so as “starmm” is updated again and returns to state 0. This process is executed NxN/4 times.

6.4.5 Software/Hardware communication

The driver is a C program written on the embedded Linux Angstrom. It is a driver that provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions with no need to know precise details about the hardware IP. The software is used to communicate with the hardware design via system buses (H2Fbridge and H2FlwBridge) connected themselves with IICT IP.
Drivers are hardware dependent and specific for each operating system. The embedded Linux distribution, Angstrom, consists of two main parts: the "address mapping" and "writing/reading". Addresses generated by the ARM processor are virtual addresses. When the MMU is enabled, it translates these virtual addresses into physical addresses. That is to say that code instructions or data can be accessed at a chosen virtual address, while the physical address is at a different location. Figure 6.6 illustrates a simple scheme for virtual to physical address mapping.

Quartus prime offers useful tools that can generate headers containing the addresses to be mapped. Using the "nmap()" of the library "sys/mman.h", FIFO addresses are provided to be used in the software that enables us to write or to read the FIFOs. For our application, the memory map of soft IP peripherals, as viewed by the microprocessor unit (MPU) of the Hard Processor System, starts at HPS-to-FPGA base address of $0xC000\_0000$ while for the Lightweight HPS-to-FPGA, it starts at the base address of $0xFF20\_0000$.
As mentioned earlier, the com-bus can only be used by one process at a time. The first step is to set the selection signal. Next is the concatenation of two "int16_t" integers type into one "int32_t" in case of 32-bit com-bus configuration. Input data is fed to the FIFOs sequentially

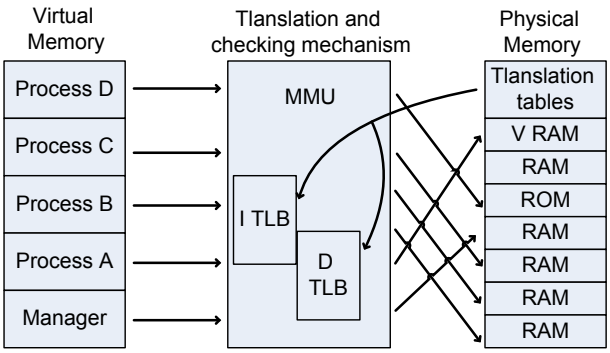

Figure 6.6 – Virtual to physical address mapping

and processed as described in the previous paragraphs. Once the outputs are available, the software ensures the reading phase using the same bridge. The status interfaces ("insignal", "outsignal") of the FIFOs are manually managed by the system in the hardware part.

6.5 Implementation on Arria 10 board

For this application, two ways of the Arria 10 SoC FPGA board are adopted:

- Quartus programmer tool:
  Following the hardware design synthesis, an SRAM Object File (SOF), the runtime file for FPGAs, is generated by the name of the project.

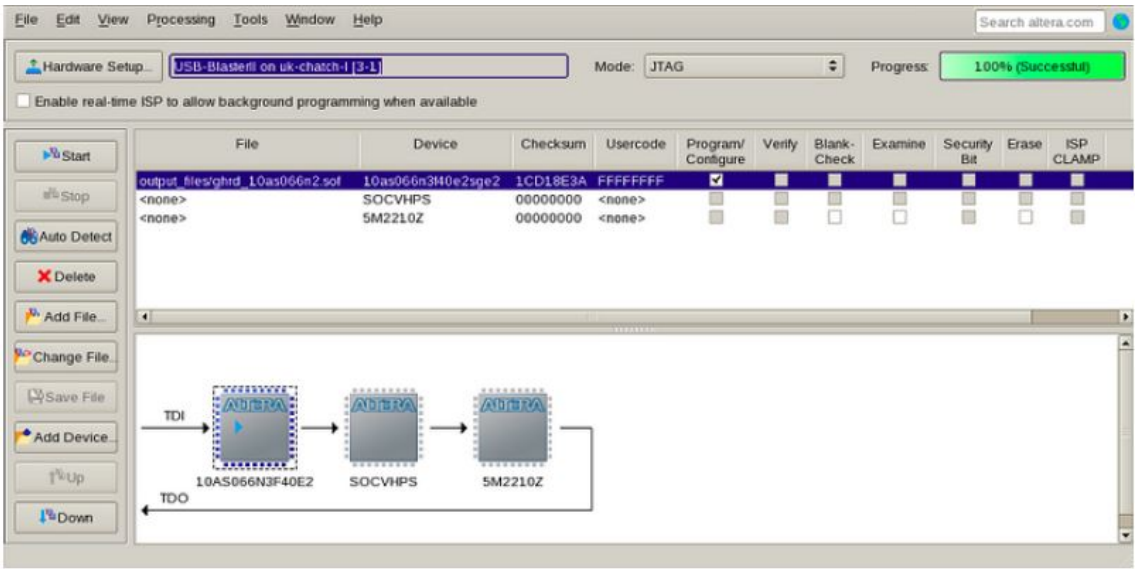

Figure 6.7 – Quartus programmer tool interface

Loading it (SOF file) into the board using the Quartus programmer tool will successfully program the FPGA gates as shown in Figure 6.7. Figure 6.8 presents a functional

simulation of the hardware state machine design using Model Sim tool [95] of Quartus prime. Bridges for hps_to_fpga communication are 64 bits wide buses and the transform selection is chosen to be "00" so as the hardware IP (IDCT) operates as IDCT4. Results are obtained sequentially two by two via "out1" and "out2" signals as shown in Figure 6.8.

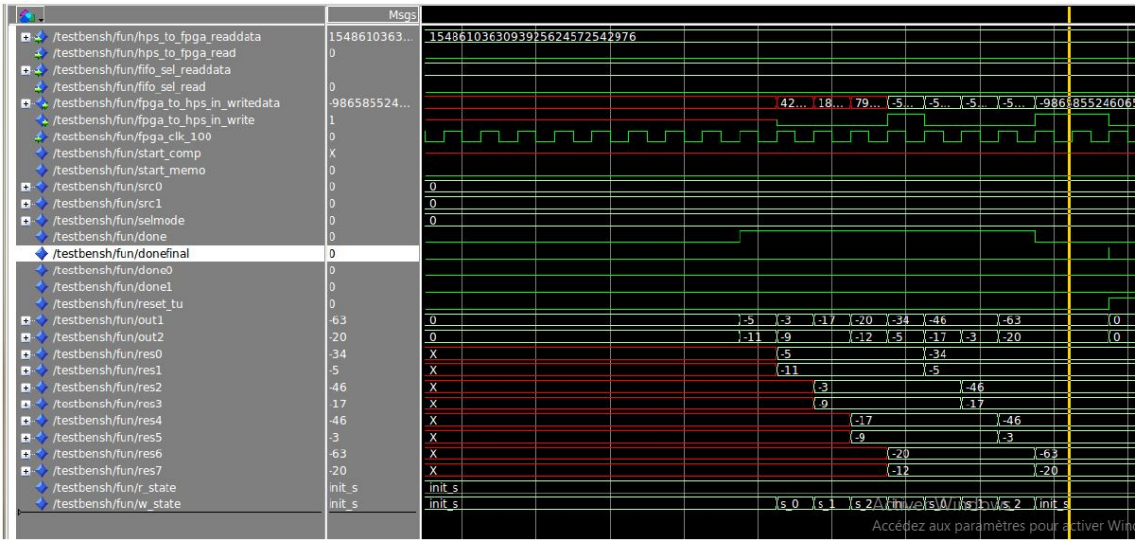

Figure 6.8 – Hardware design functional simulation using Model Sim

- Through HPS:
  The second method for Arria 10 configuration is through the HPS using the Row Binary File (RBF). This latter is obtained by conversion of the SOF output. Embedded Design Suit (EDS), offered by Intel/ Altera, is used to convert "sof" to "rbf" file generated after the design synthesis.
  It is a multifunctional tool enabling designer to convert files or generate headers through embedded shell commands. Once the row binary file is ready, it is copied in the first partition of the SD card containing the Angstrom distribution. Therefore, FPGA configuration is the first step when Linux boots up.

6.6 Implementation results and discussion

To evaluate the Co-design application, we have extracted software implementation of IDCT from the Open HEVC decoder developed in IETR lab. Open HEVC is designed to support multiple platforms and architectures such as x86 and ARMs and provides optimized software implementations.
For the transform module, the larger the input block is (4x4, 8x8, 16x16 or 32x32) the more complex and higher time-consuming the implementation would be. As a first step, an original

version of IDCT matrix multiplication is implemented using C software code in order to have an evaluation reference for the co design application. The time was measured using the library "time.h". Then, we have extracted the IDCT implementation dedicated for ARM architecture which benefits from specific intrinsic instructions in order to provide a more optimized implementation for IDCT. Table 6.1 shows the run time of different IDCT orders using the software C and ARM SSE implementation in microseconds. The time calculated is obtained as the average of 40 test bench examples for each order.

Table 6.1 – Run-time in microseconds for C software and ARM SSE implementations

	4x4	8x8	16x16	32x32
C software (μs)	2	13	87	440
ARM SSE (μs)	–	–	5	31

We can easily notice that moving from an order to a larger one, the run time implementation increase significantly especially for larger orders 16 and 32. For ARM SSE architecture, small orders do not show interesting timing reduction from normal software implementation. That is why only 16 and 32 IDCT orders are considered.
For the co-deign application there are several times to measure: the mapping time, the writing/reading times and the processing time (hardware IP). It is worth noting that the memory mapping is computed only once and is not considered here. Table 6.2 presents the different time consumption phases of the co design implementation for both com-bus configurations 32 bits and 64 bits.

Table 6.2 – Run-time in microseconds of 32-bit and 64-bit com-bus co-design application

	32 bits com-bus				64 bits com-bus			
	4x4	8x8	16x16	32x32	4x4	8x8	16x16	32x32
Writing (μs)	10	25	93	359	7	15	50	185
Reading (μs)	6	23	91	359	4	12	46	175
Processing (μs)	<5	<5	<5	<5	<5	<5	<5	<5
Total (μs)	21	33	189	713	16	32	101	365

The same observation is noticed regarding run time increase with larger input block from 4x4 to 32x32. The hardware implementation of IICT provides a significant run time acceleration (processing time) benefiting from DSP blocks and pipelined architectures. Nevertheless, for co-designing presents a bottles-neck which lies in transfer data and communication time using Memory Mapped FFIO bridges.
Although switching from 32 bit to 64 bits com-sub leads to about 50% run time reduction, the whole co-design application still presents much higher time consumption with respect to

ARM SSE implementation. Multiple ways of optimization can be adopted to further improve the HPS and FPGA communication:

- Switch to 128 bits com-bus as it is supported by Arria 10 Board. Therefore, following the same principle of changing 32 bis com-bus to 64 bits, that would further reduce the run time by half.
- Adjust the hardware IP to be able to receive and display data 4 by 4 or even 8 by 8 input/output values instead of 2 by 2 (actual). That would reduce the time consumed by the slicing operation (see Figure 6.5 state 5).
- Using more optimized tools for memory access such as DMAs.

6.7 DMA based approach for Co-design optimization

Direct Memory Access (DMA) is a feature of most computers that allows certain hardware submodules accessing the memory for reading and writing. The interaction between the subsystems and the memory is carried out independently of the CPU. The main advantages of DMA are providing high transfer rates with fewer CPU cycles for each transfer and offering the freedom to HPS to process other tasks while the transfer is in progress. DMA controller, AXI DMA and AXI lightweight buses interfaces are required in DMA communications to initiate and set up the data transfer by sending the starting address, number of words and direction of the data transfer. Next, an interrupt signal alerts the HPS once the operation is finished.

The objective of this work is to optimize the communication time of the co-design application. Table 6.3 illustrates a comparison of DMA and Avalon MM communication approaches for 8x8 DCT-II transform implementation.

Table 6.3 – Run-time comparison of DMA and Avalon MM communication interfaces for 8x8 DCT-II application

	64-bit com-bus - 8x8 DCT-II	
	DMA	Avalon MM FIFO
Writing (μs)	3.45	15
Reading (μs)	3.38	12
Processing (μs)	<5	<5
Total (μs)	11.83	32

It can be noticed that communication time which is considered as the bottles-neck of Co-designing is reduced by 4 to 5 times compared to the previous configuration (Avalon MM FIFO). As a result the total execution time supposing that the processing time is unchanged is reduced by 3 times from 32μs to 12μs. It is worth noting that the hardware IP is also

adjusted to be able to receive and display data 4 by 4.
To further benefit from DMA, several input blocks can be sent simultaneously to be processed sequentially in order to preserve the data transfer time. We have tested respectively to send 4 and 16 input 8x8 blocks to be processed and then compute the average of transfer time per block.

Table 6.4 – Run-time in microseconds of DMA communication interface for multiple input blocks processing

	64-bit com-bus - 8x8 DCT-II- DMA	
	4 input 8x8 blocks	16 input 8x8 blocks
Writing (μs)	1.17	0.36
Reading (μs)	1.15	0.35
Processing (μs)	<5	<5
Total (μs)	7.3	5.7

Results presented in Table 6.4 shows that the more data sent to be processed avoiding multiple transfer transactions, the more communication time reduction is obtained. Writing and reading run time witnessed a significant reduction allowing an efficient co design implementation.

6.8 Conclusion

In this chapter, a co-design application for transform module is developed. The target device is the Intel Arria 10 SoC FPGA. It is a reconfigurable platform equipped with several interesting software and hardware features that facilitate such application for video compression. An overview of the Arria 10 board is provided in this chapter. The hardware IP used in this application is the inverse DCT-II module used for HEVC. Creating the software/hardware communication requires a software driver and the communication offered interfaces to manage and ensure the data exchange between the ARM processor and the hardware IP. The implementation process benefited from hardware and software designing tools of the Arria 10 development kit. The co-design circuit is evaluated in terms of execution time where we confirmed that data transfer time is the bottleneck for such applications. Different com-bus configurations (32 and 64-bit) are tested in order to reduce the execution time and alternative optimization method using DMA interfaces has been investigated.

Chapter 7

Hardware implementation of Low Frequency Non Separable Transform

This chapter presents the new coding tool, Low Frequency Non Separable Transform, related to transform process and recently incorporated in the VVC standard. An overview and an efficient hardware implementation are provided.

Contents

7.1 Introduction

In traditional image and video coding schemes, separable transforms are typically employed due to their low-complexity implementations. However, the compression efficiency of separable transforms is limited for most natural image/video blocks which generally have arbitrarily directed edge and texture patterns. It is well known that non-separable transforms can achieve better compression efficiency for directional texture patterns, yet they are computationally complex, especially for larger block sizes. In order to achieve higher transform coding gains with relatively low-complexity implementations, in this chapter, a background on the non-separable transforms for video coding is presented in section 7.2. Later on, a brief description

of non-separable secondary transform coding tool history throughout the next generation video coding standardization is described. The proposed hardware implementation of the Low Frequency Non Separable Transform module is detailed and discussed in 7.3. Finally, section 7.4 concludes this chapter.

7.2 Background of the non-separable transforms for the next generation video coding

It is well known that non-separable transform is usually more efficient than separable transform for exploring the inter-pixel correlation of 2-D directional texture patterns. However, considering the non-separable transform approach for a practical video codec design has been always a questionable decision because of the following two reasons:

- Heavy encoder and decoder complexity burden
- Excessive memory requirement for storing the non-separable transform matrices.

Consider performing a transform on an NxN residual block using matrix multiplication, the memory requirement and operation counts for separable transform would be $O(N^2)$ and $O(N^3)$, respectively, while for non-separable transform, it becomes $O(N^4)$ and $O(N^4)$, respectively [112].
To apply non-separable transform on a 2-dimensional NxN input prediction residual block X denoted as below:

$$\begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3} & \dots & X_{1,N} \\ X_{2,1} & X_{2,2} & X_{2,3} & \dots & X_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{N,1} & X_{N,2} & X_{N,3} & \dots & X_{N,N} \end{bmatrix} \tag{7.1}$$

The input residual block X is first stretched to a colomn vector X with length as N^2, as shown below:

$$\begin{pmatrix} X_{1,1} \\ X_{1,2} \\ \vdots \\ X_{N-1,N} \\ X_{N,N} \end{pmatrix} \tag{7.2}$$

Then the non-separable transform is applied by performing the following calculation:

$$Y = H \cdot X, \tag{7.3}$$

where Y indicates the transform coefficient vector derived after applying non-separable transform, and H is an N^2xN^2 transform matrix. Finally, the elements in the transform coefficient vector Y are re-organized as an NxN block such that the subsequent block-based entropy coding can be efficiently applied. The re-organization of the coefficient vector is processed in a way that the coefficient with larger magnitude will be scanned first, which is aligned with the philosophy of scanning order design.
To reduce the complexity of non-separable transform as well as capturing most of its coding gain, non separable transform is applied as a secondary transform stage. As a result, it is an additional process between the primary transform (separable) and quantization processes. In case that the primary transform do not sufficiently de-correlate the residual samples, a secondary transform is further applied upon the primary transform coefficients to further reduce the statistical redundancy. The significant increase of computation and memory complexity for larger transform sizes makes using 16x16 or larger block transforms practically infeasible, although they would yield high coding gains especially for large resolution video contents (e.g., 1080p and 4K). Therefore, the secondary transform is applied only on the low-frequency components, e.g., top-left 4x4 or 8x8 [112].

7.2.1 History of LFNST coding tool throughout the VVC standardization

Since the official release of the HEVC standard, there have been efforts trying to bring non-separable transform on top of HEVC where up to 12% coding gain has again been validated for Intra coding [113]. In the early stage of the new video coding standard development, for the 3 first versions of JEM, there were in total 11×3 (for directional modes) and 1×2 (for non-directional modes) non-separable transform matrices, where 11 is the number of transform sets for the directional intra prediction mode and each transform set includes 3 transform kernels [52, 53, 54]. While for non-directional modes, i.e., Planar and DC, only one transform set is applied including 2 transform kernels. The mapping from the intra prediction mode to the transform set is given in Table 7.1. The transform set applied to luma/chroma transform coefficients is specified by the corresponding luma/chroma intra prediction modes, according to Table 7.1.

In JEM4.0, a new Hypercube-Givens Transform (HyGT) with butterfly implementation is used instead of following the conventional technique of full matrices multiplications (that yields to better compression efficiency). This new class of transforms is defined to search for the best transform parameters in order to provide low complexity implementations that are easy to parallelize [55]. Table 7.2 gives the coding and complexity performance is terms of both BD-R and run time of the HyGT non separable transform implemented on top of JEM3 [114].

Table 7.1 – Mapping from intra prediction mode to transform set index

Intra Mode	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Set index	0	0	1	2	1	2	1	2	3	4	3	4	3	4	5	5	5	6
Intra Mode	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	–
Set index	6	6	7	7	7	8	9	8	9	8	9	10	11	10	11	10	11	–
Intra Mode	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**
Set index	10	11	10	11	10	9	8	9	8	9	8	7	7	7	6	6	6	5
Intra Mode	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	–	–	–	–
Set index	5	5	4	3	4	3	4	3	2	1	2	1	2	1	0	–	–	–

Table 7.2 – Performance (%) in terms of BD-R and run time complexity of HyGT non separable transform [114]

Class	All Intra Main 10					Random Access Main 10				
	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*
A1	-0.63	-0.7	-0.6	99	102	-0.43	0.0	-0.29	99	99
A2	-1.87	-2.4	-2.1	97	98	-1.18	-2.11	-1.72	100	99
B	-0.69	-0.8	-0.7	99	102	-0.41	-0.62	-0.59	100	99
C	-1.38	-1.2	-1.1	98	99	-0.72	−0.48	-0.64	100	99
D	-0.73	-0.6	-0.2	102	101	-0.34	0.04	-0.12	103	102
E	-1.25	-0.8	-0.7	97	95	–	–	–	–	–
Av.	-1.07	-1.1	-0.9	99	100	-0.61	-0.63	-0.66	100	100

In the next versions of JEM the same approach using HyGT is used for non-separable secondary transform only with some changes in notations: There are totally 35×3 non-separable secondary transforms for both 4x4 and 8x8 block size, where 35 is the number of transform sets specified by the intra prediction mode, denoted as set, and 3 is the number of NSST candidate for each Intra prediction mode [13]. For intra prediction modes larger than 34 (diagonal prediction direction), the transform coefficient block is transposed before/after the secondary transform at the encoder/decoder.

The JEM tried to incorporate all possible new coding tools able to provide a significant coding efficiency with respect to the HEVC performance. Indeed, it presented up to 30% bit rate reduction. However, this was equipped with a huge complexity level that can no longer be neglected especially for industrial companies and real time implementations. Therefore, for the next phase of the new standard establishment, computational complexity

and resource requirements are taken into consideration along to the bitrate gain. As a result, some tools of the JEM are modified in order to reduce the encoder complexity such as the Multiple Transform Selection which includes currently only three transform types instead of 5 as mentioned in the previous chapters. Some other tools such as non-separable secondary transform are no longer considered in the standard. This decision is due to the high complexity and memory resource requirements to store and select all the transform matrices used in NSST process. While the first three drafts of VTM are established and deprived of NSST, researchers did not give up on this interesting coding tool that provide a good coding efficiency until it has showed up again in VTM5 and currently is incorporated in the VVC standard with a reduced computational complexity as Low Frequency Non Separable Transforms (LFNST) [59].

7.2.2 Description of the Low Frequency Non Secondary Transforms in the VVC Standard

In VVC, LFNST which is known as reduced secondary transform, is applied between forward primary transform (only when DCT-II is selected) and quantization (encoder) and between de-quantization and inverse primary transform (decoder) as shown in Figure 7.1 [59].

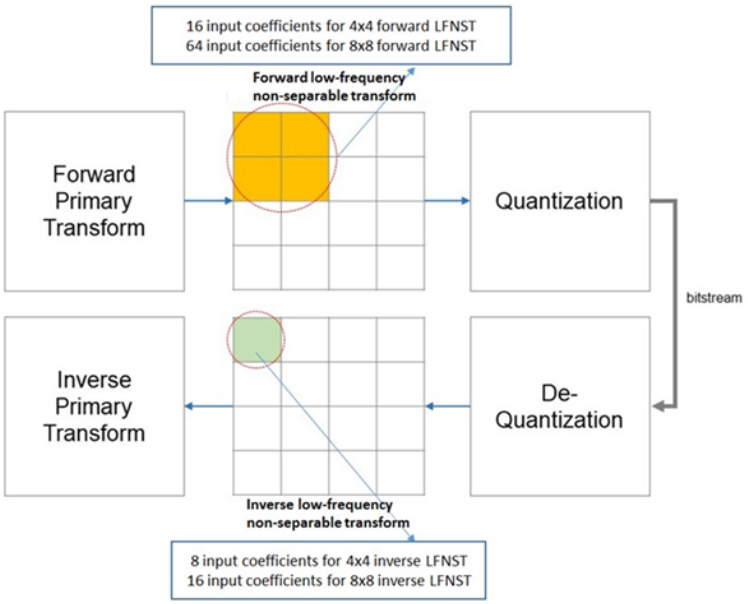

Figure 7.1 – Low-Frequency Non-Separable Transform (LFNST) process

In LFNST, 4x4 non-separable transform or 8x8 non-separable transform is applied according to the block size. For example, 4x4 LFNST is applied for small blocks (i.e., min (width, height) < 8) and will be referred as "LFNST4" and 8x8 LFNST is applied for larger blocks (i.e., min (width, height) > 4) referred as "LFNST8". LFNST is based on straightforward

matrix multiplication approach to apply non-separable transform so that it is implemented in a single pass without multiple iterations. However, the non-separable transform matrix dimension is reduced to minimize computational complexity and memory space to store the transform coefficients. Therefore, reduced non-separable transform (or RST) method is used in LFNST. The main idea of the reduced non-separable transform is to map an N (N is commonly equal to 64 for 8x8 NSST) dimensional vector to an R dimensional vector in a different space, where N/R (R < N) is the reduction factor. Hence, instead of NxN matrix, RST matrix becomes an R×N matrix as follows:

$$H_{R\times N} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \dots & h_{1,N} \\ h_{2,1} & h_{2,2} & h_{2,3} & \dots & h_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{R,1} & h_{R,2} & h_{R,3} & \dots & h_{R,N} \end{bmatrix} \tag{7.4}$$

where the R rows of the transform are R bases of the N dimensional space. The inverse transform matrix for RT is the transpose of its forward transform. For LFNST8, a reduction factor of 4 is applied, and 64x64 direct matrix, which is conventional 8x8 non-separable transform matrix size, is reduced to16x48 direct matrix as shown in Figure. 7.2 . Subsequently, the 48×16 inverse RST matrix is used at the decoder side to generate core (primary) transform coefficients in 8×8 top-left regions.

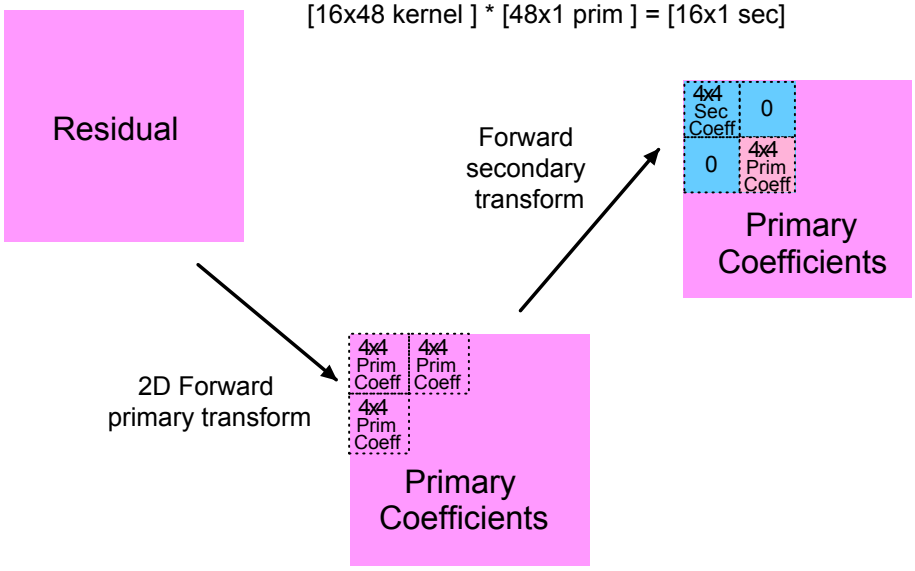

Figure 7.2 – Reduced Secondary Transform with 16x48 kernels

When16x48 matrices are applied instead of 16x64 with the same transform set configuration, only 48 input data from three 4x4 blocks in a top-left 8x8 block, excluding right-bottom 4x4 block, are considered in LFNST. With the help of the reduced dimension, memory usage for

storing all LFNST matrices is significantly reduced with reasonable performance drop. Table 7.3 gives the coding and complexity performance is terms of both BD-R and run time of the reduced non separable transform (RST) using 16x48 matrices implemented on top of VTM4 [115].

Table 7.3 – Performance (%) in terms of BD-R and run time complexity of reduced non-separable transform (RST) [115]

Class	All Intra Main 10					Random Access Main 10					Low Delay B Main 10				
	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*
A1	-2.03	-0.83	-0.93	123	99	-1.23	-1.44	-1.85	115	99	–	–	–	–	–
A2	-1.20	0.38	0.70	127	100	-0.57	-0.66	-0.71	109	99	–	–	–	–	–
B	-0.96	-0.26	-1.04	124	100	-0.56	-1.48	-2.43	110	100	-0.21	-1.11	-1.29	109	100
C	-1.07	0.05	-0.48	124	95	-0.49	-0.44	-1.0	110	99	-0.27	-0.14	-0.66	109	99
E	-1.43	-1.29	-2.16	117	99	–	–	–	–	–	-0.20	-0.71	-0.78	107	99
Av.	-1.28	-0.35	-0.8	123	98	-0.68	-1.03	-1.59	111	99	-0.23	-0.71	-0.78	107	99

In order to reduce complexity, LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant. Hence, all primary-only transform coefficients have to be zero when LFNST is applied. This allows a conditioning of the LFNST index signalling on the last-significant position, and then avoids the extra coefficient scanning in the current LFNST design, which is needed to check for significant coefficients at specific positions only [59].
There are totally 4 transform sets and 2 non-separable transform matrices (kernels) per transform set are used in LFNST.

Table 7.4 – Transform Selection Table [VTM6]

IntraPredMode	Set Index
Mode < 0	1
0<=Mode>1	0
2<=Mode>= 12	1
13<=Mode>=23	2
24<=Mode>=44	3
45<=Mod>=55	2
56<=Mode>=80	1
81<=Mode>=83	0

The mapping from the intra prediction mode to the transform set is predefined as shown in Table 7.4. For each transform set, the selected non-separable secondary transform candidate is further specified by the explicitly signalled LFNST index. The index is signalled in a bit-stream once per Intra CU after transform coefficients.

7.3 Proposed hardware implementation of LFNST

LFNST process currently consists in 4 sets including each 2 transform matrices. That is to say 8 matrices of size 16x16 for LFNST4 and another 8 matrices of size 48x16 for LFNST8 to compute larger block sizes. Although computation cost has been significantly reduced, it is still high consuming resource especially for hardware implementations. Proceeding by straightforward matrix multiplication using 16x16 matrices would surely require more logic resource than the offered by the target FPGA arria 10 SoPC, and we have already faced this problem in primary transform module implementation. Moreover, 16x16 matrices are dedicated only for LFNST4 as mentioned earlier. For LFNST8, 48x16 matrices are used which would lead to even more logic and memory requirements.

The proposed implementation approach consists in storing all involved matrices in ROM blocks offered by the FPGA platform. Every 16x16 matrix is stored in a ROM where every

16 coefficient-row are concatenated in one word. That is to say that the ROM is customized to have 16 words (4-bit address width) and 128-bit data port width (16 x8).

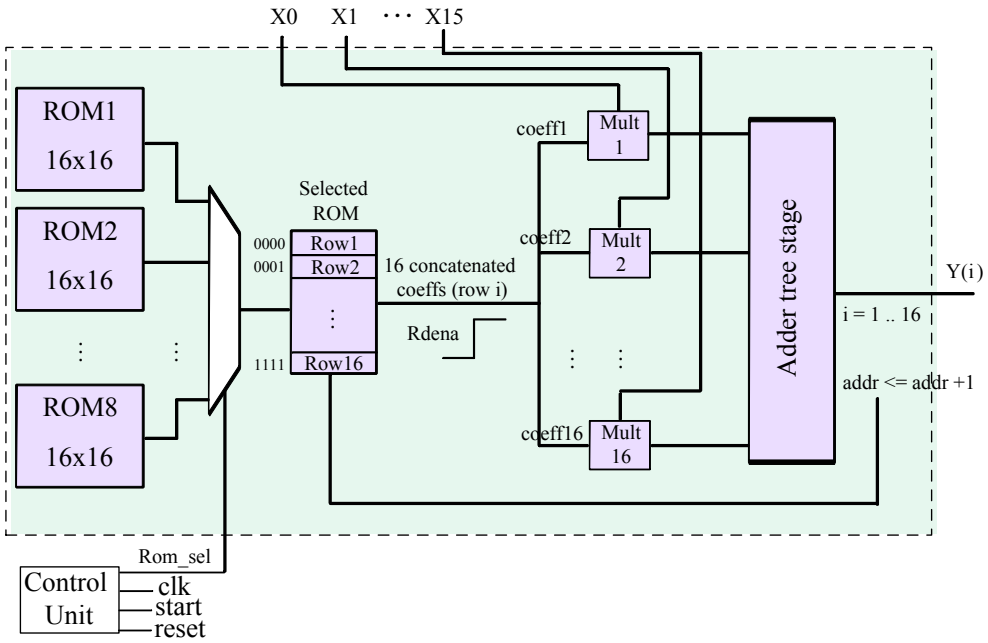

Figure 7.3 – Hardware architecture of LFNST process for small block size (i.e., min (width, height) < 8)

Figure 7.3 illustrates the proposed hardware architecture of LFNST process for small size using 16x16 multiplication matrices. 16 multipliers are implemented through LPM Core IPs using DSP blocks for the multiplication operations to further preserve the logic use. Adder tree is placed then to compute the output result. A selection dependent process is adopted to define the appropriate matrix using a multiplexer placed at the beginning of the design. Setting "rdena" to 1 leads to read the 16 row coefficients that will be dissociated as inputs to the 16 multipliers along with 16 input vector values. Once the output result is available, the process is repeated as many times as the number of the ROM words (16 in this case) sharing the same hardware resource. To sum up, only 16 multipliers are used for non constant coefficient multiplications and a ROM memory blocks to store the associated matrices are needed for the LFNST4 process.

The same principle is applied for LFNST8. As mentioned in the previous section, 16x48 matrices are used. The architecture design methodology is unchanged. All matrices are stored in ROM blocks in order to preserve the hardware logic resource. Indeed, ROM memory blocks are customized as follows: every 48 coefficient row (of 16x48 matrix) is concatenated in a single word. The number of ROM words is the same (16 i.e 4-bit address width) but with an 384-bit data port width (48x8). As a result, 48 multipliers, implemented through LPMs, are used at once for the multiplication operations followed by adder tree stage to

provide the output result. LPM Core IPs are configured weather to use or not DSP blocks to further preserve the logic use. Finally, a definite state machine manages to loop this process to compute all 16 output values as the final result of the LFNST8 module.

7.3.1 Implementation results and discussion

The reduced LFNST is recently incorporated in the latest VVC drafts. In our work, we propose a very low cost hardware implementation of LFNST4 and LFNST8. Table 7.5 and Table 7.6 present the implementation results of LFNST4 and LFNST8 using 16x16 and 16x48 multiplications matrices, respectively.

Table 7.5 – Synthesis results of the proposed LFNST4 architecture

	(+) DSP (+) ROM	**(+) DSP (-) ROM**	**(-) DSP (+) ROM**	**(-) DSP (-) ROM**
Alms	444 (<1%)	871(<1%)	1223 (<1%)	1814 (<1%)
Registers	827 (<1%)	1533 (<1%)	962 (<1%)	1917 (<1%)
DSP	12 (<1%)	12 (<1%)	0	0
ROM	16 Kb (<1%)	0	16 Kb (<1%)	0
Cycles	66	66	66	66
Frequency	306 Mhz	336 Mhz	291 Mhz	300 Mhz

Table 7.6 – Synthesis results of the proposed LFNST8 architecture

	(+) DSP (+) ROM	**(+) DSP (-) ROM**	**(-) DSP (+) ROM**	**(-) DSP (-) ROM**
Alms	1091 (<1%)	1789(<1%)	2769 (1%)	3030 (1%)
Registers	1669 (<1%)	3183 (1%)	1825 (<1%)	3901 (1%)
DSP	36 (<1%)	36 (<1%)	0	0
ROM	48 Kb (<1%)	0	48 Kb (<1%)	0
Cycles	82	82	82	82
Frequency	304 Mhz	312 Mhz	307 Mhz	309 Mhz

Tables 7.5 and 7.6 give the results for different design configurations involving inferred memory and DSP blocks offered by the target FPGA platform Arria10. It can be noticed that the proposal provides very moderate computational complexity whether DSP block and/or inferred memories are used or not. This is due essentially to the non-constant coefficient multiplication approach adopted leading to the hardware sharing of multipliers (16 for LFNST4 and 48 for LFNST8).

It is logical that using DSPs blocks and inferred memories provide more optimized implementation especially in terms of logic resource but the most important noticed observation is that all configurations do not exceed 1% of the offered hardware resource. Evidently, LFNST8

requires more resource than LFNST4 as it includes more input signals (48), larger multiplication matrices (16x48) and more multipliers (48). As a result, additional intermediate signal and routing is required. From Table 7.5, LFNST4 computation requires 66 clock cycles detailed as shown in the timing diagram of Figure 7.4.

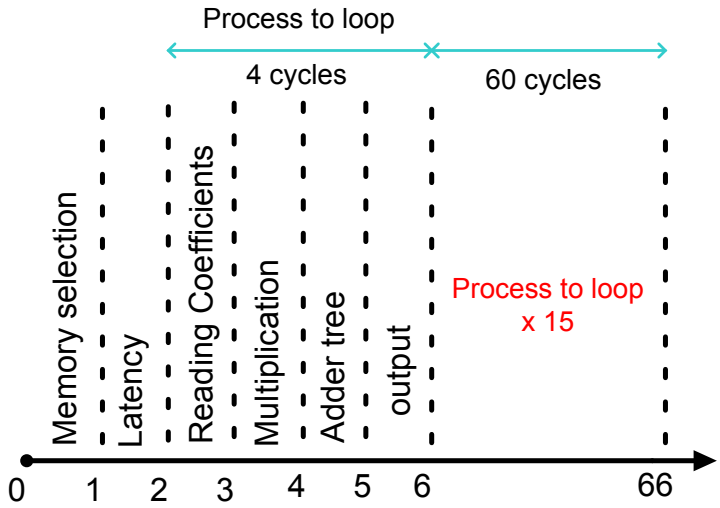

Figure 7.4 – Timing diagram for LFNST4 computation

However, for LFNST8, 82 cycles are needed to compute all output results. The difference lies in adding one clock cycle in adder tree stage compared to LFNST4 architecture. As a result, comparing to the diagram illustrated in Figure 7.4, the number of clock cycles required for LFNST8 is 82 = 7 + 5 x 15 (Table 7.6).
Due to the moderate hardware cost of the proposed architectures, processing time can be further optimized by parallelizing more multiplication operations at once. As a result, multiplying the number of multipliers used by 2 or 4 would reduce the processing time by almost 50% and 75% with very low additional hardware logic use. This would generate some changes in the proposed architecture. Figure 7.5 presents the hardware architecture of LFNS4 computation using 64 multipliers and 4x64 memory ROMs.

Compared to the diagram of Figure 7.4, the reduction factor of execution time lies in decreasing the number of loops. As a result, the clock cycles required for LFNST would be 18 = 6 + 4 x 3. The same principle can be applied for LFNST8 where the required clock cycles would decrease from 82 to 22 = 7 + 5 x 3. Several other optimization alternatives can be adopted while searching for the best trade-off between execution time and hardware constraints. Up to the best of our knowledge, this is the first hardware implementation of the Low Frequency Non Separable Transform for the next video coding standard VVC.

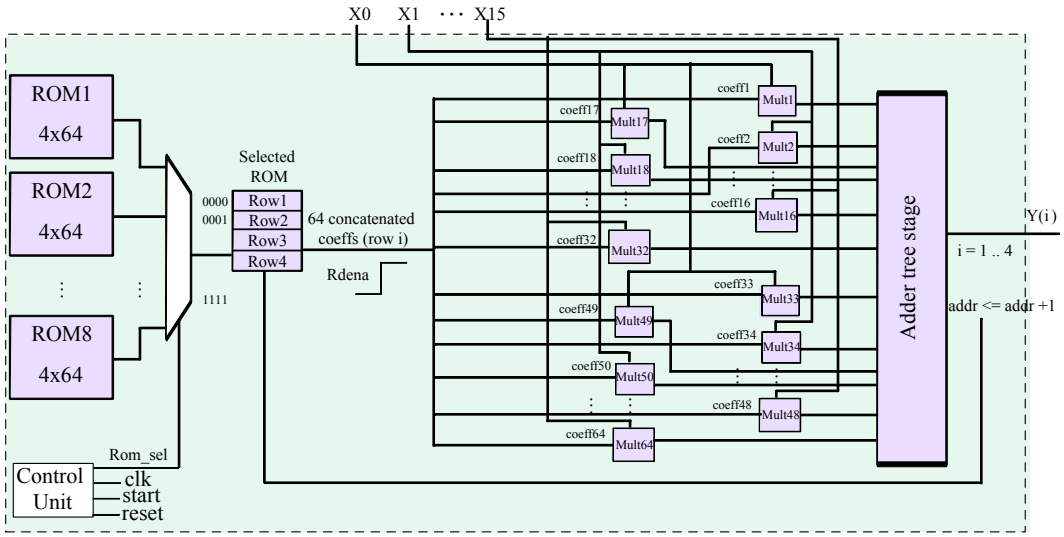

Figure 7.5 – Modified hardware architecture of LFNST4 process using 64 multipliers

7.4 Conclusion

This chapter interested in the new transform coding tool, Low Frequency Non Separable Transform, recently introduced in the latest VVC versions. It is applied as a secondary transform process to further de-correlate the residual samples and improve the coding efficiency. The LFNST is overviewed in this chapter describing its principle and its evolution throughout the standardization process. Despite the high complexity level of its design, an efficient and very moderate hardware cost architecture of LFNST is proposed in this chapter supporting all block sizes. Up to the best of our knowledge, the proposal presents the first hardware implementation of LFNST in the literature. Our approach is based on non-constant coefficient multiplication using memory block to store all the multiplication matrices (sized of 16x16 and 48x16) in order to preserve the logic use and a few number of LPM multipliers. The implementation results showed that the logic cost did not exceed 1% of the offered hardware resource for different design configurations. Moreover, alternative execution time optimization approaches are provided in this chapter.

Chapter 8

Conclusions and Perspectives

Contents

8.1 Conclusions

Video compression has been a crucial necessity in our life because of the worldwide video contents invasion and the increasing of its numerical size. This continuously raises the need for high-performance video-compression technologies enabling to reduce the amount of data to be transmitted or stored by compressing the input video signal into a bitstream file. Improving the coding efficiency was always one of the crucial issues of various compression standards that aim to get the most compact representation of the reconstructed video, with a high subjective quality. As a result, lately, every video coding standard has led to 50% bit rate reduction compared to its predecessor as a priority on top of its objectives, especially for modern video coding standards AVC, HEVC and recently VVC. Nevertheless, parallel to the bitrate reduction, it is unavoidable that a better coding efficiency is reached at the expense of higher complexity and requires more processing power as former codecs. For instance, HEVC is 3 to 5x more complex than AVC. However, VVC complexity is expected to be 7 to 10x higher than HEVC standard.

The brief description of state of art HEVC in chapter 2 ends up with an overview of the future video coding standard's most important features that enable to provide better coding efficiency beyond the one achieved by HEVC. This thesis focused on transform coding tools adopted in modern coding standards especially VVC. A background on the transform module is given, from DCT-II used in HEVC to the new concept introduced in the future video coding standard VVC. Indeed, it is based on putting in competition the DCT-II with other transform types of DCT/DST family and selecting the best one to apply providing better

coding performance in terms of RD-cost. Chapter 3 detailed the evolution of multiple transform concepts adopted throughout the standardization process. On the other hand, the new transform is equipped with significant computation and resource requirements increase. That is where lies the motivation of our work to provide some hardware acceleration through efficient hardware implementations. We will sum up the main contributions of this thesis as the following:

1. Propose, in chapter 4, an efficient pipelined hardware implementation of the 2D AMT including the five transforms types of sizes 4x4, 8x8, 16x16 and 32x32. This implementation is considered as the first proposal of 2D AMT implementation for VVC in the literature. The proposed design methodology takes advantage of the internal software/hardware resources of the target FPGA device *Arria 10* such as Library of Parameterized Modules (LPM) core IPs [19] and Digital Signal Processing (DSP) blocks, aiming to reduce the logic utilization. A unified 2D architecture embeds all 1D 4x4, 8x8, 16x16 and 32x32 transform modules taking into account all asymmetric 2D block size combinations and is able to sustain 2K video coding at 50 frames per second with an operational frequency up to 147 MHz.
2. Due to MTS high complex design, an approximation approach is proposed to reduce the computational cost of the DST-VII and DCT-VIII in chapter 5. The approximation consists in applying adjustment stages, based on sparse block-band matrices, to a variant of DCT-II family mainly DCT-II and its inverse. The optimal coefficients of the adjustment matrices are derived using a genetic optimization algorithm. Moreover, an efficient hardware implementation of the forward and inverse approximate transform module that can be integrated in both hardware VVC encoder and decoder is proposed. The architecture design includes a pipelined and reconfigurable forward-inverse DCT-II core transform as it is the main core for DST-VII and DCT-VIII approximations along with additional adjustment stage at low computational complexity and logic resource allocation.

   In terms of coding efficiency, the approximate DST-VII and DCT-VIII preserve the coding gain of the MTS. On the other hand, the proposed unified hardware architecture enables to reach a high frame rate while using a moderate hardware and logic resource of the *Arria10* FPGA device. It enables to process a video in HD and 4K resolutions at 386 and 96 fps, respectively.
3. A new coding tool named as Low Frequency Non Separable Transforms (LFNST) is recently incorporated in the VVC standard which is strongly related to the transform process to further improve the coding efficiency. In chapter 7, first a study of the LFNST is provided focusing on its evolution throughout the standardization process until the current VVC draft. An efficient and optimized hardware implementation of LFNST is proposed based on non constant coefficient multiplication using ROM

memory blocks and LPM Core IP multipliers. The proposed hardware architecture requires very moderate logic cost for the different design configurations. An alternative execution time optimization approach is also presented that enables to significantly reduce processing time with negligible increase in hardware resource requirements. Up to the best of our knowledge, it is the first hardware implementation in the literature for VVC LFNST module.

To conclude, this document presents a set of proposals that has the final purpose of reducing the high complexity level introduced by the transform module, essentially for the VVC standard. Coping and adapting with the ongoing standardization process of the VVC was the most interesting challenge that we have faced in this thesis. This latter tackles the real time implementation problem of the different Discrete Cosine/Sinus Transforms involved throughout the evolution of VVC standard, from AMT (5 types) in his first drafts to MTS (3 types) in the current version. The main objectives are reducing the computational complexity and achieving high frame rate processing for 4K videos taking into account the hardware resource constraints and limitations (memories, logic use, operational frequency...). We have successfully been among the first participants in scientific community proposing efficient hardware implementation of VVC transform coding tools enabling high frame rate performance. At the same time, we progressively tried to optimize the hardware logic use in our contributions.

8.2 Perspectives for future works

In chapter 4, we proposed a 2D implementation of the AMT process, including 5 transform types. Although it provides a good frame rate performance, it presents a fairly significant logic resource use due to its high complexity level. Indeed, the pipeline inserted is the main reason of these consequences. This work can be optimized to not only have lower hardware cost, but also better performance. In fact, preserving the logic use allows deeper pipelining which leads to better frame rate processing. An alternative way of optimization, as one of our perspectives in future works, would be adopting the non-constant coefficient multiplication approach explained in chapter 7. This would be an interesting work especially that MTS includes different matrix sizes for different transform types. Therefore, a drastic logic consumption reduction can be achieved as provided in chapter 7. It is worth noting that it would also imply some data dependencies which could slightly affect the frame rate processing. As a result, a well studied combination of both approaches would find a good trade-off between computational complexity and hardware resource constraints while preserving the desirable performance.

Currently, VVC primary transform module MTS includes only DCT-II, DST-VII and DCT-VIII as three core transform types. but it introduces the zeroing out technique for 32-point DCT-VIII and DST-VII in order to further reduce their computational complexity. As a short time perspective, it would be interesting to support this technique with the non constant coefficient multiplication to propose an implementation that is compatible with the current version of VVC standard and compare it to the approximation design proposed in chapter 5 in terms of operation count, hardware requirements and processed frames per second etc. Moreover, 64-point matrix multiplication with zeroing out technique is also defined only for DCT-II transform. That would be taken into consideration in the future works to cope with the VVC standardization process.

In chapter 6, in the context of an internship, we have experienced the co-designing approach which is an appealing motivation for researchers with the evolution of SoPC and ASIC platform products such as the Arria 10 FPGA SoC used in this thesis. As explained in this chapter, the bottleneck will be always the data transfer time for processor/FPGA communication. Regarding the developed application, using larger 128-bit com-bus and DMA memory interfaces would be interesting to reduce this transfer time. In the next step, a bigger hardware IP that encapsulates other coding tools related to transform process, such as intra coding and Low Frequency Non Separable Transforms, could be a good subject to be integrated in a whole co-design decoder where the hardware IP will be responsible for complex and high consuming tasks. Indeed, the IETR lab, which have developed the well known HEVC decoder "OpenHEVC", has already started to work on this project based on my research works, along with other internships and PhDs recently started towards "OpenVVC" decoder.

Appendix A

Author scientific publications

A.1 Journal papers

[J1] **Ahmed Kammoun**, Wassim Hamidouche, Pierrick Philippe, Olivier Deforges, Fatma Belghith, Nouri Masmoudi and Jean-Francois Nezan. "Forward-Inverse 2D Hardware Implementation ofApproximate Transform Core for the VVC Standard". Manuscript submitted for publication in IEEE Transactions on Circuits and Systems for Video Technology TCSVT 2019 (Minor Revision).

[J2] **Ahmed Kammoun**, Wassim Hamidouche, Fatma Belghith, Jean-François Nezan and Nouri Masmoudi. "Hardware Design and Implementation of Adaptive Multiple Transforms for the Versatile Video Coding Standard". IEEE Transactions on Consumer Electronics, Institute of Electrical and Electronics Engineers, 2018, 64 (4), pp.424-432.

[J3] **Ahmed Kammoun**, Fatma Belghith, Hassen Loukil and Nouri Masmoudi. "An efficient and unified 2D-inverse integer cosine transforms (IICT) FPGA-hardware implementation for HEVC standard". Analog Integrated Circuits and Signal Processing, Springer Verlag, 2019

A.2 International conference papers

[C1] **Ahmed Kammoun**, Wassim Hamidouche, Pierrick Philippe, Fatma Belghith, Nouri Massmoudi,Jean-François Nezan. "Hardware Acceleration of Approximate Transform Module for the Versatile Video Coding Standard". European Signal Processing Conference (EUSIPCO 2019), Sep 2019, Coruña, Spain.

[C2] Wassim Hamidouche, Pierrick Philippe, C.-E. Mohamed, **Ahmed Kammoun**, Daniel Menard and Olivier Deforges. "Hardware-friendly DSTVII/DCT-VIII Approximations for the Versatile Video Coding Standard", IEEE Picture Coding Symposium (PCS 2019).

[C3] **Ahmed Kammoun**, Sonda Ben Jdidia, Fatma Belghith, Wassim Hamidouche, Jean François Nezan and N. Masmoudi. “An optimized hardware implementation of 4-point adaptive multiple transform design for post-HEVC. 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Mar 2018, Sousse, France. pp.1-6.

[C4] Sonda Ben Jdidia, **Ahmed Kammoun**, Fatma Belghith, Nouri Masmoudi. "Hardware implementation of 1-D 8-point adaptive multiple transform in post-HEVC standard". 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Dec 2017, Monastir, France. pp.146-151.

[C5] **Ahmed Kammoun**, Fatma Belghith, Hassen Loukil, Nouri Masmoudi. "An optimized and unified architecture design for H.265/HEVC 1-D inverse core transforms". 2016 International Image Processing, Applications and Systems (IPAS), Nov 2016, Hammamet, Tunisia. pp.1-6.

Bibliography

[1] *Cisco Visual Networking Index :Forecast and Trends, 2017–2022,November 2018*. [Online; accessed 14-August-2019]. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper.

[2] *Ericsson Mobility Report, November 2018*. [Online; accessed 14-August-2019]. URL: https://www.ericsson.com/en/mobility-report/reports/november-2018.

[3] Y. Liu et al. "A multi-modeling electro-optical transfer function for display and transmission of high dynamic range content". *IEEE Trans. Consum. Electron.* 63.4 (Nov. 2017), pp. 350–358. ISSN: 0098-3063. DOI: 10.1109/TCE.2017.015068.

[4] G. J. Sullivan et al. "Overview of the High Efficiency Video Coding (HEVC) Standard". *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pp. 1649–1668. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221191.

[5] M. Wien. "High Efficiency Video Coding". *Signals and Communication Technology* (Dec. 2015).

[6] H265 ISO/IEC Recommendation. "High Efficiency Video Coding (HEVC)". *23008-2 MPEG-H Part 2* (2013).

[7] F. Pescador et al. "Complexity analysis of an HEVC decoder based on a Digital signal processor". *IEEE Trans. Consum. Electron.* 59.2 (May 2013), pp. 391–399.

[8] J. R. Ohm et al. "Comparison of the Coding Efficiency of Video Coding Standards;Including High Efficiency Video Coding (HEVC)". *IEEE Trans. Circuits Syst. Video Technol* 22.12 (Dec. 2012), pp. 1669–1684. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221192.

[9] T. Wiegand et al. "Overview of the H.264/AVC video coding standard". *IEEE Trans. Circuits Syst. Video Technol* 13.7 (July 2003), pp. 560–576. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815165.

[10] Joint-Video-Exploration-Team. *Several JVET meetings*. [Onine]. Available: https://jvet.hhi.fraunhofer.de/.

[11] A. Segall et al. "Joint Call for Proposals on Video Compression with Capability beyond HEVC". *JVET document H1002 (JVET-H1002 v4), 8th JVET Meeting: MACAO, China* (Oct. 2017).

[12] N. Sidaty et al. "Compression efficiency of the emerging video coding tools". *2017 IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 2996–3000. DOI: 10.1109/ICIP.2017.8296832.

[13] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 7". *JVET document G1001 (JVET-G1001), 7th JVET Meeting: Torino, IT*. July 2017.

[14] Joint-Video-Exploration-Team. *JEM software reference*. [Onine]. Available: https://jvet.hhi.fraunhofer.de/ svn/svn-HMJEMSoftware.

[15] X. Zhao et al. "Enhanced Multiple Transform for Video Coding". *Data Compression Conference (DCC)* (Mar. 2016), pp. 73–82.

[16] Cyclon-V-Device-Overview. *Intel 2016*. [Onine]. Available: https://www.altera.com/documentation/sam1403480548153.html.

[17] Intel-Arria-10-Device-Overview. *Intel 2017*. [Onine]. Available: https://www.altera.com/documentation/sam1403480274650.html.

[18] Intel-Stratix-10-GX/SX-Device-Overview. *Intel 2017*. [Onine]. Available: https://www.altera.com/ documentation/joc1442261161666.html.

[19] Intel-FPGA-Integer-Arithmetic-IP-Cores-User-Guide. *Intel 2017*. [Onine]. Available: https://www.altera.com/en_US/pdfs/literature/ug/ug_lpm_alt_mfug.pdf.

[20] "Video Codec for Audiovisual Services at p×64 kbits." *ITU-T. Recommendation H.261:* (1993).

[21] "Generic Coding of Moving Pictures and Associated Audio Information – Part2: Video". *ISO/IEC. International Standard 13818-2:* (1999).

[22] "Transmission of Non-Telephone Signals". *ITU-T. Recommendation H.262:* (1999).

[23] "Video Coding for Low Bit Rate Communication". *ITU-T. Recommendation H.263:* (1995).

[24] "Coding of Audio-Visual Objects – Part2: Visual". *ISO/IEC. International Standard 14496-2:* (1995).

[25] "Advanced Video Coding". *ITU-T. Recommendation H.264:* (2003).

[26] "High Efficiency Video Coding". *ITU-T. Recommendation H.265:* (2013).

[27] G. J. Sullivan. "Video Coding Standards Progress Report:Joint Video Experts Team Launches the Versatile Video Coding Project". *MPTE Motion Imaging Journal* 127 (2018), pp. 94–98.

[28] R. Westwater and B. Furht. "Real-Time Video Compression Techniques and Algorithms". *Springer US* (1997).

[29] I. Pitas. "Digital image processing algorithms and applications". *John Wiley and Sons* (2000).

[30] F. Bossen. "Common test conditions and software reference configurations". *Tech. Rep. JCTVC-L1100, Joint Collaborative Team on Video Coding (JCTVC)*. Jan. 2013.

[31] F. Bossen et al. "JVET common test conditions and software reference configurations". *JVET document N1010 (JVET-N1010), 14th JVET Meeting: Geneva, CH*. Mar. 2019.

[32] I. E. Richardson. "H.264 and MPEG-4 Video Compression: Video Coding for Nextgeneration Multimedia". *John Wiley and Sons* (2004).

[33] Z. Wang et al. "Image quality assessment: from error visibility to structural similarity". *IEEE transactions on image processing* 13 (2004), pp. 600–612.

[34] G. Bjontegaard. "Calculation of average PSNR differences between RD-Curves". *Technical Report VCEG-M33, ITU-T SG16/Q6 VCEG, Austin,Texas* (2001).

[35] G. Bjontegaard. "Improvements of the BD-PSNR model". *Technical Report VCEG-AI11, ITU-T SG16/Q6 VCEG, Berlin, Germany* (2008).

[36] G.J. Sullivan and T. Wiegand. "Rate-Distortion Optimization for Video Compression". *IEEE Signal Process* 15 (1998), pp. 74–90.

[37] V. Sze, M. Budagavi, and G. J. Sullivan. "High Efficiency Video Coding (HEVC): Algorithms and architectures". *Integrated Circuits and Systems. Springer International Publishing* (New York, USA, 2014).

[38] I.-K. Kim et al. "Block Partitioning Structure in the HEVC Standard". *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 22.12 (Dec. 2012), pp. 1697–1706.

[39] V. Sze and M. Budagavi. "High Throughput CABAC Entropy Coding in HEVC". *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 22.12 (Oct. 2012), pp. 1778–1791.

[40] A. Norkin et al. "Block Partitioning Structure in the HEVC Standard". *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 22.12 (Oct. 2012), pp. 1746–1754.

[41] C.M. Fu et al. "Sample Adaptive Offset in the HEVC Standard". *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 22.12 (Oct. 2012), pp. 1755–1764.

[42] Seo et al. "Video coding performance for hierarchical coding block and transform block structures". *Korea Society Broading Engineers Magazine* 15 (2010), pp. 23–34.

[43] B. Bross et al. "High effciency video coding (hevc) text spec. draft 10". *JCT-VC Doc. JCTVCL1003, 12th Meeting : Geneva, Switzerland* 1 (2013).

[44] J. Lainema et al. "Intra Coding of the HEVC Standard". *IEEE Trans. Circuits Syst. Video Technol* 22.12 (Dec. 2012), pp. 1792–1801. ISSN: 1051-8215.

[45] R. Li, B. Zeng, and M. L. Liou. "A new three-step search algorithm for block motion estimation". *IEEE Trans. Circuits Syst. Video Technol* 4.4 (1994), pp. 438–442.

[46] L.-K. Liu and E. Feig. "A block-based gradient descent search algorithm for block motion estimation in video coding". *IEEE Trans. Circuits Syst. Video Technol* 6.4 (1996), pp. 419–422.

[47] L.-M. Po and W.-C. Ma. "A novel four-step search algorithm for fast block motion estimation". *IEEE Trans. Circuits Syst. Video Technol* 6.3 (1996), pp. 313–317.

[48] E.Y. Lam and J.W. Goodman. "A mathematical analysis of the DCT coefficient distributions for images". *IEEE Transactions on Image Processing (TIP)* 9.10 (Oct. 2000), pp. 1661–1666.

[49] J. Sole et al. "Transform Coefficient Coding in HEVC". *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 22.12 (Oct. 2012), pp. 1765–1777.

[50] V. Sze, M. Budagavi, and G. J. Sullivan. "High Efficiency Video Coding (HEVC)". *Integrated Circuits and Systems. Springer International Publishing* (2014).

[51] M. Wien. "High Efficiency Video Coding". *Signals and Communication Technology. Springer Berlin* (2015).

[52] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 1". *JVET document A1001 (JVET-A1001), 1st JVET Meeting: Geneva, CH*. Oct. 2015.

[53] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 2". *JVET document B1001 (JVET-B1001), 2nd JVET Meeting: San Diego, USA*. Feb. 2016.

[54] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 3". *JVET document C1001 (JVET-C1001), 3rd JVET Meeting: Geneva, CH*. June 2016.

[55] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 4". *JVET document D1001 (JVET-D1001), 4th JVET Meeting: Chengdu*. Oct. 2016.

[56] J. Chen et al. "Algorithm Description of Joint Exploration Test Model 5". *JVET document E1001 (JVET-E1001), 5th JVET Meeting: Geneva, CH*. Jan. 2017.

[57] B. Bross. "Versatile Video Coding (Draft 1)". *JVET document J1001 (JVET-J1001), 10th JVET Meeting: San Diego, USA*. Apr. 2018.

[58] J. Chen and E. Alshina. "Algorithm description for Versatile Video Coding and Test Model 1 (VTM 1)". *JVET document J1002 (JVET-J1002), 10th JVET Meeting: San Diego, USA*. Apr. 2018.

[59] J. Chen, Y. Ye, and S.H. Kim. "Algorithm description for Versatile Video Coding and Test Model 6 (VTM 6)". *JVET document O2002 (JVET-O2002), 15th JVET Meeting: Gothenburg, SE*. July 2019.

[60] B. Bross. "High Efficiency Video Coding (HEVC): Algorithms and architectures". *8K seminar* (New York, USA, 2014).

[61] K. R. Rao and P. Yip. "The Transform and Data Compression Handbook". *Boca Raton* (2001).

[62] A. Jallouli et al. "Statistical analysis of Post-HEVC encoded videos". *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*. Oct. 2017, pp. 1–6. DOI: 10.1109/SiPS.2017.8110020.

[63] A. Arrufat et al. "Low complexity transform competition for HEVC". *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, pp. 1461–1465. DOI: 10.1109/ICASSP.2016.7471919.

[64] T. Biatek, V. Lorcy, and P. Philippe. "Transform Competition for Temporal Prediction in Video Coding". *IEEE Transactions on Circuits and Systems for Video Technology* (2018), pp. 1–1. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2018.2805877.

[65] P.K. Meher et al. "Efficient Integer DCT Architectures for HEVC". *IEEE Transactions on Circuits and Systems for Video Technology* 24.1 (Jan. 2014), pp. 168–178. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2013.2276862.

[66] X. Zhao et al. "Joint Separable and Non-Separable Transforms for Next-Generation Video Coding". *IEEE Trans. Image Process.* 27.5 (May 2018), pp. 2514–2525. ISSN: 1057-7149. DOI: 10.1109/TIP.2018.2802202.

[67] Wei-Jung Chien et al. "JVET AHG report: Tool reporting procedure (AHG13)". *JVET document N0013 (JVET-N0013), 14th JVET Meeting: Geneva, CH*. Mar. 2019.

[68] C.W. Chang et al. "A Fast Algorithm-Based Cost-Effective and Hardware-Efficient Unified Architecture Design of 4x4, 8x8, 16x16, and 32x32 Inverse Core Transforms for HEVC". *Journal of Signal Processing System 82* 82.1 (Mar. 2016), pp. 69–89.

[69] S. Shen et al. "A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards". *IEEE International Conference on Multimedia and Expo (ICME)*. July 2012, pp. 788–793.

[70] M. Kammoun et al. "An Optimized Hardware Architecture Of 4×4, 8×8, 16×16 And 32×32 Inverse Transform for HEVC". *International Conference On Advanced Technologies For Signal and Image Processing ATSIP*. Mar. 2016.

[71] E. Kalali and I. Hamzaoglu. "FPGA Implementations of HEVC Inverse DCT Using High-Level Synthesis". *IEEE Design and Architectures for Signal and Image Processing (DASIP), Poland*. Sept. 2015.

[72] S. Goto H. Sun D. Zhoi. "A low-cost VLSI architecture of multiple-size IDCT for H.265/HEVC". *IEICE Transactions on Fundamentals of Electronics* 97.12 (Dec. 2014), pp. 2467–2476.

[73] S. Shen et al. "A Unified 4/8/16/32-Point Integer IDCT Architecture for Multiple Video Coding Standards". *2012 IEEE International Conference on Multimedia and Expo*. July 2012, pp. 788–793. DOI: 10.1109/ICME.2012.7.

[74] M.Chen, Y. Zhang, and C. Lu. "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms". *International Journal of Electronics and Communications* 73 (Mar. 2017), pp. 1–8.

[75] A. Ahmed and M.U. Shahid. "N Point DCT VLSI Architecture for Emerging HEVC Standard". *VLSI Design* (2012), pp. 1–13.

[76] A.Can. Mert, E. Kalali, and I.Hamzaoglu. "High Performance 2D Transform Hardware for Future Video Coding". *IEEE Trans. Consum. Electron.* 62.2 (May 2017).

[77] M.J. Garrido et al. "A High Performance FPGA-based Architecture for Future Video Coding Adaptive Multiple Core Transform". *IEEE Trans. Consum. Electron.* (Mar. 2018).

[78] M. J. Garrido et al. "A 2-D Multiple Transform Processor for the Versatile Video Coding Standard". *IEEE Transactions on Consumer Electronics* 65.3 (Aug. 2019), pp. 274–283. ISSN: 0098-3063. DOI: 10.1109/TCE.2019.2913327.

[79] A. Kammoun et al. "An optimized hardware implementation of 4-point adaptive multiple transform design for post-HEVC". *2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. Mar. 2018, pp. 1–6. DOI: 10.1109/ATSIP.2018.8364448.

[80] A. Kammoun et al. "Hardware Design and Implementation of Adaptive Multiple Transforms for the Versatile Video Coding Standard". *IEEE Transactions on Consumer Electronics* 64.4 (Oct. 2018). ISSN: 0098-3063. DOI: 10.1109/TCE.2018.2875528.

[81] A. Said et al. "Complexity Reduction for Adaptive Multiple Transforms (AMTs) using Adjustment Stages". *Document JVET J0066 (JVET-J0066 v3), 10th JVET Meeting: San Diego, CA, USA*. Apr. 2018.

[82] A. Said et al. "Efficient Implementations of AMT with Transform Adjustment Stages". *Document JVET K0272 (JVET-K0272 v2), 11th JVET Meeting: Ljubljana, SI*. July 2018.

[83] P. Philippe and V. Lorcy. "Further simplification for AMT complexity reduction (CE6.1.2)". *Document JVET K0299 (JVET-K0299), 11th JVET Meeting: Ljubljana, SI*. July 2018.

[84] V. Lorcy and P. Philippe. "CE6: Further simplification of AMT with adjustment stages (Test CE6.1.6b)". *Document JVET L0135 (JVET-L0135 v1), 12th JVET Meeting: Macao, CN*. Oct. 2018.

[85] M. Jridi, A. Alfalou, and P. K. Meher. "A Generalized Algorithm and Reconfigurable Architecture for Efficient and Scalable Orthogonal Approximation of DCT". *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.2 (Feb. 2015), pp. 449–457. ISSN: 1549-8328. DOI: 10.1109/TCSI.2014.2360763.

[86] G. Renda et al. "Approximate Arai DCT Architecture for HEVC". *2017 New Generation of CAS (NGCAS)*. Sept. 2017, pp. 133–136. DOI: 10.1109/NGCAS.2017.38.

[87] H. A. F. Almurib, T. N. Kumar, and F. Lombardi. "Approximate DCT Image Compression Using Inexact Computing". *IEEE Transactions on Computers* 67.2 (Feb. 2018), pp. 149–159. ISSN: 0018-9340. DOI: 10.1109/TC.2017.2731770.

[88] Z. Chen, Q. Han, and W. Cham. "Low-Complexity Order-64 Integer Cosine Transform Design and its Application in HEVC". *IEEE Transactions on Circuits and Systems for Video Technology* 28.9 (Sept. 2018), pp. 2407–2412. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2018.2822319.

[89] M. Jridi and P. K. Meher. "Scalable Approximate DCT Architectures for Efficient HEVC-Compliant Video Coding". *IEEE Transactions on Circuits and Systems for Video Technology* 27.8 (Aug. 2017), pp. 1815–1825. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2016.2556578.

[90] Maher Jridi, Ayman Alfalou, and Pramod K. Meher. "Efficient approximate core transform and its reconfigurable architectures for HEVC". *Journal of Real-Time Image Processing* (Apr. 2018). ISSN: 1861-8219. DOI: 10.1007/s11554-018-0768-x. URL: https://doi.org/10.1007/s11554-018-0768-x.

[91] Y. Arai, T. Agui, and M. akajimaN. "A Fast DCT-SQ Scheme for Images". *Transactions of IEICE* E71.11 (Nov. 1988), pp. 1095–1097.

[92] Intel-Arria-10-Device-specificities. *Intel 2017*. [Onine]. Available: https://www.intel.fr/content/www/fr/fr/products/programmable/fpga/arria-10.html.

[93] Ahmed Kammoun et al. "An Optimized and Unified Architecture Design for H.265/HEVC 1-D Inverse Core Transform". *IEEE IPAS 2016, International Image Processing Applications and Systems Conference* (Nov. 2016).

[94] Intel-FPGA-Download-Center. [Onine]. Available: https://www.altera.com/downloads/download-center.html.

[95] Mentor-ModelSim-Functional-Verification-Tool-web. [Onine]. Available: https://www.mentor.com/products/fv/modelsim.

[96] P. K. Meher et al. "Efficient Integer DCT Architectures for HEVC". *IEEE Trans. Circuits Syst. Video Technol* 24.1 (Jan. 2014), pp. 168–178. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2013.2276862.

[97] A. K. Jain. "A Sinusoidal Family of Unitary Transforms". *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.4 (Oct. 1979), pp. 356–365.

[98] Gerlind Plonka and Manfred Tasche. "Fast and Numerically Stable Algorithms for Discrete Cosine Transforms". *Linear Algebra and its Applications* 394 (2005), pp. 309–345.

[99] C. Loeffler, A. Ligtenberg, and G. S. Moschytz. "Practical fast 1-D DCT algorithms with 11 multiplications". *International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* vol. 2. May 1989, pp. 988–991.

[100] Martin Vetterli and Henri J. Nussbaumer. "Simple FFT and DCT algorithms with reduced number of operations". *IEEE journal on Signal Processing* 6.4 (1984), pp. 267–278.

[101] Ravi K. Chivukula and Yuriy A. Reznik. "Fast Computing of Discrete Cosine and Sine Transforms of Types VI and VII". *Proc. SPIE*. Vol. 8135. 2011, pp. 8135 –8135 - 10.

[102] M. Puschel and J. M. F. Moura. "Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms for DCTs and DSTs". *IEEE Transactions on Signal Processing* 56.4 (Apr. 2008), pp. 1502–1521.

[103] A. Saxena and F. C. Fernandes. "DCT/DST-Based Transform Coding for Intra Prediction in Image/Video Coding". *IEEE Transactions on Image Processing* 22.10 (Oct. 2013), pp. 3974–3981.

[104] Wen-chyuan Yueh. "Eigenvalues of several tridiagonal matrices". *Applied Mathematics E-notes*. 2005, pp. 5–66.

[105] Laurence A. Wolsey. "Integer Programming". *Wiley, ISBN: 978-0-471-28366-9*. Sept. 1998.

[106] Rohit Chandra et al. "Parallel Programming in OpenMP". *Morgan Kaufmann Publishers, ISBN: 1-55860-671-8*. 2000.

[107] B. Bross, J. Chen, and S. LiU. "Versatile Video Coding (Draft 3)". *Document JVET L1001 (JVET-L1001 v9), 12th JVET Meeting: Macao, CN*. Oct. 2018.

[108] JVET. "VVC Test Model (VTM) vesion 3.0". https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware-VTM/tree/VTM-3.0, Dec. 2018.

[109] Xilinx-FPGA-products. [Onine]. Available: https://www.xilinx.com/products/silicon-devices.html.

[110] Intel-FPGA-products. [Onine]. Available: https://www.intel.fr/ content/www/fr/fr/products/programmable.html.

[111] C. Intel. "Intel Quartus Prime Pro Edition User Guide, Platform Designe". *Intel documentation center*. Oct. 2019.

[112] X. Zhao et al. "NSST, Non-separable secondary transforms for next generation video Coding". *Picture Coding Symposium PCS*. 2016.

[113] A. Arrufat, P. Philippe, and O. Deforges. "Non-separable mode dependent transforms for intra coding in HEVC". *IEEE Proc. on Visual Communications and Image Processing*. Dec. 2014, pp. 61–64.

[114] X. Zhao et al. “Improvements on non-separable secondary transform”. *JVET document D0120 (JVET-D0120), 4th JVET Meeting: Chengdu*. Oct. 2016.

[115] M. Koo et al. “Reduced Secondary Transform (RST) ”. *JVET document N0193 (JVET-N0193), 14th JVET Meeting: Geneva, Ch*. Mar. 2019.