REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de **T**echnologie

Département du **G**énie **E**lectrique

Mémoire de fin d'études de Master

Filière : télécommunications

Spécialité : Systèmes des télécommunications

Implémentation sur FPGA d'un décodeur entropique CABAC utilisé dans la nouvelle norme de compression vidéo VVC/H.266

Présentés par :

Chouihi lakhdar

Softa khalil

Proposés et Dirigés par :

Dr. MENASRI Wahiba

Dr. BENGHERBIA Billal

Promotion 2022

Remerciement

*Nous tenons à remercier « **Allah** » le tout puissant de nous avoir donné le courage, la force et la volonté pour réaliser ce mémoire.*

La présentation de ce travail nous avons offre d'exprimer toute mes reconnaissances aux personnes qui nous aidées et conseillés de façons inestimables durant la préparation de ce mémoire.

Nous tenons à exprimer notre plus haute estime à Mlle. MENASRI Wahiba ma promotrice pour s'assistance, leur encouragement et orientation qu'elle a donnée.

Nous remercons encore toutes les personnes qui ont contribué de près ou de loin la réalisation de ce mémoire.

Nous souhaitons que ce mémoire soit utile à toutes les futures.

Dédicace :

Rien n'est aussi beau à offrir qui le fruit d'un labeur qu'on dédie Du fond du cœur à ceux pour qui nous aimons et remercionsen exprimant la gratitude. Je dédie cet humble travail à ceux que j'aime dans les limites de l'imagination de ma mère et de mon père et toute la famille et de mes amis et perents a toute nos cousins a ceus qui se sont sacrifiés et qui se sont donnés pour les études et dons .

Je remercie également la professeure encadrante, Menasri Wahiba, pour son suivi et ses conseils du début à la fin des travaux.

Chouihi Lakhdar.

Dédicace :

A mon bonheur à celle qui ma arrosé de tendresse et d'es poirs à la source d'amour « « ma mère « «

A mon support dans ma vie qui ma appris ma suppotré et ma dirigé vers la gloire « « mon pére » »

A mes chères frères et sœurs.

A tout les personnes de ma grande famille et surtout

« « Ayoub et abd elbasset » »

A mes meilleurs amis kamel .Fethi. Mustapha. abd elnour.

A mes collègue oussama lakhdar

Et tout ce qui m'ont aidé à préparer ce travail.

A mes collègues étudiants de ma promotion 2022.

Softa Khalil.

Résumé

Le développement d'applications multimédias est dû à la technologie de compression vidéo. Au Juillet 2020, le JVET (Joint Video Experts Team) a innové la norme VVC/H.266 (Versatile Video Coding), qu'est le dernier standard de compression vidéo. Une des techniques indispensables dans cette norme est le codage entropique CABAC (Context Adaptive Binary Arithmetic Coding), il est considéré comme une partie essentielle dans le codeur VVC. Le décodeur CABAC est constitué de trois parties : BAD (Binary Arithmetic Decoding), CSM (Context Selection and Modeling) et le DB (De-Binarisation), où le BAD est chargé pour traiter une séquence entière de symboles comme une seule entité basée uniquement sur la distribution de probabilité des symboles à coder, il se compose de trois modes de codage : RM (Regular Mode), BM (Bypass Mode) et le TM (Terminate Mode). Les résultats de l'implémentations logicielle (Matlab) et matérielle (FPGA) montrent son bon fonctionnement et la similarité des résultats obtenus.

Mots Clés: Applications multimédias, BAD, CABAC, CSM, DB, Entropy coding, FPGA, JVET, Matlab, Video compression, VVC.

Abstract

The development of multimedia applications is due to vidéo compression technology. In July 2020, the Joint Video Experts Team (JVET) has innovated the Versatile Video Coding (VVC/H.266) standard, which is the latest video compression standard. One of the essential techniques in this standard is the entropic coding Context Adaptive Binary Arithmetic Coding (CABAC), it is considered an essential part in the VVC encoder. The CABAC decoder consists of three parts: Binary Arithmetic Decoding (BAD) , Context Selection and Modeling (CSM) and the Dé-Binary Arithmetic Decoding (DB), where the BAD is adapted to process an entire sequence of symbols as a single entity based uniquely on the probability distribution of the symbols to be encoded, it consists of three encoding modes: Regular Mode (RM), Bypass Mode (BM) and le Terminate Mode (TM). The results of the software (Matlab) and hardware (FPGA) implementations show good functioning and the accuracy of the both obtained results. it works and how similar the results are.

Key words: BAD, BM, CABAC, CSM, DB, entropy coding, FPGA, Matlab, multimedia applications, RM, TM, Video compression, VVC.

ملخص

يرجع تطور تطبيقات الوسائط المتعددة إلى تكنولوجيا الضغط للفيديو. في يوليو 2020، ابتكر JVET (فريق خبراء الفيديو المشترك) معيار VVC/H.266 (ترميز الفيديو متعدد الاستخدامات)، وهو أحدث معيار لضغط الفيديو. إحدى التقنيات الأساسية في هذا المعيار هي الترميز الإنتروبي CABAC (الترميز الحسابي الثنائي التكيفي السياق)، ويعتبر جزءًا أساسيًا في مشفر VVC. يتكون فك تشفير CABAC من ثلاثة أجزاء: BAD (فك التشفير الحسابي الثنائي) وCSM (اختيار السياق والنمذجة) وDB (فك الثنائيات)، حيث يتم تحميل BAD لمعالجة تسلسل كامل من الرموز ككيان واحد يعتمد بشكل فريد على RM (الوضع العادي)، BM (الوضع الالتفافي) وTM (الوضع النهائي). تُظهر نتائج تنفيذ البرنامج (Matlab) والأجهزة (FPGA) مدى نجاحه ومدى تشابه النتائج.

الكلمـات المفتـاحيـة: ضـــغط الفيـديو، ترميز الكون، عنـاصـــر التركيـب، CABAC، FPGA، RM،TM ، Matlab، VVC.

Liste des acronymes

A

ASIC : Application Specific Integrated Circuits

AVC : Advanced Vidéo Coding

AVI : Audio Video Interleave

B

BAD : Binary Arithmetic Decoding

Bin : Symboles Binaires

C

CABAC : Context Adavptive Binary Arithmetic Coding

CAVLC : Context Adaptive Variable Length Coding

CTU : Coding Tree Unit

CtxIdx : Context Index Adress

CSM : Context Selection and Modeling

D

DB : Dé-Binarisation

DC : Composante Directe

DCT : Discret Cosine Transform

DST : Discrete Sinus Transform

F

FLV (Format FLash Video

FSM : Finite State Machine

FPGA : Field-Programmable Gate Array

H

HD : Haute Definition

HDR : High Dynamic Range

HDTV : High Définition Télévision

HEVC : High Efficiency Video Coding

I

IDE : environnement de développement intégré

ISE : Integrated Software Environment

ISO : Organisation International de Standardisation

ITU-T : International Telecommunication Union

J

JVT : Joint Video Team

JVET : Joint Video Experts Team

L

LUT : Look Up Table

LPS : Least Probable Symbol

M

ME : estimation de mouvement

MR : Regular Mode

MB : Bypass Mode

MT : Terminate Mode

MPS : Most Probable Symbol

MPEG : Moving Pictures Expert Group

MPS : Most Probable Symbol

MTS : sélection de transformations multiples

MV : vecteurs de mouvement

N

NTSC : National Television System Committee

P

PAL : Phase Alternating Line

Q

QP : Paramètre de Quantification

Qstep : pas de quantification

R

RLE : Run length Encoding

RTL : Register Transfer Level

S

SAO : Sample Adaptive Offset

SDTV : Standard Définition Télévision

SE : Eléments Syntaxiques

T

TV : Télévision

U

UHD : Ultra High Definition

V

VCEG : Video Coding Expert Group

VHDL : Hardware Description Language

VLC : Variable Length Code

VVC : Versatile Vidéo Coding

W

WCG : Wide Colour Gamut

Figure 1.1 :	Chaine de compression/décompression vidéo ……..	05
Figure 1.2 :	Chaine de transmission vidéo …….....	06
Figure 1.3 :	Compression spatiale ……..…….....	06
Figure 1.4 :	Estimation de mouvement ……..…….....	07
Figure 1.5 :	Principales normes de codage vidéo selon l'organisme de standardisation……...	07
Figure 1.6 :	Schéma bloc du codec vidéo VVC/H.266…….....	11
Figure 1.7 :	Concept de sélection de transformées séparables 2D dans VVC ….....	12
Figure 1.8 :	Intra et Inter prédiction d'un block PU dans une séquence video …….....	13
Figure 1.9 :	Exemple simplifié de prédiction avancée du vecteur de movement …….	13
Figure 1.10 :	65 directions de l'intra-prédiction utilisés pour le VVC …….…….....	14
Figure 2.1 :	Système de communication d'informations……...	16
Figure 2.2 :	Auto-information et probabilité. d'un seul événement et d'un événement pondéré par sa probabilité ……..……...……	19
Figure 2.3:	Exemple de codage arithmétique …...……..	22
Figure 2.4 :	Architecture du codeur CABAC ……...	22
Figure 2.5 :	Architecture du décodeur CABAC……	24
Figure 2.6 :	Organigramme de sélection et modélisation du contexte (codage/décodage)…….	25
Figure 2.7 :	Division des sous intervalles dans le processus de décodage arithmétique (mode régulier)……..	26
Figure 3.1 :	Outils matériels et logiciels utilisés dans notre projet ……...	27
Figure 3.2 :	Entité et architecture d'un composant logique sur ISE……..…	28
Figure 3.3 :	Organigramme du mode régulier et renormalization……..…	30
Figure 3.4 :	Partie du programme matlab en mode régulier …….……...	31
Figure 3.5 :	Simulation saus matlab en mode régulier (les valeurs binVal)…..…….	31
Figure 3.6 :	Simulation saus matlab en mode régulier pour la variable shift0……..…	32
Figure 3.7 :	Simulation saus matlab en mode régulier pour la variable shift1……..…	32
Figure 3.8 :	Simulation saus matlab en mode régulier pour la variable pStateIdx0..…	33
Figure 3.9 :	Simulation sous matlab en Mode régulier pour la variable pStateIdx1.…	34
Figure 3.10 :	Organigramme du Mode bypass……	35
Figure 3.11 :	Partie du programme matlab en mode bypass……...…….	35
Figure 3.12 :	Simulation sous matlab de mode bypass……..……..	36
Figure 3.13 :	Organigramme du mode terminal et renormalisation……...	37
Figure 3.14 :	Partie du programme matlab en mode terminal…….……	38
Figure 3.15 :	Simulation sous matlab de mode terminal…..……	38
Figure 3.16 :	Organigramme du décodage arithmétique binaire globale……	39

Figure 3.17 :	Partie du programme matlab en décodage arithmétique binaire CABAC/VVC……...…….	40
Figure 3.18 :	Simulation sous matlab de décodage CABAC/VVC global……...…..…	42
Figure 4.1 :	Architecture Globale de module CABAC / VVC au niveau de décodage.	44
Figure 4.2 :	Architecture proposée du bloc de décodage arithmétique en mode régulier……...	45
Figure 4.3 :	Architecture de décodage arithmétique CABAC en mode bypass ……..	45
Figure 4.4 :	Architecture de décodage arithmétique CABAC en mode terminal ….....	46
Figure 4.5 :	Diagramme d'état FSM proposé pour le BAD……...……	47
Figure 4.6 :	Schéma du module RM. (a) top-level, (b) schéma RTL……	48
Figure 4.7 :	Description en VHDL de l'architecture du décodage arithmétique binaire CABAC/VVC sous ISE 14.7…….……	49
Figure 4.8 :	Schéma du module BM. (a) top-level, (b) schéma RTL …...……	50
Figure 4.9 :	Schéma du module TM. (a) top-level, (b) schéma RTL …...……	52
Figure 4.10:	Schéma obtenus du module CABAC de décodage arithmétique (a) top-level. (b) Schéma RTL……..	53
Figure 4.11:	Emulation sous model-Sim du mode régulier (les valeurs de binVal décode)……...	54
Figure 4.12:	Emulation sous model Sim du mode régulier (la mise à jour des variables pStateIdx0 et pStateIdx1)……..……..	55
Figure 4.13:	Emulation sous model-Sim du mode bypass……..…	55
Figure 4.14:	Emulation sous model-Sim du mode terminal……..…….	56
Figure 4.15:	Emulation du l'étape de la renormalisation (renormalisationd)…….	56
Figure 4.16:	Comparaison des valeurs binval obtenues depuis matlab et model-sim en mode régulier……...…….……	57
Figure 4.17:	Comparaison les valeurs de shift1 obtenus en matlab et model-sim en mode régulier……..…….....	58
Figure 4.18:	Comparaison entre les valeurs pStatIndx1 mises à jour obtenus en matlab et model-sim en mode régulier.……...……..….	59
Figure 4.19:	Comparaison des valeurs binval obtenues depuis matlab et model-sim en mode baypass……...……..	60
Figure 4.20:	Comparaison des valeurs binval obtenues depuis matlab et model-sim en mode terminale……...……..…….	60

Listes des tableaux

Tableau 1.1 : Comparaison des standards H.264, H.265 et H.266.…..……... 10

Tableau 4.1 : Synthèse de la partie CABAC en mode RM…….....…… 49

Tableau 4.2 : Synthèse de la partie CABAC en mode BM …… 51

Tableau 4.3 : Synthèse de la partie CABAC en mode TM ……...…….. 52

Tableau 4.4 : Synthèse de la partie CABAC par le bloc de décodage arithmétique... 54

Sommaire

Introduction générale 01

Chapitre 01 : Généralités sur la compression vidéo

1.1 Introduction 03
1.2 Généralités sur la vidéo 03
1.2.1 Définition de la vidéo 03
1.2.2 Formats vidéo 03
1.2.3 Objet 03
a) Les objets de contexte 03
b) Les objets mobiles 04
1.2.4 Types de vidéo 04
a) Vidéo analogique 04
b) Vidéo numérique 04
1.2.5 Paramètres clés d'une vidéo 04
a) Vitesse d'affichage 04
b) Résolution 04

1.3 Généralités sur la compression vidéo 04
1.3.1 Définition de la compression vidéo 04
1.3.2 Techniques de compression vidéo 05
1.3.3 Principe de compression vidéo 05
1.3.3.1 Prédiction spatiale (technique intra-image) 06
1.3.3.2 Redondance temporelle (technique intra-image) 06

1.3.4 Codeurs vidéo 07
a) Evolution des normes de compression vidéo et standardisation……..…07
b) Comparaison entre les standards H.264, H.265 et H.266……..……..10

1.4 Codeur VVC/H.266 10

1.4.1 Définition 10
1.4.2 Caractéristiques de la norme H.266 11
1.4.3 Transformation 11
1.4.4 Quantification 12
1.4.5 Prédiction 12
a) Inter Prédiction 13
b) Intra Prédiction 14
1.4.6 CABAC 14

1.5 Conclusion 15

Chapitre 02 : Codage entropique CABAC

2.1 Introduction 16

2.2 Système de communication 16
2.2.1 Codage source 17
2.2.2 Codage canal 17
2.2.3 Modulation 17
a) Modulation analogique 18
b) Modulation numérique 18

2.3 Généralités sur la théorie de l'information 18

2.3.1 Définition de l'information 19
2.3.2 Entropie 19
2.3.3 Codage entropique 20

2.3.3.1 Codage VLC 20

2.3.3.2 Codage de Shannon-Fano 20

2.3.3.3 Codage du Huffman 20

2.3.3.4 Codage RLE (Run-lengthEncoding) 21

2.3.3.5 Codage Arithmétique 21

2.4 Codage CABAC VVC 22

2.4.1 Binarisation 23

2.4.2 Sélection et modélisation du contexte 23

2.4.3 Codage arithmétique binaire 23

2.5 Décodage CABAC 23

2.5.1 Sélection et modélisation du contexte 24

2.5.2 Décodage arithmétique binaire 25

2.5.1.1 Mode régulier 25

2.5.1.2 Mode bypass 26

2.5.2.3 Mode terminal 26
2.5.3 Dé-binarisation 26

2.6 Conclusion 26

Chapitre 03 : Implémentation logicielle de Décodage BAD sous MATLAB

3.1 Introduction 27

3.2 Outils utilisés 27

3.2.1 Model Sim (PE Student Edition 6.3) 28
3.2.2 MATLAB 28
3.2.3 Xilinx ISE 14.5 28
3.3 Algorithme de décodage arithmétique binaire 29
3.3.1 Mode régulier 29
3.3.1.1 Organigramme 30
3.3.1.2 Programme sous Matlab 31
3.3.1.3 la simulation 31
3.3.2 Mode bypass 34
3.3.2.1 Organigramme 35
3.3.2.2 Programme sous Matlab 35
3.3.2.3 Simulation 36
3.3.3 Mode terminal 36
3.3.3.1 Organigramme 37
3.3.3.2 Programme sous Matlab 38
3.3.3.3 Simulation 38
3.4 Processus de décodage arithmétique global 38
3.4.1 Organigramme 38
3.4.2 Programme sous Matlab 40
3.4.3 la simulation 40
3.5 Conclusion 42

Chapitre 04 : Conception et Implémentation sur FPGA

4.1Introduction 43
4.2Architectures proposées pour la partie CABAC / VVC 43
4.2.1 Architecture Globale 43
4.2.2 Architecture proposée pour le Mode Régulier (RM) 43
4.2.3 Architecture proposée pour le Mode Terminal (TM) 45
4.2.4 Architecture proposée pour le Mode Bypass (BM) 46
4.2.5 finit state machine (FSM) 46
4.3 Partie Implémentation, synthèse et emulation 47

4.3.1 Synthèse et schéma RTL des codes de décodage arithmétique 47
a) Synthèse et schéma RTL de la partie CABAC en mode Régulier 47
b) Synthèse et schéma RTL de la partie CABAC en mode Baypass 49
c) Synthèse et schéma RTL de la partie CABAC en mode TM 51
d) Synthèse et schéma RTL de la partie CABAC en BAD 53
4.3.2 Emulation des 3 modes de décodage arithmétique CABAC/VVC 54
4.4 Comparaison et validation des Résultats obtenus 57
4.5 Conclusion 61
Conclusion générale 62

Introduction générale

Introduction générale

Durant la dernière décennie, la majorité des dispositifs électroniques ont la capacité de capturer et de mémoriser des vidéos (téléphones portables, appareils photo, …etc). Les vidéos brutes capturées ont une taille énorme qui est presque impossible à transmettre sur des canaux à bande passante limitée. Par conséquent, des techniques de compression vidéo efficaces sont nécessaires pour réduire la taille de la vidéo avant de la transmettre.

Le standard H.266/VVC (Versatile Vidéo Coding) est le codeur vidéo qui est utilisé aujourd'hui et qui garantit une capacité de compression plus efficace que leurs prédécesseurs. La partie CABAC (Context Adaptive Binary Arethmetic Coding) est l'une des techniques les plus importants utilisés dans la norme de compression vidéo VVC ; il est utilisé pour coder et décoder les éléments syntaxiques avant la transmissions ou le stockage des contenus vidéo.

Le décodeur CABAC est composé de trois fonctions : BAD (Binary Arithmetic Decoding), CSM (Context Selection and Modeling) et la Dé-binarisation [1]. La modélisation du contexte transforme les chaînes binaires en flux binaire selon les modèles de probabilité définis dans la norme VVC. La binarisation convertit les éléments syntaxiques d'entrée en chaînes binaires [2]. Le décodeur arithmétique binaire BAD traite la séquence complète de symboles comme un seul concept basé seulement sur la distribution de probabilité des symboles à décoder. Le BAD est basé sur trois modes de décodage :

- RM (Regular Mode) qui utilise le modèle du contexte ;
- BM (Bypass Mode) basé sur la probabilité équivalente ;
- TM (Terminate Mode) appelé à la fin de processus de décodage de chaque CTU (Coding Tree Unit).

Notre travail s'inscrit sur la partie BAD, nous nous sommes intéressées à l'implémentation d'un décodeur BAD dans la norme de compression vidéo VVC/H.266 sur une plateforme FPGA. Il est à signaler qu'avant de commencer à l'implémentation de notre algorithme sur FPGA de passer par l'implémentation sous MATLAB afin de maitriser l'algorithme.

Ce mémoire est organisé en quatre chapitres :

Le premier chapitre expose des généralités sur les normes de compression vidéo. En effet, dans une première étape, les techniques de compression sont données. La deuxième étape consiste à décrire les normes de compression vidéo depuis l'année 1994, à savoir le MPEG2, le

H.263, H.264, H.265 et le H.266. Ensuite, nous détaillerons la norme H.266/VVC et nous soulignons l'importance du codage CABAC.

Le deuxième chapitre présente d'abord la description de quelques notions de base sur le système de communication et la théorie de l'information. Ensuite, nous mettons l'accent sur le codage entropique CABAC ses types et plus précisément le décodeur binaire BAD.

On commence le troisième chapitre par les outils logiciels utilisés dans notre travail, suivi par la représentation des organigrammes des codes de décodage BAD avec leurs simulations sous Matlab.

Dans le quatrième chapitre, les architectures proposées de trois modes constituent le BAD du décodeur entropique CABAC dans la norme VVC seront présentées et en enchaine par la proposition des architectures matérielles de chaque mode. La partie contrôle est assuré par une FSM qui assure la gestion globale du système. A la fin du ce chapitre, une synthèse, une simulation et une évaluation de la consommation énergétique on a assuré par l'ISE et le Model -Sim suivi par des commentaires sur les résultats obtenus. De plus, une comparaison est effectuée entre les résultats obtenus par Matlab et Model-Sim est réalisée afin de vérifier le bon fonctionnement de notre algorithme.

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

Chapitre 01

1.1 Introduction

Au cours de la dernière décennie, les domaines d'application et les besoins de la compression vidéo ont changé. Pour cela, plusieurs normes de codage vidéo ont été employées. De plus, l'émergence de standards de compression vidéo permet d'obtenir une meilleure efficacité de compression avec la même qualité visuelle. Le but de ce chapitre est d'examiner certaines techniques de compression utilisées. Par conséquent, nous allons d'abord introduire le principe de la compression vidéo. Ensuite, nous présenterons les normes de codage vidéo et nous détaillerons la dernière norme de compression vidéo VVC/H.266.

1.2 Généralités sur la vidéo

1.2.1 Définition de la vidéo

Un flux vidéo est composé d'une succession d'images qui défilent à un rythme fixe pour donner l'illusion du mouvement par exemple 25 images par seconde dans la norme PAL (Phase Alternating Line) ou 30 images par seconde dans la norme NTSC (National Television System Committee).

De plus on retrouve généralement un flux audio ainsi que d'autres informations comme des sous titres, des menus, des informations de propriétés sur la vidéo comme la date de création, le nom de la vidéo, son auteur, etc…

1.2.2 Formats vidéo

Il existe une multitude de formats vidéo : Le format AVI (Audio Video Interleave) de Microsoft, du format MPEG (Moving Picture Experts Group) de l'organisme du même nom, du format mov d'Apple ou encore du format FLV (Format FLash Video) d'Adobe.

En effet, un format vidéo résulte en fait de deux concepts techniques distincts : les conteneurs et les codecs [3].

1.2.3. Objet

Les objets physiques sont les objets du monde réel qui apparaissent dans les scènes observées par les caméras. Il sont divisés en deux types : les objets de contexte et les objets mobiles [4].

a) Les objets de contexte : sont des objets physiques qui sont habituellement statiques (par exemple les murs). Dans le cas où ils ne sont pas statiques, leurs mouvements peuvent être prédits par les informations contextuelles comme. les chaises, les portes sont des objets de contexte [5].

b) Les objets mobiles : sont des objets physiques qui peuvent être perçus dans les scènes par leurs mouvements [5].

1.2.4. Types de vidéo

Le signal vidéo est le signal qui permet de transporter une séquence d'images de la source à un dispositif d'affichage sous forme électrique. Selon la façon dont les signaux sont traités on peut distinguer les deux modes :

a) Vidéo analogique

Elle décrit le signal analogique comme un signal électrique dont l'intensité varie dans le temps de façon continue. La qualité du signal final dans ce mode est plus faible car le bruit rajouté au signal lors son traitement altère sa qualité.

b) Vidéo numérique

Est un signal qui porte une information représentée par une suite de valeurs minimales ou maximales correspondant respectivement au 0 et au 1. L'un des facteurs qui avantage le signal numérique par rapport au signal analogique est la facilité de distinguer l'information émise du bruit [6].

1.2.5 Paramètres clés d'une vidéo

Le stockage et la diffusion d'une vidéo exigent un espace volumineux et un taux de transfert plus élevé. De plus, le contrôle de qualité.et la taille d'une séquence vidéo est déterminé par deux paramètres :

a) Vitesse d'affichage : Le nombre d'images du système visuel humain exigé en général une vitesse d'affichage de 25 ou 30 images par seconde.

b) Résolution : Ce terme désigne que la quantité de l'information est limitée dans l'image. Autrement, c'est le nombre de pixels qui peuvent être affichés par un dispositif d'affichage.

Trouver le compromis entre ces paramètres et les limitations imposées par la technologie permet d'obtenir une qualité de vidéo optimale.

1.3 Généralités sur la compression vidéo

1.3.1 Définition de la compression vidéo

En général, la compression des données consiste à prendre un flux de symboles et à les transformer en codes [7]. La compression est effectuée par un programme qui utilise une

formule ou un algorithme pour déterminer comment réduire la taille des données pour ganier en espace dans le cas de stockage et en bande passante dans le cas de transmission.

La compression de données est également appelée codage source ou réduction de débit. C'est un processus de modification, d'encodage ou de conversion de la structure des bits de données de manière à consommer moins d'espace sur le disque.

L'objectif de la compression des données est de représenter une source d'information aussi précisément que possible en utilisant le plus petit espace de stockage.

1.3.2 Techniques de compression vidéo

La compression de données vidéo permet de réduire la quantité d'informations à stocker ou à transmettre. Elle est basée sur plusieurs techniques pour réduire les redondances dans une vidéo. Les blocs suivants représentent toutes les étapes dans une chaine de compression/ décompression vidéo (Figure 1.1).

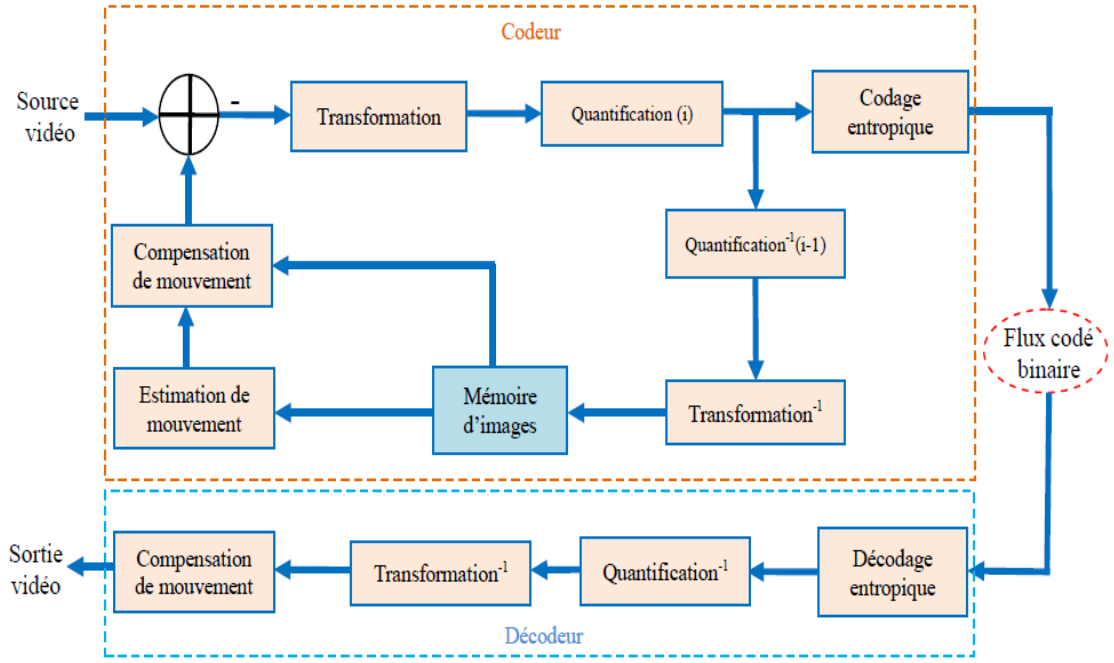


Figure 1.1 : Chaine de compression/décompression vidéo [8].

1.3.3 Principe de compression vidéo

Les algorithmes de compression vidéo sont basés sur une notion simple : il existe généralement au sein d'une vidéo de nombreuses redondances d'information, c'est-à-dire une similitude dans la même image ou entre les images constituant le flux vidéo. Aussi, le principe de la compression vidéo a pour objectif de réduire et d'éliminer cette redondance.

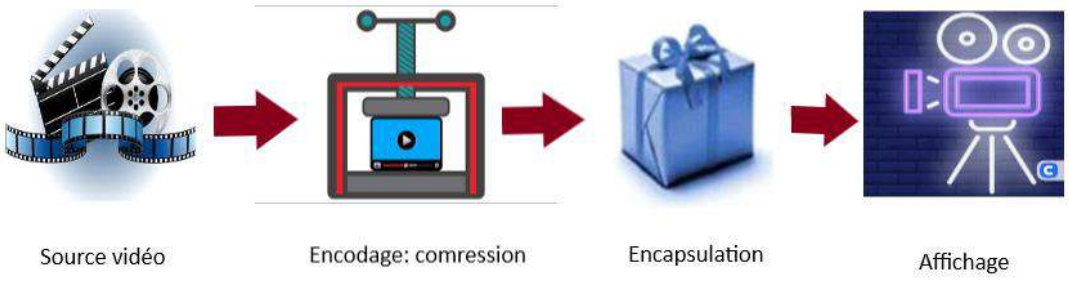


Figure 1.2 : Chaine de transmission vidéo [9].

La compression des images et des données vidéo peut suivre deux approches différentes [10] :

1.3.3.1 Prédiction spatiale (technique intra-image)

Les algorithmes de compression vidéo intra-image s'appuient sur les principes classiques de compression d'image où chaque image est travaillée individuellement. Les données sont réduites en supprimant les informations redondantes ou inutiles car non remarquables par l'œil humain.



Figure 1. 3 : Compression spatiale

1.3.3.2 Redondance temporelle (technique inter-image)

Il existe dans certaines séquences de vidéo des éléments redondants fixes ou mobiles. Les zones redondantes sont reprises des images précédentes. Dès lors, une image est décrite enfonction de sa différence par rapport à la précédente. Cela implique différents mécanismes d'identification des redondances :

- **Codage différentiel :** Où une image est comparée à une image de référence et seuls les pixels qui ont changé par rapport à cette image de référence sont codés. Cela permet de réduire le nombre de valeurs codées et stockées pour chaque pixel. Le codage différentiel compare des éléments statiques d'une image à l'autre.
- **Compensation de mouvement :** Utilisée afin de réduire encore davantage la quantité de données. Cette technique tient compte du fait qu'une bonne partie de ce qui compose une nouvelle image dans une séquence vidéo se retrouve dans une image

adjacente, mais pas forcément au même endroit.

Il est donc possible de composer ou de « prédire » une nouvelle image bloc par bloc en recherchant un bloc identique dans une image de référence. Si un bloc référent est trouvé, l'encodeur code l'emplacement dans l'image de référence où se situe le bloc identique ainsi que le vecteur de mouvement qui utilise moins d'espace que le codage du contenu réel d'un bloc.

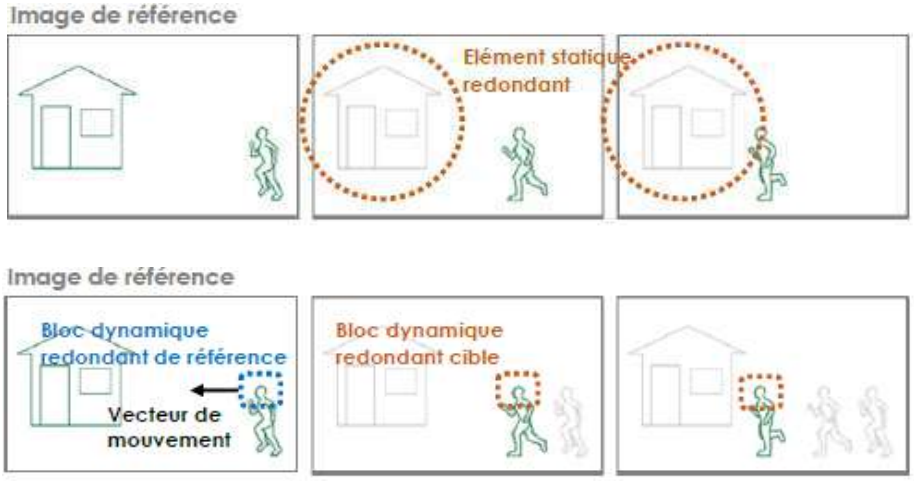


Figure 1. 4 : Estimation de mouvement

1.3.4 Codeurs vidéo

a) Evolution des normes de compression vidéo et standardisation

Pendant ces dernières années, plusieurs standards de codage vidéo ont vus le jour et se sont succédé, pour répondre à certaines applications, les algorithmes de codage et décodage vidéo ont évolués dont le but d'améliorer l'efficacité de la compression.

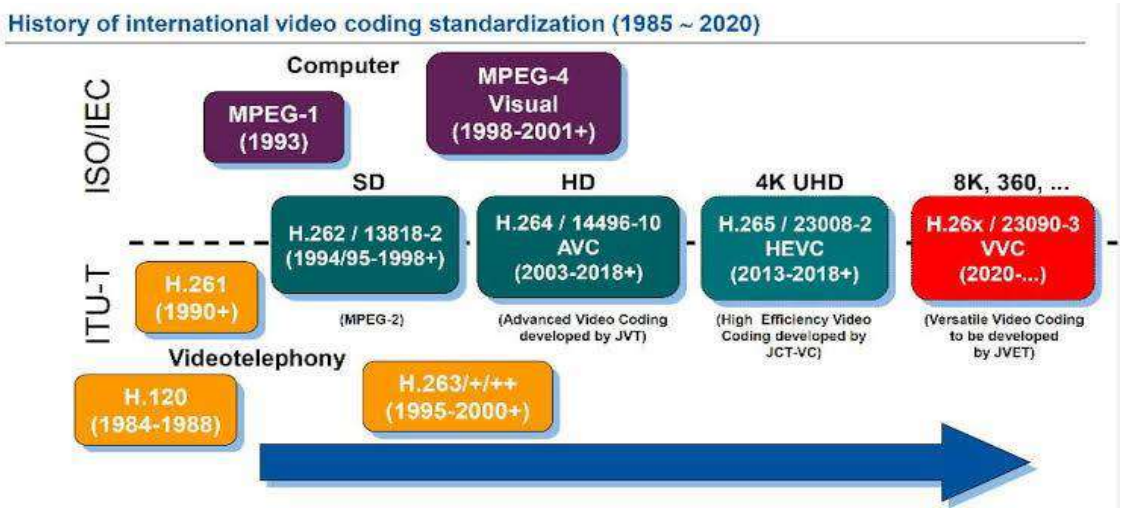


Figure 1.5 : Principales normes de codage vidéo selon l'organisme de standardisation [11].

L'historique des différentes familles de normes H.26x et MPEG-x établie par l'ITU-T (*International Telecommunication Union*), MPEG (Moving Picture Experts Group) ou JVT (*Joint Video Team*) est illustré sur la Figure (1.5). Parmi les codecs les plus répandus, nous citons :

MPEG-1 : En novembre 1991, le comité MPEG a lancé son premier standard permettant la compression des images en mouvements, en vue d'un stockage numérique d'une image ayant une résolution de 352 pixels par 240 pixels avec un débit pouvant atteindre 1.5Mb/s [12].

MPEG-2 : Dans l'objectif de répondre aux limitations présentées par l'ancien standard MPEG1 (basse qualité d'image, résolution limitée), le standard MPEG2 a été annoncé en novembre 1994 supportant la compression entrelacée de vidéos avec des débits variant de 1,5 à 100 Mb/s. Il a initialement été élaboré pour la transmission de la vidéo de qualité TV ayant un débit compris entre 4 et 9Mb/s. En outre, MPEG-2 est un schéma de compression couramment utilisée dans les DVD, ainsi que les émissions de radiodiffusion. Il est capable d'assurer le codage du SDTV (Standard Définition Télévision) et HDTV (High Définition Télévision) à des débits respectivement de 3,5 Mbit/s et 15Mbit/s [13].

MPEG-4 : Dans le but d'englober les codecs MPEG existants et de leur rajouter une nouvelle dimension donnant naissance à un standard beaucoup plus souple et étendu, un nouveau standard MPEG-4 apparaît en 1995. Ce standard est destiné à la manipulation d'objets multimédias, et a comme objectif de coder de manière efficace les séquences à très bas débit dédiées aux applications mobiles, mais il est capable de supporter des débits d'environ 1 Gbit/s. Dans le même contexte, les normes : H.261, H.262, H.263, H.264, H.265 et H.266 ont été développées pour la vidéo de communication [14].

H.261 : Est la norme internationale de compression vidéo la plus utilisée pour la vidéoconférence. Cette norme décrit les méthodes de codage et de décodage vidéo pour la composante d'image animée d'un service audiovisuel aux taux de p * 64 Kbps où p se situe dans la plage de 1 à 30. Les cibles standard et est réellement adapté pour applications utilisant des réseaux à commutation de circuits comme canaux de transmission [15].

H.262 : La norme de compression H.262 ou MPEG-2 Partie 2 (officiellement connu sous le nom de Recommandation UIT-T H.262 et ISO / CEI 13818-2, également appelé MPEG-2 Vidéo) est un format de codage vidéo normalisé et conjointement géré par le Groupe d' experts du codage vidéo (VCEG) de la Commission d'études 16 de l' UIT-T et le Groupe d' experts ISO / CEI en images animées (MPEG).

H.263 : La norme de compression H.263 est conçue pour une transmission vidéo à très faible débit. L'inconvénient de cette norme est que l'image est dégradée lorsque les objets sont en

mouvement. De plus elle était initialement destinée aux applications de vidéoconférence et non à la surveillance ou les détails ont plus d'importance que la régularité du débit [17].

H.264 (AVC) : En 2001, nous avons commencé à nous intéresser aux problématiques du codage vidéo. La norme de compression utilisée est la norme H.264, ou MPEG-4 /AVC (Advanced Vidéo Coding) qui est ouverte et sous licence compatible avec la plupart des techniques de compression disponibles. Elle est développée conjointement par l'ISO/IEC MPEG et ITU-T VCEG suite à leur fusion en 2001 sous le nom de JVT. La norme H.264 a permis de surpasser les codeurs précédents en modifiant le schéma de codage classique par l'introduction de certaines techniques communes comme la prédiction spatiale, les estimations et compensations de mouvements, la transformation en DCT (Discrete Cosine Transform) et le codage entropique [18].

H.265 (HEVC) : En 2013, la nouvelle norme de compression vidéo High Efficiency Video Coding (HEVC) a été développée conjointement par ITU-T VCEG, et ISO/IEC MPEG. L'objectif principal de la norme est de fournir des outils de codage efficaces pour des contenus de résolution Ultra High Definition (UHD) tout en prenant en compte les nouveaux formats High Dynamic Range (HDR) et Wide Colour Gamut (WCG). La norme HEVC permet une réduction de 50% de débit par rapport a sont prédécesseur H.264 pour une même qualité vidéo. Ces performances sont rendues possibles grâce à des changements sur plusieurs volets comme par exemple dans la structure de codage , le nombre de modes de la prédiction spatiale (qui dépasse 33 modes), l'utilisation d'algorithmes d'estimation et de compensation de mouvement qui offrent plus de précision, l'utilisation de la DCT exacte et de la DST (Discrete Sinus Transform), pour des informations résiduelles de tailles 4 × 4.

Le codage HEVC est de type hybride qui se base sur un schéma de prédiction Inter/Intra et la transformations du signal vidéo. Les prédictions Inter et Intra sont dédiées à l'élimination des redondances spatiales et temporelles dans la même trame ou dans des trames successives, alors que la transformation est employée pour décorréler le signal résiduel après prédiction. De même, comme tous ses prédécesseurs, le gain en efficacité de codage se fait au détriment d'une complexité croissante [19–20].

H.266 (VVC) : H.266/VVC ou MPEG, ou simplement VVC (Versatile Video Coding), est un codec vidéo publié le 6 juillet 2020 par le Joint Video Experts Team (JVET).Il a été développé par l'Institut Fraunhofer pour les télécommunications en partenariat avec des industriels [21]. Il est annoncé pour offrir un taux de compression deux fois supérieur au H.265/HEVC pour une qualité identique.

b) Comparaison entre les standards H.264, H.265 et H.266

Le tableau 1.1 souligne les principales caractéristiques de la norme de compression vidéo H.266/VVC et la norme antérieure H.265/HEVC.

Tableau 1.1 : Comparaison des standards H.264, H.265 et H.266 [20].

Éléments de codage	**H.264/AVC**	**H.265/HEVC**	**H.266/VVC**
Date d'édition	2003	2013	2020
Débit/ Espace	-44 -59% /H.263	-44 - 59% /H.264	-44 - 59% /H.265
Modèle de prédiction	**Prédiction temporelle :** • Partition MB et sub-MB pour l'estimation de mouvement ; • MB de taille 6x16 pixels.	**Prédiction temporelle :** • Partition CTU (CU, PU, TU). • CTU de taille 64x64 pixels	**Prédiction temporelle :** •unité de codage: 4x4 to 128x128 • taille du pipeline64x64
	Prédiction spatiale : • Intra Prédiction ; • 9 modes.	**Prédiction spatiale :** • Intra Prédiction ; • 35 modes.	**Prédiction spatiale :** • 81 modes: 65 angulaire+ planaire + DC + Sub • 6-MPM • MRL, Matrice pondérée pred Chroma à partir de Luma, PDPC, grand angle combiné partitions, Sub-partitions
Codage entropique	CABAC et CAVLC	CABAC	• CABAC + quantification scalaire dépendante • LUT basée sur CAE
Filtrage en boucle	Anti-bloc	Anti-bloc et SAO	Déblocage SAO • ALF • LMCS
Spécification	• Définition jusqu'à HD ; • Vitesse jusqu'à 60 fps.	• Support jusqu'à 8K (8192×4320) ; • Support jusqu'à 300 fps.	Résolutions d'images allant de 4k à8k ainsi que des vidéos à 360 degrés

1.4 Codeur VVC/H.266

1.4.1 Définition

Le H.266 est un codeur vidéo polyvalent qui doit réduire la taille des vidéos de moitié par rapport à prédécesseur le H.265 / HEVC pour une qualité vidéo équivalente. L'institut Fraunhofer Heinrich-Hertz vient d'annoncer la finalisation de ce nouveau standard nommé H.266 /VVC. C'est un format adapté à l'arrivée des vidéos en définition 8K, de la même

manière que le H.264 avec le déploiement de la définition Full HD et le H.265 avec la définition 4K [22].

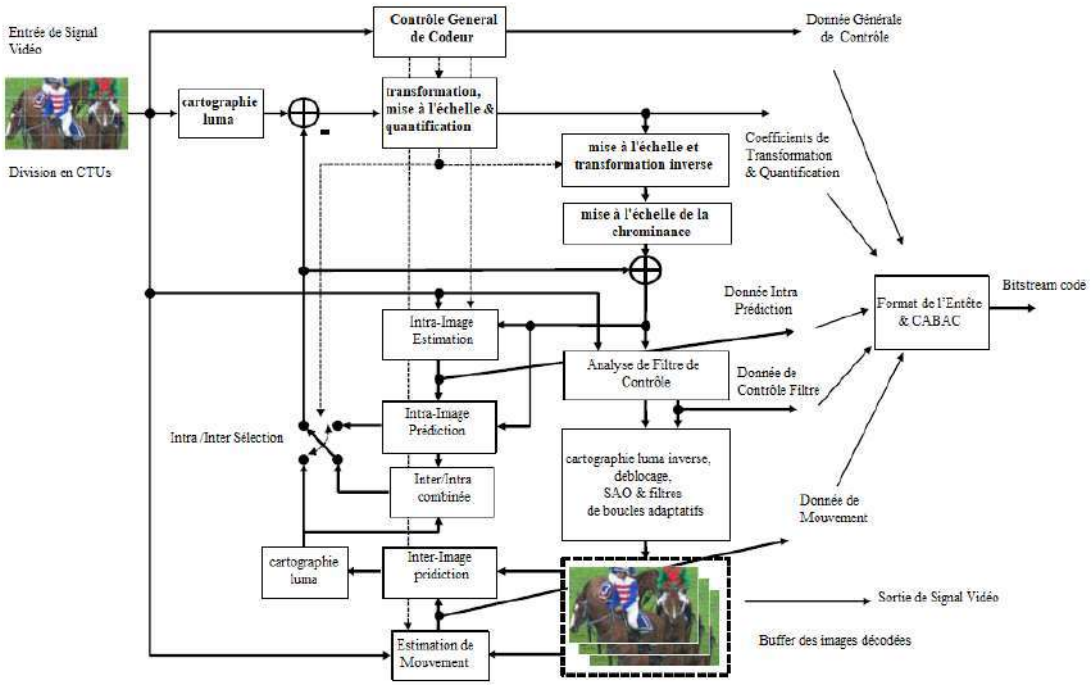


Figure 1.6 : Schéma bloc du codec vidéo VVC/H.266 [23].

1.4.2 Caractéristiques de la norme H.266

En plus d'une meilleure gestion des définitions d'images importantes, le format H.266 est adapté aux vidéos en HDR (High Dynamic Range) mais aussi aux vidéos filmées à 360 degrés. Le standard H.266 permet également de supporter la définition 16K, des fréquences de rafraichissement variables jusqu'à 120 Hz des formats panoramiques.

Selon l'institut Fraunhofer Heinrich-Hertz, le format H.265 requiert 10 Go de données pour une vidéo en 4K UHD de 90 minutes, mais seulement 5 Go avec le nouveau standard H.266 [22].

Le H.266 A basé sur différentes Techniques de compression :

1.4.3 Transformation

Le concept de sélection de transformations multiples (MTS) du VVC définit trois types de transformations trigonométriques séparables : la transformée discrète en cosinus (DCT)-II, VIII et la transformée discrète en sinus (DST)-VII.

Comme l'illustre la figure 1.7, le concept MTS sélectionne, pour les éléments suivants blocs Luma de taille inférieure à 64, l'ensemble de transformées qui minimise le coût de la distorsion du taux parmi cinq ensembles de transformées et le saut de blocs [24].

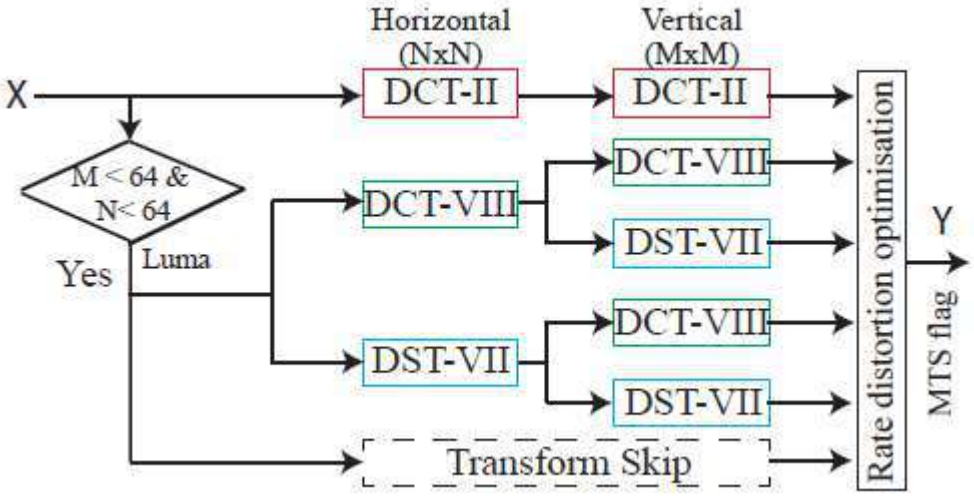


Figure 1.7 : Concept de sélection de transformées séparables 2D dans VVC.

1.4.4 Quantification

Le but principal de la quantification est la réduction de la dynamique des données d'une image après l'étape de la transformation en supprimant les valeurs ou bien les coefficients insignifiants des hautes fréquences puisque l'oeil humain est plus sensible aux erreurs de reconstruction liées aux basses fréquences qu'à celles liées aux hautes fréquences.

Pour la quantification, le VVC utilise essentiellement la même procédure de quantification, contrôlée par un paramètre de quantification (*QP*) comme dans la norme HEVC. La gamme des valeurs QP est définie entre (51-63), et une augmentation de 6 fois de la taille de pas de quantification. De plus, le QP a une influence directe sur la qualité visuelle et le taux de compression des images ou de vidéo. Par conséquent, si le QP est faible ou égal à 1, la reconstruction est parfaite et le taux de compression est très faible et inversement [25].

De plus, la quantification consiste à diviser les coefficients par le Qstep (*pas de quantification*), tandis que la quantification inverse consiste à une multiplication par le Qstep. En effet, le Qstep donné par l'équation (1.1) permet à l'encodeur la possibilité d'un contrôle exacte et souple pour améliorer le compromis entre le débit binaire et la qualité visuelle .

$$QStep = 2^{(QP-4)/6} \qquad (1.1)$$

1.4.5 Prédiction

Deux types de prédiction sont adoptées par le codec H.266 l'intra prédiction et l'inter prédiction. La Figure (1.8) montre que l'intra prédiction est utilisée pour éliminer les

redondances spatiales entre les blocs de la même image et l'inter prédiction pour éliminer les redondances temporelles entre images successives.

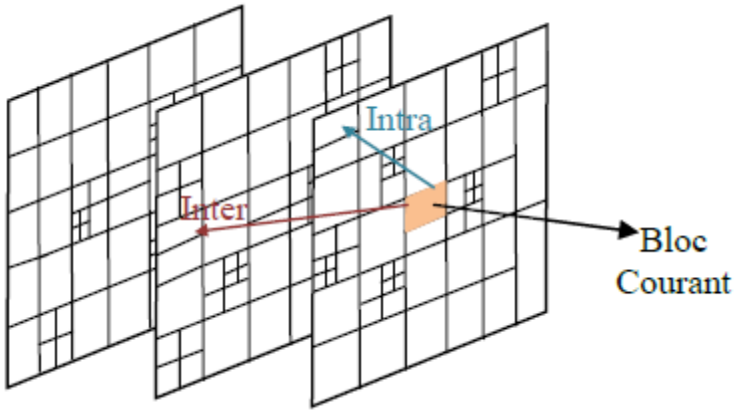


Figure 1.8 : Intra et Inter prédiction d'un block PU dans une séquence vidéo [8].

a) Inter prédiction

La prédiction inter exploite la redondance temporelle dans la séquence vidéo. Elle prédit les échantillons d'une unité de codage à partir d'échantillons d'images de référence précédemment codées. Le processus consistant à trouver le bloc le plus similaire dans les images de référence est appelé estimation de mouvement (ME). Le ME repose sur la correspondance des blocs et constitue l'une des opérations les plus complexes du point de vue du calcul dans le processus de codage.

Elle produit un ou plusieurs vecteurs de mouvement (MV), chacun composé de d'une composante verticale et d'une composante horizontale, représentant la translation de l'UC du cadre de référence au cadre actuel [24].

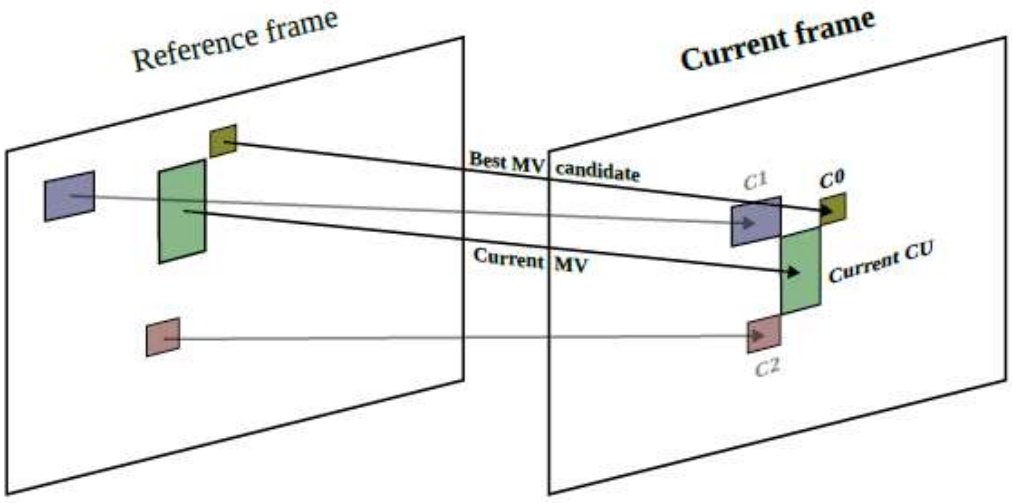


Figure 1.9 : Exemple simplifié de prédiction avancée du vecteur de mouvement.

b) Intra prédiction

L'intra prédiction angulaire est une méthode d'intra prédiction directionnelle prise en charge à la fois par AVC et HEVC et qui fait également partie du VVC. Par rapport au HEVC, l'intra prédiction angulaire du VVC a été modifiée. En augmentant la précision de la prédiction et en s'adaptant aux besoins de l'utilisateur au nouveau cadre de partitionnement. De plus, le nombre de modes directionnels disponibles pour un bloc donné passe de 33 directions dans la norme HEVC à 65 dire dans la norme VVC.

Les directions ayant des indices pairs entre 2 et 66 sont équivalentes aux directions des modes angulaires supportés par le HEVC (figure 1.10).

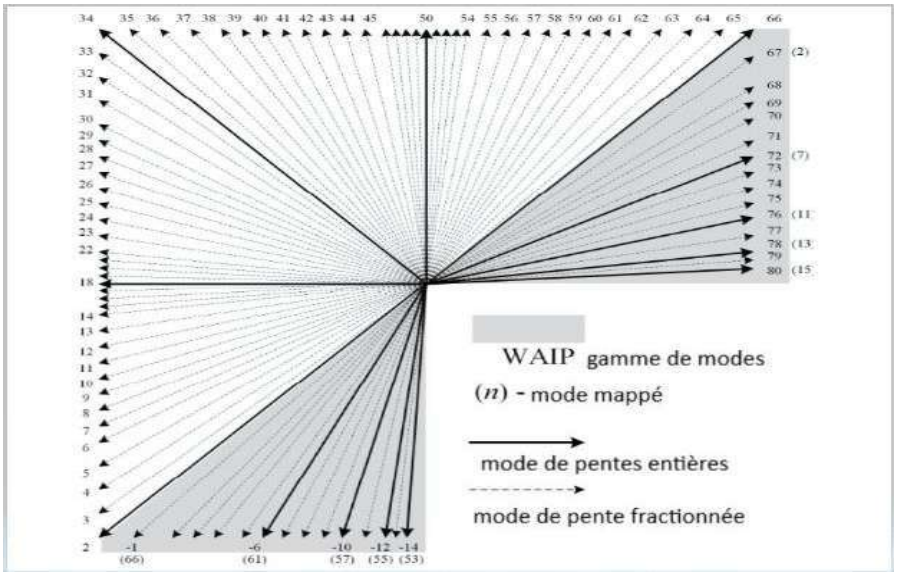


Figure 1.10 : 65 directions de l'intra-prédiction utilisés pour le VVC [26].

1.4.6 CABAC

La nouvelle version du codage arithmétique binaire adaptatif basé sur le contexte (CABAC) utilisé dans la dernière technologie de codage vidéo polyvalent (VVC) est à la pointe de la technologie dans le domaine de l'entropie. Il assure une estimation encore plus précise des probabilités des symboles de données codées [27].

Le CABAC est appliqué en trois étapes : premièrement, il convertit tous les symboles non binaires en des symboles binaires (*bins*) car le CABAC est basé sur un codage binaire (binarisation). Puis pour chaque bit, un modèle de probabilité le plus adapté est sélectionné par la CSM (context selection Modling) en se basant sur les données de voisinage pour optimiser l'estimation de cette probabilité. Enfin un codage arithmétique binaire (BAC) est appliqué pour

compresser les symboles binaires selon le mode de codage utilisé au niveau de décodage les opérations inverses de ces trois étapes sont a appliquées.

Dans notre Travail on s'intéresse à la partie l'implémentation logicielle et matérielle de la partie BAD.

1.5 Conclusion

Dans ce chapitre, les normes de compression vidéo récentes ont été présentées ainsi que les techniques associées à la norme de compression VVC ont été détaillées, à savoir la transformation en cosinus discret, la quantification, l'intra et l'inter prédiction, et le codage entropique CABAC. Une étude détaillée du CABAC VVC fera l'objet du chapitre prochain.

Chapitre 02

2.1 Introduction

CABAC est la technique de codage entropique utilisée dans la norme AVC, HEVC et VVC, avec une haute efficacité de codage. Il repose principalement sur trois fonctions : la binarisation, la modélisation de contexte et le codage arithmétique. Dans ce chapitre, nous commencerons par décrire quelques concepts de base des systèmes de communication et de la théorie de l'information. Ensuite, nous nous concentrons sur le codage entropique et ses types. A la fin du chapitre le CABAC/VVC sera détaillé.

2.2 Système de communication

La théorie des communications s'intéresse aux moyens de transmettre une information depuis la source jusqu'à un utilisateur à travers un canal. La nature de la source peut être très variée. Il peut s'agir par exemple d'une voix, d'un signal électromagnétique ou d'une séquence de symboles binaires.

Le canal peut être une ligne téléphonique, une liaison radio, un support magnétique ou optique. Le codeur représente l'ensemble des opérations effectuées sur la sortie de la source avant la transmission. Ces opérations peuvent être par exemple la modulation, la compression, le brouillage, l'ajout de redondance pour combattre les effets du bruit, ou encore l'adaptation à des contraintes de spectre. Elles ont pour but de rendre la sortie de la source compatible avec le canal. Enfin, le décodeur doit être capable, à partir de la sortie du canal, de restituer de façon acceptable l'information fournie par la source [28].

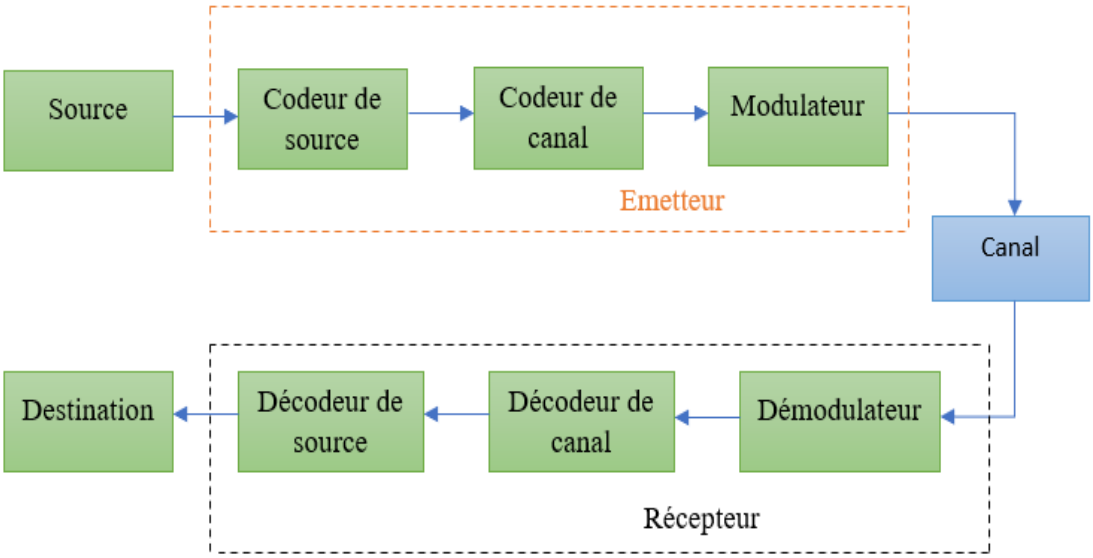


Figure 2.1 : Système de communication d'informations [8].

2.2.1 Codage source

Le code source est un texte qui détaille les instructions d'un programme informatique dans un langage de programmation compréhensible et utilisable. Il traduit les instructions qu'a souhaité laisser le programmeur lors de la conception du programme.

La plupart du temps, le code source se résume à un ensemble de fichiers textes. Une fois écrit, il déclenche un code binaire essentiel pour qu'un processeur puisse exécuter une tâche précise. Sinon, dans certains cas, il peut être directement interprété, ce qui engendre une exécution immédiate, ou encore, il arrive qu'il soit transmis sous forme de code intermédiaire, c'est-à-dire un code faisant le lien avec les instructions machine. Toute page internet, ou tout logiciel possède son code source.

De plus, les applications multimédias actuelles traitent des contenus hétérogènes, qui se caractérisent par un degré élevé de redondance des informations. Pour pouvoir les transmettre sur des canaux physiques et respecter les contraintes de temps réel, il est nécessaire de réduire cette redondance d'informations [29]. Le codage source, ou compression des données, est utilisé pour fournir une représentation efficace des données à des taux de compression élevés, tout en préservant les informations pertinentes qu'elles contiennent. Peut être fait avec ou sans perte d'information.

2.2.2 Codage canal

Le codage canal a pour rôle de protéger l’information émise contre les perturbations du canal de transmission susceptibles de modifier son contenu. Il s’agit donc de rajouter de la redondance de manière à détecter et éventuellement corriger les erreurs lors de la réception si la stratégie adoptée le permet. Les conditions d’un codage correct sont déterminées par le second théorème de Shannon [30].

2.2.3 Modulation

En télécommunications, le signal transportant une information doit passer par un moyen de transmission entre un émetteur et un récepteur. Le signal est rarement adapté à la transmission directe par le canal de communication choisi, hertzien, filaire, ou optique. La modulation peut être définie comme le processus par lequel le signal est transformé de sa forme originale en une forme adaptée au canal de transmission, par exemple en faisant varier les paramètres d'amplitude et d'argument (phase/fréquence) d'une onde sinusoïdale appelée porteuse.

Le dispositif qui effectue cette modulation, est en général électronique, L'opération inverse permettant d'extraire le signal de la porteuse est la démodulation. Deux types de modulation existent, à savoir :

a) Modulation analogique

L'objectif des modulations analogiques est d'assurer la qualité suffisante de transmission d'une information analogique (voix, musique, image) dans les limites du canal utilisé et de l'application. Trois types de modulation analogiques sont utilisées, à savoir :

• Modulation d'amplitude AM ;

• Modulation de la fréquence FM ;

• Modulation de la phase PM ;

b) Modulation numérique

L'objectif des modulations numériques est d'assurer un débit maximum de données binaires, avec un taux d'erreur acceptable. La modulation est l'élément principal de la couche physique, divers types de modulations numériques sont utilisées, à savoir :

• Modulation d'amplitude ASK (*Amplitude-shift keying*) ;

• Modulation de la fréquence FSK (*Frequency-shift keying*) ;

• Modulation de la phase PSK (*Phase-shift keying*);

• Modulation de la phase et de l'amplitude APK (*Amplitude-phase keying*).

2.3 Généralités sur la théorie de l'information

La théorie de l'information a été créée par Shannon dans les années 40. Il s'agit d'une théorie mathématique qui décrit les aspects les plus fondamentaux des systèmes de communication. Elle consiste en l'élaboration et l'étude de modèles pour la source et le canal qui utilisent différents outils comme les probabilités et les automates finis [28].

Parmi les branches importantes de la théorie de l'information de Shannon, on peut citer :

• Le codage d'information ;

• La compression de données ;

• La mesure quantitative de redondance d'un texte ;

• La cryptographie [8].

2.3.1 Définition de l'information

Le concept théorique d'information a été introduit par Shannon à partir de recherches théoriques sur les systèmes de télécommunication. Ainsi, ce concept a été l'objet de la théorie de l'information alors qu'il s'agissait, en réalité d'une théorie mathématique de la télécommunication de l'information, or cette expression est exactement celle de Shannon et Weaver. L'information propre « x » généré par une source d'information « S » est une fonction de sa probabilité. Elle est donnée par l'équation (Eq.2.1), l'information x est constituée par des symboles xi [8].

$$I(x) = f(p(xi)) = -\log2\ (p(xi)) \qquad (2.1)$$

Où p(xi) est la probabilité du symbole xi, i= 1…….,n.

2.3.2 Entropie

Le terme entropie a été introduit en 1865 par Rudolf Clausius à partir d'un mot grec signifiant « transformation ». En thermodynamique classique, l'entropie est définit comme une fonction d'état proportionnelle à la quantité de matière en présence. Elle caractérise le désordre microscopique d'un système ou son degré de désorganisation. L'entropie d'un système rend donc compte du degré de dispersion de l'énergie (thermique, chimique, etc.) au sein même du système [8].

Par analogie, l'entropie en théorie de l'information caractérise le degré de désorganisation ou d'imprédictibilité du contenu en information d'une source.

Mathématiquement l'entropie est une fonction qui définit la quantité d'informations moyenne contenue ou délivrée par une source d'information. L'entropie notée H (x) est calculée en fonction de la probabilité des symboles xi et donnée par l'équation (Eq.2.2) :

$$H(x) = -\Sigma\ p\ (xi)\ \log2\ (p\ (xi)\ )\ ni\ =1 \qquad (2.2)$$

Où p(xi) est la probabilité du symbole xi, i= 1, ……, n.

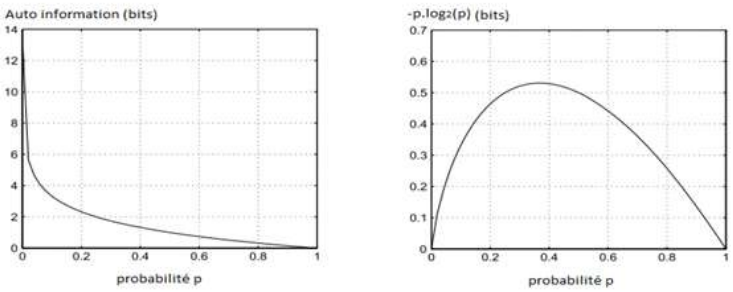


Figure 2.2 : Auto-information et probabilité. d'un seul événement et d'un événement pondéré par sa probabilité.

2.3.3 Codage entropique

Le codage entropique de manière générale consiste à transformer le message d'une source en une séquence d'information de façon à minimiser la taille du message en éliminant les redondances, tout en gardant la possibilité de retrouver le message d'origine à partir de la séquence de substitution. Le but de cette opération est d'optimiser les ressources nécessaires à la transmission (temps, puissance, bande passante, etc.). Dans les sous sections suivantes nous allons détailler quelques types de codage à longueur fixe et à longueur variable.

2.3.3.1 Codage VLC

Le codage à longueur variable VLC (*Variable Length Code*) transforme les symboles d'entrée en une série de mots de code variable qui contiennent un nombre entier de bits. Son fonctionnement est basé sur le codage des symboles les plus fréquents avec des mots de code de faible taille, tandis que les symboles moins fréquents sont représentés avec des mots de code de taille plus longue. Pour un nombre suffisamment grand de symbole, cela conduit à la compression moyenne des données [8].

2.3.3.2 Codage de Shannon-fano

Le codage de Shannon-Fano [31] est développé à partir des propriétés statistiques des messages. Cette méthode se base sur le partage des symboles en deux groupes de valeur quasi équivalente. Pour chaque groupe, on attribue la somme des probabilités d'apparition des symboles qu'il contient. On fait un choix arbitraire pour désigner un des groupes par « 0 » et l'autre par« 1 ».De même, ces groupes sont divisés en deux et indiqués par 0 ou 1 jusqu'au avoir un seul symbole dans chaque groupe. Cette méthode forme alors un arbre binaire constitué de branches (bits d'informations 0 ou 1) et de feuilles (caractères). Le code d'un caractère est déterminé en partant du sommet et en parcourant les branches jusqu'à la feuille correspondante. Les caractères près du sommet sont alors les plus fréquents et nécessitent moins de bits de codage.

2.3.3.3 Codage de Huffman

Le codage de Huffman est un algorithme de compression de données sans perte. Il a été inventé par David Albert Huffman, et publié en 1952 [32]. Le codage de Huffman utilise un code à longueur variable pour représenter un symbole de la source (par exemple un caractère dans un fichier). Le code est déterminé à partir d'une estimation des probabilités d'apparition des symboles de source, un code court étant associé aux symboles de source les plus fréquents.

Un code de Huffman est optimal au sens de la plus courte longueur pour un codage par symbole, et une distribution de probabilité connue. Des méthodes plus complexes réalisant une modélisation probabiliste de la source permettent d'obtenir de meilleurs taux de compression.

2.3.3.4 Codage RLE (Run- length Encoding)

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

2.3.3.5 Codage arithmétique

Ce codeur permet de construire un seul mot de code à partir d'une longue séquence arbitraire de symboles. L'algorithme consiste à attribuer un mot de code à chaque ensemble de données. Ces mots de code correspondent à des sous-intervalles semi-ouverts de l'intervalle unité semi-ouvert [0,1[. Les codes les plus courts concordent aux sous-intervalles les plus larges c'est à dire aux ensembles de données les plus probables [33]. L'algorithme de codage arithmétique est composé les étapes suivantes:

1) Diviser l'intervalle unité semi-ouvert [0,1[en N sous-intervalles correspondant aux N symboles possibles. La taille de chaque sous-intervalle est relative à la probabilité évaluée que le symbole correspondant soit le suivant dans la suite à coder.

2) Sélectionner le sous-intervalle, correspondant au premier symbole de la série à coder et le diviser en N sous-intervalles corrélativement aux probabilités des symboles.

3) Refaire l'étape 2 jusqu'au dernier symbole dans la suite à coder.

4) Arborer les limites inférieure et supérieure du sous-intervalle correspondant au dernier symbole de la suite par leurs valeurs en fraction binaire et choisir le nombre binaire qui distingue le dernier sous-intervalle.

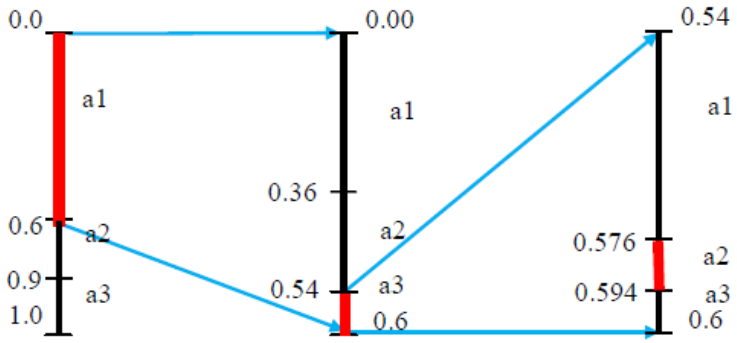


Figure 2.3 : Exemple de codage arithmétique [8].

Récemment, les codes arithmétiques ont attirés l'attention des chercheurs dans le domaine de la compression par leurs utilisations dans des standards de compression tels MPEG4, H.264, H.265, et H.266. Malgré sa complexité d'implémentation et grâce à son efficacité de compression, le codage arithmétique, dans sa version adaptative CABAC (*Context Adaptative Binary Arithmetic Coding*) a été sélectionné par la norme de compression H.266/VVC.

2.4 Codage CABAC VVC

CABAC combine un codage arithmétique binaire adaptatif avec une modélisation du Contexte. L'encodage d'un élément par CABAC se déroule en trois étapes principales :

- Binarisation.
- Sélection et modélisation du contexte.
- Codeur arithmétique binaire.

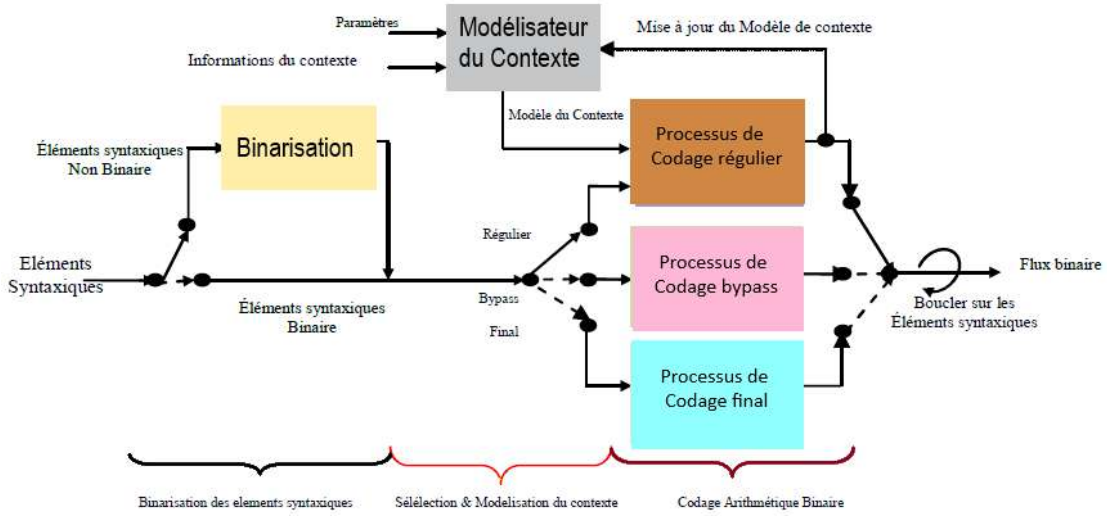


Figure 2. 4 : Architecture du codeur CABAC.

2.4.1 Binarisation

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

2.4.2 Sélection et modélisation du contexte

Une des propriétés les plus importantes du codage arithmétique est la possibilité d'utiliser une interface entre la modélisation et le codage. La modélisation permet d'assigner aux symboles un modèle de distribution de probabilité qui est ensuite utilisé par le codeur arithmétique pour produire une représentation de ces symboles en adéquation avec ce modèle de distribution.

Autrement dit, il s'agit d'adapter la probabilité d'apparition d'un symbole binaire à son contexte comme : type de l'information codée, valeurs des symboles déjà traités dans un voisinage spatio-temporel... Il est donc important d'avoir des modèles statistiques bien adaptés aux données et de les garder à jour tout au long du codage.

2.4.3 Codage arithmétique binaire

Le codage arithmétique binaire traite la séquence entière de symboles comme une seule entité basée uniquement sur la distribution de probabilité des symboles à coder. Utilisez le codage arithmétique adaptatif CABAC pour attribuer un nombre de décimales à chaque symbole à coder. Par conséquent, le code adaptatif généré est ensuite adapté aux statistiques des symboles variables [8]. Le codage arithmétique binaire est basé sur trois modes de codage :

•Mode normal ou régulier (*regular mode*) qui utilise le modèle du contexte;

•Mode bypass (*bypass mode*) basé sur la probabilité équivalente ;

•Mode terminal (*terminate mode*) appelé à la fin de processus de codage ou de décodage de chaque CTU (Coding Tree Unit).

2.5 Décodage CABAC

Le décodage arithmétique binaire d'un élément syntaxique se déroule en trois étapes principales. La figure 2.5 illustre le schéma synoptique de décodage CABAC VVC qui est basé sur les opérations suivantes :

• Sélection et modélisation du contexte ;

• Décodage arithmétique binaire pour les trois modes : Bypass, régulier et final.

• Dé-binarisation ;

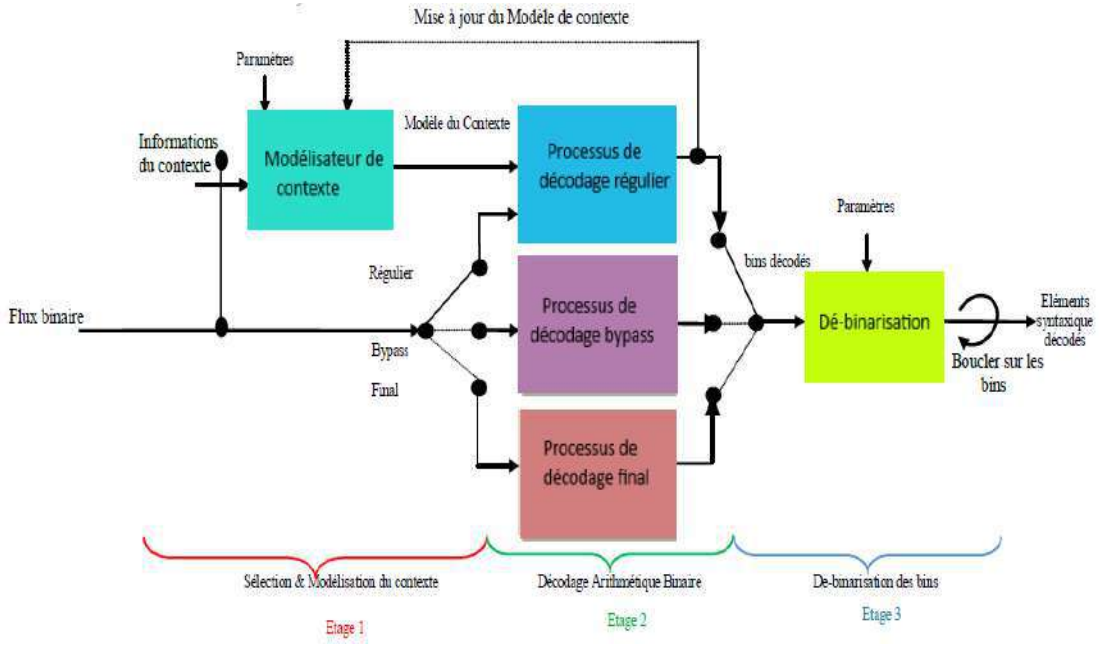


Figure 2.5: Architecture du décodeur CABAC.

2.5.1 Sélection et modélisation du contexte

Une des propriétés les plus importantes du CABAC VVC est sa capacité à utiliser une interface propre entre la modélisation des probabilités et la procédure de codage. Dans l'étape de modélisation, un modèle de probabilité appelé contexte, spécifiant la probabilité de *LPS* (Least Probable Symbol) et *MPS* (Most Probable Symbol), est attribué à la case actuelle. Chaque contexte se compose d'un *valMPS* et d'un *pStateIdx*, où *valMPS* définit ce qu'est MPS et *pStateIdx* indique l'état de probabilité de LPS. Dans l'étape de codage suivante, le moteur de codage proprement dit est alors appliqué pour générer une séquence de bits en tant que représentation codée de selon le contexte sélectionné. Puisque le contexte détermine le flux binaire codé et son efficience, il est crucial de concevoir un modèle adéquat qui explore les dépendances statistiques dans une large mesure et maintient ce modèle à jour pendant la procédure de codage.

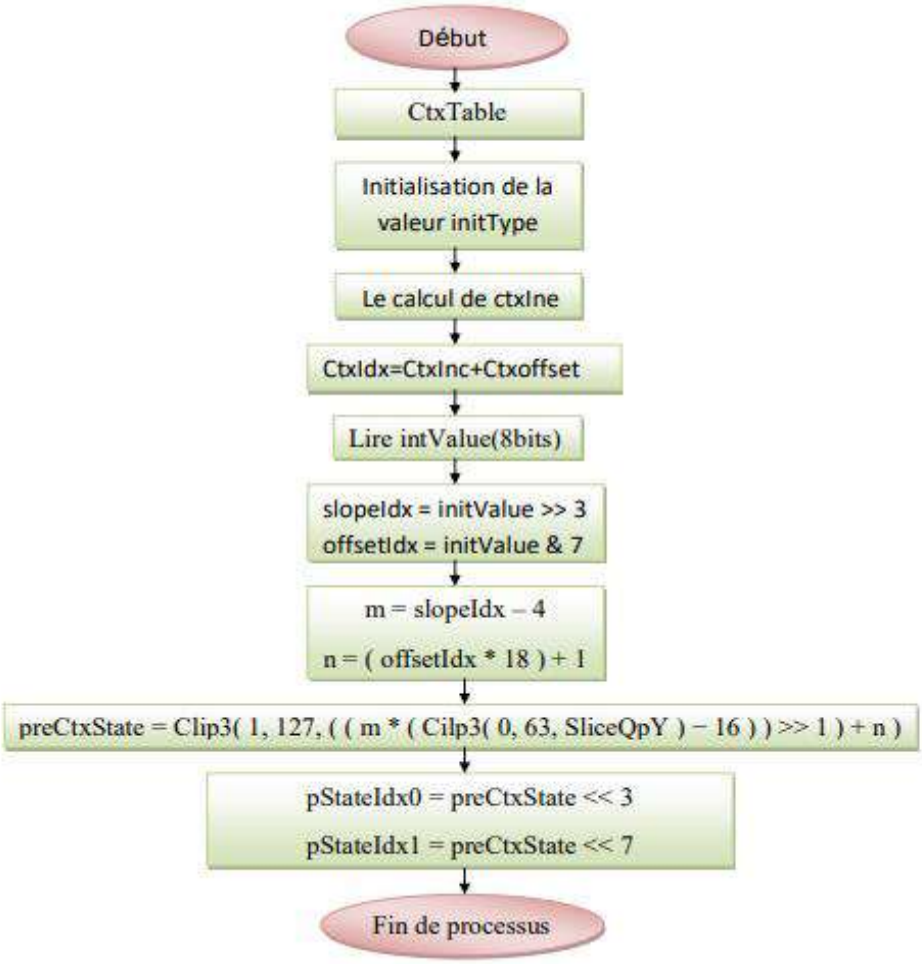


Figure 2.6: Organigramme de sélection et modélisation du contexte (codage/décodage) [34].

2.5.2 Décodage arithmétique binaire

Le décodeur arithmétique décode chaque bin selon trois modes : régulier, bypass et terminal. Dans le mode régulier le décodage du bin est effectué selon le modèle de probabilité sélectionné. Les deux modes de décodage bypass et terminal sont exécutés sans modélisation du contexte.

2.5.2.1 Mode régulier

Ce mode est basé sur le modèle de contexte obtenu avec une précision de l'estimation de la probabilité pour obtenir un codage efficace. La variable *ctxIdx* est calculée pour chaque *bin* en fonction du type d'élément syntaxique à décoder, du *bin* index (*binIdx*) et des propriétés des unités de codage spatialement voisines.

Le décodage arithmétique binaire est basé sur la division récursive d'intervalle. L'intervalle est initialisé entre la valeur « 0 » et la valeur de « *Range* ». Il est divisé en deux sous-intervalles basés sur la probabilité des symboles *MPS* et *LPS* Figure 2.7.

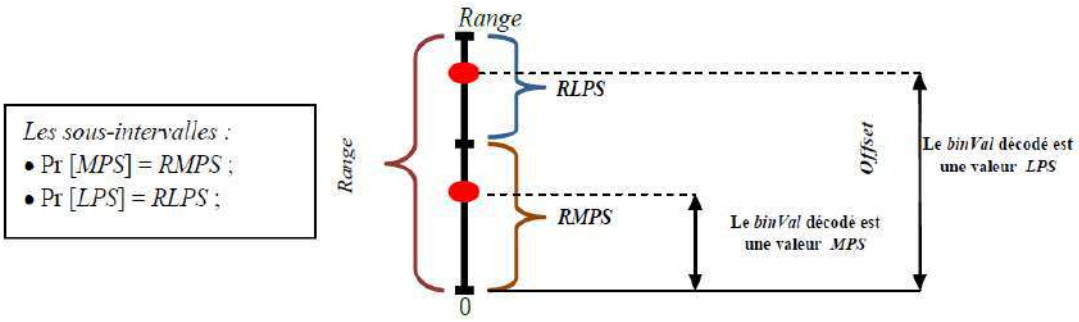


Figure 2.7: Division des sous intervalles dans le processus de décodage arithmétique (mode régulier) [8].

2.5.2.2 Mode bypass

Ce mode de décodage est la partie la moins complexe dans le décodage arithmétique binaire. Aucun modèle de contexte n'est requis et la possibilité d'apparence de MPS est égale à la possibilité d'apparence de LPS.

2.5.2.3 Mode terminal

Le mode terminal est appelé uniquement à la fin de chaque CTU pour vérifier si cette CTU est la dernière dans le slice courant ou non. Si le dernier symbole de l'élément syntaxique est égal à 0, le mode régulier est en cours de traitement, sinon le dernier symbole est égal à 1, alors le mode terminal est en cours de traitement [8].

2.5.3 Dé-binarisation

La dé-binarisation est l'opération inverse du processus de binarisation appliqué au niveau de l'encodage. Il est situé en sortie du décodeur CABAC et comprend des éléments de syntaxe de reconstruction de bin obtenus à partir de la sortie du décodeur selon :

- Schémas de binarisation employés à l'encodage ;
- Type de l'élément syntaxique décodé ;
- Ordre de décodage.

2.6 Conclusion

Dans ce chapitre, les différents modules du système de communication et la théorie de l'information ont été introduit. Ensuite, le codage entropique a été présenté avec quelques types de codage existants. Ensuite, nous avons aussi montré le processus de codage et décodage CABAC dans la norme VVC.

Chapitre 03

3.1 Introduction

La norme VVC définit trois modes de décodage arithmétique binaire (Mode régulier, Mode bypass, Mode terminal), nous commencerons ce chapitre par les outils logiciels utilisés dans notre Travail. Ensuite, nous présenterons les organigrammes proposés des modes de décodage arithmétique binaire avec leurs simulations sous Matlab, il est à signaler que le but de ce chapitre est de maitriser les algorithmes de décodage entropique CABAC avant de passer à la description matérielle en langage VHDL sous l'ISE.

3.2 Outils utilisés

Il est essentiel de présenter les différents outils matériels et logiciels ainsi que leurs versions utilisées dans ce travail car chaque outil a ses caractéristiques et ses fonctions spécifiques. Nous avons porté notre choix sur le langage de description matérielle VHDL (*Hardware Description Language)* pour la description matérielle du Décodage Arithmétique Binaire de la dernière norme de compression vidéo VVC sur plateforme FPGA *(Field Programmable Gate Array).*

Nous avons utilisé le logiciel ISE pour la description matérielle de notre algorithme, le Model -Sim pour la simulation et le Matlab pour la validation de l'architecture proposée.



Figure 3.1 : Outils matériels et logiciels utilisés dans notre projet.

3.2.1 Model-Sim (PE Student Edition 6.3)

Model-Sim est un outil de Mentor graphiques qui fournit un environnement complet de simulation et de débogage pour les designs complexes en ASIC *(Application Specifie Integrated Circuits)* et en FPGA.

3.2.2 MATLAB

Matlab est un logiciel de calcul numérique commercialisé par la société Math Works pour permettre aux étudiants de travailler à partir d'un outil de programmation de haut niveau. Matlab signifie Matrix laboratory. De plus, c'est un langage pour le calcul scientifique de l'analyse de données de visualisation et de développement d'algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution de commandes, et d'autre part, un environnement de développement intégré (IDE).

3.2.3 Xilinx ISE 14.7

ISE (*Integrated Software Environment*) est un environnement de développement intégré. pour la synthèse et l'analyse des conceptions des systèmes électroniques. Il permet au développeur de synthétiser leurs conceptions, d'effectuer une analyse de timing, d'examiner les diagrammes RTL *(Register Transfer Level)*, de simuler la réaction d'une conception et de configurer le périphérique cible avec le programmateur. Ainsi, il permet de réaliser toutes les phases du flot de conception d'applications sur des composants reconfigurables *FPGA*.

Il est utilisé pour la description des systèmes à plusieurs niveaux, la synthèse niveau RTL, la simulation par composant, un outil d'aide à la conception et aussi il assure l'extensibilité des architectures [8]. La figure (3.2) illustre l'entité d'un port AND a un bit d'entrée.

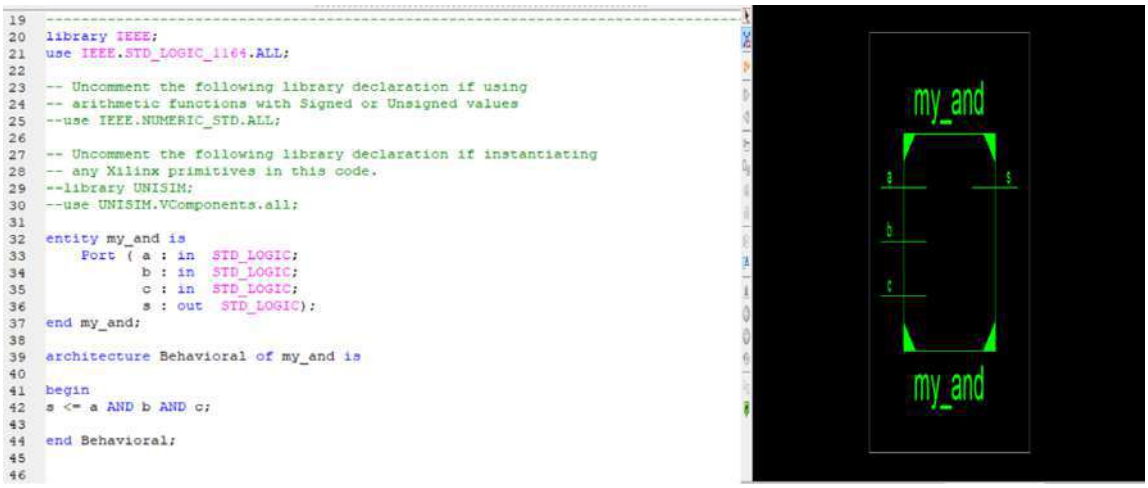


Figure 3.2 : Entité et architecture d'un composant logique sur ISE.

3.3 Algorithme de Décodage CABAC/VVC

Le décodeur arithmétique décode chaque *bin* selon trois modes : régulier, bypass et terminal. Dans le mode régulier le décodage du *bin* est effectué selon le modèle de probabilité sélectionné. Les deux modes de décodage bypass et terminal sont exécutés sans modélisation du contexte.

3.3.1 Mode régulier

Les entrées de ce processus sont les variables *ctxTable*, *ctxIdx*, *ivlCurrRange*, et *ivlOffset*.Les sorties de ce processus sont la valeur décodée *binVal*, et la mise a jour des variables *ivlCurrRange* et *ivlOffset* dans le mode :

1) La valeur de la variable *ivlLpsRange* est dérivée comme suit :

- Étant donné la valeur actuelle de *ivlCurrRange*, la variable *qRangeIdx* est dérivée comme suit:

qRangeIdx = *ivlCurrRange*>> 5

- Étant donné *qRangeIdx*, pStateIdx0 et pStateIdx1 associés à *ctxTable* et *ctxIdx*, *valMps* et *ivlLpsRange* sont dérivés comme suit :

pState = *pStateIdx1* + 16 * *pStateIdx0* *valMps* = *pState*>> 14

ivlLpsRange = (*qRangeIdx* * ((*valMps* ? 32767 - *pState* : *pState*) >> 9) >> 1) + 4

2) La variable *ivlCurrRange* est définie comme étant égale à *ivlCurrRange* - *ivlLpsRange* et ce qui suit s'applique :

- Si *ivlOffset* est supérieur ou égal à *ivlCurrRange*, la variable *binVal* est égale à 1 - *valMps*, *ivlOffset* est décrémenté de *ivlCurrRange*, et *ivlCurrRange* est égale à *ivlLpsRange*.
- Sinon, la variable *binVal* est égale à *valMps* [23].

3) Processus de transition d'état Les entrées de ce processus sont les pStateIdx0 et pStateIdx1 actuels, et la valeur décodée *binVal*.

Les sorties de ce processus sont les pStateIdx0 et pStateIdx1 mis à jour de la variable de contexte associée à *ctxTable* et *ctxIdx*.

Les variables shift0 et shift1 sont dérivées de la valeur *shiftIdx* associée à *ctxTable* et *ctxIdx*

shift0 = (*shiftIdx* >> 2) + 2 ; *shift1* = (*shiftIdx* & 3) + 3 + *shift0*

En fonction de la valeur décodée *binVal*, la mise à jour des deux variables pStateIdx0 et pStateIdx1 associées à *ctxTable* et *ctxIdx* est dérivée comme suit

pStateIdx0 = *pStateIdx0* − (*pStateIdx0* >> *shift0*) + (1023 * *binVal* >> *shift0*)

pStateIdx1 = *pStateIdx1* − (*pStateIdx1* >> *shift1*) + (16383 * *binVal* >> *shift1*)

4) Processus de renormalisation dans le moteur de décodage arithmétique

À la fin de ce processus de décodage, Les entrées de ce processus sont les bits des données de la tranche et les variables *ivlCurrRange* et *ivlOffset*. Les sorties de ce processus sont les variables mises à jour *ivlCurrRange* et *ivlOffset*.

Un organigramme de la renormalisation est présenté après la mise à jour des deux variables *pStateIdx0* et *pStateIdx1* à la Figure 3.3. La valeur actuelle de *ivlCurrRange* est d'abord comparée à 256 et les étapes suivantes sont spécifiées comme suit :

- Si *ivlCurrRange* est supérieur ou égal à 256, aucune renormalisation n'est nécessaire et le processus renormalisation est terminé. Sinon (*ivlCurrRange* est inférieur à 256), la boucle de renormalisation est entrée. Dans cette boucle, la valeur de *ivlCurrRange* est doublée, c'est-à-dire décalée à gauche de 1 et un seul bit est décalé dans *ivlOffset* en utilisant *read_bits* (1).

Le flux binaire ne doit pas contenir de données dont la valeur de *ivlOffset* est supérieure ou égale à *ivlCurrRange* à la fin de ce processus [34].

3.3.1.1 Organigramme

Le processus de décodage arithmétique dans le mode régulier illustré sur la figure 3.3 est appliqué après l'obtention des valeurs du modèle de contexte (*pStateIdx, MPSVal*).

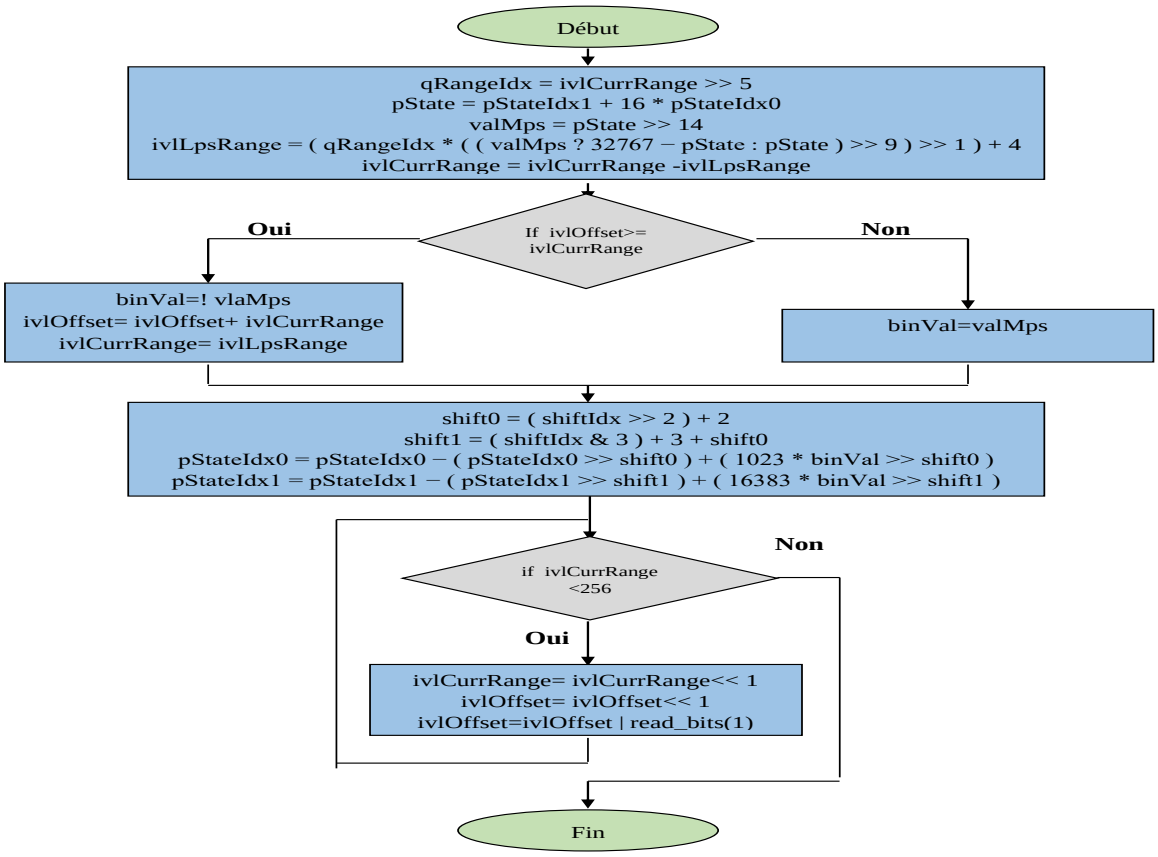


Figure 3.3 : Organigramme du mode régulier avec la renormalisation.

3.3.1.2 Programme sous Matlab

La figure 3.4 montre une partie programme réaliser sous matlab de l'algorithme CABAC/VVC mode Régulier RM.

```
%%%regular%%%mode/////////////////==============================


qRangeIdx = bitshift (ivlCurrRange,-5);
pState = pStateIdx1 + 16 * pStateIdx0;

valMps = bitshift (pState,-14);

if valMps == 0
    aaaaa = 32767;
else
    aaaaa = pState;
end

ivlLpsRange = bitshift(qRangeIdx * (bitshift(aaaaa,-9)),-1)+4;
ivlCurrRange = ivlCurrRange - ivlLpsRange;

% la_boucle-----------------------------------------------
if ivlOffset >= ivlCurrRange
     binVal = ~valMps;
     ivlOffset = ivlOffset + ivlCurrRange;
     ivlCurrRange = ivlLpsRange;

else
    binVal = valMps;

end
```

Figure 3.4 Partie du programme Matlab en mode régulier

3.3.1.3 la Simulation

Valeurs *binVal* sont obtenus pour chaque élément syntaxique et chaque intervalle (*ivlOffset*) :

```
binval =

     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     1     0     0     1     0     0     0     0     1     0     0     0     0     0     0     0
     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
```

Figure 3.5 : Simulation sous Matlab de Mode régulier (les valeurs *binVal*).

En fonction de la valeur décodée *binVal*, les variables *shift0*, *shift1*et la mise à jour des deux variables *pStateIdx0* et *pStateIdx1* est effectuée en fonction de la table de *ctxTable* et l'adresse de l'indexe *ctxIdx* :

```
Shift0 =

  Columns 1 through 18

     2     2     3     3     2     2     3     4     4     4     5     2     4     4     2     2     2     2

  Columns 19 through 36

     3     4     3     4     2     3     2     2     2     3     4     3     3     2     2     3     3     2

  Columns 37 through 54

     5     2     3     2     3     2     4     4     5    24     4     4     3     2     2     3     4     2

  Columns 55 through 72

     2     3     2     3     4     3     3     3     2     4     4     4     2     2     3     4     4     4

  Columns 73 through 74

     4     3
```

Figure 3.6 : Simulation saus Matlab en mode Régulier pour la variable *shift0*.

```
Shift1 =

  Columns 1 through 18

     5     5     6     6     5     5     6     7     7     8     8     5     7     7     6     6     6     6

  Columns 19 through 36

     6     8     7     8     7     8     6     6     5     6     8     7     6     5     5     6     7     5

  Columns 37 through 54

    10     5     6     6     6     6     9     7     9    29     7     8     7     5     5     6     9     5

  Columns 55 through 72

     6     6     5     6     8     7     7     7     7     7     7     7     6     6     6     7     7     8

  Columns 73 through 74

     9     7
```

Figure 3.7 : Simulation saus Matlab en mode Régulier pour la variable *shift1*.

```
pStateIdx000 =

 Columns 1 through 17

  597   414   386   386   261   558   134   431    71   401   287   261   161   518   405   261   438

 Columns 18 through 34

  261   302   491   134   131   525   554   261   261   285   595   101   246   134   261   261   302

 Columns 35 through 51

  218   261   318   510   302   261   134   621   341    71   132   104   221   341   386   621   261

 Columns 52 through 68

  134   401   558   414   498   261   246    71   554   134   134   309   251   251   251   261   261

 Columns 69 through 74

  134    71    71   221    71   134
```

Figure 3.8 : Simulation saus Matlab en mode Régulier pour la variable *pStateIdx0*.

```
Command Window
pStateIdx111 =

  Columns 1 through 9

       7579        8556        4917        4917         635       11532         381        6350         254

  Columns 10 through 18

       5801        4271         635        1778        8763        3405         381        9198         381

  Columns 19 through 27

       3405        7331         254        1211        5842        7841         381         381        1131

  Columns 28 through 36

      10710         701        2286         381         635         635        3405        1778         635

  Columns 37 through 45

       4747       10540        3405         381         381        7941        4758         254        1692

  Columns 46 through 54

       1664        2794        4781        4826        8075         635         381        5780       11532

  Columns 55 through 63

       8694        6933         635        2397         191        7874         254         254        1270
```

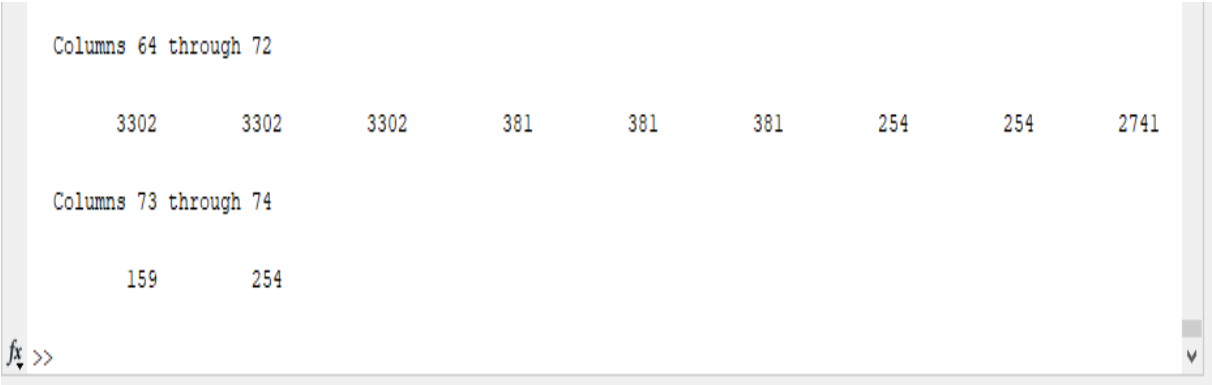


Figure 3.9 : Simulation sous Matlab en Mode Régulier pour la variable *pStateIdx1*.

3.3.2 Mode bypass

Le mode bypass est utilisé pour simplifier et accélérer le codage d'informations binaires caractérisées par des répartitions équiprobables comme les bits de signe, les bits les moins significatifs.

Les entrées de ce processus sont les bits de données et les variables *ivlCurrRange* et *ivlOffset*. Les sorties de ce processus sont : la mise a jours de la variable *ivlOffset* mise à jour et la valeur décodée *binVal*. Le processus de décodage par contournement est invoqué lorsque le drapeau *bypassFlag* est égal à 1. La figure 3.10 montre un organigramme du processus correspondant.

Tout d'abord, la valeur de *ivlOffset* est doublée, c'est-à-dire décalée à gauche de 1 et un seul bit est décalé dans *ivlOffset* en utilisant *read_bits* (1). Ensuite, la valeur de *ivlOffset* est comparée à la valeur de *ivlCurrRange* et les étapes suivantes sont spécifiées comme suit :

- Si *ivlOffset* est supérieur ou égal à *ivlCurrRange*, la variable *binVal* est mise à 1 et *ivlOffset* est décrémenté avec la valeur de *ivlCurrRange*.
- Sinon (*ivlOffset* est inférieur à *ivlCurrRange*), la variable *binVal* est mise à 0.

Le flux binaire ne doit pas contenir de données dont la valeur de *ivlOffset* est supérieure ou égale à *ivlCurrRange* à la fin de ce processus.

3.3.2.1 Organigramme

La Figure 3.10 illustre le processus de décodage CABAC en mode bypass BM.

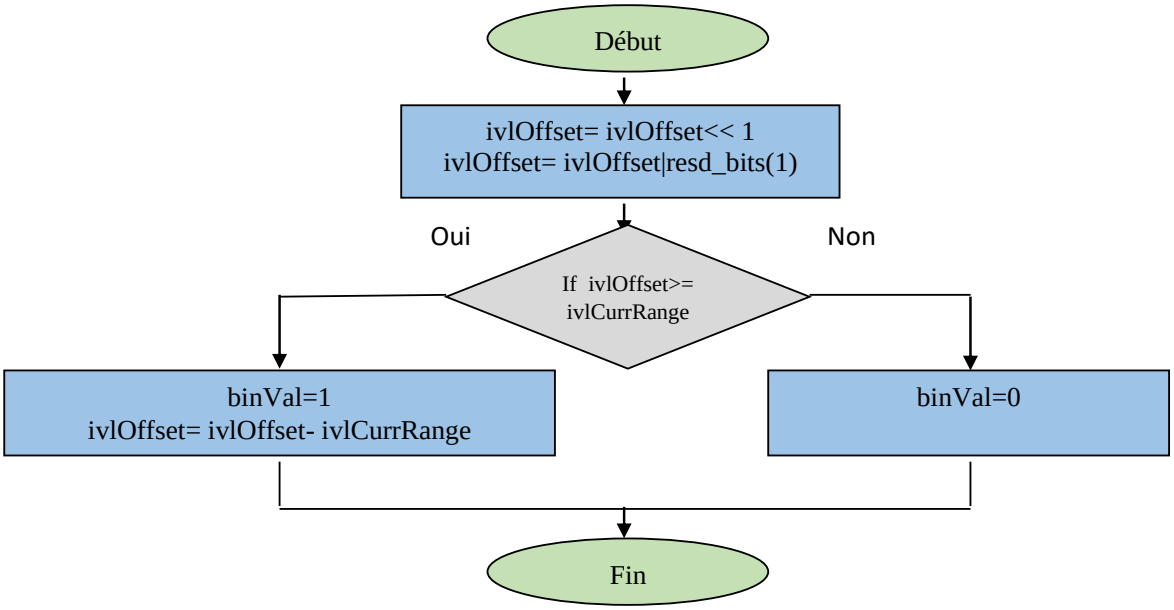


Figure 3.10 : Organigramme du Mode bypass.

3.3.2.2 Programme sous Matlab

Cette figure montre une parité des programmes réaliser sous Matlab de décodage CABAC/VVC.

```
181          %byPass%mode/////////////==========================================
182
183
184 -        if byPassFlag == 1
185
186 -            ivlOffset= bitshift(ivlOffset,1);
187
188              % script de read_bit(1)----------------------------------------
189
190
191 -            ivlOffset = bitor(ivlOffset,read_bits1(ivlOffset));
192
193              % la_boucle----------------------------------------------------
194
195 -            if ivlOffset>=ivlCurrRange
196
197 -                binVal=1;
198 -                ivlOffset=ivlOffset-ivlCurrRange;
```

Figure 3.11 : Partie du programme Matlab en mode bypass.

3.3.2.3 Simulation

Les valeurs de *binval* décodées pour chaque élément syntaxique et chaque bit de *ivlOffset* est données sur la figure 3.12.

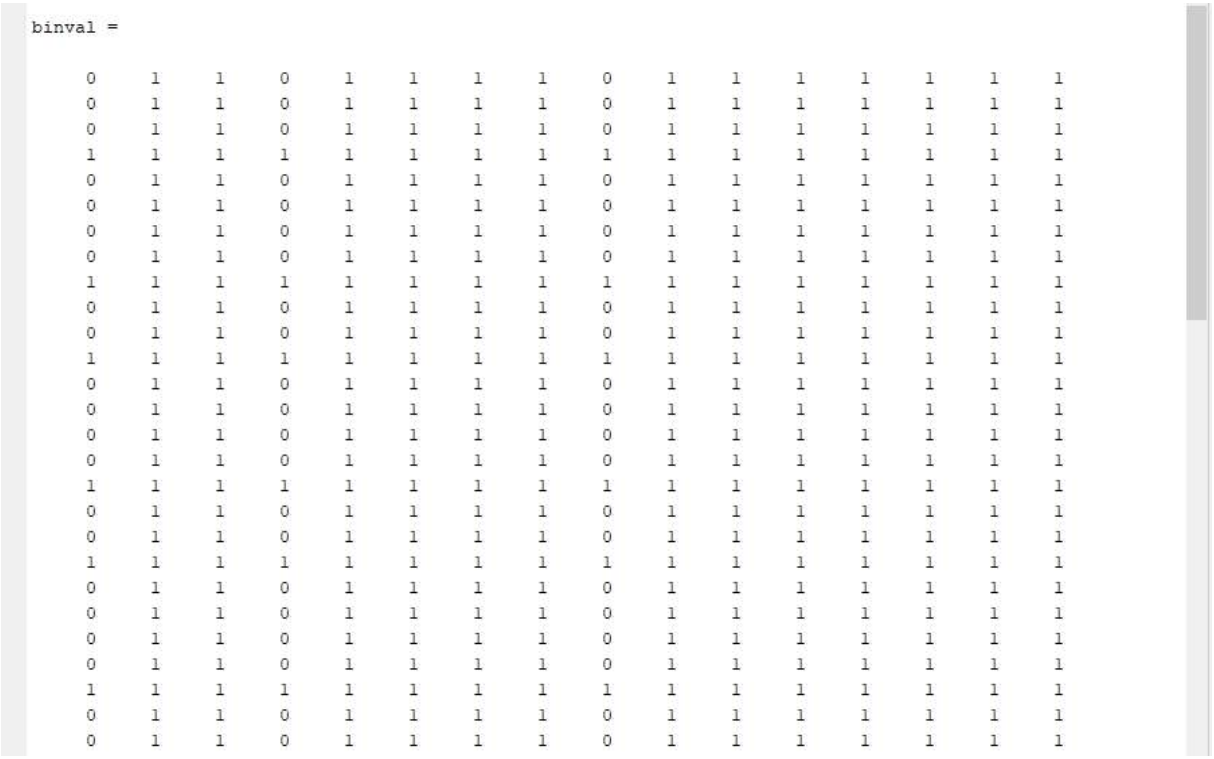


Figure 3.12 : Simulation sous Matlab de Mode bypass

3.3.3 Mode terminal

Les entrées de ce processus sont les bits de données de la tranche et les variables *ivlCurrRange* et *ivlOffset*. Les sorties de ce processus sont les variables mises à jour *ivlCurrRange* et *ivlOffset*, et la valeur décodée *binVal*.

Ce processus de décodage est appliqué sur les éléments syntaxiques de fin lice un bit, fin de carreau un bit, et fin de sous ensemble un bit correspondant à *ctxTable* égal à 0 et *ctxIdx* égal à 0. La figure 3.13 montre l'organigramme du processus de décodage correspondant, qui est appliqué comme suit :

D'abord, la valeur de *ivlCurrRange* est décrémentée de 2. et la valeur de *ivlOffset* est comparée à la valeur de *ivlCurrRange* avant de commencer le processus:

Si *ivlOffset* est supérieur ou égal à *ivlCurrRange*, la variable *binVal* est mise à 1, aucune renormalisation n'est effectuée et le décodage CABAC est terminé. Le dernier bit inséré dans le registre *ivlOffset* est égal à 1. Lors du décodage de fin de slice un bit, ce dernier bit inséré

dans le registre *ivlOffset* est interprété comme rbsp stop un bit. Lors du décodage, ce dernier de fin de tille un bit ou fin de sous-ensemble un bit inséré dans le registre *ivlOffset* est interprété comme bit d'alignement d'octet égal à un. Sinon le *ivlOffset* est inférieur à *ivlCurrRange*, la variable *binVal* est mise égale à 0, et la renormalisation est effectuée.

Cette procédure pourrait également être implémentée à l'aide de mode régulier (*ctxTable*, *ctxIdx*, *bypassFlag*) avec *ctxTable* = 0, *ctxIdx* = 0 et *bypassFlag* = 0. Dans le cas où la valeur décodée est égale à 1, 7 bits supplémentaires seraient lus par décision de décodage (*ctxTable*, *ctxIdx*, *bypassFlag*) et un processus de décodage devrait ajuster son pointeur de flux binaire en conséquence pour décoder correctement les éléments syntaxiques suivants [23].

3.3.3.1 Organigramme

La figure 3.13 montre le processus de décodage CABAC/VVC en mode terminal. Il est basée sur les valeurs des variables « ivlcurr*Range* » et « ivl*Offset* » pour décoder *binVal*. Par ailleurs, la renormalisation est appliquée à la fin dans le cas où le *binVal* décodé est égal à 0.

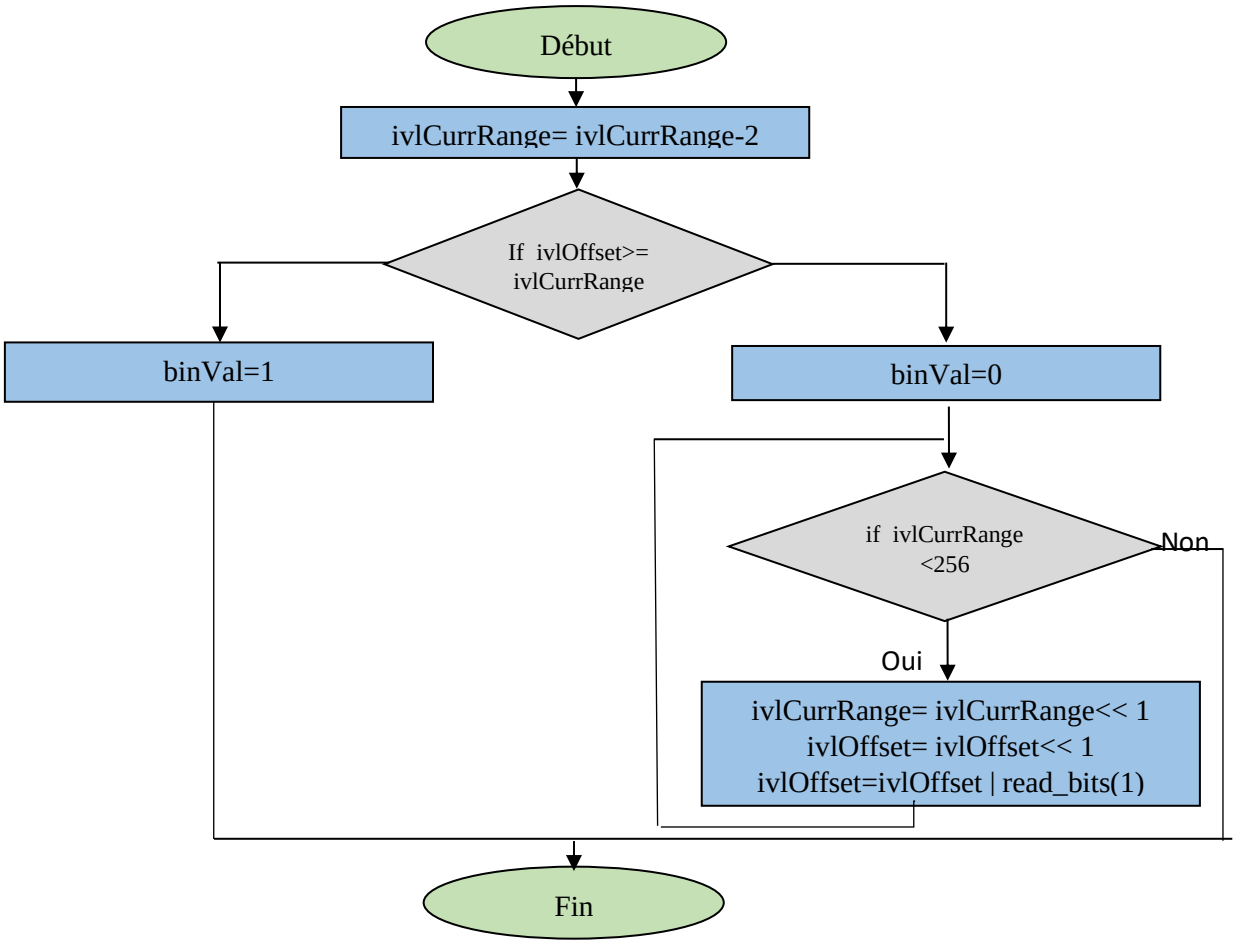


Figure 3.13 : Organigramme du Mode terminal avec la Renormalisation.

3.3.3.2 Programme sous Matlab

Cette figure montre une capture l'écran d'une partie de programme réaliser sons matlab de décodage CABAC/VVC pour le mode Terminal.

```
206          %terminate%mode/////////////=======================================
207
208 -        elseif ctxTable == 0 &  ctxIdx == 0
209
210
211              % DecodeTerminate
212
213 -            a=ivlCurrRange;
214 -            ivlCurrRange=a-2;
215
216              % la_boucle-----------------------------------------------------
217
218 -            if ivlOffset>=ivlCurrRange
219 -                binVal=1;
220 -            else
221 -                 binVal=0;
222
```

Figure 3.14 Partie du programme Matlab en mode terminal.

3.3.3.3 Simulation

La variable *binVal* obtenue pour chaque élément syntaxique et chaque pixel (*ivlOffset*) est donnée sur la figure suivante :

```
binval =

     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
```

Figure 3.15 : Simulation sous Matlab de mode terminal.

3.4 Processus de décodage arithmétique global

Les entrées de ce processus sont *ctxTable*, *ctxIdx* et *bypassFlag* les variables d'état le ivlCurrRange et le ivlOffset. La sortie de ce processus est la valeur du *binval* si le drabeau *bypassFlag* est égal à 1, ModeBypass( ) est invoqué Sinon, si *bypassFlag* est égal à 0, *ctxTable* est égal à 0 et *ctxIdx* est égal à 0, le ModeTerminate() est invoqué.

Sinon si le drabeau *bypassFlag* est égal à 0 et le *ctxTable* n'est pas égal à 0, le Mode régulier (*ctxTable*, *ctxIdx*) est invoqué.

3.4.1 Organigramme

La figure 3.16 illustre l'ensemble du processus de décodage arithmétique CABAC/VVC. Pour décoder la valeur d'un bin, la table d'index du contexte ctxTable, le *ctxIdx* et le drabeau *bypassFlag* sont passés au processus de décodage arithmétique.

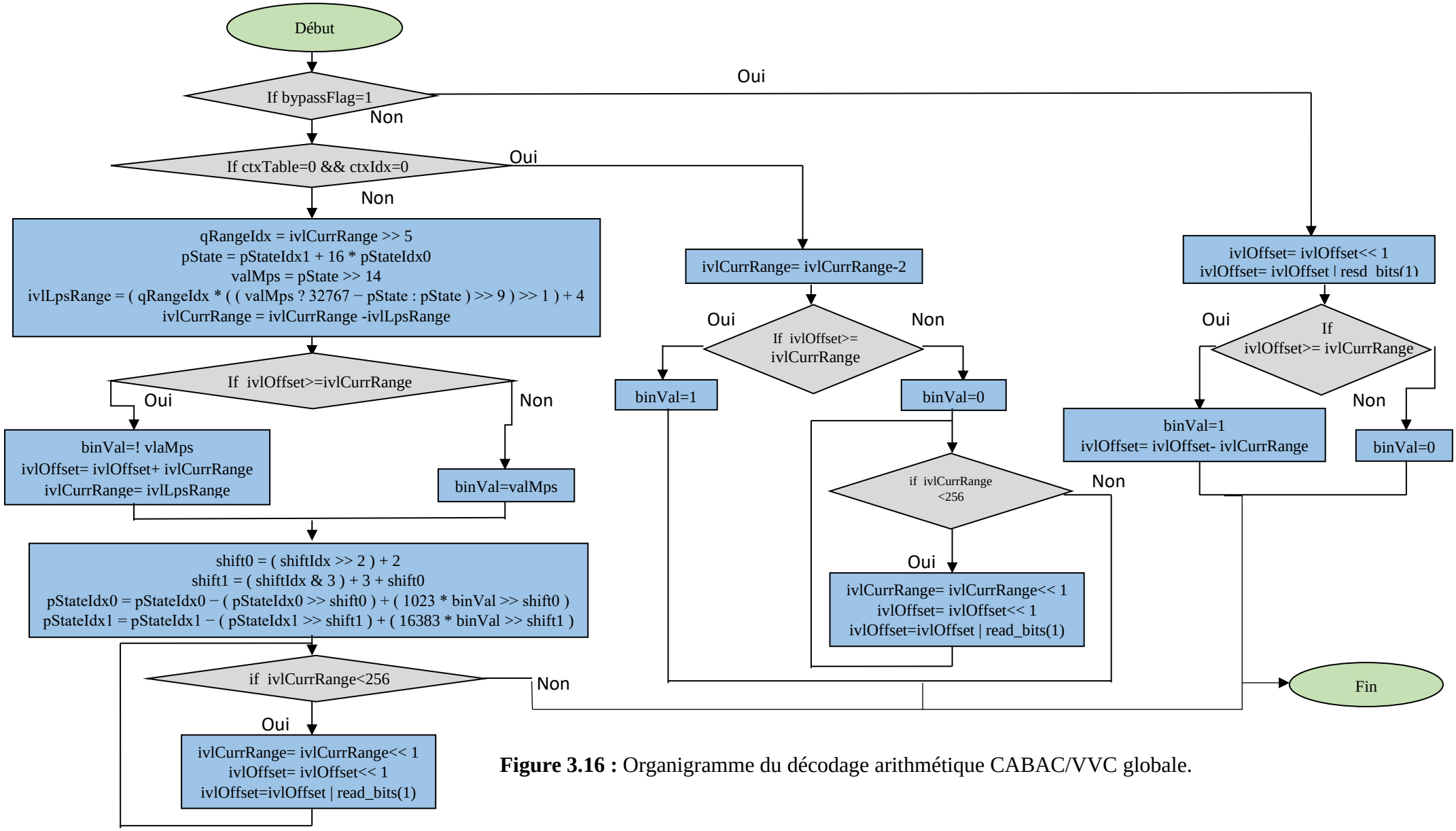


Figure 3.16 : Organigramme du décodage arithmétique CABAC/VVC globale.

3.4.2 Programme sous Matlab

Une partie du code Matlab réaliser pour l'algorithme de décodage CABAC/VVC est représentée sur la figure 3.17 qui commence par les pixels du bloc de luminance.

```
148 -   for x = 1:16
149 -           pix = luma (x);
150 -           ivlOffset = v (x);
151             % binvalvec = cell (1,74 );
152
153 -       for i=1:104
154
155 -            ctxIdx=A(i,2);
156 -            initValue=A(i,3);
157 -            shiftIdx=A(i,4);
158 -            byPassFlag = A(i,5);
159 -            ctxTable=A(i,6);
160
```

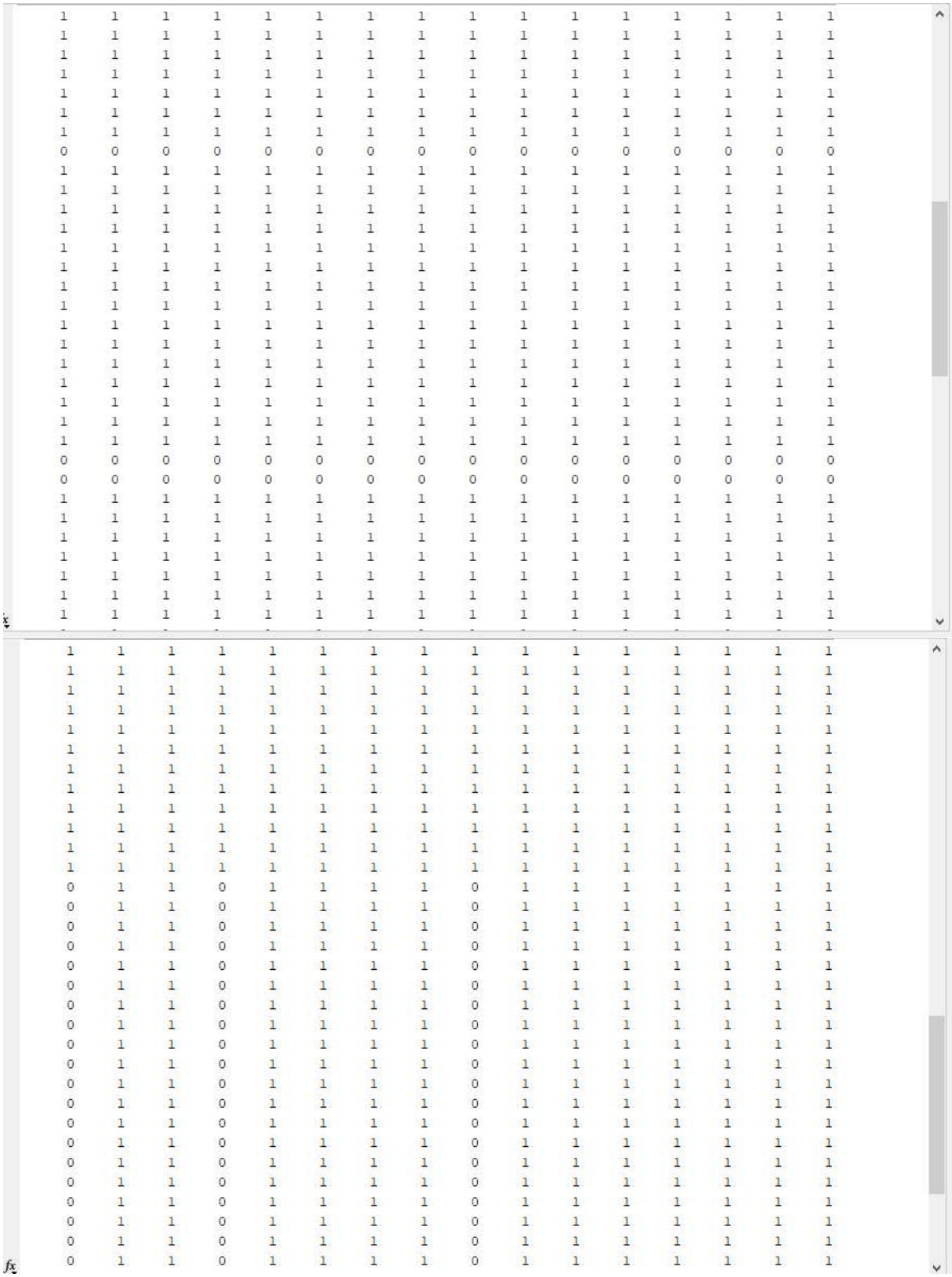
Figure 3.17 Partie du programme Matlab pour le décodage arithmétique binaire CABAC/VVC.

3.4.3 Simulation

La figure suivante montre les bins décodés par les trois modes de décodage CABAC.

```
binval =

     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     1     0     0     1     0     0     0     0     1     0     0     0     0     0     0     0
     0     1     1     0     1     1     1     1     0     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
```



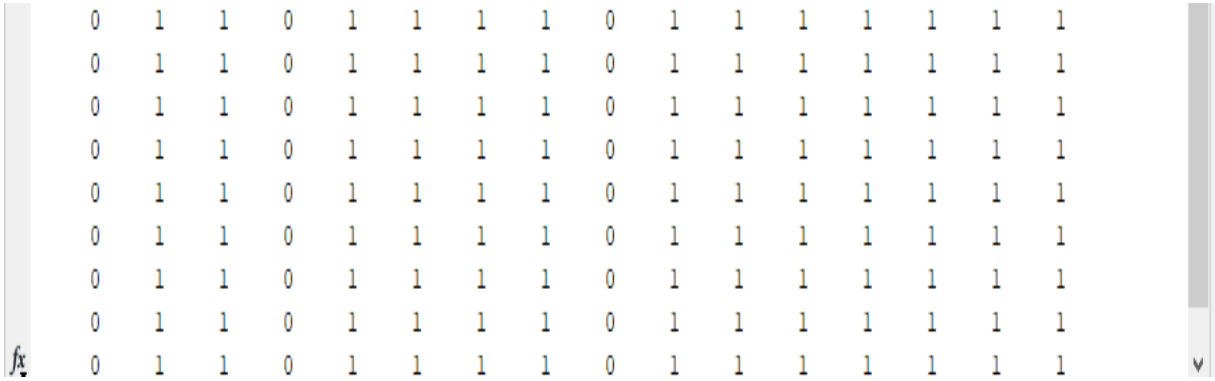


Figure 3.18 : Simulation sous Matlab de décodage BAD/CABAC global.

3.5 Conclusion

Dans ce chapitre, les différents outils matériels et logiciels utilisés ont été présentés. Ensuite nous avons aussi concevoir les organigrammes des décodages arithmétiques binaire avec leurs implémentations sous Matlab. Ce passage nous a permis de maitriser les algorithmes décodage arithmétique binaire utilisés dans la norme de compression vidéo VVC avant de passer à la description matérielle en langage VHDL sous ISE. Pour cela, le chapitre suivant sera consacré à la conception et l'implémentation matérielle de cet algorithme.

Chapitre 04

4.1 Introduction

Ce dernier chapitre est réservé pour l'implémentation de l'algorithme de décodage arithmétique binaire CABAC / VVC sur FPGA. Les architectures proposées seront données en détails pour chaque partie de cet algorithme avec la partie contrôle (FSM) suivi par une synthèse sous l'ISE, la simulation sous Model-Sim et l'estimation de l'énergie consommée sous l'analyseur XPower (ISE). Nous terminerons le chapitre par une comparaison et une discussion des résultats obtenus.

4.2 Architectures proposées pour la partie CABAC / VVC

La conception de notre architecture est basée sur des blocs fonctionnels où chaque bloc assure une tâche bien précise. Ces blocs fonctionnels sont interconnectés entre eux et sont contrôlés par une FSM (machine à états finis) qui séquence le bon fonctionnement du décodeur BAD.

4.2.1 Architecture Globale

A l'entrée du bloc on a le : *ctxTable*, *ctxIdx*, et *bypassFlag* et les variables *ivlcuurRange* et *ivlOffset* tandis qu'à la sortie nous trouvons la valeur de *binVal* ainsi que la mise à jour des variables *ivlcuurRange* et *ivlOffset*. La figure 4.1 illustre le bloc de décodage arithmétique.

CABAC qui est basé sur trois modes de décodage : RM, BM et TM. Dans les sous sections suivantes nous présenter l'architecture de chaque mode se parai ment.

4.2.2 Architecture proposée pour le Mode Régulier (RM)

Le bloc RM est utilisé pour les éléments syntaxiques traités par le modèle du contexte. Le *ctxIdx* est calculé pour chaque *bin* en se basant sur le type de l'élément syntaxique à décoder, l'indexe du *bin* (*binIdx*) et les propriétés du voisinage des unités de codage. Le bloc de décodage arithmétique représenté sur figure 4.2 est appliqué après avoir sélectionné le modèle du contexte (*pStateIdx , valMps , shiftidx*). En sortie, nous obtiendrons le *binVal* (*bin* décodé) et la mise à jour des variables *pStateIdx ; valMps ; ivlcuurRange* et *ivlOffset.*

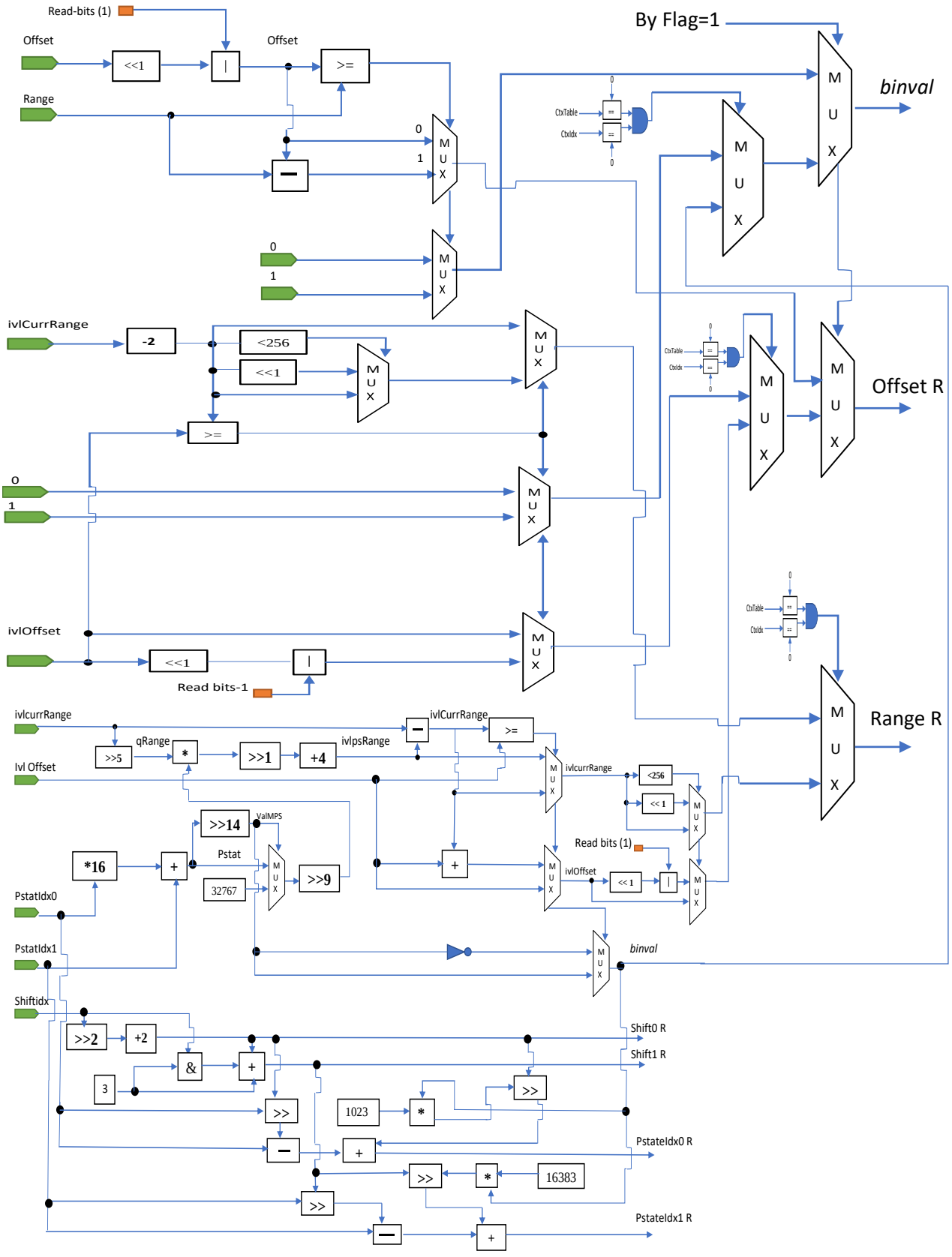


Figure 4.1: Architecture globale de module CABAC / VVC au niveau de décodage.

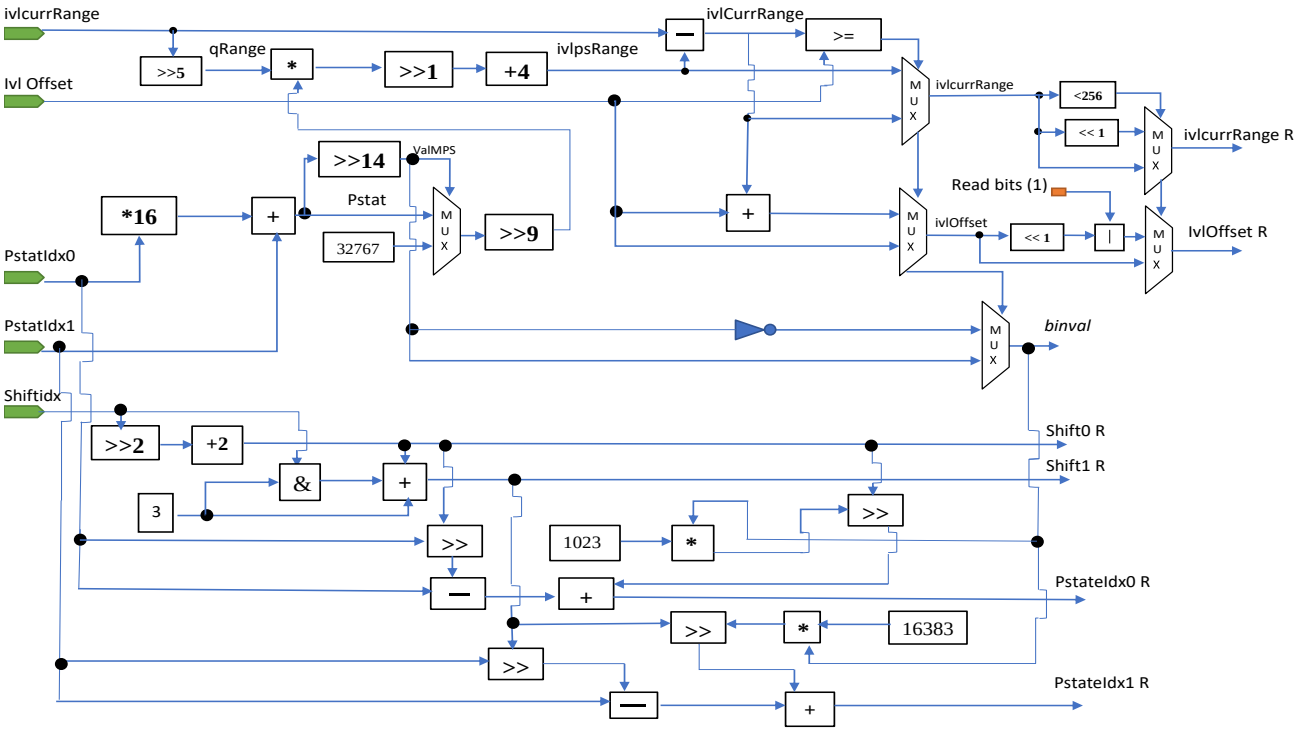


Figure 4.2: Architecture proposée du bloc de décodage arithmétique en mode régulier

4.2.3 Architecture proposée pour le Mode Bypass (BM)

Le bloc BM est la partie la moins complexe dans l'étage de décodage arithmétique binaire. Sachant que, le modèle du contexte est non utilisé dans ce mode. La figure 4.3 illustre le bloc de décodage bypass. À l'entrée de cette architecture on a deux variables *ivlOffset* et *ivlcuurRange*. À la sortie du bloc on trouve le *binVal* (la valeur de *bin* décodé) ainsi que la valeur *ivlOffset'* (la mise à jour de la variable Offset).

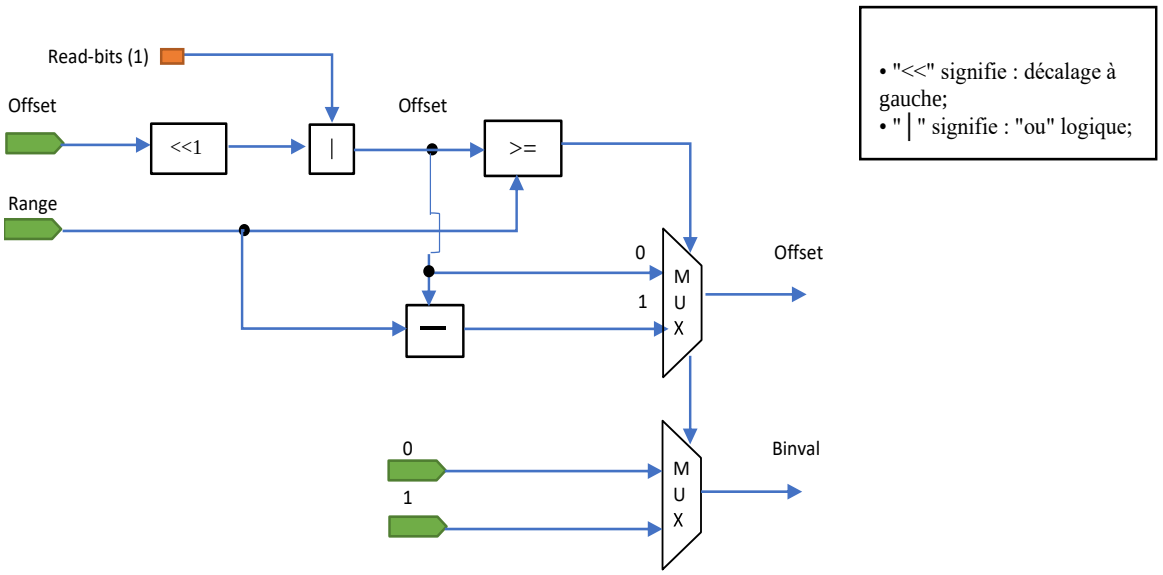


Figure 4.3: Architecture de décodage arithmétique CABAC en mode bypass.

4.2.4 Architecture proposée pour le Mode Terminal (TM)

Le bloc TM est utilisé seulement à la fin de chaque CTU dans les slices Figure 4.4 . A l'entrée du bloc on a les deux variables *ivlcuurRange* et *ivlOffset* tandis qu'à la sortie nous trouvons la valeur de *binVal* ainsi que la mise à jour des variables *ivlcuurRange'* et *ivlOffset.*

L'étape de la renormalisation (renormalisationD) est représentée sur le même bloc de décodage CABAC en mode terminal (Figure 4.4).

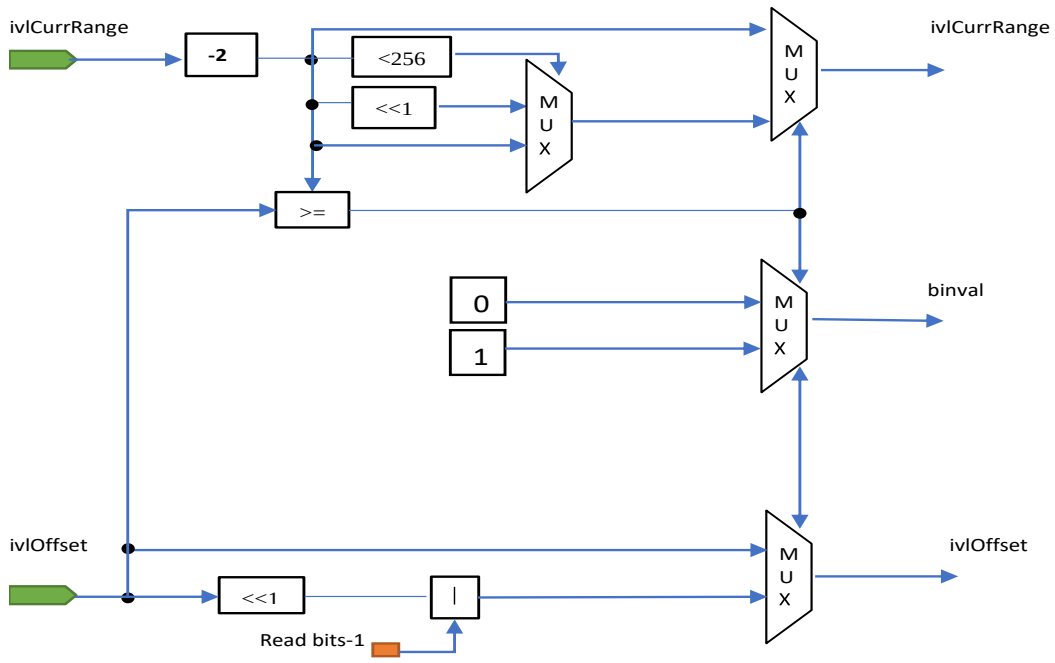
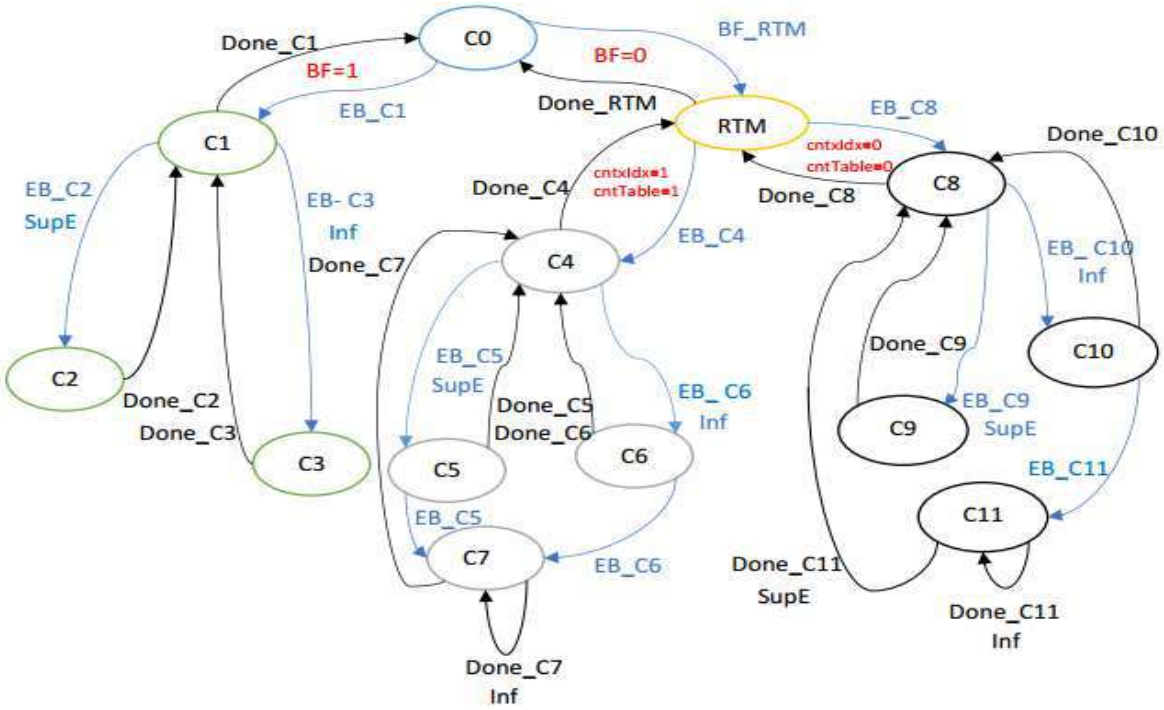


Figure 4.4: Architecture du bloc de décodage CABAC en mode terminal.

4.2.5 (finit state machine) FSM

Les machines à états finis (FSM) sont des circuits séquentiels utilisés dans de nombreux systèmes numériques pour contrôler le comportement, la gestion ainsi que le chemin de flux de données.

La figure suivante montre une FSM proposée basée sur un diagramme d'état pour gérer notre architecture proposée BAD du décodeur CABAC dans la norme de compression VVC. Il est à signaler que la transition d'un état a un autre est assurée par des signaux d'activations (Enable), signaux fin de tâche (Done) et les conditions de validation.



Légende:
13 état sont utilisées dons notre bloc de contrôle FSM.
C0 : l'Etat initiale pour commencer le processus de Contrôle ;
RTM : l'Etat RTM pour gérer le décodage arithmétique ;
C1, C2 et C3 : sont adaptés pour gérés le BM ; SupE : supérieure ou égale ;
C4, C5 ,C6 et C7 : sont adaptés pour gérés le RM ; Inf : inferieur ;
C8, C9, C10 et C11 : sont adaptés pour gérés le TM ;
EB RTM, EB C1, C2,….., EB-C11 :signal d'activation Bloc RTM,C1, C2,….,C11 ;

Figure 4.5 : Diagramme d'état FSM proposé pour le BAD.

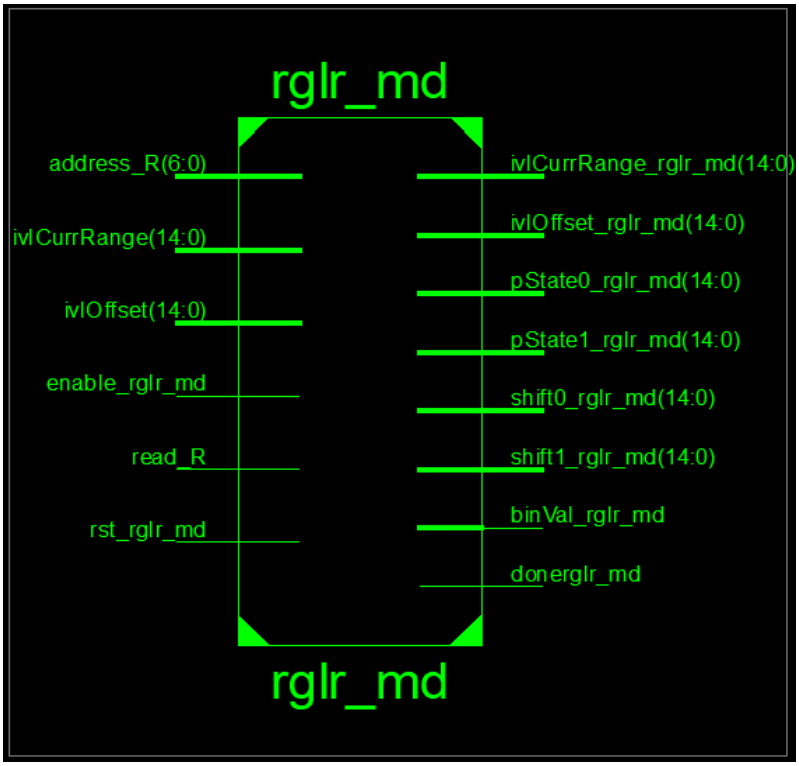
4.3 Partie Implémentation, synthèse et Emulation

4.3.1 Synthèse et schéma RTL des codes de décodage arithmétique

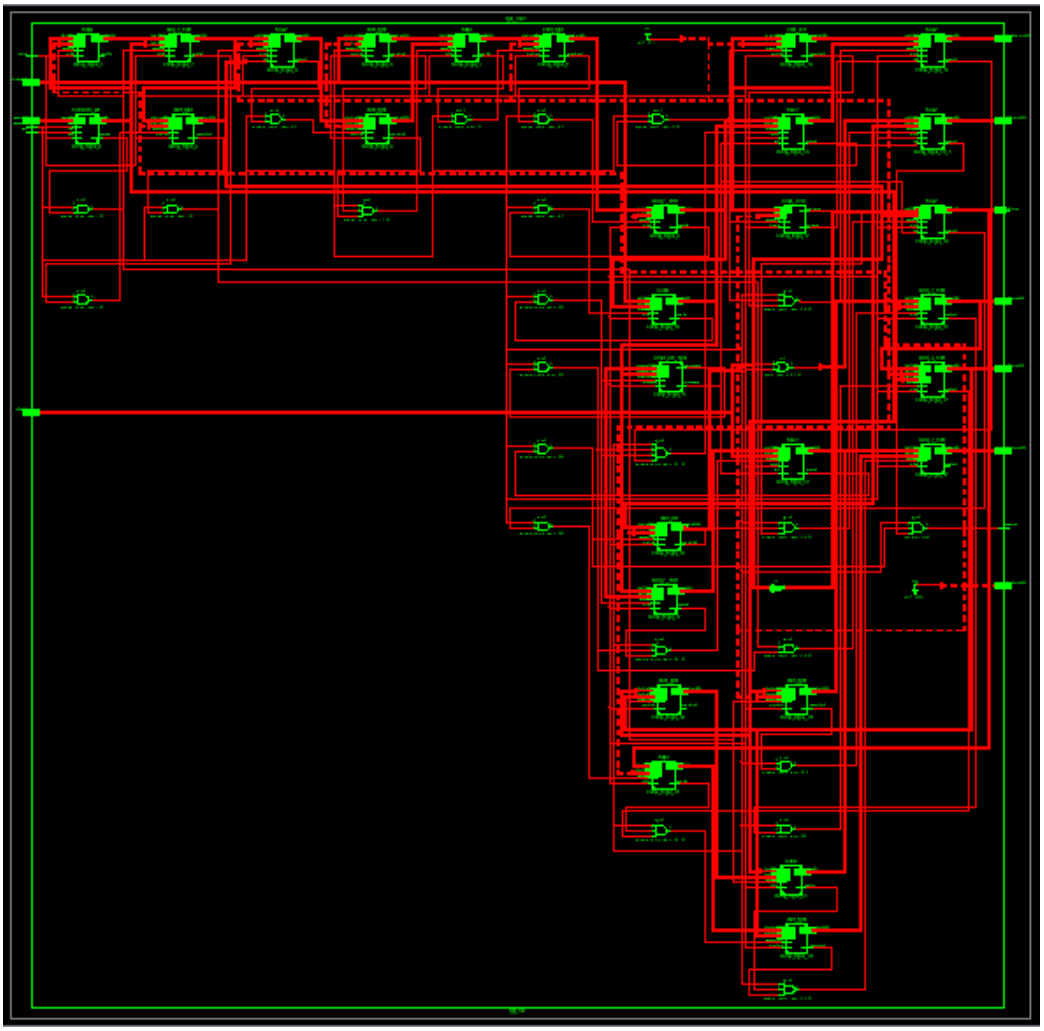
Dans notre travail nous utilisés le logiciel de description Matérielle (ISE ou VIVADO) en fin d'implumter de décodeur CABAC dans la norme VVC.

a) Synthèse et schéma RTL de la partie CABAC en mode Régulier :

La figure suivante montre le schéma Top Level du Régulier Mode



(a)



(b)

Figure 4.6 : Schéma du Module RM. (a) Top-Level, (b) Schéma RTL.

Le Tableau 4.1 montre les ressources consommées et la synthèse du bloc RM en termes de slices, fréquence et énergie consommée.

Tableau 4.1: Synthèse de la partie CABAC en mode RM.

Composants	Décodage CABAC RM
Slices	23
Slice Flip Flops	23
4input LUTs	62
Path delay (ns)	35.003

L'architecture du décodage arithmétique binaire CABAC VVC proposée est décrite en langage de description matérielle HDL, et synthétisée à l'aide de l'outil ISE de Xilinx version 14.7. De plus, son implémentation est effectuée sur le circuit FPGA (xc6vlx75t-2ff-484).

Notre architecture de décodage CABAC/VVC est Emmulé en utilisant le simulateur Model-Sim (Isim p.2013). La figure suivante illustre la description en VHDL de l'architecture du décodage arithmétique binaire CABAC/VVC sous ISE 14.7.

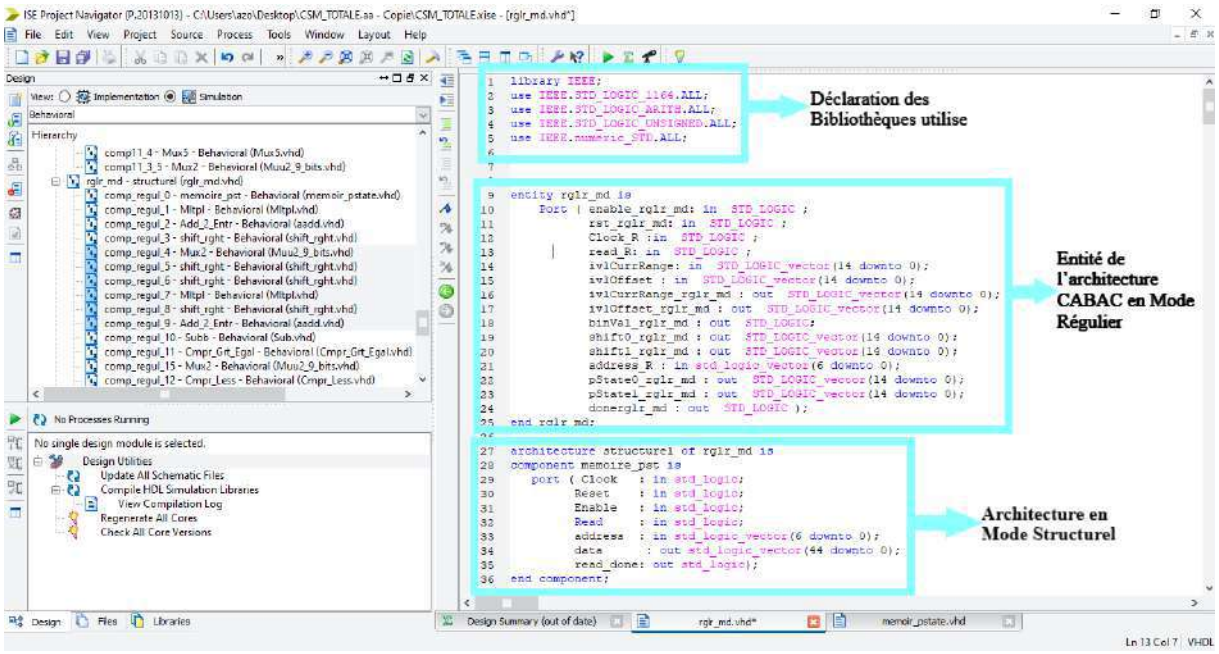
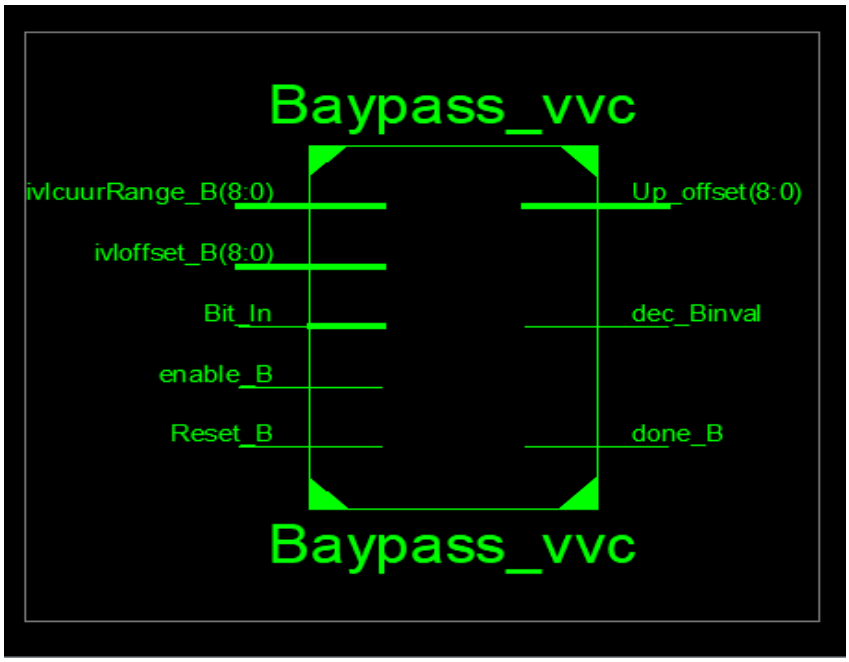


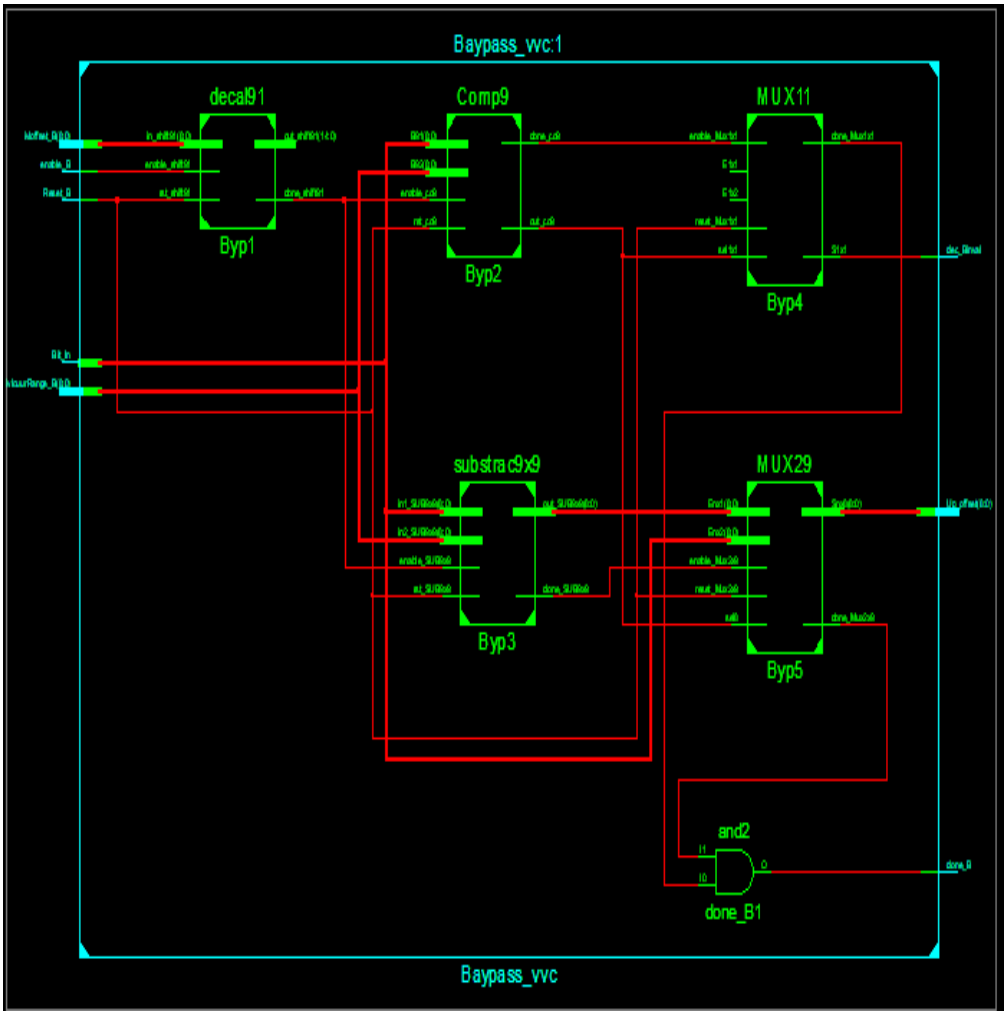
Figure 4.7: Description en VHDL de l'architecture du décodage arithmétique binaire CABAC/VVC sous ISE 14.7.

b) Synthèse et schéma RTL de la partie CABAC en mode Baypass :

La figure suivante montre le schéma Top Level du Bypass Mode



(a)



(b)

Figure 4.8 : Schéma du Module BM. (a) Top-Level, (b) Schéma RTL.

De plus, la synthèse de ce dernier avec la consommation est représenté sur le tableau 4.2.

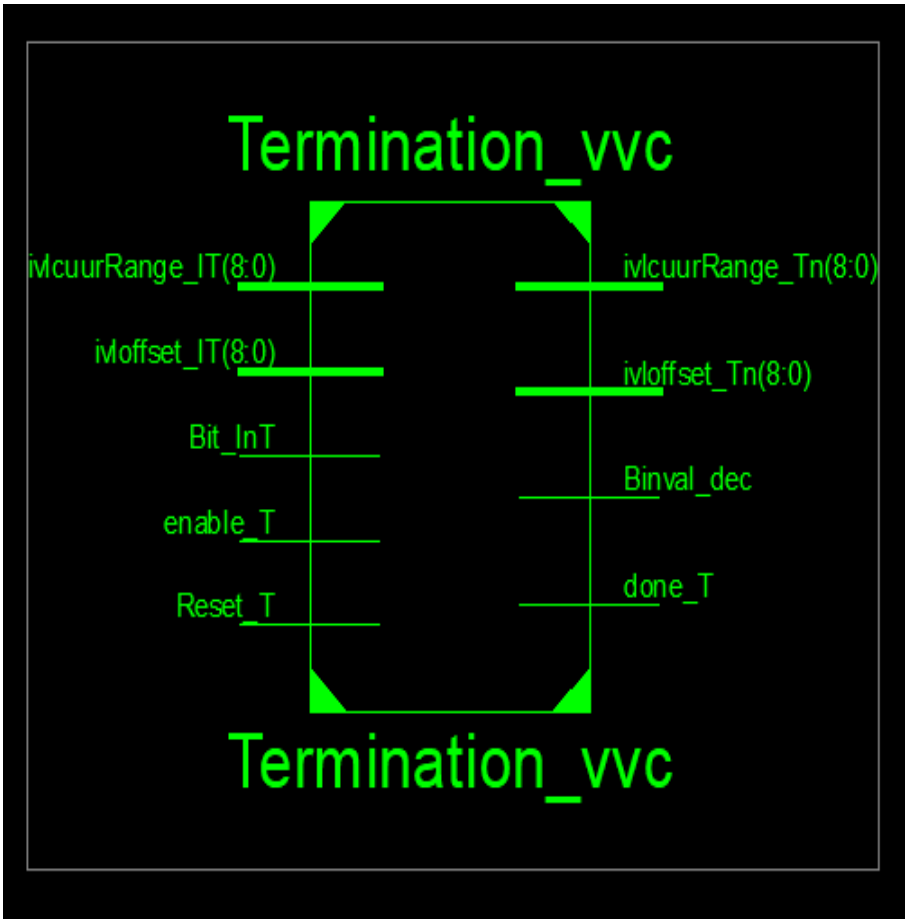
Tableau 4.2: Synthèse de la partie CABAC en mode BM.

Composants	Décodage CABAC BM
Slices	48
Slice Flip Flops	23
4input LUTs	23
Path delay (ns)	0.894
Consommation énergétique (mW)	1,293

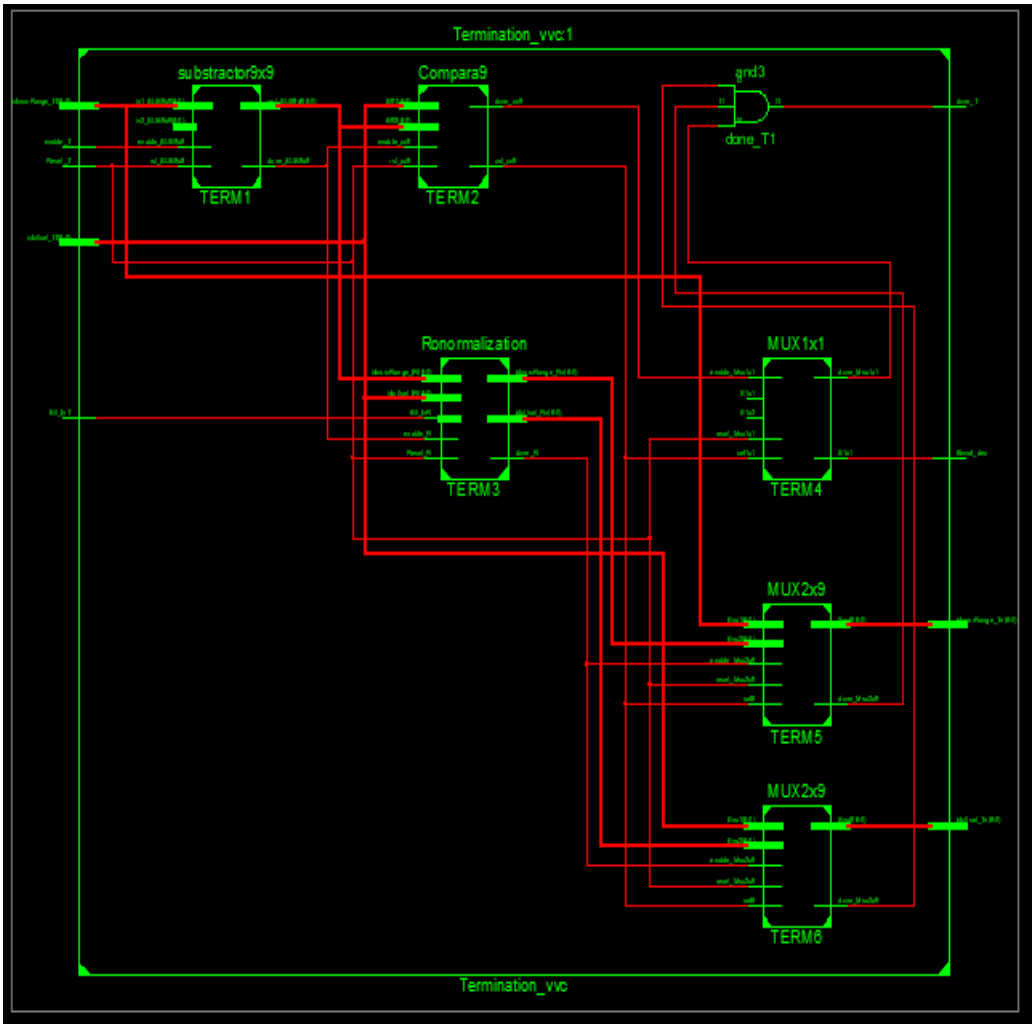
c) Synthèse et schéma RTL de la partie CABAC en mode TM

La figure suivante montre le schéma Top Level et le schéma RTL du module CABAC en mode Terminale.

Il est a signaler que le schéma top level représente un bloc qui eantient les Entrées et les sortes utilisés avec le nombre de bits pour entrées / sorties.



(a)



(b)

Figure 4.9 : Schéma obtenus du module TM (a) Top-Level. (b) Schéma RTL.

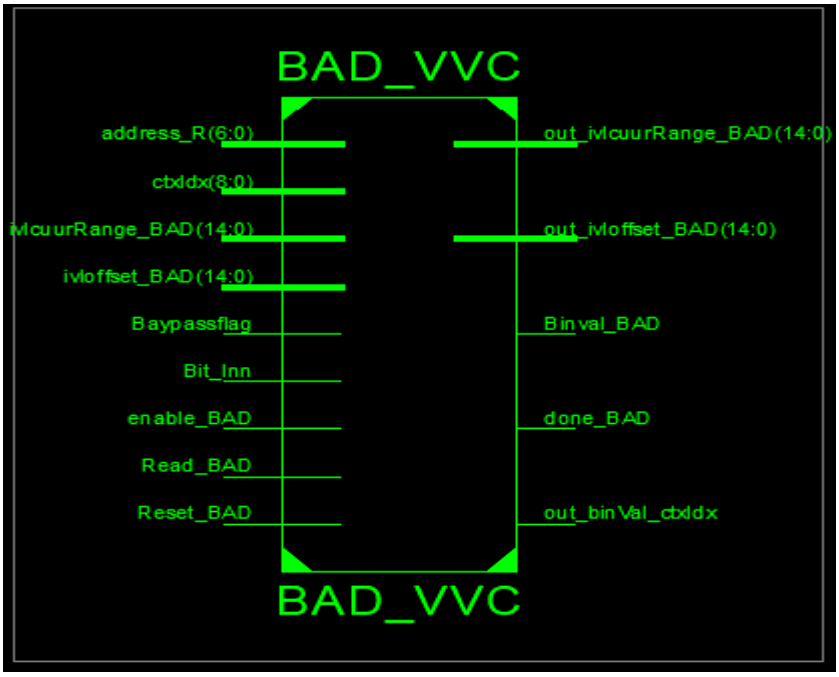
Le Tableau 4.3 montre les ressources consommées et la synthèse du bloc TM en termes de slices, fréquence et énergie consommée.

Tableau 4.3: Synthèse de la partie CABAC en mode TM.

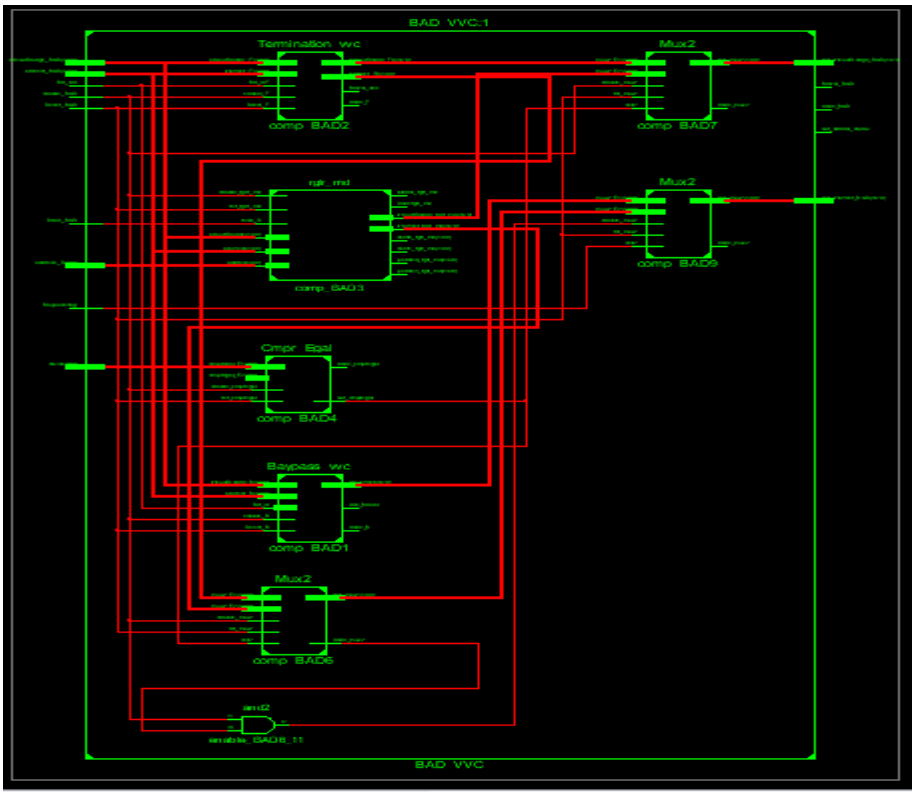
Composants	Décodage CABAC TM
Slices	68
Slice Flip Flops	50
4input LUTs	46
Path delay (ns)	0.965

d) Synthèse et schéma RTL de la partie CABAC en BAD :

La figure suivante montre le schéma Top Level et le schéma RTL du module CABAC de décodage arithmétique**.**



(a)



(b)

Figure 4.10 : Schéma obtenus du Module CABAC de décodage arithmétique (a) Top-Level. (b) Schéma RTL.

Le Tableau 4.4 montre les ressources consommées et la synthèse du bloc de décodage arithmétique en termes de slices, fréquence et énergie consommée.

Tableau 4.4: Synthèse de la partie CABAC par le bloc de décodage arithmétique.

Composants	Décodage CABAC TM
Slices	106
Slice Flip Flops	199
4input LUTs	271
Path delay (ns)	26.27
Fréquence (mhz)	643.91

4.3.2 Emulation des 3 modes de décodage arithmétique CABAC/VVC :

Les figures suivantes montrent la simulation des trois modes de décodage arithmétique CABAC/VVC sous Model-Sim Version 10.1c.

Les Bin décodés sont donnés avec la couleur blanc, rouge et violet sur les trois figures de Emulation obtenue.

Il est à rempaquer que même les signaux interne sont représentés sur le Résultats.

A) RM

La figure suivante montre le *binval* décodé (blanc) en fonction des variables d'entrées pour un élément de syntaxique **:**

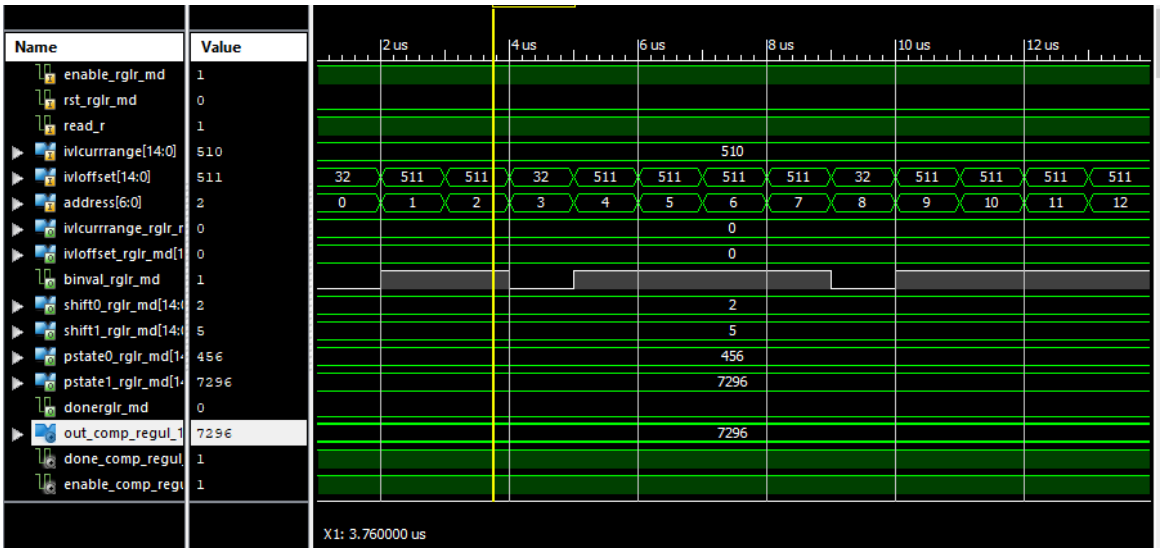


Figure 4.11 : Emulation sous Model Sim du Mode Régulier (les valeurs de binVal décode).

En fonction de la valeur décodée *binVal*, les variables *shifte0* et *shift1* et la mise à jour des deux variables *pStateIdx0* et *pStateIdx1* associées à *ctxTable* et *ctxIdx* :

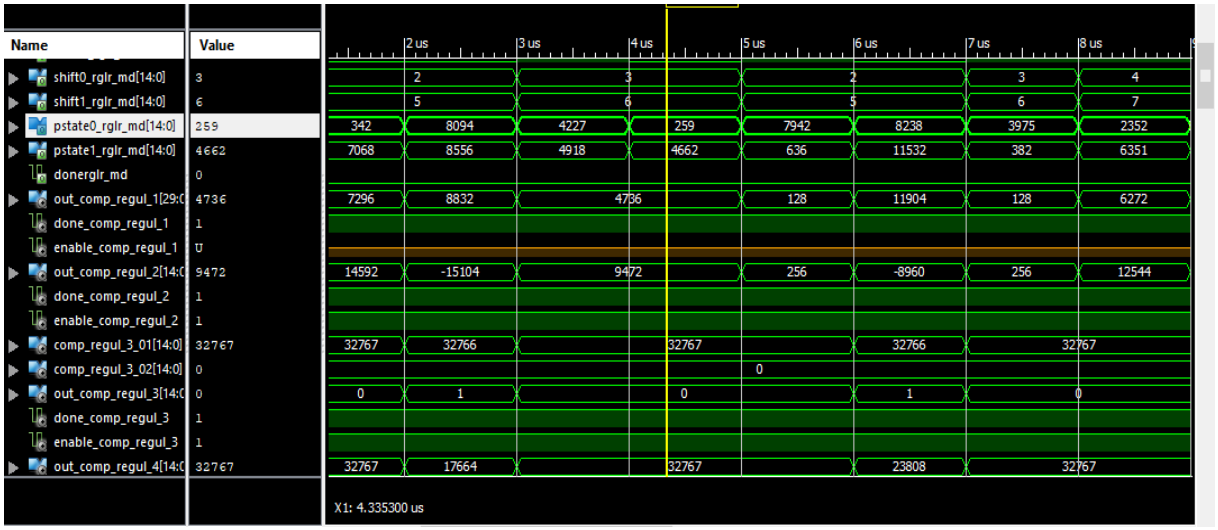


Figure 4.12 : Emulation sous Model Sim du Mode Régulier (les variables *shift0* et *shift1* la mise à jour des variables *pStateIdx0* et *pStateIdx1*).

il est a signaler que avec le Modelsim ona pas afficher toutes les valeurs obtenis de mise a jours des variable utilisée comme le Matlab. Pour ce la ona choisi de comparer quelques valeurs obtenues. parceque lorsque on fait aplique le zoom le resultat devient illisible.

B) BM

le *binval* décodé est représenté en Rouge sur le figure pour un élément syntaxique traité par le Mode Bypass.

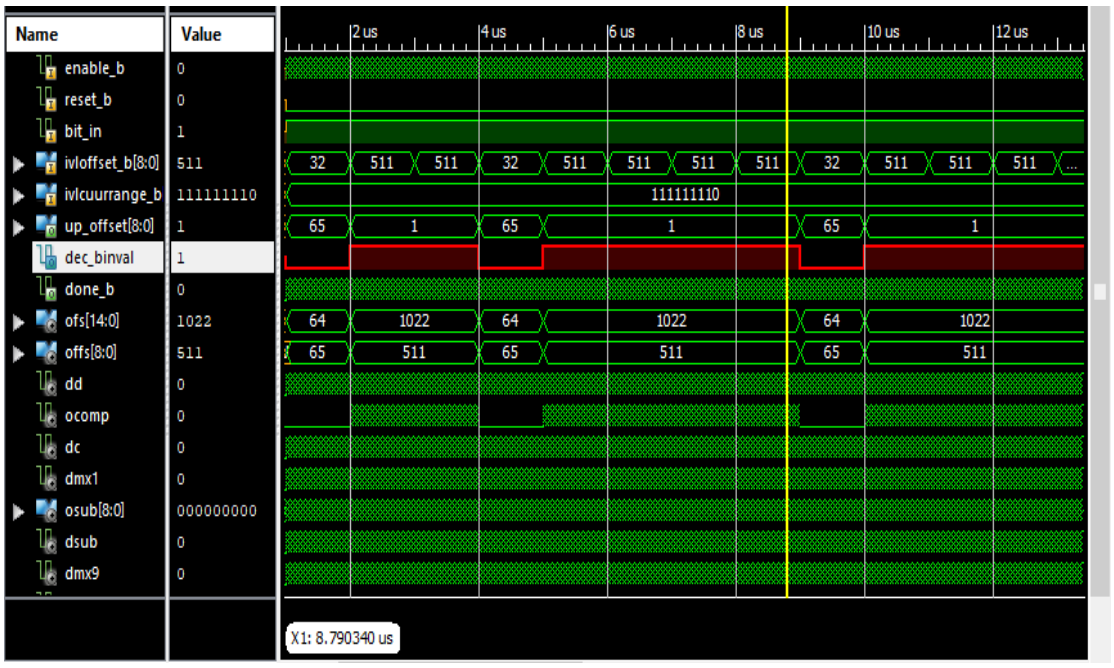


Figure 4.13 : Emulation sous Model-Sim du Mode Bypass.

C) TM

le binval décodé est représenté en violet sur le figure pour un élément syntaxique traité par le Mode Terminal.

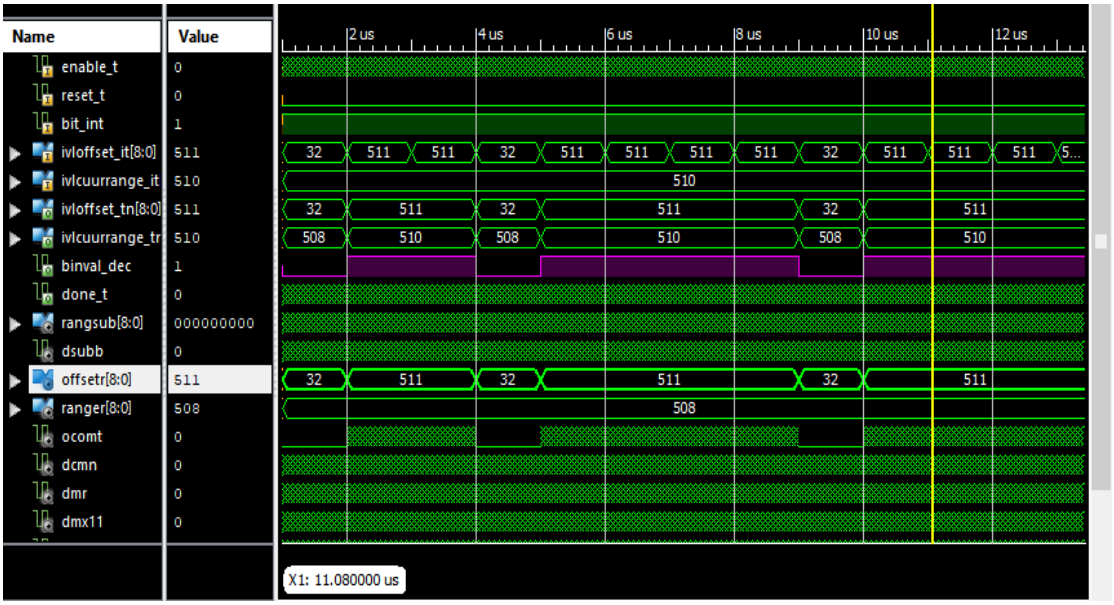


Figure 4.14 : Emulation sous Model-Sim du Mode Terminal.

Pour la renormalisation ou niveau de décodage l'émulation suivante est obtenue (figure 4.15) pour valider les architectures proposés une comparaison des résultats obtenus sous Matlab et sons Model-Sim est effectué.

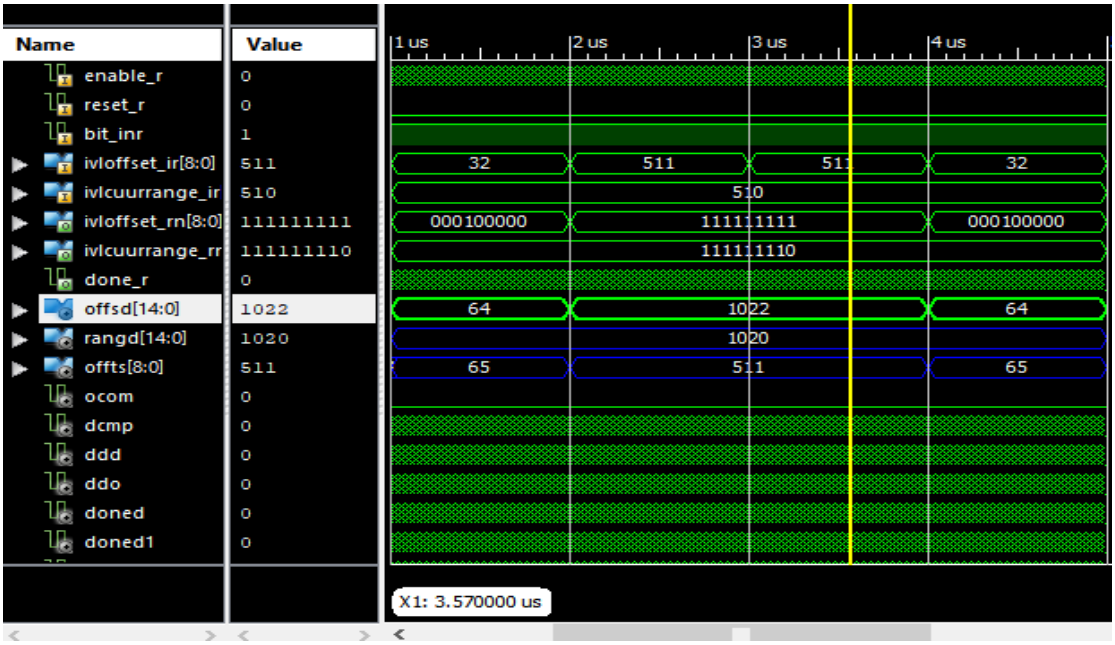


Figure 4.15 : Emulation du l'étape de la renormalisation (renormalisationD).

4.4 comparaison et validation des Résultats obtenus

A) RM

La figure (4.16) montre que les *binval* décodés obtenus sous Matlab sont les mêmes décodés sous Model-Sim.

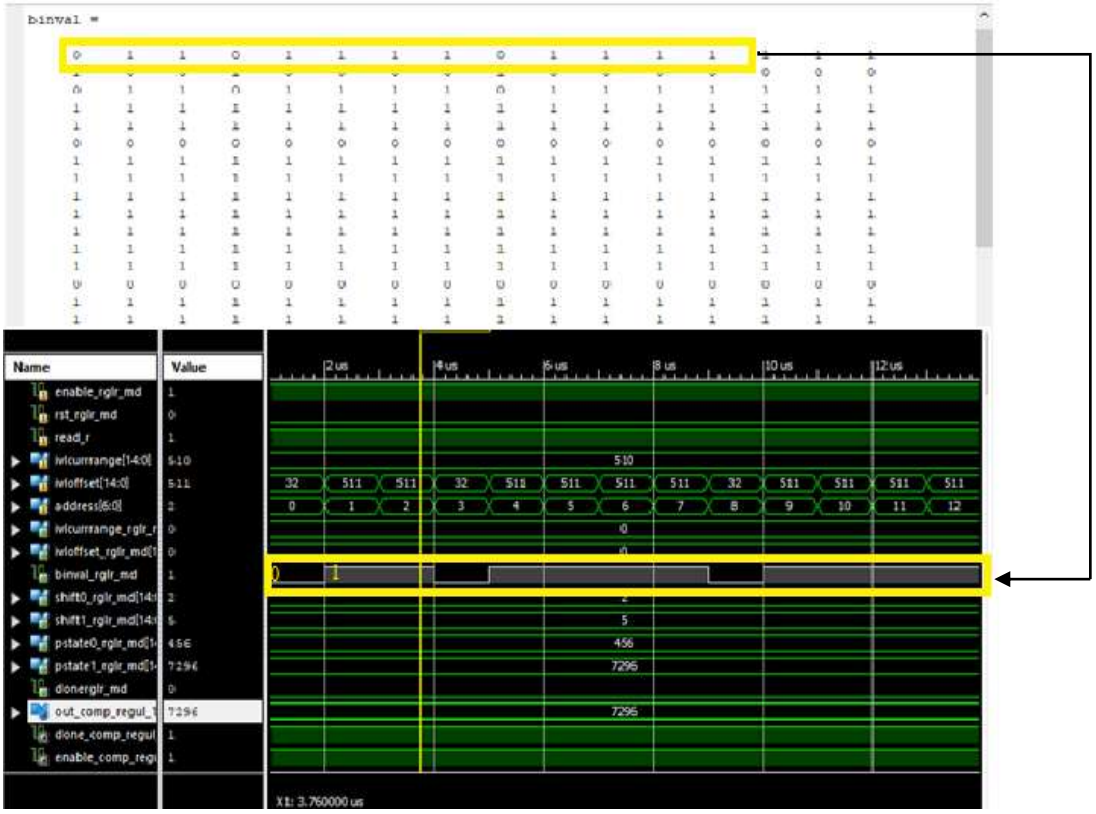


Figure 4.16 : comparaison des valeurs *binval* obtenues depuis sous Matlab et sous Model-Sim en Mode Régulier.

De plus, Après le décodage de chaque *binval* une mise à jours des variables *shift1* et *pstateIdx1* pour cela une validation des résultats obtenus par le Matlab et Model-Sim est donnée pour *shift1*et *pstateIdx1* sur la figure (4.17) et la (figure 4.18) respectivement.

```
Shift1 =

  Columns 1 through 18

     5     5     6     6     5     5     6     7     7     8     8     5     7     7     6     6     6     6

  Columns 19 through 36

     6     8     7     8     7     8     6     6     5     6     8     7     6     5     5     6     7     5

  Columns 37 through 54

    10     5     6     6     6     6     9     7     9    29     7     8     7     5     5     6     9     5

  Columns 55 through 72

     6     6     5     6     8     7     7     7     7     7     7     7     6     6     6     7     7     8

  Columns 73 through 74

     9     7
```

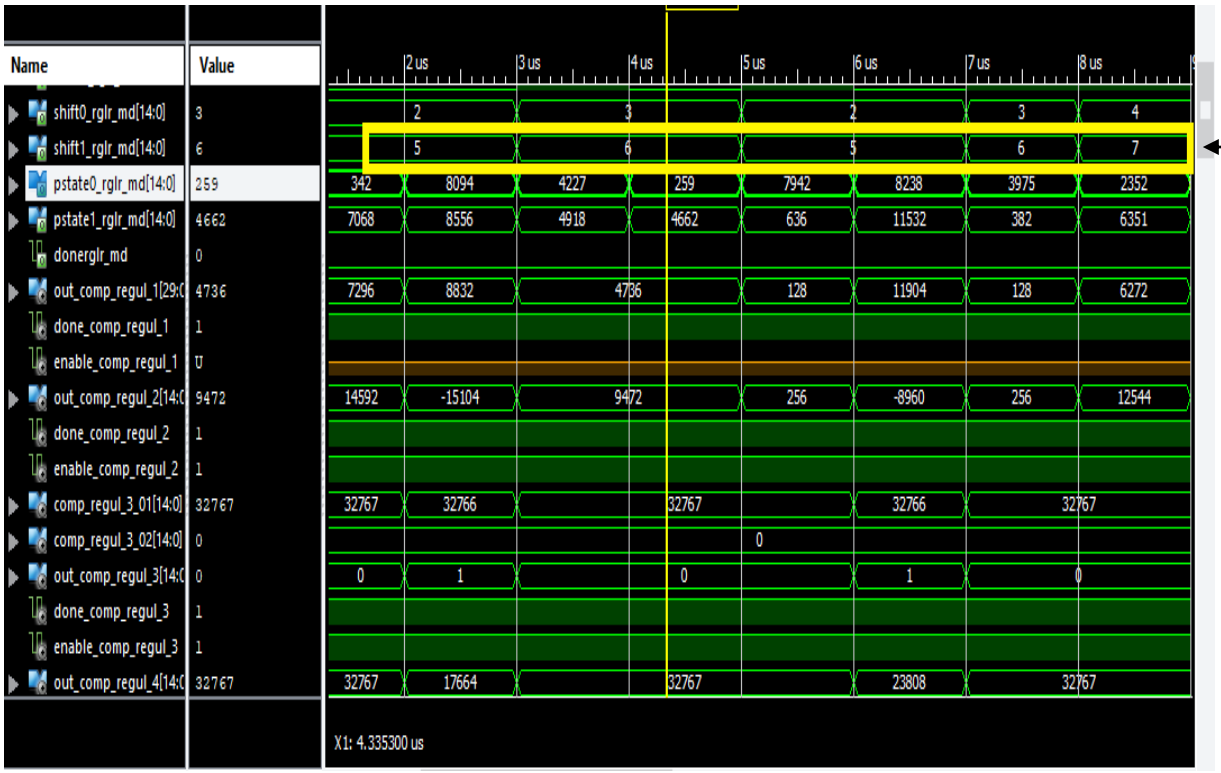


Figure 4.17 : comparaison des valeurs de *shift1* obtenus sous Matlab et sous Model-sim en Mode Régulier..

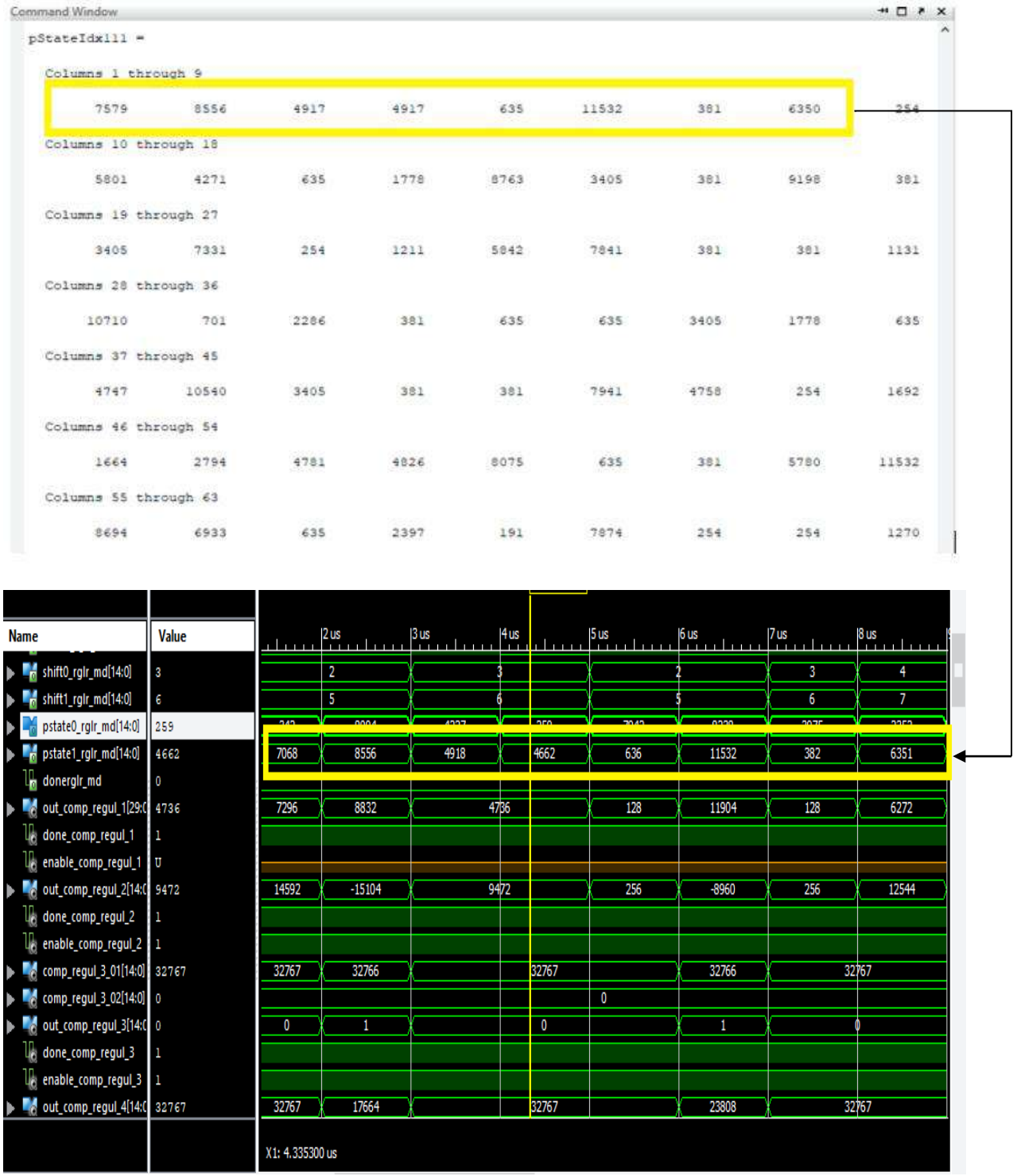


Figure 4.18 : Comparaison des valeurs de *pStatIndx1* obtenus sous Matlab et sous Model-Sim en Mode Régulier.

B) BM

Après avoir comparé les résultats obtenus dans Matlab et Modelsim pour un élément syntaxique, nous remarquons que nous avons obtenu les mêmes valeurs variables de *binval.*

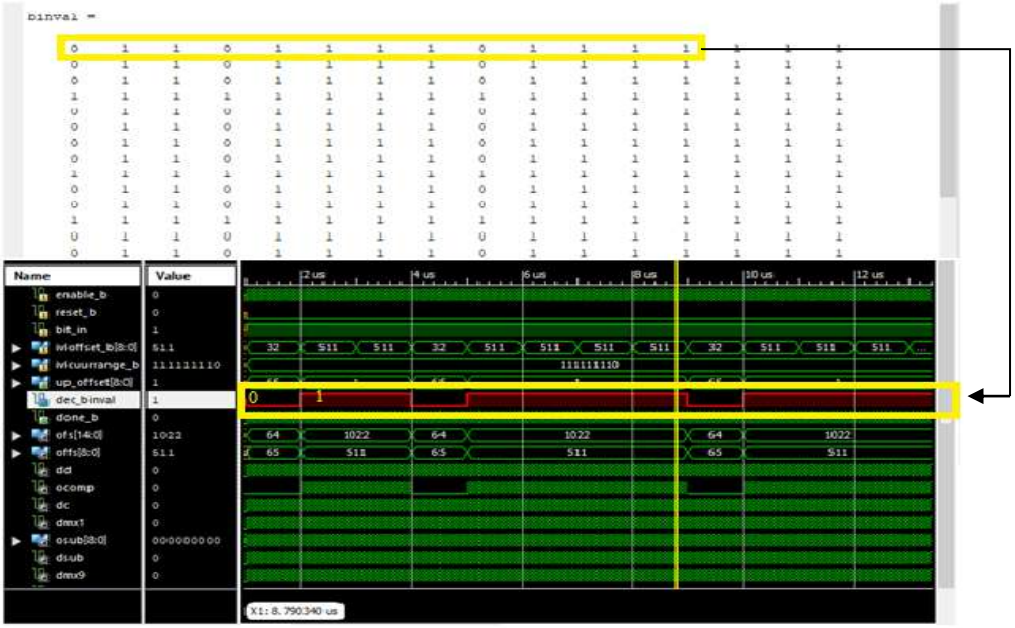


Figure 4.19 : comparaison des valeurs binval obtenues depuis Matlab et model-sim en Mode Baypass.

C) TM

Comparaison des résultats Mode Terminate.

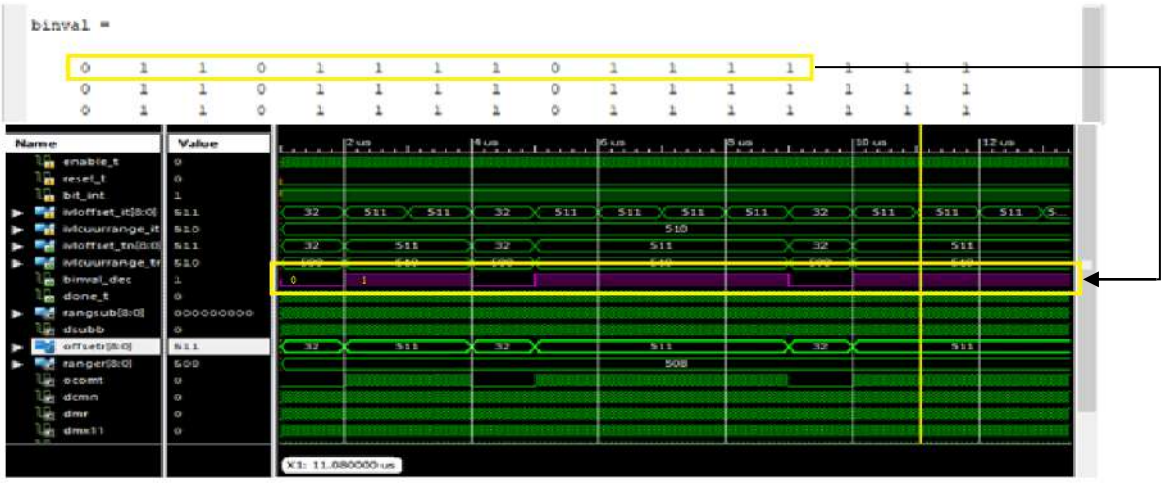


Figure 4.20 : Comparaison des *binval* obtenues por le Matlab et Model-Sim en Mode Terminal.

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'architecture globale proposée du décodeur arithmétique binaire CABAC/VVC, ainsi que les trois modes RM, BM et TM. Une architecture est proposée pour chaque mode avec une Emulation sous Model-Sim et une synthèse sous ISE.

De plus, un bloc de contrôle est préposé afin de gérer le processus de décalage basé sur une FSM.

A la fin du chapitre, une comparaison des résultats obtenus sous Model-Sim avec celles obtenus sous Matlab est effectuée afin de valider notre travail.

Conclusion générale

Conclusion générale

Avant de circuler sur Internet, télévision, téléphones portables et tablettes ; une vidéo doit passer essentiellement par une opération de compression.

Dans ce mémoire nous s'intéressons à l'étude, la conception et l'implémentation du décodeur entropique CABAC VVC. En effet, le codeur CABAC constitue le bloc d'entrée du décodeur VVC, et le bloc de sortie du codeur VVC. Nous savons que le décodeur entropique CABAC est basé sur trois blocs : CSM, décodage BAD et Dé-binarisation.

Notre travail est organisé principalement comme suite :

Premièrement, une explication des normes de compression vidéo récentes ont été présentées ainsi que les techniques de compression VVC ont été détaillées. Ensuite, les différents modules du système de communication et la théorie d'information ont été introduit. Suivi par une description de quelques notions de base sur le système de communication et la théorie de l'information. Par la suite, nous mettons en évidence le codage entropique et ses types plus précisément le décodeur entropique CABAC / VVC.

Deuxièmement, Nous avons concevoir les organigrammes des trois modes du décodage entropique CABAC avec leurs implémentations sous Matlab. Ce passage de l'implémentation logicielle sous Matlab nous a permis de maitriser l'algorithmes du décodage entropique CABAC utilisés dans la norme de compression vidéo VVC avant de passer à l'implémentation en langage VHDL sous l'ISE.

Troisièmement, nous avons implémenté le décodeur CABAC sur FPGA.

Notre architecture du décodeur entropique CABAC / VVC proposée est décrite en langage de description matérielle HDL. Synthétisée à l'aide de l'outil ISE de Xilinx version 14.7 en utilisant la carte de développement FPGA virtex4 (xc6vlx75-2ff-484). Et émulée sous Model-Sim.

Le décodeur BAD/VVC fonction avec une fréquence de 643.19 MHZ, un chemin critique de 26.27 ns et consommation en ressources acceptables.

Come perspectives, nous proposeurs d'implémenter un décodeur CABAC/VVC avec un chemin critique optimiser et un throughput (dé bit) amélioré en se basant sur les techniques de parallélisme et le pipeline. Afin d'assurer les contraintes temps réels.

De plus, a fin de valider les résultats obtenus sous Model-Sim nous avons les comparés avec celles obtenus peur Matlab.

Références

[1] YUCEL CETIN, ANIL CELEBI, « On the hardware implementation of binarization for high efficienty video coding », Electronics and Communication Engineering, Department KOCAELI Turkey, Proceedings of Academicsera International Conference, Istanbul, Turkey, 23rd-24th October 2017.

[2] Jean-Marc Thiesse, Codage vidéo flexible par association d'un décodeur intelligent et d'un encodeur basé optimisation débit-distorsion, thèse de Doctorat, Université Nice Sophia Antipolis, France, 19 Juillet 2012.

[3] "Video Formats, a Guide to Understanding Video Containers & Codecs" Content compiled from Matt Buchanan of Gizmodo.com at

[4] Brogui M. & Boumaraf N. "Detection et suivi d‟un objet suspect dans le contexte de vidéo surveillance", Université Kasdi Merbah- Ouargla, 2015.

[5] Thi-Lan LE, université de NICE-SOPHIA ANTIPOLIS, " Indexation et recherche de vidéo pour la vidéo surveillance" , 3 Février 2009.

[6] Touati R. , "Reconnaissance des actions humaines à partir d‟une séquence vidéo", Mémoire pour l‟obtention du grade de Maîtres sciences (M.Sc.) en informatique, Université de Montréal, Janvier 2014.

[7] Mark Nelson and Jean-loup Gailly, "The data compression book" second edition, League for Programming Freedom, Cambridge MA 02139, 1995

[8] MENASRI Wahiba, Implémentation d'un Décodeur CABAC HEVC / H.265 sur une Plateforme FPGA, thèse de doctorat, Université des sciences et technologie Houari Boumediene, Alger, Algérie, 2019

[9] Amira Yahi, Implantation de la chaîne inter-prédiction utilisée dans l'encodeur H.264/AVC sur une plateforme multi-composant, thèse de Doctorat, université badji mokhtar, ANNABA, Algérie, 2016.

[10] "Guide de la vidéo surveillance 2010", On www.boschsecurity.fr

[11] https://androidpctv.com/vvc-h-266-the-codec-that-enters-to-fight-for-the-8k dominance/

[12] ISO/IEC 11172-2, "Standard mpeg-1 : coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbits/s", tech. rep., 1991.

[13] ISO/IEC 13818-2, "Standard mpeg-2: information technology-generic coding of moving pictures and associated audio information", tech. rep., 1994.

[14] ISO/IEC 14496-2, "Standard mpeg-4: information technology-generic coding of audio-visual objects", tech. rep., 1998.

[15] Jerry D. Gibson, Toby Berger, Tom Lookabaugh, Dave Lindbergh and Richard L. Baker. “Digital Compression for Multimedia: Principles and Standards”, Morgan Kaufmann; 1 edition January 15, 1998.

[16] "H.262: Technologies de l' information - Codage générique des images animées et des informations sonores associées: vidéo" . Site Web de l'UIT-T . Union internationale des télécommunications - Secteur de la normalisation des télécommunications (UIT-T). Février 2000 . Récupéré le 13 août 2009 .

[17] “Video Coding for Low Bit rate Communication”, tech. rep., 1996

[18] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan.
Rateconstrained coder control and comparison of video coding standards. IEEE Trans. Circ. Syst. Vid., 13(7):688–703, July 2003.

[19] F. Bossen, B. Bross, K. Suhring, and D. Flynn. HEVC complexity and implementation analysis. IEEE Trans. Circ. Syst. Vid., 22(12):1685–1696, 2012.

[20] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz. Performance
and computational complexity assessment of high-efficiency video encoders. IEEE
Trans. Circ. Syst. Vid., 22(12):1899–1909, December 2012.

[21] Julien Lausson, « H.266/VVC : le codec qui définira l’avenir de la vidéo est prêt » [archive], sur https://www.numerama.com/tech/635488-h-266-vvc-le-codec-ecolo-qui-definira-lavenir-de-la-video-est-pret.html

[22] https://www.frandroid.com/android/732339_format-video-h-266-vvc

[23]. B. Bross et al., "Overview of the Versatile Video Coding (VVC) Standard and its Applications," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3736-3764, Oct. 2021, doi: 10.1109/TCSVT.2021.3101953

[24]. Thomas Amestoy. Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle. Traitement des images [eess.IV]. INSA de Rennes, 2021. Français. NNT : 2021 ISAR0009. tel-03566101

[25] www.codetd.com/en/article/11847286

[26] J. Pfaff et al., "Intra Prediction and Mode Coding in VVC," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3834-3847, Oct. 2021, doi: 10.1109/TCSVT.2021.3072430

[27] D. Karwowski, "Precise Probability Estimation of Symbols in VVC CABAC Entropy Encoder," in IEEE Access, vol. 9, pp. 65361-65368, 2021, doi: 10.1109/ACCESS.2021.3075875

[28] Institut Gaspard Monge, Université de Marne-la-Vallée, 77454 Marne-la-Vallée Cedex2, France. beal@univ-mlv.fr.†INRIA Rocquencourt, B.P. 105 78153 Le Chesnay Cedex, France.nicolas.sendrier@inria.fr.

[29] Lars Erik, Songe Paulsen, Design and analyses of an H.265 entropy encoder, thèse de Master, Norvégien Université de Science and Technologie, Norvège, Août 2017.

[30] Shannon, C.E. (1948), "A Mathematical Theory of Communication", Bell SystemTechnical Journal, 27, pp. 379–423 & 623–656, July &October, 1948.

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

[32] Jean-Marc Thiesse, Codage vidéo flexible par association d'un décodeur intelligent et d'un encodeur basé optimisation débit-distorsion, thèse de Doctorat, Université Nice Sophia Antipolis, France, 19 Juillet 2012.

[33] T.V. Ramabadran, K. Chen., Efficient compression of medical images througharithmeticcoding, ProceedingsSPIE:Medical Imaging, vol.1234, pp.761-775, 1990.

[34] ITU-T, TelecommunicationStandardizationSector of ITU, SeriesH:Audiovisual And MultimediaSystems, Infrastructure of audiovisual services – Coding of movingvideo, «Versatile videocoding », 08/2020.