REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère De L'enseignement Supérieur Et De La Recherche Scientifique

Université Dr Yahia Fares De Médéa

Faculté De La Technologie

Département De Génie Electrique



Projet de Fin d'Etudes

Présenté pour l'obtention du Diplôme de Master en Télécommunications

Spécialité : Systèmes des télécommunications

THEME

Implémentation sur FPGA d'un binarisateur / dé-binarisateur dans la norme de compression vidéo HEVC / H.265

Proposé et encadré par :

Dr. MENASRI Wahiba

Co-encadré par :

Dr. BOUHEDDA Mounir

Préparé par :

CHENNOUFI Sarah

DJABRI Manel

Année Universitaire 2020/2021

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 :

Avec un très grand amour et beaucoup de respect, je dédie ce modeste travail :

A l'homme, mon précieux offre du dieu, qui doit ma vie, ma réussite et tout mon respect : ***mon cher père Amor***

A la femme qui a souffert sans me laisser souffrir, qui n'a jamais dit non âmes exigences et qui n'a épargné aucun effort pour me rendre heureuse : ***mon adorable mère Saadia****.*

A mes chères sœurs Soumia et Amina *pour leurs encouragements permanents, et leur soutien moral,*

A mon cher frère, Ayoub *pour leur appui et leur encouragement,*

A mon adorable petite sœur Hadjer *qui sait toujours comment procurer la joie et le bonheur pour toute la famille.*

A mon très cher jumeau, à mon fiancé , et à mon compagnon de vie NADIR *, Tu as toujours été à mes côtés pour me soutenir et m'encourager, tu as ma source de joie et de bonheur, Que ce travail traduit ma gratitude et mon affection.*

A ma copine MANEL, *en témoignage de l'amitié sincère qui nous a liées et des bons moments ensemble, je te souhaite une vie pleine de succès.*

A ma grand-mère, mes oncles et mes tantes*. Que Dieu leur donne une longue et joyeuse vie.*

À mes chers professeurs *en témoignage de fraternité, avec mes souhaits de bonheur de santé et de succès.*

A mes meilleurs amis *avec lesquelles j'ai partagé de beaux moments :****sabrine, soulef, Mounia ,issraa, Farida, sarah*** *puisse dieu vous donne santé, bonheur, courage et surtout réussite.*

A *mes collègues étudiants de ma promotion 2020.*

CHENNOUFI Sarah.

Dédicace :

Je dédier ce travail à mes chers parents mon père Abdelmadjid et ma mère BABA Kheira, que mille dédicaces ne puissent mes sincères sentiments, pour leur patience illimitée, leurs encouragements contenus, leur aide on témoignage de mon profond amour et respect pour leur grands sacrifices.

À mes chers frères et sœurs : Laid, Orouba, Meriem, Chaïma pour leur grand amour et leur soutien qu'ils trouvent ici l'expression de ma haute gratitude.

À mes chers professeurs en témoignage de fraternité, avec mes souhaits de bonheur de santé et de succès.

À Mes très chères amies : Amira, Aya, Aziza, Doaa, Fatima, Hadjer, Hanane, Meriem, Nadjet, Nesrine, Rachida et tous mes collègues qui sans leur encouragement ce travail n'aura jamais vu le jour.

À Mes très chers cousins et cousines : Abdel Wahab, Youcef, Fatima, Hadjer, Houda, Somia.

À ma moitié et complément Youness.

Et à toute ma famille et tous ceux que j'aime.

DJABRI Manel.

Résumé

Le but principal de ce projet est l'implémentation sur plateforme FPGA d'un binarisateur/ dé-binarisateur dans la norme de compression vidéo HEVC/H.265 (*High Efficiency Video Coding*). La technique de binarisation dans la norme HEVC joue un rôle essentiel dans le cas de codage ou dans le cas de décodage. Il est à signaler que la binarisation est un des blocs qui constituent le codeur entropique CABAC. Dans le cas de codage la binarisation est placée à l'entrée du bloc CABAC afin de convertir les éléments syntaxiques (*ES*) non binaire car le CABAC utilise que les éléments binaires par contre le dé-binarisateur est placé à la sortie du bloc CABAC au niveau de décodage pour reconstruire les ES. Les résultats de comparaison de l'implémentation de notre binarisateur/ dé-binarisateur CABAC sous Matlab et ISE montrent son bon fonctionnement et la similarité de simulation des codes obtenus. Après la comparaison de notre architecture avec quelques travaux récents nous avons remarqué que nos résultats sont compétitifs en termes de consommation énergétique.

Mots Clés : Compression vidéo, HEVC, CABAC, codeur entropique, FPGA, Matlab, ISE, éléments syntaxiques.

Abstract

The main goal of this project is the implementation on FPGA platform of binarizer/debinarizer algorithm in the HEVC/H.265 (*High Efficiency Video Coding*) Video compression standard. The binarization technique plays an essential role in HEVC standard at the encoder or decoder sides. Moreover, the binarization is one of the tree blocks that constituate the CABAC entropy encoder. The binarization is placed as the first bloc in the CABAC encoding in order to convert the non-binary syntactic elements (ES) because the CABAC uses only the binary elements on the other hand the debinarizer is placed as the last stage of the CABAC decoding for reconstruct the ES. The comparison results of our CABAC binarizer/ debinarizer implementation on Matlab and ISE show its good performance and the similarity of the obtained simulation codes. After comparing our architecture with some recent works, we noticed that our results are competitive in terms of energy consumption.

Key words: Video compression, HEVC, CABAC, entropy coding, FPGA, Matlab, ISE, syntactic elements

ملخص

الهدف الرئيسي من هذا المشروع هو التنفيذ على منصة FPGA لخوارزمية binarisateur / dé-binarisateur في معيار ضغط الفيديو HEVC / H.265 (ترميز الفيديو عالي الكفاءة). تلعب تقنية الترميز الثنائي دورًا أساسيًا في معيار HEVC على جانبي جهاز التشفير أو وحدة فك التشفير. علاوة على ذلك، فإن الترميز الثنائي هو أحد الترميز التي تشكل مشفر إنتروبيا CABAC. يتم وضع الترميز الثنائي باعتباره الترميز الأول في ترميز CABAC من أجل تحويل العناصر غير الثنائية (ES) لأن CABAC تستخدم العناصر الثنائية فقط من ناحية أخرى، يتم وضع أداة إزالة النواتج عند مخرج CABAC لإعادة هيكلة ES. تُظهر نتائج المقارنة لتطبيق CABAC الثنائي / المنحل على Matlab وISE الأداء الجيد والتشابه بين أكواد المحاكاة التي تم الحصول عليها. بعد مقارنة هندستنا ببعض الأعمال الحديثة، لاحظنا أن نتائجنا تنافسية من حيث استهلاك الطاقة.

الكلمات المفتاحية: ضغط الفيديو، HEVC، CABAC، ترميز، ES ، FPGA، Matlab ، ISE.

Liste des acronymes

A

AC : Composante Alternative

ADSL : Asymmetric Degital Subscriber Ligne

ASIC : Application Specifie Integrated Circuits

AVC: Advanced Vidéo Coding

B

B : Bidirectionnelle

Bin : Symboles Binaires

BinIndex : Index de chaque bit dans la chaine binaire

C

CABAC : Context Adavptive Binary Arithmetic Coding

CALR : cofficient absolute level remaining

CAVLC : Context Adaptive Variable Length Coding

CMax : Maximal bin Size (la taille maximale du code binaire)

CTB : Coding Tree Block

CTUs : Coding Tree Unit

CtxIdx : Context Index Adress

D

DB : Dé-Binarisation

DC : Composante Directe

DCT : Discret Cosine Transform

DVD : Digital Versatile Disc

E

EGK : Exp-Golomb K -ème ordre

F

FL : Longueur fixe

FSM : Finite State Machine

FPGA : Field-Programmable Gate Array

FPS : Frame Per Second

G

GSM : Global System Mobile

H

HD : Haute Definition

HEVC : High Efficiency Video Coding

I

ISE : Integrated Software Environment

ISO : Organisation International de Standardisation

ITU : International Telecommunication Union

J

JVT : Joint Video Team

L

LPS : Least Probable Symbol

LSC : Last significant coefficient

M

MPEG-2 : Moving Pictures Expert Group

MPS : Most Probable Symbol

P

P : Prédite

PPS : Picture Parameter Set

Q

QP : Paramètre de Quantification

Qstep : pas de quantification

R

RTL : Register Transfer Level

S

SAO : Sample Adaptive Offset

SCF : Significatif Coefficient Flag

SD : Standard Définition

SE : Eléments Syntaxiques

SPS : Sequence Parameter Set

SVH : Système Visuel Humain

Syn Val : Sytax Element Value

Syn Idx : Sytax Element Index

T

TB : Bloc de Transformation

TR : Rice Tronqué

TRU : Tronqué Rice Unaire

TRK : Tronqué Rice de K -ème ordre

TU : Tronqué Unaire

TV : Télévision

V

VCEG : Video Coding Expert Group

VHDL : Hardware Description Language

VLC : Variable Length Code

VPS : Video Parameter Set

Liste des figures

Figure 1.1 :	La chaine de transmission vidéo...……..	04
Figure 1.2 :	Représentation numérique d'une image…….	05
Figure 1.3 :	Les résolutions existantes pour une vidéo……..	05
Figure 1.4 :	Chaine de compression/décompression vidéo……...	06
Figure 1.5 :	Les principales normes de codage vidéo selon l'organisme de standardisation……	07
Figure 1.6 :	Schéma bloc du codec vidéo HEVC/H.265……...	10
Figure 1.7 :	L'intra et l'inter prédiction d'un block PU dans une séquence vidéo…	12
Figure 1.8 :	Les images de référence de l'inter prédiction P et B……..	12
Figure 1.9 :	Les tailles possibles pour les blocs de prédiction PBs en HEVC……...	13
Figure 1.10 :	Les 35 directions de l'intra-prédiction utilisés pour le HEVC……...	13
Figure 2.1 :	Système de communication d'informations……...	15
Figure 2.2 :	Exemple de codage arithmétique……...……	20
Figure 2.3:	Algorithme de calcul des intervalles……..	20
Figure 2.4 :	Schéma de principe de l'encodeur CABAC……...	21
Figure 2.5 :	Architecture du décodeur CABAC……	22
Figure 2.6 :	Architecture de binariseur proposé……	27
Figure 3.1 :	Outils matériels implémentation et logiciels utilisés dans notre implémentation……	36
Figure 3.2 :	Entité et Architecture d'un composant logique sur ISE……..	37
Figure 3.3 :	Organigramme de code Tronqué Unaire (TU)……	38
Figure 3.4 :	Organigramme de code Tronqué Race Unaire (TRU)……	39
Figure 3.5 :	Organigramme de code Fixed_Length (FL)……...	40
Figure 3.6 :	Organigramme de code Tronqué Rice K -ème (TRK)…….	41
Figure 3.7 :	Organigramme de code Exp-Golomb (EGK)…….	44
Figure 3.8 :	Organigramme de code CALR……...	47
Figure 3.9 :	Organigramme de code QP Delta……...	48
Figure 3.10 :	Organigramme de code Part Mode…….	49
Figure 3.11 :	Organigramme de code Intra prédiction mode……...	52
Figure 3.12 :	Organigramme de code Inter Prédiction Mode……...	53
Figure 3.13 :	Simulation sous Matlab des dix codes de binarisation (format unique,	

	format combiné, format personnalisé) ……	54
Figure 4.1:	Préparation du flux au niveau de transformation pour un bloc TU 4*4.	61
Figure 4.2 :	Les coefficients résiduels et la position (x, y)……..……...	61
Figure 4.3 :	Schéma bloc de décodage niveau bloc de transformation……..	62
Figure 4.4 :	Architecture proposée de binarisation / dé-binarisation CABAC HEVC……..	63
Figure 4.5 :	Conception matérielle proposée du code TU……..….	64
Figure 4.6 :	Conception matérielle proposée du code TRU……..…..	64
Figure 4.7 :	Conception matérielle proposée du code FL……...….	65
Figure 4.8 :	Conception matérielle proposée du code TRK…….……..	66
Figure 4.9 :	Conception matérielle proposée du code EGK…….……..	67
Figure 4.10 :	Conception matérielle proposée du code CALR……..……...	68
Figure 4.11 :	Conception matérielle proposée du code QP Delta……....	68
Figure 4.12 :	Conception matérielle proposée du code Intra Prédiction…….…….	69
Figure 4.13 :	Conception matérielle proposée du code Inter Prédiction…… ….	69
Figure 4.14 :	Conception matérielle proposée du code Part Mode…….….	70
Figure 4.15 :	Architectures proposées pour la dé-binarisation. (a) pour les résidus de coefficients. (b) pour la position de dernier coefficient non nul (LSCx, LSCy)……..	71
Figure 4.16 :	Diagramme d'état FSM proposé pour le binarisateur ……	72
Figure 4.17 :	Schéma RTL du code TU. (a) Top-Level, (b) Schéma détaillé……..	73
Figure 4.18 :	Schéma RTL Top-Level du code TRU……...	74
Figure 4.19 :	Schéma RTL Top-Level du code FL.…….	74
Figure 4.20 :	Schéma RTL Top-Level du code TRK …….....	75
Figure 4.21 :	Schéma RTL Top-Level du code EGK. …….....	75
Figure 4.22 :	Schéma RTL Top-Level du code QP Delta ...……	76
Figure 4.23 :	Schéma RTL Top-Level du code CALR. ……..	76
Figure 4.24 :	Schéma RTL Top-Level du code Intra Prédiction Mode……	77
Figure 4.25 :	Schéma RTL Top-Level du code Inter Prédiction Mode ……...	77
Figure 4.26 :	Schéma RTL Top-Level du code Part Mode.....	78
Figure 4.27 :	Schéma RTL Top-Level des coefficients résiduels……	78
Figure 4.28 :	Schéma RTL Top-Level de la position (x, y)…….....	79
Figure 4.29 :	Description en VHDL de l'architecture du binarisateur/dé-binarisateur	

CABAC HEVC sous ISE 14.5…… 80
Figure 4.30 : Simulation sous Model Sim du code TU…… 83
Figure 4.31 : Simulation sous Model Sim du code FL…… 83
Figure 4.32 : Simulation sous Model Sim du code TRU…….. 83
Figure 4.33 : Simulation sous Model Sim du code TRK…….. 84
Figure 4.34 : Simulation sous Model Sim du code EGK……. 84
Figure 4.35 : Simulation sous Model Sim du code CALR……... 84
Figure 4.36 : Simulation sous Model Sim du code QP Delta……... 85
Figure 4.37 : Simulation sous Model Sim du code Intra Prédiction Mode…….. 85
Figure 4.38 : Simulation sous Model Sim du code Inter Prédiction Mode…….. 85
Figure 4.39 : Simulation sous Model Sim du code Part Mode……. 86
Figure 4.40 : Simulation sous Model Sim du la position (x,y)……. 86
Figure 4.41 : Simulation sous Model Sim des coefficients résiduels……... 86

Listes des tableaux

Tableau 1.1 :	Les normes MPEG et leurs applications……..	08
Tableau 1.2 :	Comparaison entre les standards H.264 et H.265…….	09
Tableau 2.1 :	Les partitions des éléments syntaxiques au niveau CTU pour le slice type I…….	25
Tableau 2.2 :	Code tronqué unaire, tronqué race unaire et de longueur fixe de binarisation utilisés par le HEVC pour une longueur de 7……..	28
Tableau 2.3 :	Code Rice tronqué (TRK). Valeurs cMax déduites des règles de binarisation HEVC à tout ordre (K) donné……...……..	29
Tableau 2.4 :	Code Exp-Golomb (EGK) d'ordre inverse…….....	29
Tableau 2.5 :	Binarisation pour inter prédiction…….	32
Tableau 2.6 :	Binarisation pour le mode Intra Prédiction……..	33
Tableau 2.7 :	Binarisation pour part mode…….……	33
Tableau 4.1 :	Ressources, fréquences d’opération, chemin critique et puissance consommée pour les codes de binarisation/dé-binarisation...……..	81

Sommaire

Introduction générale……..... 01

Chapitre 01 : Généralités sur la compression vidéo

1.1 Introduction…….....……..……. 03
1.2 Généralités sur la compression vidéo…….…… 03
1.2.1 Compression et codage vidéo, une technique toujours nécessaire ? …….. 03
a) Principe de base de la compression vidéo……... 04
b) Notions de base sur la vidéo……..…….. 05
1.2.2 Techniques de compression vidéo…….. 06
1.2.3 Définition et historique de la compression vidéo …….. 06
1.2.4 Codeurs vidéo……..... 07
a) Evolution des normes de compression vidéo et standardisation……..... 07
b) Comparaison entre les standards H.264 et H.265 ……... 08
1.3 Codeur HEVC/H.265……..... 09
1.3.1 Définition…….……..…….. 09
1.3.2 Organisation de la norme HEVC…… 10
1.3.3 Transformation…….…... 10
1.3.4 Quantification…….. 11
1.3.5 Prédiction…… 12
a) Inter Prédiction……..……..... 12
b) Intra Prédiction…….……... 13
1.3.6 CABAC…….…….. 14
1.4 Conclusion…….…… 14

Chapitre 02 : Codage entropique CABAC

2.1 Introduction…….... 15
2.2 Système de communication……..... 15
2.2.1 Codage source……..……. 16
2.2.2 Codage canal……. 16
2.2.3 Modulation……..... 16
2.3 Généralités sur la théorie de l'information……. 17

2.3.1 Définition de l'information…….…….. 17
2.3.2 Entropie……. 17
2.3.3 Codage entropique…… 18
2.3.3.1 Codage VLC…… 18
2.3.3.2 Codage de Shannon-Fano……... 18
2.3.3.3 Codage du Huffman…….... 19
2.3.3.4 Codage Arithmétique……...… 19
2.4 Codage CABAC…… 21
2.4.1 Codeur arithmétique binaire BEA…… 22
2.4.2 Sélection et modélisation du contexte……... 23
2.4.3 Binarisation……... 23
2.5 Elément syntaxique…… 24
2.6 Codes binaires utilisé dans la norme HEVC…….. 26
2.6.1 Architecture supérieure…… 26
a) Unaire Tronqué (TU)…….... 27
b) Longueur Fixe (FL)……... 27
c) Rice Tronqué de K -ème ordre (TRK)…….. 28
d) Golomb Exponentiel (EGK)……. 29
e) Coefficient Absolu Niveau Restant (CALR)…… 30
f) QP Delta…… 31
g) Inter Prédiction Mode……..... 32
h) Intra Prédiction Mode…… 32
i) Part Mode…….. 33
2.7 Conclusion…….. 34

Chapitre 03 : Programmation des codes de binarisation sous Matlab

3.1 Introduction…… 35
3.2 Outils utilisés…….. 35
3.2.1 Xilinx ISE 14.5……. 36
3.2.2 Model Sim (PE Student Edition 10.4a)…… 37
3.2.3 MATLAB……. 37
3.3 Algorithme de binarisation /dé-binarisation……... 37
3.3.1 Organigramme…….. 38

a) Code TU…… 38
b) Code TRU…… 38
c) Code FL……..... 39
d) Code TRK…… 40
e) Code EGK…… 44
f) Code CALR……..…... 47
g) Code QP Delta…….. 47
h) Part Mode…….. 48
i) Intra Prédiction Mode……... 52
j) Inter Prédiction Mode……... 52
3.3.2 Simulation sous MATLAB……... 53
3.4 Conclusion……..…… 59

Chapitre 04 : Conception et Implémentation des codes de binarisation / dé-binarisation sur FPGA

4.1 Introduction…… 60
4.2 Préparation et récupération des flux binaire……... 60
4.3 Architecture proposée globale de binarisation / dé-binarisation…… 62
4.3.1 Architectures proposées pour les codes de binarisation…….. 64
a) Architecture du code TU……... 64
b) Architecture du code TRU…… 64
c) Architecture du code FL……... 65
d) Architecture du code TRK…… 66
e) Architecture du code EGK…… 67
f) Architecture du code CALR……. 68
g) Architecture du code QP Delta……. 68
h) Architecture du code Intra Prédiction Mode…… 69
i) Architecture du code Inter Prédiction Mode…… 69
j) Architecture du code Part Mode……... 69
4.3.2 Architecture proposée pour la dé-binarisation……. 71
4.3.3 FSM (*finite state machine*)……... 72
4.4 Partie synthèse et simulation…….. 73
4.4.1 Synthèse et schéma RTL des codes de binarisation/dé-binarisation…… 73

A) Binarisation……... 73
B) Dé-binarisation…….. 78
4.4.2 Ressources, fréquences d'opération, chemin critique et puissance consommée…… 80
4.4.3 Simulation des codes de binarisation/dé-binarisation…….……...…… 83
A) Binarisation……... 83
B) Dé-binarisation…….. 86
4.5 Conclusion…….. 87

Conclusion générale……. 88

Introduction générale

Introduction générale

De nos jours, La plupart des appareils électroniques ont la capacité de capturer et de stocker 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.

La standard H.265/HEVC (*High Efficiency Video Coding*) est le codeur vidéo qui est actuellement utilisé, le plus récent et qui assure une meilleure capacité de compression. Le codage arithmétique CABAC (*Context Adaptive Binary Arethmetic Coding*) est l'une des techniques les plus importants adoptés en HEVC ; il est utilisé pour coder les éléments syntaxiques codés avant la transmission. En général, le CABAC se compose de trois processus : binarisation, modélisation du contexte et codage arithmétique au niveau de décodage les opérations inverses sont appliquées [1]. La binarisation convertit les éléments syntaxiques d'entrée en chaînes binaires. Les deux autres parties transforment les chaînes binaires en flux binaire selon les modèles de probabilité définis dans la norme HEVC [2].

La binarisation HEVC CABAC possède plusieurs formats de binarisation de base tels que U (*Unaire*), TU (*Unaire Tronqué*), TR (*Rice Tronqué*), EGK (*Exp-Golomb d'ordre k*) et FL (*Longueur Fixe*). Presque tous les éléments syntaxiques exploitent ces formats de binarisation de base, mais certains éléments syntaxiques tels que CALR (*Coefficient Absolute Level Remaining*) et QP Delta (*cu_qp_delta_ abs*) exploitent deux ou plusieurs combinaisons de binarisation de base. Certains éléments syntaxiques tels que : Inter Prédiction Mode (*inter_pred_idc*), Intra Prédiction Mode (*intra_ pred_mode*) et part mode sont basés sur des codes spécifiques.

Une nouvelle contrainte au débit du codage et du décodage HEVC (vitesse ou througput). Par conséquent, le throughput demeure toujours un défi à surmonter par les chercheurs dans ce domaine. Pour cela, différentes solutions technologiques sont largement exploitées pour répondre aux exigences des systèmes de communications telles que les ASIC (*Application Specific Integrated Circuit*), les FPGA (*Field Programmable Gate Array*) et les DSP (*Digital Signal Processor*).

C’est dans ce contexte que s’inscrivent notre travail, nous nous sommes intéressées à l’implémentation d’un binarisateur /dé-binarisateur dans la norme de compression vidéo

HEVC/H.265 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. Ensuite, nous détaillerons la norme H.265/HEVC 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 et ses types plus précisément le codage CABAC. Nous terminerons ce chapitre par l'étude des codes de binarisation qui fait l'objet de notre travail.

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

Nous mettrons l'accent sur la préparation et la récupération du flux binaire au début de quatrième chapitre. Ensuite, l'architecture proposée du binarisation/ dé-binarisation dans la norme HEVC sera présentée et en enchaine par la proposition des architectures pour chaque code de binarisation. La partie contrôle est donnée par une FSM qui assure la gestion globale du système. A la fin du chapitre, une synthèse, simulation et évaluation de la consommation énergétique sont assurés par l'ISE et le Model Sim suivi par des commentaires sur les résultats obtenus.

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

Chapitre 01

Généralités sur la compression vidéo

1.1 Introduction

Durant la dernière décennie, les domaines d'applications et les besoins en matière de compression vidéo ont évolués. Pour cela, plusieurs normes de codage vidéo ont été adoptées. De plus, l'émergence des normes de compression vidéo a permet d'atteindre une meilleure efficacité de compression à qualité visuelle équivalente. Ces codeurs de compression vidéo sont normalisés par des institutions de standardisation. L'objectif de ce chapitre est d'étudier quelques techniques de compression utilisées. Nous allons donc commencer par présenter les principes de la compression vidéo. Ensuite, nous introduirons les normes de codage vidéo et nous détaillerons le standard HEVC/H.265.

1.2 Généralités sur la compression vidéo

1.2.1 Compression et codage vidéo, une technique toujours nécessaire ?

L'utilité de la compression n'est plus à prouver : les mobiles GSM (*Global System Mobile*) (nos téléphones portables) ne pourrait exister sans compression du signal audio faute de débit suffisant, de même pour la télévision sur ADSL (*Asymmetric Degital Subscriber Ligne*). En effet, si l'on considère une séquence de télévision en SD (*Standard Definition*) contenant 4 composantes de luminance pour 4 composantes de chrominance (format 4: 2: 2) dont chacune d'entre elles est représentée par 8 bits (soit 16 bits par pixel) avec une fréquence de 24 images seconde, le débit nécessaire sans compression est de 720×576×16×24 = 159,25 Mbits/s alors que le débit maximum proposé par les fournisseurs d'accès, pour une connexion ADSL, est de 30 Mbits/s. Plus encore, Pour certaines applications et certains équipements, par exemple les lecteurs DVD (*Digital Versatile Disc*), la transmission de la télévision numérique, la vidéo conférence, la sécurité vidéo et les caméras numériques, une bande de transmission limitée ou une capacité mémoire limitée contribuent à une demande de rapports de compression toujours plus élevés [3].

Un troisième point qui met en évidence ce qu'on appelle les propriétés du système visuel humain (*SVH*) [4], en effet, une scène vidéo se compose de plusieurs objets, chacun avec ses formes, textures, éclairages, couleurs, etc. Le mouvement, la couleur et la luminosité d'une scène sont interprétées par les systèmes visuels humains, ou certains aspects sont moins

sensibles à l'œil humain que d'autres, par exemple, l'œil humain détecte mieux le vert que le bleu, dans un environnement sombre, uniquement le noir et le blanc peuvent être détectés [3].

Par ailleurs, l'image formée sur la rétine reste pendant 10 à 30 millisecondes, en conséquence, l'œil humain ne peut pas détecter la différence entre une vidéo à 60 fps *(Frame Par Second)* et à 120 fps et ne peut percevoir les objets détaillés.

a) Principe de base de la compression vidéo

La compression vidéo (Figure 1.1) vise à éliminer les redondances spatiales ou temporelles qui peuvent se présenter ; pour cela on distingue deux types de compression : sans perte et avec perte. La première catégorie est la compression sans perte, conduisant à une reconstruction parfaite. Ce type de compression est notamment utilisé dans des domaines où l'on souhaite comprimer des informations, sans toutefois introduire une quelconque erreur sur des donnes hautement sensibles [5]. D'un point de vue plus général, la compression sans perte ne permet pas d'atteindre des taux de compression suffisamment élevés. Ainsi les efforts actuels portent plus spécifiquement sur l'élaboration de nouvelles techniques en compression d'images avec pertes. La différence majeure, en comparaison avec la première catégorie, est que l'on ne cherche plus à obtenir une reconstruction identique à la source. On tolère ainsi une approximation qui n'induit pas de dégradations visuellement perceptibles. Bien évidemment, tout le savoir-faire réside dans la maîtrise de ces pertes. L'enjeu de cette compression avec pertes est de rechercher le meilleur compromis entre la qualité de l'image reconstruite et le taux de compression atteint.

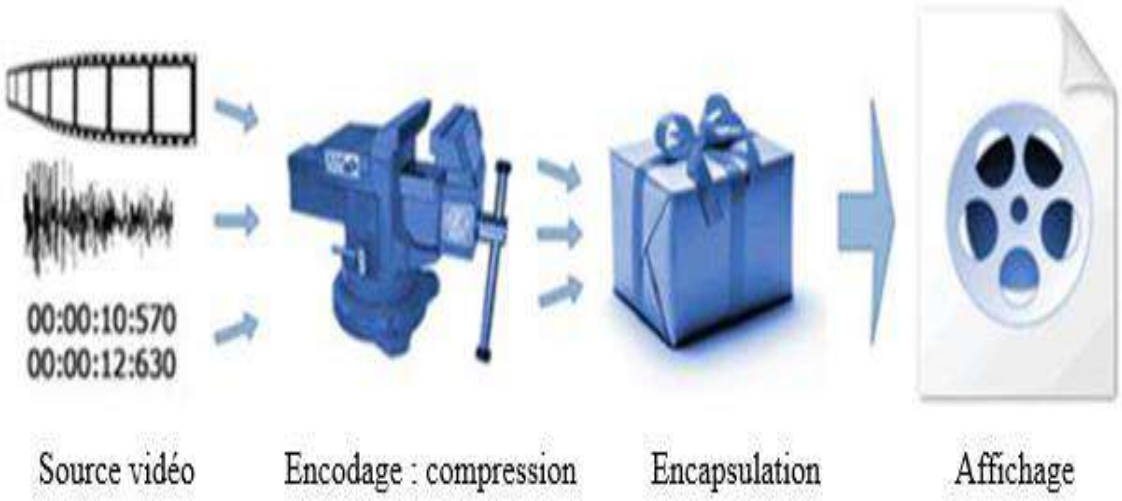


Figure 1.1 : La chaine de transmission vidéo [6].

b) Notions de base sur la vidéo

Une vidéo numérique est constituée d'une suite d'images numériques. Chaque image est formée d'une matrice de pixels (Figure 1.2).

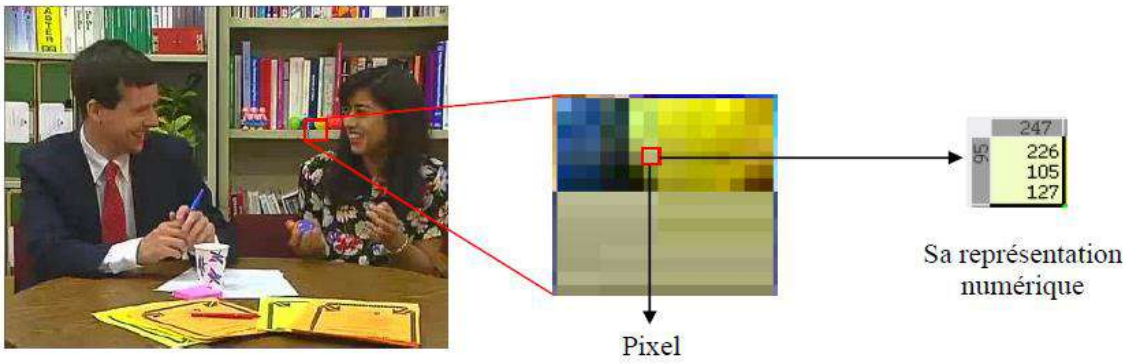
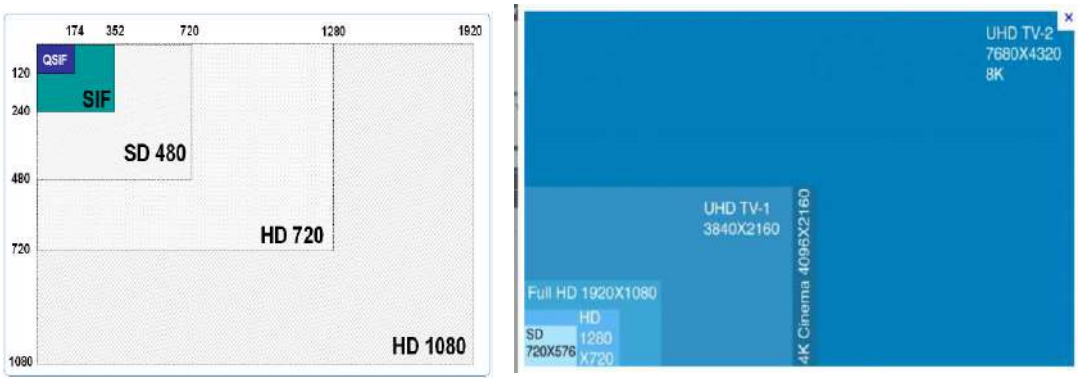


Figure 1.2 : Représentation numérique d'une image [6].

L'œil humain est capable de distinguer le rafraichissement des images uniquement quand la fréquence de mise à jour est inférieure à 25 images par seconde. On caractérise la fluidité d'une vidéo par sa résolution temporelle, c'est à dire par le nombre d'images par seconde *(frame rate)* exprimé en fps [7]. Ce dernier est toujours supérieur à 25 fps et peut atteindre pour certaines applications 100 fps. Une séquence vidéo est aussi caractérisée par sa résolution spatiale. Cette caractéristique définit le nombre de lignes et le nombre de colonnes de pixels composant chacune des images.



(a) Résolution atteint l'HD. (b) Résolution atteint l'UHD.

Figure 1.3 : Les résolutions existantes pour une vidéo [6].

A titre d'exemple, la résolution spatiale d'une image extraite d'une vidéo haute définition (*HD*) est de 1920 pixels en largeur et de 1080 pixels en hauteur. Récemment la résolution a augmenté grâce aux développements de nouveaux standards de compression vidéo

notamment le HEVC ; ce dernier peut atteindre jusqu'à la résolution 8K (7680 × 4320) (Figure 1.3).

1.2.2 Techniques de compression vidéo

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 (Figure 1.4) [8].

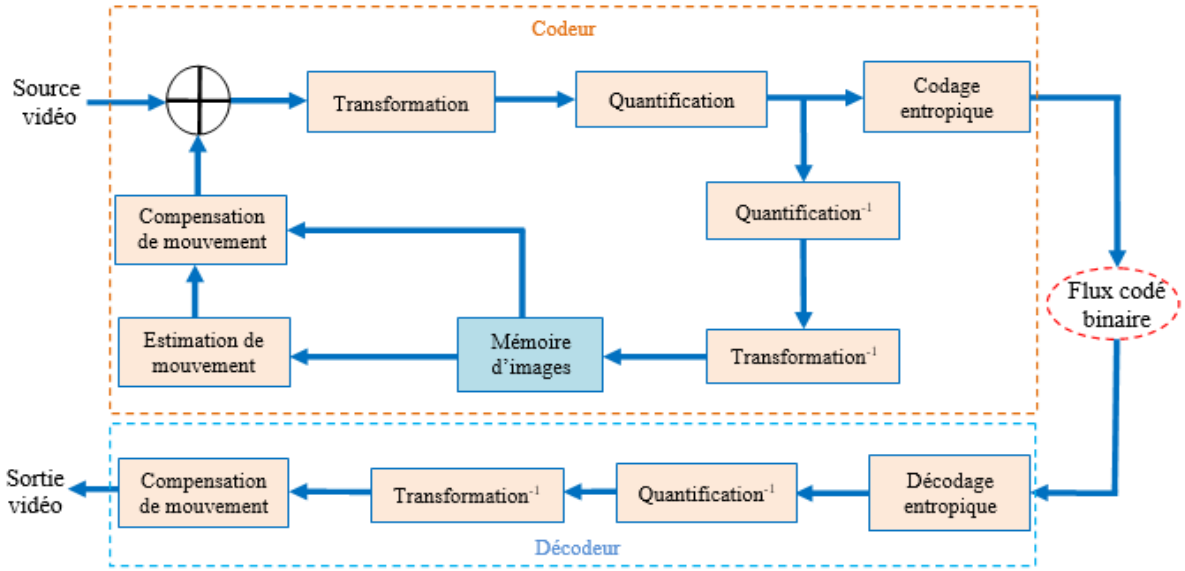


Figure 1.4 : Chaine de compression/décompression vidéo [8].

1.2.3 Définition et historique de la compression vidéo

Les algorithmes de compression vidéo ont un grand nombre d'applications allant de la vidéoconférence à la vidéo à la demande aux téléphones vidéo. Les normes de compression vidéo (telles que MPEG (*Moving Picture Expert Group*) 1, 2, 4, 7) et les normes de téléconférence (telles que H.26X) sont des algorithmes essentiels utilisés dans ces applications et d'autres applications multimédia [9].

Le codage vidéo est le processus de compression et de décompression d'un signal vidéo numérique.

Les technologies de compression vidéo sont en train de réduire et de supprimer les données vidéo redondantes afin qu'un fichier vidéo numérique puisse être envoyé efficacement sur un réseau et stocké sur des disques d'ordinateur. Avec des techniques de compression efficaces, une réduction significative de la taille du fichier peut être obtenue avec peu ou pas d'effets néfastes sur la qualité visuelle.

La qualité de la vidéo, cependant, peut être affectée si la taille du fichier est encore abaissée en augmentant le niveau de compression pour une technique de compression donnée [9].

Le but de la compression de la vidéo numérique c'est de prendre moins d'espace de stockage et de bande passante de transmission.

1.2.4 Codeurs vidéo

a) Evolution des normes de compression vidéo et standardisation

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

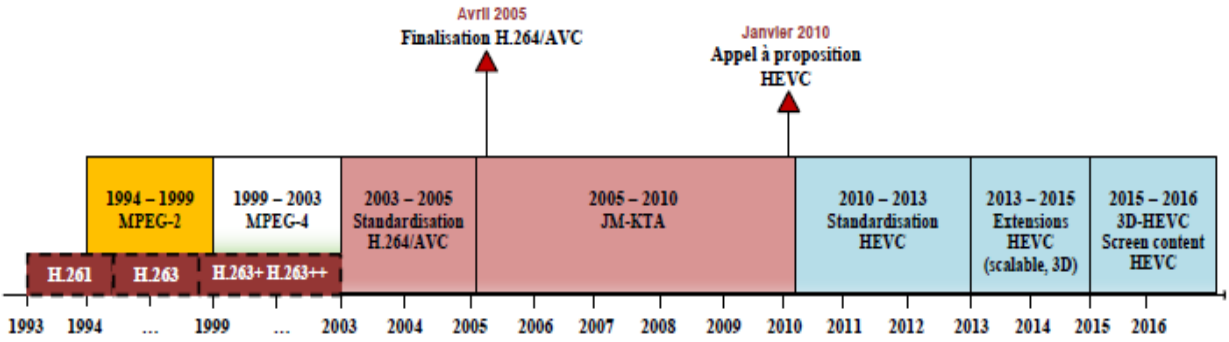


Figure 1.5 : Les principales normes de codage vidéo selon l'organisme de standardisation [6].

L'historique des différentes familles de normes H.26x et MPEG-x établie par l'ITU-T (*International Telecommunication Union*), MPEG ou JVT (*Joint Video Team*) est illustré sur la Figure (1.5) [6].

Parmi les codecs les plus répandus, nous citons :

- MPEG-2 [10] : Cette norme a été développée il y a environ 20 ans pour les applications liées à la TV (*Télévision*) numérique : professionnelle (production audiovisuelle, etc.) et grand public (diffusion vers les postes TV). Elle permet une transmission efficace des signaux de télévision par satellite, par câble ou terrestre. De plus, elle permet aussi la transmission d'autres médias sur l'Internet haut débit (xDSL : *Digital Subscriber Line*). Ce standard a été adopté pour les services de TV numérique par voie hertzienne terrestre et satellite. Il est également utilisé comme format de codage du DVD Vu que la bande passante disponible n'est pas toujours suffisante, l'utilisation de MPEG-2 devient limitée, indiquant ainsi un besoin de compression vidéo à améliorer encore.

- H.264 (ou MPEG-4 partie 10/AVC (*Advanced Video Coding*)) : a été développé en fin 2001, le VCEG (*Video Coding Expert Group*) et le MPEG ont formé une équipe de travail conjointe JVT pour mener à terme le standard de compression vidéo H.264/AVC. L'objectif consiste à doubler l'efficacité de compression en comparaison avec toutes les autres normes vidéo existantes [11]. Ce codec vidéo répond donc à la double dénomination H.264 (pour l'ITU) et à la partie 10 du MPEG-4 (pour l'ISO (*Organisation International de Standardisation*)). Il vise des applications de transmission des données multimédia via des canaux sans fil et de la diffusion de vidéos HD sur les ondes radio. Ainsi, MPEG-4 AVC/H.264 [12] permet de diminuer le débit nécessaire (jusqu'à 50 %) par rapport au MPEG-2, ouvrant la voie à de nouveaux services, tels que le vidéo haute définition.
- HEVC (*High Efficiency Video Coding*) : est développée par le groupe JVT en avril 2013 pour obtenir un gain en compression de 50% avec la même qualité visuelle que la norme (H.264) [13]. Cette norme est basée sur des outils et des techniques très sophistiqués, comme la prédiction temporelle et spatiale, la DCT (*Discret Cosine Transform*), la quantification, le filtrage en boucle et le codage entropique CABAC (*Context Adaptive Binary Arithmetic Coding*). Le HEVC peut supporter les contenus vidéo 4K et 8K avec une vitesse d'affichage qui peut aller jusqu'à 120 trames par second.

Tableau 1.1 : Les normes MPEG et leurs applications.

Standard	**Application**
MPEG-1	Vidéo-CD, CDI.
MPEG-2	Télévision numérique, Satellite, TNT, TV Haute Définition, DVD.
MPEG-4 Visual	Internet, vidéo mobile, VOD (Video On Demand), Studio.
MPEG-4 AVC H. 264	Internet, vidéo mobile, Haute Définition, VOD, Studio.
H.265/HEVC	Smartphone, Tablette, TV, 4K TV.

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

Le tableau (1.2) souligne les principales caractéristiques de la norme de compression vidéo H.265/HEVC et la norme antérieure H.264/AVC. Dans le point suivant on va parler en détail sur le codeur HEVC /H.265.

Tableau 1.2 : Comparaison entre les standards H.264 et H.265 [8].

Éléments de codage	**H.264/AVC**	**H.265/HEVC**
Date d'édition	2003	2013
Débit/ Espace	-44 -59% /H.263	-44 - 59% /H.264
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 spatiale : • Intra Prédiction ; • 9 modes.	**Prédiction spatiale :** • Intra Prédiction ; • 35 modes.
Codage entropique	CABAC et CAVLC	CABAC
Filtrage en boucle	Anti-bloc	Anti-bloc et SAO
Spécification	• Définition jusqu'à HD ; • Vitesse jusqu'à 60 fps.	• Support jusqu'à 8K (8192×4320) ; • Support jusqu'à 300 fps.

1.3 Codeur HEVC/H.265

1.3.1 Définition

Le codec HEVC est le dernier standard de compression vidéo, il est développé conjointement par les deux groupes des experts (MPEG et VCEG) afin de remplacer la norme H.264. Ce standard supportera les résolutions 1080p et Ultra HD. Une amélioration de 50% du taux de compression avec une qualité visuelle accrue de 25% est annoncée [14]. Cette amélioration des performances est obtenue au détriment de la complexité calculatoire des étapes de compression et de décompression.

Le codec HEVC est basé sur une approche hybride exploitant les redondances spatiales-temporelles et la transformation du signal vidéo. Il garantit un gain de compression très important indépendamment de la résolution des vidéos avec notamment plus d'efficacité dans des vidéos Ultra HD. Le H.265 a permis d'obtenir un gain en compression significatif, qui peut aller jusqu'à 50% par rapport à son prédécesseur le H.264 [15].

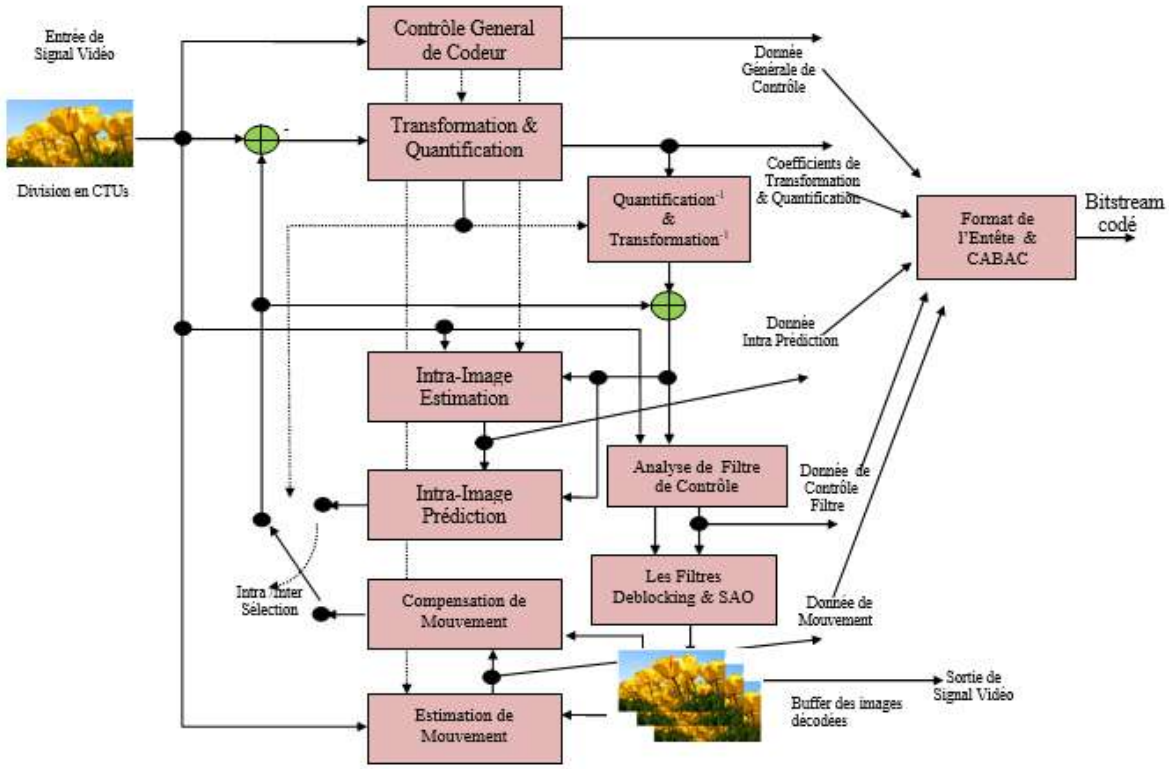


Figure 1.6 : Schéma bloc du codec vidéo HEVC/H.265 [8].

1.3.2 Organisation de la norme HEVC

Dans la norme H.265/HEVC, seule la partie décodage est normalisée, laissant le soin aux différents développeurs de sélectionner les paramètres d'encodage. En effet, certains outils de codage, par exemple les méthodes de prédiction, sont en compétition au niveau de l'encodeur. Il revient alors à l'utilisateur de choisir le plus approprié selon ses propres critères de choix et des applications. Ainsi, il existe différentes manières d'envisager les traitements au sein de l'encodeur en fonction des utilisations souhaitées.

1.3.3 Transformation

Cette technique est basée sur la conversion de l'image du domaine temporel vers le domaine fréquentiel. Son rôle principal est de préparer l'image au codage, pour augmenter le taux de compression et obtenir ainsi un gain en codage très intéressant. Dans la norme HEVC la transformation en DCT est adoptée.

Transformation en DCT

La transformée en cosinus discrète DCT (*Discret Cosinus Transform*), est une transformation mathématique qui transforme un ensemble de données d'un domaine spatial vers un domaine fréquentiel et inversement. De plus, elle est la plus utilisée [15] et la plus adoptée par toutes les normes courantes de codage vidéo, H.264 et H.265 à la fois pour sa simplicité algorithmique et ses performances. Cette opération permet de rassembler l'énergie du signal

sur un faible nombre d'échantillons. De plus, elle est calculée par une matrice carrée et son résultat sera représenté par une matrice de même taille [8].

Dans la norme HEVC, avant d'appliquer la transformation sur une image, il faut la partitionner en blocs de taille réduite (4 x 4, 8 x 8, 16 x 16 et 32 x 32 pixels) pour simplifier le calcul. Alors, chaque bloc subit d'une transformation mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel indépendamment des autres blocs [16] ce qui permet de lancer le traitement de plusieurs blocs en parallèle.

Après la transformation, la matrice résultante obtenue, contenant deux types de composantes : basses fréquences DC (*composante directe*) et hautes fréquences AC (*composante alternative*). Où le DC est placé sur la position (0,0), représente une moyenne de la grandeur de la matrice d'entrée et les coefficients des positions restantes sont des AC. Pour la DCT, l'information signifiante est concentrée en haut à gauche de la matrice de sortie autrement l'information moins signifiante est placée en bas à droite de cette matrice [8].

Dans le codeur HEVC, la transformée est suivie de la quantification des coefficients, étape à laquelle la perte d'informations vidéo est justifiée par le rapport de compression vidéo.

1.3.4 Quantification

Le but principal de la quantification est la rédaction 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'œil humain est plus sensible aux erreurs de reconstruction liées aux basses fréquences qu'à celles liées aux hautes fréquences.

Pour la quantification, le HEVC utilise essentiellement la même procédure de quantification, contrôlée par un paramètre de quantification (*QP*) comme dans la norme H.264/AVC. La gamme des valeurs QP est définie entre 0-51, 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 [8].

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 [16].

$$Qstep(QP) = (2^{1/6})^{QP-4} \quad (1.1)$$

1.3.5 Prédiction

Deux types de prédiction sont adoptées par le codec H.265 [17], l'intra prédiction et l'inter prédiction. La Figure (1.7) 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.

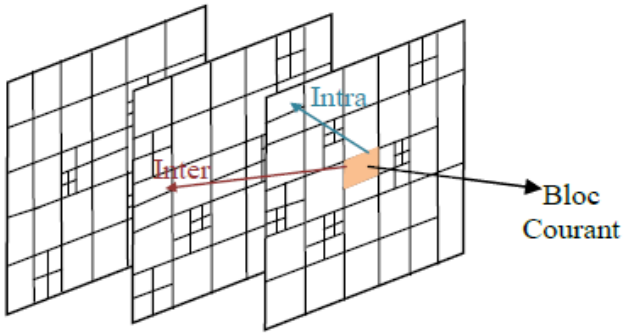
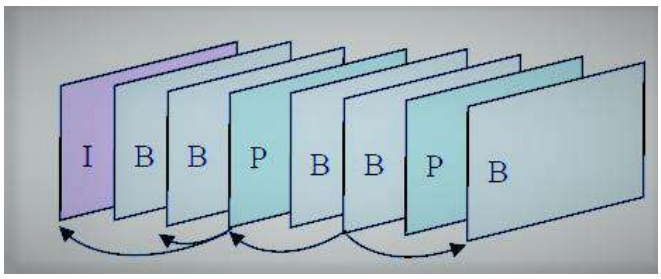


Figure 1.7 : l'intra et l'inter prédiction d'un block PU dans une séquence vidéo [8].

a) Inter prédiction

L'inter-prédiction est basée sur une technique de l'estimation du mouvement en éliminant les redondances. Selon la figure (1.7), cette estimation repose sur l'hypothèse qu'un bloc de pixels de l'image en cours peut être modélisé en tant que translation d'un bloc de l'image précédente [18]. En HEVC, plusieurs images décodées sont utilisées comme des images de référence pour prédire l'image courante. La Figure (1.8) montre que l'inter prédiction est utilisée pour les deux types images P (*Prédite*) et B (*Bidirectionnelle*). La première est basée sur les images codées antérieures (le bloc prédit est de type P) et la deuxième est basée sur les images antérieures et/ou postérieures futures (le bloc prédit est de type B). De plus, le rendement en compression des blocs B est supérieur à celui des blocs P. Cependant, un délai de décodage est produit, puisque les images P doivent être décodées avant les images de type B.



Légende :
I : intra image ;
B : bidirectionnelle ;
P : prédite.

Figure 1.8 : Les images de référence de l'inter prédiction P et B [8].

L'inter prédiction supporte différentes tailles de blocs symétriques et asymétriques où N est égal à 64 (Figure 1.9).

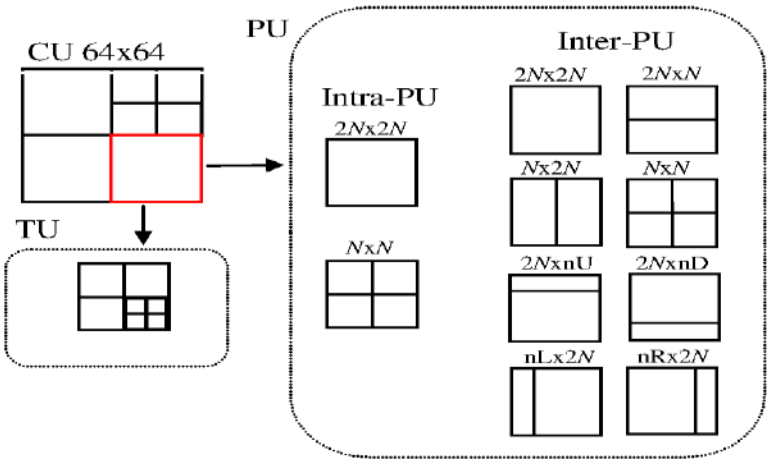


Figure 1.9 : Les tailles possibles pour les blocs de prédiction PBs en HEVC [19].

b) Intra prédiction

La norme HEVC permet l'élimination des redondances spatiales en utilisant l'intra prédiction. Les CTUs (*Coding Tree Unit*) dans image du type I sont codées en se référant à des informations dans l'image courante. La différence entre le bloc courant et le bloc prédit sera codée, ce qui favorise la réduction de l'information. Elle utilise seulement les blocs symétriques dont les tailles suivantes : 64 x 64, 32 x 32, 16 x 16, 8 x 8 et 4 x 4.

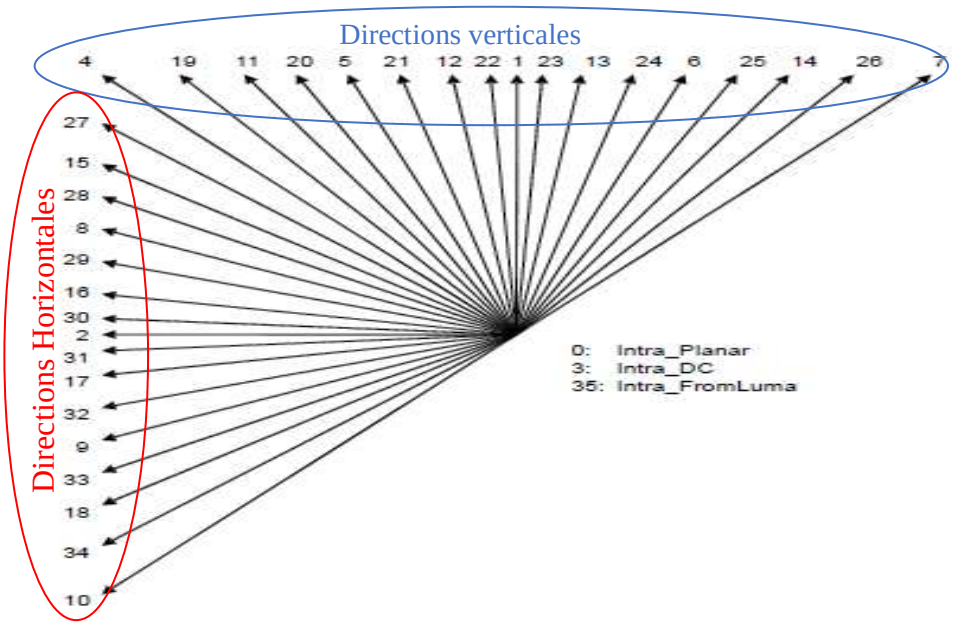


Figure 1.10 : Les 35 directions de l'intra-prédiction utilisés pour le HEVC [20].

L'intra utilise 35 modes de prédiction, classée en trois catégories :

- La prédiction angulaire qui est basée sur 33 modes (2, 3, …, 34) ;
- La prédiction planar qui contient un seul mode (0) ;
- La prédiction DC qui contient aussi un seul mode (1).

1.3.6 CABAC

CABAC (*Context Adaptive Binary Arethmetic Coding*) est un codage entropique binaire sans perte adapté au contexte employé dans la norme AVC, ainsi que dans la norme HEVC. Deux types de codage entropique CABAC et CAVLC (*Context Adaptive Variable Length Coding*) sont utilisés dans le codeur AVC, mais le HEVC adopte seulement le CABAC qui assure une réduction du débit binaire entre 9 à 14% par rapport au CAVLC. Le codage entropique CABAC réduit le nombre moyen de bits employés pour représenter l'information vidéo compressée. De plus, il convertit une série de symboles représentant les éléments syntaxiques (*SE*) de la séquence vidéo en un bit Stream prêt à transmettre ou à stocker. Ces éléments syntaxiques représentent le résiduel des coefficients, les vecteurs de mouvements, types de prévisions, les indications de quantifieur, les informations de partitionnement…etc. [8].

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. Puis, pour chaque bit un modèle de probabilité le plus adapté est sélectionné en se basant sur les données de voisinage pour optimiser l'estimation de cette probabilité. Enfin, un codage arithmétique est appliqué pour compresser les symboles binaires selon le mode de codage utilisé. Dans notre mémoire on s'intéresse à la première partie.

1.4 Conclusion

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

Chapitre 02

Chapitre 02

Codage entropique CABAC

2.1 Introduction

Le CABAC est une technique de codage entropique utilisée dans la norme H.264/AVC et également dans la norme HEVC avec une efficacité de codage élevée. Il est basé principalement trois fonctions : La binarisation, la modélisation du contexte et le codage arithmétique. Dans ce chapitre nous allons commencer par description de quelques notions de base de système de communication et théorie d'information. Ensuite, nous mettons l'accent sur le codage entropique et ses types. Le codage CABAC HEVC est présenté en détail pour la description de ses compositions. Nous terminons ce chapitre par l'étude des codes de binarisation sur la partie de transformation qui fait l'objet de notre thèse.

2.2 Système de communication

Dans le domaine des communications on s'intéresse aux moyens de transmettre une information depuis la source qui génère cette dernière jusqu'à un destinataire. La transmission de l'information consiste à une partie importante dans la chaine de communication. La chaîne de transmission utilisée dans les applications standards est représentée sur la Figure (2.1). L'émetteur représente l'ensemble des installations physiques et les opérations effectuées sur la sortie de la source avant la transmission : le codage source, le codage canal et la modulation. Tandis au niveau du récepteur, les opérations inverses sont réalisées : la démodulation, le décodage canal et le décodage de source [8].

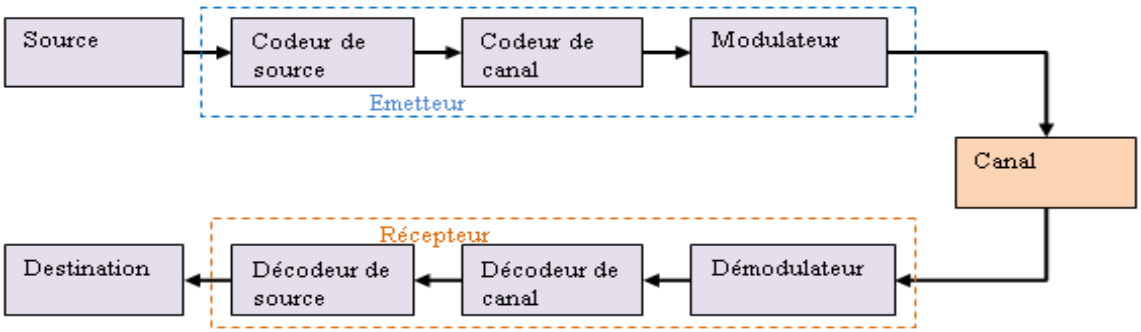


Figure 2.1 : Système de communication d'informations [8].

L'information générée par la source peut être par exemple de la vidéo, de la voix, d'un signal électromagnétique ou d'une séquence de symboles binaires. Tandis que, le canal peut être une ligne téléphonique, une liaison radio ou encore un support magnétique ou optique [8].

2.2.1 Codage source

Les applications multimédia actuelles manipulent des contenus hétérogènes, Ces contenus sont caractérisés par une forte redondance d'information. Afin de pouvoir les transmettre sur un canal physique et de respecter les contraintes de temps réel, il est nécessaire de réduire cette redondance d'information [18]. Le codage de source ou compression des données sert à fournir une représentation efficace des données avec un taux de compression important tout en conservant l'information pertinente qu'elles contiennent. Il peut s'effectuer avec ou sans perte d'information.

2.2.2 Codage canal

Le codeur de canal représente l'ensemble des opérations effectuées sur la sortie de codeur source avant la modulation. Son rôle est de réduire le taux d'erreur dues aux multiples distorsions que subit le message dans le canal de transmission. Il consiste à l'ajout d'une redondance pour combattre les effets du bruit. Comme pour le codage de source, Shannon a établi un deuxième théorème concernant le codage de canal montre qu'il est possible de transmettre des données numériques sur un canal même bruité presque sans erreur à un débit maximum calculable [21].

2.2.3 Modulation

L'opération de modulation vise à mettre le signal obtenu à la sortie du codeur canal sous une forme plus adaptée physiquement à la transmission sur le canal. De plus, 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 ou d'argument (phase/fréquence) d'une onde sinusoïdale appelée porteuse. La démodulation est obtenue en appliquant l'opération inverse pour extraire l'information transmise de la porteuse [8].

2.3 Généralités sur la théorie de l'information

Les recherches en théorie de l'information ont été initiées par Shannon vers l'année 1948, qui est une théorie probabiliste permettant de quantifier le contenu moyen en information d'un ensemble de messages. Ce domaine trouve son origine scientifique avec Claude Shannon qui en est le père fondateur avec son article. Parmi les branches importantes de la théorie de l'information, 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 x_i [8].

$$I(x) = f(p(x_i)) = -\log_2(p(x_i)) \quad (2.1)$$

Où $p(x_i)$ est la probabilité du symbole x_i, 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 définie comme une fonction d'état proportionnelle à la quantité de matière en présence. Par analogie, l'entropie de la théorie de l'information caractérise le degré de désorganisation du contenu en information d'une source [8].

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 x_i et donnée par l'équation (Eq.2.2) :

$$H(x) = -\sum_{i=1}^{n} p(xi) \log2\ (p(xi)) \qquad (2.2)$$

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

2.3.3 Codage entropique

Historiquement, le codage entropique a été développé dans les années 1940-1950 avec la théorie de l'information, le codage entropique est devenu une technique fondamentale en compression de données. En effet, il est présent dans de nombreux programmes de compression et de normes de compression d'image et de compression vidéo. 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 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 [22].

2.3.3.3 Codage de Huffman

Dans l'algorithme de Huffman, nous procédons différemment pour un résultat comparable. Nous commençons par choisir les deux symboles qui ont le moins d'occurrences, nous leur donnons les codes 0 et 1 et nous les regroupons dans un arbre binaire auquel nous attribuons un nombre d'occurrences : la somme des deux symboles qu'il regroupe. Ensuite, nous continuons : nous choisissons deux symboles ou arbres qui ont le moins d'occurrences et nous les regroupons. Nous regroupons ainsi deux par deux des symboles ou des arbres en fonction du nombre d'occurrences qu'ils représentent [2].

2.3.3.4 Codage arithmétique

Le codage arithmétique est une technique de codage très puissante. Il traite l'ensemble d'une séquence de symboles comme une seule entité ce qui lui permet de réussir à sortir de la contrainte imposée par le code Hoffman qui attribuent un mot de code court aux symboles probables, et des mots plus longs aux symboles moins fréquents [23]. Ainsi, le codage arithmétique permet de coder les séquences de symboles par un intervalle entre le 0 et 1. Cette opération de codage se fait par des subdivisions itératives de l'intervalle selon les probabilités des symboles issus par la source. Il est à noter que toutes les valeurs de cet intervalle peuvent représenter la valeur codée de la séquence. De plus, les symboles successifs réduisent cet intervalle en concordance avec la probabilité d'apparition du symbole.

Considérons un exemple d'une source de symboles est A = {a_1, a_2, a_3} de probabilité 0.6, 0.3, 0.1 respectivement. On souhaite encoder la séquence S = a_1 a_3 a_2 (figure 2.2) :

- L'intervalle I est [0.0, 1.0] est d'abord subdivisé en I_1 = [0.0, 0.6 [, I_2 = [0.6, 0.9 [et I_3 = [0.9, 1.0[.
- On sélectionne l'intervalle I_1 puisque le premier symbole de S est a_1. L'intervalle courant devient donc I = [0.0, 0.6[;

- On répète ces opérations pour le deuxième symbole a_3 en utilisant l'algorithme de calcul des deux limites d'intervalle [B_{inf}(S), B_{sup}(S) [; (voir la figure 2.3). L'intervalle courant devient alors I = [0.54, 0.6[;
- Enfin, après traitement du dernier symbole de S, l'intervalle final est égal à I = [0.576, 0.594[par ce que le dernier symbole à codé est a_2.
- Notons que la taille de l'intervalle final est égale à la probabilité de la séquence à émettre ;
- Le train binaire résultant de l'encodage est 100. Ces trois bits sont suffisants pour distinguer l'intervalle final de tous les intervalles obtenus en encodant d'autres séquences.

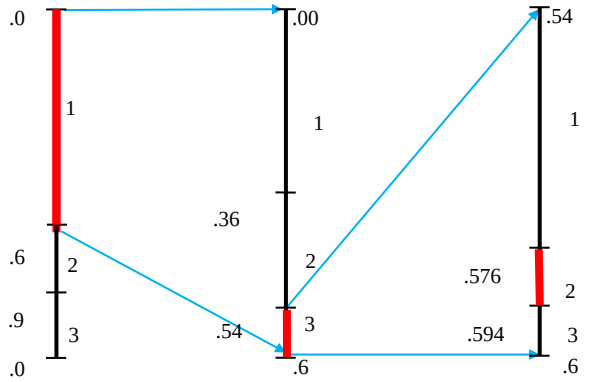


Figure 2.2 : Exemple de codage arithmétique [8].

Algorithme de calcul de l'intervalle

Initialisation

Binf = 0.0 ; Bsup = 1.0

a1 symbole à codé ;

Taille = Bsup – Binf ;

Bsup = Binf + taille * Bsup (a1) ;

Binf = Binf + taille * Bsup (a1) ;

Figure 2.3 : Algorithme de calcul des intervalles [8].

Récemment, les codes arithmétiques ont attiré l'attention des chercheurs dans le domaine de la compression par leur utilisation dans des standards de compression tels JPEG 2000, MPEG4, H.264, et H.265. 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.265 [24].

2.3 Codage CABAC

Le codage entropique est utilisé dans le codage vidéo pour compresser sans perte des éléments syntaxiques (par exemple, des vecteurs de mouvement, des modes de prédiction, des coefficients) en bits codés. Il s'agit de la dernière étape du codeur vidéo et de la première étape du décodeur vidéo. CABAC est une forme de codage entropique utilisée à la fois dans AVC et HEVC. Dans AVC, deux méthodes de codage entropique sont utilisées : CAVLC et CABAC. La méthode CABAC offre une efficacité de codage de 9 à 14 % supérieure à celle de la méthode CAVLC [25], mais la méthode CAVLC offre un meilleur débit. Dans HEVC, le CABAC est le seul moteur de codage entropique ; il doit donc fournir à la fois une efficacité de codage et un débit élevé.

Le codeur CABAC effectue un codage arithmétique binaire adaptatif grâce à une méthode de modélisation basée sur le contexte qui sélectionne un modèle de probabilité pour chaque élément de contexte.

Ainsi, l'encodeur CABAC se compose d'un binarisateur, d'un sélecteur et modélisateur de contexte et d'un codeur arithmétique binaire BAE, comme le montre la figure (2.4).

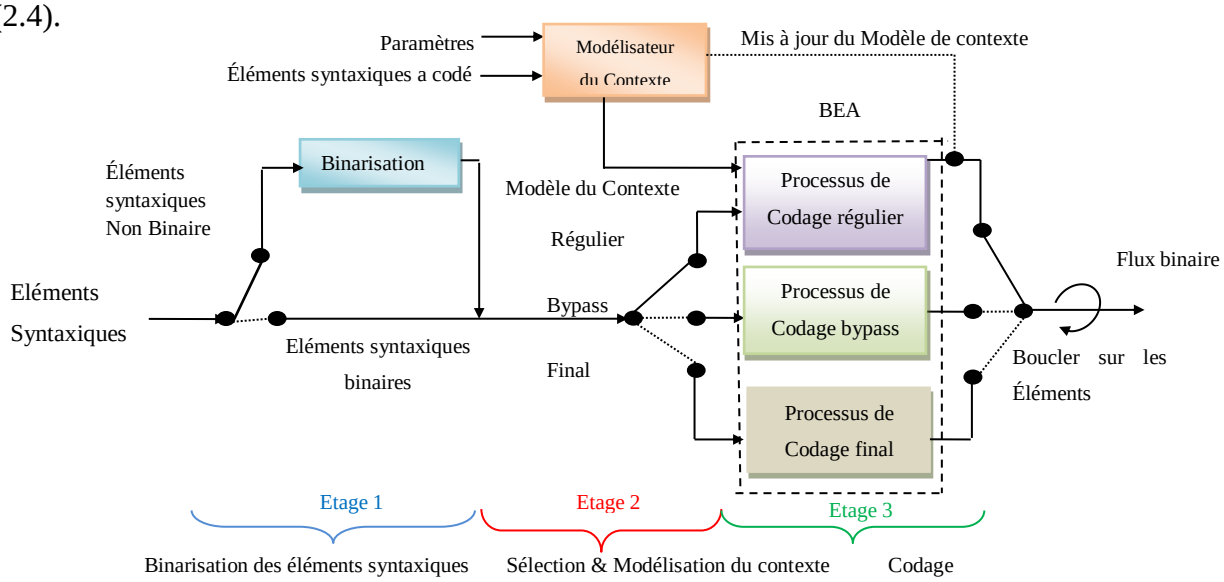


Figure 2.4 : Schéma de principe de l'encodeur CABAC [8].

La figure suivante montre le décodage arithmétique binaire d'un élément syntaxique se déroule en trois étapes principales. La figure (2.5) illustre les schémas synoptiques de décodage CABAC HEVC.

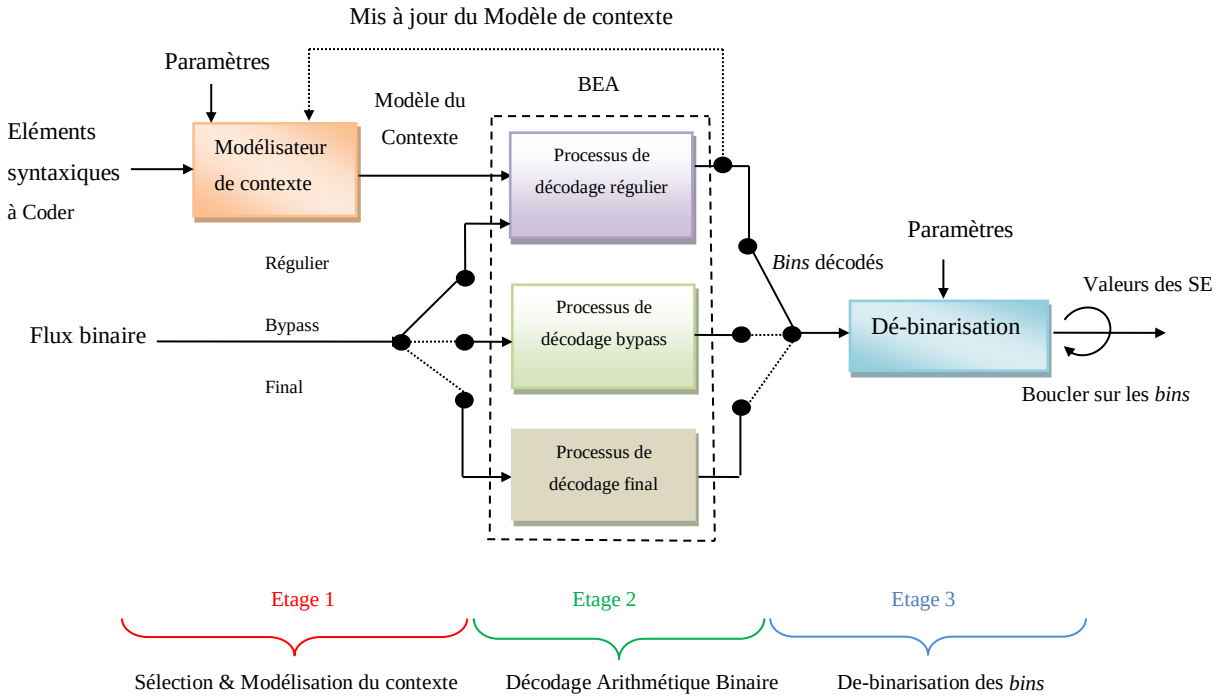


Figure 2.5 : Architecture du décodeur CABAC [8].

2.4.1 Codeur arithmétique binaire BEA

Le codage arithmétique binaire traite l'ensemble d'une séquence de symboles comme une seule entité en se basant uniquement sur la distribution de la probabilité des symboles à coder. L'utilisation du codage arithmétique adaptatif CABAC permet d'assigner un nombre de bits fractionnaire à chaque symbole à coder. Ainsi, les codes adaptatifs générés sont alors adaptés aux statistiques des symboles variables [8].

Le codage arithmétique binaire est basé sur trois modes de codages :

- Mode normal ou régulier (*regular mode*) qui utilise le modèle du contexte (*ctxIdx : Context Index Adress)* 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 : l'index de chaque*

bit dans la chaine binaire) et des propriétés des unités de codage spatialement voisines [8] ;

- Mode bypass (*bypass mode*) basé sur la probabilité équivalente ce mode de codage est la partie la moins complexe dans le codage arithmétique binaire. Aucun modèle de contexte n'est requis et, la possibilité d'apparence de *MPS (Most Probable Symbol)* est égale à la possibilité d'apparence de *LPS (Last Probable Symbol)* [8] ;
- Mode terminal (*terminate mode*) appelé à la fin de processus de codage ou de décodage de chaque CTU (*Coding Tree Unit*) 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, si non le dernier symbole est égal à 1, alors le mode terminal est en cours de traitement [8].

2.4.2 Sélection et modélisation du contexte

La modélisation et la sélection du contexte sont utilisées pour modéliser avec précision la probabilité de chaque bin. La probabilité de la case dépend du type d'élément syntaxique auquel elle appartient, de l'indice de la case dans l'élément syntaxique (par exemple, la case la plus significative ou la case la moins significative) et des propriétés des unités de codage spatialement voisines. Il existe plusieurs centaines de modèles de contexte différents utilisés dans le standard HEVC. Par conséquent, une grande machine à états finis est nécessaire pour sélectionner le contexte correct pour chaque case. En outre, la probabilité estimée du modèle de contexte sélectionné est mise à jour après le codage ou le décodage de chaque symbole binaire [26].

2.4.3 Binarisation

Le CABAC est un codage arithmétique qui ne traite que des symboles binaires « 0 et 1 ». A l'entrée de processus de codage on trouve des éléments syntaxiques binaires et non binaires. Pour cela, il faut convertir tous les symboles non binaires avant de passer à l'étape de codage. La binarisation transforme les éléments syntaxiques non binaire comme les coefficients résiduels, les vecteurs de mouvement, la position du dernier coefficient non nul en des symboles binaire *bins* (symbole binaire qui contient un ou plusieurs bits) [8]. Par conséquent, différents codes de binarisation sont adoptés par la norme HEVC :

- Rice tronqué (*TR*) ;
- Tronqué unaire (*TU*) ;
- K -ème ordre Exp-Golomb (*EGK*) ;
- Longueur fixe (*FL*) ;
- Coefficient Absolu Niveau Restant (*Coefficient Absolute Level Remaining*) (*CALR*) ;
- QP Delta ;
- Inter Prédiction Mode ;
- Intra Prédiction Mode ;
- Part Mode.

2.5 Eléments syntaxiques

La norme H265/HEVC organise le flux binaire (*bitstream*), généré par le codeur dans un flux en couches, où chaque couche est constituée d'un entête et d'un champ de données (Figure 2.4). Les éléments syntaxiques (*syntax elements*) décrivent comment la séquence vidéo peut être reconstruite du côté du décodeur.

Le CABAC utilise 48 éléments syntaxiques divisés en plusieurs groupes selon le type de la syntaxe : coefficients résiduels, prédiction, transformation, filtrage, partitionnement et quantification). Chaque groupe contient un nombre différent des éléments syntaxiques [2].

En général, la syntaxe est classée en deux niveaux :

a) **Le niveau haut :** contient les couches VPS (*Video Parameter Set*), SPS (*Sequence Parameter Set*), PPS (*Picture Parameter Set*) et l'entête de slice. Les éléments syntaxiques sont encodés avec une longueur variable ou fixe.

b) **Le niveau bas :** est réservé pour le champ de données de chaque slice. Chaque slice est constitué d'un nombre de CTU et chaque CTU est codée par les groupes d'éléments syntaxiques suivant :
   - Type de la CTU ;
   - Données de filtrage SAO *(Sample Adaptive Offset)* et pas de quantification (*QP*) ;
   - Prédiction intra et inter ;

- Paramètres du bloc de codage CTB *(Coding Tree Block)* et les données résiduelles.

Les éléments syntaxiques regroupés en quatre partitions illustrées sur le Tableau (2.1).

Tableau 2.1 : Les partitions des éléments syntaxiques au niveau CTU pour le slice type I.

Partitions information	**SE décodés avec le mode RM**	**SE décodés avec le mode BM**	**SE décodés avec le mode TM**
Pr1: CTU & CU	*split_cu_flag, cu_transquant_bypass_flag,*	n/a	*end_of_slice_segment_flag, end_of_sub_stream_one_bit*
Pr2: SAO & QP	*sao_merge_left_flag, sao_merge_up_flag, sao_type_idx_luma, sao_type_idx_chroma, cu_qp_delta_abs*	*sao_offset_abs, sao_offset_sign, sao_band_position, sao_eo_class_luma ,sao_eo_class_chroma, cu_qp_delta_sign_flag*	n/a
Pr3: Intra Prédiction	*part_mode, intra_chroma_pred_mode, prev_intra_luma_pred_flag*	*Rem_intra_luma_pred_mod ,*	n/a
Pr4: TU & résiduels data	*split_transform_flag,cbf_luma, cbf_cr,cbf_cb, transform_skip_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, coded_sub_block_flag, coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag*	*last_sig_coeff_x_suffix, last_sig_coeff_y_suffix, coeff_abs_level_remaining, coeff_sign_flag*	n/a

On s'intéresse à la partition (Pr4) de transformation qui utilise principalement les dix codes de binarisation. Les éléments syntaxiques de la position de dernier coefficient non nul est signalé au niveau de TU, mais les autres éléments syntaxiques tels que : sub-TB flag, la carte de signification des coefficients, le vecteur signe, le vecteur des coefficients plus grand que 1, le vecteur des coefficients plus grand que 2 et les informations résiduelles sont signalées au niveau sous-TB de 4x4. Les éléments syntaxiques de cette partition constituent environ 90% du flux binaire total [27].

En HEVC, la carte de signification est indiquée en transmettant deux éléments syntaxiques *significatif_coeff_flag* (*SCF*) et *last_significant_coeff_x / y* (*LSC*). Afin d'estimer l'emplacement des coefficients non nuls et la position du dernier coefficient significatif dans la TB *(bloc de transformation)* [28]. La position est binarisée avec le code TRU pour la partie préfixe et le code à longueur fixe FL pour la partie suffixe.

L'analyse des informations de coefficients transformés est appliqué sous-bloc par sous-bloc. Lors de derniers et de cinquième balayage, les valeurs absolues de données restantes de sous-bloc sont transmises en utilisant le *SE coeff_abs_level_remaining* [15]. Cette syntaxe est binarisée par la combinaison de deux types de codes : Truncated et ExGolomb avec une longueur maximale de chaîne binaire (bin string) de 32 [29].

2.6 Codes binaires utilisés dans la norme HEVC

2.6.1 Architecture supérieure

Le binarisateur proposé se compose de quatre groupes de sous-modules tels que le module contrôleur, les modules de format unique, les modules de format combiné et les modules de format personnalisé. Ses entrées sont synVal (*Sytax Element Value*) et synIdx (*Sytax Element Index*) qui déterminent le processus de binarisation spécifique pour l'élément syntaxique correspondant. Ses sorties sont la chaîne bin et la longueur bin de chaque élément syntaxique [19].

Le module contrôleur reçoit les synVal et synIdx du bus périphérique. Il catégorise le type de format de binarisation de chaque élément syntaxique en fonction de synIdx et l'affecte au sous-module correspondant [19].

Les modules de format unique sont les modules TU, TR (TRU, TRK), FL et EGK, qui traitent les formats de binarisation uniques correspondants. Les modules de format combiné sont les modules CALR (*coeff_abs_level_remaining*) et QP Delta (*cu_qp_delta_abs*), qui invoquent les modules de format unique et traitent leurs formats de binarisation combinés correspondants [19].

Les modules de formats personnalisés sont les modules Inter Prédiction Mode (*inter_pred_idc*), Intra Prédiction Mode (*intra_chroma_pred_mode*), et Part Mode

(*part_mode*). Ils traitent les formats de binarisation personnalisés et sont implémentés comme des tables de consultation [19].

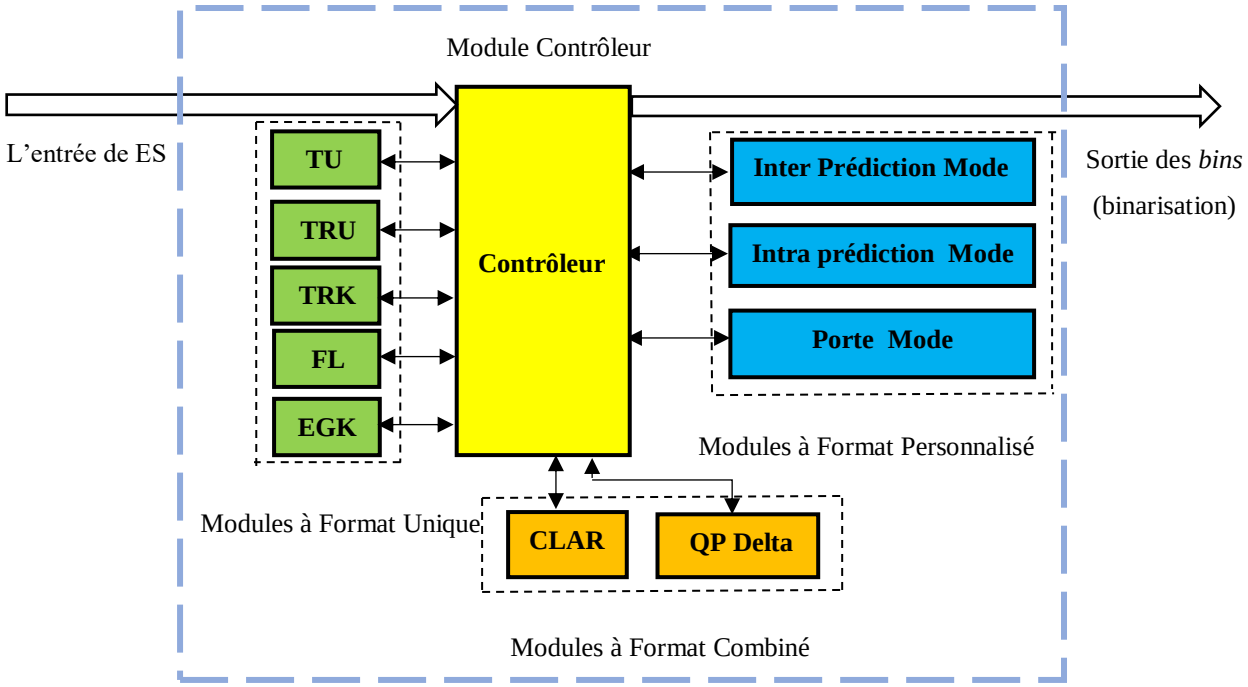


Figure 2.6 : Architecture du binarisateur proposé [19].

Maintenant on va parler sur les méthodes de codage utilisées pour la binarisation des éléments syntaxiques de codage résiduels.

a) Unaire tronqué (*TU*)

Le codage unaire tronqué est une méthode de codage le plus simple utilisée dans HEVC. Le codage tronconique (*TU*) offre une longueur de code initiale courte avec une croissance linéaire, la longueur étant déduite du code. Il en résulte un code flexible adapté à la binarisation d'éléments syntaxiques où la plupart des valeurs N sont petites, mais qui peut encore fonctionner pour des plages plus importantes [18].

b) Longueur fixe (*FL*)

Fixed_Length est une simple méthode de codage, il exige que la longueur soit connue à la fois du côté de l'encodeur et du décodeur. FL est donc peu flexible, mais convient

néanmoins pour certains éléments syntaxiques. Les éléments syntaxiques de longueur 1 (1 bit avec la valeur 0 ou 1), sont dits FL [18].

$$fixedLenght = (Log2(cMax + 1)) \quad (2.3)$$

Les codes de binarisation utilisés par le HEVC pour une longueur de 7 : code Tronqué unaire, tronqué race unaire et longueur fixe sont illustrés sur le Tableau (2.2).

Tableau 2.2 : code Tronqué unaire, tronqué race unaire et de longueur fixe de binarisation utilisés par le HEVC pour une longueur de 7.

N	Tronqué Unaire (*TU*)	Tronqué Race Unaire (*TRU*)	Longueur Fixe (*FL*)
0	0	0	000
1	10	10	001
2	110	110	010
3	1110	1110	011
4	11110	11110	100
5	111110	111110	101
6	1111110	1111110	110
7	11111110	1111111	111

c) Rice tronqué de K -ème ordre (*TRk*)

HEVC utilise la méthode de binarisation Rice tronquée de k-nième ordre. TRk d'ordre 0 - 4 est utilisé dans le processus de binarisation du niveau de Coeff abs_restant, où l'ordre dépend du niveau de Coeff abs_restant binarisé précédemment. TRk est similaire à Unaire et TU en ce sens que la longueur du code est déduite du code lui-même. La différence est que le code est divisé en une partie préfixe et une partie suffixe. Le préfixe est constitué du code TU, et le suffixe est une représentation binaire FL des bins les moins significatifs. Le suffixe a la longueur k, et le préfixe est incrémenté chaque fois que le suffixe déborde. Cela permet à TRk d'ajuster dynamiquement le compromis entre la longueur minimale des bins et la portée. La plus grande valeur de TRk est définie par la variable cMax *(Maximal Bin Size (Taille Maximale du Code Binaire))* [21].

Tableau 2.3 : Code Rice tronqué (TRk). Valeurs cMax déduites des règles de binarisation HEVC à tout ordre (k) donné.

N	K				
	0(cMax=3)	1(cMax=7)	2(cMax=15)	3(cMax=31)	4(cMax=63)
0	0	00	000	0000	00000
1	10	01	001	0001	00001
2	110	100	010	0010	00010
3	111	101	011	0011	00011
4	NA	1100	1000	0100	00100
5	NA	1101	1001	0101	00101
6	NA	1110	1010	0110	00110
7	NA	1111	1011	0111	00111

d) Golomb exponentiel (*EGk*)

HEVC utilise la technique de codage Exp-Golomb k-th en ordre inverse. EGk de 1 à 5ème ordre est utilisé dans la binarisation des Coeff-abs de niveau restant. Ce code utilise également un préfixe unaire, ainsi qu'un suffixe FL dont la longueur est égale à la longueur du préfixe + k. EGk n'a pas de limite de portée inhérente, car la longueur du suffixe est signalée avec le préfixe. Il s'agit d'une propriété importante car EGk est utilisé comme méthode de codage finale du niveau absolu [18].

Tableau 2.4 : code Exp-Golomb (EGk) d'ordre inverse.

N	K				
	1	2	3	4	5
0	00	000	0000	00000	000000
1	01	001	0001	00001	000001
2	1000	010	0010	00010	000010
3	1001	011	0011	00011	000011
4	1010	10000	0100	00100	000100
5	1011	10001	0101	00101	000101

6	110000	10010	0110	00110	000110
7	110001	10011	0111	00111	000111
8	110010	10100	100000	01000	001000

e) Coefficient Absolu Niveau Restant (*CALR*)

Le CALR *(Coefficient Absolute Level Remaining*) est utilisé pour binariser les valeurs de coefficient restantes obtenues à la suite du processus de quantification. La binarisation de CALR est essentiellement une concaténation du codage Rice tronqué et du codage Golomb exponentiel.

Les variables cLast_Abs_Level et cLast_Rice_Param sont dérivées comme suit :

- Si ce processus est invoqué pour la première fois pour l'indice de balayage i du sous-bloc en cours, cLast_Abs_Level et cLast_Rice_Param sont fixés à 0.
- Sinon (ce processus n'est pas invoqué pour la première fois pour l'index de balayage i du sous-bloc en cours), cLast_Abs_Level et cLast_Rice_Param sont définis comme étant égaux aux valeurs de cAbs_Level et cRice_Param, respectivement, qui ont été obtenues lors de la dernière invocation du processus de binarisation pour l'élément syntaxique coeff_abs_level_remaining, comme spécifié dans cette clause [30].

La variable cAbs_Level est égale à base Level + coeff_abs_level_remaining [n]. La variable cRice_Param est dérivée de cLast_Abs_Level et cLast_Rice_Param en tant que :

cRice_Param = Min (cLast_Rice_Param + (cLast_Abs_Level > (3 * (1 << cLast_Rice_Param)) ? 1: 0), 4) (2.4)

La variable cMax est dérivée de cRice_Param comme :

$$cMax = 4 << cRice_Param \quad (2.5)$$

La binarisation de l'élément syntaxique coeff_abs_level_remaining [n] est une concaténation d'une chaîne bin préfixe et (lorsqu'elle est présente) d'une chaîne bin suffixe.

Pour la dérivation du préfixe bin string, les règles suivantes s'appliquent :

- La valeur du préfixe de coeff_abs_level_remaining [n], prefixVal, est dérivée comme suit :

$$prefixVal = Min (cMax, coeff_abs_level_remaining [n] \quad (2.6)$$

- La chaîne bin du préfixe est spécifiée en invoquant le processus de binarisation TR prefixVal avec les variables cMax et cRice_Param comme entrées.

Lorsque la chaîne bin de préfixe est égale à la chaîne de bits de longueur 4 avec tous les bits égaux à 1, la chaîne bin de suffixe est présente et elle est dérivée comme suit :

- La valeur du suffixe de coeff_abs_level_remaining [n], suffixVal, est dérivée comme suit :

$$suffixVal = coeff_abs_level_remaining[n] – cMax \quad (2.7)$$

- La chaîne de binarisation du suffixe est spécifiée en invoquant le processus de binarisation EGk pour suffixVal avec l'ordre Exp-Golomb k fixé égal à cRice_Param + 1 [30].

f) QP Delta

Le QP Delta invoque une combinaison de code TU et de code EG0. La chaîne bin est une concaténation d'une chaîne bin préfixe et d'une chaîne bin suffixe (le cas échéant). La synVal, qui est égale au minimum de cu_qp_delta_abs et de cMax, invoque la binarisation de TU avec cMax comme autre entrée de TU. La suffixe Val, qui est égale à la différence entre cu_qp_delta_abs et cMax, invoque la binarisation EGk avec k comme autre entrée. QP Delta reçoit les sorties de TU et EGk comme entrées et calcule les sorties respectives.

Pour la dérivation du préfixe bin string, les règles suivantes s'appliquent :

- La valeur du préfixe de cu_qp_delta_abs, prefixVal, est dérivée comme suit :

$$prefixVal = Min(cu_qp_delta_abs, 5) \quad (2.8)$$

- La chaîne bin du préfixe est spécifiée en invoquant le processus de binarisation TR pour prefixVal avec cMax = 5 et cRice_Param = 0.

Lorsque prefixVal est supérieur à 4, le suffixe bin string est présent et il est dérivé comme suit :

- La valeur du suffixe de cu_qp_delta_abs, suffixVal, est dérivée comme suit :

$$suffixVal = cu_qp_delta_abs - 5 \quad (2.9)$$

- La chaîne bin du suffixe est spécifiée en invoquant le processus de binarisation EGk pour suffixVal avec l'ordre Exp-Golomb k fixé à 0 [30].

g) Inter Prédiction Mode

Il existe trois modules de format personnalisé dans binarisateur. Tous les modules de format personnalisé peuvent être conçus avec des tables de consultation [17].

Le processus de binarisation pour l'élément syntaxique en utilisant le module Inter Prédiction. Les entrées de ce processus sont une demande de binarisation pour l'élément syntaxique inter_pred_idc, la largeur actuelle du bloc de prédiction luma nPbW, et la hauteur actuelle du bloc de prédiction luma nPbH. La binarisation pour l'élément syntaxique inter_pred_idc est spécifiée dans le tableau (2.5) [30].

Tableau 2.5 : Binarisation pour inter prédiction [30].

Valeur de l'inter_pred_idc	**Nom de l'inter_pred_idc**	**Bin string**	
		(nPbW + nPbH) != 12	(nPbW + nPbH) = = 12
0	PRED_L0	00	0
1	PRED_L1	01	1
2	PRED_BI	1	-

h) Intra Prédiction Mode

La binarisation du mode de prédiction intra prédiction mode (intra-chromatique) est basée sur la table de consultation présentée dans le tableau (2.6) [17].

Tableau 2.6 : Binarisation pour le mode Intra Prédiction [30].

Valeur de intra_chroma_prédiction_mode	**Bin string**
4	0
0	100
1	101
2	110
3	111

i) Part Mode (Mode Pièce)

La binarisation du part mode dépend de nombreux facteurs. Tous les paramètres sont considérés comme des entrées de ce bloc pour la binarisation (un emplacement luma (xCb, yCb),spécifiant l'échantillon supérieur gauche du bloc de codage luma actuel par rapport à l'échantillon luma supérieur gauche de l'image actuelle et une variable log2CbSize spécifiant la taille du bloc de codage luma actuel). Par conséquent, il est prévu de déterminer ces valeurs à partir du codec. Il est conçu dans Simulink en exploitant les tables de consultation qui sont décrites dans les documents de référence HEVC [17].

La binarisation pour l'élément syntaxique part_mode est spécifiée dans le Tableau (2.7) en fonction des valeurs de Cu_Pred_Mode [xCb] [yCb] et log2CbSize.

Tableau 2.7 : Binarisation pour part mode [30].

CuPredMode [xCb] [yCb]	**Mode participation**	**Part Mode**	**Bin string**			
			Log2CbSize > MinCbLog2SizeY		Log2CbSize = = MinCbLog2SizeY	
			! amp_enabled_flag	amp_enabled_flag	log2CbSize = = 3	Log2CbSize > 3
Mode_Intra	0	PART_2Nx2N	-	-	1	1
	1	PART_NxN	-	-	0	0
	0	PART_2Nx2N	1	1	1	1

Mode_Inter	1	PART_2NxN	01	011	01	01
	2	PART_Nx2N	00	001	00	001
	3	PART_NxN	-	-	-	000
	4	PART_2NxnU	-	0100	-	-
	5	PART_2NxnD	-	0101	-	-
	6	PART_nLx2N	-	0000	-	-
	7	PART_nRx2N	-	0001	-	-

Pour la partie dé-binarisation l'opération inverse du processus de binarisation est appliquée au niveau de codage. Contrairement qu'au processus du codage la dé-binarisation est placée à la sortie de décodeur CABAC. Elle consiste à reconstruire les éléments syntaxiques à partir des *bins* obtenus à la sortie du décodeur en se basant sur les points suivants :

- Schémas de binarisation employés à l'encodage ;
- Type de l'élément syntaxique décodé ;
- Ordre de décodage [8].

2.7 Conclusion

Dans ce chapitre, les différents modules du système de communication et la théorie d'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 CABAC dans la norme HEVC. Nous avons présenté la binarisation, la dé-binarisation, ainsi que les éléments syntaxiques utilisés au niveau de transformation. Nous avons terminé le chapitre par une représentation détaillée sur les codes binaires utilisés dans la norme HEVC.

Chapitre 03

Chapitre 03

Programmation des codes de Binarisation sous MATLAB

3.1 Introduction

La norme HEVC définit dix méthodes de binarisation (TU, TRU, TRK, FL, EGK, CALR, QP Delta, Inter Prédiction Mode, Intra Prédiction Mode, Part Mode), certaines étant composées de la combinaison de deux d'entre elles par un mode préfixe-suffixe. L'objectif principal du binarisateur est de générer une représentation binaire plus petite pour les éléments qui se produisent à un taux d'occurrence plus élevé. Nous commencerons ce chapitre par les outils logiciels utilisés dans notre Travail (Xilinx ISE 14.5, Model Sim et le Matlab). Ensuite, nous présenterons les organigrammes des codes de binarisations avec leurs simulations sous Matlab. Il est à signaler que le but de ce chapitre est de maitriser les algorithmes de binarisations 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 binarisation /débinarisation de la norme de compression vidéo HEVC sur une plateforme FPGA *(Field Programmable Gate Array).* Nous avons utilisé le logiciel ISE pour la description de notre algorithme, le Model Sim pour la simulation et la validation de l'architecture proposée. La consommation de l'énergie est évaluée en utilisant l'analyseur XPower de l'ISE [8]. La Figure (3.1) illustre la vue d'ensemble des outils utilisés pour notre implémentation.

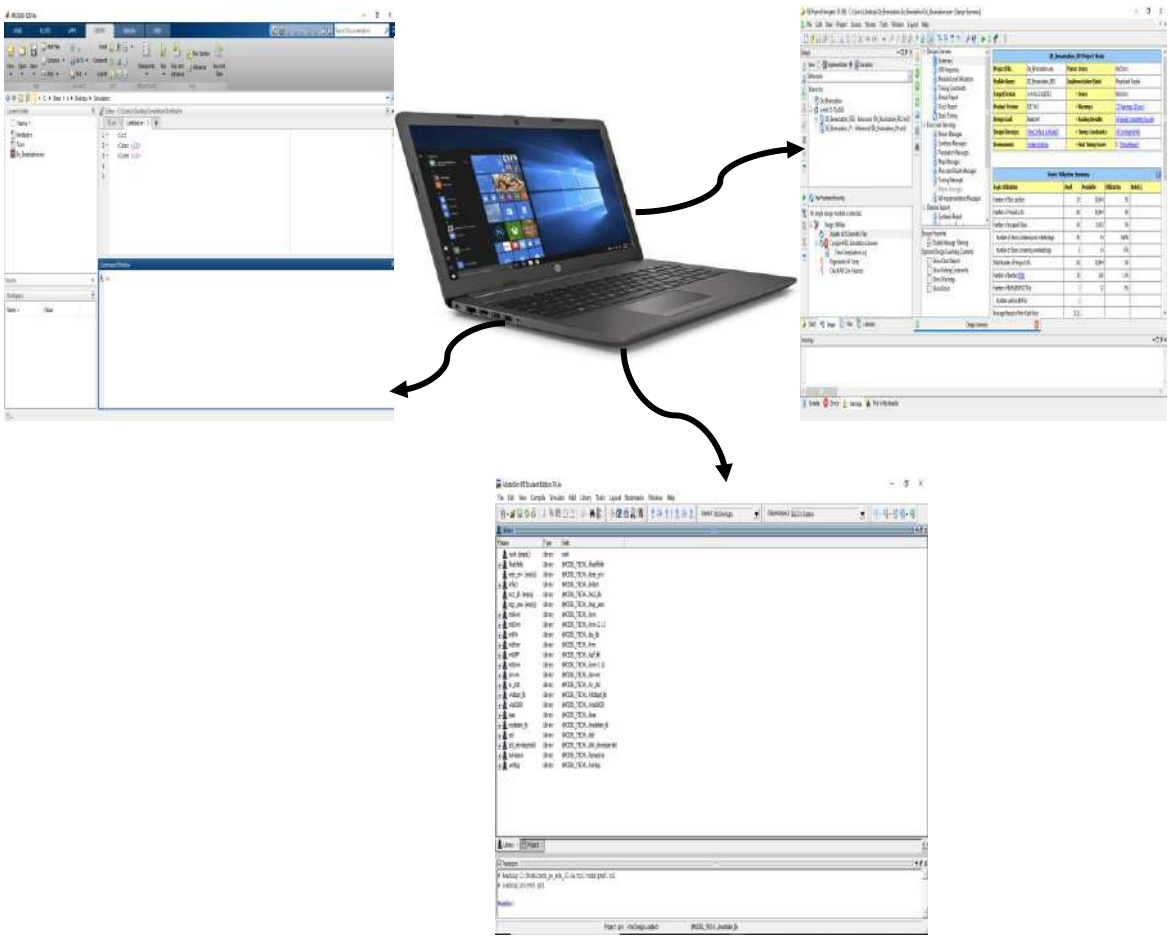


Figure 3.1 : Outils matériels et logiciels utilisés dans notre projet.

3.2.1 Xilinx ISE 14.5

ISE (*Integrated Software Environment*) est un environnement de développement intégré développé et commercialisé par la société Xilinx [7] [20] 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*) de la société Xilinx.

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.

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity my_and is
33      Port ( a : in  STD_LOGIC;
34             b : in  STD_LOGIC;
35             c : in  STD_LOGIC;
36             s : out  STD_LOGIC);
37  end my_and;
38
39  architecture Behavioral of my_and is
40
41  begin
42  s <= a AND b AND c;
43
44  end Behavioral;
45
46
```

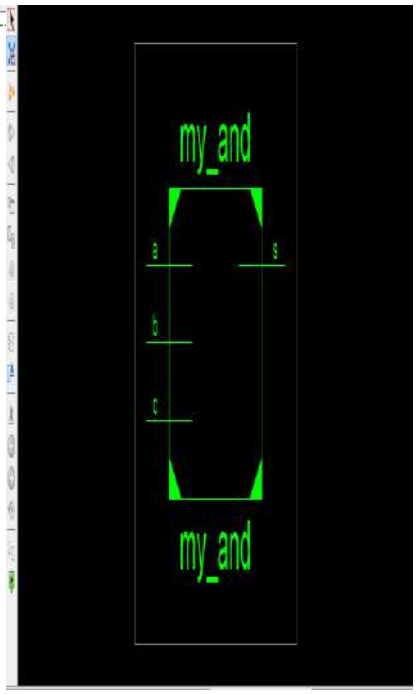


Figure 3.2 : Entité et Architecture d'un composant logique sur ISE.

3.2.2 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 [31].

3.2.3 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) [32].

3.3 Algorithme de binarisation /dé-binarisation

Dans cette partie on va présenter les organigrammes de fonctionnement des codes de binarisation cités au précédent avec leurs implémentations sous Matlab.

3.3.1 Organigramme

a) Code TU

Le code Unaire (*U*) convertit un nombre entier non signé en une séquence de '1' concaténés avec un seul '0' à la fin, avec autant de '1' que la valeur originale du nombre. Le code Tronqué Unaire (*TU*) fonctionne de la même manière que le code Unaire.

La figure suivante montre l'organigramme du code de binarisation TU qui est basé sur la variable 'i' (nombre de cas traité).

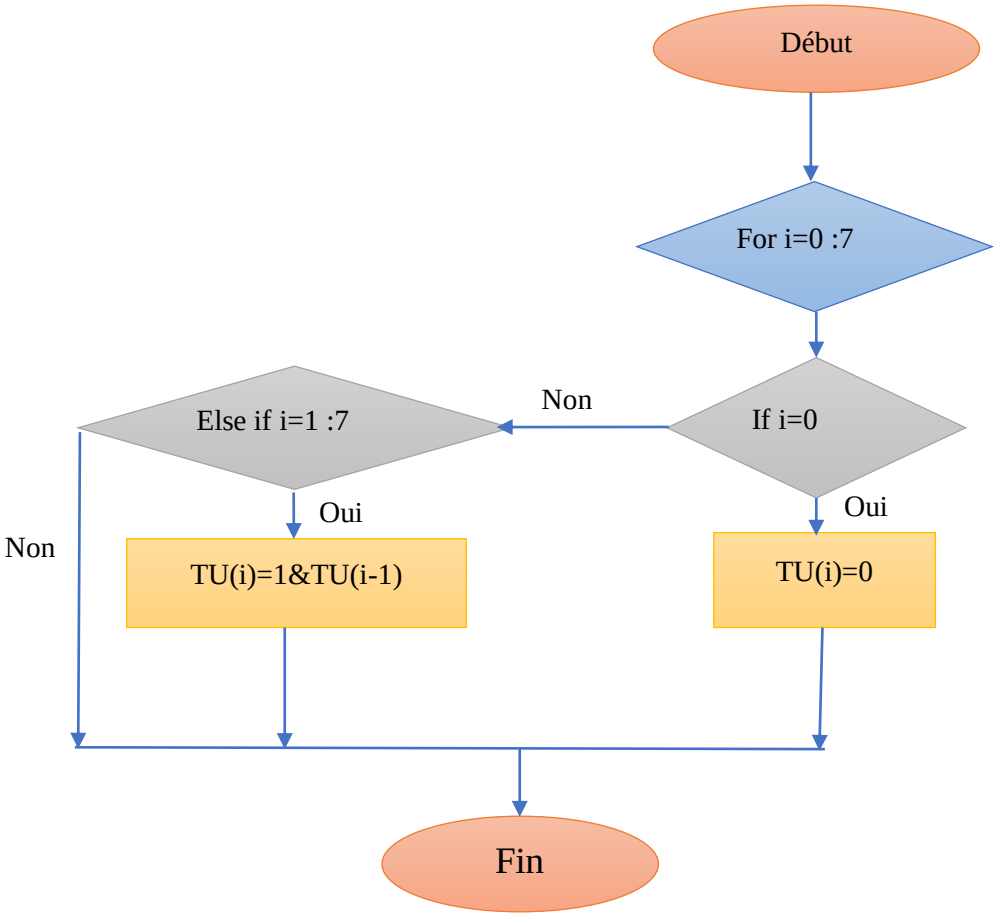


Figure 3.3 : Organigramme du code Tronqué Unaire (*TU*).

b) Code TRU

Le code tronqué unaire (*TRU*) est une méthode de codage plus simple utilisée dans le standard HEVC. Pour plus de détail (voir le chapitre 02).

La figure suivante montre l'organigramme du code TRU qui est basé sur la variable i (nombre de cas traité).

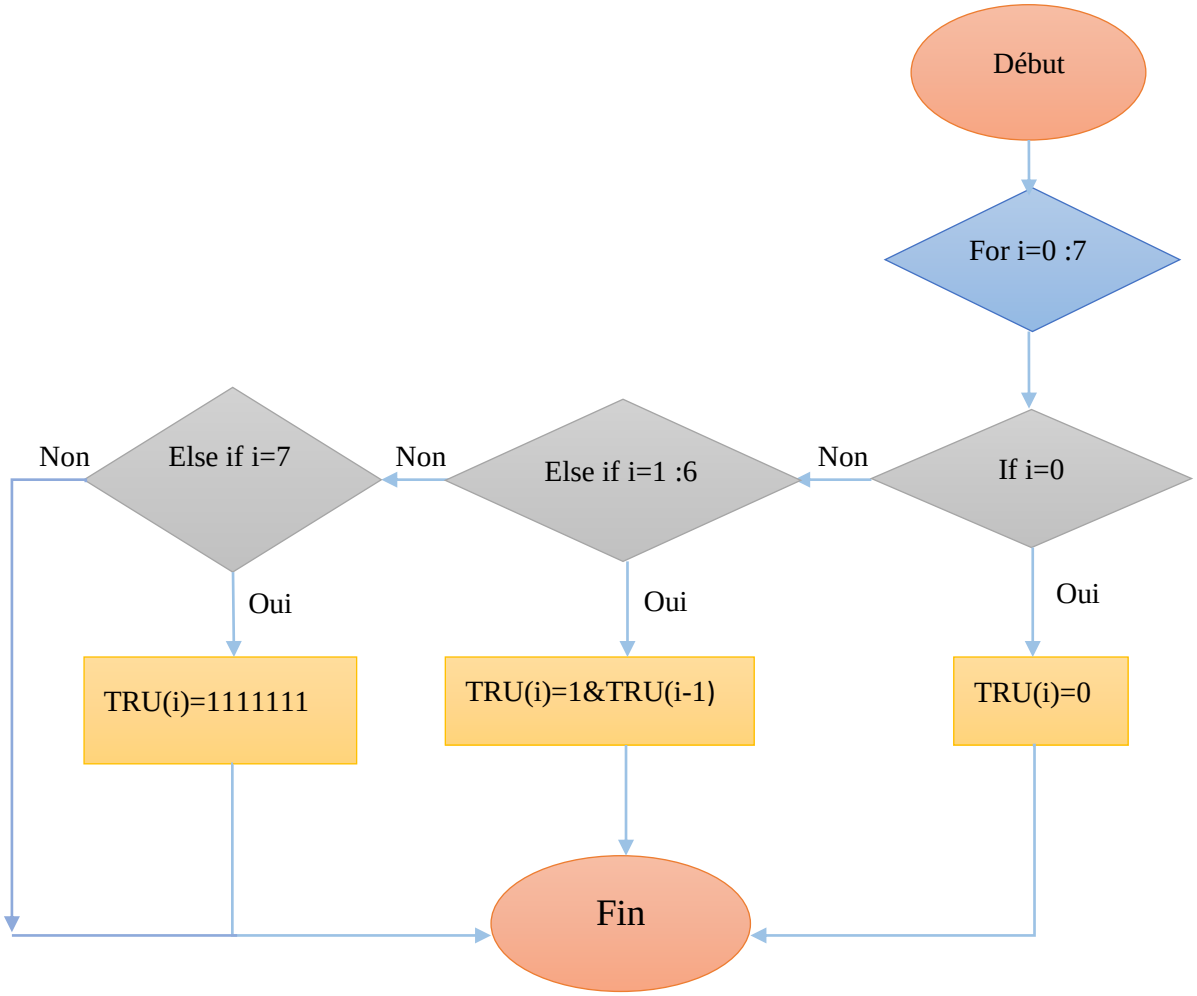


Figure 3.4 : Organigramme du code Tronqué Race Unaire (*TRU*).

c) Code FL

Le code Fixed_Length (*FL*) ne code pas la valeur d'élément syntaxique (*SE*) mais elle limite simplement le nombre des symboles binaires générés par la même variable *cMax* de la taille maximale du code binaire pour ce processus de binarisation.

La figure suivante représente l'organigramme du code FL qui utilise le i (nombre de cas traité) comme variable d'entrée.

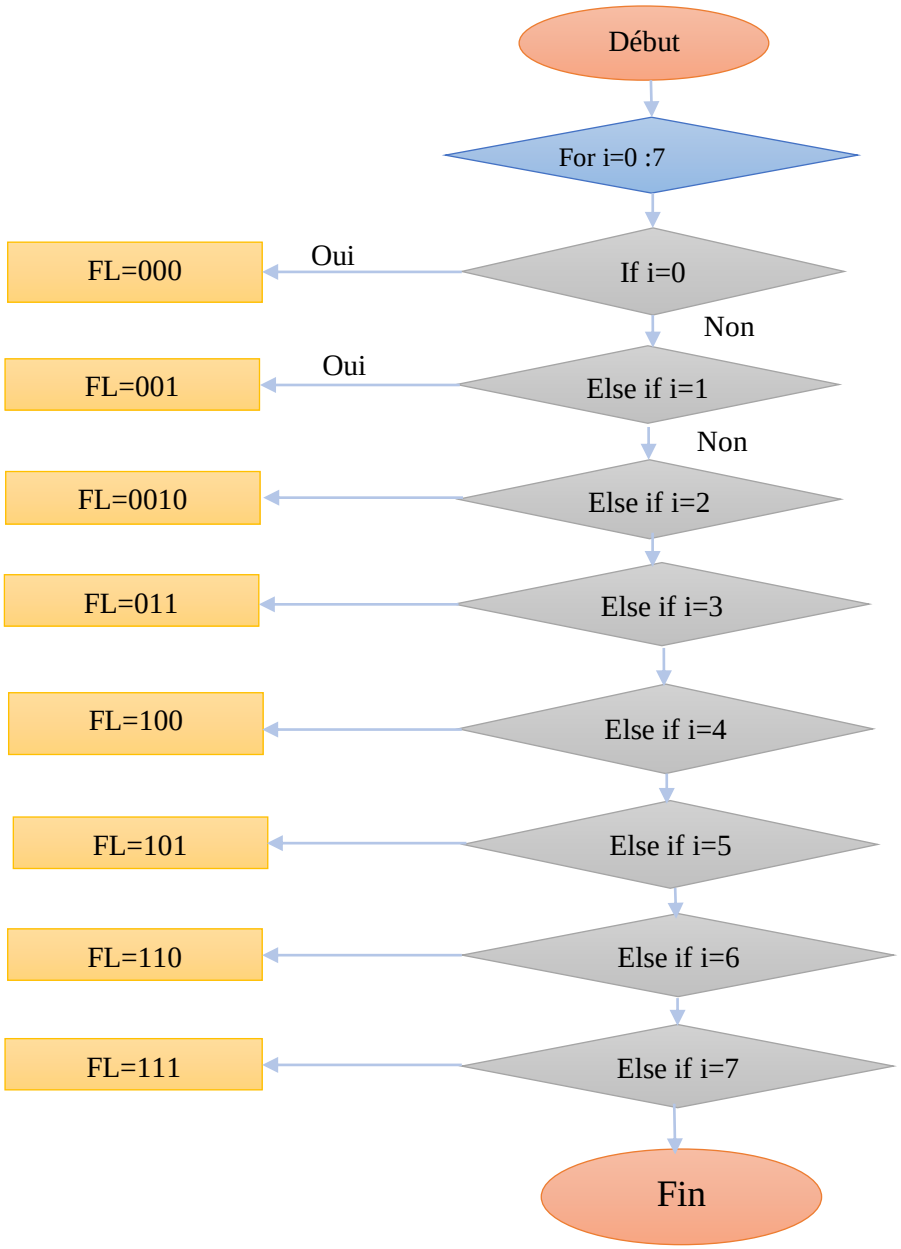
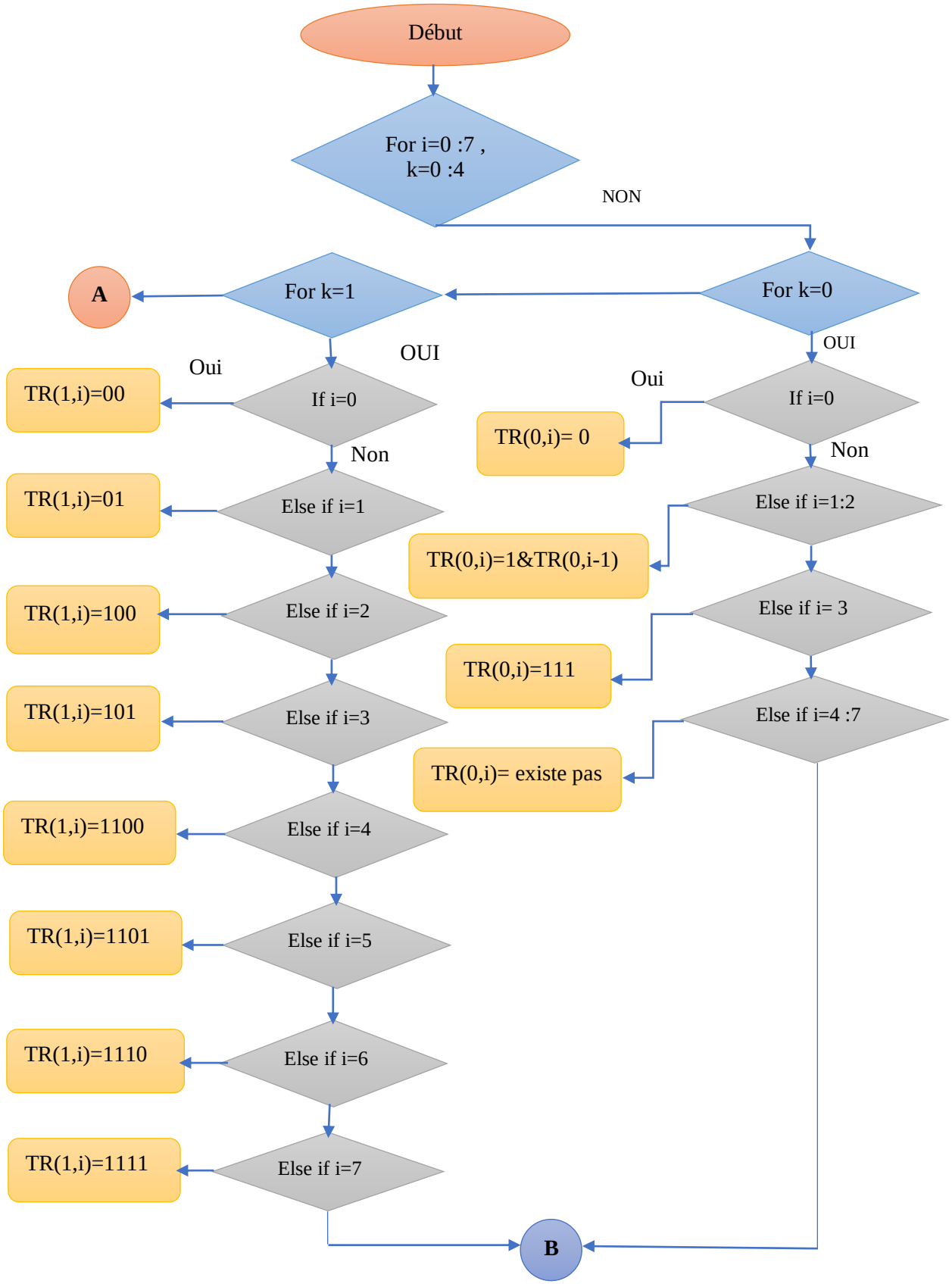


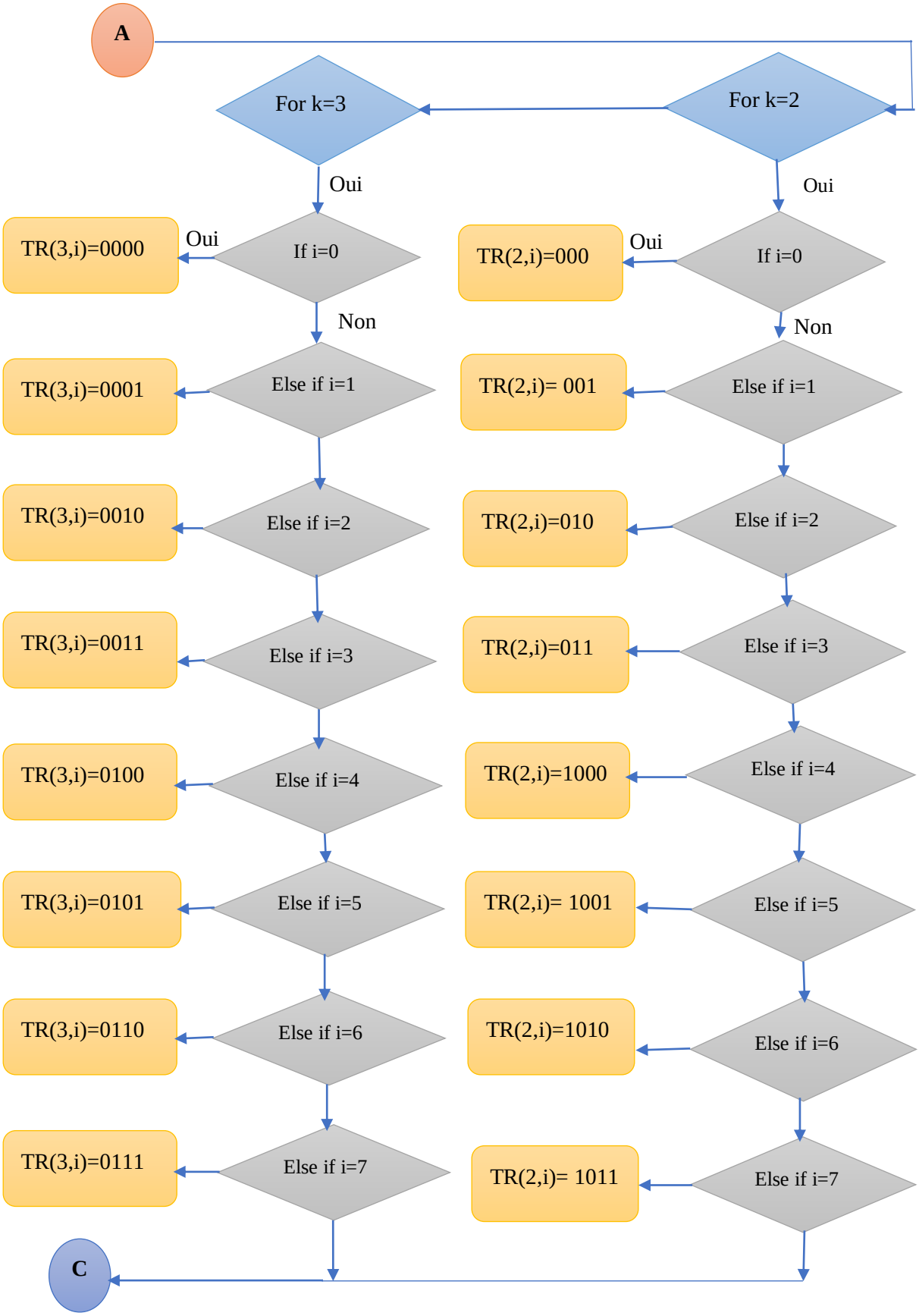
Figure 3.5 : Organigramme du code Fixed_Length (*FL*).

d) Code TRK

Le code binarisation Rice tronqué (*TR*) est composé de deux parties, à savoir le préfixe et le suffixe (lorsqu'il est présent) dans la chaine binaire. La partie préfixe appelle la binarisation TU. Par contre, la partie suffixe invoque la binarisation FL.

La figure suivante illustre l'organigramme du code TRK qui est basé sur deux variables i (nombre de cas traité) et K (variable spécifique au code).


Début
For i=0 :7 ,
k=0 :4
NON
For k=0
For k=1
A
OUI
OUI
If i=0
If i=0
Oui
Oui
TR(1,i)=00
TR(0,i)= 0
Non
Non
Else if i=1
Else if i=1:2
TR(1,i)=01
TR(0,i)=1&TR(0,i-1)
Else if i=2
Else if i= 3
TR(1,i)=100
TR(0,i)=111
Else if i=3
Else if i=4 :7
TR(1,i)=101
TR(0,i)= existe pas
Else if i=4
TR(1,i)=1100
Else if i=5
TR(1,i)=1101
Else if i=6
TR(1,i)=1110
Else if i=7
TR(1,i)=1111
B


A
For k=3
For k=2
Oui
Oui
TR(3,i)=0000
Oui
If i=0
TR(2,i)=000
Oui
If i=0
Non
Non
TR(3,i)=0001
Else if i=1
TR(2,i)= 001
Else if i=1
TR(3,i)=0010
Else if i=2
TR(2,i)=010
Else if i=2
TR(3,i)=0011
Else if i=3
TR(2,i)=011
Else if i=3
TR(3,i)=0100
Else if i=4
TR(2,i)=1000
Else if i=4
TR(3,i)=0101
Else if i=5
TR(2,i)= 1001
Else if i=5
TR(3,i)=0110
Else if i=6
TR(2,i)=1010
Else if i=6
TR(3,i)=0111
Else if i=7
TR(2,i)= 1011
Else if i=7
C

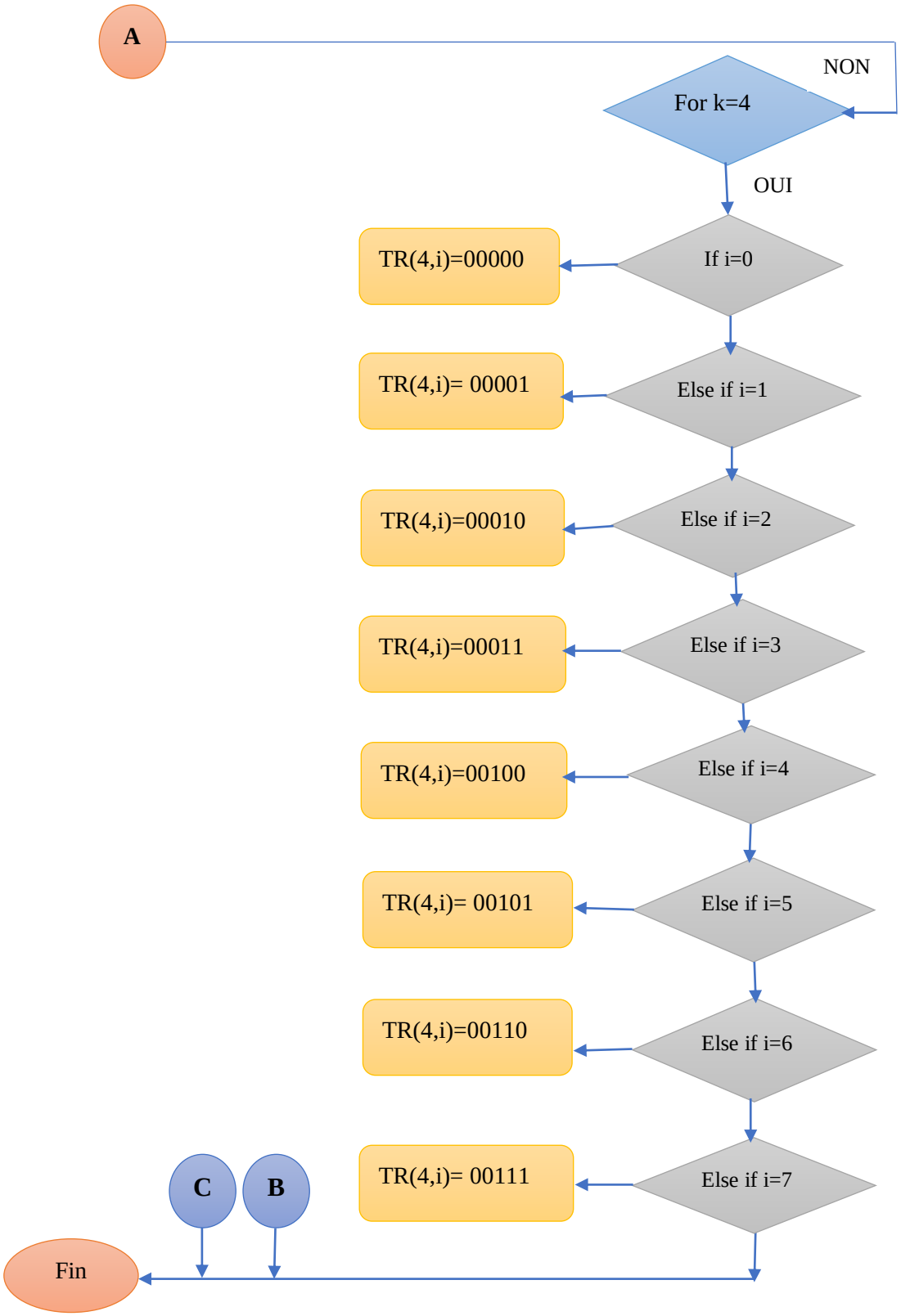
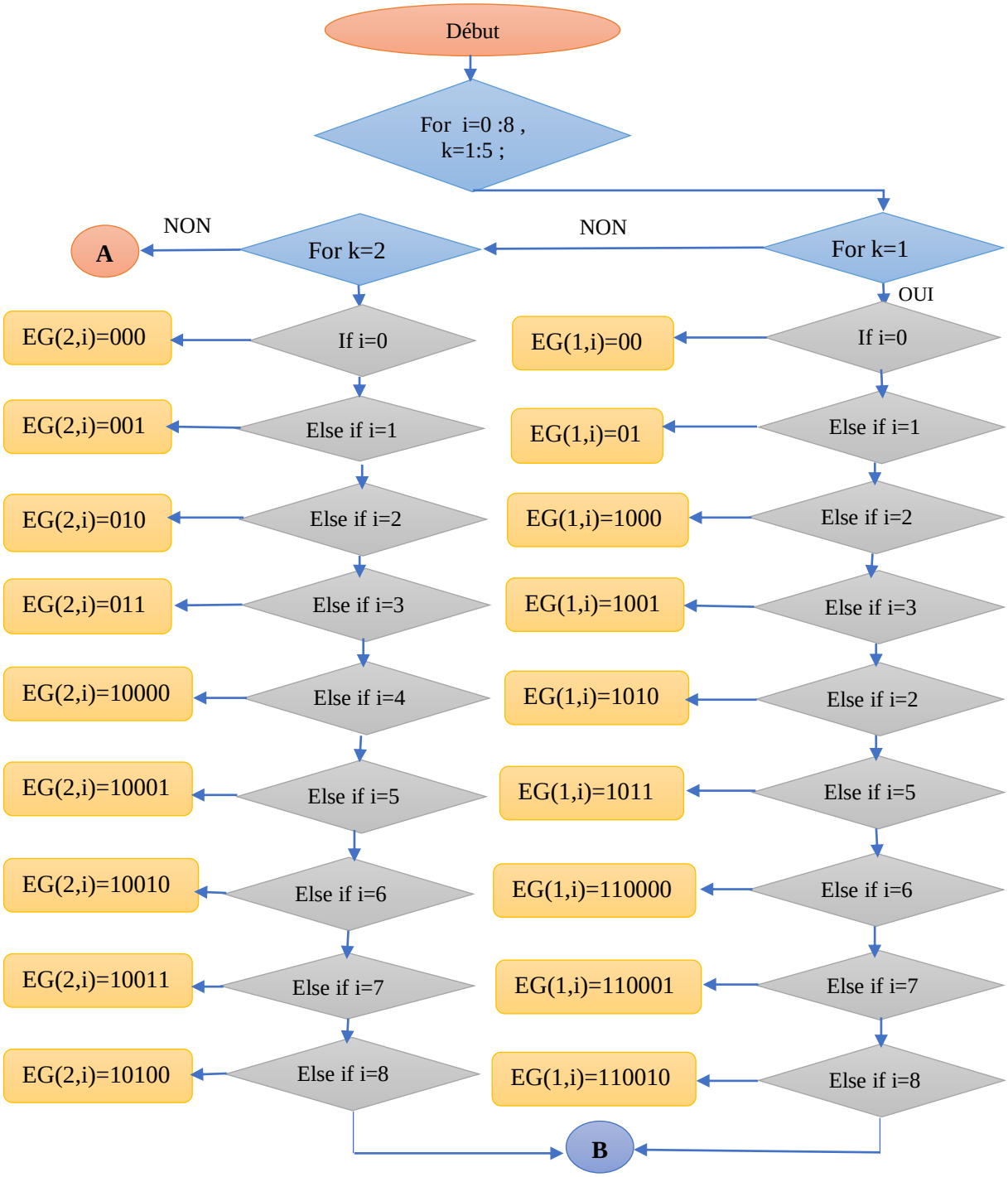


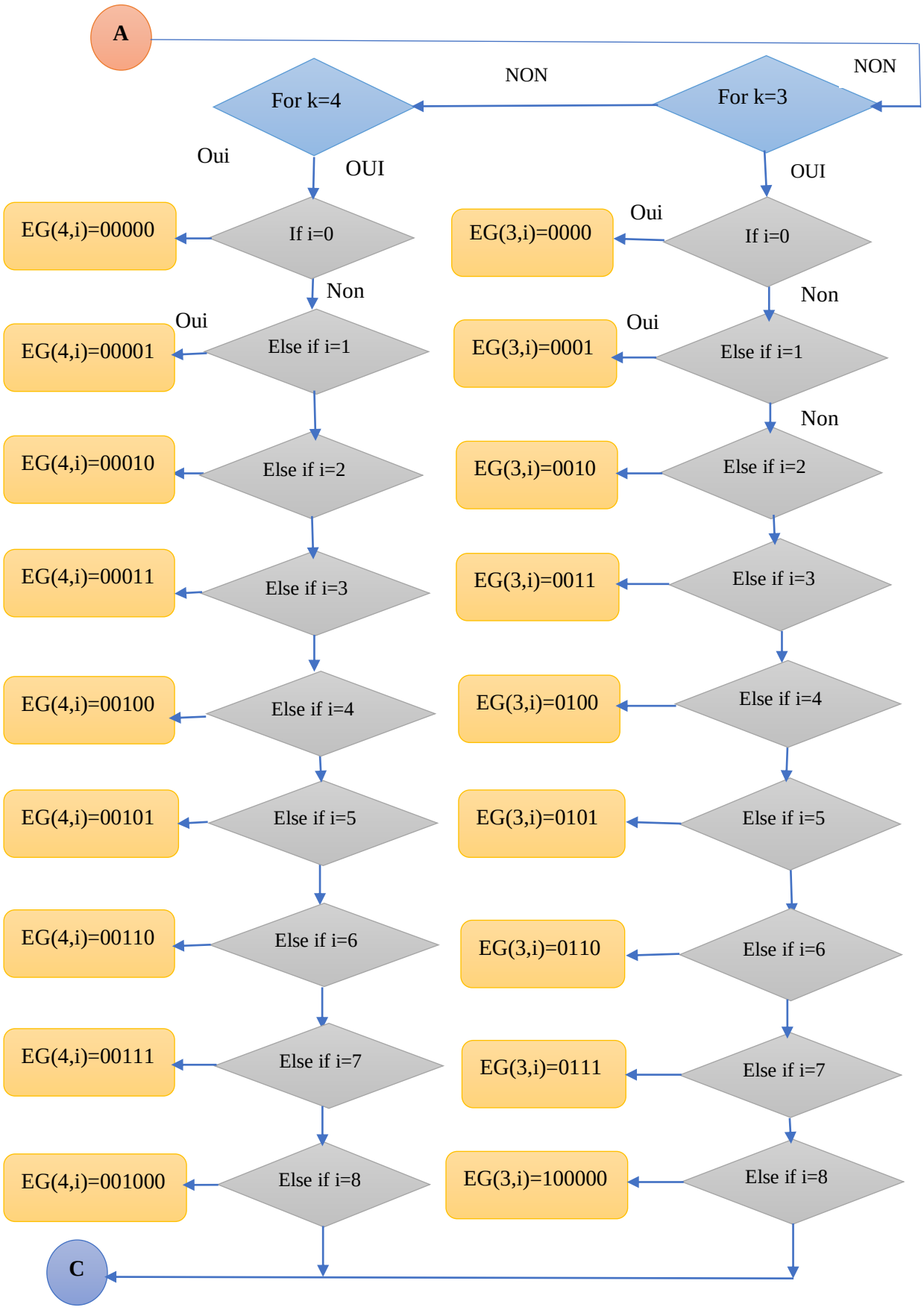
Figure 3.6 : Organigramme du code Tronqué Rice K-ème (*TRK*).

e) Code EGK

Le code de binarisation Exp-Golomb (*EGk*) est également réalisée par une concaténation du préfixe avec le suffixe. La variable 'K' détermine les bits initiaux du suffixe.

La figure ci-dessous illustre l'organigramme du code EGK qui est appel en entrée deux variables i (nombre de cas traité) et K (variable spécifique au code).




A
For k=4
NON
For k=3
NON
Oui
OUI
OUI
If i=0
EG(4,i)=00000
Oui
EG(3,i)=0000
If i=0
Non
Non
Oui
Else if i=1
EG(4,i)=00001
Oui
EG(3,i)=0001
Else if i=1
Non
Else if i=2
EG(4,i)=00010
EG(3,i)=0010
Else if i=2
Else if i=3
EG(4,i)=00011
EG(3,i)=0011
Else if i=3
Else if i=4
EG(4,i)=00100
EG(3,i)=0100
Else if i=4
Else if i=5
EG(4,i)=00101
EG(3,i)=0101
Else if i=5
Else if i=6
EG(4,i)=00110
EG(3,i)=0110
Else if i=6
Else if i=7
EG(4,i)=00111
EG(3,i)=0111
Else if i=7
Else if i=8
EG(4,i)=001000
EG(3,i)=100000
Else if i=8
C

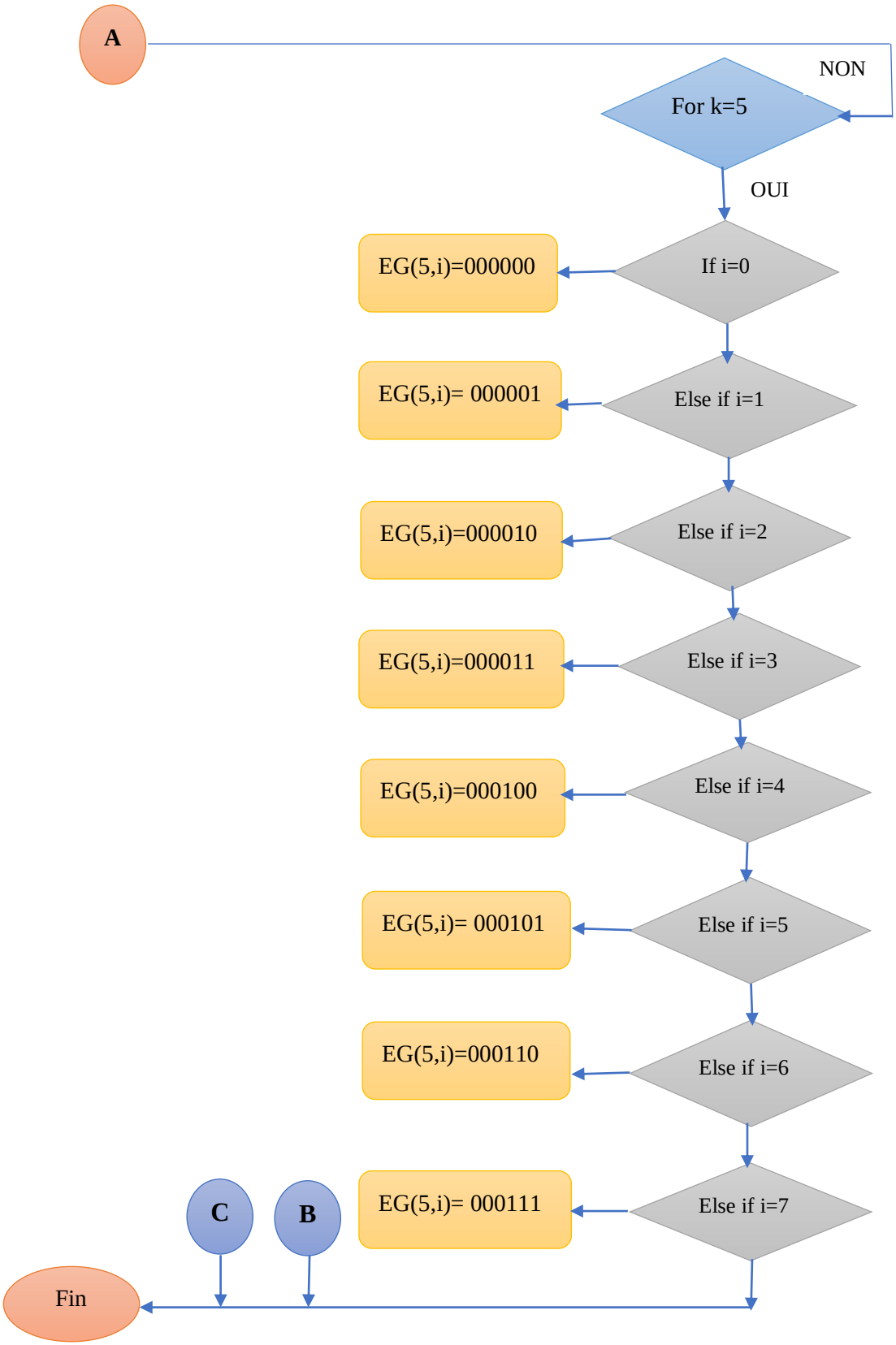


Figure 3.7 : Organigramme du code Exp-Golomb (*EGK*).

f) Code CALR

Le CALR *(Coefficient Absolute Level Remaining*) est utilisé pour binariser les résiduel des coefficients obtenus après le processus de quantification. La code CALR est essentiellement basé sur une concaténation des deux codes Rice tronqué TRK et code Golomb exponentiel EGK. La figure suivante illustre l'organigramme du code CALR.

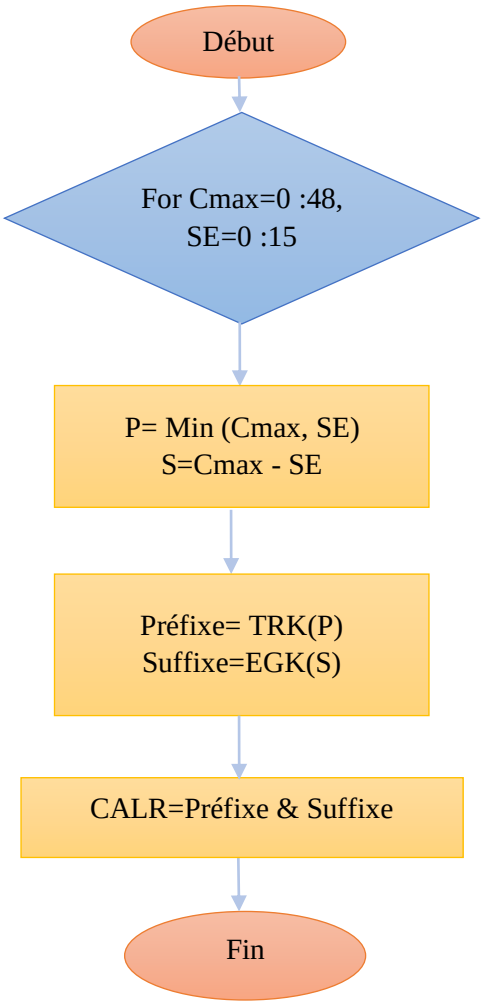


Figure 3.8 : Organigramme du code CALR.

g) Code QP Delta

Le QP Delta invoque une combinaison de code TU et de code EGK quand K=0. La chaîne binaire est une concaténation de deux chaînes binaires préfixe et suffixe.

La figure suivante illustre l'organigramme du code de binarisation QP Delta.

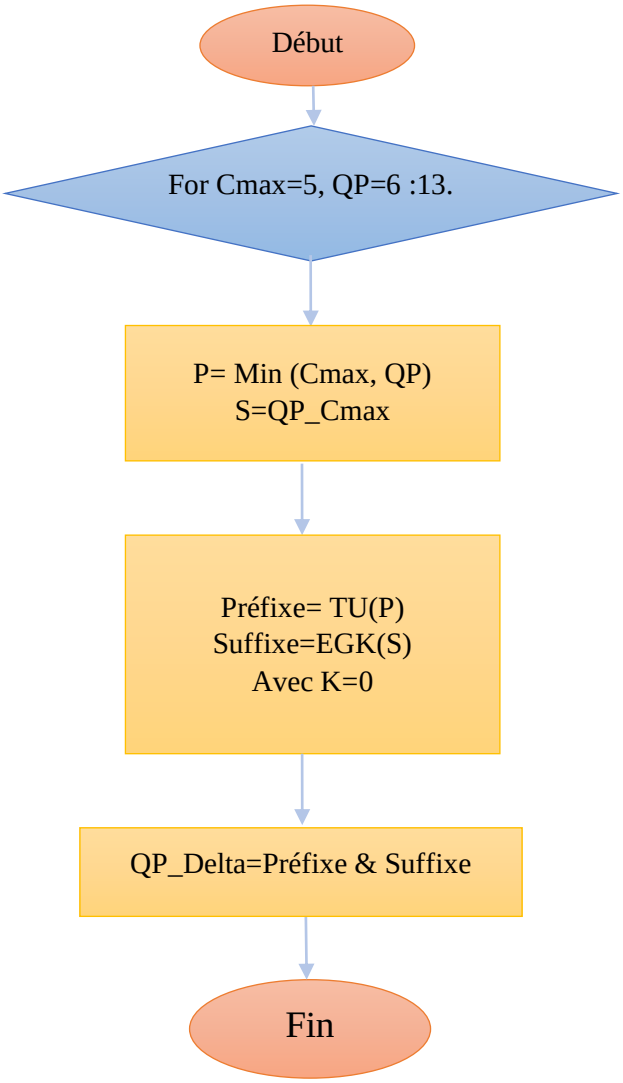
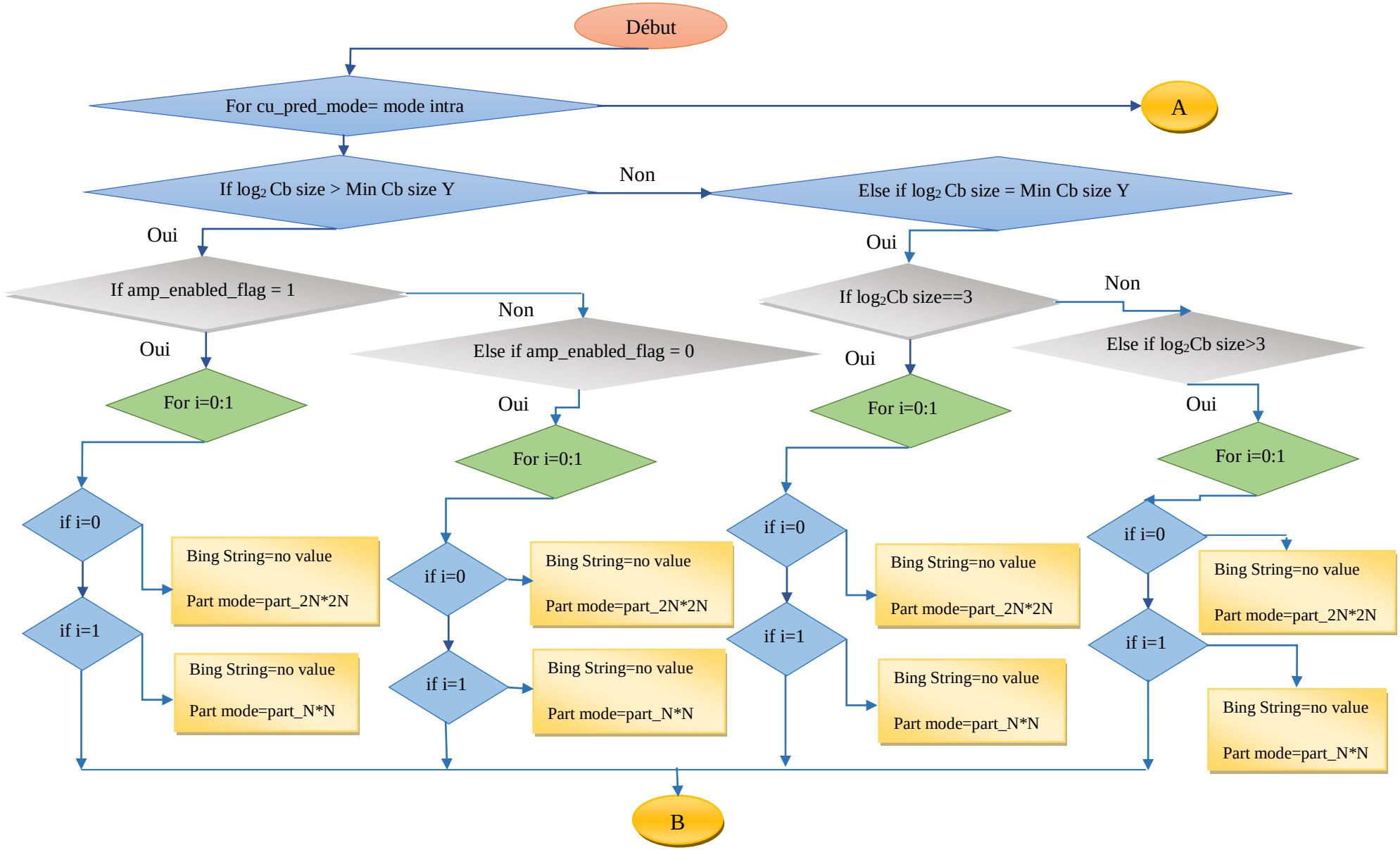


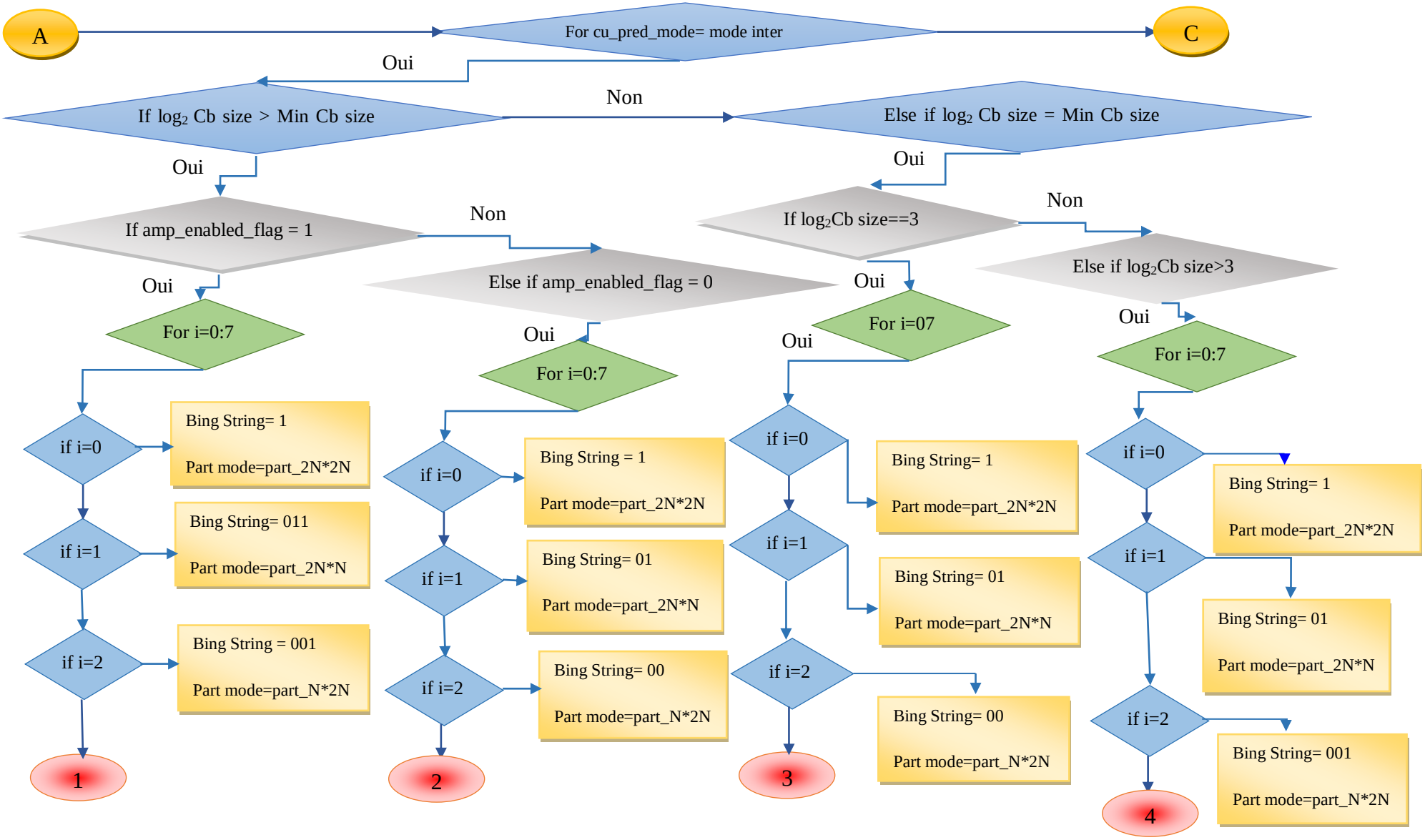
Figure 3.9 : Organigramme du code QP Delta.

h) Part Mode

La binarisation du part mode dépend de nombreux facteurs. Tous les paramètres (cu_pred_mode, amp_enabled_flag, log2 Cb size, Cb size Y, i) sont considérés comme des entrées de ce bloc pour la binarisation.

La figure suivante illustre l'organigramme du code Part Mode.


Début
For cu_pred_mode= mode intra
A
If log2 Cb size > Min Cb size Y
Non
Else if log2 Cb size = Min Cb size Y
Oui
Oui
If amp_enabled_flag = 1
Non
Else if amp_enabled_flag = 0
If log2Cb size==3
Non
Else if log2Cb size>3
Oui
Oui
Oui
For i=0:1
For i=0:1
For i=0:1
For i=0:1
if i=0
if i=1
Bing String=no value
Part mode=part_2N*2N
Bing String=no value
Part mode=part_N*N
if i=0
if i=1
Bing String=no value
Part mode=part_2N*2N
Bing String=no value
Part mode=part_N*N
if i=0
if i=1
Bing String=no value
Part mode=part_2N*2N
Bing String=no value
Part mode=part_N*N
if i=0
if i=1
Bing String=no value
Part mode=part_2N*2N
Bing String=no value
Part mode=part_N*N
B


A
For cu_pred_mode= mode inter
C
Oui
If log2 Cb size > Min Cb size
Non
Else if log2 Cb size = Min Cb size
Oui
If amp_enabled_flag = 1
Non
Else if amp_enabled_flag = 0
Oui
If log2Cb size==3
Non
Else if log2Cb size>3
Oui
For i=0:7
Oui
For i=0:7
Oui
For i=07
Oui
For i=0:7
if i=0
Bing String= 1
Part mode=part_2N*2N
if i=1
Bing String= 011
Part mode=part_2N*N
if i=2
Bing String = 001
Part mode=part_N*2N
1
if i=0
Bing String = 1
Part mode=part_2N*2N
if i=1
Bing String= 01
Part mode=part_2N*N
if i=2
Bing String= 00
Part mode=part_N*2N
2
if i=0
Bing String= 1
Part mode=part_2N*2N
if i=1
Bing String= 01
Part mode=part_2N*N
if i=2
Bing String= 00
Part mode=part_N*2N
3
if i=0
Bing String= 1
Part mode=part_2N*2N
if i=1
Bing String= 01
Part mode=part_2N*N
if i=2
Bing String= 001
Part mode=part_N*2N
4

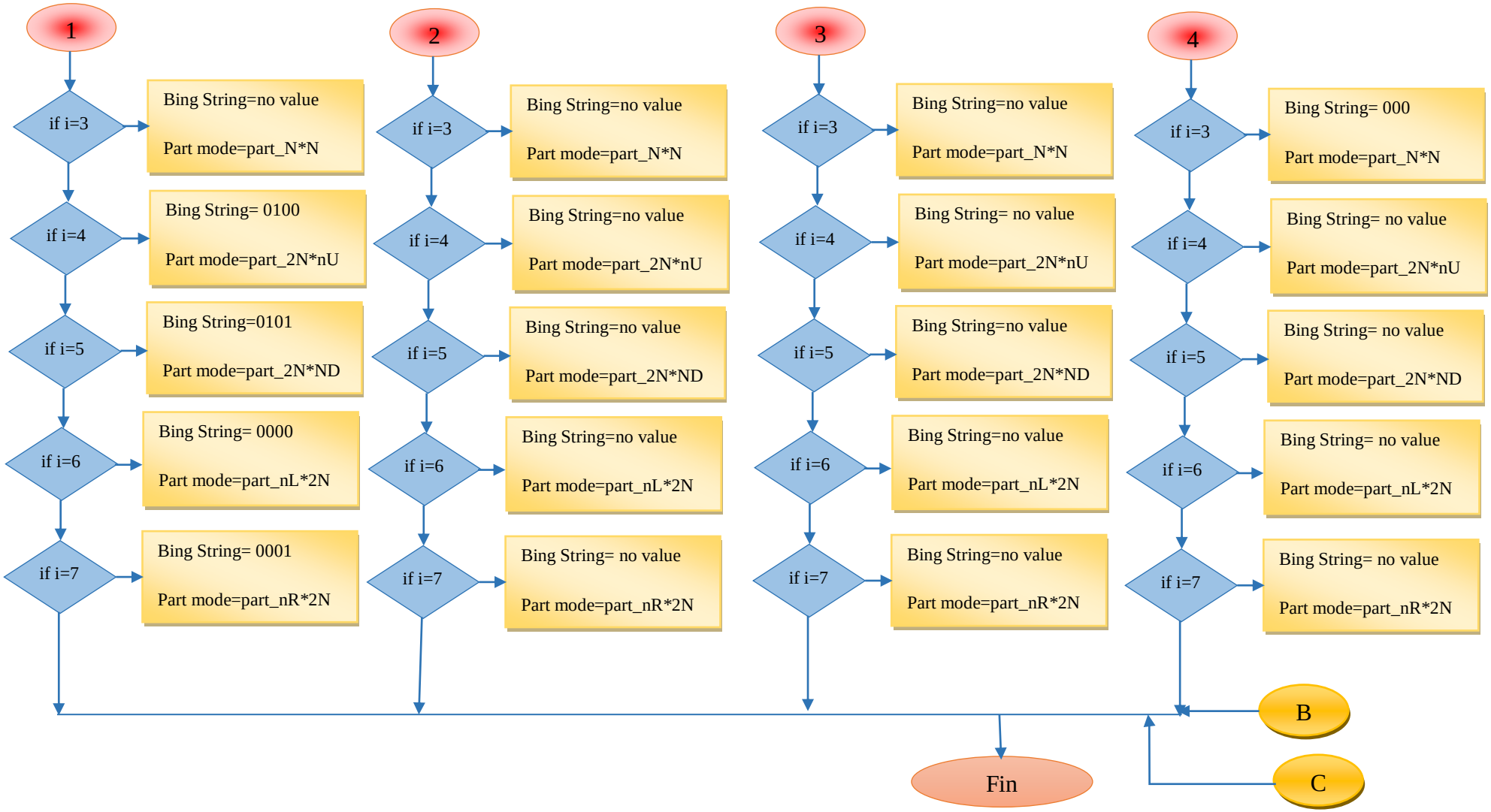


Figure 3.10 : Organigramme du code Part Mode.

i) Intra Prédiction Mode

Ce code est un module de format personnalisé, il est basé sur un tableau de consultation. On a déjà parler en détails sur ce code dans le chapitre précédent.

La figure (3.19) illustre l'organigramme du code Intra Prédiction Mode.

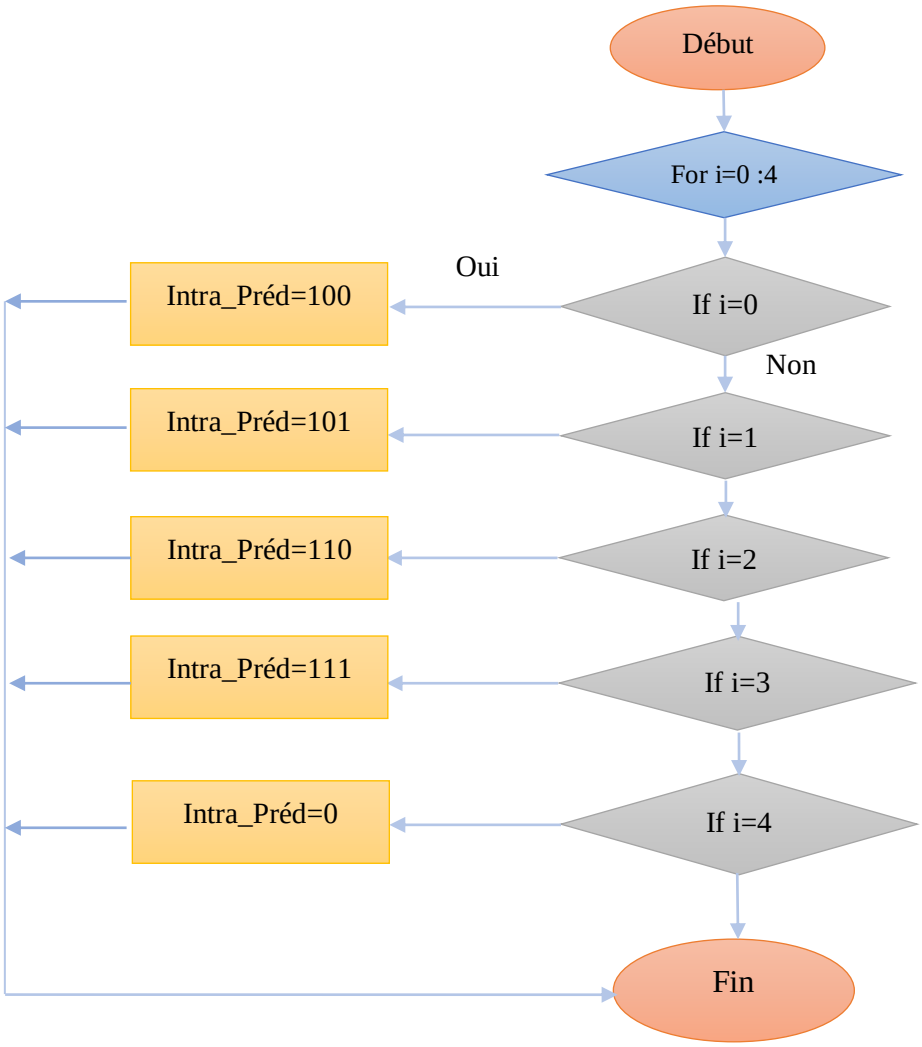


Figure 3.11 : Organigramme du code Intra prédiction mode.

j) Inter Prédiction Mode

Ce code est un module de format personnalisé, il est basé sur un tableau de consultation. On a déjà parler en détails sur ce code dans le chapitre précédent.

La figure (3.21) illustre l'organigramme du code Inter prédiction mode.

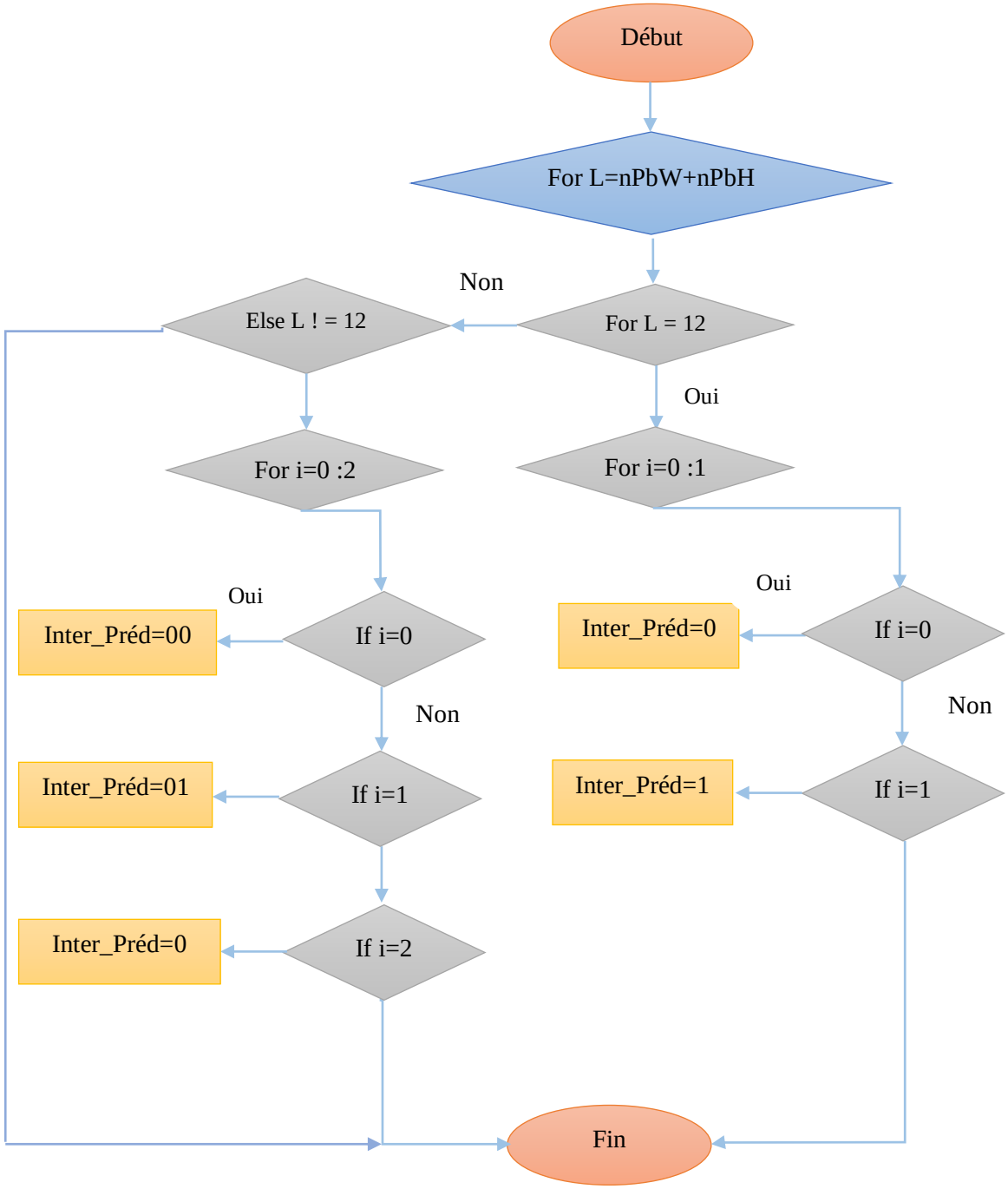


Figure 3.12 : Organigramme du code Inter Prédiction Mode.

3.3.2 Programme sous Matlab

Les figures 3.13 a, b, c, d, e, f, g respectivement montrent la simulation des codes de binarisation format unique (5 codes), format combiné (2 codes) et format personnalisé (3 codes) sous Matlab.

```
TU =

     0     10    110   1110  11110 111110 1111110 11111110

>> FL

FL =

    '000'    '001'    '010'    '011'    '100'    '101'    '110'    '111'

>> TRU

TRU =

     0     10    110   1110  11110 111110 1111110 1111111

>> TR1

TR1 =

    '00'    '01'    '100'    '101'    '1100'    '1101'    '1110'    '1111'
```

(a)

```
TR2 =

    '00'    '01'    '100'    '101'    '1100'    '1101'    '1110'    '1111'

>> TR3

TR3 =

    '000'    '001'    '010'    '011'    '1000'    '1001'    '1010'    '1011'

>> TR4

TR4 =

    '0000'    '0001'    '0010'    '0011'    '0100'    '0101'    '0110'    '0111'

>> TR5

TR5 =

    '00000'    '00001'    '00010'    '00011'    '00100'    '00101'    '00110'    '00111'
```

(b)

```
EG1 =

    '00'    '01'    '1000'    '1001'    '1010'    '1011'    '110000'    '110001'

>> EG2

EG2 =

    '000'    '001'    '010'    '011'    '10000'    '10001'    '10010'    '10011'

>> EG3

EG3 =

    '0000'    '0001'    '0010'    '0011'    '0100'    '0101'    '0110'    '0111'

>> EG4

EG4 =

    '00000'    '00001'    '00010'    '00011'    '00100'    '00101'    '00110'    '00111'
```

(c)

```
EG5 =

    '000000'    '000001'    '000010'    '000011'    '000100'    '000101'    '000110'    '000111'
```

```
>> CALR

CALR =

    '00110001'    '01110001'    '100110001'    '101110001'    '1100110001'    '1101110001'    '1110110001'    '1111110001'

QP_Delta =

    '1111000'    '1111001'    '111101000'    '111101001'    '111101010'    '111101011'    '11110110000'    '11110110001'

>>
```

(d)

```
>> Part

Part =

    '1'    '011'    '001'    'exite pas'    '0100'    '0101'    '0000'    '0001'

>> Intra_Pred

Intra_Pred =

    '100'    '101'    '110'    '111'    '0'

>> Inter_Pred

Inter_Pred =

    '00'    '01'    '1'
```

(e)

```
Part =

    '1'    '01'    '00'
```

(f)

```
Part =

    '1'    '01'    '001'    '000'
```

(g)

Figure 3.13 : Simulation sous Matlab des dix codes de binarisation (format unique, format combiné, format personnalisé).

3.4 Conclusion

Dans ce chapitre, les différents outils matériels et logiciels utilisés ont été présentés. Ensuite, nous avons aussi concevoir les organigrammes des codes de binarisation avec leurs implémentations sous Matlab. Ce passage nous a permis de maitriser les algorithmes des codes de binarisation et dé-binarisation utilisés dans la norme de compression vidéo HEVC avant de passer à la description matérielle en langage VHDL sous l'ISE. Pour cela, le chapitre suivant sera consacré à la conception et l'implémentation des codes de binarisation / dé-binarisation sur l'espace de développement ISE (FPGA).

Chapitre 04

Chapitre 04

Conception et Implémentation des codes de binarisation / dé-binarisation sur FPGA

4.1 Introduction

Ce dernier chapitre est réservé pour l'implémentation de l'algorithme de binarisation/ dé-binarisation CABAC HEVC sur FPGA. Pour cela, nous commençons ce chapitre par la présentation des éléments syntaxiques qui ont besoin d'être binarisés. Ensuite, les architectures proposées seront données en détails pour chaque partie de cet algorithme avec la partie contrôle (FSM) suivi par la 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 la comparaison et la discussion des résultats obtenus.

4.2 Préparation et récupération du flux binaire

- **Préparation du flux binaire**

Dans le CABAC HEVC au niveau de transformation il y'a des éléments syntaxiques qui ont besoin d'être binarisés avant de passer à la partie codage arithmétique. Les deux figures suivantes représentent préparation de flux binaire pour un bloc de transformation de taille 4x4 avant de passer à l'étape de binarisation et de codage.

Il est signalé que ce bloc de transformation de taille 4x4 mentionner sur la figure (4.1) utilisé par la commuté scientifique comme un bloc de référence. Le bloc sur la droite présente le balayage diagonal adopté par la norme de compression vidéo HEVC et le bloc sur la gauche donne le bloc de transformation de 4x4.

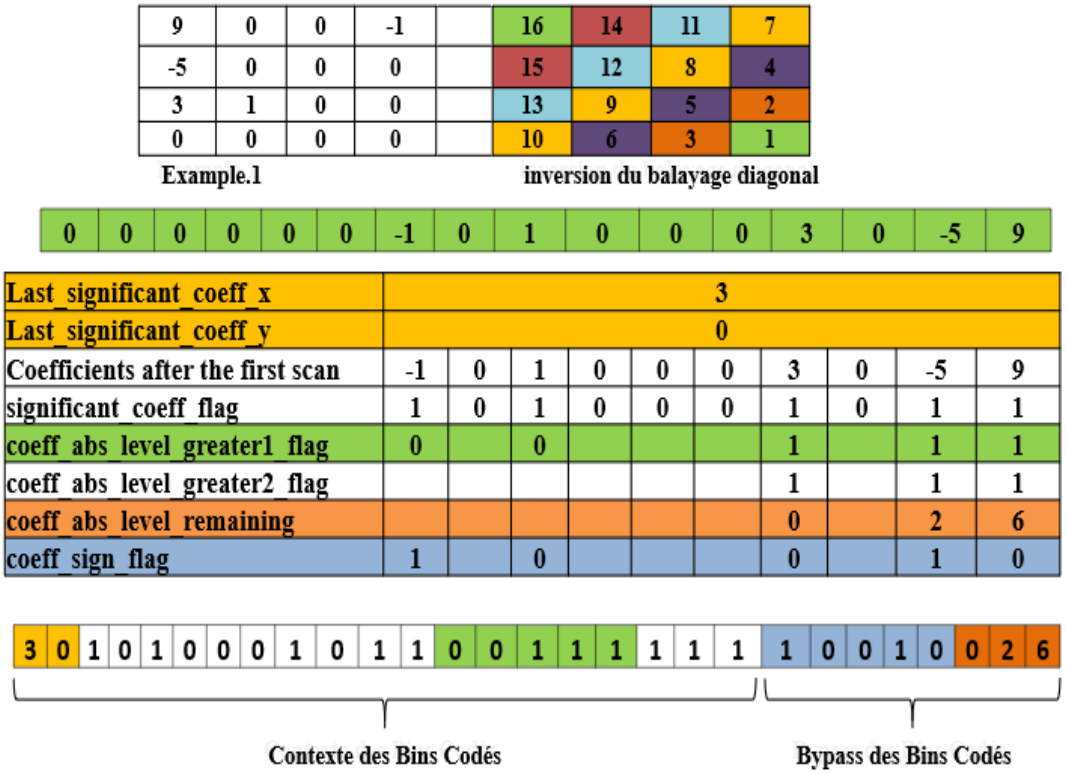


Figure 4.1 : Préparation du flux au niveau de transformation pour un bloc TU 4x4.

La deuxième figure (4.2) montre que les coefficients résiduels et la position (x,y) qui ont besoin d'être binariser.

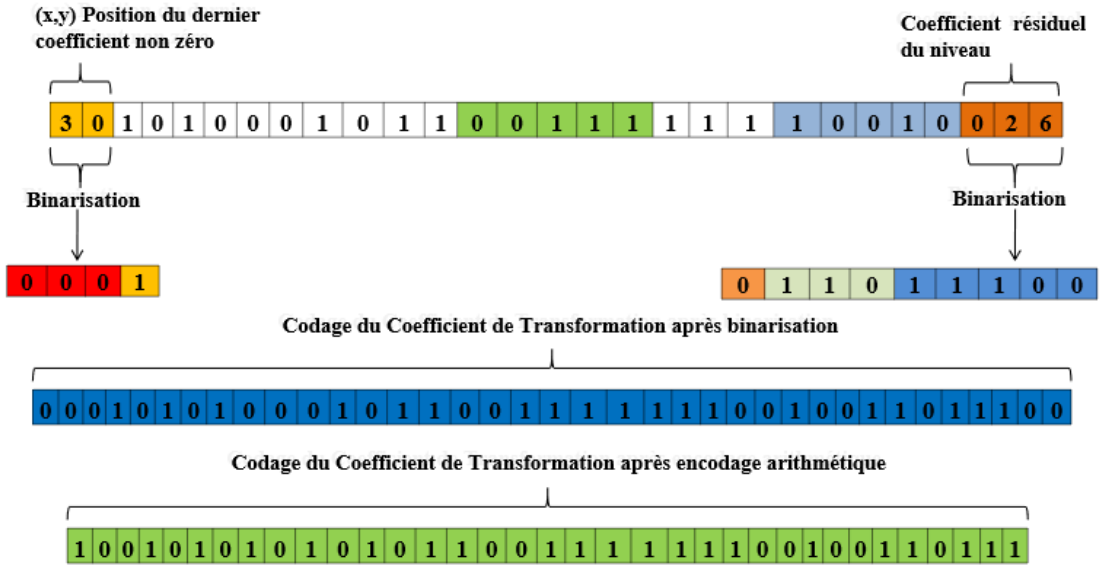


Figure 4.2 : Coefficients résiduels et la position (x, y).

- **Récupération du flux binaire**

Le schéma bloc suivant montre le décodage au niveau bloc de transformation. Toutes les étapes sont de récupération des données d'un bloc TU de 4x4 sont représentées sur la figure (4.3).

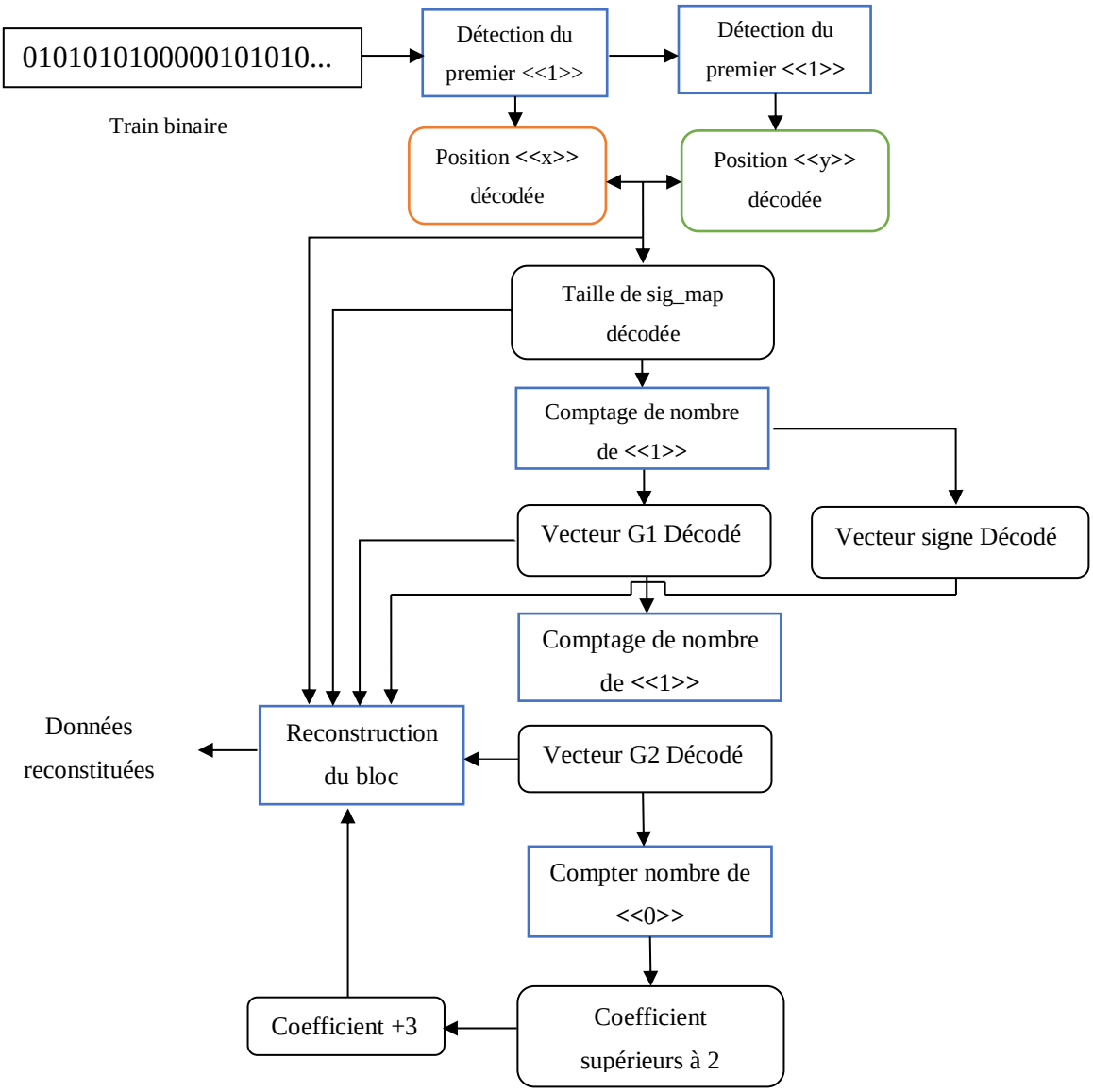


Figure 4.3 : Schéma bloc de décodage niveau bloc de transformation.

4.3 Architecture proposée de binarisation / dé-binarisation

L'architecture de binarisation / dé-binarisation contient dix codes de binarisation où chaque code est basé sur architecture différente. La figure suivante montre le HEVC CABAC binarisateur qui converti tous les éléments syntaxiques non binaire en des codes binaires adaptés au codeur CABAC.

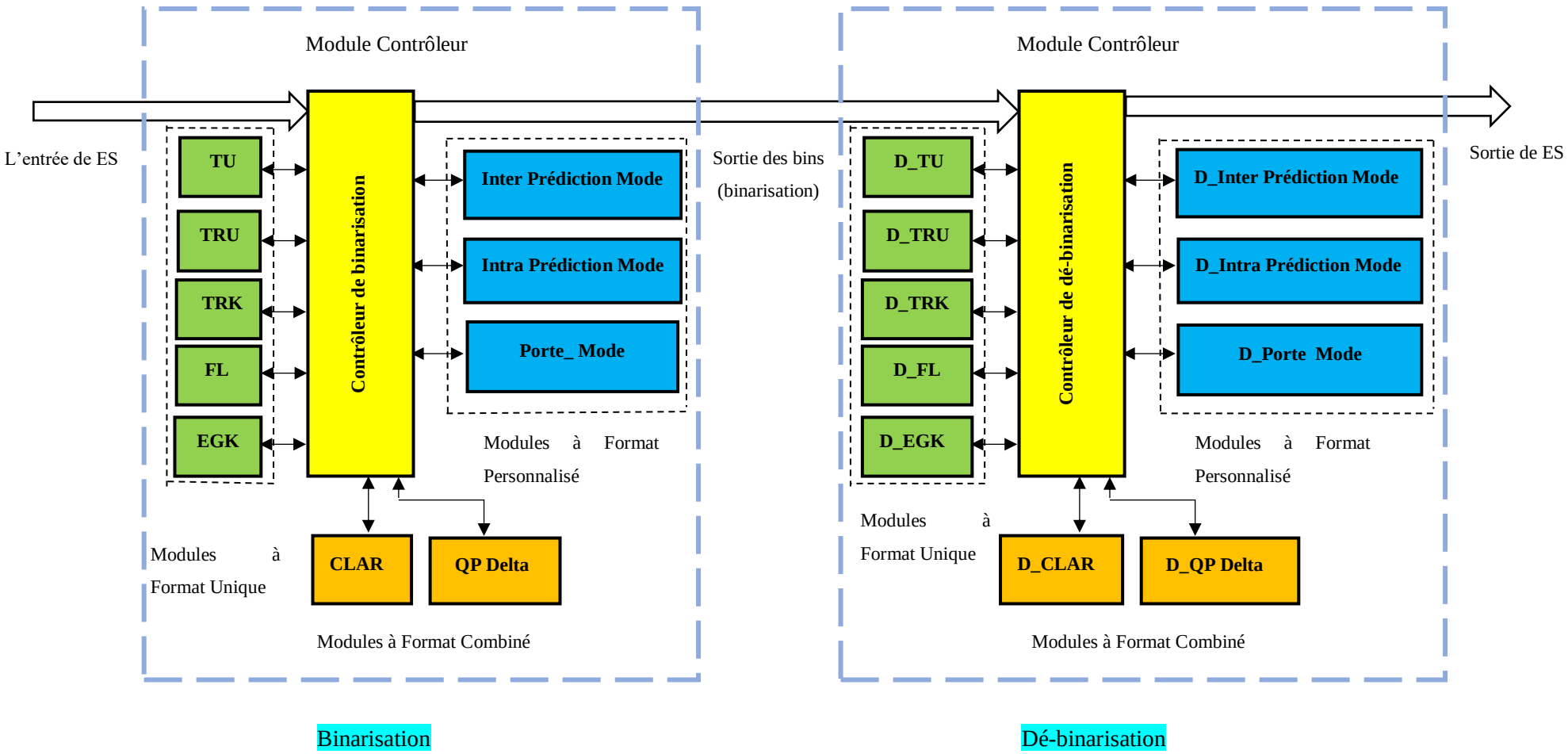


Figure 4.4: Architecture proposée de binarisation / dé-binarisation CABAC HEVC.

4.3.1 Architectures proposées pour les codes de binarisation

a) Architecture du code TU

À l'entrée de cette architecture on a trois variables T(i), T(i-1) et i. À la sortie on trouve le code binaire TU.

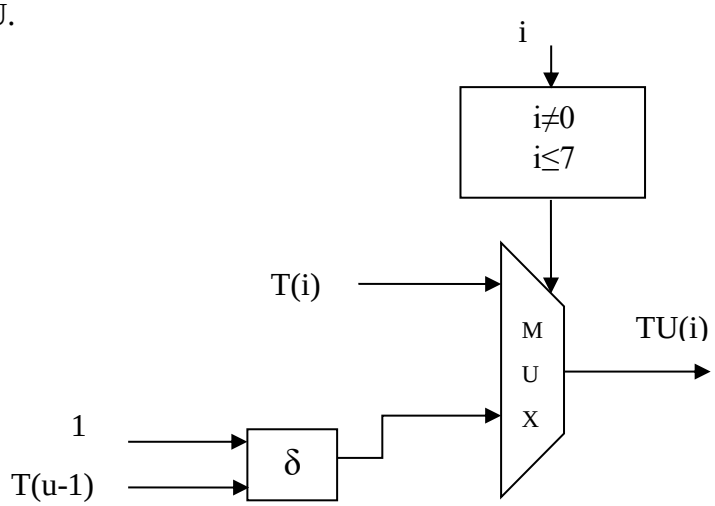


Figure 4.5 : Conception matérielle proposée du code TU.

b) Architecture du Code TRU

À l'entrée de cette architecture on a trois variables TRU, TRU(i-1) et i. À la sortie on trouve le code binaire TRU.

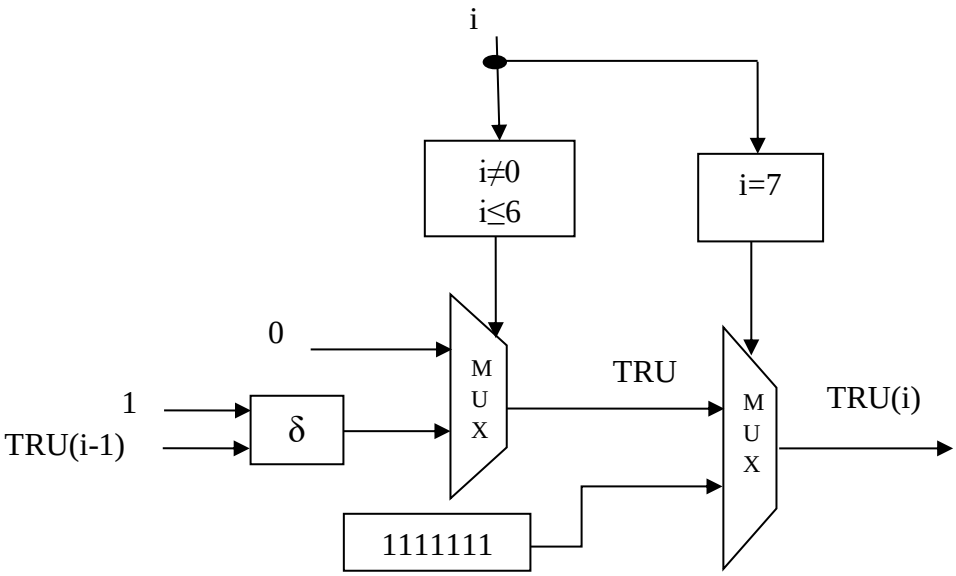


Figure 4.6 : Conception matérielle proposée du code TRU.

c) Code FL

À l'entrée de cette architecture on a deux variables le code binaire et i. À la sortie on trouve le code binaire FL.

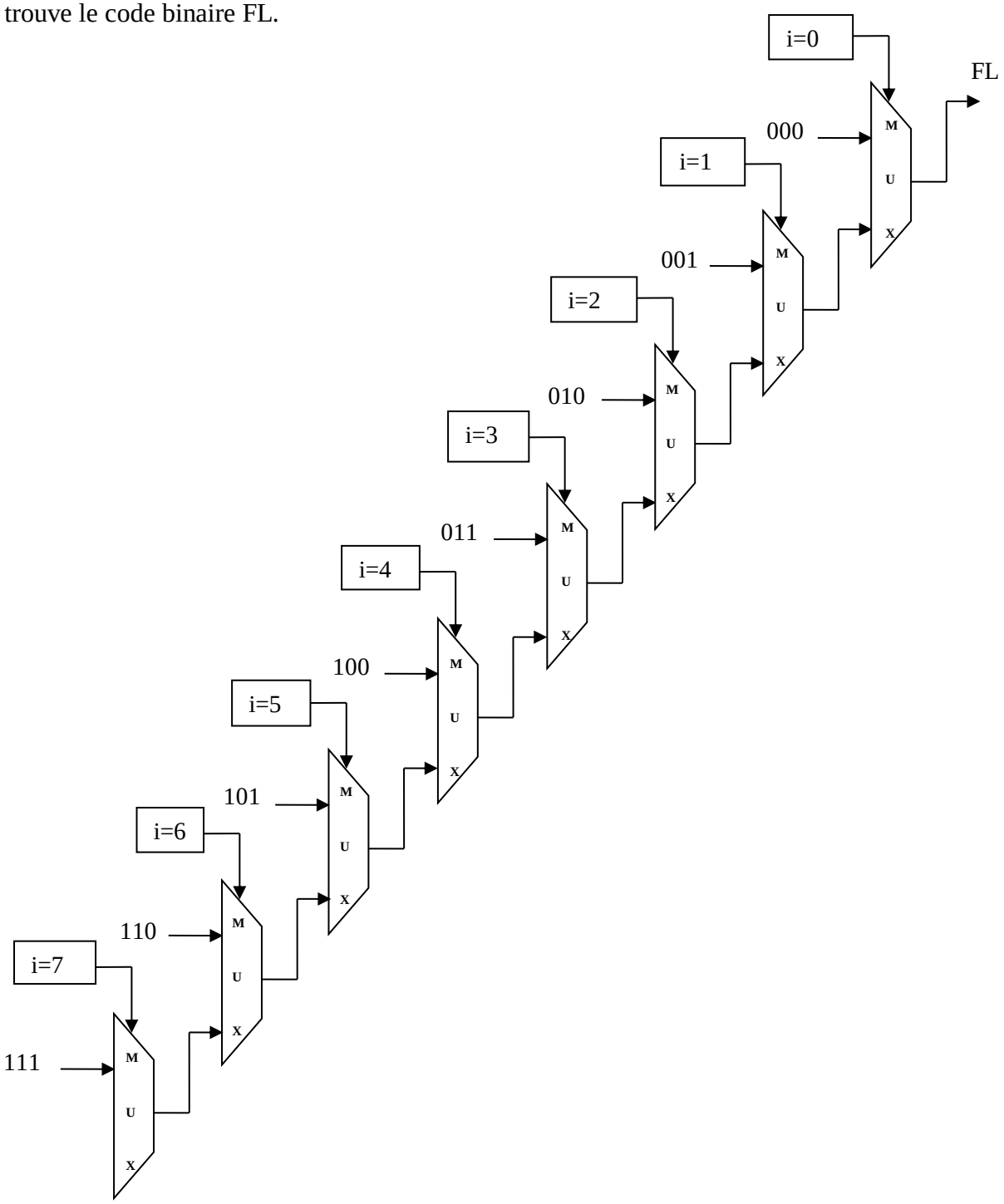


Figure 4.7 : Conception matérielle proposée du code FL.

d) Code TRK

À l'entrée de cette architecture on a trois variables TR (K, i) avec K=0 :4, K et i. À la sortie on trouve le code binaire TRK.

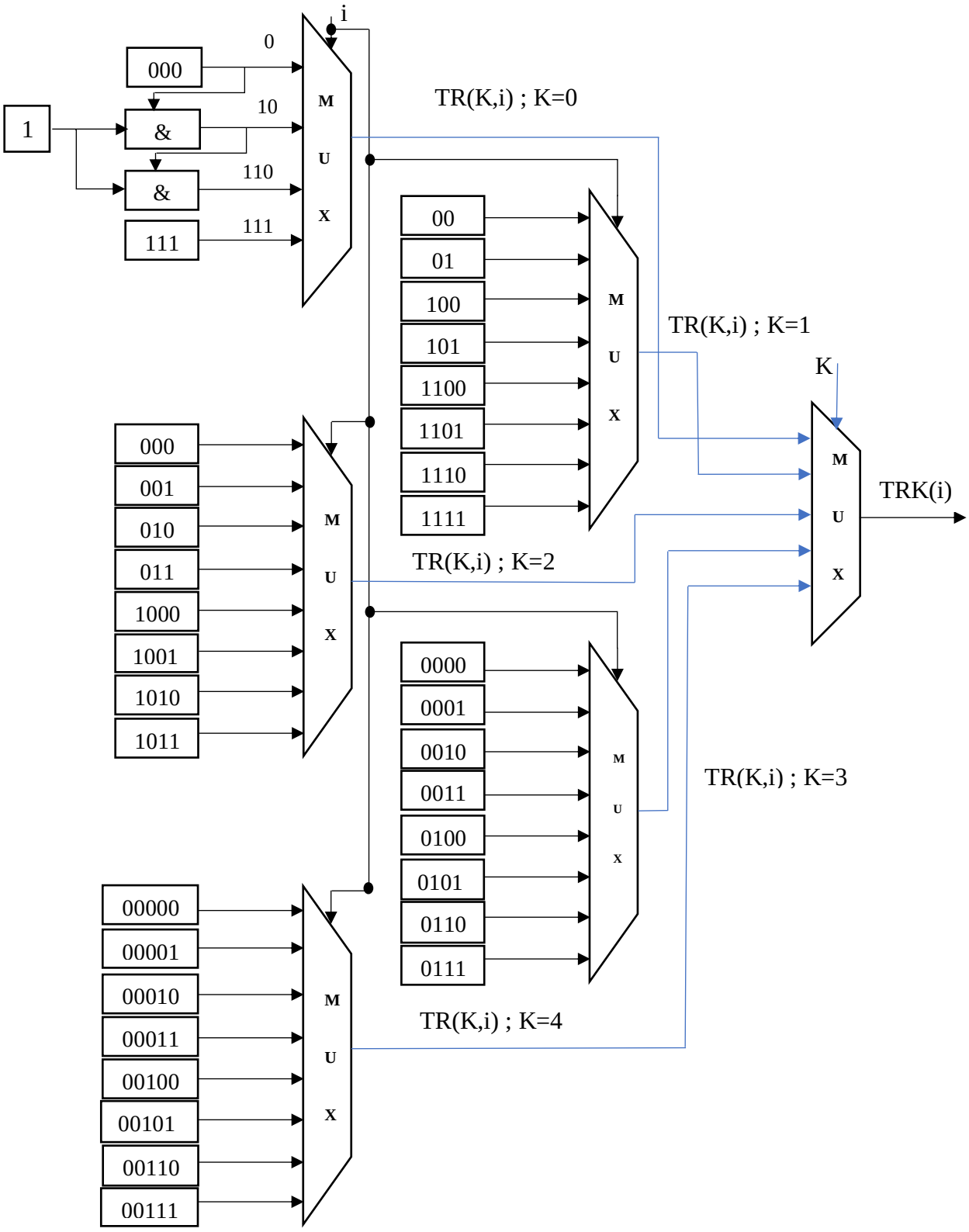


Figure 4.8 : Conception matérielle proposée du code TRK.

e) Code EGK

À l'entrée de cette architecture on a trois variables EG (K, i) avec K=1 :5, K et i. À la sortie on trouve le code binaire EGK.

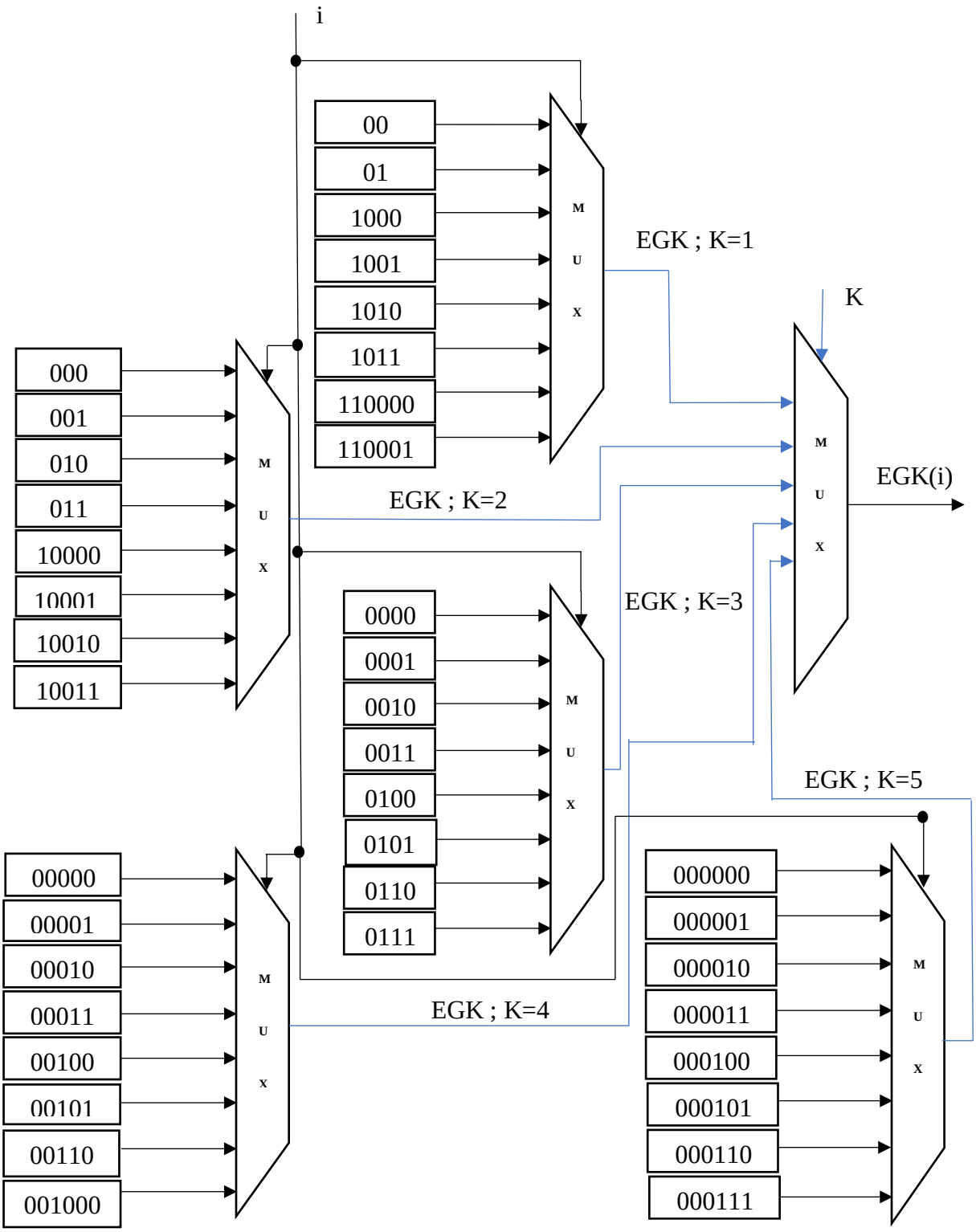


Figure 4.9 : Conception matérielle proposée du code EGK.

f) Code CALR

À l'entrée de cette architecture on a deux variables cRicePar et SE. À la sortie on trouve le code binaire CALR.

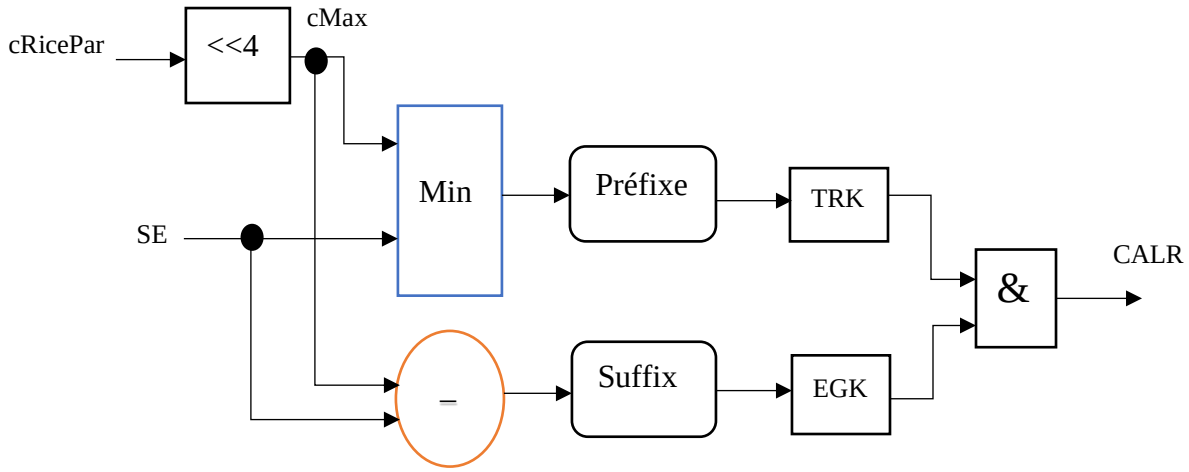


Figure 4.10 : Conception matérielle proposée du code CALR.

g) Code QP Delta

À l'entrée de cette architecture on a deux variables cRicePar=5 et Cu_qp_delta_abs. À la sortie on trouve le code binaire QP Delta.

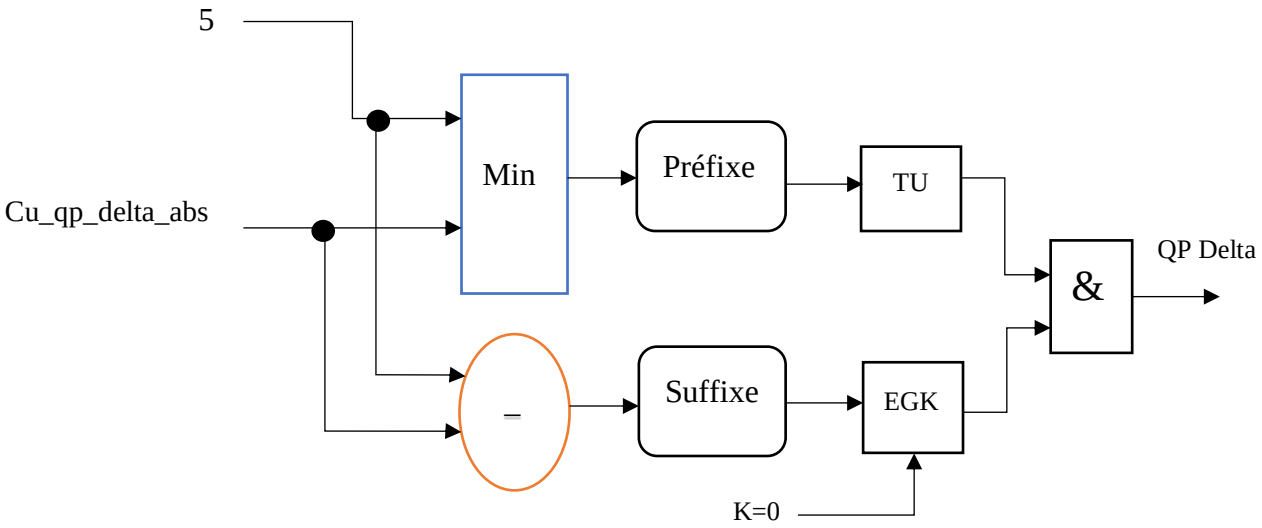


Figure 4.11 : Conception matérielle proposée du code Qp Delta.

h) Intra Prédiction Mode

À l'entrée de cette architecture on a une variable i. À la sortie on trouve le code binaire Intra Prédiction Mode.

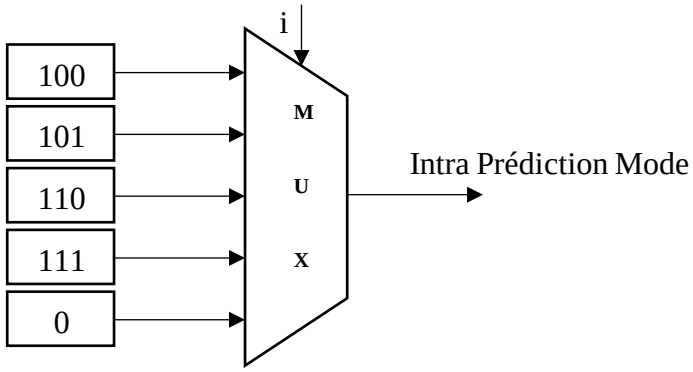


Figure 4.12 : Conception matérielle proposée du code Intra Prédiction.

i) Inter Prédiction Mode

À l'entrée de cette architecture on a une variable i. À la sortie on trouve le code binaire Inter Prédiction Mode.

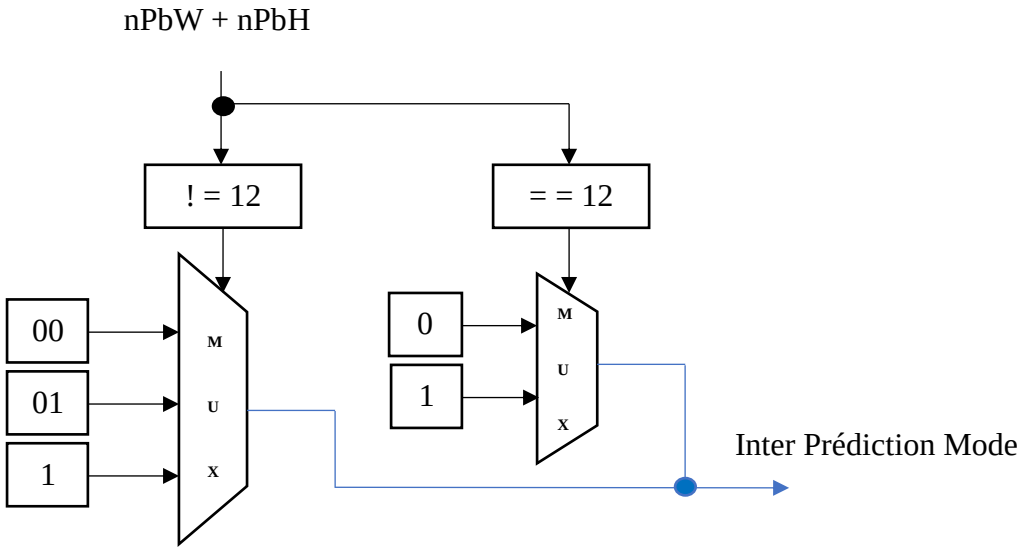


Figure 4.13 : Conception matérielle proposée du code Inter Prédiction.

j) Part Mode

À l'entrée de cette architecture on a une variable i, amp_enabled_flag, Cu_Pred_Mode et comp. À la sortie on trouve le code binaire Part Mode.

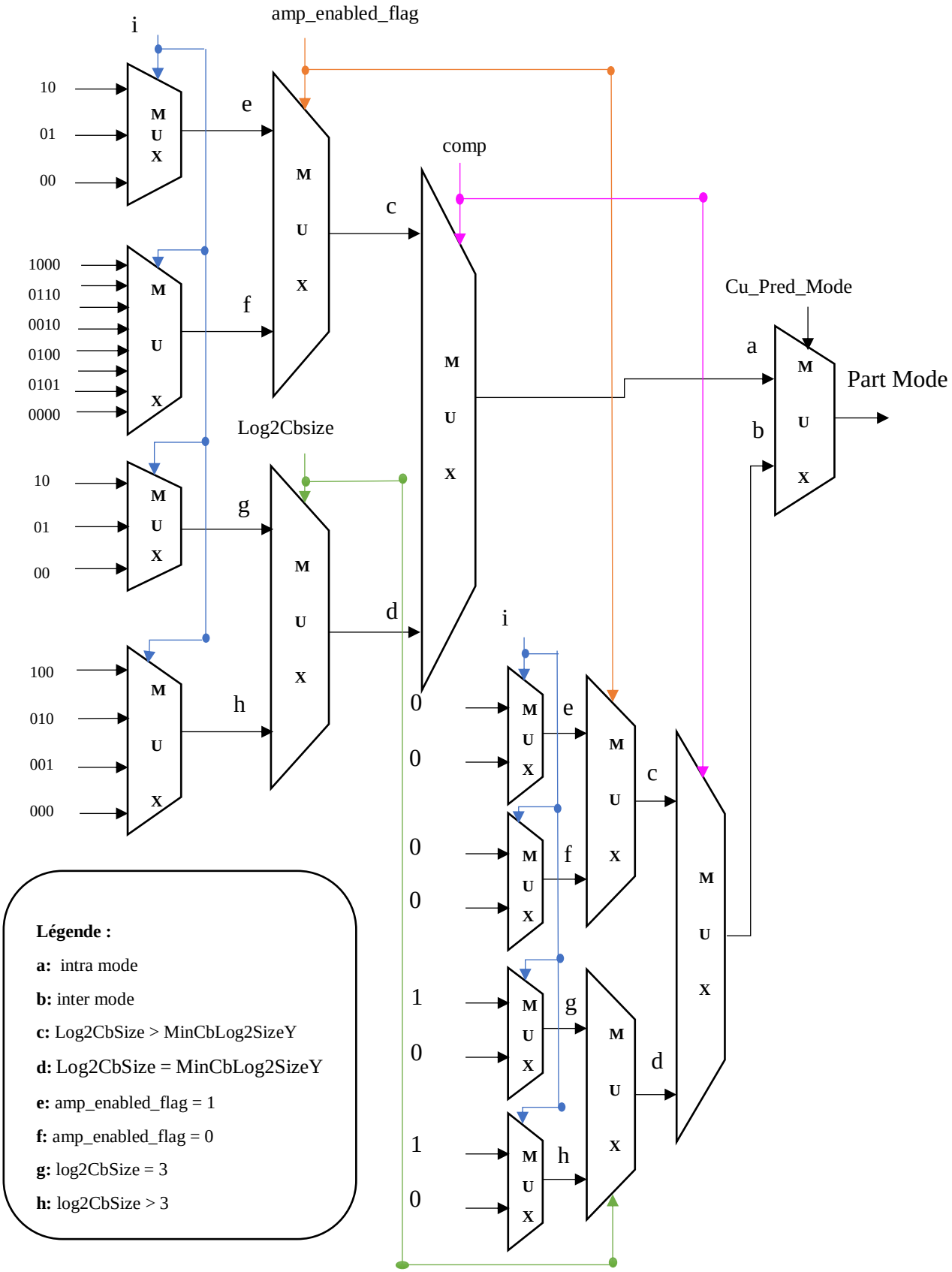


Figure 4.14 : Conception matérielle proposée du code Part mode.

4.3.2 Architecture proposée pour la dé-binarisation

Le processus de dé-binarisation est représenté sur le bloc DB (*Dé-Binariser*), placé à la sortie du décodeur afin de reconstruire les éléments syntaxiques à partir des bins décodés en appliquant l'opération inverse des codes binaires utilisés. De plus, le bloc de dé-binarisation converti les *bins* décodés en éléments de syntaxes selon leurs codes binaires utilisés au niveau du codage : TR (*Truncated rice*), TRU (*Truncated Unary*), EGk (*K-th order Exp-Golomb*), et FL (*Fixed length*).

La figure (4.15) (a et b) montre les architectures proposées pour la dé-binarisation des éléments syntaxiques suivants :

- Le reste des coefficients (*coeff_abs_level_remaining) ;*
- La position de dernier coefficient non nul (LSCx, LSCy) qui est basée sur quatre éléments syntaxiques : *last_sig_coeff_x_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_prefix, et last_sig_coeff_y_suffix* ;

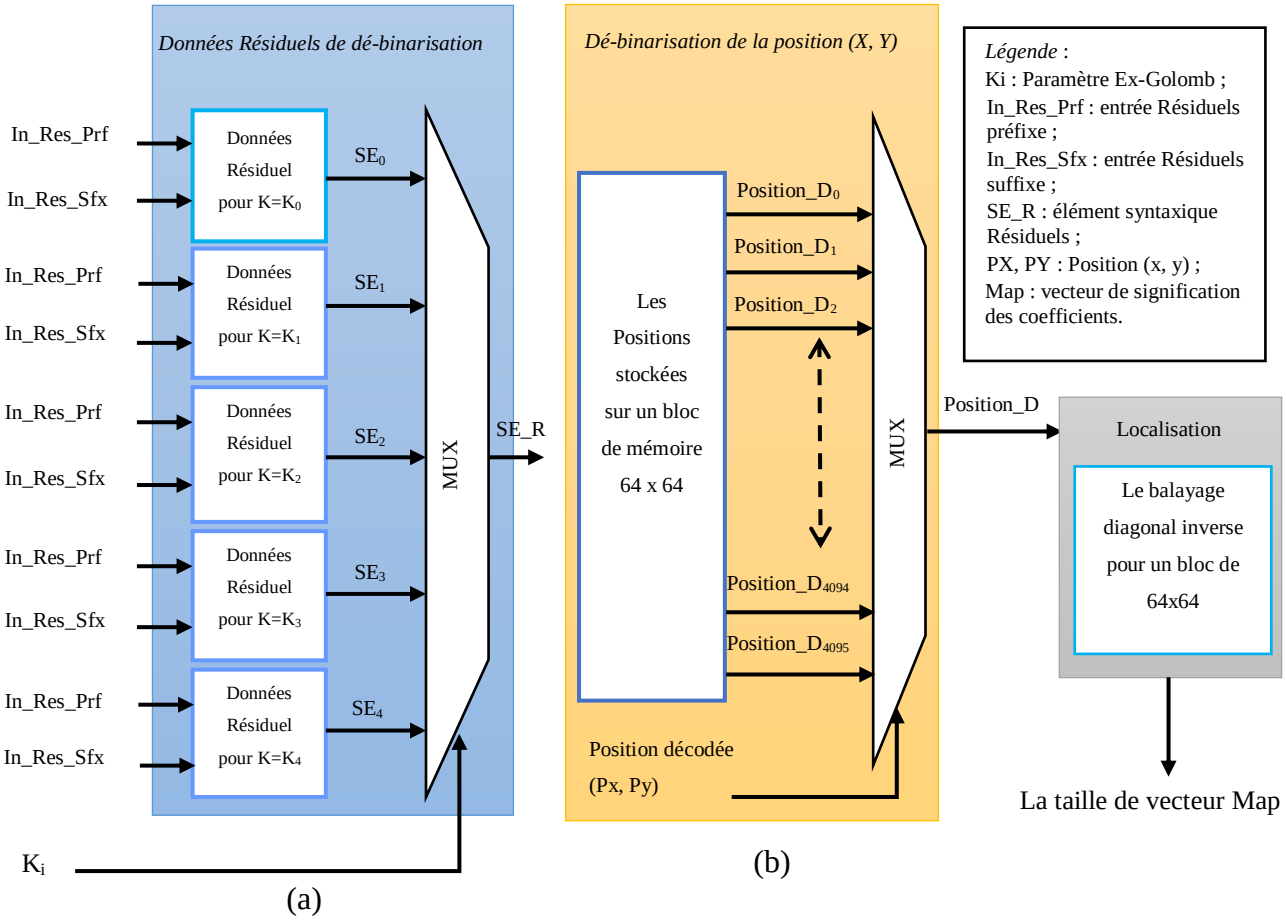
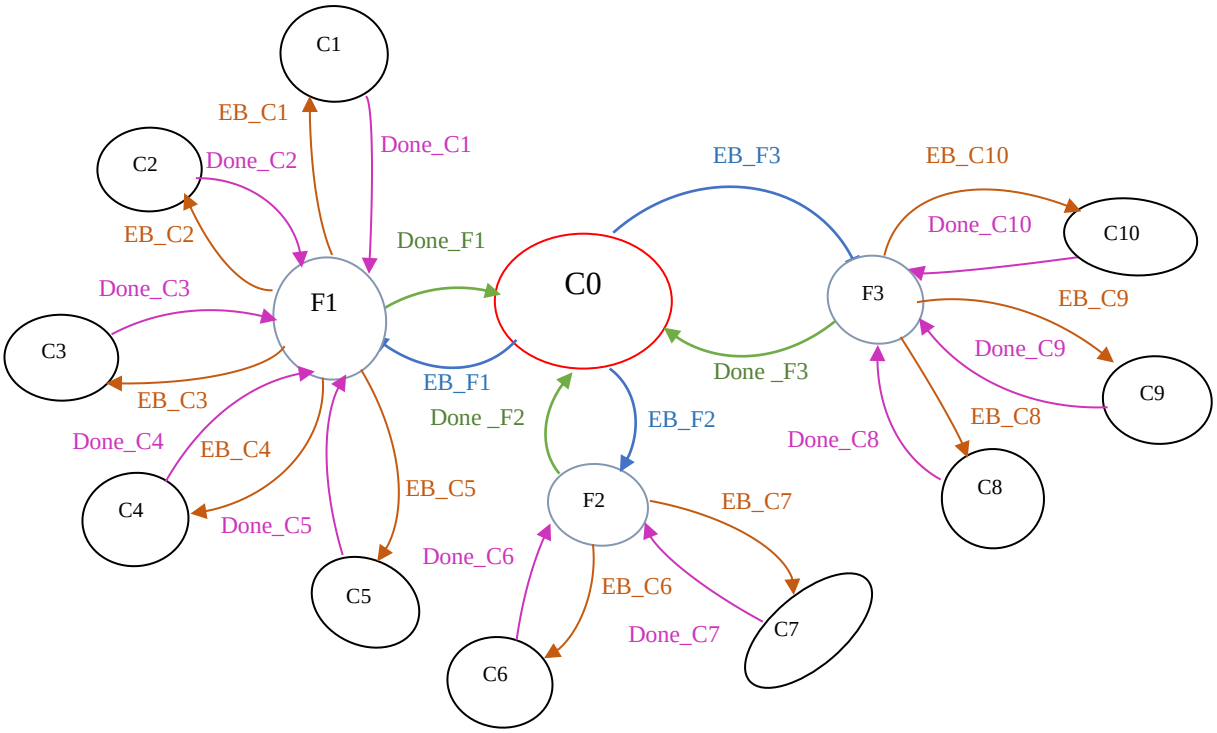


Figure 4.15 : Architectures proposées pour la dé-binarisation. (a) pour les résidus de coefficients. (b) pour la position de dernier coefficient non nul (LSCx, LSCy).

4.3.3 FSM *(finit state machine)*

Les machines à états finis (FSM) sont des circuits séquentiels utilisés dans de nombreux systèmes numériques pour contrôler le comportement des systèmes et des flux de données des systèmes et des chemins de flux de données. Les unités de contrôle et les séquenceurs sont des exemples de FSM.

La figure suivante montre un diagramme d'état FSM proposé pour l'architecture du notre binariseur. Nous pouvons remarquer que la transition d'un état a un autre est basée sur le signal d'activation (Enable) par contre le retour à l'état précédente ou la fin de l'état actuelle est basé sur le signal fin de tache (done).



Légende :

C0 : Contrôleur ;
C1, C2, …. , C10 : codes de binarisation ;
F1 : modules à formats unique ;
F2 : modules à format combiné ;
F3 : modules à format personnalisé ;
EB_F1, F2, F3 : signal d'activation Bloc F1, F2, F3 ;
EB_C1, C2, …., C10 : signal d'activation Bloc C1, C2,…,C10 ;

Done_F1, F2, F3 : Fin de processus ;
Done_C1, C2, ……, C10 : Fin de processus.

Figure 4.16 : Diagramme d'état FSM proposé pour le binarisateur.

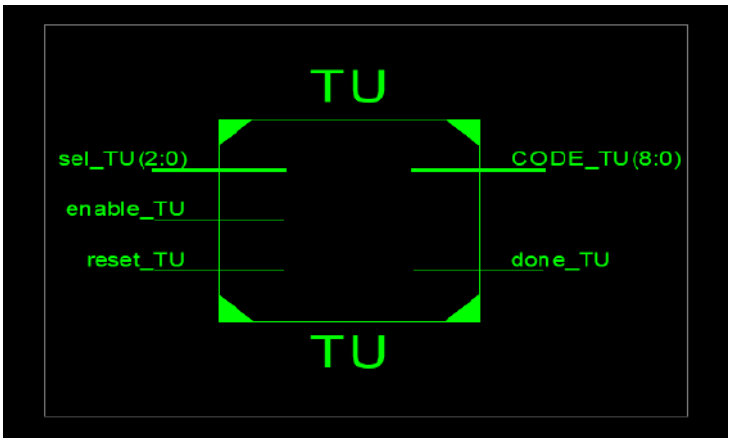
4.4 Partie synthèse et simulation

4.4.1 Synthèse et schéma RTL des codes de binarisation/dé-binarisation

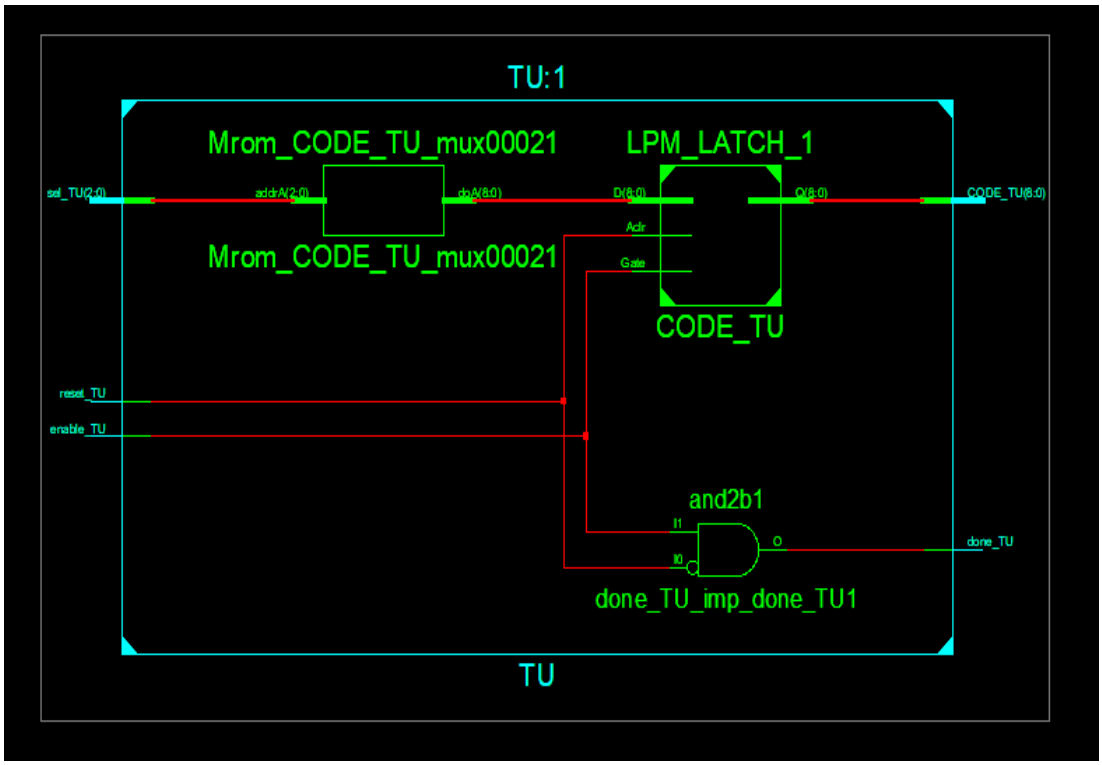
A) Binarisation

Les figures suivantes montrent les schémas RTL des dix codes de binarisation sous ISE.14.5.

- **Code TU :** La figure suivante montre le schéma Top Level du code de binarisation TU.



(a)



(b)

Figure 4.17 : Schéma RTL du code TU. (a) Top-Level, (b) Schéma détaillé.

- **Code TRU :** La figure suivante montre le schéma Top Level du code de binarisation FL.

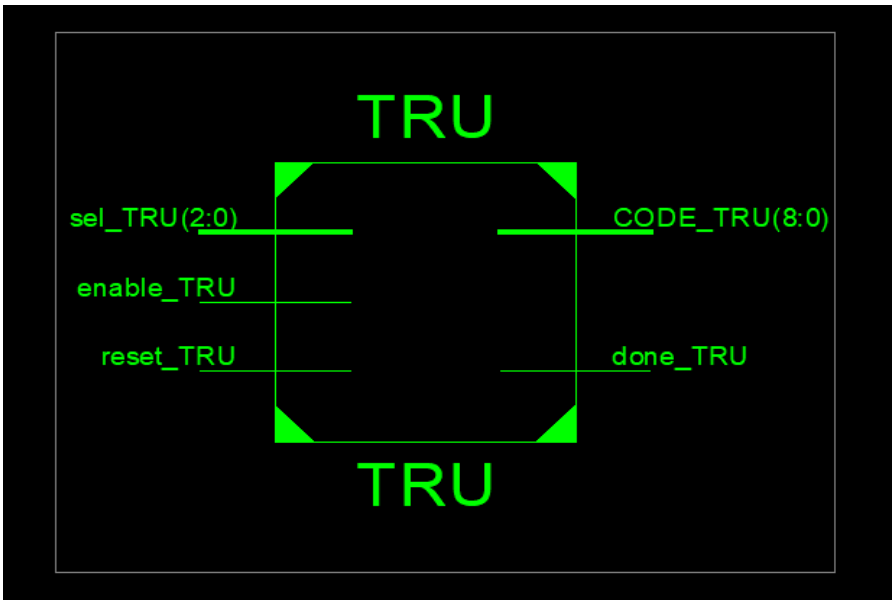


Figure 4.18 : Schéma RTL Top-Level du code TRU.

- **Code FL :** La figure suivante montre le schéma Top Level du code de binarisation FL.

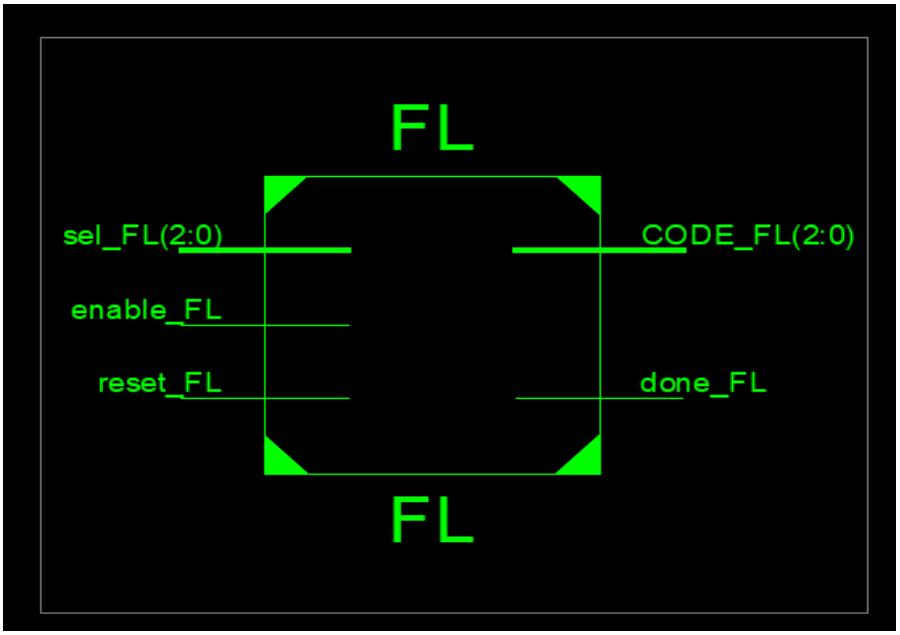


Figure 4.19 : Schéma RTL Top-Level du code FL.

- **Code TRK :** La figure suivante montre le schéma Top Level du code de binarisation TRK.

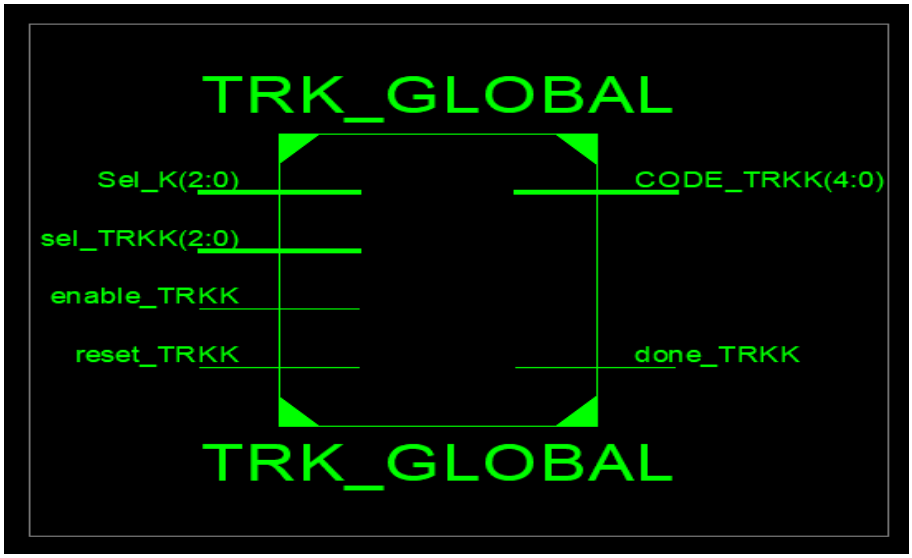


Figure 4.20 : Schéma RTL Top-Level du code TRK.

- **Code EGK :** La figure suivante montre le schéma Top Level du code de binarisation EGK.

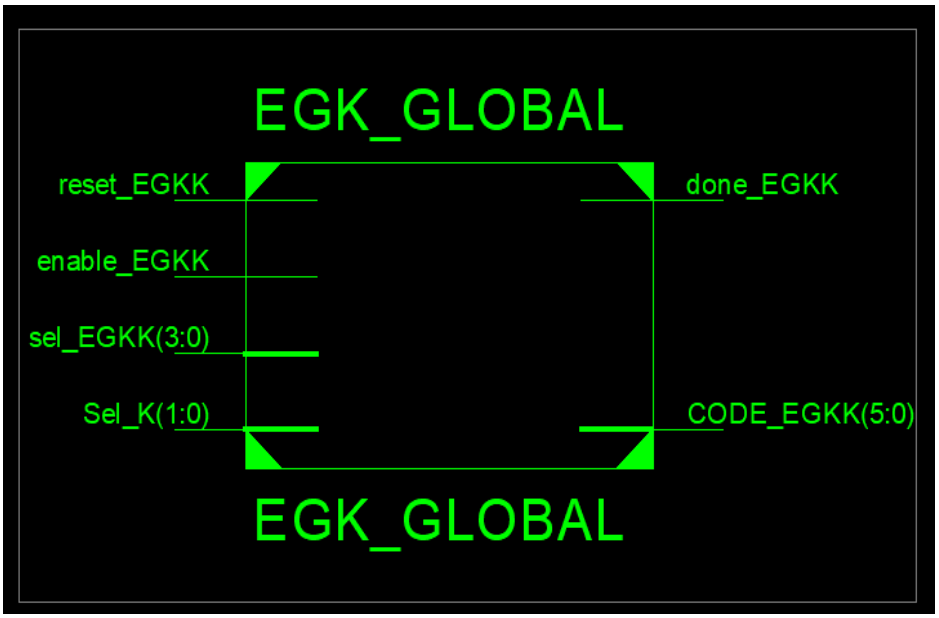


Figure 4.21 : Schéma RTL Top-Level du code EGK.

- **Code QP Delta :** La figure suivante montre le schéma Top Level du code de binarisation QP Delta.

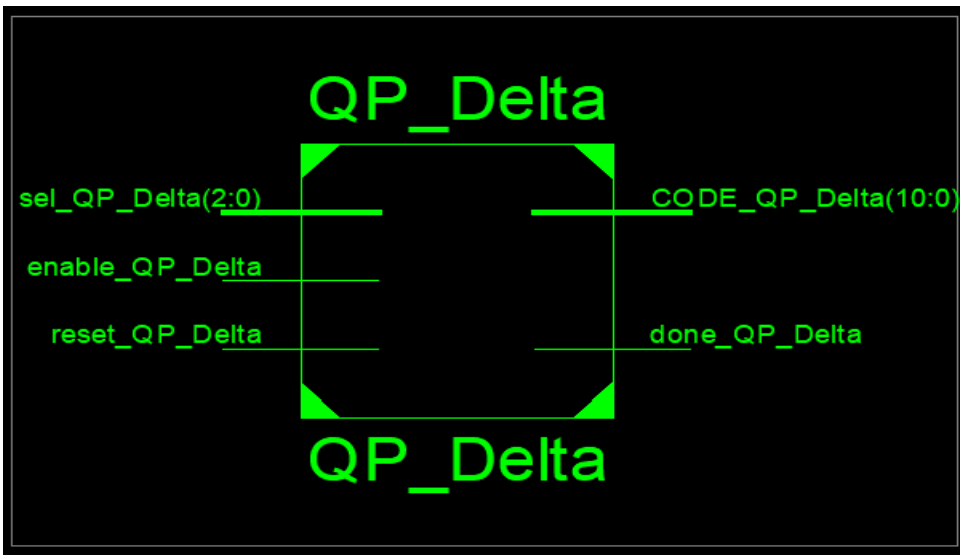


Figure 4.22 : Schéma RTL Top-Level du code QP Delta.

- **Code CALR :** La figure suivante montre le schéma Top Level du code de binarisation CALR.

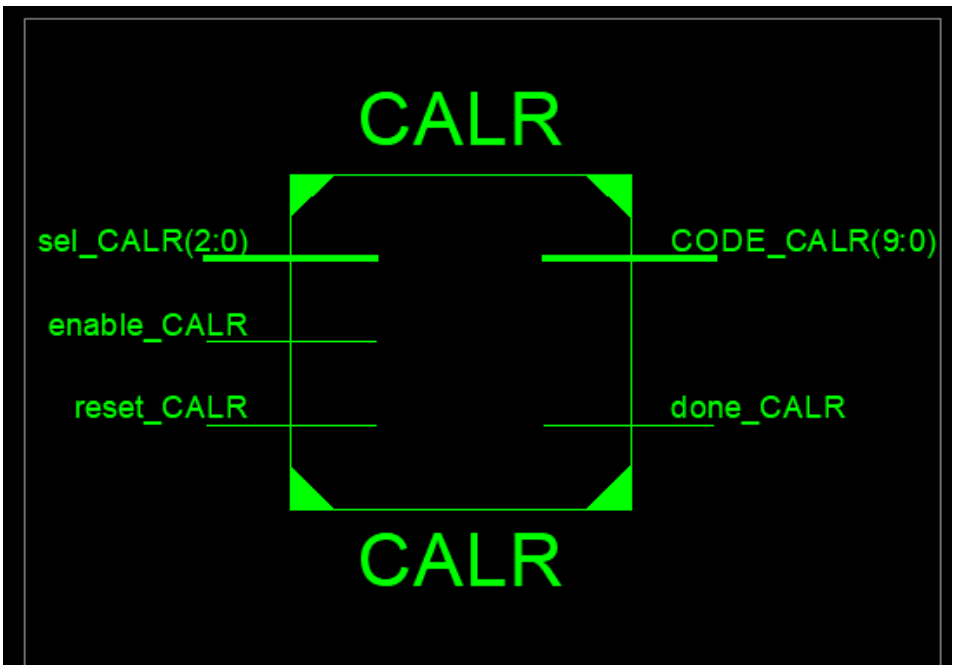


Figure 4.23 : Schéma RTL Top-Level du code CALR.

- **Code Intra Prédiction Mode :** La figure suivante montre le schéma Top Level du code de binarisation Intra Prédiction Mode.

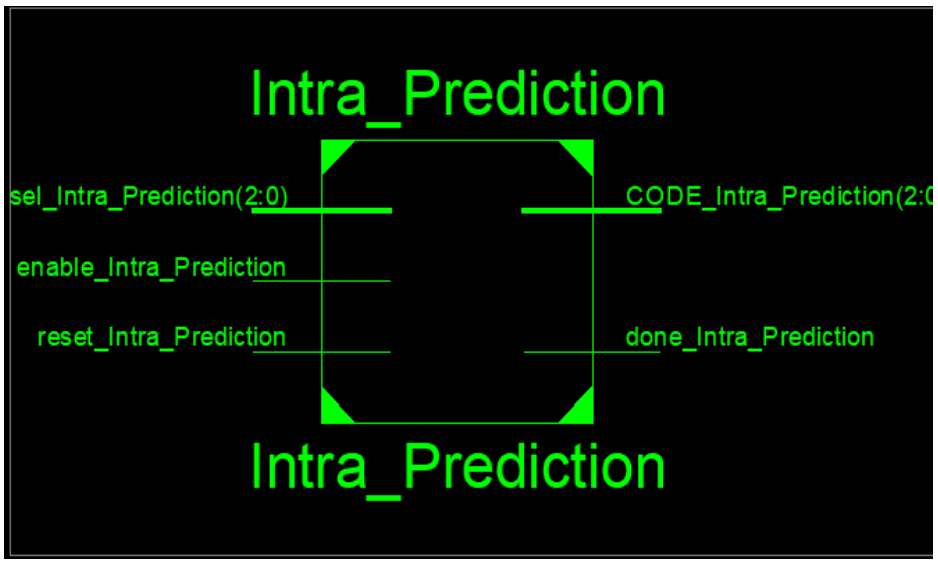


Figure 4.24 : Schéma RTL Top-Level du code Intra Prédiction Mode.

- **Code Inter Prédiction Mode :** La figure suivante montre le schéma Top Level du code de binarisation Inter Prédiction Mode.

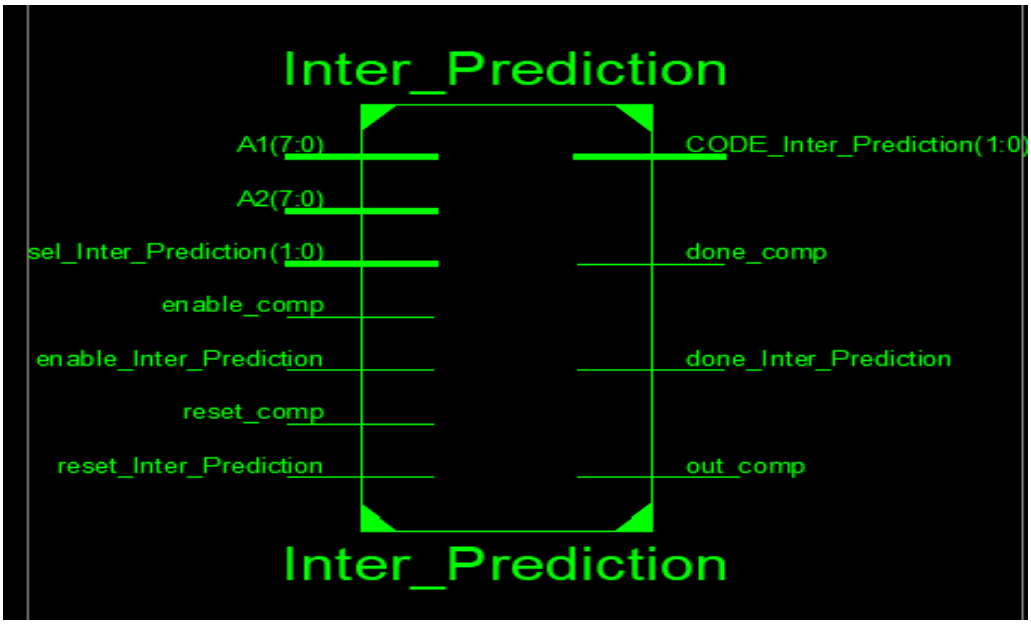


Figure 4.25: Schéma RTL Top-Level du code Inter Prédiction Mode.

- **Code Part Mode :** La figure suivante montre le schéma Top Level du code de binarisation Part Mode.

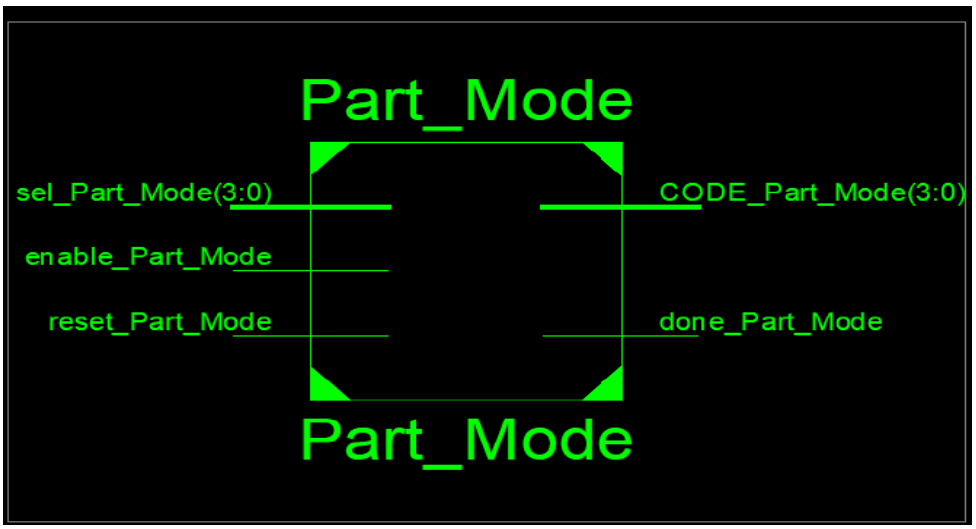


Figure 4.26 : Synthèse RTL Top-Level du code Part Mode.

B) Dé-binarisation

Les figures suivantes montrent les schémas RTL des codes de dé-binarisation sous ISE.14.5.

- **Le reste des coefficients :** L'architecture du La figure suivante montre le schéma Top Level du reste des coefficients de dé-binarisation.

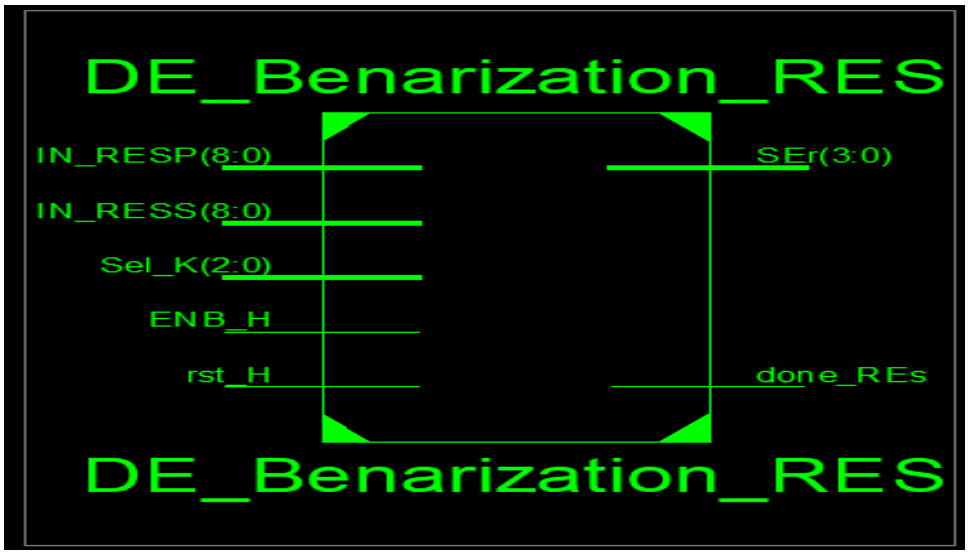


Figure 4.27 : Schéma RTL Top-Level des coefficients résiduels.

- **La position de dernier coefficient non nul (LSCx, LSCy) :** La figure suivante montre le schéma Top Level de la position de dernier coefficient non nul de dé-binarisation.

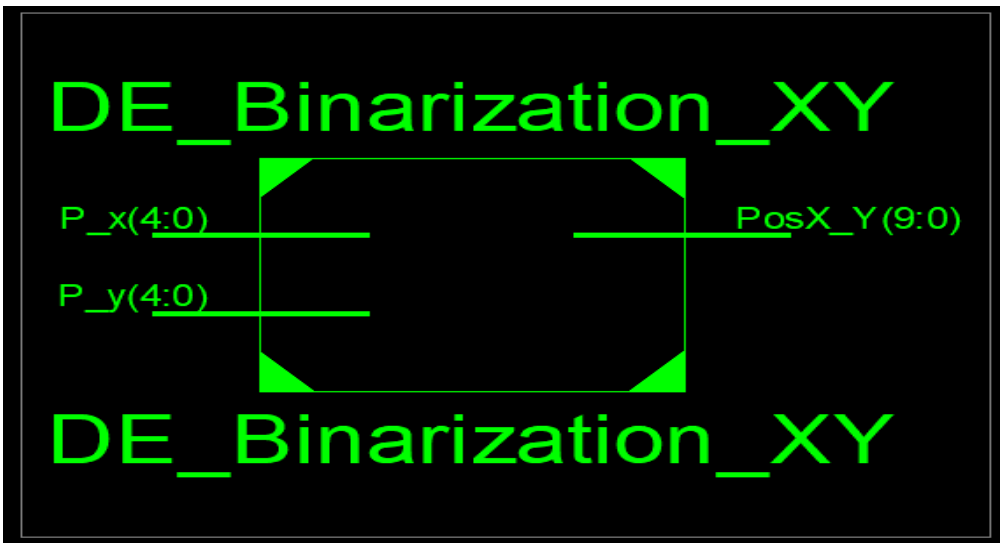


Figure 4.28 : Schéma RTL Top-Level de la position (x, y).

L'architecture du binarisateur / dé-binarisateur CABAC HEVC proposée est décrite en langage de description matérielle HDL. Il est synthétisé à l'aide de l'outil ISE de Xilinx version 14.5 et son implémentation est simulée sur le circuit FPGA virtex4 (xc4vlx15-12ff-668).

Notre binarisateur CABAC HEVC est simulé en utilisant le simulateur de Model Sim (PE Student Edition 6.3). La figure suivante illustre la description en VHDL de l'architecture du binarisateur / dé-binarisateur CABAC HEVC sous ISE 14.5.

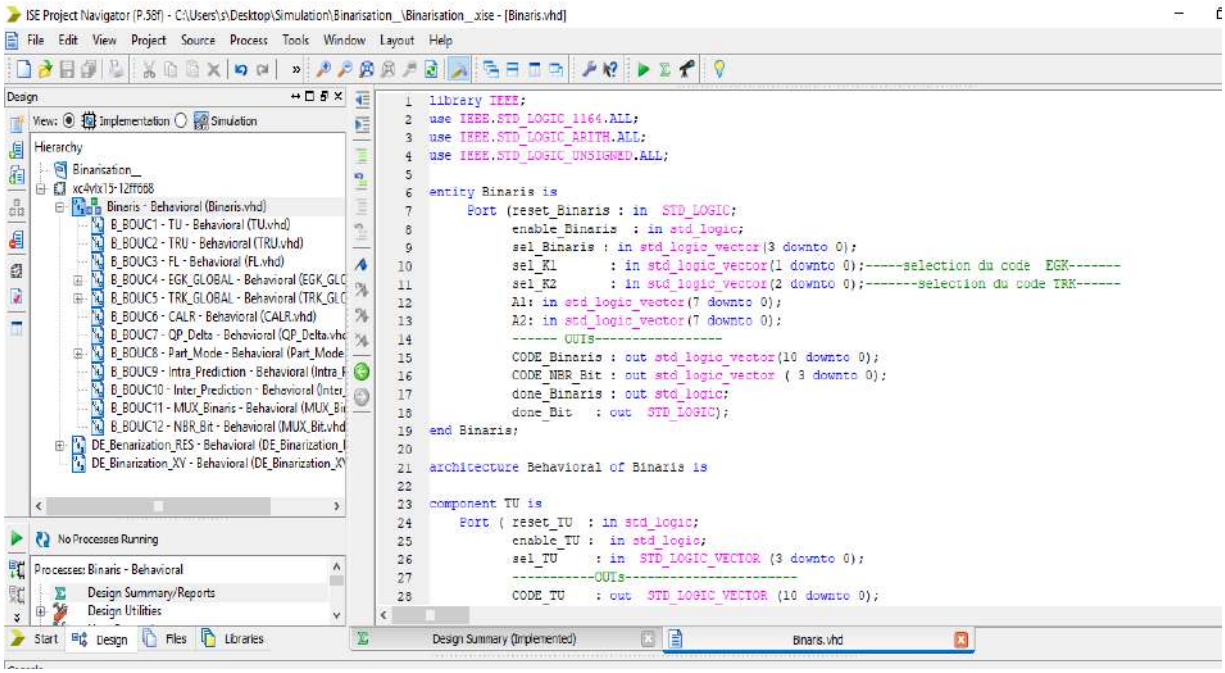


Figure 4.29 : description en VHDL de l'architecture du binarisateur / dé-binarisateur CABAC HEVC sous ISE 14.5.

4.4.2 Ressources, fréquences d'opération, chemin critique et puissance consommée

Le tableau (4.1) illustre la synthèse des codes de binarisation et dé-binarisation avec l'estimation de la consommation énergétique pour chaque Bloc. Nous remarquons que le chemin critique le plus petit est obtenu par l'architecture du code de binarisation FL. Par contre, il consomme plus d'énergie avec un nombre de ressources plus grand. La fréquence d'opération maximale est obtenue par les blocs qui assurent la binarisation des codes suivants : TU, TRU, FL, CALR et QP Delta. Mais le nombre de ressources est intéressant. De plus, Les petites valeurs de la consommation d'énergie sont estimées par les blocs TRU, TRK, CALR, Part Mode, Inter Prédiction et Intra Prédiction. Cependant la consommation en ressources est plus élevée pour les six blocs.

La comparaison des performances en consommation énergétique de notre architecture avec les travaux les plus récents dans le domaine de la compression vidéo : Il est à signaler que l'énergie consommée par notre architecture est meilleur que les architectures proposées par **Alonso** [33] et **Ramos** [34] par contre elle est plus grande que l'énergie estimée par les solutions **Pham** [35] et **Dinh** [36]. Notre architecture est basée sur la binarisation/dé-binarisation cependant que les autres solutions sont basées sur le binarisateur seulement.

Tableau 4.1 : Ressources, fréquences d'opération, chemin critique et puissance consommée pour les codes de binarisation/dé-binarisation.

Paramètres \ Code		Binarisation										Dé-binarisation	
		TU	**TRU**	**FL**	**TRK**	**EGK**	**CALR**	**QP Delta**	**Part mode**	**Inter_pré d**	**Intra_préd**	**Le reste des coefficients**	**La position de dernier coefficient non nul**
Fréquence maximale (MHz)		_	_	_	1083.658	1083.658	_	_	664.077	948.047	1083.658	1083.658	_
Chemin Critique (ns)		4.939	4.921	4.868	5.001	5.236	4.921	4.962	5.263	_	4.868	4.875	_
Puissance Consommée Totale (W)		0.173	0.169	0.184	0.169	0.169	0.168	0.175	0.169	0.169	0.169	0.171	0.166
Nombre de Slices	Utilisé	1	19	1	4	14	2	3	_	3	4	_	202
	Disponible	6144	6144	6144	6144	6144	6144	6144	_	6144	6144	_	5472
Nombre de flip flops de slices	Utilisé	_	17	_	_	12	_	_	_	_	_	_	_
	Disponible	_	12288	_	_	12288	_	_	_	_	_	_	_
Nombre de LUTs à 4 entrées	Utilisé	1	35	1	7	25	3	6	6	5	7	7	373
	Disponible	12288	12288	12288	12288	12288	12288	12288	12288	12288	12288	10944	10944
Nombre d'IOBs cautionnés	Utilisé	9	15	9	15	14	16	17	17	16	14	15	20
	Disponible	320	320	320	320	320	320	320	320	320	320	240	240
	Utilisé	1	1	1	1	1	1	1	_	1	1	_	_

Nombre de GCLKs	Disponible	32	32	32	32	32	32	32	_	32	32	_	_
Nombre de slices occupées	Utilisé	_	_	_	_	_	_	_	4	_	_	4	_
	Disponible	_	_	_	_	_	_	_	6144	_	_	5472	_
IOBs loquets	Utilisé	_	_	_	_	_	_	_	10	_	_	7	_
Nombre de slices contenant uniquement la logique liée	Utilisé	_	_	_	_	_	_	_	4	_	_	4	_
	Disponible	_	_	_	_	_	_	_	4	_	_	4	_
Nombre de slices contenant la logique sans liaison	Utilisé	_	_	_	_	_	_	_	0	_	_	0	_
	Disponible	_	_	_	_	_	_	_	4	_	_	4	_
Nombre de BUFG / BUFGCTRLs	Utilisé	_	_	_	_	_	_	_	1	_	_	1	_
	Disponible	_	_	_	_	_	_	_	32	_	_	32	_
Nombre utilisé comme BUFG Utilisé		_	_	_	_	_		_	1	_	_	1	_
Fanout moyen des réseaux non-clockés		_	_	_	_	_		_	1.58	_	_	1.75	_

4.4.3 Simulation des codes de binarisation/dé-binarisation

A) Binarisation

Les figures suivantes montrent la simulation des dix codes de binarisation sous Model Sim Version 10.4a.

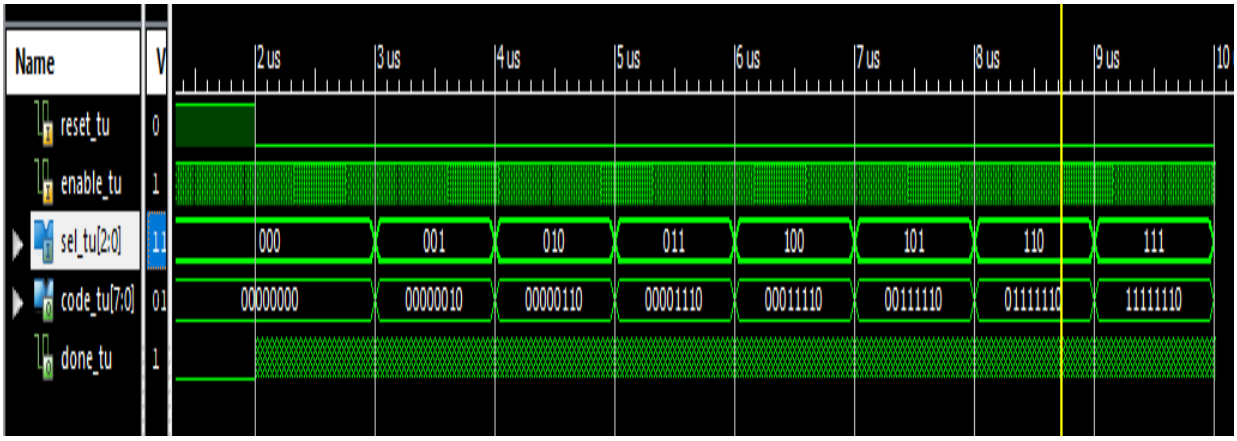


Figure 4.30 : Simulation sous Model Sim du code TU.

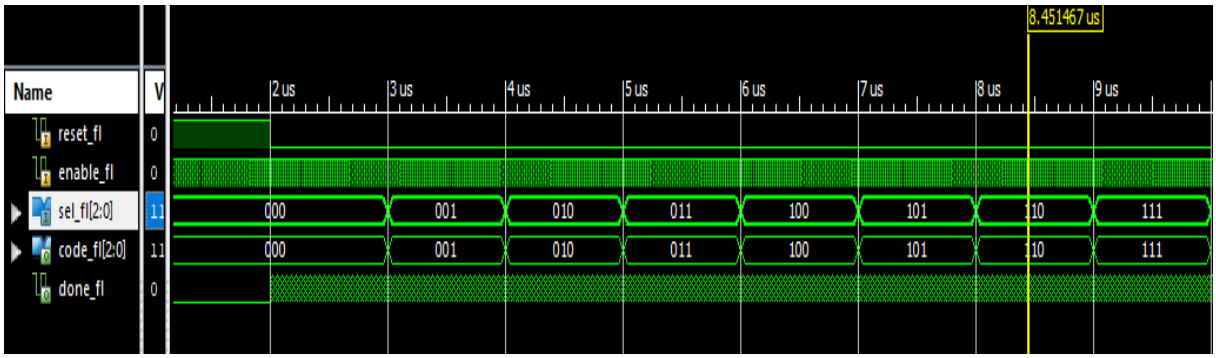


Figure 4.31 : Simulation sous Model Sim du code FL.

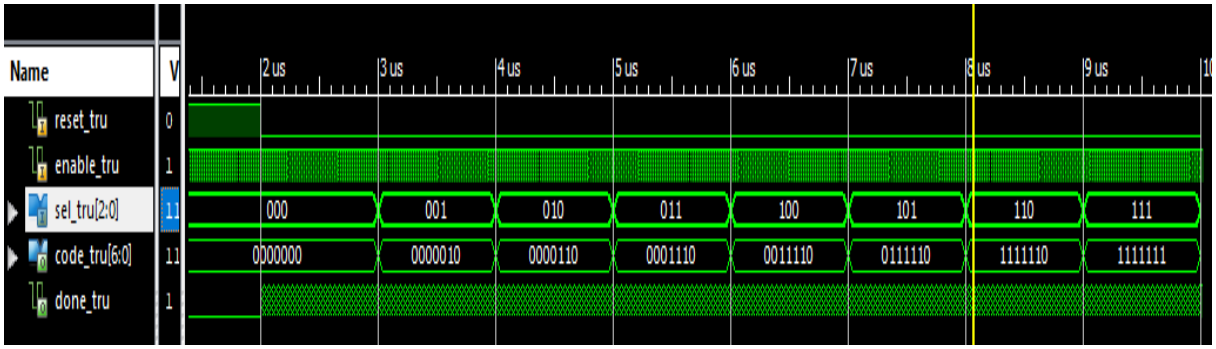


Figure 4.32 : Simulation sous Model Sim du code TRU.



Figure 4.33 : Simulation sous Model Sim du code TRK.

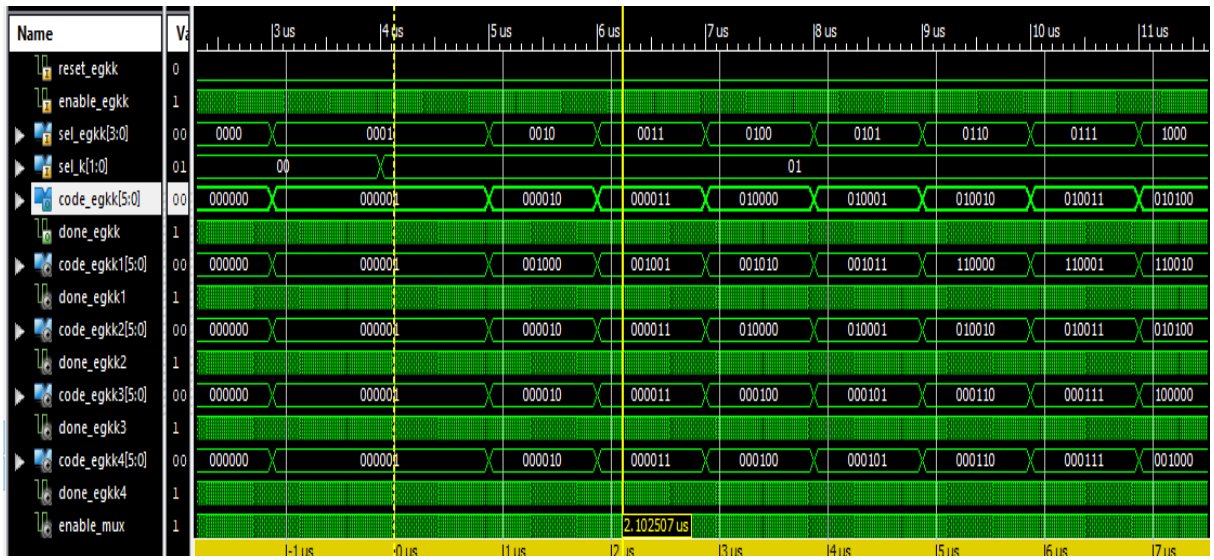


Figure 4.34 : Simulation sous Model Sim du code EGK.

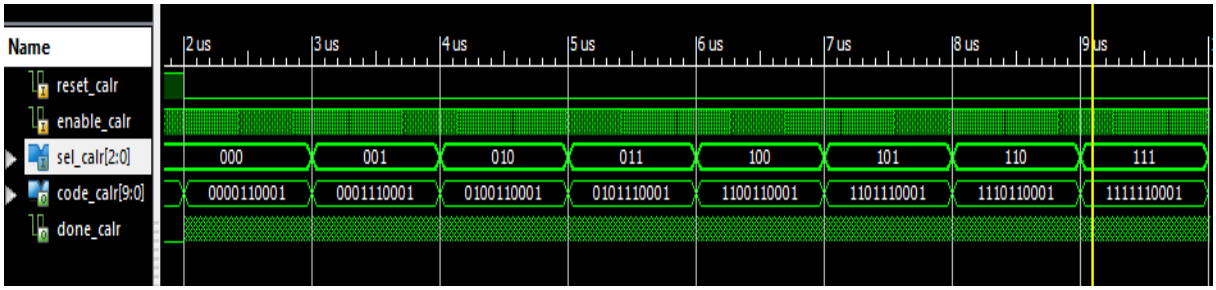


Figure 4.35 : Simulation sous Model Sim du code CALR.

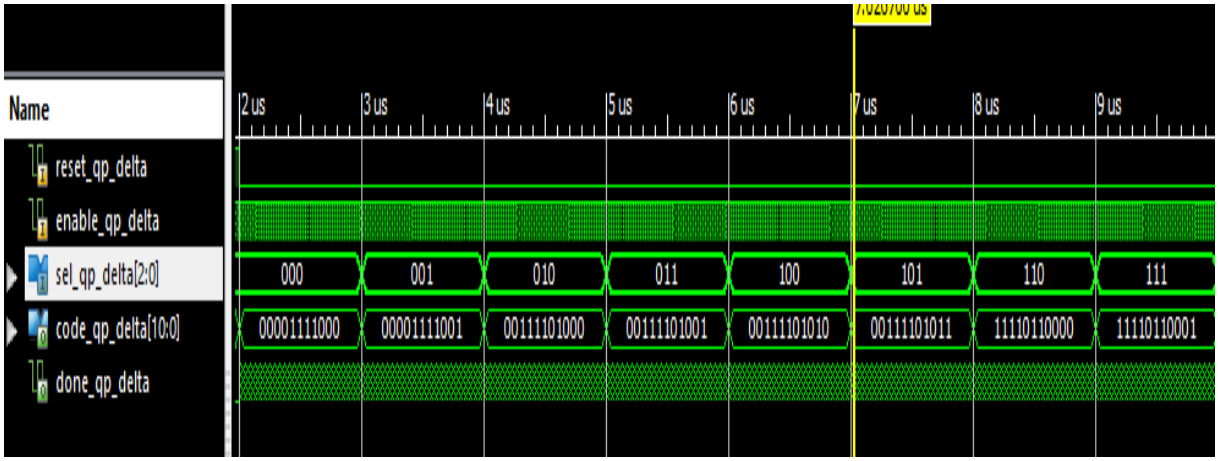


Figure 4.36 : Simulation sous Model Sim du code QP Delta.

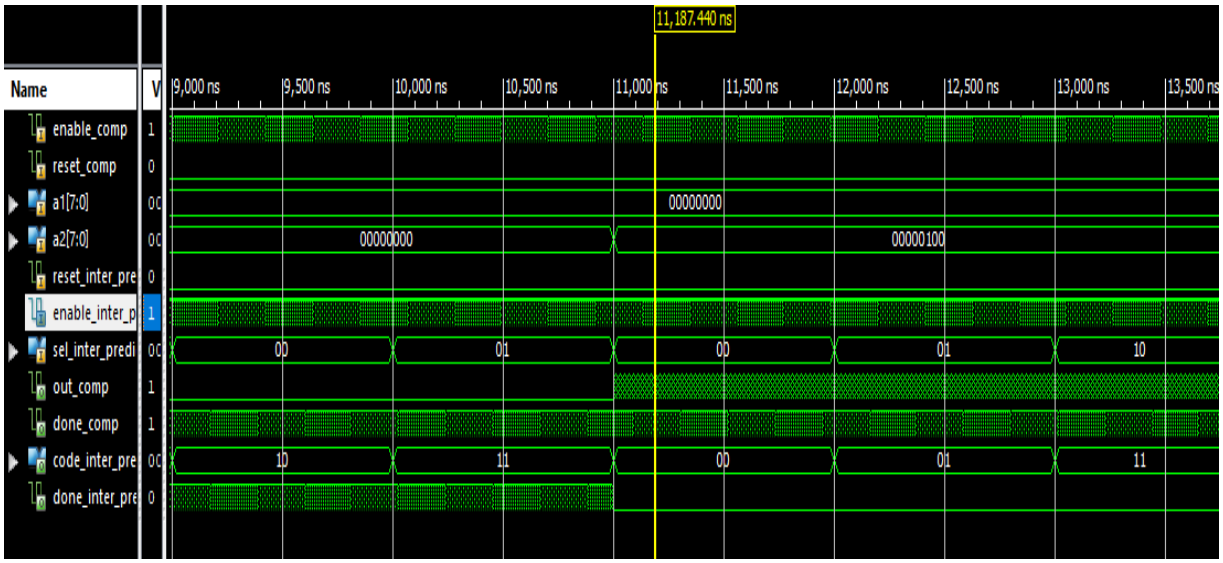


Figure 4.37 : Simulation sous Model Sim du code Inter Prédiction.

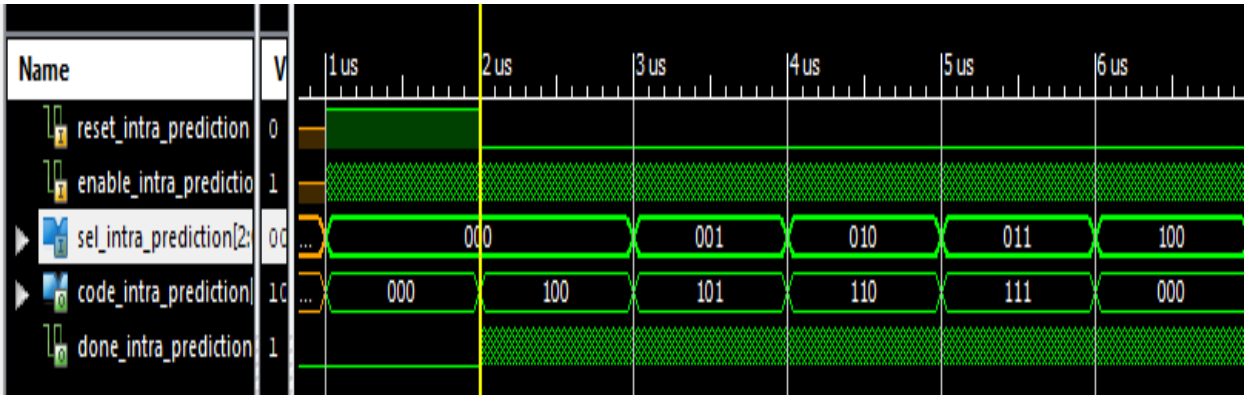


Figure 4.38 : Simulation sous Model Sim du code Intra Prédiction.

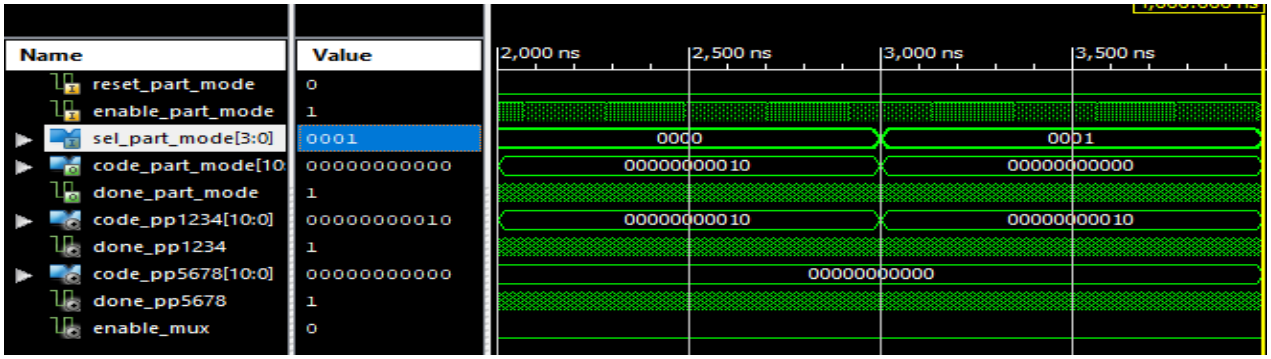


Figure 4.39 : Simulation sous Model Sim du code Part Mode.

B) Dé-binarisation

Les figures suivantes montrent la simulation de la position (x,y) et les coefficient résiduels de dé-binarisation sous Model Sim Version 10.4a.

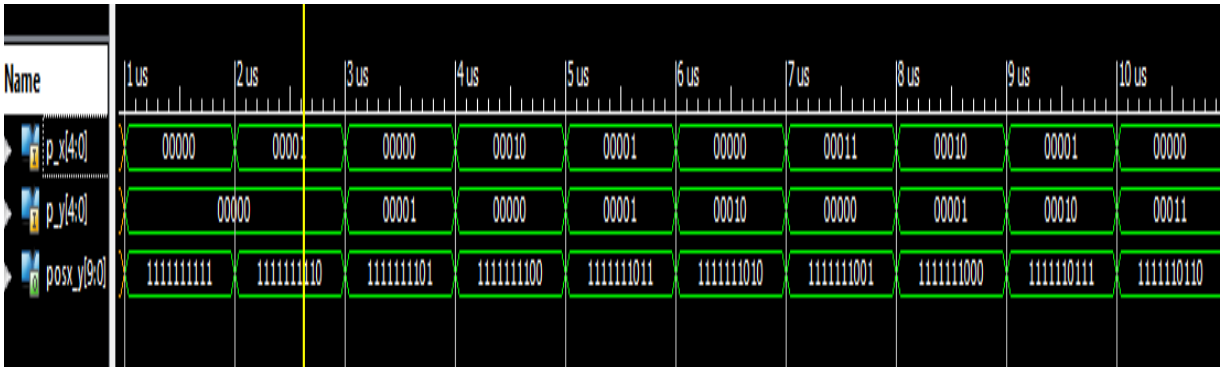


Figure 4.40 : Simulation sous Model Sim du la position (x,y).

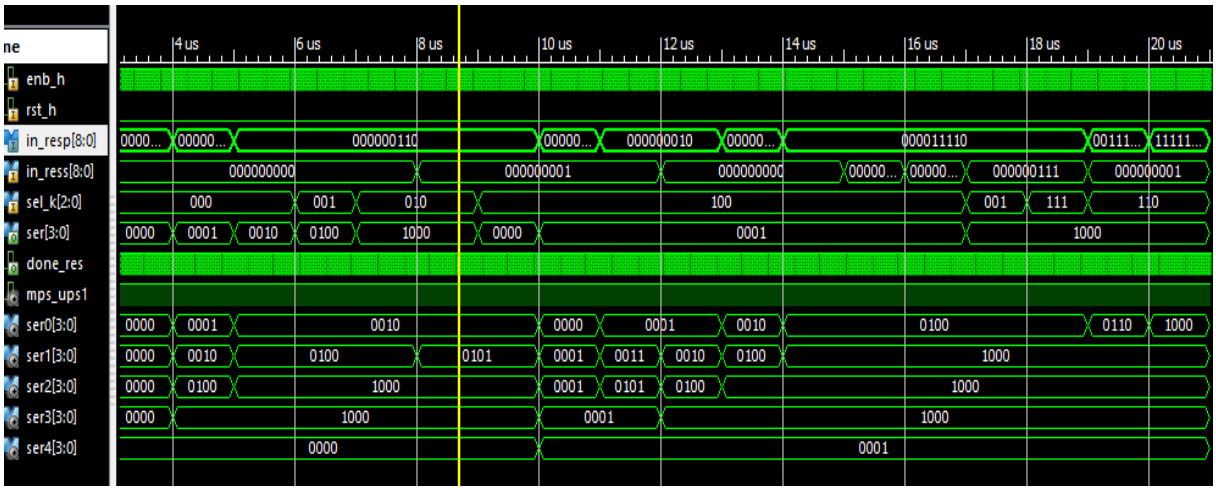


Figure 4.41 : Simulation sous Model Sim des coefficients résiduels.

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'architecture globale proposée du binarisateur / dé-binarisateur CABAC HEVC, ainsi pour les codes de binarisation et la dé-binarisation. Une étude détaillée de chaque code a été donnée à part, avec la partie contrôle. De plus, la description matérielle des deux parties (binarisation/dé-binarisation) a été présentée suivi par une synthèse, une simulation et une présentation des résultats obtenus pour les ressources, fréquences d'opération, chemin critique et puissance consommée. A la fin du chapitre, une comparaison des résultats est donnée avec les travaux existants dans la littérature.

Conclusion générale

Conclusion générale

Les recherches actuelles en matière de compression vidéo portent essentiellement sur le codage des données multimédias circulant sur Internet, ordinateurs de bureau et portables, tablettes et télévision.

L'objectif principal de notre travail est d'étudier, de concevoir et d'implémenter sur FPGA un binarisateur / dé-binarisateur dans la norme de compression vidéo HEVC/H.265.

Les travaux présentés dans ce mémoire s'intéressent à l'étude, la conception et l'implémentation du binarisateur / dé-binarisateur CABAC HEVC. En effet, le codeur CABAC constitue le bloc d'entrée du décodeur HEVC, et le bloc de sortie du codeur HEVC. Il est à signaler que le binarisateur fait parties des trois blocs du CABAC : binarisation, codage arithmétique et modalisation de contexte.

Notre travail a été organisé principalement en deux phases, phase théorique et phase pratique :

Dans la première phase, d'abord une étude détaillée du les normes de compression vidéo récentes ont été présentées ainsi que les techniques associées à la norme de compression HEVC ont été détaillées. Ensuite, les différents modules du système de communication et la théorie d'information ont été introduit. Suivi par la présentation du codage entropique avec quelques types de codage existants. Puis nous avons présenté la binarisation, la dé-binarisation, ainsi que les éléments syntaxiques utilisés au niveau de transformation dans le CABAC HEVC.

Dans la deuxième phase, nous avons concevoir les organigrammes des dix codes de binarisation avec leurs implémentations sous Matlab. Ce passage de l'implémentation logicielle sous Matlab nous a permis de maitriser les algorithmes des codes de binarisation /dé-binarisation utilisés dans la norme de compression vidéo HEVC avant de passer à l'implémentation en langage VHDL sous l'ISE. Nous avons terminé cette phase par la conception et l'implémentation sur FPGA des codes de binarisation / dé-binarisation en utilisant l'espace de développement ISE.

Notre architecture du binarisateur / dé-binarisateur CABAC HEVC proposée est décrite en langage de description matérielle HDL. Elle est synthétisée à l'aide de l'outil ISE de Xilinx version 14.5 en utilisant la carte de développement FPGA virtex4 (xc4vlx15-12ff-668). De plus, nous avons utilisé le Model Sim pour vérifier l'architecture et confirmer les résultats

obtenus par Matlab. L'estimation de l'énergie consommée par notre architecture effectuée sous XPower de l'ISE est égale à 1.63 W et après la comparaison de ce résultat avec quelques travaux récents nous avons remarqué que nos résultats sont compétitifs en termes de consommation énergétique.

Perspectives :

- On propose de réaliser un binarisateur avec une consommation d'énergie optimisée en introduisant des techniques d'optimisations comme : clock gating et scaling voltages.
- De plus, une augmentation de throughput fait partie de nos perspectives dans le futur par l'utilisation des techniques de pipeline et de parallélisme.

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] Phillipe Bellaïche, « Les secrets de l'image vidéo », Eyrolles 4eme édition, section 1.2, section 1.3.3, section 1.5, section 1.6, section 5.2, section 6.1, 2002.

[4] Document « Le système visuel humain », centre de recherche Inria, Université de Renne (France).

[5] Hongtao Yu, Zhiping Lin, Feng Pan, « Applications and improvement of H.264 in medical video compression », IEEE transactions on circuits and systems, Volume 52, Numéro 12, pp 2707-2716, Décembre 2005.

[6] 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.

[7] http://www.xilinx.coml.

[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] Selmi Mohammed El-Amin, Yala Mohammed, Synthèse et étude comparative sur les méthodes de compression vidéo, thèse de master, Université Abou Bakr Belkaid,Tlemcen, Algérie, 02/07/2017 .

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

[11] T. Wiegand and G. Sullivan: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. (ITU-T Rec H.264|ISO/IEC 14496-10 VC) Mars 2003.

[12] Iain E. Richardson, «The H.264 Advanced Video Compression Standard», NJohn Wiley and Sons, second edition, 2010

[13] Telecommunication standarization sector of ITU: Series H: Audiovisual and multimedia systems, Infrastructure of audio visuel services-Coding of moving video (High efficiency video coding), H.265 04/2013.

[14] “H265.” http ://www.h265.net/.

[15] Olivier Brouard. « Pre-analyse de la vidéo pour un codage adapte. Application au codage de la TVHD en flux H.264 ».

[16] Iain E. Richardson. «The H.264 advanced video compression standard, second edition» WILEY.

[17] International Telecommunication Union Standardization Sector, H.265 (04/2015), 2015.

[18] 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.

[19] Duyen Hai Pham, Jeonhak Moon, Seong soo Lee, Hardware Implementation of HEVC CABAC Binarizer, j. inst-Korean-electr-electron-eng. Vol. 18, No 3,356~361, September 2014, http://dx.doi.org/10.7471/ikeee. 2014-18-3.356.59.

[20] Mathias Wien, «High Efficiency Video Coding, Coding Tools and Specification». Springer, Signals and Communication Technology.

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

[22] Nihel NEJI, Conception conjointe sur FPGA d’un décodeur HD H264/AVC utilisant un codage entropique de type CABAC, thèse de Doctorat, Université de SFAX, Tunis, 17 Décembre 2015.

[23] Atef M, Mohamed salim, B : un nouvel algorithme de compression exploitant le codage arithmétique en lui introduisant de nouveaux paramètres de codage.

[24] D. Marpe, H. Schwarz, and T. Weigand: context based adaptive binary arithmetic coding in the H.264/AVC video compression standard, IEEE transaction on circuits and systems for video technology, 13 (7): p.620-636, 2003.

[25] D. Marpe, H. Schwarz, and T. Wiegand, «Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard», IEEE *Trans. on CSVT*, vol. 13, no. 7, pp. 620–636, July 2003.

[26] Vivienne Sze and Madhukar Budagavi, A comparison of CABAC throughput for HEVC/H.265 vs. AVC/H.264, Massachusetts Institute of Technology, Conference Paper, October 2013.

[27] M. Zhou V. Sze and M. Budagavi, «Parallel Tools in HEVC for High-Throughput Processin», Texas Instruments Inc,12500 TI Blvd., Dallas, TX-75243, Proc. Of SPIE Vol. 8499 849910-1 10/23/2014.

[28] Gary J. Sullivan, J.Ohm, W. Han, and T. Wiegand: Overview of the High Effeciency Video Coding (HEVC) Standard, IEEE Transactions on circuits and systems for video technology y, December, 2012.

[29] Chen. J, Chien. W, Joshi. R, Sole. J, Karczewicz. M : 'Non-ce1 : throughput improvement on cabac coefficients level coding', Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0554, San Jose, 2012.

[30] Recommendation ITU-T H.265, SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services _ Coding of moving Video "High efficiency video coding» Printed in Switzerland Geneva, 04/2013.

[31] Weiwei Shen, Yibo Fan, Yufeng Bai, Leilei Huang, Qing Shang, Cong Liu and Xiaoyang Zeng: A Combined Deblocking Filter and SAO Hardware Architecture for HEVC, IEEE Transactions on Multimedia, 18(6):1-1, june, 2016.

[32] Yassine ARIBA, Jérome Cadieux, Manuel Matlab, Icam de Toulouse, Version 0.1, Départements des génie électrique et mécanique.

[33] Alonso, C.-M.; Ramos, F.-L.-L.; Zatt, B.; Porto, M.; Bampi, S. Low-power HEVC Binarizer Architecture for the CABAC Block targeting UHD Video Processing. In Proceedings of the 30th Symposium on Integrated Circuits and Systems Design, Fortaleza, Brazil, 28 August–1 September 2017.

[34] Ramos, F.-L.-L.; Saggiorato, A.-V.-P.; Zatt, B.; Porto, M.; Bampi, S. Residual Syntax Elements Analysis and Design Targeting High-Throughput HEVC CABAC. IEEE Trans. Circuits Syst. 2019, 67, 475–488.

[35] Pham, D.-H.; Moon, J.; Le, S. Hardware Implementation of HEVC CABAC Binarizer. J. IKEEE 2014, 18, 356–361.

[36] Dinh-Lam Tran, Xuan-Tu Tran, Duy-Hieu Bui and Cong-Kha Pham, An Efficient Hardware Implementation of Residual Data Binarization in HEVC CABAC Encoder, Recue: 10 Mars 2020; Accepter: 21 Avril 2020; Publier: 23 Avril 2020.