REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de **T**echnologie

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

Mémoire de fin d'études de Master

Filière : Télécommunications

Spécialité : Systèmes des télécommunications

Implémentation sur FPGA de la partie sélection et modélisation du contexte pour le décodeur entropique CABAC VVC/H.266

Présentés par :

Chellali Hichem

Chahboub Brahim

Proposés et Dirigés par :

Dr. MENASRI Wahiba

Dr. Bengharbia Bilal

Promotion 2022

Remerciement

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

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

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

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

Dédicace

Je tiens à dédier cet humble travail à :

A ma tendre mère HOURIA et mon très cher père MUSTAPHA

A mes frères : Abd El Kader et Bilal

A mes sœurs : Fadila, Chirifa et Amina.

A mes meilleurs amis :

Abd Elhakim; Amine; Mustapha; Fares; Houssem; Khaled; Khalil; Lahcen; Lakhder; Sohayeb;

Tous ceux qui m'aiment et que j 'aime.

Chellali Hichem.

Dédicace

Je tiens à dédier cet humble travail à :

A ma tendre mère et mon très cher père

A mes frères et mes sœurs

A mes meilleurs amis :

Amine; Faycal; Fares; Khaled; Khalil; Lahcen; Lakhder;

Tous ceux qui m'aiment et que j 'aime.

Chahboub Brahim.

Résumé

Les techniques de compression sont responsables à l'évolution des applications multimédia. VVC/H.266 (Versatile Video Coding) est le dernier standard de compression vidéo finalisée à la fin de 2020. Il adopte différentes techniques avancées, parmi eux on trouve le codage entropique CABAC (Context Adaptive Binary Arithmetic Coding), qu'est considéré comme une partie essentielle dans le codec vidéo VVC. Le décodeur CABAC est basé sur trois modules : BAD (Binary Arithmetic Decoding), CSM (Context Selection and Modeling) et le DB (De-Binarisation), où la CSM est chargée de générer les modes de contexte pour les éléments syntaxiques traités par le mode régulier de la partie BAD. De plus, elle est considérée comme le cerveau de CABAC. Les résultats de l'implémentation logicielle (Matlab) et matérielle (FPGA) montrent son bon fonctionnement et la similarité des résultats obtenus.

Mots Clés : BAD, CABAC, Compression vidéo, CSM, DB, décodage, entropique, FPGA, Matlab, VVC.

Abstract

Compression techniques are responsible for the evolution of multimedia applications. VVC/H.266 (Versatile Video Coding) is the latest video compression standard finalized at the end of 2020. It adopts various advanced techniques, the entropic coding CABAC (Context Adaptive Binary Arithmetic Coding) is one of those techniques, which is considered as an essential part in the video codec VVC. the CABAC decoder is based on three modules: BAD (Binary Arithmetic Decoding), CSM (Context Selection and Modeling) and DB (De-Bigration), where the CSM is responsible for generating context modes for the syntax elements processed by the regular mode of the BAD part. In addition, it is considered as the brain of CABAC. The results of the software (Matlab) and hardware (FPGA) implementation show the good functioning and the accuracy of the obtained results.

Key words: BAD, CABAC, CSM, DB, entropy coding, FPGA, Matlab, Video compression, VVC.

ملخص

تقنيات الضغط مسؤولة عن تطور تطبيقات الوسائط المتعددة. VVC/H.266 (ترميز الفيديو متعدد الاستخدامات) هو أحدث معيار لضـــغط الفيديو تم الانتهاء منه في نهاية عام 2020. يتبنى العديد من التقنيات المتقدمة، من بينها ترميز الإنتروبي CABAC (الترميز الحسابي الثنائي التكيفي السياق)، والذي يعتبر جزءًا أساسيًا في ترميز الفيديو VVC. يعتمد فك تشفير CABAC على ثلاث وحدات: BAD (فك التشفير الحسابي الثنائي)، CSM (اختيار السياق والنمذجة) وDB (فك الثنائيات)، حيث يكون CSM مسؤولاً عن إنشاء أنماط السياق لعناصر التركيب التي تتم معالجتها بواسطة النمط العادي BAD. تُظهر نتائج تنفيذ البرنامج (Matlab) والأجهزة (FPGA) مدى نجاحه ومدى تشابه النتائج.

الكلمات المفتاحية: ضغط الفيديو، ترميز الكون، عناصر التركيب، CABAC، FPGA، Matlab،VVC.

Liste des acronymes

A

AC: Composante Alternative

ALF: Adaptive Loop Filter

ASIC: Application Specifie IntegratedCircuits

AVC: Advanced Vidéo Coding

AVI: Audio Video Interleave

B

BAD: Binary Arithmetic Decoding

BBC R&D: British Broadcasting Corporation Research & Development

Bin : Symboles Binaires

B image**:** Bidirectional image

BM: Bypass Mode

C

CABAC: Context Adavptive Binary Arithmetic Coding

CAVLC: Context Adaptive Variable Length Coding

CD: Compact Disc

CIE: Commission Internationale de l'Eclairage

CTB: Coding Tree Block

CTUs: Coding Tree Unit

Chroma: Chrominance

CMP: Cube Map Projection

CtxInc: Contexte Inclément

CtxIdx: Context Index Adress

CSM: Context Selection and Modelin

CVS: Coded Video Sequence

CU: Unité de Codage

D

DB: Dé-Binarisation

DC: Composante Directe

DCT: Discret Cosine Transform

DST: Discrete Sinus Transform

DVD: Digital Versatile Disc

E

ERP: Equirectangular Projection Format

F

FLV: Format FLash Video

FSM: Finite State Machine

FPGA: Field-Programmable Gate Array

FPS: Frame Per Second

G

GOP: Group Of Pictures

H

H: Height

HD: Haute Definition

HDR: High Dynamic Range

HDTV: High Définition Télévision

HEVC: High Efficiency Video Coding

I

IDE : environnement de développement intégré

IF: Interpolation Filter

I image: Intra image

IntValue: Initial Value

ISE: Integrated Software Environment

ISO : Organisation International de Standardisation

IUT : International Telecommunication Union

J

JEM: Joint Exploration Model

JPEG: Joint Photographic Experts Group

JVT: Joint Video Team

JVET: Joint Video Experts Team

L

LMCS: Luma Mapping with Chroma Scaling

LocNumSig: Number of Significant Location

LocSumAbsPass1: Sum of Absolute values after the first Pass Location

Luma: Luminance

LUT: Look Up Table

LPS: Least Probable Symbol

M

ME: Estimation de Mouvement

MR:Regular Mode

MB: Bypass Mode

MT: Terminate Mode

MTT: Multi-Type Tree

MP3: MPEG Audio Layer 3

MPEG: Moving Pictures Expert Group

MPS: Most Probable Symbol

MTS: sélection de transformations multiples

MV: Vecteurs de Mouvement

N

NTSC: National Television System Committee

P

PAL: Phase Alternating Line

P image: Prédite image

PStateIdx: Probability State Index

PU: Unité de Prédiction

PPS: Picture Parameter Set

Q

QP: Paramètre de Quantification

Qstep: Pas de Quantification

QT: Quadratic Tree

R

RLE: Run Length Encoding

RM: Regular Mode

RTL: Register Transfer Level

RTP/IP: Real-time Transport Protocol Internet Protocol

RVB: Rouge, Vert et Bleu

S

SAO: Sample Adaptive Offset

SDTV: Standard Définition Télévision

SE: Eléments Syntaxiques

SECAM: Système Electronique Couleur Avec Mémoire

SPS: Sequence Parameter Set

T

TM: Terminate Mode

TNT : Turner Network Television

TTM: Time To Market

TU: Terminal Unit

U

UHD: Ultra High Definition

V

VCEG: Video Coding Expert Group

VHDL: Hardware Description Language

VLC: Variable Length Code

VM: Vecteur de Mouvement

VPS: Video Parameter Set

VVC: Versatile Vidéo Coding

W

W: Weight

Liste des figures

Figure I.1 : Représentation d'une séquence vidéo……..……..03

Figure I.2 : Décomposition de l'image en composantes Y Cr Cb……..……..……04

Figure I.3 : Intra prédiction……..……07

Figure I.4 : Modes d'intra prédiction angulaire dans VVC……..……08

Figure I.5 : Compensation de Mouvement……..……08

Figure I.6 : Evolution des codecs vidéo……..……..10

Figure I.7 : Efficacité de H.266 VVC par rapport à H.265 HEVC……..……11

Figure I.8 : Le schéma bloque de la norme VVC/H.266……..……13

Figure I.9 : Fractionnement ternaire et fractionnement binaire interdits dans VVC lorsque la largeur ou la hauteur du bloc de codage luma est de 128 pour permettre l'opération 64 x 64……..……14

Figure I.10 : Exemple de Quad-Tree avec structure de bloc de codage d'arbre multi type Imbriquée……..……..……..……14

Figure I.11 : Exemple simplifié de prédiction avancée du vecteur de mouvement……..……15

Figure I. 12 : Prédiction intra angulaire large……..……...16

Figure II.1 : Auto-information et probabilité de l'auto-information…..……..……..20

Figure II.2 : Construction d'un code de Hufman……..……..…..21

Figure II.3 : Codage arithmétique. (a) Exemple, (b) Algorithme de calcul des intervalles……....22

Figure II.4 : Structure hiérarchique du flux binaire……..……..……..24

Figure II.5 : Architecture du codeur CABAC……..……..……...24

Figure II.6 : Organigramme de CSM (codage/décodage)……..…26

Figure III.1 : Schéma global des outils matériels et logiciels utilisés……....32

Figure III.2 : Organigramme global de la partie CSM dans la norme VVC……..…33

Figure III.3 : Organigramme proposé de clause 1……..……35

Figure III.4 : Programme et simulation du clause 1……..……..36

Figure III.5 : Organigramme proposé de clause 2……..……38

Figure III.6 : Programme et résultat de la simulation pour la clause 2..……..……...38

Figure III.7 : Organigramme proposé de clause 3……..……...39

Figure III.8 : Programme et résultat de la simulation pour la clause 3..……..……..…...40

Figure III.9 : Organigramme proposé de clause 4…..……..…..40

Figure III.10 : Programme et résultat de la simulation pour la clause 4……..……...41

Figure III.11 : Organigramme proposé de clause 5……..……..……..41

Figure III.12 : Programme et résultat de la simulation pour la clause 5……..……....42
Figure III.13 : Organigramme proposé de clause 6……..…….…..….42
Figure III.14 : Programme et résultat de la simulation pour la clause 6……..……..……...43
Figure III.15 : Organigramme proposé de clause 7……...43
Figure III.16 : Programme et résultat de la simulation pour la clause 7…….……..…...44
Figure III.17 : Organigramme proposé de clause 8…….…….…....44
Figure III.18 : Programme et résultat de la simulation pour la clause 8…….……...……..45
Figure III.19 : Programme et résultat de la simulation pour la clause 9…….……..……...45
Figure III.20 : Organigramme proposé de clause 9……..…46
Figure III.21 : Organigramme proposé de clause 10……..……...……47
Figure III.22 : Programme et résultat de la simulation pour la clause 10……..…….47
Figure III.23 : Organigramme proposé de clause 11……..……..……48
Figure III.24 : Programme et résultat de la simulation pour la clause 11……..……..……48
Figure III.25 : Implémentation sous Matlab de la partie CSM……...……...49
Figure III.26 : Résultats de simulation de la partie CSM…….……...51
Figure IV.1 : Architecture proposée de clause 1…….……....53
Figure IV.2 : Schéma RTL et le top level de la clause 1……..……54
Figure IV.3 : Schéma RTL et le top level de la clause 2 …….……...54
Figure IV.4 : Architecture proposée de clause 2…….……..…..55
Figure IV.5 : Architecture proposée de clause 3……..……..….56
Figure IV.6 : Schéma RTL et le top level de la clause 3…….……..……56
Figure IV.7 : Architecture proposée de clause 4…….……..……...57
Figure IV.8 : Schéma RTL et le top level de la clause 4…….…..57
Figure IV.9 : Architecture proposée de clause 5…….……....58
Figure IV.10 : Schéma RTL et le top level de la clause 5……..……...58
Figure IV.11 : Architecture proposée de clause 6……..…59
Figure IV.12 : Schéma RTL et le top level de la clause 6……..……..60
Figure IV.13– Schéma RTLet le top level de la clause 7……..…...……60
Figure IV.14 : Architecture proposée de clause 7 (locNumSig)……..……..61
Figure IV.15 : Architecture proposée de clause 7 (locSumAbsPass1)……...…….62
Figure IV.16 : Architecture proposée de clause 8…….…….63
Figure IV.17 : Schéma RTL et le top level de la clause 8……...……..63

Figure IV.18 : Schéma RTL et le top level de la clause 9……64

Figure IV.19 : Architecture proposée de clause 9 ……65

Figure IV.20 : Architecture proposée de clause 10……66

Figure IV.21 : Schéma RTL et le top level de la clause 10……66

Figure IV.22 : Architecture proposée de clause 11……67

Figure IV.23 : Schéma RTL et le top level de la clause 11……67

Figure IV.24 : Architecture matérielle proposée pour la CSM/CABAC VVC……68

Figure IV.25 : Schémas RTL et le top level de l'architecture globale du CSM/CABAC……68

Figure IV.26 : Comparaison de résultats des implémentations matérielle et logicielle……70

Figure IV.27 : Comparaison de résultats des implémentations matérielle et logicielle des clauses selon leurs tables du contexte……71

Listes des tableaux

Tableau I.1 : Aperçu des outils adoptés dans HEVC et VVC……...……12

Tableau II.1 : Fréquences d'apparitions des lettres……..……21

Tableau II.2 : Code Hufman correspond à chaque lettre…….……...……..21

Tableau II.3 : Codes Golomb pour les valeurs d'indice i = 0 à 9……..……23

Tableau II.4 : Association de ctxIdx et d'éléments syntaxique pour chaque type d'initialisation……..…….27

Tableau IV.1 : Synthèse de la partie CSM/CABAC……...……69

Sommaire

Introduction générale……01

Chapitre I : Généralités sur la compression vidéo

I.1 Introduction……..…….....…….03

I.2 Signaux Vidéo……..……03

I.2.1 Qu'est-ce qu'une vidéo……...03

I.2.2 Espaces de couleurs……...……04

I.2.3 Formats de sous-échantillonnage…….…..05

I.3 Compression Vidéo……..……06

I.3.1 Principes fondamentaux des algorithmes de compression vidéo…….…..06

I.3.2 Estimation et compensation de mouvement……..06

a) Intra prédiction……..07

b) Inter prédiction……..08

I.3.3 Transformation…….……09

I.3.4 Quantification……09

I.4 Codecs Vidéo……...……09

I.4.1 Normes de codage……10

a) MPEG……10

b) AVC/H.264……10

c) HEVC/H.265……11

d) VVC/H.266……11

I.4.2 Comparaison entre les standards H.265 et H.266……..11

I.5 Nouvelle norme VVC/H.266……12

I.5.1 Normalisation et développement……...……12

a) Phase d'exploration (2015–2017)……...12

b) Phase de normalisation (2018-2020)…….….12

I.5.2 Codec Vidéo VVC/H.266……..13

a) Partitionnement de blocs du codec VVC……13

b) Inter prédiction……...15

c) Intra prédiction……...15

d) Transformation et quantification……16
e) Filtrage en boucle……...16
f) Outils de codage vidéo 360°……...17
g) Codage entropique……17
I.6 Conclusion……18
Chapitre II : Codage Entropique CABAC
II.1 Introduction……..……19
II.2 Information et entropie……19
II.2.1 Auto-information……19
II.2.2 Entropie……19
II.3 Codage entropique……..20
II.3.1 Codage de Huffman……20
II.3.2 Codage arithmétique……21
II.3.3 Codage Golomb……...23
II.4 Organisation du flux vidéo……23
II.4.1 Syntaxe de niveau haut……23
II.4.2 Syntaxe de niveau bas……24
II.5 Codage CABAC……24
II.5.1 Binarisation……..25
II.5.2 Codage arithmétique binaire……25
II.5.3 Sélection et modélisation du contexte……..25
II.6 Décodage CABAC pour la norme VVC……26
II.6.1 Sélection et modélisation du contexte……..26
a) Initialisation……..……..……27
b) Calcul de ctxInc, ctxIdx et intValue……..…..30
c) Modélisation de Contexete……30
II.6.2 Décodage arithmétique binaire……30
a) Mode régulier……..….31
b) Mode bypass……31
c) Mode terminal……31

II.6.3 Dé-binarisation……31
II.7 Conclusion…..……31
Chapitre III : Programmation de décodeur CABAC sous Matlab
III.1 Introduction……...32
III.2 Outils utilisés……32
III.3 Algorithme de sélection et de modélisation du contexte (CSM) dans la norme VVC…...33
III.3.1 Initialisation de données…….……34
III.3.2 Sélection de contexte……..34
III.3.2.1 Clause 1…….…...35
III.3.2.2 Clause 2…….…...36
III.3.2.3 Clause 3…….…...39
III.3.2.4 Clause 4…….…...40
III.3.2.5 Clause 5…….…...41
III.3.2.6 Clause 6…….....42
III.3.2.7 Clause 7…….…...43
III.3.2.8 Clause 8……...……44
III.3.2.9 Clause 9…….……...45
III.3.2.10 Clause 10…….…….……..47
III.3.2.11 Clause 11…….……...48
III.3.3 Modélisation du contexte……....49
III.4 Implémentation logicielle et résultat de simulation de la CSM globale……....49
III.4.1 Implémentation sous Matlab de la CSM……49
III.4.2 Simulation de la partie CSM…….…..49
III.5Conclusion…….……50
Chapitre IV : Implémentation du décodeur CABAC sur FPGA
IV.1 Introduction……...52
IV.2 Architectures matérielles proposées pour la partie CSM/CABAC……52
IV.2.1 Architecture proposée pour CSM/clause 1…….……52
IV.2.2 Architecture proposée pour CSM/clause 2…….…54
IV.2.3 Architecture proposée pour CSM/clause 3……..…56

IV.2.4 Architecture proposée pour CSM/clause 4……57
IV.2.5 Architecture proposée pour CSM/clause 5……58
IV.2.6 Architecture proposée pour CSM/clause 6……58
IV.2.7 Architecture proposée pour CSM/clause 7……59
IV.2.8 Architecture proposée pour CSM/clause 8……63
IV.2.9 Architecture proposée pour CSM/clause 9……64
IV.2.10 Architecture proposée pour CSM/clause 10……66
IV.2.11 Architecture proposée pour CSM/clause 11……67
IV.2.12 Architecture Globale proposée du CSM/CABAC……68
IV.3 Synthèse de la partie CSM/CABAC VVC……69
IV.4 Validation des résultats obtenus……70
IV.5 Conclusion……71
Conclusion général……72

Introduction générale

Introduction générale

Durant la dernière décennie, on a vécu une évolution importante dans les applications grand public. Les applications multimédias, en particulier celles en chargent de l'encodage, de la transmission, du stockage et du décodage vidéo, jouent un rôle essentiel dans ce cadre. Selon une étude récente de Cisco, le contenu vidéo occupe aujourd'hui environ 82% du trafic internet mondial [1]. Cela a créé un besoin d'améliorer ou de développer des algorithmes de codage vidéo encore plus efficaces que ceux existants pour limiter l'augmentation du taux de transmission de données et assurer une meilleure qualité de service.

Actuellement, VVC (Versatile Vidéo Coding) [2] est la norme de compression vidéo la plus récente, elle offre des améliorations significatives par rapport à ses prédécesseurs. Le codage entropique CABAC est parmi les techniques utilisées dans la norme VVC, il est constitué de trois étapes : BAD, CSM et DBZ. La CSM est considérée comme une partie critique dans le CABAC. Pour cela, nous nous sommes intéressées à l'implémentation cette partie sur FPGA. Cette partie est utilisée afin d'estimer avec une précision la probabilité requise afin d'obtenir une efficacité de décodage élevée. Notre travail consiste à traiter 74 parmi 109 éléments syntaxiques par le mode régulier qui est basé sur un décodage adapté au contexte. Ce contexte est calculé en passant par deux étapes : sélection de l'adresse du contexte et modélisation du contexte.

L'implémentation sur FPGA de la partie CSM, offre des avantages de haute performance, un coût relativement acceptable (prix de la solution) et un temps de commercialisation court TTM (Time To Market) [3].

Notre 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 vidéo sont données. La deuxième étape consiste à décrire les normes de compression vidéo. Ensuite, nous détaillerons la norme H.266/VVC et nous soulignons l'importance du codage CABAC au niveau du codeur et du décodeur.

Le deuxième chapitre présente une description de quelques notions de base sur le système de communication et la théorie de l'information. Ensuite, nous détaillons le codage entropique et ses types plus précisément le décodeur entropique CABAC. Ainsi, les différentes améliorations du codage CABAC/VVC par rapport à son prédécesseur son mentionnées.

Le troisième chapitre contient une présentation des outils logiciels utilisés dans notre travail, suivi par la représentation des organigrammes proposés des codes de sélection et modélisation du contexte avec leurs simulations sous Matlab.

Le quatrième chapitre est réservé pour les architectures matérielles proposées de la partie sélection et modélisation du contexte du décodeur entropique CABAC dans la norme VVC avec leurs descriptions en langage VHDL. Ensuite, on a assuré une synthèse, simulation et évaluation de la consommation énergétique sur l'ISE et le ModelSim suivi par des commentaires sur les résultats obtenus. Une comparaison est effectuée entre les résultats obtenus sous Matlab avec celles obtenus par ModelSim, afin de valider les implémentations matérielles et logicielles proposées.

Finalement, une conclusion vient clôturer ce manuscrit et présenter les perspectives pour des travaux futurs.

Chapitre 01

Chapitre I : Généralités sur la compression vidéo

I.1 Introduction

Durant la dernière décennie, les domaines d'applications et les besoins en matière de compression vidéo ont évolué. Pour cela, tous les 10 ans, une nouvelle norme est publiée avec une réduction de 50% du débit pour une qualité subjective égale à celle de son prédécesseur. Dans ce chapitre nous nous mettons en évidence les notions de base sur la représentation et les particularités des vidéos. Ensuite, nous présenterons les normes de compression vidéo existantes. Enfin, nous terminerons le chapitre par une étude détaillée de la nouvelle norme de compression vidéo VVC /H.266.

I.2 Signaux Vidéo

I.2.1 Qu'est-ce qu'une vidéo

La vidéo regroupe l'ensemble des techniques permettant l'enregistrement ainsi que la restitution d'images dynamiques, accompagnées ou non de son, sur un support adapté à l'électronique, le mot vidéo vient du latin *vidéo* qui signifie " je vois ". Un flux vidéo est composé d'une succession d'images qui défilent à un rythme fixe pour donner l'illusion du mouvement. On caractérise la fluidité d'une vidéo par sa résolution temporelle, à savoir le nombre d'images par seconde exprimé en *fps* (frames per second) [4], [5]. Généralement, cette résolution temporelle est supérieure à 25 *fps* et peut atteindre120 *fps* pour certaines applications [6].

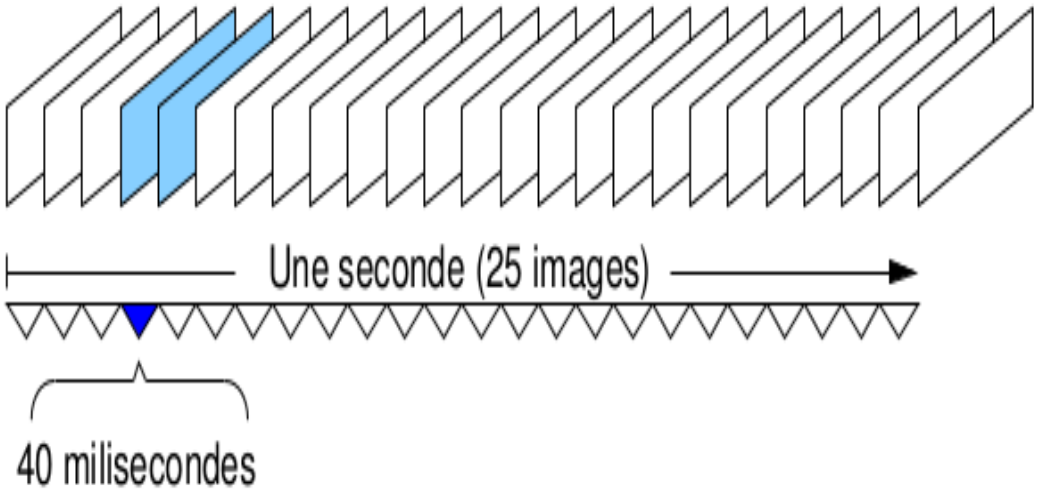


Figure I.1 – Représentation d'une séquence vidéo.

On distingue généralement plusieurs grandes familles d'images dynamiques :

- Le cinéma, consistant à stocker sur une pellicule la succession d'images en négatif. La restitution du film se fait alors grâce à une source lumineuse projetant, à partir d'une copie en positif, les images successives sur un écran.
- La vidéo analogique, représentant l'information comme un flux continu de données analogiques, destiné à être affichées sur un écran de télévision basé sur le principe du balayage. Il existe plusieurs normes pour la vidéo analogique. Les trois principales sont :
  - ✓ PAL (Phase Alternative Line);
  - ✓ NTSC (National Television System Committee);
  - ✓ SECAM (Système Electronique Couleur Avec Mémoire).
- La vidéo numérique, consistant à coder la vidéo en une succession d'images numériques.

I.2.2 Espaces de couleurs

Un téléviseur génère les couleurs à partir d'un mélange additif de sources de couleurs Rouge, Vert et Bleue (RVB). Mais l'image résultante n'est pas naturelle, ce qui explique pourquoi les systèmes à base des couleurs RVB n'ont jamais eu de succès ni à la télévision, ni dans l'animation. Un des concepts spéciaux introduit par la CIE (Commission Internationale de l'Eclairage), était l'isolation de la luminance et de la chrominance.

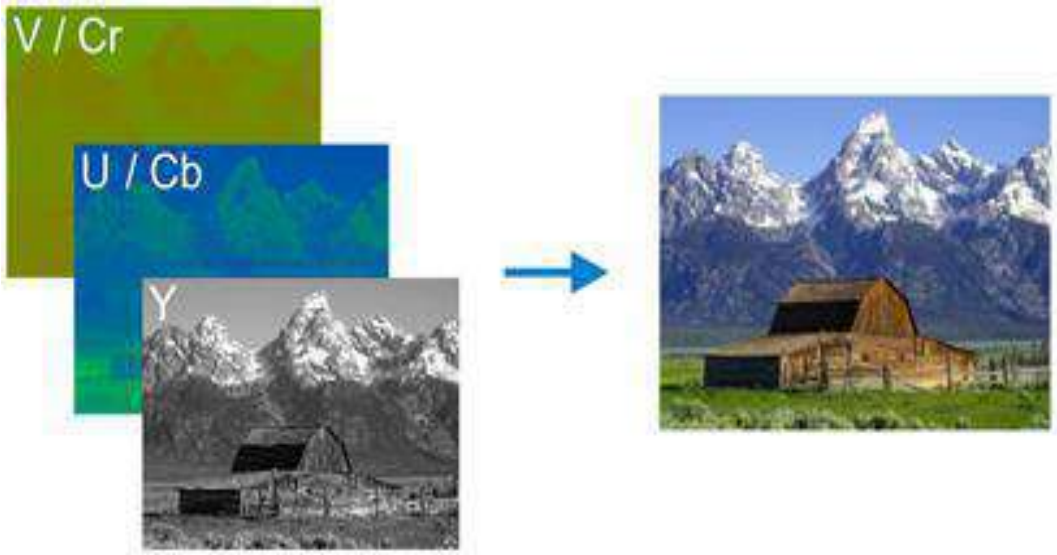


Figure I.2 – Décomposition de l'image en composantes Y C_r C_b.

En utilisant le concept de la CIE comme référence, l'organisme NTSC a adopté un système de transmission de signaux vidéo sous forme de signaux de luminance (Y) et de chrominance (C_r , C_b), plutôt que le format basé sur les composantes RVB des téléviseurs [7].

Cet espace est le plus utilisé car il offre l'avantage de séparer les informations de l'intensité lumineuse et de la couleur dans l'image [6] (voir la figure I.2).

En effet, il existe moins de corrélation entre les composantes Y, C_r et C_b qu'entre les composantes R, V et B. Les valeurs numériques des sources de couleurs RVB doivent être comprises entre 0 et 255. La conversion produise des valeurs de Y comprises entre 0 et 255 et de C_r et C_b comprises entre -128 et 128. Ces deux dernières sont décalées de 128 afin d'obtenir des valeurs toujours positives.

Le format Y C_r C_b demande moins de bande passante à la transmission par rapport au format RVB. La matrice de conversion s'écrit comme suit :

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} = \begin{bmatrix} R \\ V \\ B \end{bmatrix}$$

La matrice de conversion inverse est la suivante :

$$\begin{bmatrix} R \\ V \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 0.5 & 1.772 & 0 \end{bmatrix} = \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

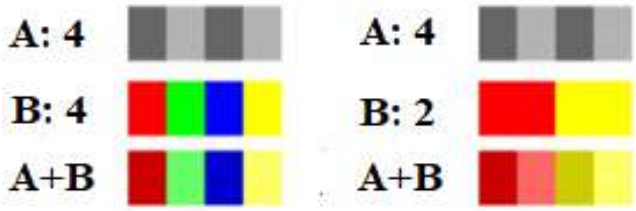
I.2.3 Formats de sous-échantillonnage

La quantité de données pour les vidéos doit rester gérable dans le monde du home cinéma d'aujourd'hui afin qu'une vidéo puisse également tenir sur un Blu-ray ou via les « canaux étroits » d'Internet. C'est là que les ingénieurs des codecs bénéficient de la physiologie de l'œil humain. Parce que nous réagissons plus sensibles aux différences de luminosité (luminance) qu'aux couleurs (chrominance). Le sous-échantillonnage des couleurs signifie que les informations de couleur sont superposées sur une image en niveaux de gris. Les termes 4: 4: 4, 4: 2: 2 et 4: 2: 0 indiquent comment les informations de luminosité et de couleur sont pondérées sur deux lignes et généralement quatre colonnes. Pour ce faire, nous divisons les chiffres dans les lettres : A : B : C.

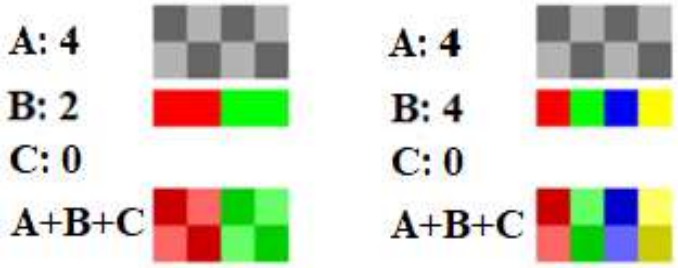
- ***Chiffre A*** : Est généralement numéroté avec "4" une faible puissance de deux. Quatre pixels à l'horizontale forment une série de nuances de gris, que nous appelons la série chroma par souci de simplicité.



- ***Chiffre B*** **:** Indique la quantité d'informations de couleur disponibles pour la série de chrominance horizontale nommée. Un "4" signifie que chaque pixel se voit attribuer une valeur de couleur en plus des informations de luminosité. Un "2" signifie que deux fois moins d'informations de couleur que d'informations de luminosité sont introduites.



- ***Chiffre C*** **:** Indique le nombre de couleurs disponibles pour les pixels de chrominance respectifs de la deuxième ligne. Si le chiffre "C" est spécifié comme zéro, le codec remplit la ligne du bas avec les informations de couleur de la ligne du haut.



I.3 Compression Vidéo

La compression vidéo est une méthode de compression de données, qui consiste à diminuer la quantité de données, en limitant au maximum l'impact sur la qualité visuelle de la vidéo. L'intérêt de la compression vidéo est de diminuer les coûts de stockage et de transmission des fichiers vidéo.

I.3.1 Principes fondamentaux des algorithmes de compression vidéo

Les séquences vidéo contiennent une très grande redondance statistique, autant dans le domaine temporel que dans le domaine spatial. La propriété statistique principale sur laquelle les techniques de compression se fondent, est la corrélation entre pixels. Cette corrélation est à la fois spatiale, les pixels adjacents de l'image courante sont identiques, et temporelle, les pixels des images passées et futures sont aussi particulièrement proches du pixel courant.

I.3.2 Estimation et compensation de mouvement

La compensation de mouvement, est un moyen puissant pour diminuer les redondances temporelles entre les images. Le concept de la compensation du mouvement est basé sur l'estimation du mouvement entre les images vidéo ; si l'ensemble des éléments d'une

scène vidéo sont proches dans l'espace, le mouvement entre trames peut être décrit avec un nombre limité de paramètres (vecteurs de mouvement des pixels). La meilleure prédiction d'un pixel est donnée par la prédiction de mouvement de la trame précédente. Si la corrélation spatiale entre vecteurs de mouvement est assez haute, un vecteur de mouvement pourra représenter un bloc de pixels adjacents. Ces blocs sont fréquemment constitués d'un groupe de 16x16 pixels, et uniquement un vecteur de mouvement est estimé, codé et transmis pour chaque bloc.

a) ***Intra prédiction*** : Utilisée pour réduire les redondances spatiales en exploitant le principe de la corrélation entre les blocs dans une même image vidéo [6]. Trois modes de prédiction sont utilisés dans les codecs vidéo récent comme le montre la figure I.3 :

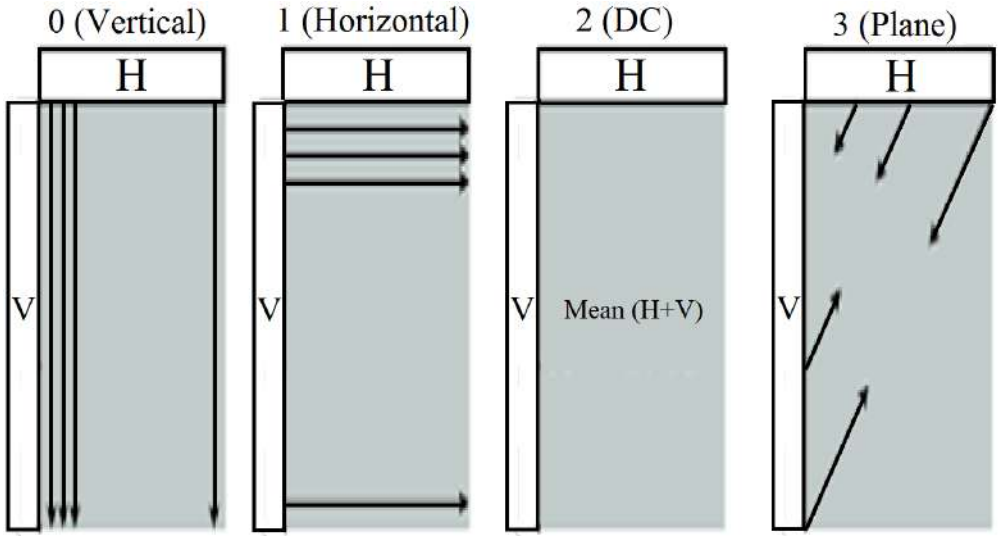


Figure I.3– Intra prédiction.

Le premier, connu sous le nom de mode DC (Composante directe), fait la moyenne des échantillons de référence voisins et utilise cette valeur comme prédiction pour l'ensemble du bloc, pour chaque échantillon. Le mode planaire, par contre, représente les échantillons à prévoir comme plan en utilisant des combinaisons linéaires en fonction de la position des échantillons de référence. La troisième option, les modes angulaires, interpole les échantillons de référence le long d'une direction/angle spécifique. Le mode angulaire vertical, par exemple, ne fait que dupliquer les échantillons de référence ci-dessus le long de chaque colonne. VVC a augmenté le nombre de modes d'intra prédiction directionnelles de 35 à 67 modes (voir la figure I.4). De plus, non seulement le nombre de modes est augmenté, mais de nouvelles approches sont également ajoutées, comme une prédiction intra-image fondée sur une matrice qui utilise l'apprentissage automatique [8].

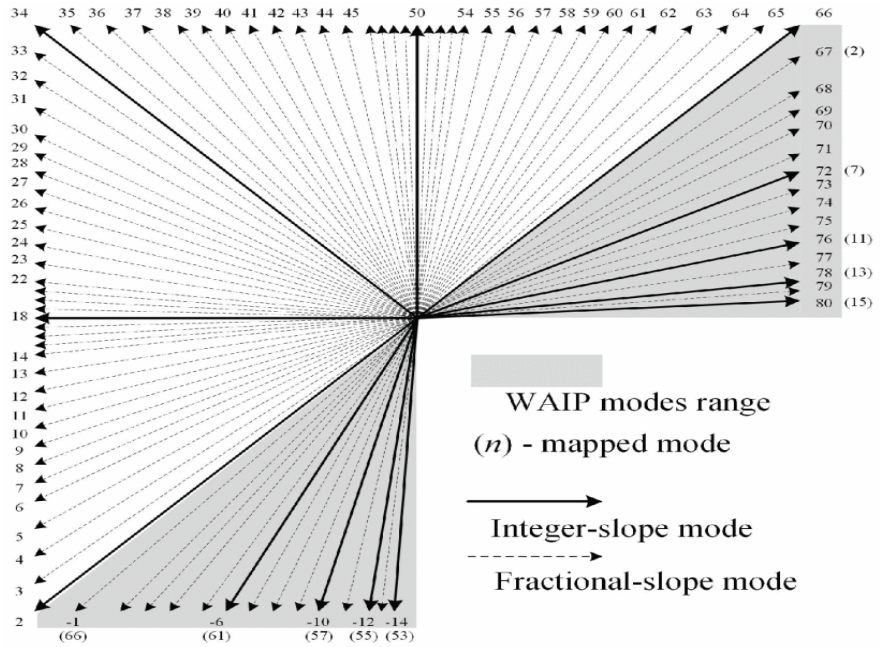


Figure I.4 –Modes d'intra prédiction angulaire dans VVC

b) *Inter prédiction* : Est basée sur la technique d'estimation de mouvement, elle est utilisée pour éliminer les redondances temporelles dans une séquence vidéo où les images successives peuvent se ressembler.

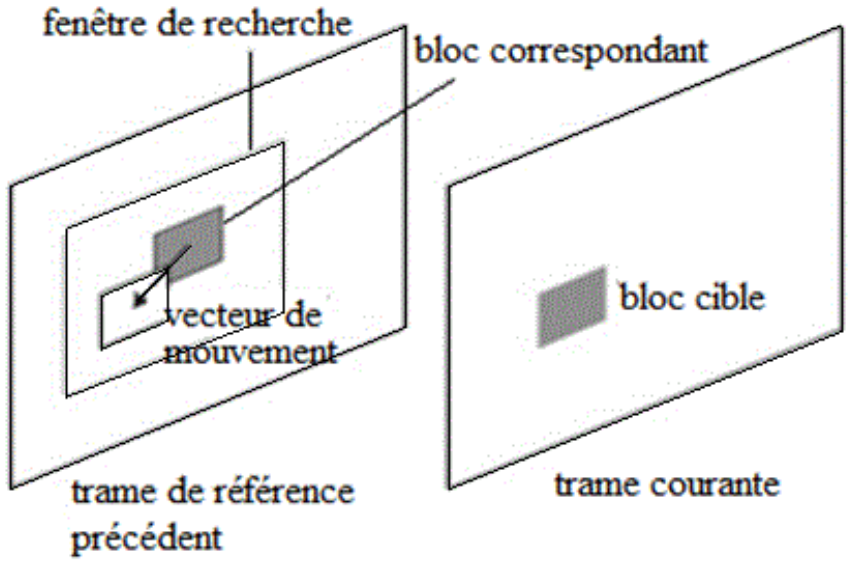


Figure I.5 – Compensation de mouvement.

Comme la figure I.5 nous montre, la position relative de la prédiction par rapport au bloc original est appelée un vecteur de mouvement (VM) et il est transmis au décodeur avec le signal résiduel. En outre, l'utilisation de plusieurs images de référence provenant de diverses positions temporelles permet des structures de prédiction hiérarchiques à l'intérieur d'un groupe d'images (GOP), ce qui augmente encore l'efficacité du codage [9]. VVC, la norme la plus récente, va encore plus loin que le modèle de mouvement translationnel en

rapprochant le mouvement affine et en utilisant une technique d'estimation de mouvement différente pour affiner le mouvement au décodeur.

I.3.3 Transformation

Cette procédure réduit la corrélation d'un signal en le convertissant du domaine spatial en un domaine transformé (généralement un domaine de fréquence) en utilisant une base de transformation appropriée. La prédiction résiduelle (qu'elle provienne de la prédiction inter ou intra-image), c'est-à-dire la différence entre la prédiction et le signal vidéo d'entrée original, est transformée en normes de codage vidéo hybrides. Dans le domaine de la transformation, l'information pertinente est généralement concentrée dans un nombre limité de coefficients. Pour reconstruire les échantillons résiduels au décodeur, une transformation inverse doit être appliquée [9].

I.3.4 Quantification

Chaque échantillon est " pesé ", tout comme un aliment, afin d'en déterminer son poids. En numérique, ce pesage est appelé quantification, il s'effectue, pour reprendre notre analogie, à l'aide d'une balance à deux plateaux : dans un des plateaux se trouve l'échantillon à peser, dans l'autre les poids nécessaires pour trouver l'équilibre. En vidéo, le poids de l'échantillon est la tension du signal électrique à numériser et la balance d'un quantificateur. La quantification permet d'approximer les échantillons du signal transformé afin de réduire la quantité d'informations. Souvent, la précision de cette approximation est contrôlée à l'aide d'un pas de quantification exprimant la plus petite valeur absolue représentable [6].

I.4 Codecs Vidéo

La technologie de compression vidéo de bout en bout implique, un encodeur pour compresser la vidéo à partir de la source dans un flux de bits et un décodeur pour décompresser ce flux de bits à la destination. Le « codec » désigne la combinaison d'une compression et d'une décompression [9]. Un codeur vidéo est une unité vidéo qui convertit une source vidéo analogique en un format numérique à l'aide d'un algorithme de compression standard, comme le H.266, H.265 ou H.264.

La figure I.6, montre la conséquence d'évolution des normes de compression vidéo entre les années 1994 et 2020. En effet, deux organismes de normalisation internationaux, ITU (Union Internationale de Télécommunication) et ISO (Organisation Internationale de Standardisation) travaillent pour améliorer et optimiser ces standards de compression [6].

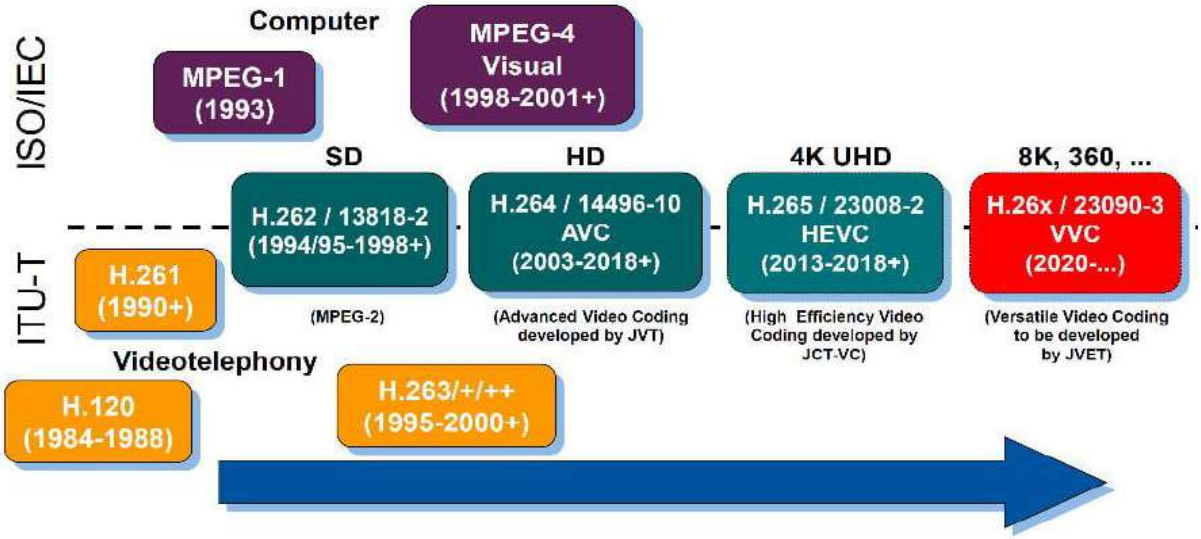


Figure I.6 – Evolution des codecs vidéo [13].

I.4.1 Normes de compression

a) MPEG : (Moving Picture Experts Group) est un groupe d'experts chargé du développement de normes internationales. Ils ont développé les normes suivantes :

- *MPEG-1* : Première norme audio et vidéo utilisé pour les Vidéo CDs, elle comprend le populaire format audio MPEG-1 partie 3 audio couche 3 (MP3).
- *MPEG-2* : Norme applicable au codage de l'audio et la vidéo, mais aussi leur transport pour la télévision numérique par satellite, par câble, terrestre, et (avec quelques restrictions) pour les vidéo disques DVD.
- *MPEG-4* : Utilisé pour compresser les chaînes TNT HD ou encore certains contenus accessibles en streaming. En comparaison à ses prédécesseurs, il est plus efficace, avec une qualité d'image équivalente le poids des fichiers obtenu est moindre.
- *MPEG-7* : Norme de description pour la recherche du contenu multimédia.
- *MPEG-21* : Norme proposant une architecture pour l'interopérabilité et l'utilisation simple de l'ensemble des contenus multimédia.
- *MPEG-A* : Tourné vers les applications multimédia. En cours de standardisation.

b) AVC/H.264 : AVC (Advanced Video Coding) C'est le produit d'un effort de partenariat entre VCEG de l'ITU et MPEG de l'ISO, connu sous le nom H. 264/AVC est par conséquent adapté à une très grande variété de réseaux et de dispositifs, par exemple, pour la diffusion de la télévision, le stockage HD DVD et Blu-ray, le streaming RTP/IP, et des dispositifs de téléphonie propre à l'UIT.

c) HEVC/H.265 : (High Efficiency Video coding) c'est un 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 équivalente.

d) VVC/H.266 : (Versatile Video Coding) est un codec vidéo publié le 6 juillet 2020 par le JVET (Joint Video Experts Team). Il a été développé par l'Institut Fraunhofer pour les télécommunications en partenariat avec des industriels [11]. Il est annoncé comme offrant un taux de compression deux fois supérieur au H.265/HEVC pour une qualité identique [12]. VVC est adapté aux vidéos en HDR (High Dynamic Range) et aux vidéos filmées à 360° degrés [13]. Les fondateurs de cette norme donnent un exemple. Il faut 10 Go pour stocker 90 min de vidéo 4K UHD en H.265, en H.266, la même vidéo ne prend plus que 5 Go.

I.4.2 Comparaison entre les standards H.265 et H.266

VVC/H.266 peut réduire les besoins de données d'environ 50% du débit binaire par rapport au HEVC sans diminuer la qualité visuelle. Plus précisément, BBC R&D (British Broadcasting Corporation Research & Development) a constaté que VVC a réalisé 27 % de meilleurs résultats que HEVC pour les séquences HD et 35 % pour les séquences UHD.

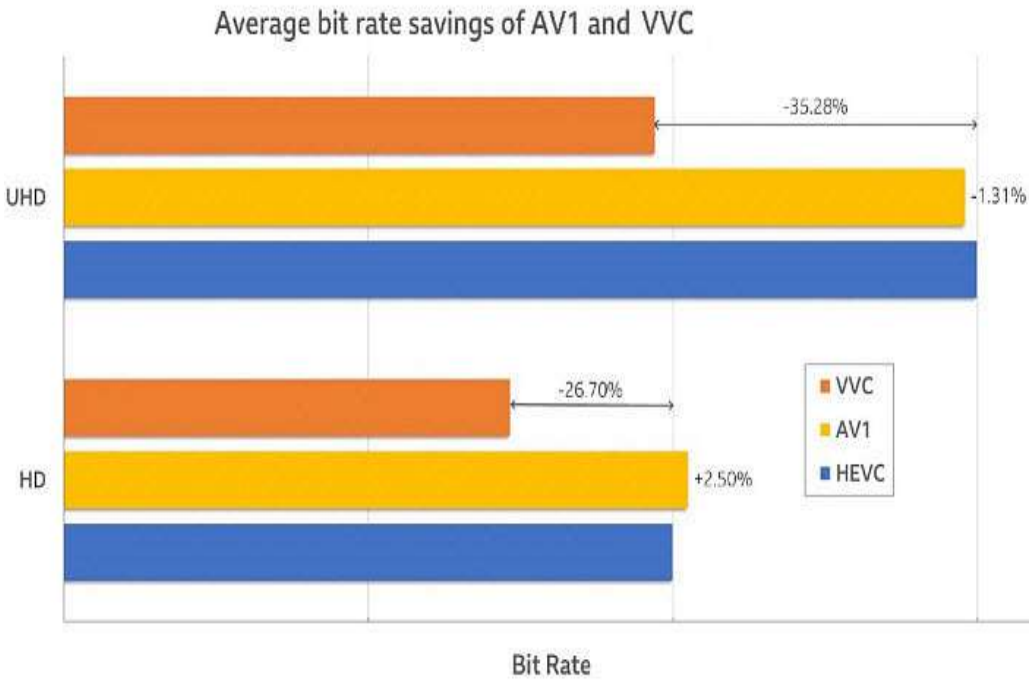


Figure I.7 – Efficacité de H.266 VVC par rapport à H.265 HEVC

Le tableau I.1 souligne les principales caractéristiques de la norme de compression vidéo H.266/VVC et la norme antérieure H.265/HEVC.

Tableau I.1 –Aperçu des outils adoptés dans HEVC et VVC [9].

Élément constitutif du système hybride	HEVC	VVC
Partitionnement de blocs	•Taille CTU max 64x64	•Taille CTU max 128x128 •Taille du pipeline 64x64
Inter Prediction	• 1/4 pixel luma MV • 2 filtres d'interpolation pour MC	• MV luma 1/16 pixel • 2 filtres d'interpolation pour MC • DMVR (amélioration de la MV côté décodeur)
Intra Prediction	• 35 prédicteurs	• 81 modes : 65 angulaires + planaires + DC + Sub
Transformation	• transformations carrées basées sur la DCT • 1 transformation DST4x4	• DCT2 (jusqu'à 64x64) + MTS (<=32x32, DST7, DCT8, H/V indépendant) • Codage du coefficient : conception HEVC + DQ, masquage du panneau
Codage entropique	• CABAC	• CABAC + quantification scalaire dépendante
Filtrage en boucle	• Déblocage • SAO	• Déblocage • SAO • ALF • LMCS
Codage vidéo à 360 degrés	/	• MV Wrap-Around
	/	• Limites virtuelles pour le filtrage en boucle

I.5 Nouvelle norme VVC/H.266

I.5.1 Normalisation et développement

Le développement de VVC peut être divisé en deux phases, comme illustré ci-dessous:

a) Phase d'exploration (2015–2017) : La nécessité d'une compression encore plus efficace que la norme HEVC a motivé ITU-T VCEG et ISO/IEC MPEG à unir leurs forces en octobre 2015 pour explorer de nouvelles technologies de codage au sein d'une nouvelle équipe appelée JVET (Joint Video Exploration Team). À la fin de 2017, le JVET avait développé la base de code [14] du logiciel JEM (Joint Exploration Model), qui présentait des économies pouvant atteindre 30 % par rapport au HEVC.

b) Phase de normalisation (2018-2020) : Le développement de VVC a commencé en avril 2018 avec la première version du document de spécification et du logiciel de modèle de test, après un examen des idées les plus performantes parmi toutes les soumissions. Cette

première version comprenait que le Quad-Tree avancé avec le partitionnement de blocs Multi-Type Tree (QT+MTT), qui a été identifié comme un élément commun dans toutes les propositions, surtout que sa mise en œuvre aurait un impact majeur sur la conception de tous les autres outils de codage par blocs [9].

I.5.2 Codec Vidéo VVC/H.266

VVC utilise une architecture de codage vidéo hybride à base de blocs semblable à celles de ses prédécesseurs. Malgré que la même architecture soit utilisée, de nouveaux outils sont intégrés dans chaque bloc de construction fondamental pour améliorer la compression. La figure I.8 ci-dessous représente le schéma bloque de la norme VVC/H.266.

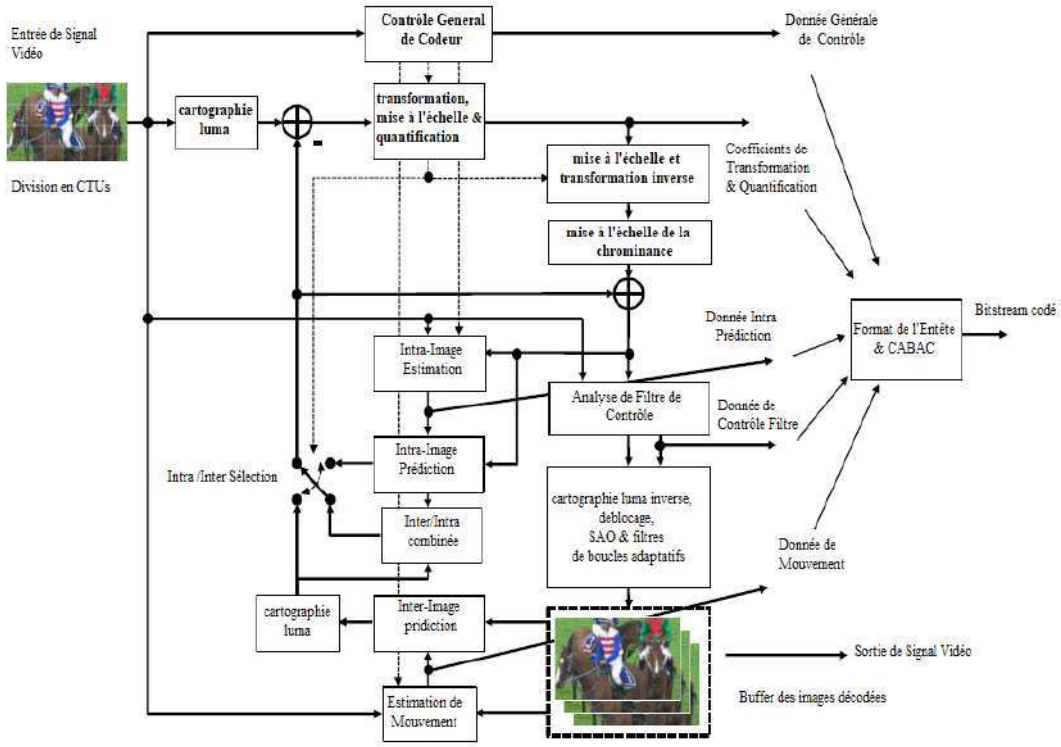


Figure I.8 – Le schéma bloque de la norme VVC/H.266.

a) Partitionnement de blocs du codec VVC : La norme VVC adopte le schéma de partitionnement QT+MTT, qui utilise des divisions quaternaires suivis de divisions binaires et ternaires, permettant un partitionnement de CU (Unité de Codage) plus flexible. En raison des divisions binaires et ternaires, les formes rectangulaires en PU (Unité de Prédiction) sont remplacées par des formes rectangulaires en CU. Les TUs (Unité de Transformation) multiples dans des CUs se produire à partir une division implicite des CUs qui sont plus longues que la longueur de transformation maximale et des CU comportant des sous-partitions internes [9]. En outre, la taille de la CTU (Coding Tree Unit) a été étendue à 128 échantillons de luminance.

- **Arbre chromatique séparée :** Le schéma d'arbre de codage dans VVC permet à la luminance et la chrominance d'avoir leurs propres structures d'arbre de partitionnement séparées. Les CTBs (Coding Tree Block) de luminance et de chrominance dans une CTU doivent avoir la même structure d'arborescence de codage pour les slices codées inter-images.

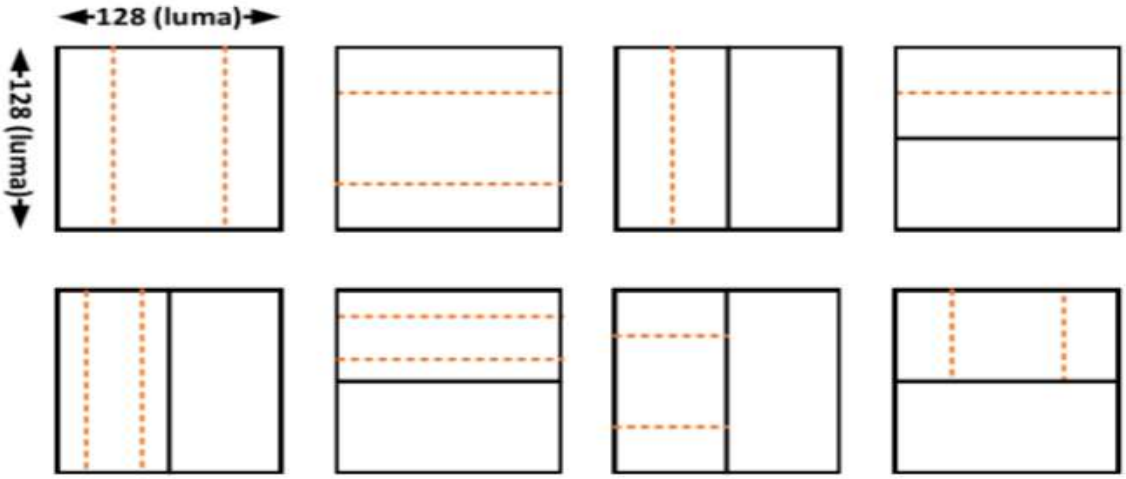


Figure I.9 – Fractionnement ternaire et fractionnement binaire interdits dans VVC lorsque la largeur ou la hauteur du bloc de codage luma est de 128 pour permettre l'opération 64 x 64 [9].

- **Codage de l'arbre quaternaire avec une arbre multi type** :

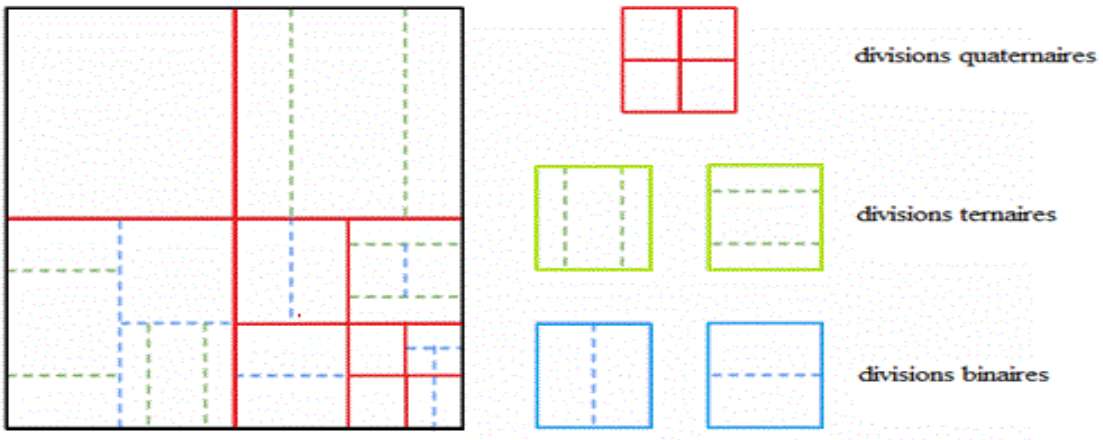


Figure I.10 – Exemple de Quad-Tree avec structure de bloc de codage d'arbre Multi-type imbriquée [9].

Une structure Quad-Tree est utilisée pour partitionner une CTU, ensuite, une structure arborescente multi type peut fractionner les nœuds foliaires Quad-Tree. Les nœuds foliaires multi types sont appelés CUs, et cette segmentation est utilisée pour la prédiction et le traitement de transformation sans partitionnement supplémentaire jusqu'à ce que la CU soit trop grande pour la longueur de transformation maximale. Cela signifie que dans la plupart des situations, la CU, la PU ou la TU ont la même taille de bloc dans la structure de codage QT + MTT.

b) Inter prédiction

L'inter prédiction de l'image exploite la redondance temporelle dans une séquence vidéo. Elle prédit les échantillons d'une unité de codage à partir d'échantillons d'images de référence précédemment codées. Le processus consistant à trouver le bloc le plus similaire dans les images de référence est appelé estimation de mouvement (ME).

Elle produit un ou plusieurs vecteurs de mouvement (MV), chacun composé d'une composante verticale et d'une composante horizontale, représentant la translation de la CU du cadre de référence au cadre actuel [10].

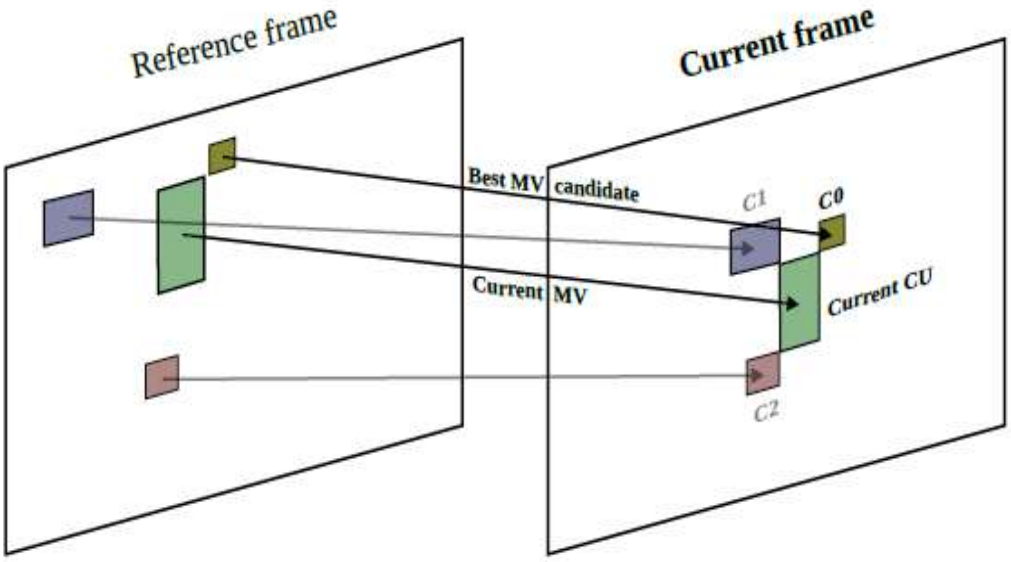


Figure I.11– Exemple simplifié de prédiction avancée du vecteur de mouvement.

c) Intra prédiction

Les échantillons de référence dans les blocs voisins à gauche et au-dessus du bloc présent, qui ont déjà été décodés (avant le filtrage en boucle) dans la même image, sont utilisés pour prédire les échantillons d'un bloc intra-image codé. VVC étend les possibilités en fournissant des outils supplémentaires, comme les suivants :

- **93 angles d'intra prédiction d'image directionnelle :** VVC double la précision angulaire pour fournir 65 modes angulaires directionnels dans la même plage angulaire, en plus de la composante continue DC. La figure I.12 illustre un exemple de modes de prédiction « wide angle » pour un bloc de 8x4 (WxH). Le nombre total de modes d'intra prédiction d'image pour toute taille de bloc est vaut 67. L'inclusion de grands angles pour les blocs non carrés porte le nombre total de directions supportées à 93, et donc le nombre total de modes à 95.

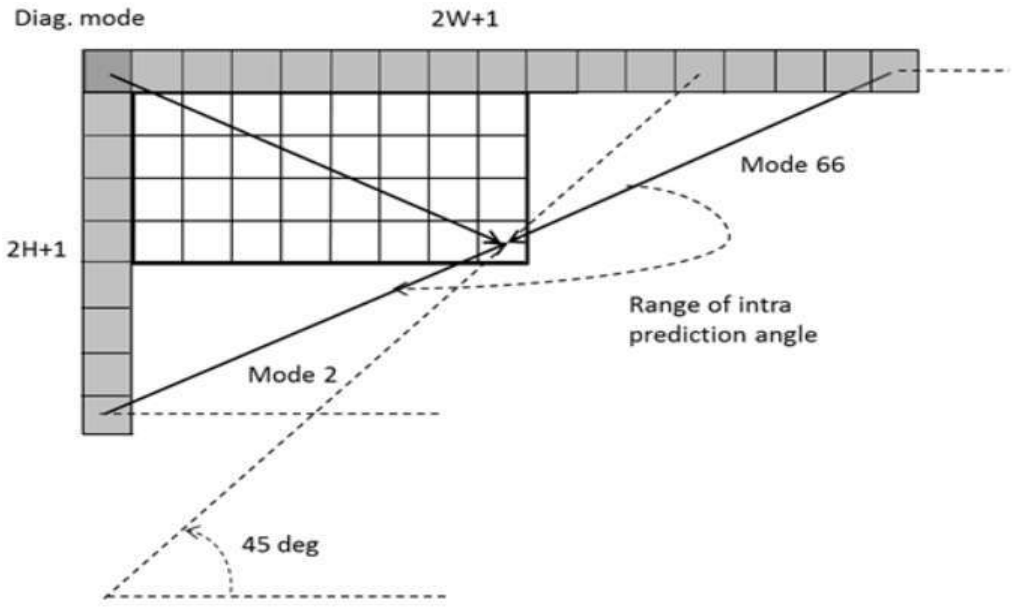


Figure I. 12 – Prédiction intra angulaire large [14].

- **Deux ensembles de filtres d'interpolation à quatre prises** : Les échantillons de prédiction placés à des emplacements d'échantillons fractionnés pour les modes angulaires sont générés avec des seuils de fréquences variés et une précision d'échantillon de 1/32. L'IF est choisi en fonction de la taille du bloc ainsi que de la distance angulaire entre les modes horizontal et vertical.

d) Transformation et quantification

VVC fournit des techniques supplémentaires pour produire un plus grand compactage d'énergie des signaux résiduels et minimiser l'erreur de quantification des coefficients transformés. La taille de transformation maximale a été augmentée à 64x64 afin d'améliorer la compression d'énergie pour les signaux résiduels dans les grandes régions lisses.

e) Filtrage en boucle

Pour modifier le domaine de représentation et diminuer les différentes formes de parasites, une opération de remappage et trois filtres en boucle peuvent être appliqués successivement à l'image reconstruite. Pour commencer, une nouvelle technique d'échantillonnage connue sous le nom de LMCS (Luma Mapping with Chroma Scaling) est réalisée. Les parasites de blocage sont ensuite réduits à l'aide d'un filtre de déblocage. L'image débloqué est ensuite soumise à un filtre SAO (Sample Adaptive Offset) pour réduire les problèmes de résonance et de gammes. Enfin, en réduisant les distorsions éventuelles supplémentaires produites par les opérations de quantification et de transformation, un ALF

(Adaptive Loop Filter) peut être utilisé. La conception d'ALF pour la norme VVC est divisée en deux étapes [9] :

- ALF pour les données luma et chroma en utilisant l'adaptation de filtre par bloc ;
- ALF pour les échantillons de chroma.

f) Outils de codage vidéo 360°

Un autre objectif de VVC est de rendre le codage vidéo efficace le plus possible, cela inclut la vidéo 360°, qui est généralement encodée comme une image 2D créée par le mappage de projection à partir d'une sphère 3D. L'ERP (Equirectangular Projection Format) est un exemple d'une telle cartographie, dans laquelle la sphère est projetée sur une image rectangulaire avec certaines distorsions géométriques, notamment aux pôles. Le CMP (Cube Map Projection) est une autre technique de cartographie, dans laquelle la sphère est projetée sur les six faces d'un cube, puis compactée en une seule image [9].

Pour améliorer l'efficacité du codage des images vidéo, VVC a introduit la capacité d'identifier ces formats ainsi que les deux techniques suivantes :

- **Enveloppement MV (Motion Vector) :** Lorsqu'un MV est situé à l'extérieur de la région codée, le bouclage MV permet aux échantillons de prédiction de « boucler » à partir de la limite gauche ou droite opposée. En raison de la nature 360° de la cartographie de projection, qui pourrait entraîner un objet en mouvement qui est partiellement à la limite gauche et en partie à la limite droite d'une image.
- **Limites virtuelles pour le filtrage en boucle :** Cette technique empêche le filtrage en boucle sur certaines limites « virtuelles », comme celles qui correspondent aux limites de la face du CMP (Clear Motion Picture) dans les images du CMP, plutôt que les limites de slice ou de tile. Les emplacements de ces limites sont signalés au niveau du CVS (Coded Video Sequence).

g) Codage entropique

Le CABAC (Context-Adaptive Binary Arithmetic Coding) est utilisé comme méthode de codage d'entropie unique dans VVC, de la même façon que le HEVC. Néanmoins, la conception du CABAC dans VVC présente un certain nombre d'améliorations de l'efficacité du codage par rapport à l'architecture HEVC. Cette section passe en revue les modifications apportées aux deux principaux éléments du codage d'entropie : le moteur CABAC et le codage du coefficient de transformation [9].

- **CABAC avec estimation de probabilité multi hypothèse :** Le concept principal du CABAC utilisé dans le HEVC est conservé dans VVC, mais le codeur arithmétique binaire est remplacé par un modèle de probabilité de mise à jour multi hypothèse. Ce modèle est basé sur deux estimations de probabilité *pStateIdx0* et *pStateIdx1* qui sont associées à chaque modèle de contexte et mises à jour séparément avec des taux d'adaptation variables.
- **Amélioration du codage du coefficient de transformation :** VVC a proposé un modèle de probabilité plus sophistiqué pour les composantes de syntaxe liées aux valeurs absolues des niveaux de coefficient de transformation, basées sur les valeurs des niveaux absolus. Deux endroits voisins à droite, deux en dessous, et un en dessous de droite par rapport au point de balayage actuel composent le modèle.

I.6 Conclusion

Dans ce chapitre, on a introduit les notions de base relatives à la vidéo numérique et sa compression. Ensuite, les normes de compression vidéo récentes ont été présentées. Les techniques associées à la norme de compression VVC ont été données, à savoir la transformation, la quantification, l'intra et l'inter prédiction, les filtres en boucle et le codage entropique CABAC. Enfin, une étude détaillée du CABAC/VVC fera l'objet du chapitre prochain.

Chapitre 02

Chapitre II : Codage Entropique CABAC

II.1 Introduction

De même que ces prédécesseurs, le VVC utilise le codage arithmétique adaptatif binaire CABAC pour le codage entropique de tous les éléments de syntaxe de niveau bas sauf qu'il introduit un nouveau moteur pour le mode de codage régulier qui est conçu pour être plus flexible et efficace. Les codeurs entropiques adaptés aux contextes utilisés dans les normes H.265 et H.266 ont été présentés. Nous terminons ce chapitre par une présentation de l'algorithme de décodage CABAC VVC et les améliorations appliquées pour lui rendre plus efficace.

II.2 Information et entropie

L'entropie quantifie la valeur de l'information contenue dans un message, il s'agit d'une mesure de l'incertitude dans une variable aléatoire. Le concept de production d'une mesure quantitative de l'information a été créé par Shannon en 1948 [15], s'appuyant sur les travaux de Nyquist (1924) et Hartley (1928). Comprendre les bases de la théorie de l'information est important dans le contexte de l'image et de la compression vidéo, car il est largement utilisé pour l'encodage de symboles sans perte après transformation et quantification. Généralement, les coefficients de transformation quantifiés sont balayés et encodés au cours de l'exécution. Les symboles représentant des séries de zéros et des valeurs de coefficients non nuls sont codés à l'aide de codes à longueur variable, d'une manière qui reflète leur contenu d'information. De cette façon, une réduction sans perte du débit binaire peut être obtenue.

Dans les sections II.1.1 et II.1.2, nous examinons quelques fondements de la théorie de l'information.

II.2.1 Auto-information

L'information est utilisée pour réduire l'incertitude au sujet d'un événement ou d'un signal. Shannon [15] a défini l'auto-information d'un événement A, qui pourrait par exemple comprendre un ensemble de résultats d'une expérience, comme suit :

$$i(A) = \log_2 (1 / P(A)) = -\log_2 (P(A)) \qquad \text{II.1}$$

II.2.2 Entropie

Mathématiquement l'entropie est une fonction qui définit la quantité d'informations moyenne contenue ou délivrée par une source d'information. La théorie de l'information

caractérise le degré de désorganisation ou d'imprédictibilité du contenu en information d'une source.

$$H(X) = -\sum_{i=1}^{n} p(xi) \log2\ (p(xi)) \qquad \text{II.2}$$

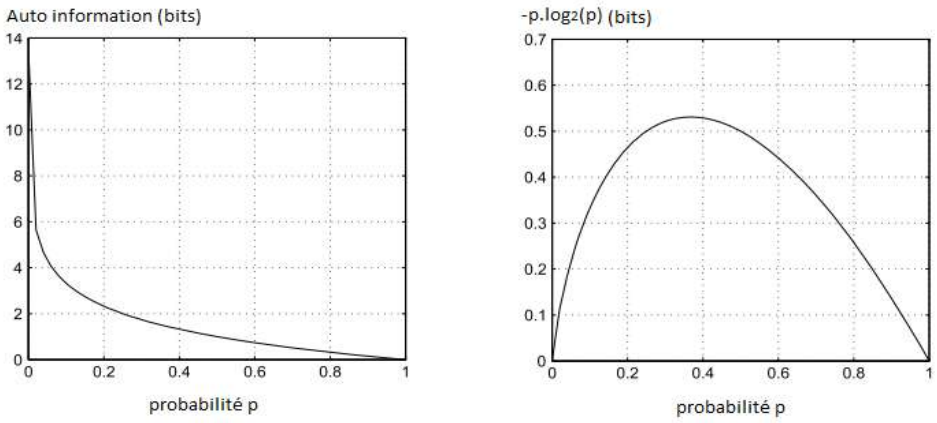


Figure II.1 – Auto-information et probabilité de l'auto-information.

II.3 Codage entropique

C'est une méthode de codage de source sans pertes, dont le but est de transformer la représentation d'une source de données pour sa compression ou sa transmission sur un canal de communication. Dans les sous sections suivantes nous allons détailler quelques types de codage entropique.

II.3.1 Codage de Huffman

David Huffman a proposé en 1952 une méthode statistique qui permet d'attribuer un mot de code binaire aux différents symboles à compresser (pixels ou caractères par exemple). Les symboles les plus fréquents (qui apparaissent le plus souvent) sont codés avec de petits mots de code, tandis que les symboles les plus rares reçoivent de plus longs codes binaires. On parle de codage à longueur variable VLC (Variable Code Length) préfixé pour désigner ce type de codage car aucun code n'est le préfixe d'un autre. Ainsi, la suite finale de mots codés à longueurs variables sera en moyenne plus petite qu'avec un codage de taille constante.

Le codeur de Huffman crée un arbre ordonné à partir de tous les symboles et de leur fréquence d'apparition. Les branches sont construites récursivement en partant des symboles les moins fréquents. La construction de l'arbre se fait en ordonnant dans un premier temps les symboles par fréquence d'apparition. Successivement les deux symboles de plus faible fréquence d'apparition sont retirés de la liste et rattachés à un nœud dont le poids vaut la somme des fréquences des deux symboles. Le symbole de plus faible poids est affecté à la branche 1,

l'autre à la branche 0 et ainsi de suite en considérant chaque nœud formé comme un nouveau symbole, jusqu'à obtenir un seul nœud parent appelé racine. Le code de chaque symbole correspond à la suite des codes le long du chemin allant de ce caractère à la racine. Ainsi, plus le symbole est "profond" dans l'arbre, plus le mot de code sera long. Soit la phrase suivante : " MARCHANTE_COMM_E".

Tableau II.1 – Fréquences d'apparitions des lettres.

M	A	C	E	_	H	O	N	T	R
3	2	2	2	2	1	1	1	1	1

Voici l'arbre correspondant :

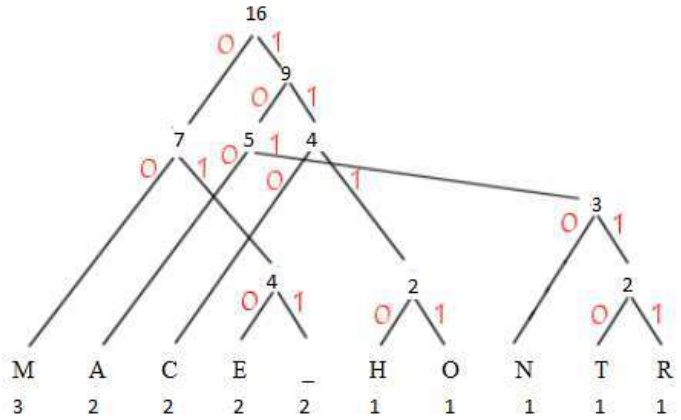


Figure II.2 – Construction d'un code de Hufman.

Les codes correspondants à chaque caractère sont tels que les codes des caractères les plus fréquents sont courts et ceux correspondant aux symboles les moins fréquents sont longs :

Tableau II.2 – Code Hufman correspond à chaque lettre.

M	A	C	E	_	H	O	N	T	R
00	100	110	010	011	1110	1111	1010	10110	10111

II.3.2 Codage arithmétique

Bien que le codage de Huffman soit encore largement utilisé dans des normes telles que JPEG, une méthode alternative plus flexible a gagné en popularité ces dernières années. Les variantes sur le codage arithmétique sont employées comme mécanisme de codage d'entropie préféré dans la plupart des normes de codage vidéo avec le codage arithmétique adaptatif binaire CABAC basé sur le contexte formant maintenant la base de AVC, HEVC et VVC [16].

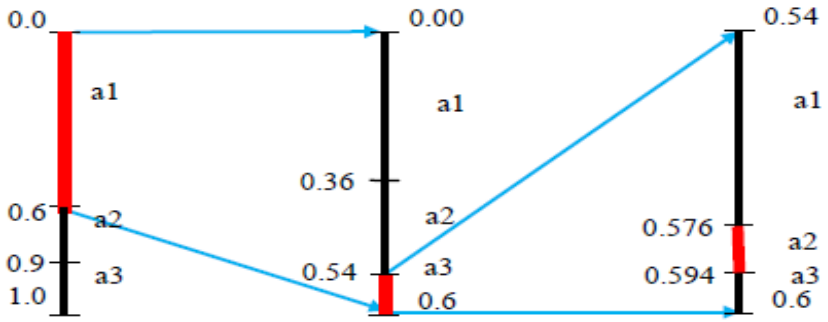
Un encodeur arithmétique représente une chaîne de symboles d'entrée en tant que fraction binaire. Dans le codage arithmétique, une plage d'encodage semi-ouverte [0,1) est

subdivisée en fonction des probabilités des symboles encodés. Le premier symbole encodé se situera dans une subdivision donnée ; cette subdivision forme un nouvel intervalle qui est ensuite subdivisé de la même manière pour l'encodage du deuxième symbole. Ce processus est répété itérativement jusqu'à ce que tous les symboles de la séquence aient été encodés.

Considérons un exemple d'une source de symboles A = {a1, a2, a3} de probabilité = {0.6, 0.3, 0.1} respectivement. On souhaite encoder la séquence S = a1a3a2 (figure II.3.a) :

- L'intervalle I = [0.0, 1.0] est d'abord subdivisé en I1 = [0.0, 0.6[, I2 = [0.6, 0.9[et I3 = [0.9, 1.0[.
- On sélectionne l'intervalle I1 puisque le premier symbole de S est a1. L'intervalle courant devient donc I = [0.0, 0.6[.
- On répète ces opérations pour le deuxième symbole a3 en utilisant l'algorithme de calcul des deux limites d'intervalle [Binf(S), Bsup(S) [(Figure II.3.b). 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[parce que le dernier symbole à coder est a2.
- Notons que la taille de l'intervalle final est égale à la probabilité de la séquence à émettre et 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.

(a)



(b)

Algorithme de calcul de l'intervalle
Initialisation
Binf = 0.0 ; Bsup = 1.0 ;
a1 symbole à coder ;
Taille = Bsup – Binf ;
Bsup = Binf + taille * Bsup (a1) ;
Binf = Binf + taille * Binf (a1) ;

Figure II.3 – Codage arithmétique. (a) Exemple, (b) Algorithme de calcul des intervalles [6].

II.3.3 Codage Golomb

La famille Golomb comprend Unary, Golomb, Golomb–Rice et les codes exponentiels Golomb [17, 18]. Ce sont des codes de longueur variable qui sont générés par une correspondance fixe entre un index et un mot de code. De la même manière qu'avec les codes Huffman, des mots de code plus courts peuvent être assignés à des symboles avec des probabilités plus élevées et vice versa. Les codes Golomb ont l'avantage de pouvoir être simplement encodés et décodés algorithmiquement sans avoir besoin de tables de recherche. Ils possèdent également la propriété que, pour toute distribution géométrique, il est possible de trouver un encodage qui est un code préfixé optimal.

Tableau II.3 – Codes Golomb pour les valeurs d'indice i = 0 à 9.

i	Unary	Golomb–Rice(m=4)	Exp-Golomb
0	1	1|00	1|
1	01	1|01	01|0
2	001	1|10	01|1
3	0001	1|11	001|00
4	00001	01|00	001|01
5	000001	01|01	001|10
6	0000001	01|10	001|11
8	000000001	01|11	0001|000
9	0000000001	001|00	0001|001

Des méthodes adaptatives ont ainsi été développées et exploitées dans des normes telles que la forme sans perte de JPEG (Joint Photographic Experts Group) – JPEG-LS (Lossless JPEG) [19].

II.4 Organisation du flux vidéo

En général, la syntaxe est classée en deux niveaux : les syntaxes de niveau haut et les syntaxes de niveau bas.

II.4.1 Syntaxe de niveau haut

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. La figure II.4 montre qu'une séquence vidéo est composée de plusieurs images, et que chaque image est constituée de plusieurs slices.

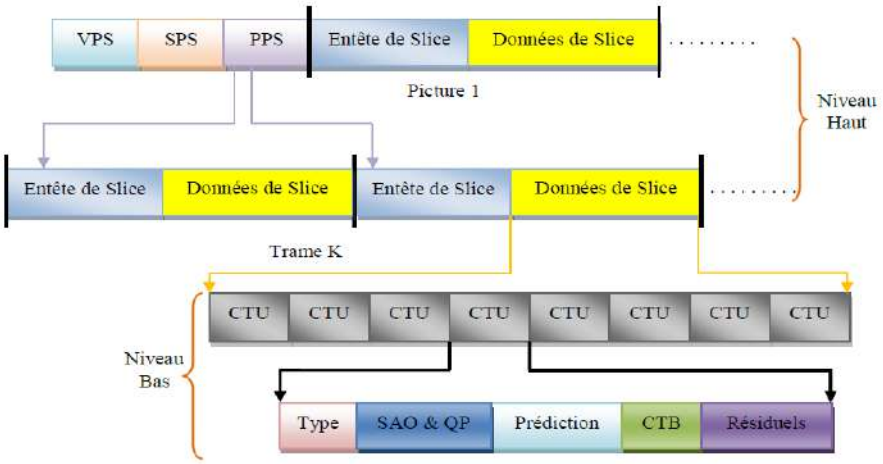


Figure II.4 – Structure hiérarchique du flux binaire [6].

II.4.2 Syntaxe de niveau bas

Le niveau bas est réservé pour le champ de données de chaque slice. Selon la figure II.4, chaque slice est constitué d'un certain nombre de CTU (Coding tree unit) et chaque CTU est codée par les groupes d'éléments syntaxiques suivant :

- Type de la CTU ;
- Données de filtrage SAO et paramètre de quantification QP ;
- Prédiction intra et inter ;
- Paramètres du bloc de codage CTB (Coding tree block) et les données résiduelles.

II.5 Codage CABAC

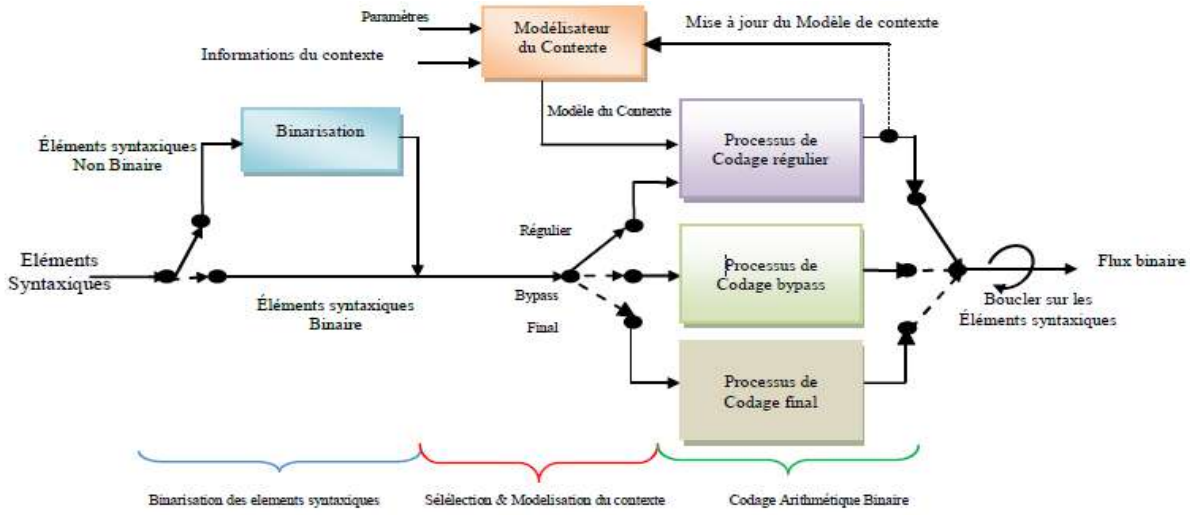


Figure II.5– Architecture du codeur CABAC.

Le codage arithmétique adaptatif binaire contextuel (CABAC) est une méthode très sophistiquée d'application du codage arithmétique pour la compression entropique de données d'éléments syntaxiques dans un encodeur vidéo (voir la figure II.5).

Il faut préciser ici, que la dernière technologie de compression vidéo VVC contient la version la plus efficace de la technique CABAC, qui a été développée au cours des dernières années grâce à de nombreuses améliorations des versions du CABAC utilisées auparavant dans les technologies AVC et HEVC [20]. Le codage CABAC repose sur les étapes suivantes :

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

II.5.1 Binarisation

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) [6]. Différents codes de binarisations sont utilisés comme : les codes simples, les codes spécifiques et les codes combinés.

II.5.2 Codage arithmétique binaire

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

- Mode normal ou régulier RM (Regular Mode) : qui utilise le modèle du contexte ;
- Mode bypass BM (Bypass Mode) : basé sur la probabilité équivalente ;
- Mode terminal TM (Terminate Mode) : appelé à la fin de processus de codage ou de décodage de chaque CTU.

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 [6].

II.5.3 Sélection et modélisation du contexte

Une des propriétés les plus importantes du codage arithmétique est la possibilité d'utiliser une interface entre la modélisation du contexte et le codage [21]. La modélisation permet d'assigner un modèle de contexte aux symboles qui seront ensuite utilisés par le codeur

arithmétique afin de produire une représentation de ces symboles en adéquation avec ce modèle du contexte. Par ailleurs, le choix du modèle de contexte est lié aux données suivantes :

- Valeurs des symboles voisins déjà codés ;
- Taille des blocs à coder.

Il est donc important d'avoir des modèles statistiques bien adaptés aux données et de les garder à jour tout au long du codage. Le même processus de fonctionnement de cet étage est appliqué au niveau du décodage [6].

II.6 Décodage CABAC pour la norme VVC

Le décodage arithmétique binaire d'un élément syntaxique se déroule en trois étapes principales [6].

- Sélection et modélisation du contexte ;
- Décodage arithmétique binaire pour les trois modes : Bypass, régulier et terminal ;
- Dé-binarisation ;

II.6.1 Sélection et modélisation du contexte

Il est à noter que cette partie est appliquée uniquement pour les éléments syntaxiques décodés avec le mode régulier. Cette modélisation permet d'assigner un modèle de distribution de probabilité aux symboles (*bins*) qui seront ensuite utilisés par le décodeur arithmétique binaire afin de produire une représentation de ces symboles en adéquation avec ce modèle de distribution [22].

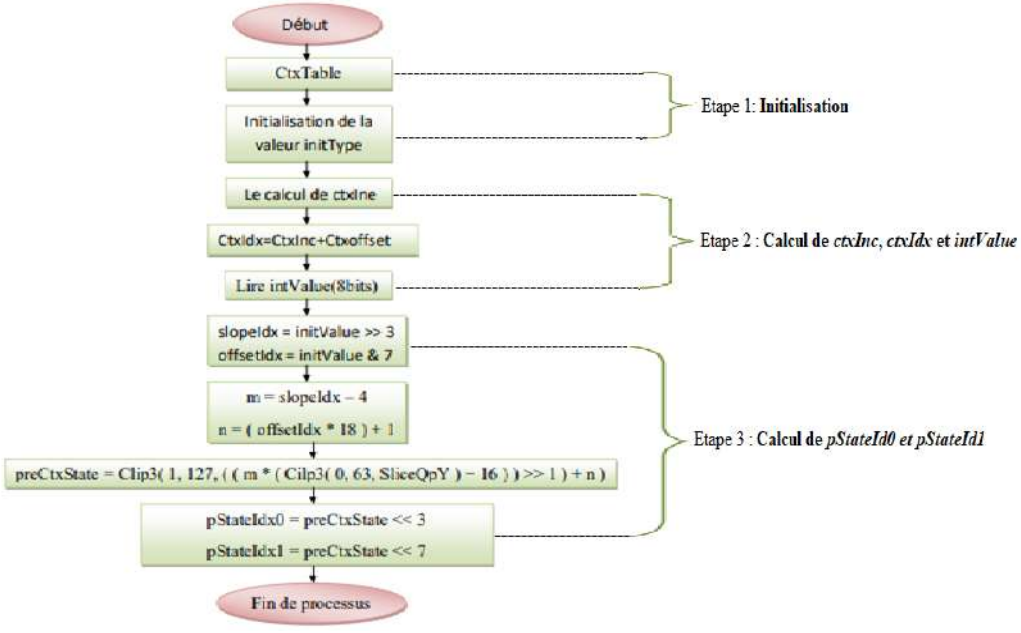


Figure II.6 – Organigramme de CSM (codage/décodage).

Dans le décodage en mode régulier, chaque élément syntaxique est décodé à l'aide d'un modèle de contexte sélectionné et modélisé en passant par les trois étapes suivantes (figure II.6).

a) Initialisation

L'initialisation de la table de contexte *ctxTable* est la première étape dans le processus de décodage [6]. Les sorties de ce processus sont les variables de contexte CABAC initialisées et indexées par *ctxTable* et *ctxIdx*.

Le tableau II.3 répertorie la plage de valeurs *ctxIdx* pour laquelle l'initialisation est nécessaire pour chacun des trois types d'initialisation, spécifiés par la variable *initType*. Il donne également le numéro de table qui inclut les valeurs de *initValue* nécessaires à l'initialisation pour chaque valeur de *ctxIdx*. Pour les types P et B, la dérivation du type *initType* dépend de la valeur de l'élément syntaxique *sh_cabac_init_flag* [23].

Tableau II.4 –Association de *ctxIdx* et d'éléments syntaxique pour chaque type d'initialisation [23].

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
coding_tree_unit( )	alf_ctb_flag[ ][ ][ ]	Table 52	0..8	9..17	18..26
	alf_use_aps_flag	Table 53	0	1	2
	alf_ctb_filter_alt_idx[ ][ ][ ]	Table 56	0..1	2..3	4..5
	alf_ctb_cc_cb_idc[ ][ ]	Table 54	0..2	3..5	6..8
	alf_ctb_cc_cr_idc[ ][ ]	Table 55	0..2	3..5	6..8
sao( )	sao_merge_left_flag sao_merge_up_flag	Table 57	0	1	2
	sao_type_idx_luma sao_type_idx_chroma	Table 58	0	1	2
coding_tree( )	split_cu_flag	Table 59	0..8	9..17	18..26
	split_qt_flag	Table 60	0..5	6..11	12..17
	mtt_split_cu_vertical_flag	Table 61	0..4	5..9	10..14
	mtt_split_cu_binary_flag	Table 62	0..3	4..7	8..11
	non_inter_flag	Table 63		0..1	2..3

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
coding_unit( )	cu_skip_flag[ ][ ]	Table 64	0..2	3..5	6..8
	pred_mode_ibc_flag	Table 65	0..2	3..5	6..8
	pred_mode_flag	Table 66		0..1	2..3
	pred_mode_plt_flag	Table 67	0	1	2
	cu_act_enabled_flag	Table 68	0	1	2
	intra_bdpcm_luma_flag	Table 69	0	1	2
	intra_bdpcm_luma_dir_flag	Table 70	0	1	2
	intra_mip_flag	Table 71	0..3	4..7	8..11
	intra_luma_ref_idx	Table 72	0..1	2..3	4..5
	intra_subpartitions_mode_flag	Table 73	0	1	2
	intra_subpartitions_split_flag	Table 74	0	1	2
	intra_luma_mpm_flag[ ][ ]	Table 75	0	1	2
	intra_luma_not_planar_flag[ ][ ]	Table 76	0..1	2..3	4..5
	intra_bdpcm_chroma_flag	Table 77	0	1	2
	intra_bdpcm_chroma_dir_flag	Table 78	0	1	2
	cclm_mode_flag	Table 79	0	1	2
	cclm_mode_idx	Table 80	0	1	2
	intra_chroma_pred_mode	Table 81	0	1	2
	general_merge_flag[ ][ ]	Table 82	0	1	2
	inter_pred_idc[x0][y0]	Table 83		0..5	6..11
	inter_affine_flag[ ][ ]	Table 84		0..2	3..5
	cu_affine_type_flag[ ][ ]	Table 85		0	1
	sym_mvd_flag[ ][ ]	Table 86		0	1
	ref_idx_l0[ ][ ], ref_idx_l1[ ][ ]	Table 87		0..1	2..3
	mvp_l0_flag[ ][ ], mvp_l1_flag[ ][ ]	Table 88	0	1	2
	amvr_flag[ ][ ]	Table 89		0..1	2..3
	amvr_precision_idx[ ][ ]	Table 90	0..2	3..5	6..8
	bcw_idx[ ][ ]	Table 91		0	1
	cu_coded_flag	Table 92	0	1	2
	cu_sbt_flag	Table 93		0..1	2..3
	cu_sbt_quad_flag	Table 94		0	1
	cu_sbt_horizontal_flag	Table 95		0..2	3..5

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
	cu_sbt_pos_flag	Table 96		0	1
	lfnst_idx	Table 97	0..2	3..5	6..8
	mts_idx	Table 98	0..3	4..7	8..11
palette_coding( )	copy_above_palette_indices_flag	Table 99	0	1	2
	palette_transpose_flag	Table 100	0	1	2
	run_copy_flag	Table 101	0..7	8..15	16..23
merge_data( )	regular_merge_flag[ ][ ]	Table 102		0..1	2..3
	mmvd_merge_flag[ ][ ]	Table 103		0	1
	mmvd_cand_flag[ ][ ]	Table 104		0	1
	mmvd_distance_idx[ ][ ]	Table 105		0	1
	ciip_flag[ ][ ]	Table 106		0	1
	merge_subblock_flag[ ][ ]	Table 107		0..2	3..5
	merge_subblock_idx[ ][ ]	Table 108		0	1
	merge_idx[ ][ ] merge_gpm_idx0[ ][ ] merge_gpm_idx1[ ][ ]	Table 109	0	1	2
mvd_coding( )	abs_mvd_greater0_flag[ ]	Table 110	0	1	2
	abs_mvd_greater1_flag[ ]	Table 111	0	1	2
transform_unit( )	tu_y_coded_flag[ ][ ]	Table 112	0..3	4..7	8..11
	tu_cb_coded_flag[ ][ ]	Table 113	0..1	2..3	4..5
	tu_cr_coded_flag[ ][ ]	Table 114	0..2	3..5	6..8
	cu_qp_delta_abs	Table 115	0..1	2..3	4..5
	cu_chroma_qp_offset_flag	Table 116	0	1	2
	cu_chroma_qp_offset_idx	Table 117	0	1	2
	transform_skip_flag[ ][ ][ ]	Table 118	0..1	2..3	4..5
	tu_joint_cbcr_residual_flag[ ][ ]	Table 119	0..2	3..5	6..8
residual_coding( )	last_sig_coeff_x_prefix	Table 120	0..22	23..45	46..68
	last_sig_coeff_y_prefix	Table 121	0..22	23..45	46..68
	sb_coded_flag[ ][ ]	Table 122	0..6	7..13	14..20
	sig_coeff_flag[ ][ ]	Table 123	0..62	63..125	126..188
	par_level_flag[ ]	Table 124	0..32	33..65	66..98
	abs_level_gtx_flag[ ][ ]	Table 125	0..71	72..143	144..215
	coeff_sign_flag[ ]	Table 126	0..5	6..11	12..17

b) Calcul de *ctxInc*, *ctxIdx* et *intValue*

Avant d'entamer le calcul de la variable *ctxInc*, il faut savoir quel est le type de l'élément syntaxique à traiter pour avoir la possibilité de collecter seulement les informations nécessaires pour cet élément syntaxique comme la position du bloc dans la CTU, la position du pixel dans le sous-bloc, les informations de voisinage. La variable *ctxInc* est déduite du tableau II.4, sauf pour les éléments syntaxiques suivants qui ont une méthode de dérivation spécifique :

- *pred_mode_ibc_flag* ; *alf_ctb_flag[ ][ ][ ] ;*
- *ntra_mip_flag* ; *alf_ctb_cc_cb_idc[ ][ ];*
- *run_copy_flag* ; *alf_ctb_cc_cr_idc[ ][ ];*
- *merge_subblock_flag[ ][ ]* ; *split_cu_flag;*
- *inter_affine_flag[ ][ ]* ; *split_qt_flag;*
- *tu_y_coded_flag[ ][ ]* ; *mtt_split_cu_vertical_flag;*
- *sb_coded_flag[ ][ ]* ; *non_inter_flag;*
- *sig_coeff_flag[ ][ ]* ; *cu_skip_flag[ ][ ];*
- *par_level_flag[ ]* ; *pred_mode_flag;*
- *abs_level_gtx_flag[ ][ ];*
- *coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 1 && n <= lastScanPosPass1 && !sh_ts_residual_coding_disabled_flag.*

Après le calcul de la variable *ctxInc*, l'adresse *ctxIdx* est calculée par l'équation (II.3). Où la valeur de *ctxIdxOffset* dépend du type d'élément syntaxique et aussi du type du slice.

$$ctxIdx = ctxInc + ctxIdxOffset \qquad \text{II.3}$$

c) Modélisation de Contexete

Cette étape est la partie finale de calcul du contexte, à l'entrée du processus on a la valeur *intValue* et à la sortie on obtient la variable *pStateIdx0* et *pStateIdx1*.

II.6.2 Décodage arithmétique binaire

Le décodeur arithmétique décode chaque bin selon trois modes : régulier, bypass et terminal. Dans le mode régulier le décodage du *bin* est effectué selon le modèle de probabilité sélectionné. Les deux modes de décodage bypass et terminal sont exécutés sans modélisation du contexte [6].

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

b) Mode bypass : Ce mode de décodage est la partie la moins complexe dans le décodage arithmétique binaire. Il est basé sur une probabilité équivalente.

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

II.6.3 Dé-binarisation

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

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

II.7 Conclusion

Dans ce chapitre nous avons détaillé le principe de fonctionnement de l'encodeur/décodeur CABAC/VVC. En raison de ces performances élevées et des fonctionnalités supportées, ce codeur entropique est adopté dans le dernier standard vidéo VVC avec une version améliorée. L'implémentation logicielle de la partie CSM fera l'objet du chapitre 03.

Chapitre 03

Chapitre III : Programmation de décodeur CABAC sous Matlab

III.1 Introduction

L'implémentation d'un algorithme de décodage CSM/CABAC dans la norme VVC est réalisée dans notre travail avec deux aspects : matériel et logiciel. Dans ce chapitre, nous présenterons les outils utilisés ainsi que l'implémentation de cette partie sous Matlab avec les résultats de simulation. Il est à noter que la CSM est constituée de 11 clauses pour générer le modèle du contexte.

III.2 Outils utilisés

Dans une première partie, la simulation logiciel du CSM de la norme VVC est réalisée par le fameux MATLAB et dans une deuxième partie, on a utilisé le langage VHDL (Hardware Description Language) pour la description matérielle sur une plateforme FPGA (Field Programmable Gate Array). Nous avons utilisé aussi, le logiciel ISE14.7 (Integrated Software Environment) pour la description de nos algorithmes et le logiciel ModelSim10.6d pour la simulation et la validation des architectures proposées. La consommation de l'énergie est visionnée en utilisant l'analyseur XPower de l'ISE.

Une vue d'ensemble des outils matériel et logiciel utilisés pour notre implémentation est illustrée sur la figure III.1 ci-dessous.



Figure III.1– Schéma global des outils matériels et logiciels

III.3 Algorithme de sélection et de modélisation du contexte (CSM) dans la norme VVC

L'algorithme de décodage pour la partie CSM est donné sur la figure suivante. Il est divisé en trois parties : initialisation de données, sélection du contexte et modélisation du contexte.

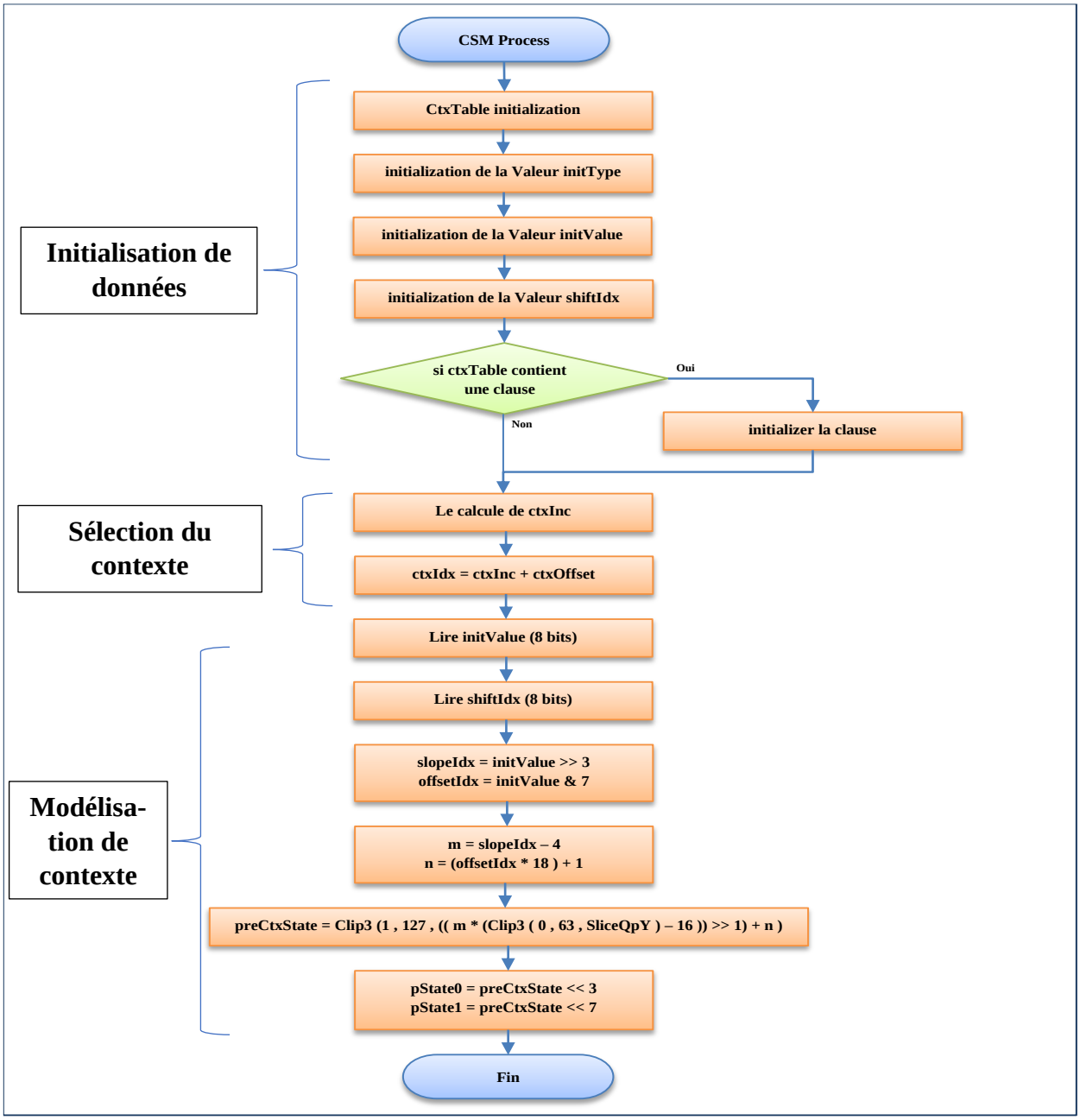


Figure III.2– Organigramme global de la partie CSM dans la norme VVC.

III.3.1 Initialisation de données

Toutes les données et les variables utilisées dans la partie CSM sont initialisées dans cette partie comme : les tables de contexte *CtxTable*, la variable *intType* qui définit le type de slice ou de l'image à traiter I (Intra image), B (Bidirectional image) ou P (Prédite image), ainsi que les variables *intValue* et *ShiftIdx* qui constituent la table du contexte.

III.3.2 Sélection de contexte

La partie sélection du contexte consiste à calculer les variables *intValue* et *ShiftIdx* pour chaque élément syntaxique traiter par le mode régulier en utilisant les clauses suivantes qui sont spécifiées par la norme VVC [23].

Il est à noter que dans la partie CSM/VVC, 75 SEs (éléments syntaxiques) parmi 109 SEs sont passées par le mode régulier et utilisées pour coder et décoder l'information vidéo. Parmi ces 75 SEs, on trouve 20 éléments qui ont un calcul spécifique du contexte, basé sur les clauses :

- *alf_ctb_flag[ ][ ][ ]* ; *par_level_flag[ ];*
- *alf_ctb_cc_cb_idc[ ][ ]* ; *pred_mode_flag*
- *alf_ctb_cc_cr_idc[ ][ ]* ; *pred_mode_ibc_flag;*
- *split_cu_flag* ; *ntra_mip_flag;*
- *split_qt_flag* ; *run_copy_flag;*
- *mtt_split_cu_vertical_flag* ; *merge_subblock_flag[ ][ ];*
- *non_inter_flag* ; *inter_affine_flag[ ][ ];*
- *cu_skip_flag[ ][ ]* ; *tu_y_coded_flag[ ][ ];*
- *abs_level_gtx_flag[ ][ ];* ; *sb_coded_flag[ ][ ];*
- *sig_coeff_flag[ ][ ];*
- *coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 1 && n <= lastScanPosPass1 &&!sh_ts_residual_coding_disabled_flag.*

D'après le standard de la norme VVC [23], les 11 clauses existantes sont rédigées par les développeurs afin de calculer le *ctxInc* (contexte inclément), ce dernier sera utilisé par la suite afin de calculer le *ctxIdx*.

III.3.2.1 Clause 1

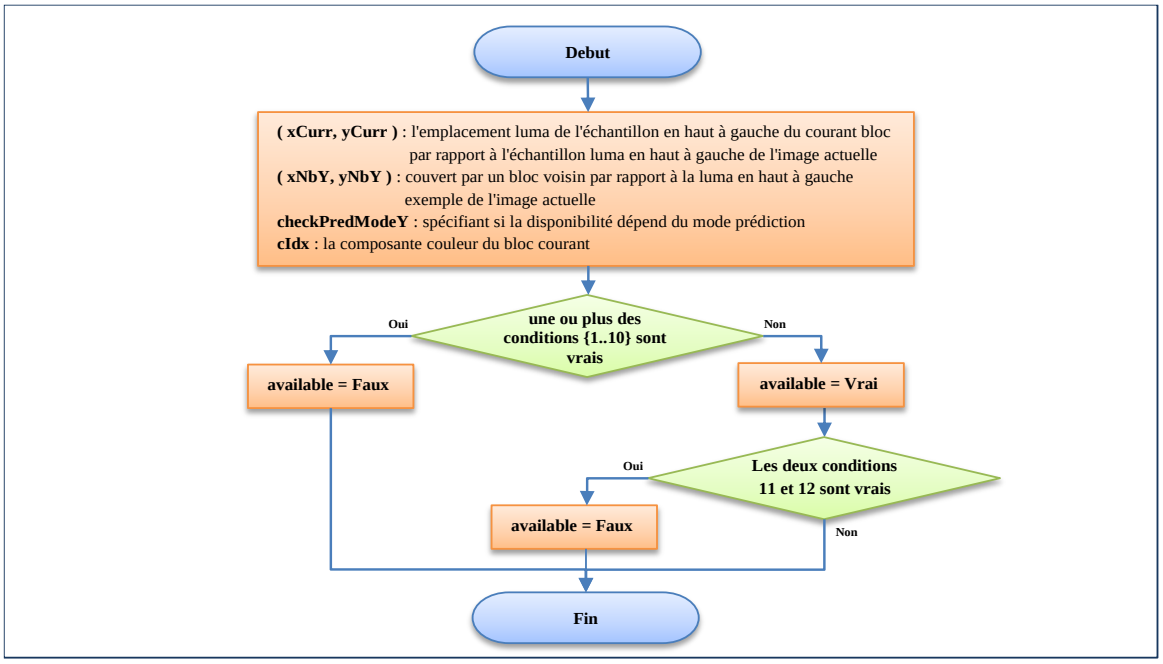


Figure III.3– Organigramme proposé de clause 1.

Cette clause permet de calculer la probabilité *AvialableN* qui sera utiliser dans la clause 2 et 3. Le résultat de calcul est obtenu selon les conditions suivantes :

- **Condition 01:** *xNbY* < 0.
- **Condition 02:** *yNbY* < 0.
- **Condition 03:** *xNbY* >= *pps_pic_width_in_luma_samples*.
- **Condition 04:** *yNbY* >= *pps_pic_height_in_luma_samples*.
- **Condition 05:** (*xNbY* >> *CtbLog2SizeY*) > (*xCurr* >> *CtbLog2SizeY*) et (*yNbY* >> *CtbLog2SizeY*) >= (*yCurr* >> *CtbLog2SizeY*).
- **Condition 06:** (*yNbY* >> *CtbLog2SizeY*) >= (*yCurr* >> *CtbLog2SizeY*) +1.
- **Condition 07:** est-ce *Available*[*cIdx*][*xNbY*][*yNbY*] = Faux.
- **Condition 08:** Le bloc voisin est contenu dans une tranche différente de celle du courant bloc.
- **Condition 09:** Le bloc voisin est contenu dans une tuile différente de celle de l'actuel bloc.
- **Condition 10:** *sps_entropy_coding_sync_enabled_flag* = 1 et (*xNbY* >> *CtbLog2SizeY*) >= (*xCurr* >> *CtbLog2SizeY*) + 1.
- **Condition 11:** *checkPredModeY* = Vrai.
- **Condition 12:** *CuPredMode*[0][*xNbY*][*yNbY*] ~= *CuPredMode*[0][*xCurr*][*yCurr*].

Le programme et le résultat de simulation obtenu pour le processus de calcul de la clause 1 est présenté sur la figure III.4. Sur la gauche, on trouve le programme Matlab et sur la droite, on trouve le résultat de calcul des variables *availableA* et *availableL*.

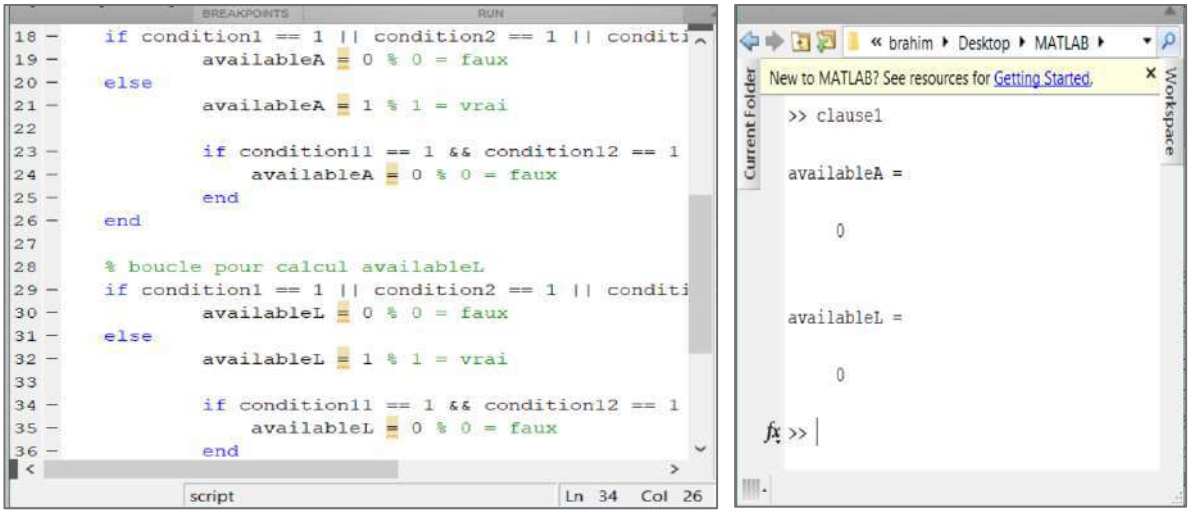
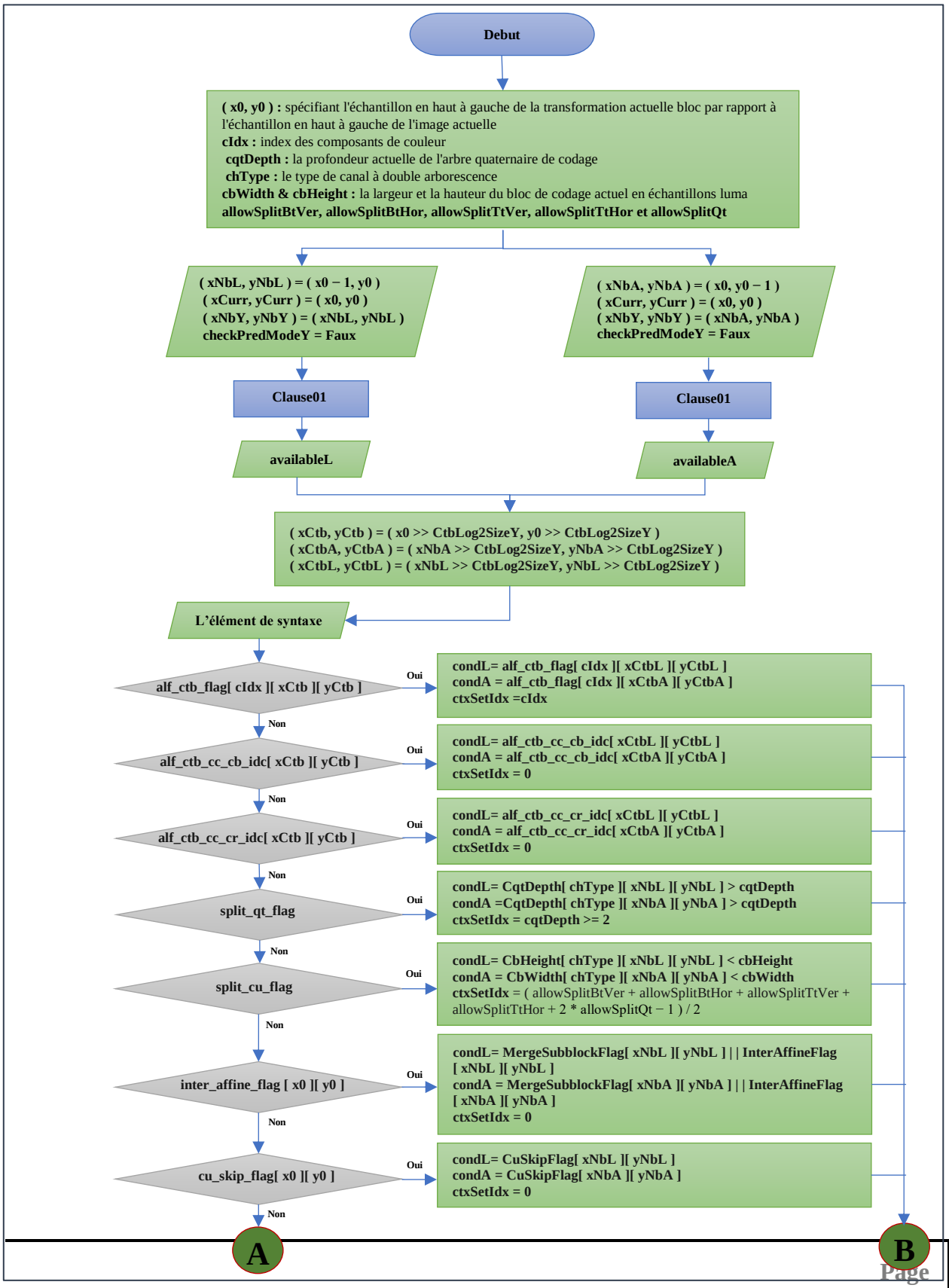


Figure III.4– Programme et résultat de simulation du clause 1.

III.3.2.2 Clause 2

Cette clause permet de calculer la variable *ctxInc* pour les éléments syntaxiques suivants :

- alf_ctb_flag[cIdx][xCtb][yCtb] ;
- alf_ctb_cc_cb_idc[xCtb][yCtb] ;
- alf_ctb_cc_cr_idc[xCtb][yCtb] ;
- split_qt_flag ;
- split_cu_flag ;
- cu_skip_flag[x0][y0] ;
- pred_mode_ibc_flag[x0][y0] ;
- intra_mip_flag , inter_affine_flag[x0][y0] ;
- merge_subblock_flag[x0][y0] ;
- pred_mode_flag[x0][y0] et non_inter_flag.


Debut
(x0, y0) : spécifiant l'échantillon en haut à gauche de la transformation actuelle bloc par rapport à l'échantillon en haut à gauche de l'image actuelle
cIdx : index des composants de couleur
cqtDepth : la profondeur actuelle de l'arbre quaternaire de codage
chType : le type de canal à double arborescence
cbWidth & cbHeight : la largeur et la hauteur du bloc de codage actuel en échantillons luma
allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSplitTtHor et allowSplitQt
(xNbL, yNbL) = (x0 − 1, y0)
(xCurr, yCurr) = (x0, y0)
(xNbY, yNbY) = (xNbL, yNbL)
checkPredModeY = Faux
(xNbA, yNbA) = (x0, y0 − 1)
(xCurr, yCurr) = (x0, y0)
(xNbY, yNbY) = (xNbA, yNbA)
checkPredModeY = Faux
Clause01
Clause01
availableL
availableA
(xCtb, yCtb) = (x0 >> CtbLog2SizeY, y0 >> CtbLog2SizeY)
(xCtbA, yCtbA) = (xNbA >> CtbLog2SizeY, yNbA >> CtbLog2SizeY)
(xCtbL, yCtbL) = (xNbL >> CtbLog2SizeY, yNbL >> CtbLog2SizeY)
L'élément de syntaxe
alf_ctb_flag[cIdx][xCtb][yCtb]
Oui
condL= alf_ctb_flag[cIdx][xCtbL][yCtbL]
condA = alf_ctb_flag[cIdx][xCtbA][yCtbA]
ctxSetIdx =cIdx
Non
alf_ctb_cc_cb_idc[xCtb][yCtb]
Oui
condL= alf_ctb_cc_cb_idc[xCtbL][yCtbL]
condA = alf_ctb_cc_cb_idc[xCtbA][yCtbA]
ctxSetIdx = 0
Non
alf_ctb_cc_cr_idc[xCtb][yCtb]
Oui
condL= alf_ctb_cc_cr_idc[xCtbL][yCtbL]
condA = alf_ctb_cc_cr_idc[xCtbA][yCtbA]
ctxSetIdx = 0
Non
split_qt_flag
Oui
condL= CqtDepth[chType][xNbL][yNbL] > cqtDepth
condA =CqtDepth[chType][xNbA][yNbA] > cqtDepth
ctxSetIdx = cqtDepth >= 2
Non
split_cu_flag
Oui
condL= CbHeight[chType][xNbL][yNbL] < cbHeight
condA = CbWidth[chType][xNbA][yNbA] < cbWidth
ctxSetIdx = (allowSplitBtVer + allowSplitBtHor + allowSplitTtVer + allowSplitTtHor + 2 * allowSplitQt − 1) / 2
Non
inter_affine_flag [x0][y0]
Oui
condL= MergeSubblockFlag[xNbL][yNbL] | | InterAffineFlag [xNbL][yNbL]
condA = MergeSubblockFlag[xNbA][yNbA] | | InterAffineFlag [xNbA][yNbA]
ctxSetIdx = 0
Non
cu_skip_flag[x0][y0]
Oui
condL= CuSkipFlag[xNbL][yNbL]
condA = CuSkipFlag[xNbA][yNbA]
ctxSetIdx = 0
Non
A
B

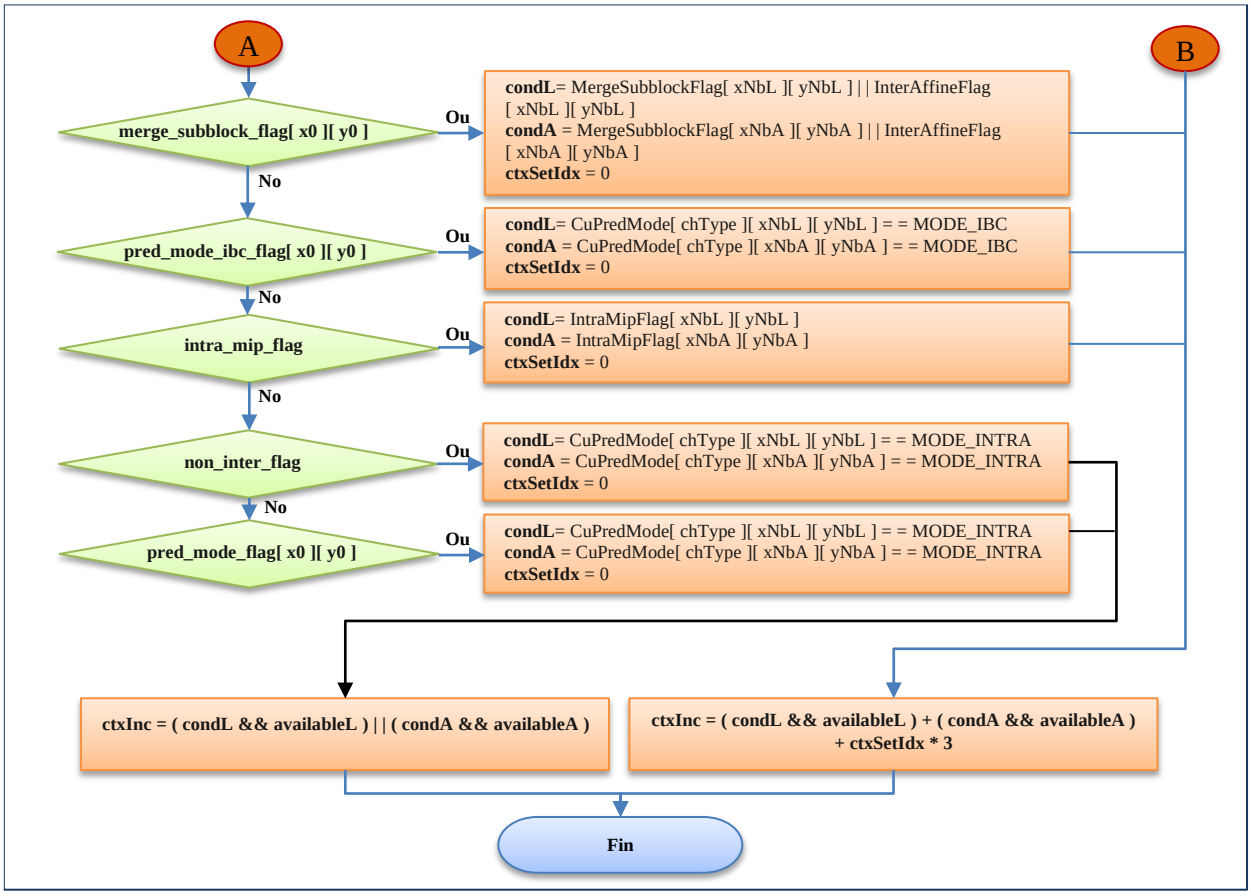


Figure III.5– Organigramme proposé de clause 2

. La figure III.5, illustre l'organigramme de calcul du contexte dans la clause 2. Il est à remarquer que le résultat de la clause 1 est utilisé comme entrée pour cette clause.

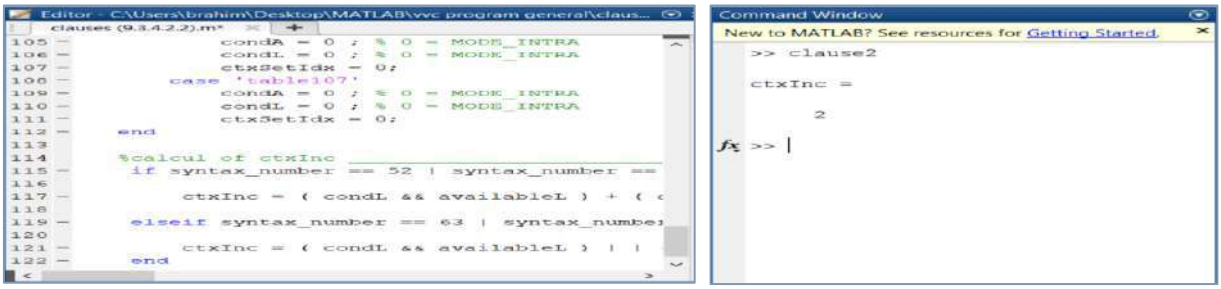


Figure III.6– Programme et résultat de simulation pour la clause 2.

Le résultat de simulation obtenu pour le processus de calcul de la clause 2 est présenté sur la figure III.6. Sur la gauche, on trouve le programme Matlab et sur la droite, on obtient la valeur de contexte *CtxInc*.

III.3.2.3 Clause 3

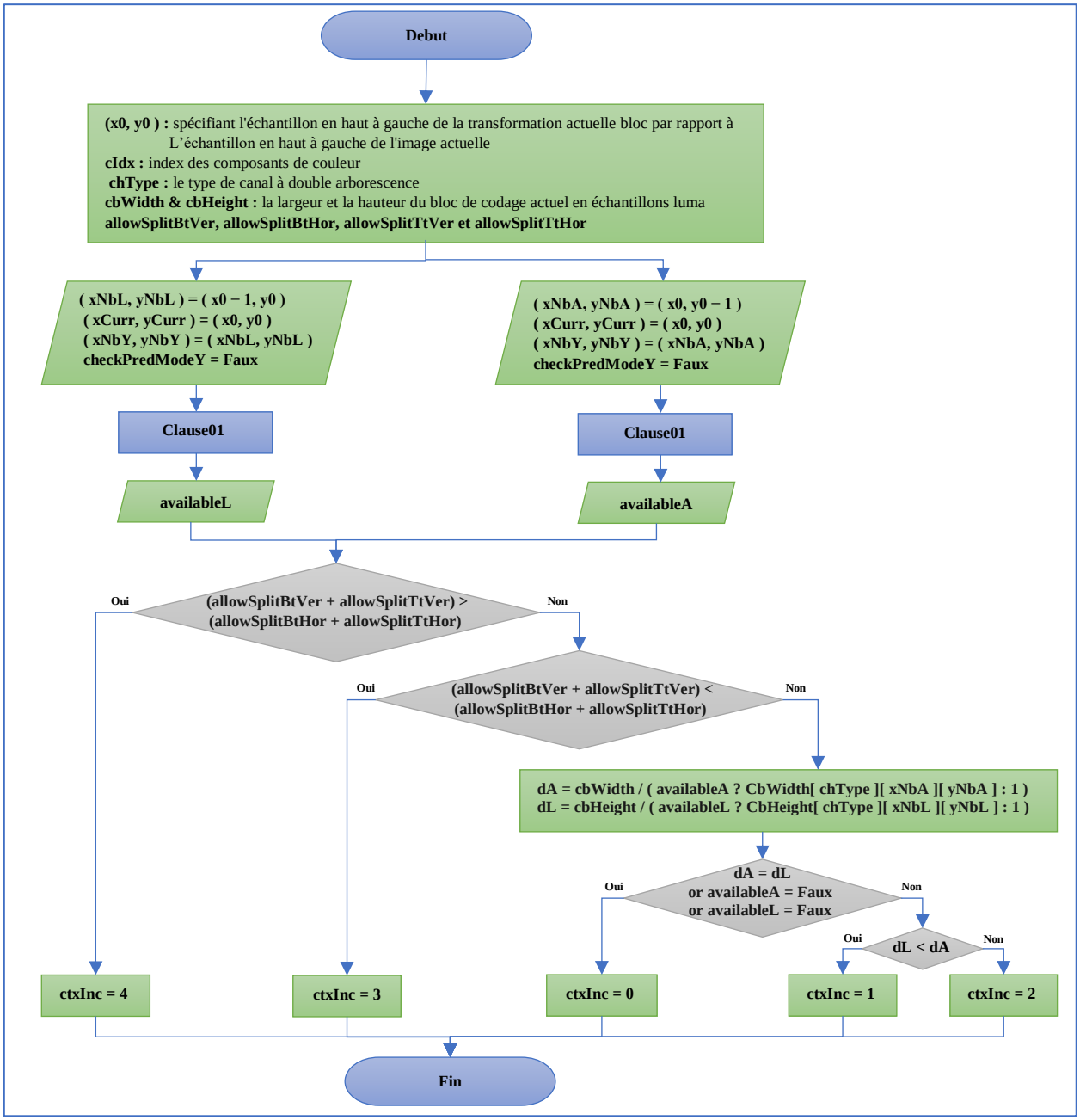


Figure III.7– Organigramme proposé de clause 3.

La clause 3 permet de calculer la variable ctxInc pour l'élément syntaxique mtt_split_cu_vertical_flag. La figure III.7 montre l'organigramme proposé pour cette clause. De plus, le programme et résultat de simulation obtenu pour le calcul de cette clause est présenté sur

la figure III.8. Il est à remarquer que le résultat de la clause 1 sera utilisé comme entrée pour cette clause.



Figure III.8– Programme et résultat de simulation pour la clause 3.

III.3.2.4 Clause 4

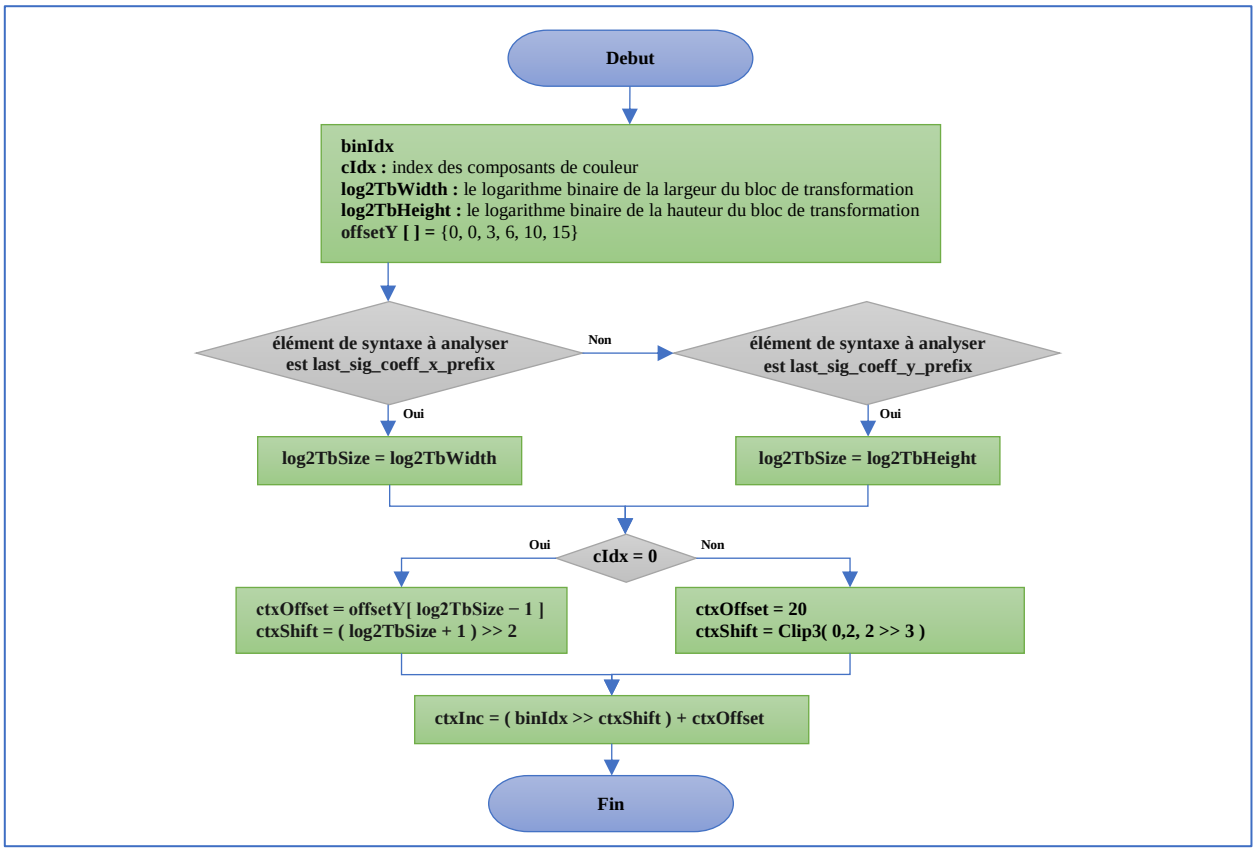


Figure III.9– Organigramme proposé de clause 4.

Cette clause permis de calculer la variable *ctxIdx* pour les éléments syntaxiques *last_sig_coeff_x_prefix* et *last_sig_coeff_y_prefix*. La figure III.9 montre l'organigramme de fonctionnement cette clause dans la norme VVC.

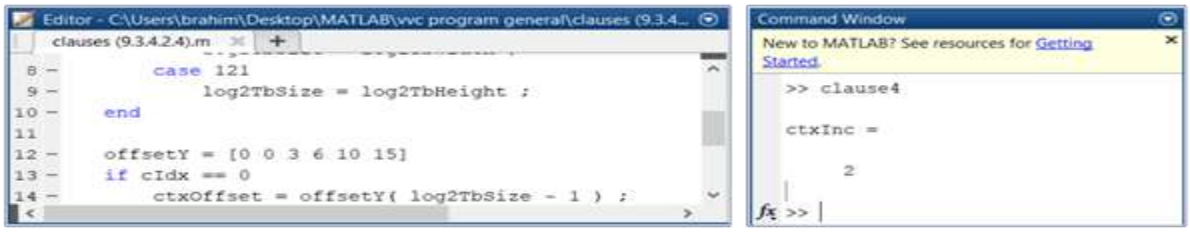


Figure III.10– Programme et résultat de simulation pour la clause 4.

Le résultat de simulation avec le programme qui effectue le calcul de *ctxIdx* dans la clause 4 sont données sur la figure III.10.

III.3.2.5 Clause 5

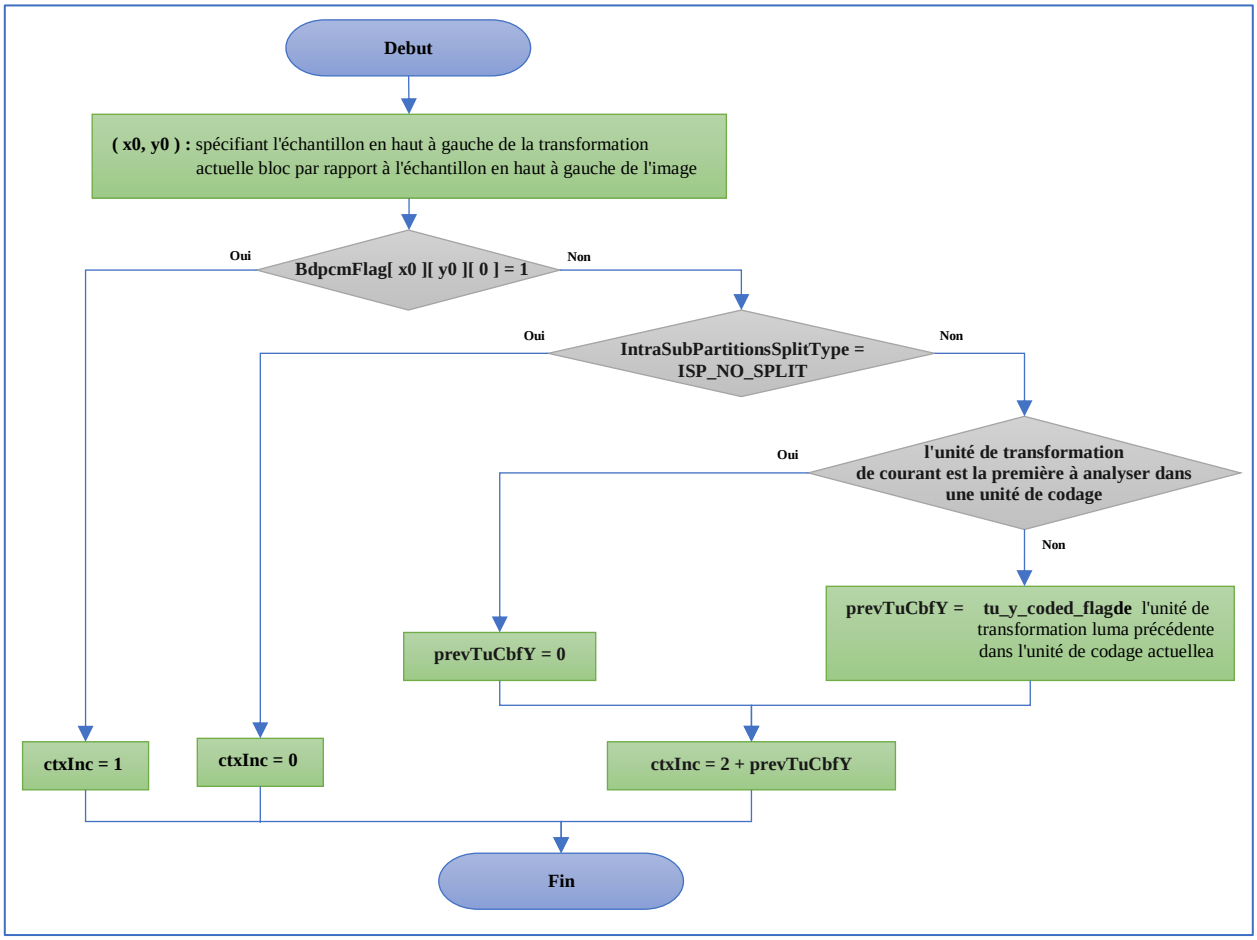


Figure III.11– Organigramme proposé de clause 5.

Cette clause permet de calculer la variable *ctxInc* pour l'élément syntaxique *tu_y_coded_flag*. La figure III.11 illustre l'organigramme de fonctionnement de la clause 5 dans la

norme VVC. Les résultats de simulation avec le programme qui effectue le calcul de l'adresse *ctxIdx* dans la clause 5 sont données sur la figure III.12.

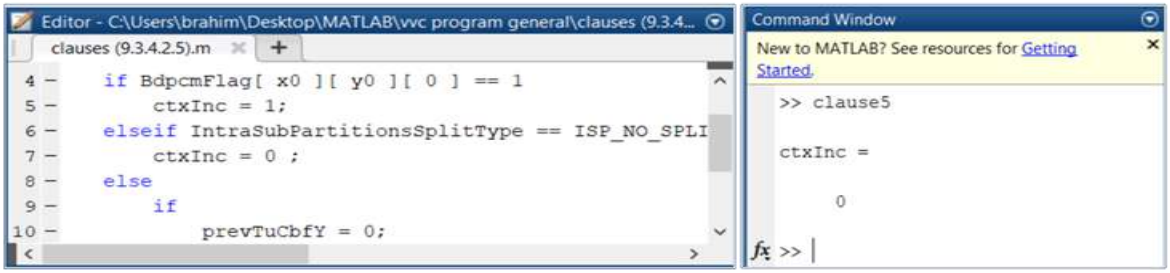


Figure III.12– Programme et résultat de simulation pour la clause 5.

III.3.2.6 Clause 6

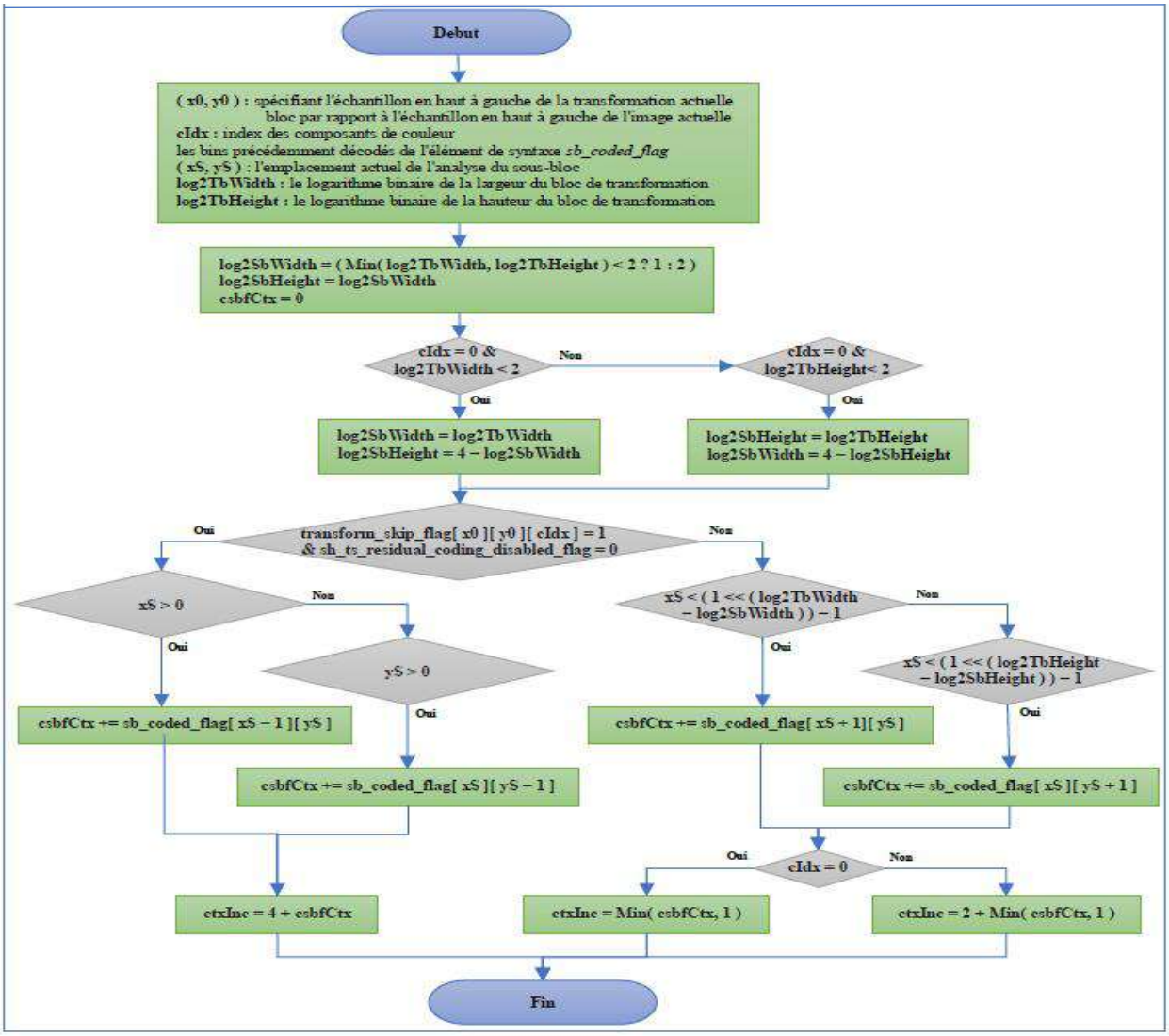


Figure III.13– Organigramme proposé de clause 6.

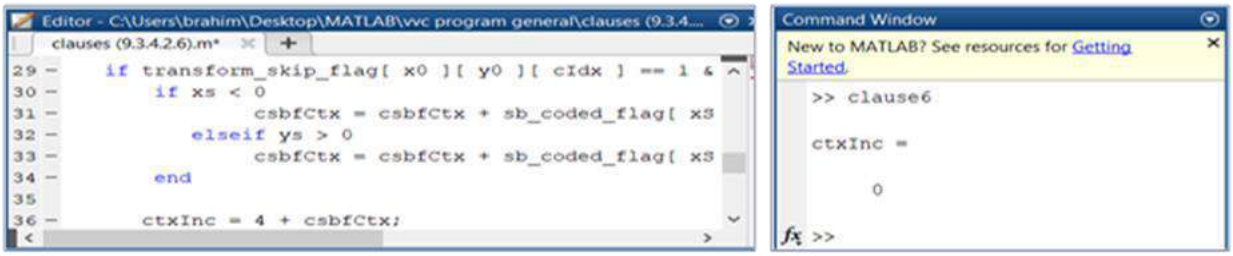


Figure III.14– Programme et résultat de la simulation pour la clause 6.

Cette clause permis de calculer la variable *ctxIdx* pour l'élément syntaxique *sb_coded_flag*, le programme et le résultat de simulation de calcul de la clause 6 est présenté sur la figure III.14.

III.3.2.7 Clause 7

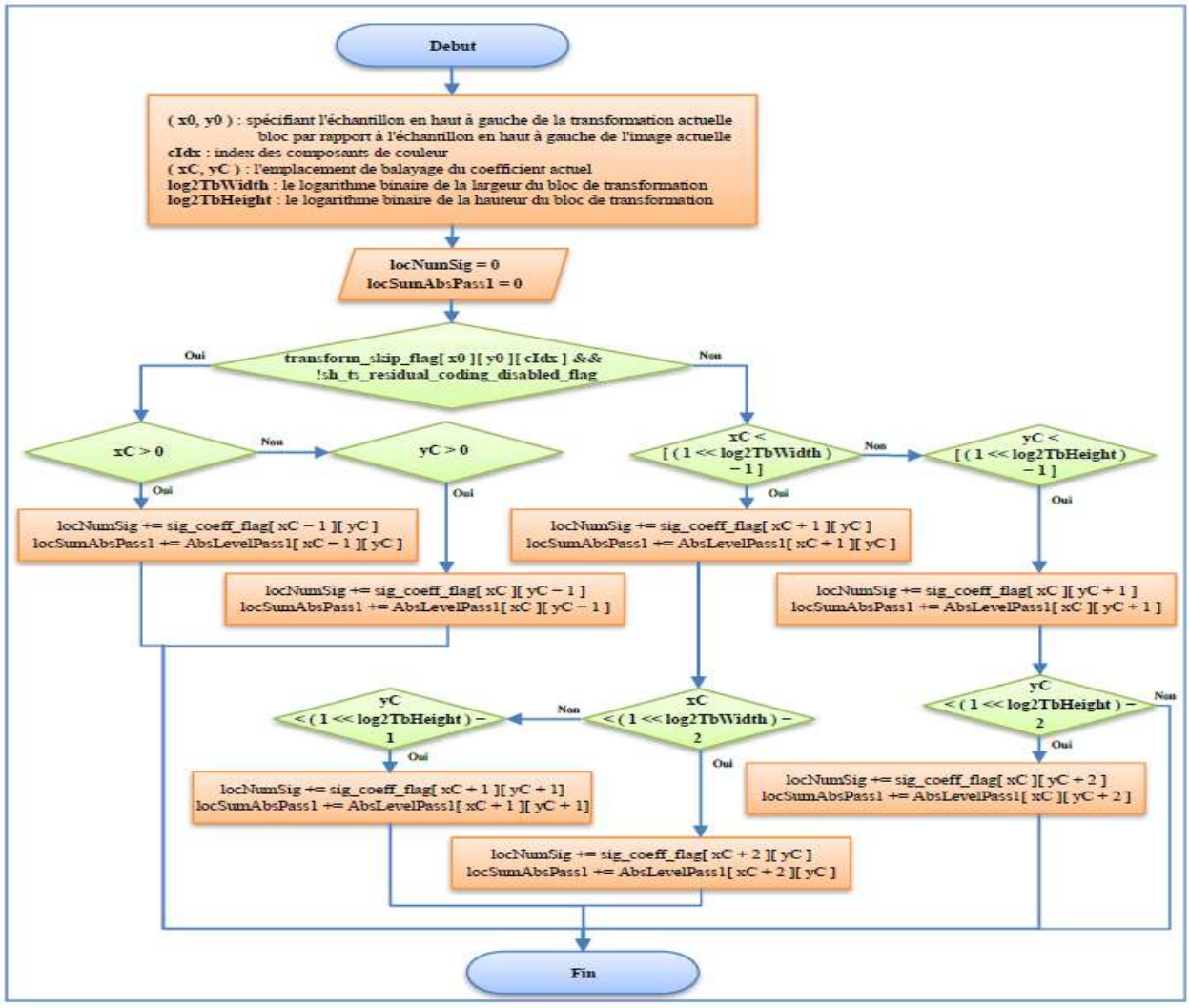


Figure III.15– Organigramme proposé de clause 7.

Cette clause permet de calculer les variables *locNumSig* et *locSumAbsPass1*, l'organigramme proposé est représenté sur la figure III.15. Le programme et le résultat de simulation obtenu de calcul de cette clause est présenté sur la figure III.16.



Figure III.16– Programme et résultat de simulation pour la clause 7.

III.3.2.8 Clause 8

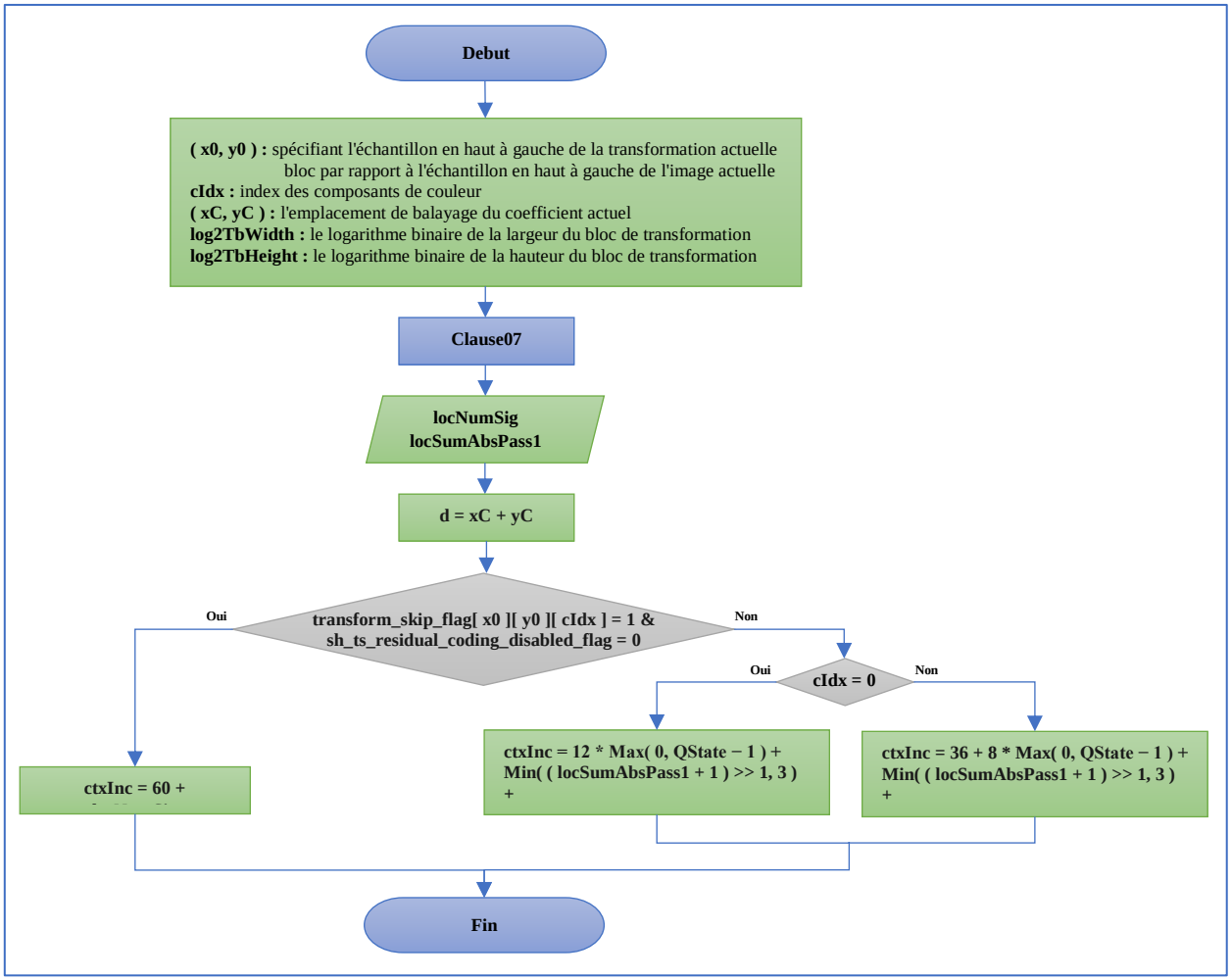


Figure III.17– Organigramme proposé de clause 8.

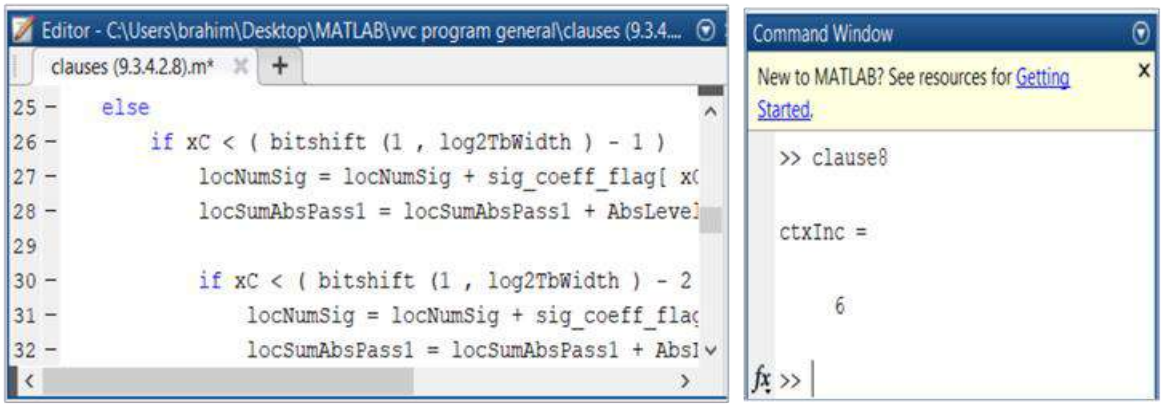


Figure III.18– Programme et résultat de simulation pour la clause 8.

Cette clause permet de calculer la variable *ctxInc* pour l'élément syntaxique *sig_coeff_flag*. Il est à remarquer que la clause 7 est utilisée comme entrée pour cette clause. La figure III.17 représente l'organigramme global de la clause 8. Le programme et le résultat de simulation obtenu pour le calcul de cette clause est donnée sur la figure III.18.

III.3.2.9 Clause 9

Cette clause permet de calculer la variable *ctxInc* pour les éléments syntaxiques *par_level_flag* et *abs_level_gtx_flag*. La figure III.20 représente l'organigramme global de la clause 9. Il est à remarquer que la clause 7 est utilisé comme entrée pour cette clause.

Le programme et le résultat de simulation obtenu pour le calcul de la variable *ctxInc* par cette clause est présenté sur la figure III.19.

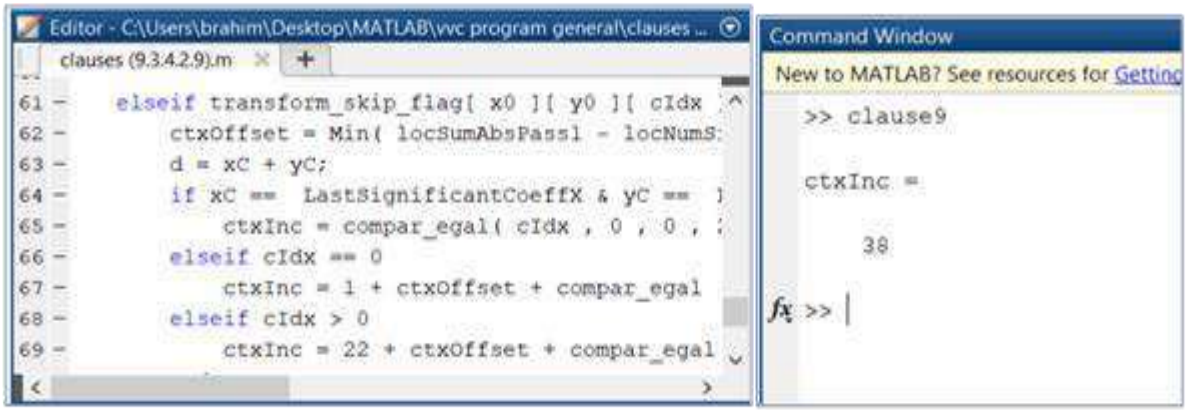


Figure III.19– Programme et résultat de simulation pour la clause 9.

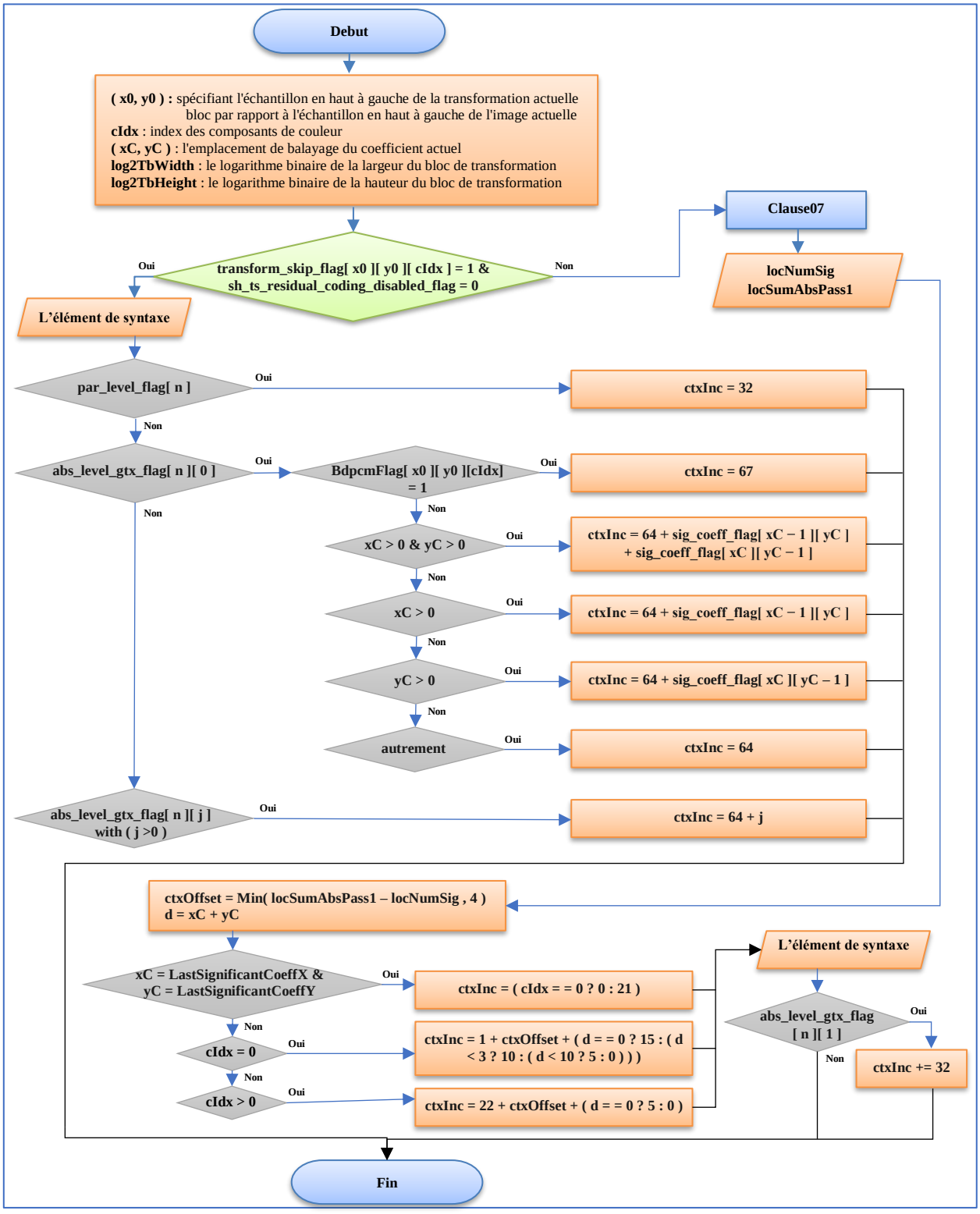


Figure III.20– Organigramme proposé de clause 9.

III.3.2.10 Clause 10

Cette clause permet de calculer la variable *ctxInc* pour l'élément syntaxique *coeff_sign_flag*. La figure III.21 représente l'organigramme de cette clause.

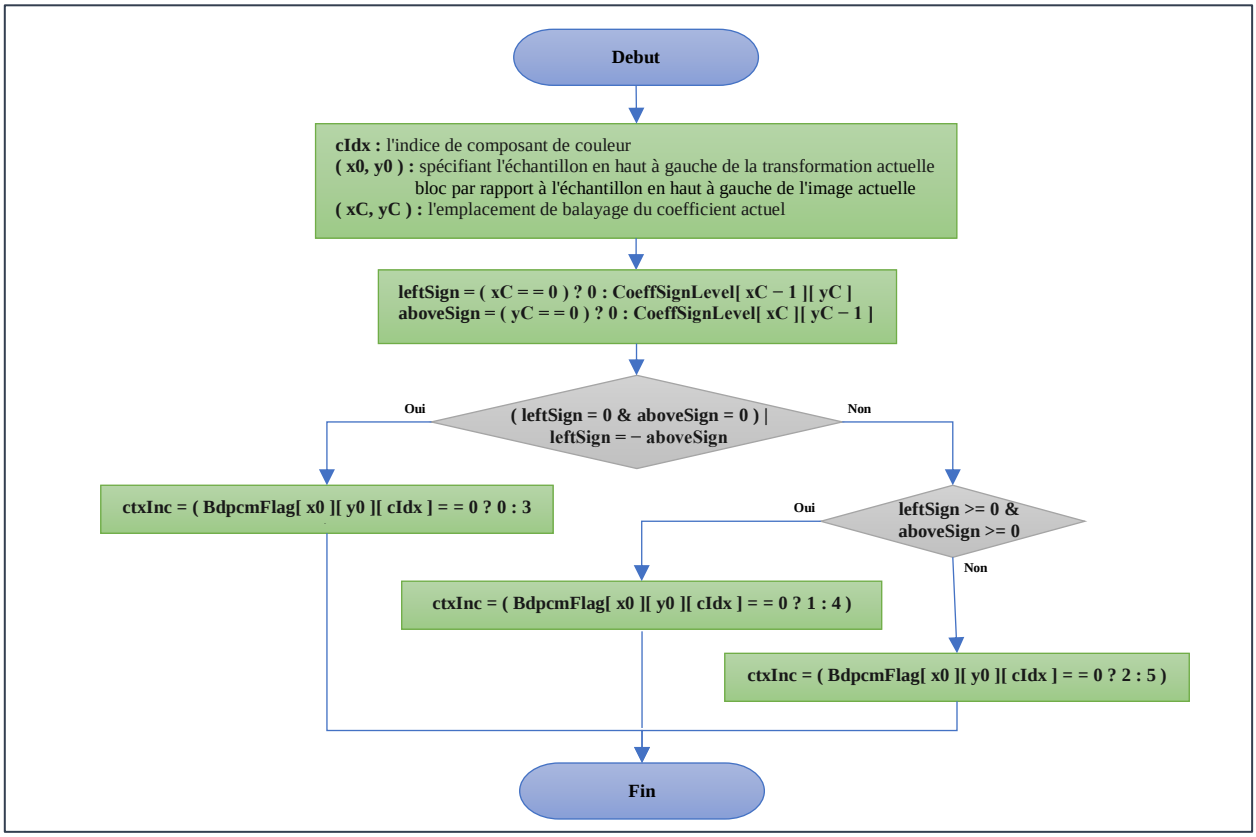


Figure III.21– Organigramme proposé de clause 10.

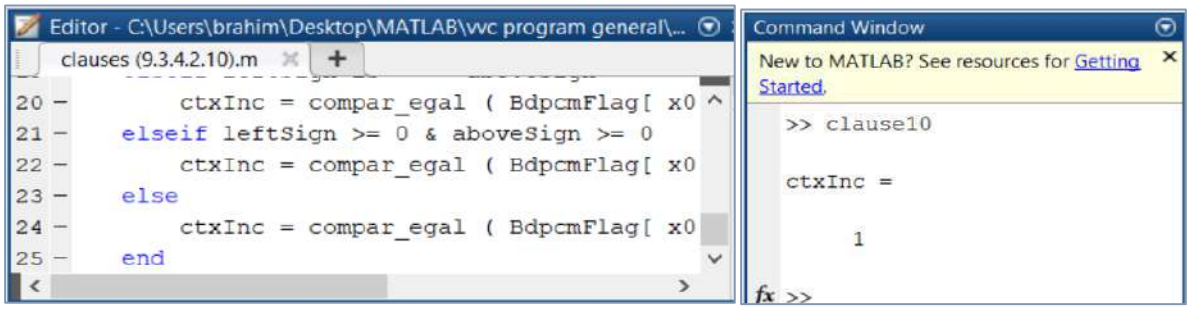


Figure III.22– Programme et résultat de simulation pour la clause 10.

Le programme et le résultat de simulation obtenu pour le processus de calcul de la clause 10 est présenté sur la figure III.22.

III.3.2.11 Clause 11

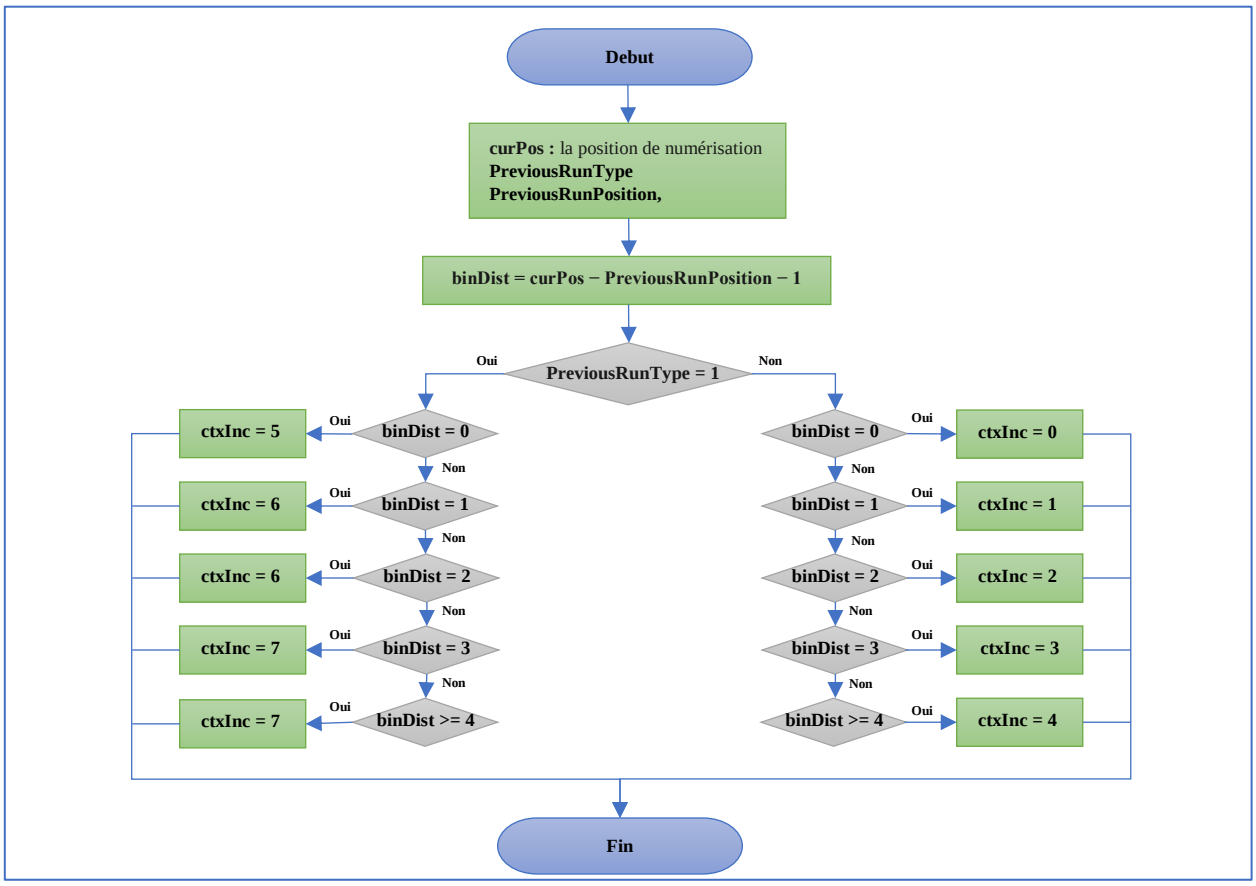


Figure III.23– Organigramme proposé de clause 11.

Cette clause permet de calculer la variable *ctxInc* pour l'élément syntaxique *run_copy_flag*, la figure III.23 représente l'organigramme de la clause 11. Le programme et le résultat de simulation obtenu pour le calcul de cette clause est présenté par la Figure III.24.

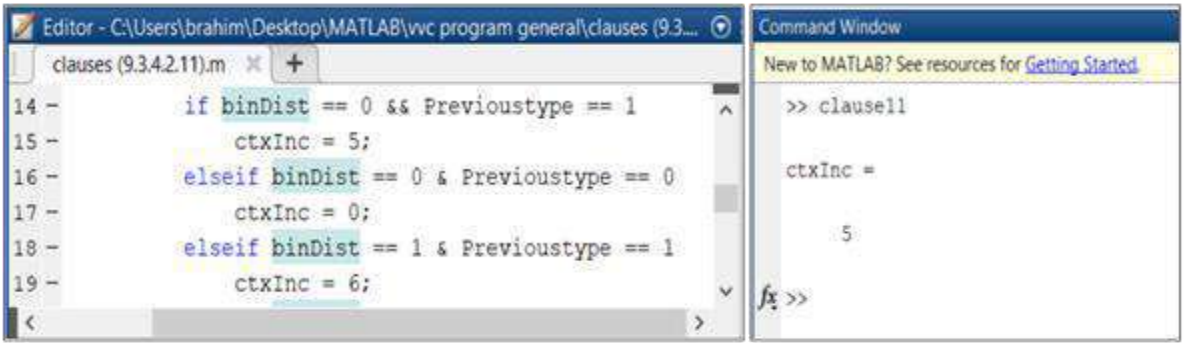


Figure III.24– Programme et résultat de simulation pour la clause 11.

III.3.3 Modélisation du contexte

La partie modélisation du contexte consiste à calculer le modèle de contexte (*pStateIdx0*, *pStateIdx1*) à partir de valeurs obtenues de la table de contexte (*intValue* et *ShiftIdx*) en utilisant l'adresse sélectionnée *ctxIdx*. Il est à remarquer que l'adresse *ctxIdx* est égale à la somme de la variable *ctxInc* et la valeur de *ctxIdxOffset* qui est spécifiée pour la norme VVC.

III.4 Implémentation logicielle et résultat de simulation de la CSM globale

III.4.1 Implémentation sous Matlab de la CSM

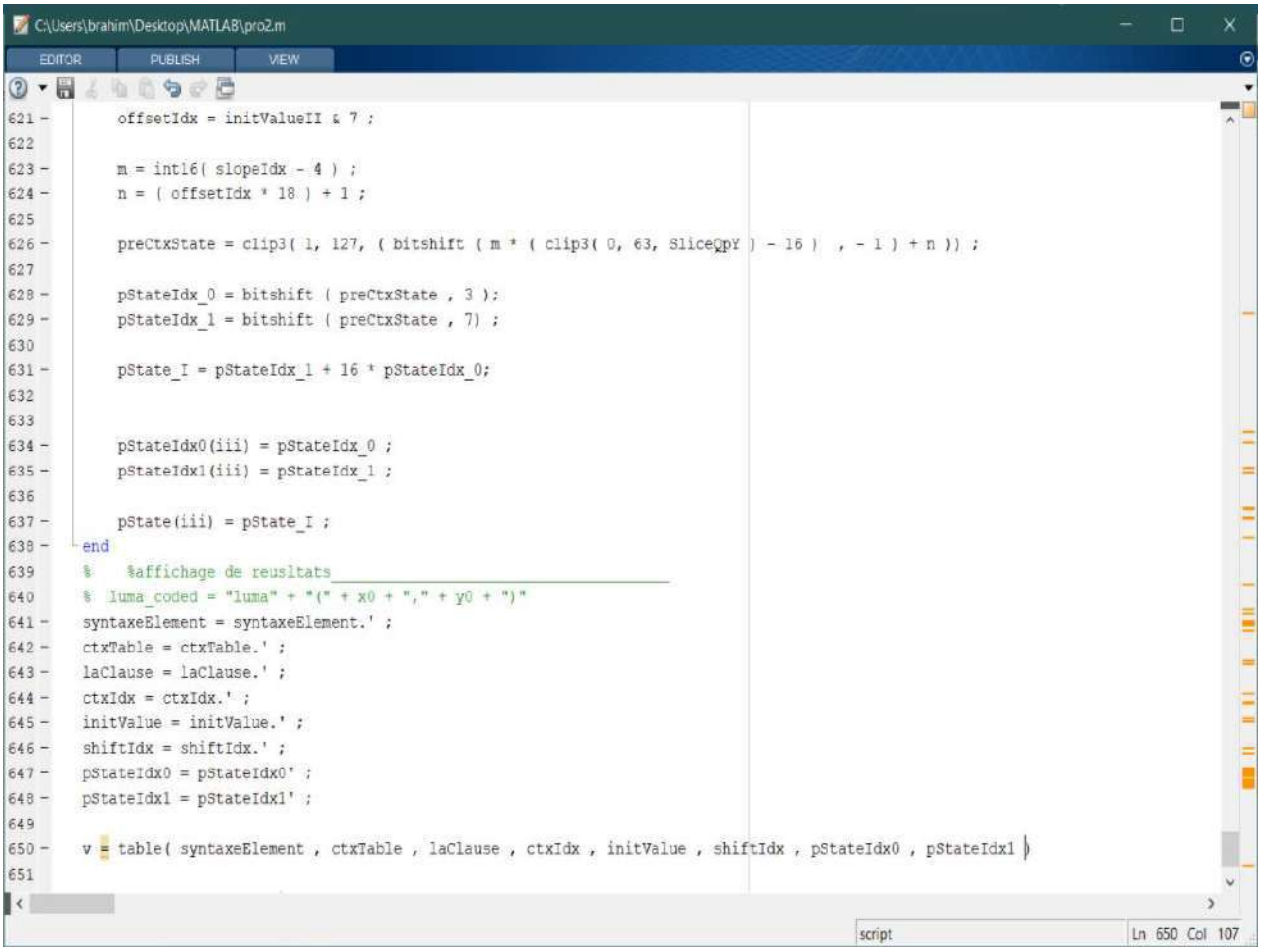
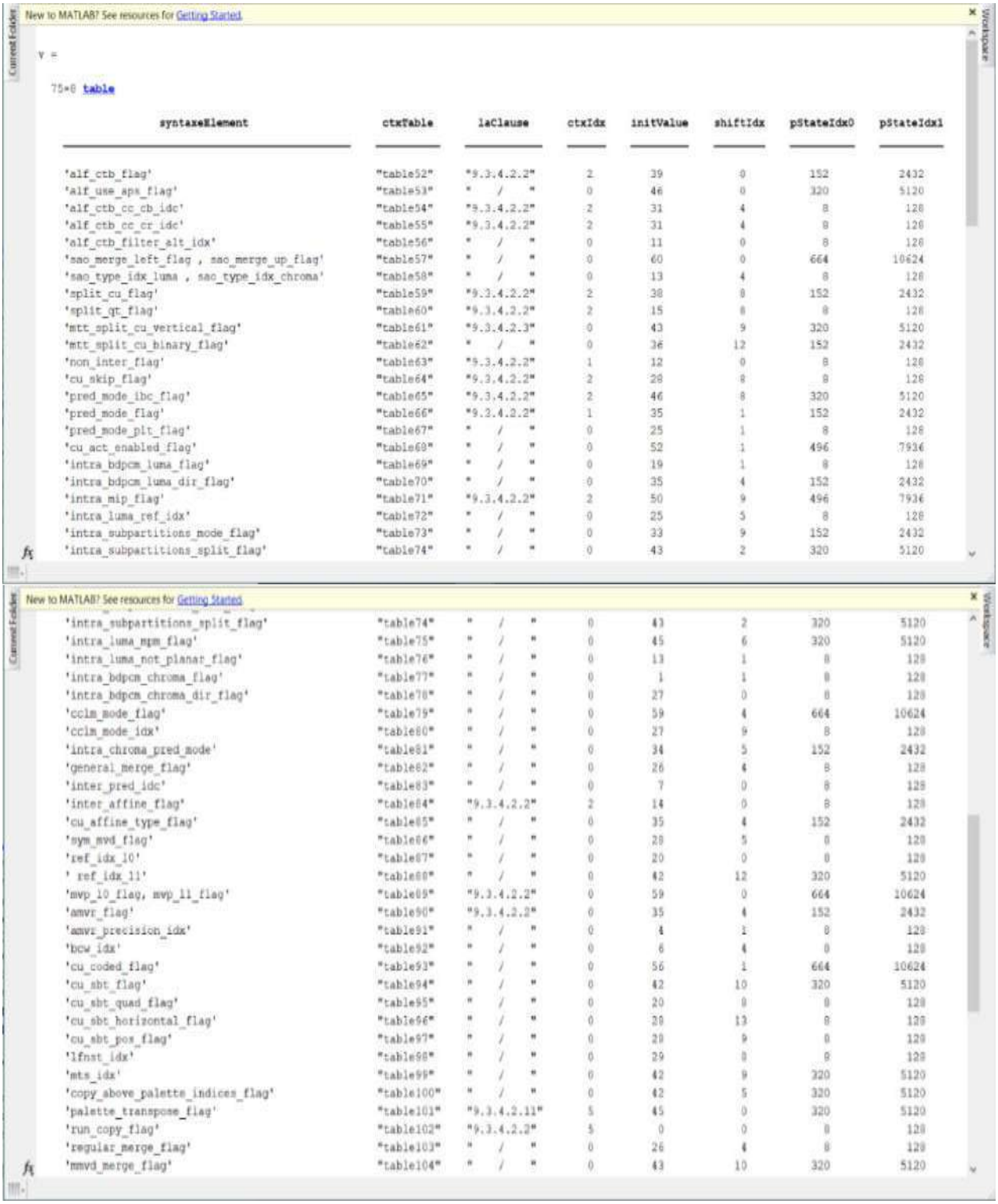


Figure III.25–Implémentation sous Matlab de la partie CSM.

Dans cette section, les trois étapes de la CSM sont rassemblées dans un même programme global pour tous les éléments syntaxiques. La figure III.25 illustre une partie du programme réalisé sous Matlab.

III.4.2 Simulation de la partie CSM



New to MATLAB? See resources for Getting Started

```
v =

  75×8 table
```

syntaxeElement	ctxTable	laClause	ctxIdx	initValue	shiftIdx	pStateIdx0	pStateIdx1
'alf_ctb_flag'	"table52"	"9.3.4.2.2"	2	39	0	152	2432
'alf_use_aps_flag'	"table53"	" / "	0	46	0	320	5120
'alf_ctb_cc_cb_idc'	"table54"	"9.3.4.2.2"	2	31	4	8	128
'alf_ctb_cc_cr_idc'	"table55"	"9.3.4.2.2"	2	31	4	8	128
'alf_ctb_filter_alt_idx'	"table56"	" / "	0	11	0	8	128
'sao_merge_left_flag , sao_merge_up_flag'	"table57"	" / "	0	60	0	664	10624
'sao_type_idx_luma , sao_type_idx_chroma'	"table58"	" / "	0	13	4	8	128
'split_cu_flag'	"table59"	"9.3.4.2.2"	2	38	8	152	2432
'split_qt_flag'	"table60"	"9.3.4.2.2"	2	15	8	8	128
'mtt_split_cu_vertical_flag'	"table61"	"9.3.4.2.3"	0	43	9	320	5120
'mtt_split_cu_binary_flag'	"table62"	" / "	0	36	12	152	2432
'non_inter_flag'	"table63"	"9.3.4.2.2"	1	12	0	8	128
'cu_skip_flag'	"table64"	"9.3.4.2.2"	2	28	8	8	128
'pred_mode_ibc_flag'	"table65"	"9.3.4.2.2"	2	46	8	320	5120
'pred_mode_flag'	"table66"	"9.3.4.2.2"	1	35	1	152	2432
'pred_mode_plt_flag'	"table67"	" / "	0	25	1	8	128
'cu_act_enabled_flag'	"table68"	" / "	0	52	1	496	7936
'intra_bdpcm_luma_flag'	"table69"	" / "	0	19	1	8	128
'intra_bdpcm_luma_dir_flag'	"table70"	" / "	0	35	4	152	2432
'intra_mip_flag'	"table71"	"9.3.4.2.2"	2	50	9	496	7936
'intra_luma_ref_idx'	"table72"	" / "	0	25	5	8	128
'intra_subpartitions_mode_flag'	"table73"	" / "	0	33	9	152	2432
'intra_subpartitions_split_flag'	"table74"	" / "	0	43	2	320	5120

New to MATLAB? See resources for Getting Started

'intra_subpartitions_split_flag'	"table74"	" / "	0	43	2	320	5120
'intra_luma_mpm_flag'	"table75"	" / "	0	45	6	320	5120
'intra_luma_not_planar_flag'	"table76"	" / "	0	13	1	8	128
'intra_bdpcm_chroma_flag'	"table77"	" / "	0	1	1	8	128
'intra_bdpcm_chroma_dir_flag'	"table78"	" / "	0	27	0	8	128
'cclm_mode_flag'	"table79"	" / "	0	59	4	664	10624
'cclm_mode_idx'	"table80"	" / "	0	27	9	8	128
'intra_chroma_pred_mode'	"table81"	" / "	0	34	5	152	2432
'general_merge_flag'	"table82"	" / "	0	26	4	8	128
'inter_pred_idc'	"table83"	" / "	0	7	0	8	128
'inter_affine_flag'	"table84"	"9.3.4.2.2"	2	14	0	8	128
'cu_affine_type_flag'	"table85"	" / "	0	35	4	152	2432
'sym_mvd_flag'	"table86"	" / "	0	28	5	8	128
'ref_idx_l0'	"table87"	" / "	0	20	0	8	128
' ref_idx_l1'	"table88"	" / "	0	42	12	320	5120
'mvp_l0_flag, mvp_l1_flag'	"table89"	"9.3.4.2.2"	0	59	0	664	10624
'amvr_flag'	"table90"	"9.3.4.2.2"	0	35	4	152	2432
'amvr_precision_idx'	"table91"	" / "	0	4	1	8	128
'bcw_idx'	"table92"	" / "	0	6	4	8	128
'cu_coded_flag'	"table93"	" / "	0	56	1	664	10624
'cu_sbt_flag'	"table94"	" / "	0	42	10	320	5120
'cu_sbt_quad_flag'	"table95"	" / "	0	20	8	8	128
'cu_sbt_horizontal_flag'	"table96"	" / "	0	28	13	8	128
'cu_sbt_pos_flag'	"table97"	" / "	0	28	9	8	128
'lfnst_idx'	"table98"	" / "	0	29	8	8	128
'mts_idx'	"table99"	" / "	0	42	9	320	5120
'copy_above_palette_indices_flag'	"table100"	" / "	0	42	5	320	5120
'palette_transpose_flag'	"table101"	"9.3.4.2.11"	5	45	0	320	5120
'run_copy_flag'	"table102"	"9.3.4.2.2"	5	0	0	8	128
'regular_merge_flag'	"table103"	" / "	0	26	4	8	128
'mmvd_merge_flag'	"table104"	" / "	0	43	10	320	5120

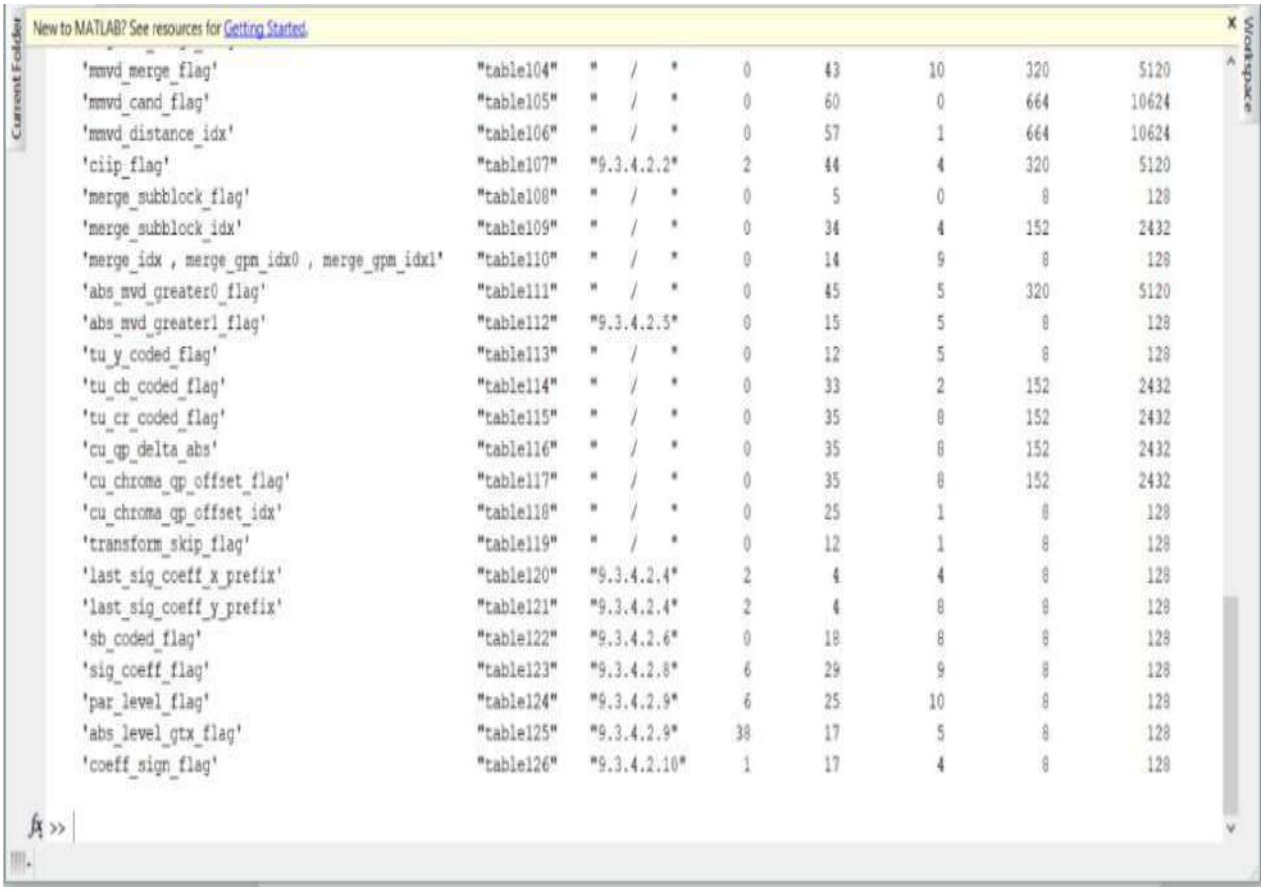


Figure III.26– Résultats de simulation de la partie CSM.

Les résultats de simulation de la partie CSM est donnée sur la figure III.26. De plus, tous les éléments de chacune des trois étapes sont affichés pour tous les éléments syntaxiques traités avec le mode régulier et qui ont des méthodes spécifiques pour calculer le modèle de contexte.

On peut voir clairement que parmi les 74 éléments syntaxiques traités par le mode régulier, 20 éléments passent par les clauses afin de calculer le modèle du contexte.

Conclusion

Dans ce chapitre, nous avons présentés les outils matériels et logiciels utilisés dans notre travail avant de passer à l'implémentation logicielle ainsi que la simulation de la partie CSM dans le CABAC/VVC sous Matlab. La description matérielle de cette même partie fera l'objet du chapitre prochain.

Chapitre 04

Chapitre IV : Implémentation du décodeur CABAC sur FPGA

IV.1 Introduction

Ce dernier chapitre est réservé pour l'implémentation matérielle de l'algorithme de sélection et de modélisation du contexte CSM/CABAC sur FPGA. Les architectures proposées suivi par une synthèse sous l'ISE, une simulation sous ModelSim ainsi qu'une estimation de l'énergie consommée sous l'analyseur XPower (ISE) seront données en détails pour chaque clause de cet algorithme. Nous terminerons le chapitre par une comparaison et une discussion des résultats obtenus.

IV.2 Architectures matérielles proposées pour la partie CSM/CABAC

La conception de notre architecture est basée sur des blocs fonctionnels où chaque bloc assure une tâche bien précise. Ces blocs fonctionnels sont interconnectés entre eux et sont contrôlés à travers une FSM (Finit State Machine) qui séquence le bon fonctionnement du CSM/VVC. Avant de donner l'architecture globale de notre algorithme CSM/CABAC qu'est basé sur 11 clauses de calcul du contexte selon la norme VVC, nous allons commencer par la présentation des architectures proposées pour chacune de clause. D'après la norme VVC chaque clause est chargée de calculer le contexte incrément (*ctxInc*) qui sera par la suite utilisé pour obtenir le *ctxIdx* qui sera à son tour injecté à l'entrée du mémoire qui contient les valeurs *intValue*. La variable *intValue* est utilisée comme entrée pour modéliser le contexte *pStateIdx0* et *pStateIdx1*.

IV.2.1 Architecture proposée pour CSM/clause 1

La figure IV.1 représente le calcul du *ctxInc* dans la clause 1 en utilisant un ensemble de variables et de valeurs fixes, spécifiées par la norme VVC.

Les schémas RTL et le top level sont représentés sur la figure IV.2

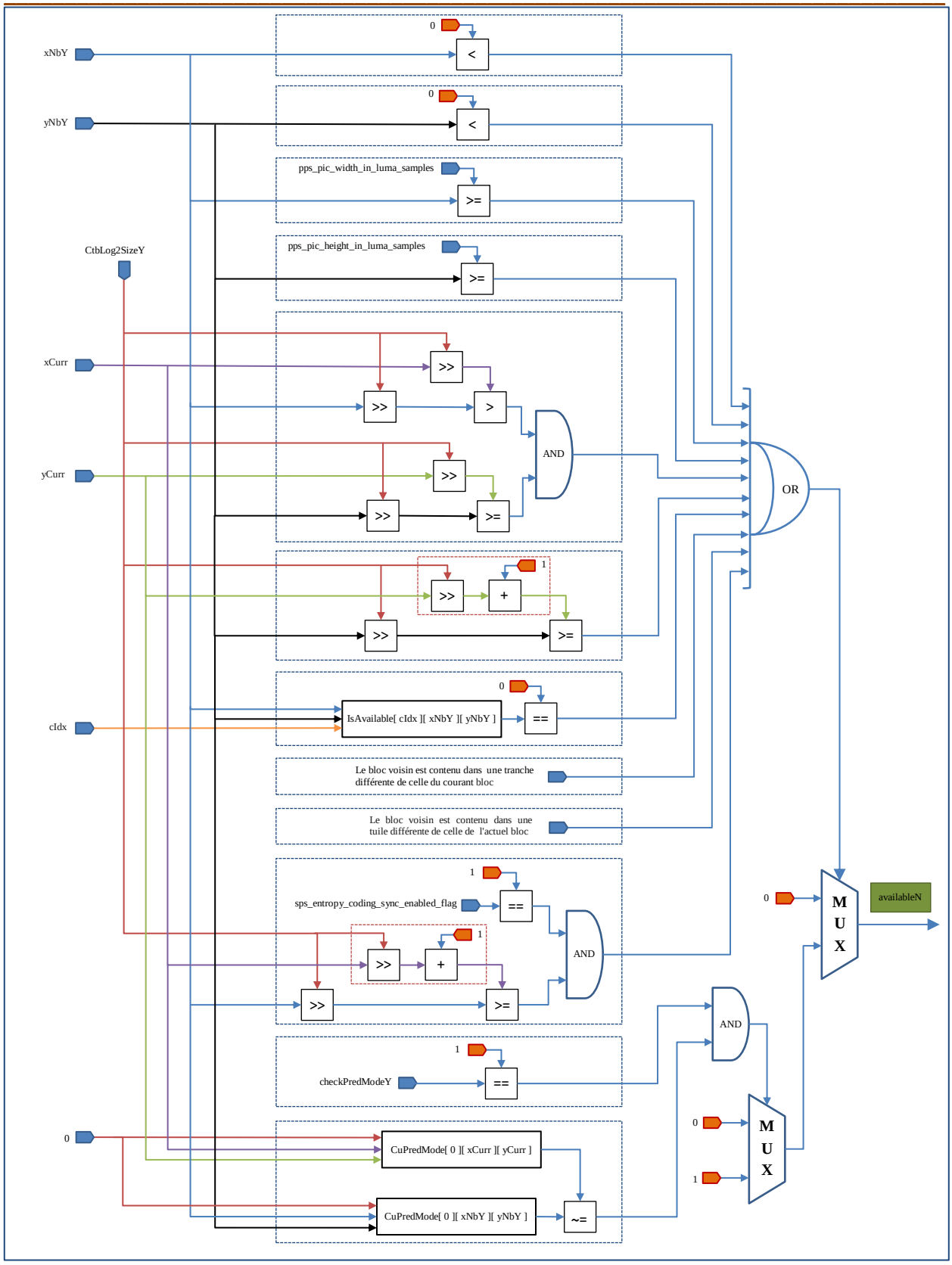


Figure IV.1– Architecture proposée de clause 1.

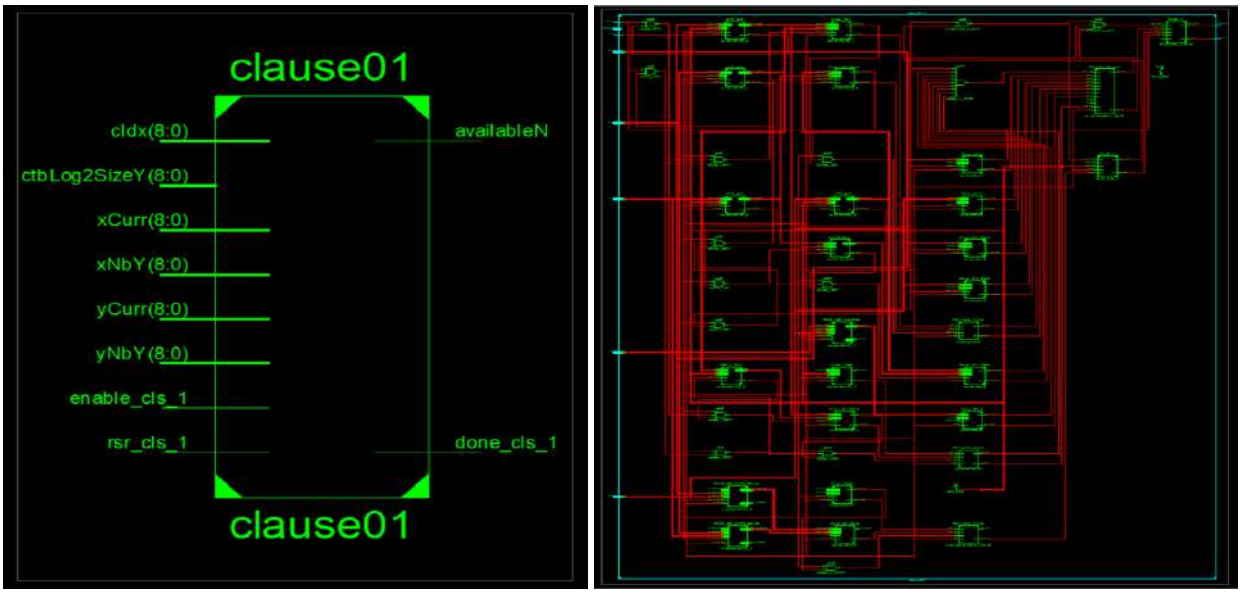


Figure IV.2– Schéma top level et le schéma RTL de la clause 1.

IV.2.2 Architecture proposée pour CSM/clause 2

Les schémas RTL et top level sont depicter sur la figure IV.3.

La figure IV.4 représente le calcul su *ctxInc* dans la clause 2 en utilisant un ensemble de variables et de valeurs fixes.

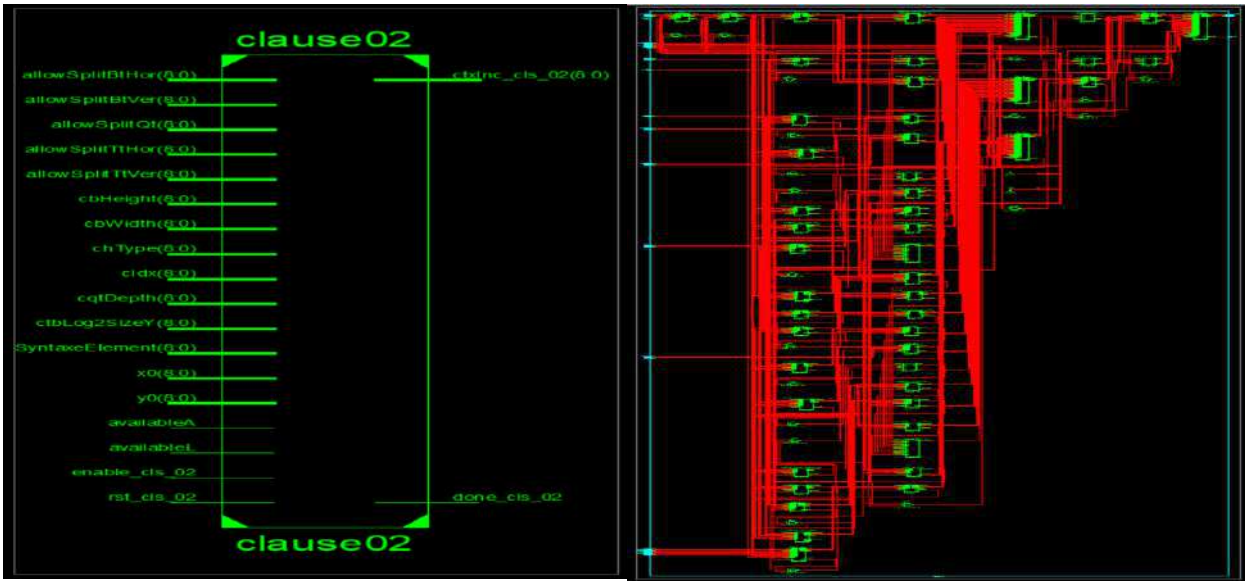


Figure IV.3– Schéma top level et le schéma RTL de la clause 2.

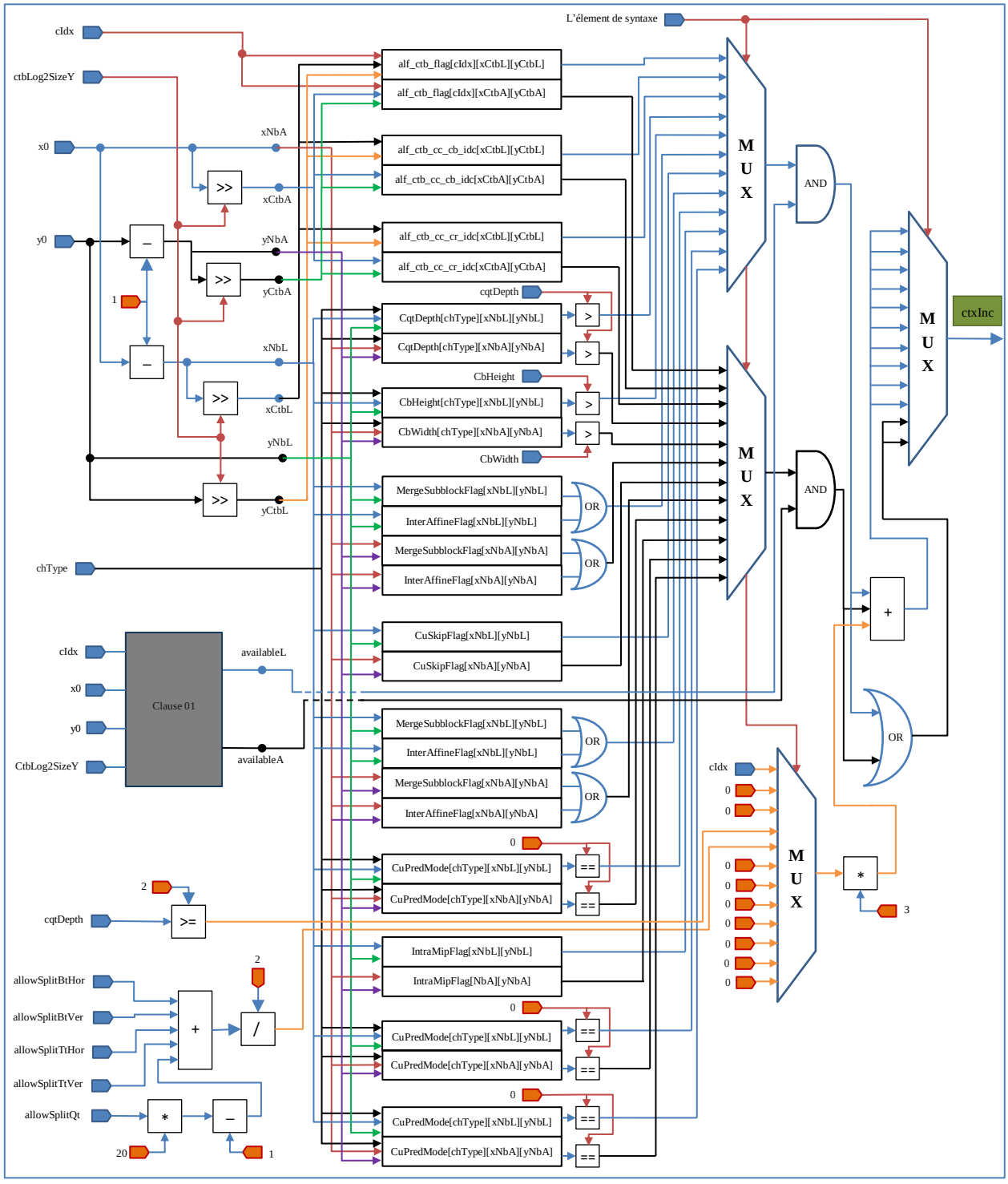


Figure IV.4– Architecture proposée de clause 2.

IV.2.3 Architecture proposée pour CSM/clause 3

La figure IV.5 représente le calcul su *ctxInc* dans la clause 3 en utilisant un ensemble de variables et de valeurs fixes. De plus, Les schémas RTL et top level de cette clause sont donnés sur la figure IV.6.

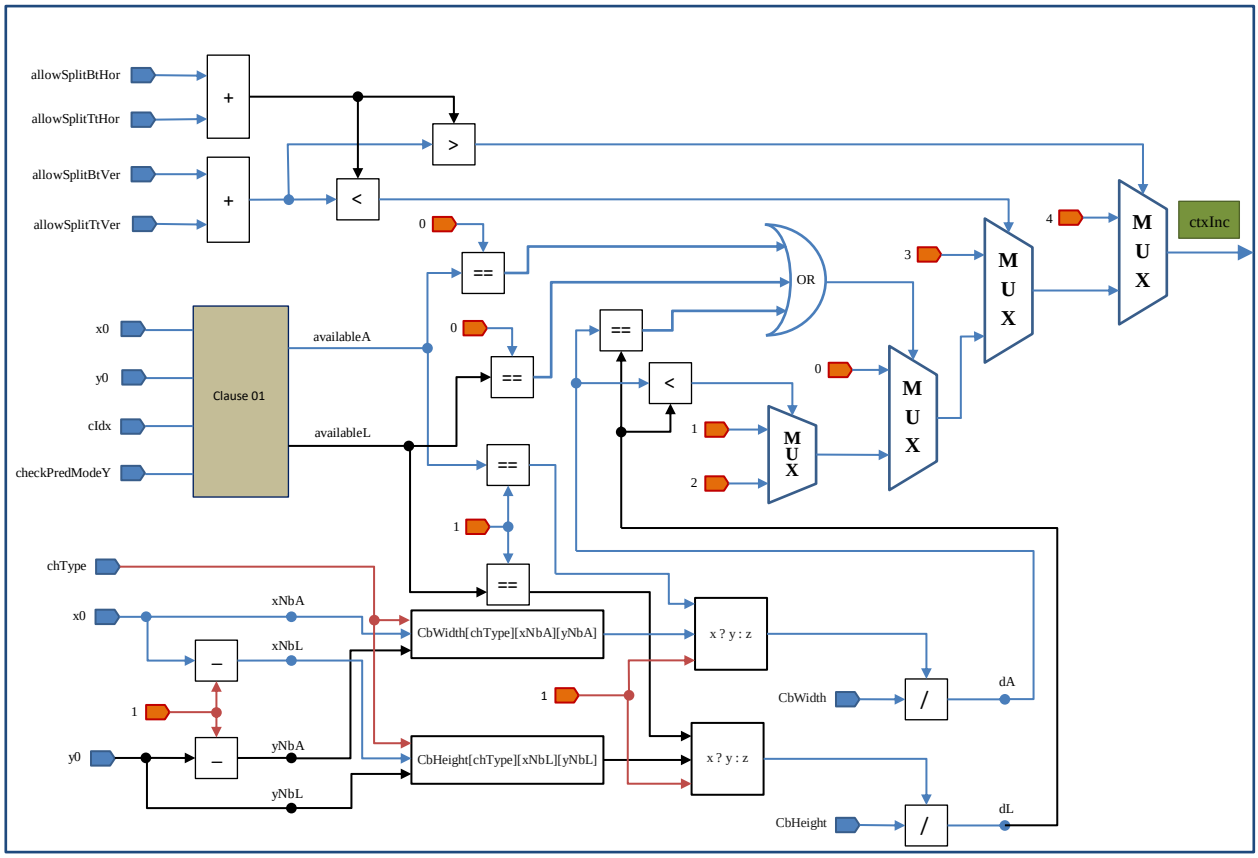


Figure IV.5– Architecture proposée de clause 3.

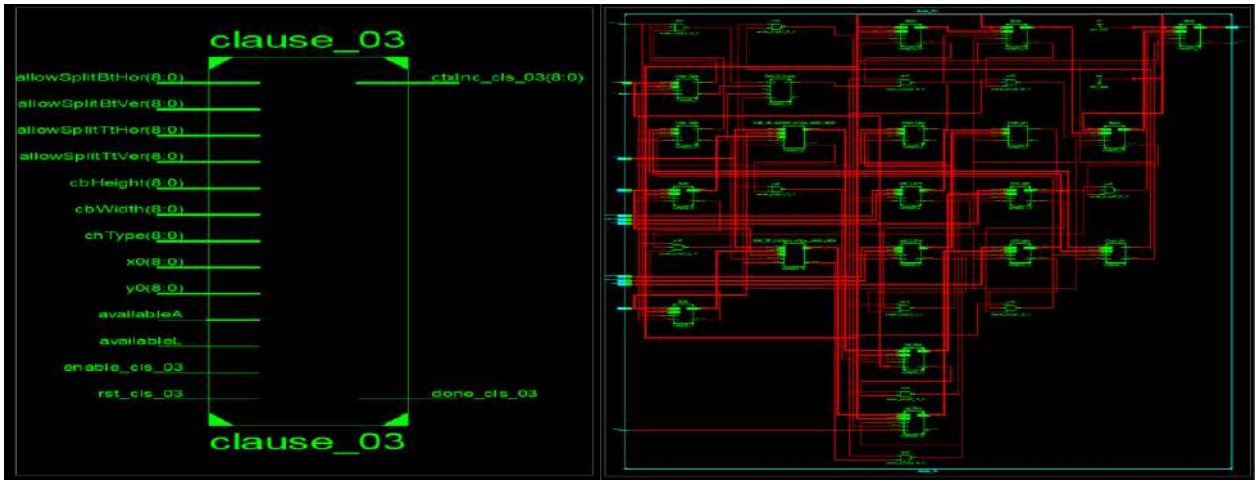


Figure IV.6– Schéma top level et le schéma RTL de la clause 3.

IV.2.4 Architecture proposée pour CSM/clause 4

La figure IV.7 représente le calcul du *ctxInc* dans la clause 4 dont un ensemble de variables et de valeurs fixes. Les schémas RTL et top level sont donnés sur la figure IV.8.

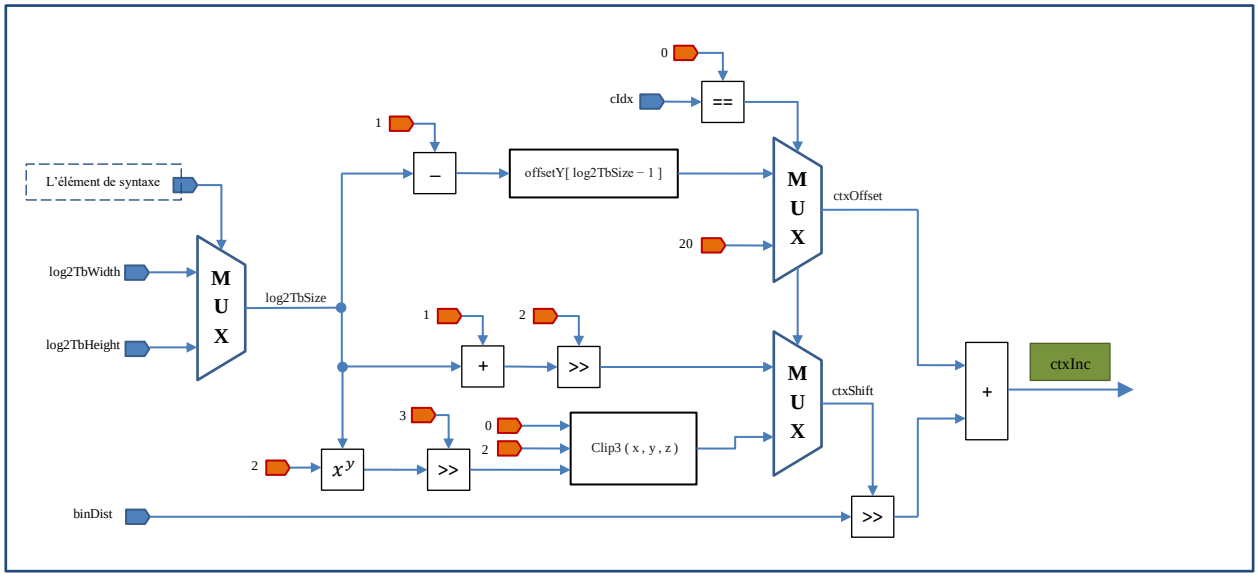


Figure IV.7– Architecture proposée de clause 4.

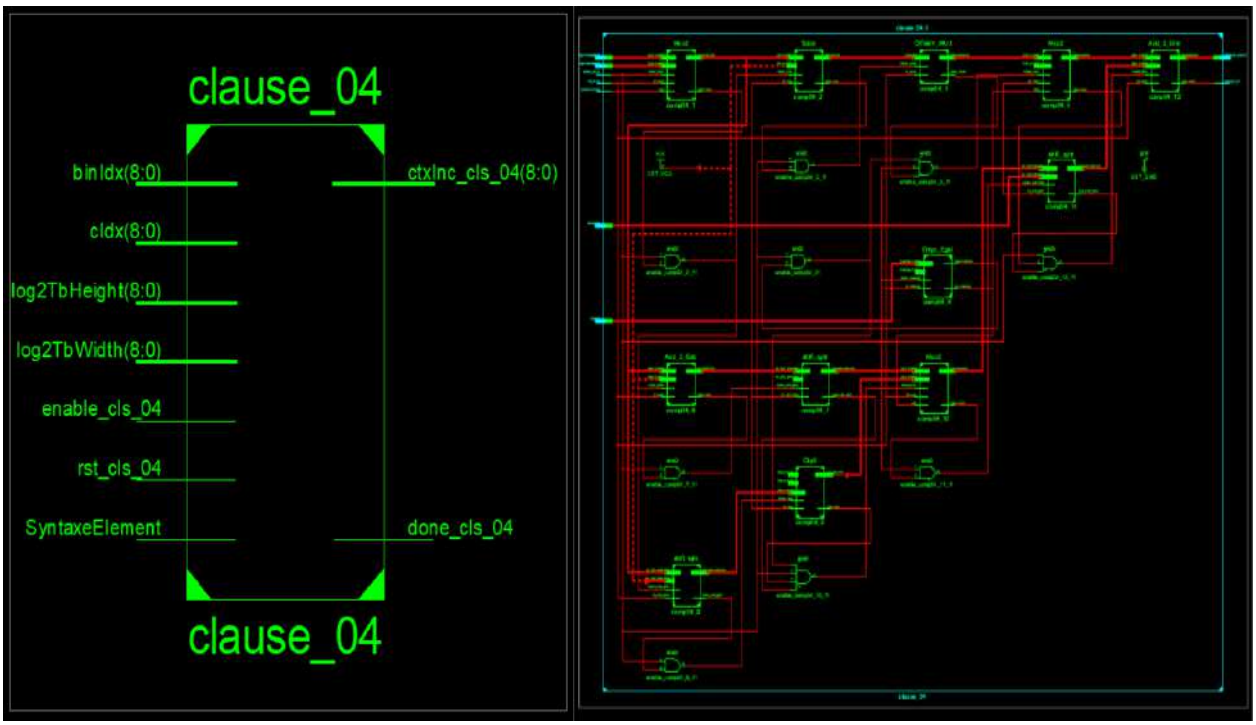


Figure IV.8– Schéma top level et le schéma RTL de la clause 4.

IV.2.5 Architecture proposée pour CSM/clause 5

La figure IV.9 représente le calcul su *ctxInc* dans la clause 5 dont un ensemble de variables et de valeurs fixes. De plus, Les schémas RTL et top level de cette clause sont donnés sur la figure IV.10.

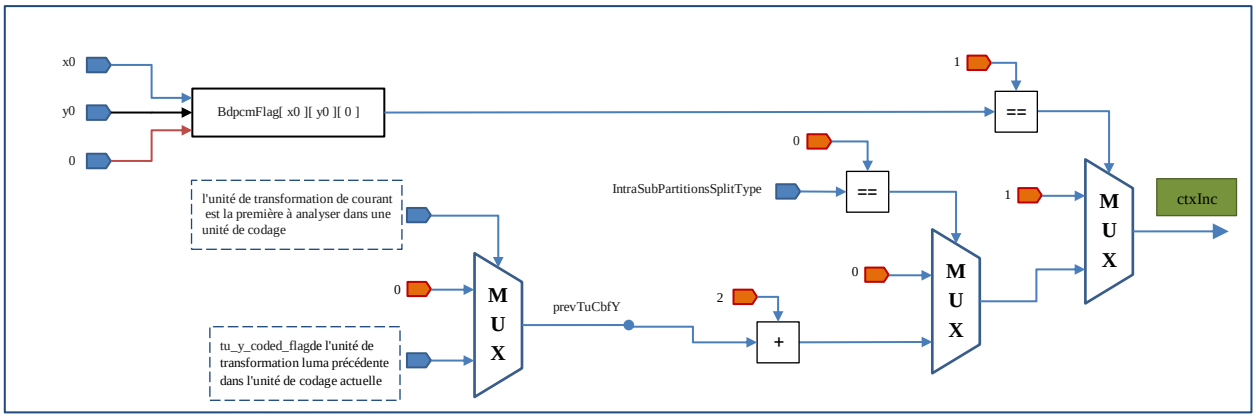


Figure IV.9– Architecture proposée de clause 5.

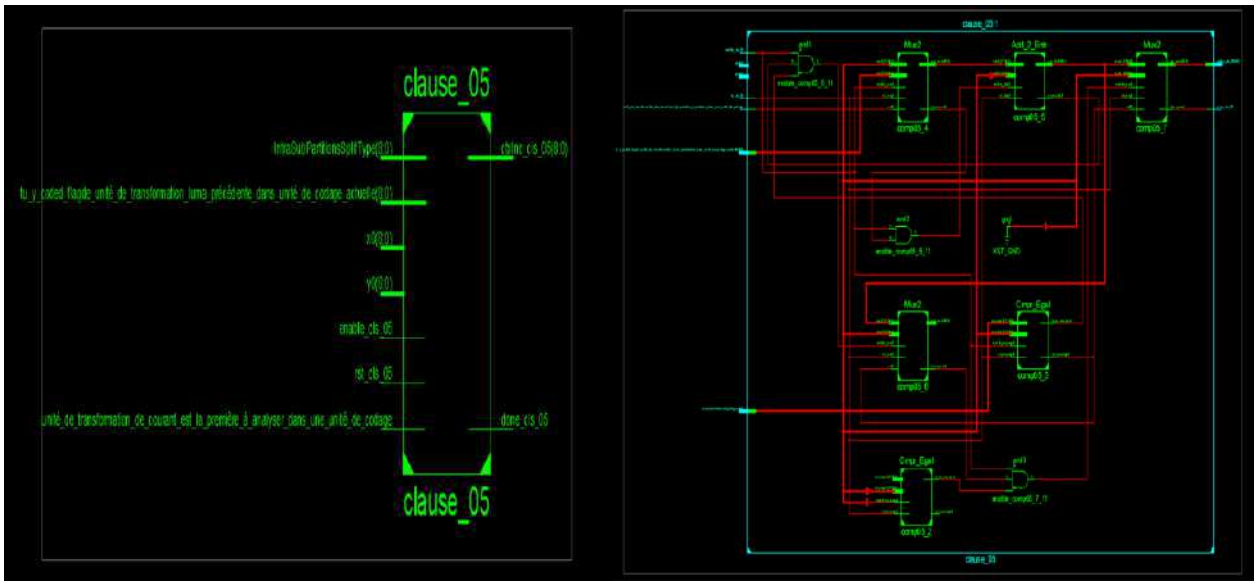


Figure IV.10– Schéma top level et le schéma RTL de la clause 5.

IV.2.6 Architecture proposée pour CSM/clause 6

La figure IV.11 représente le calcul su *ctxInc* dans la clause 6 en utilisant un ensemble de variables et de valeurs fixes. Les schémas RTL et top level sont donnés sur la figure IV.12.

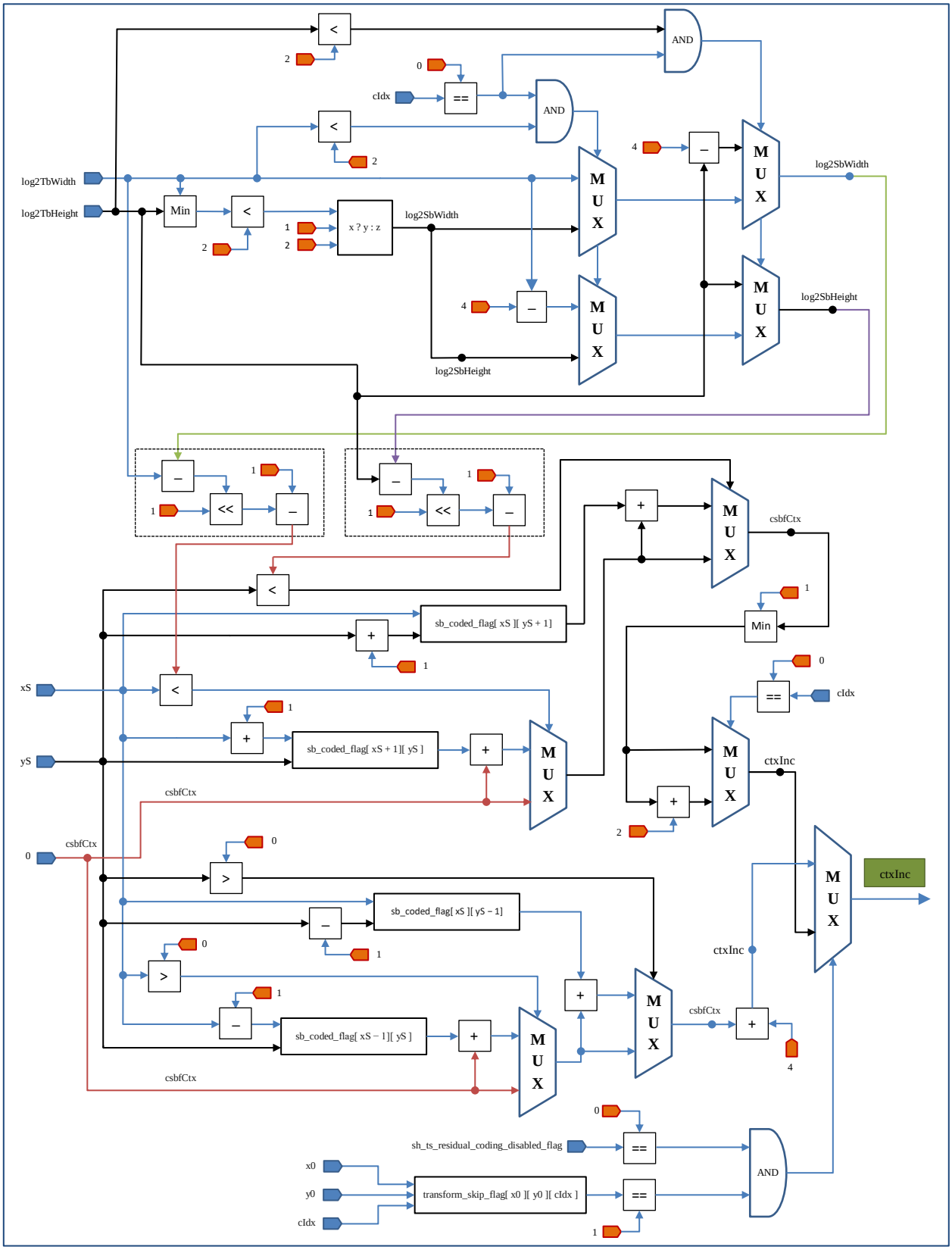


Figure IV.11– Architecture proposée de clause 6.

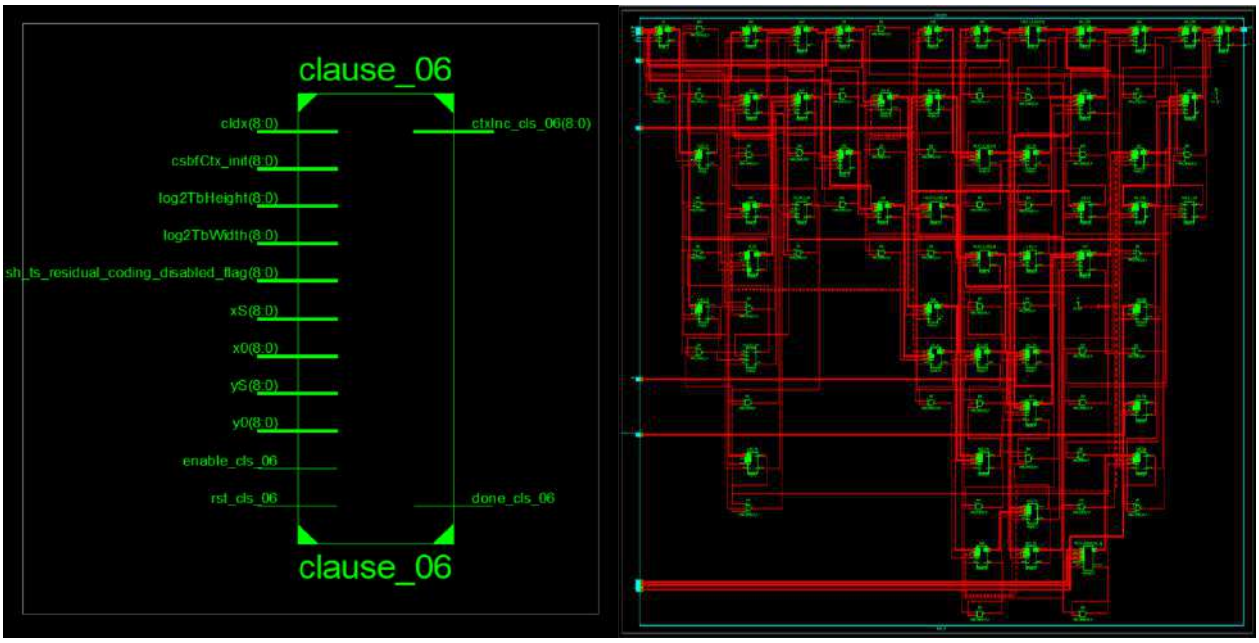


Figure IV.12 Schéma top level et le schéma RTL de la clause 6.

IV.2.7 Architecture proposée pour CSM/clause 7

Les schémas RTL et le top level de la clause 7 sont donnés sur la figure IV.13.

Les figures IV.14 et IV.15 représentent le calcul du *locNumSig* et *locSumAbsPass1* respectivement dans cette clause en utilisant un ensemble de variables et de valeurs fixes.

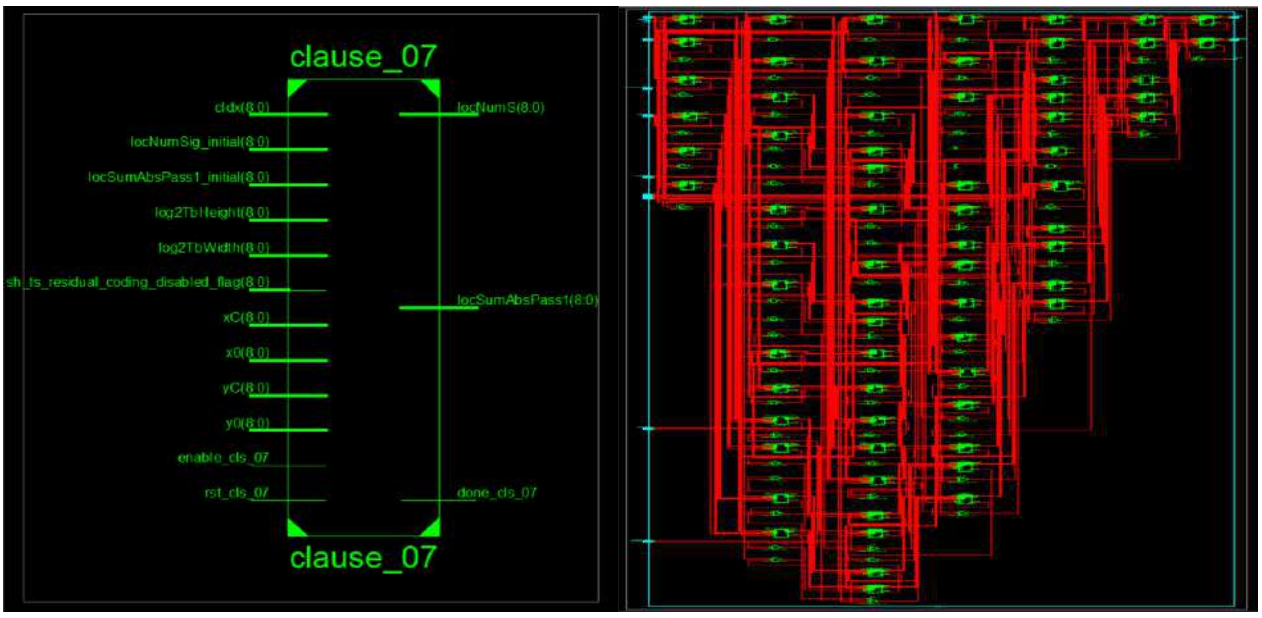


Figure IV.13– Schéma top level et le schéma RTL de la clause 7.

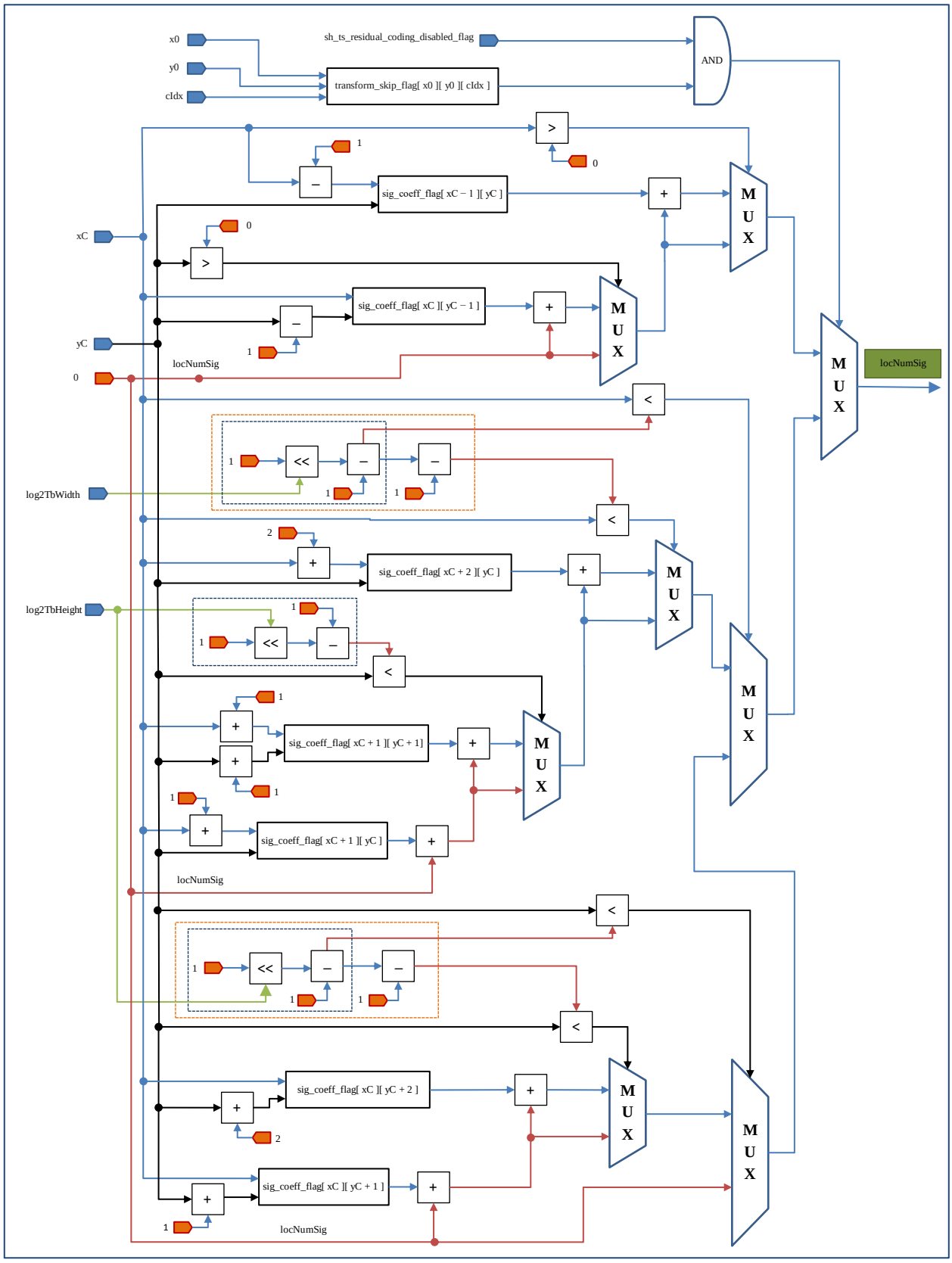


Figure IV.14– Architecture proposée de clause 7 (locNumSig).

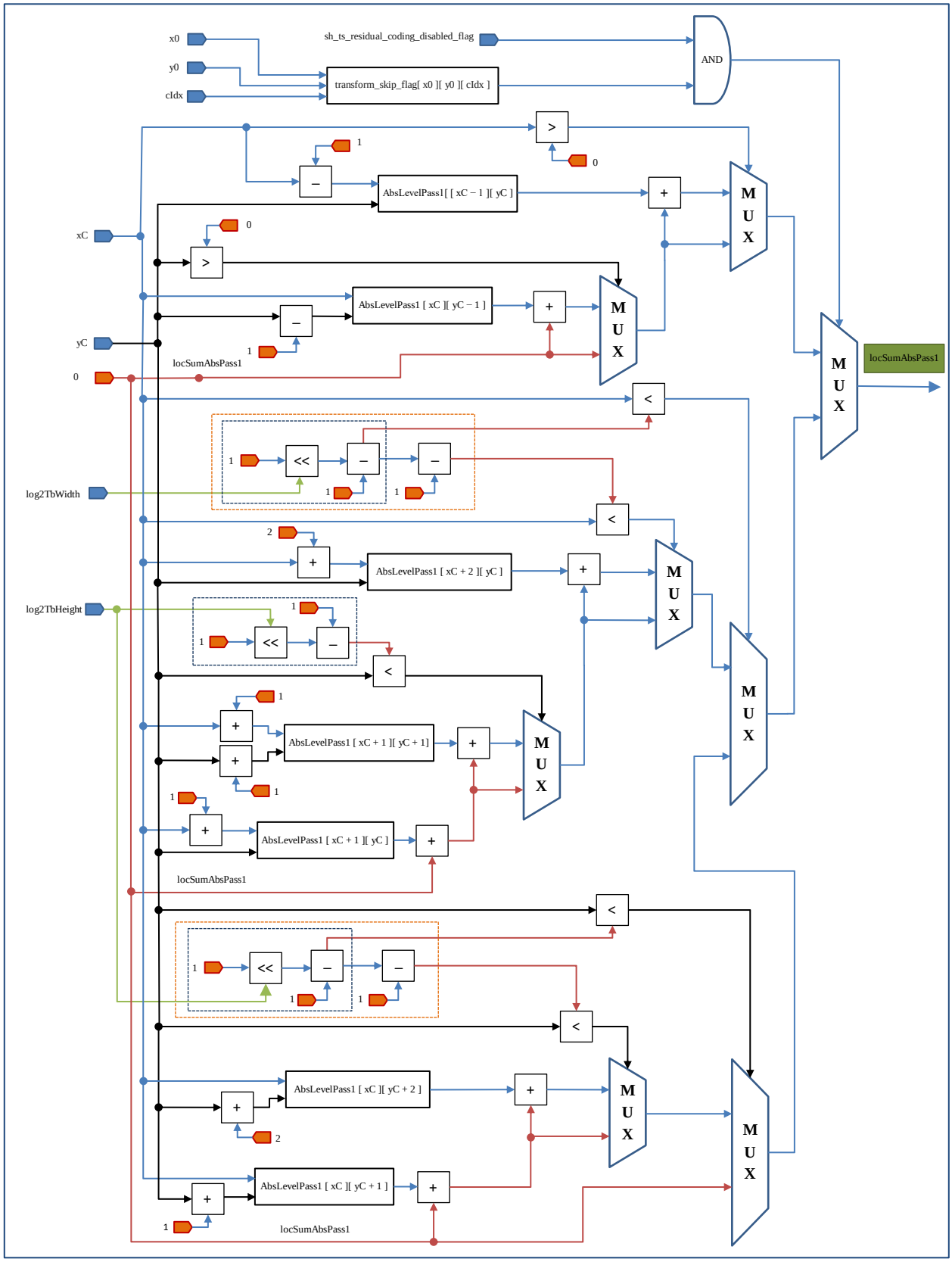


Figure IV.15– Architecture proposée de clause 7 (locSumAbsPass1).

IV.2.8 Architecture proposée pour CSM/clause 8

La figures IV.16 représente le calcul du *ctxInc* par la clause 8 dont un ensemble de variables et de valeurs fixes. Les schémas RTL et top level sont donnés sur la figure IV.17.

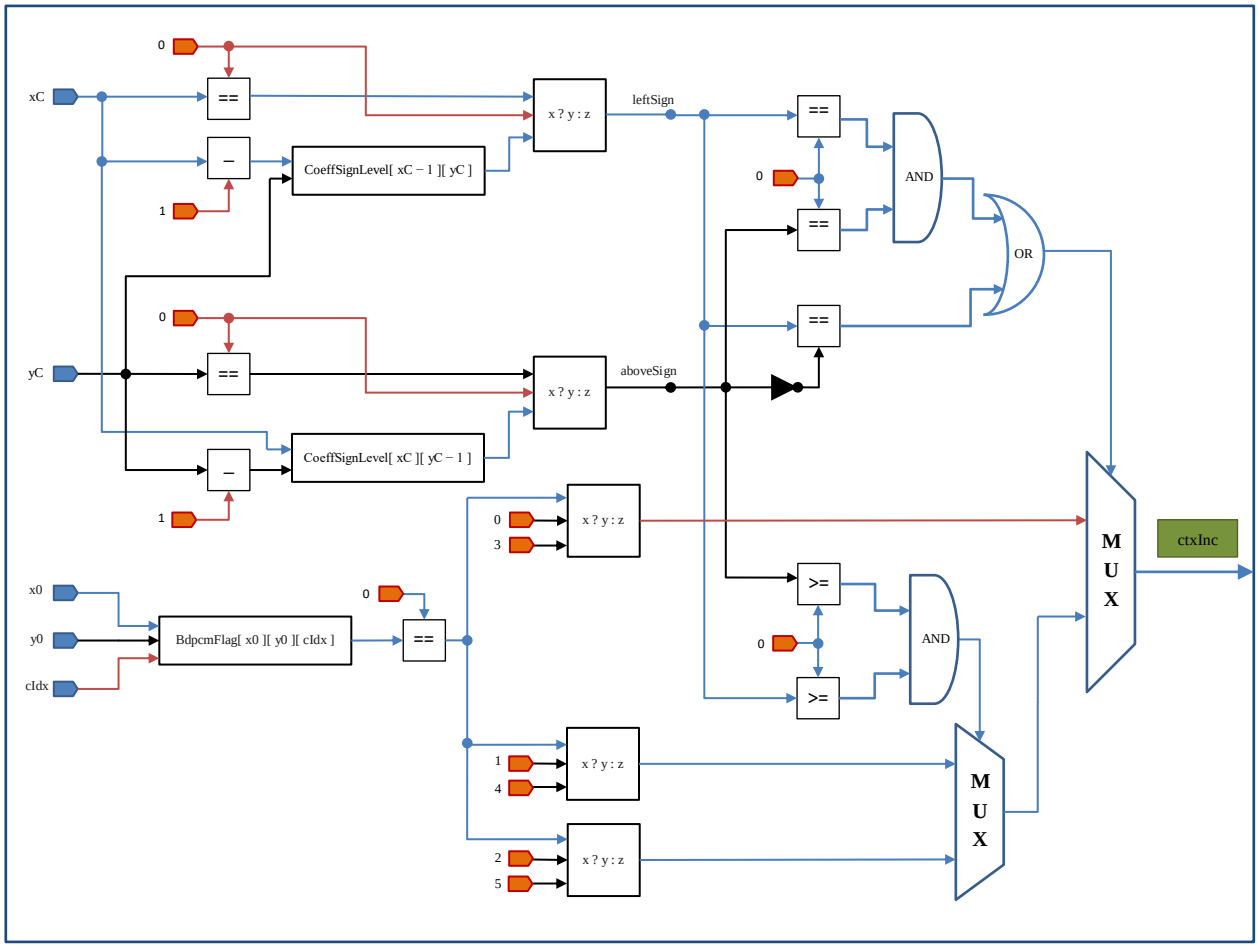


Figure IV.16– Architecture proposée de clause 8.

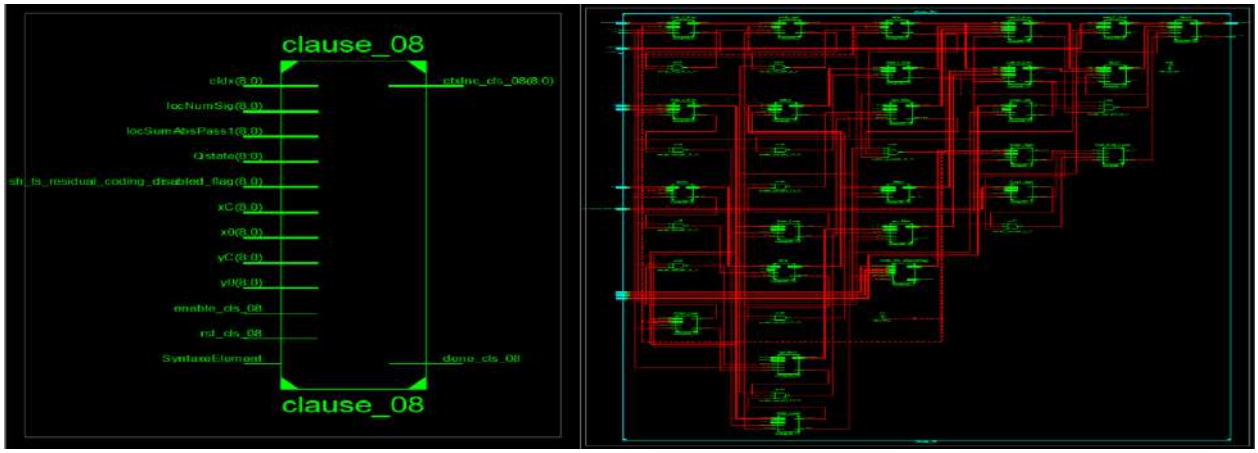


Figure IV.17– Schéma top level et le schéma RTL de la clause 8.

IV.2.9 Architecture proposée pour CSM/clause 9

La figures IV.18 représente les schémas RTL et le top level de cette clause. De plus, le calcul du *ctxInc* par la clause 9 dont un ensemble de variables et de valeurs fixes est donné sur la figure IV.19.

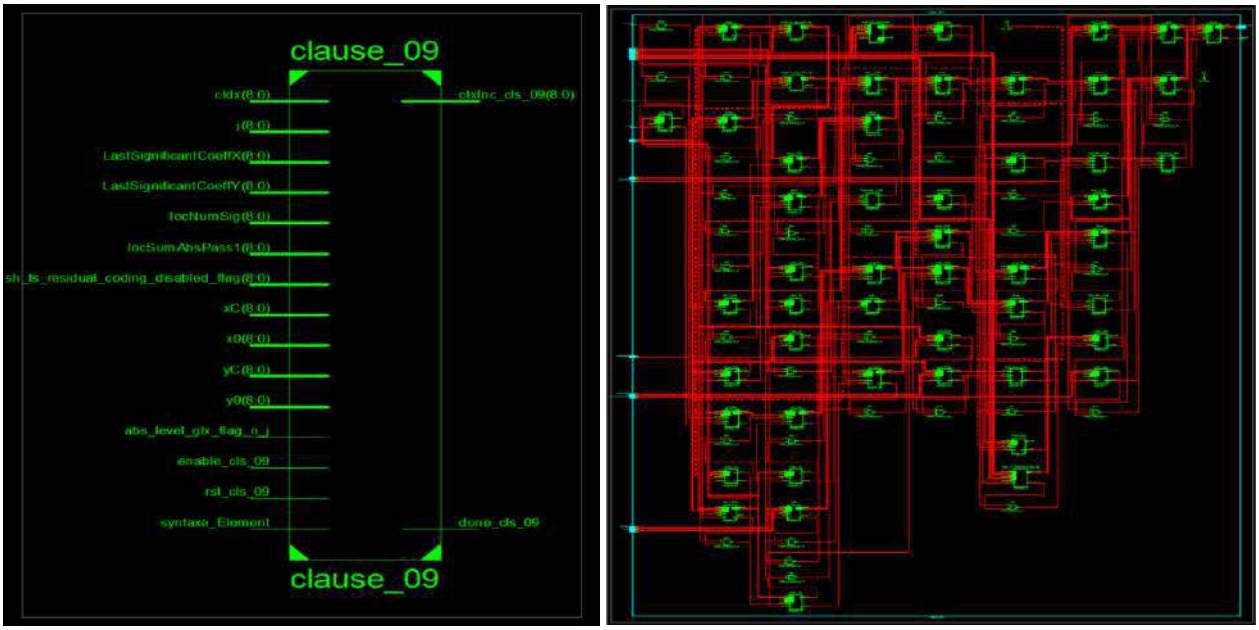


Figure IV.18– Schéma top level et le schéma RTL de la clause 9.

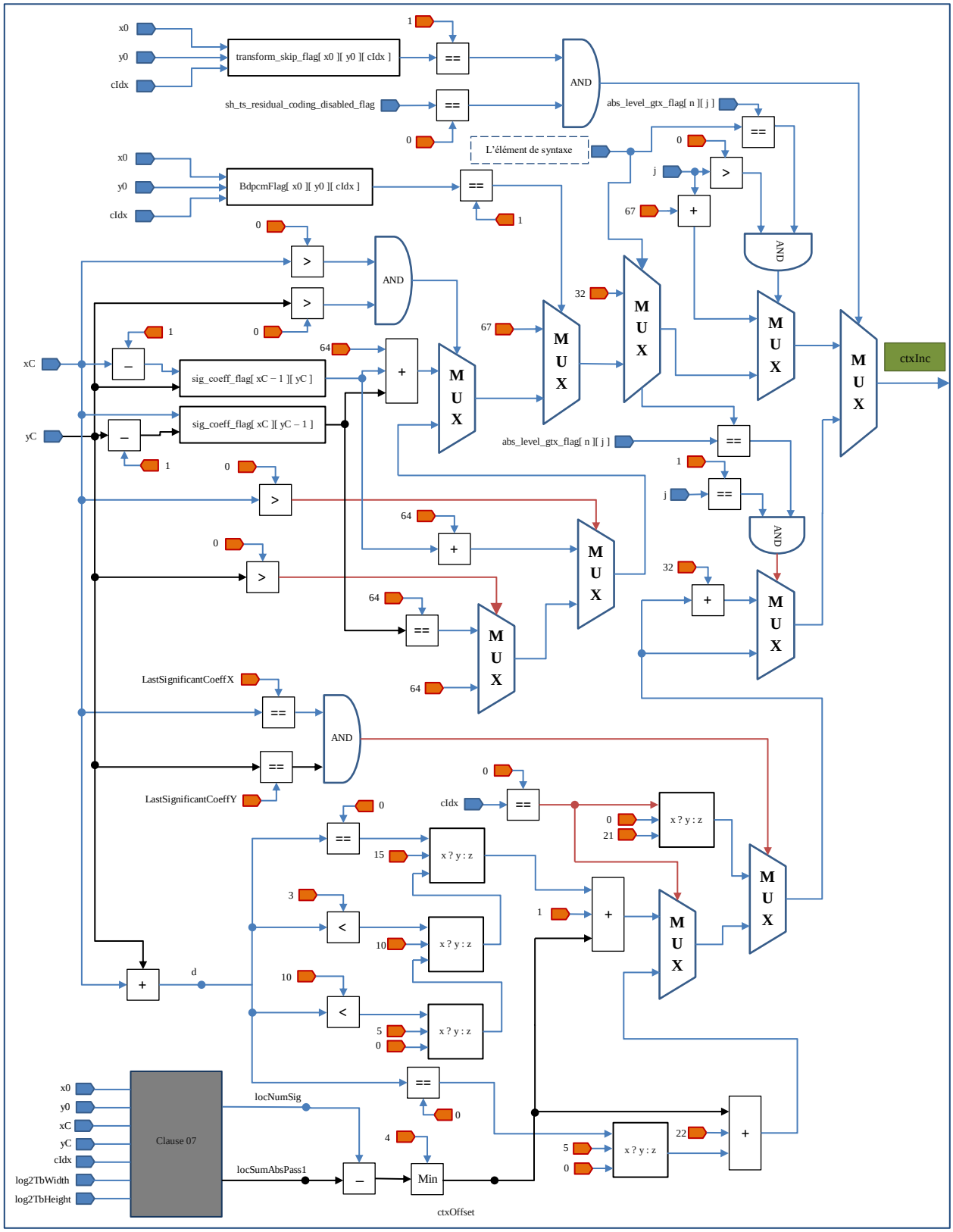


Figure IV.19– Architecture proposée de clause 9.

IV.2.10 Architecture proposée pour CSM/clause 10

La figures IV.20 représente le calcul du *ctxInc* par la clause 10 dont un ensemble de variables et de valeurs fixes. Les schémas RTL et le top level de la clause 10 sont donnés sur la figure IV.21.

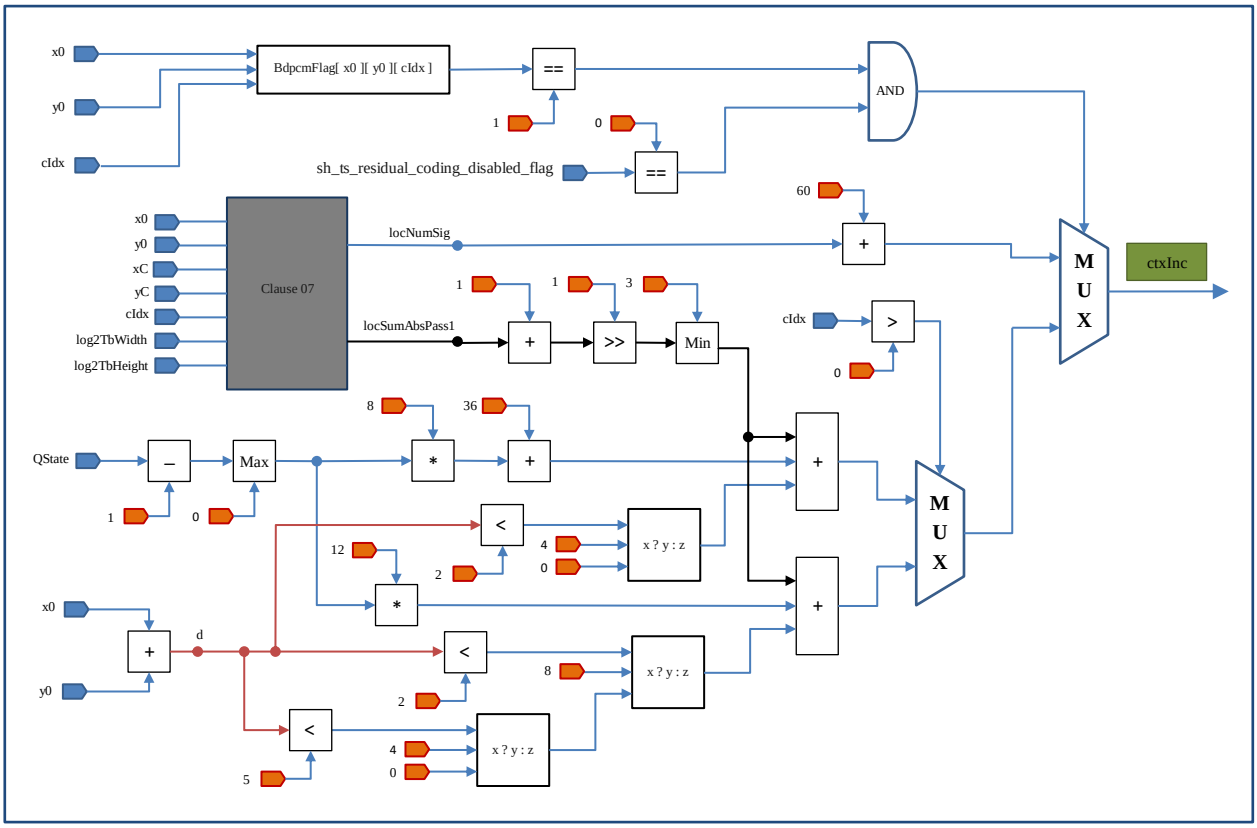


Figure IV.20– Architecture proposée de clause 10.

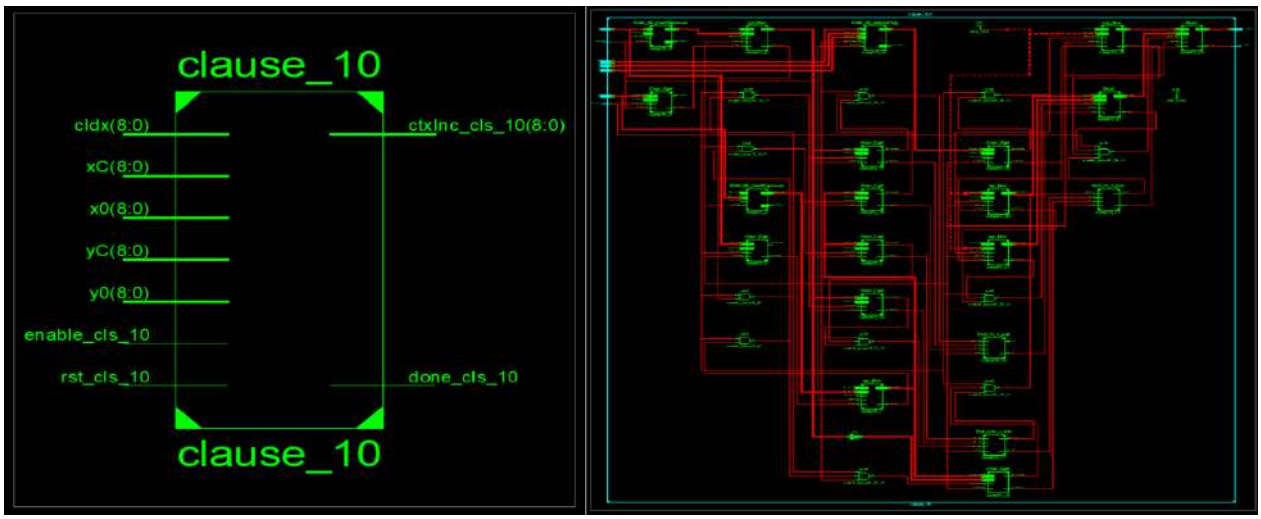


Figure IV.21– Schéma top level et le schéma RTL de la clause 10.

IV.2.11 Architecture proposée pour CSM/clause 11

La figures IV.22 représente le calcul du *ctxInc* par la clause 11 dont un ensemble de variables et de valeurs fixes. Les schémas RTL et le top level cette clause sont donnés sur la figure IV.23.

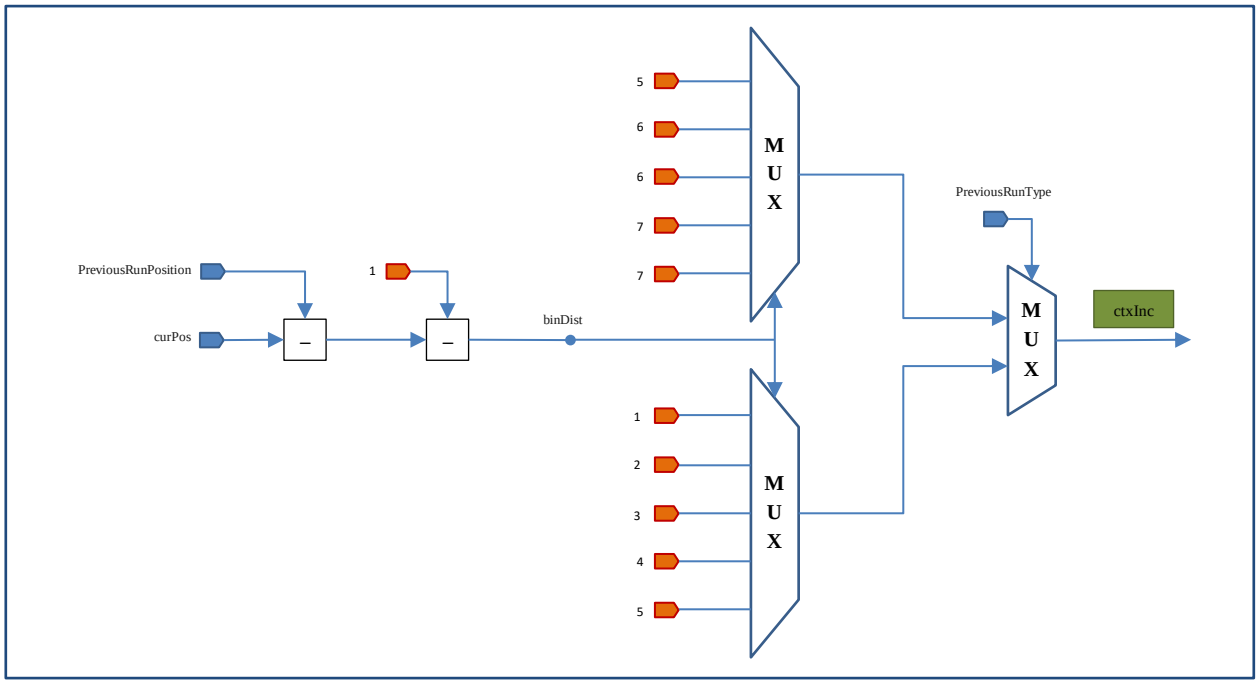


Figure IV.22– Architecture proposée de clause 11.

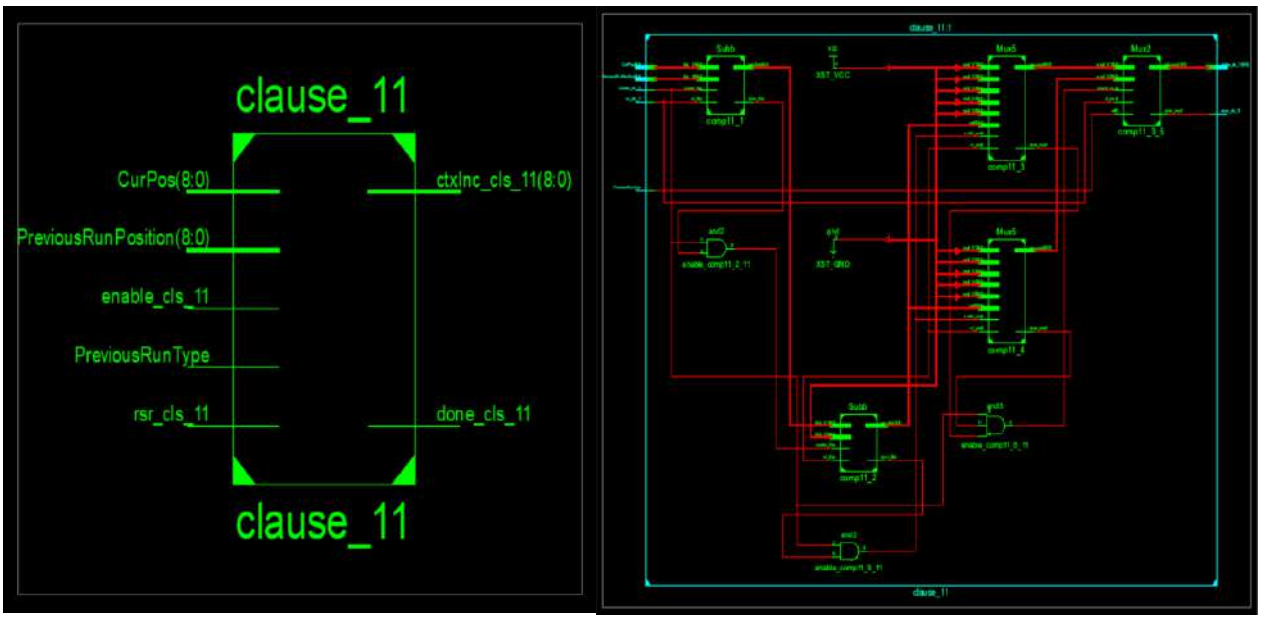


Figure IV.23– Schéma top level et le schéma RTL de la clause 11.

IV.2.12 Architecture Globale proposée du CSM/CABAC

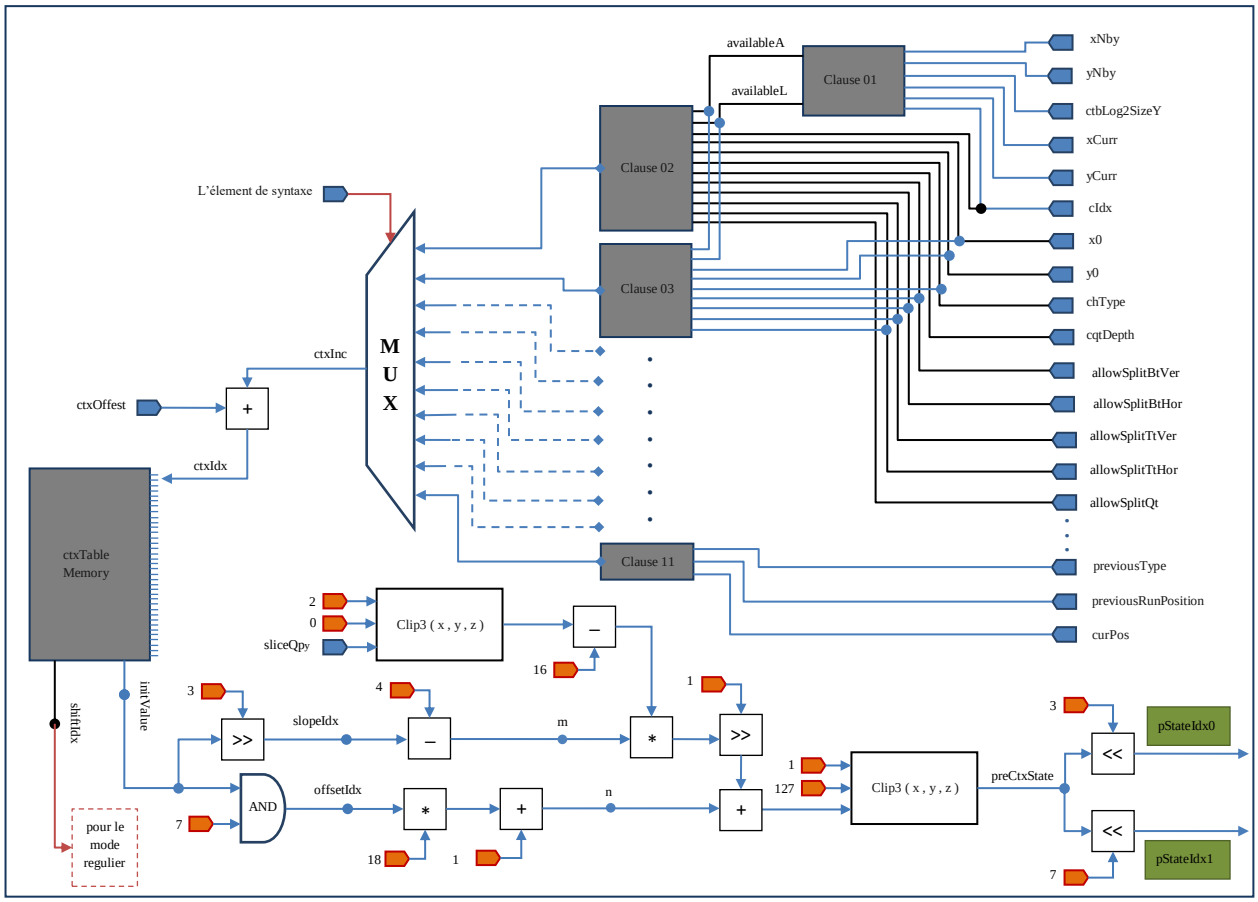


Figure IV.24– Architecture matérielle proposée pour la CSM/CABAC VVC.

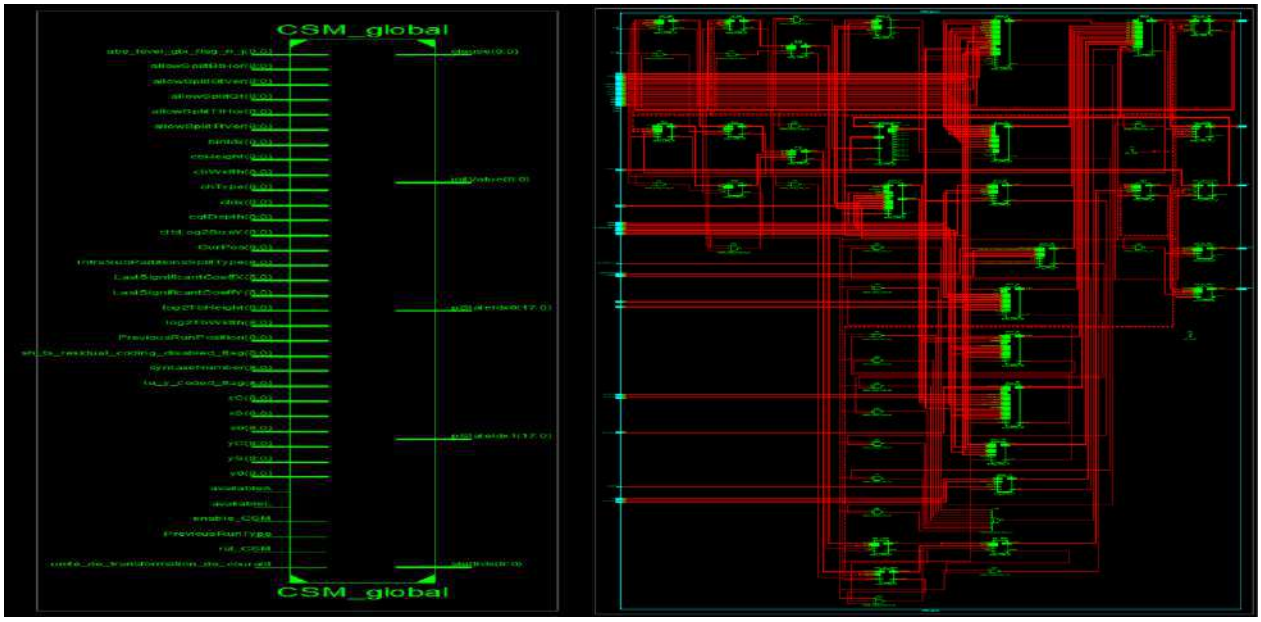


Figure IV.25– schémas RTL et le top level de l'architecture globale du CSM/CABAC.

Il est à noter que la CSM/CABAC VVC est basée sur 11 clauses qui sont spécifiées par la norme VVC pour les éléments syntaxiques traités par le mode régulier avec une méthode de calcul du contexte spécifique. Les schémas RTL et le top level de l'architecture globale du CSM/CABAC est donnés sur la figure IV.25.

IV.3 Synthèse des architectures proposées

Après la réalisation des architectures qui constitues la CSM/CABAC, nous avons passé à l'implémentation sur ISE en utilisant le langage VHDL, ainsi que la synthèse et la simulation sous ModelSim.

Comme notre travail est basé sur des architectures qui prend beaucoup d'espace, nous avons évité de mettre tous les schémas RTL obtenus sous l'ISE. La synthèse de la partie CSM/CABAC est donnée par le tableau IV.1.

Tableau IV.1– Synthèse de la partie CSM/CABAC

	Slice	LUT	Slice Flip Flop	Path delay (ms)
Clause 1	16	31	5	5.11
Clause 2	35	74	12	20.279
Clause 3	115	0	0	7.286
Clause 4	32	60	0	5.507
Clause 5	5	3	0	4.871
Clause 6	190	44	0	9.965
Clause 7	99	150	150	6.068
Clause 8	4	6	6	10.948
Clause 9	120	170	165	11.477
Clause 10	6	4	4	6.675
Clause 11	11	4	0	8.626
CSM Globale	639	556	345	4.871

IV.4 Validation des résultats obtenus

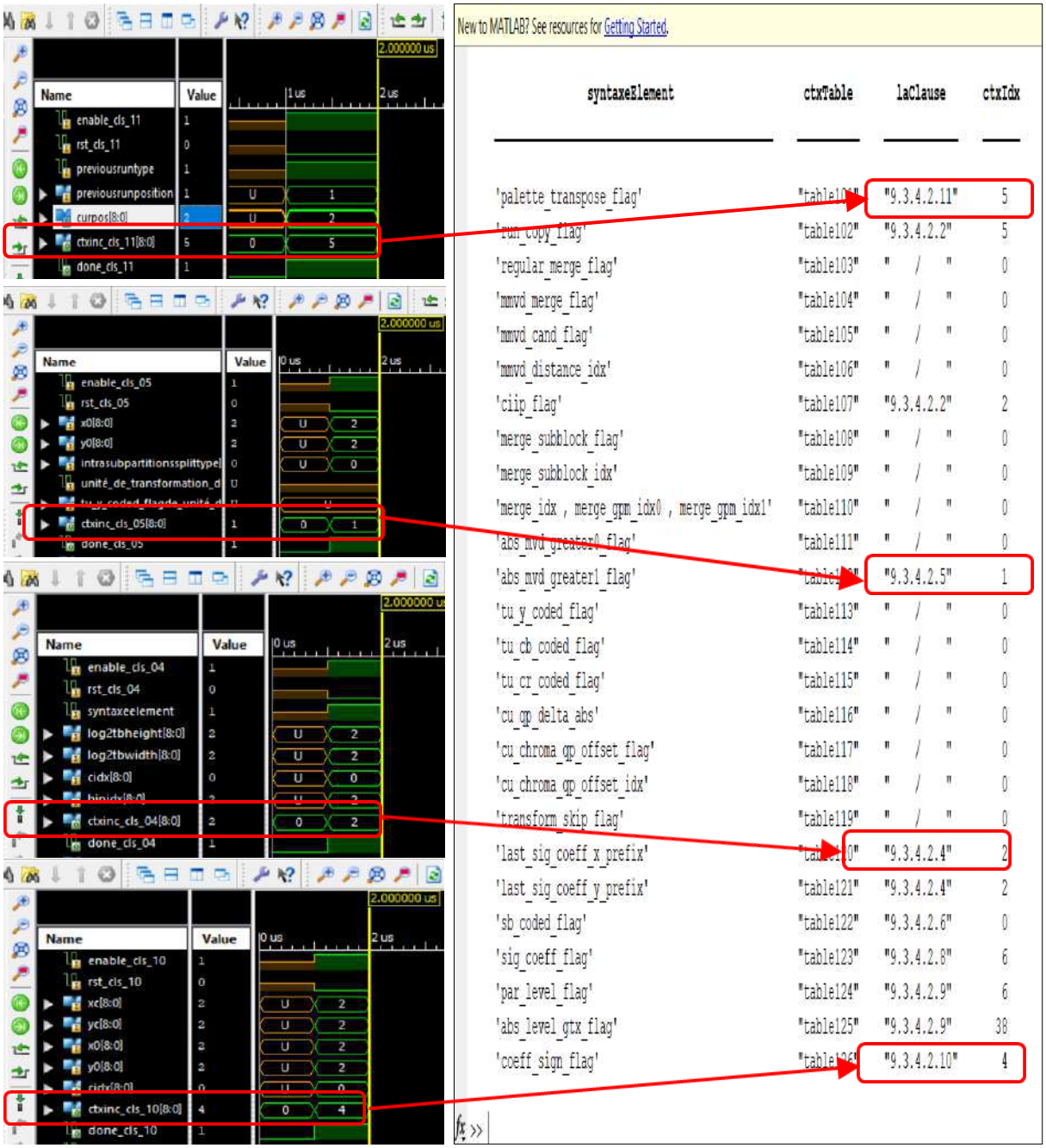


Figure IV.26– Comparaison de résultats des implémentations matérielle et logicielle

Afin de valider les résultats obtenus des clauses par le ModelSim, on a choisi le résultat de calcul du *ctxInc* des quatres clauses 4, 5, 10 et 11 obtenus par Matlab. D'après la figure IV.26 on peut voir clairement la similarité des résultats obtenus de notre algorithme par les deux types d'implémentation matérielle et logicielle.

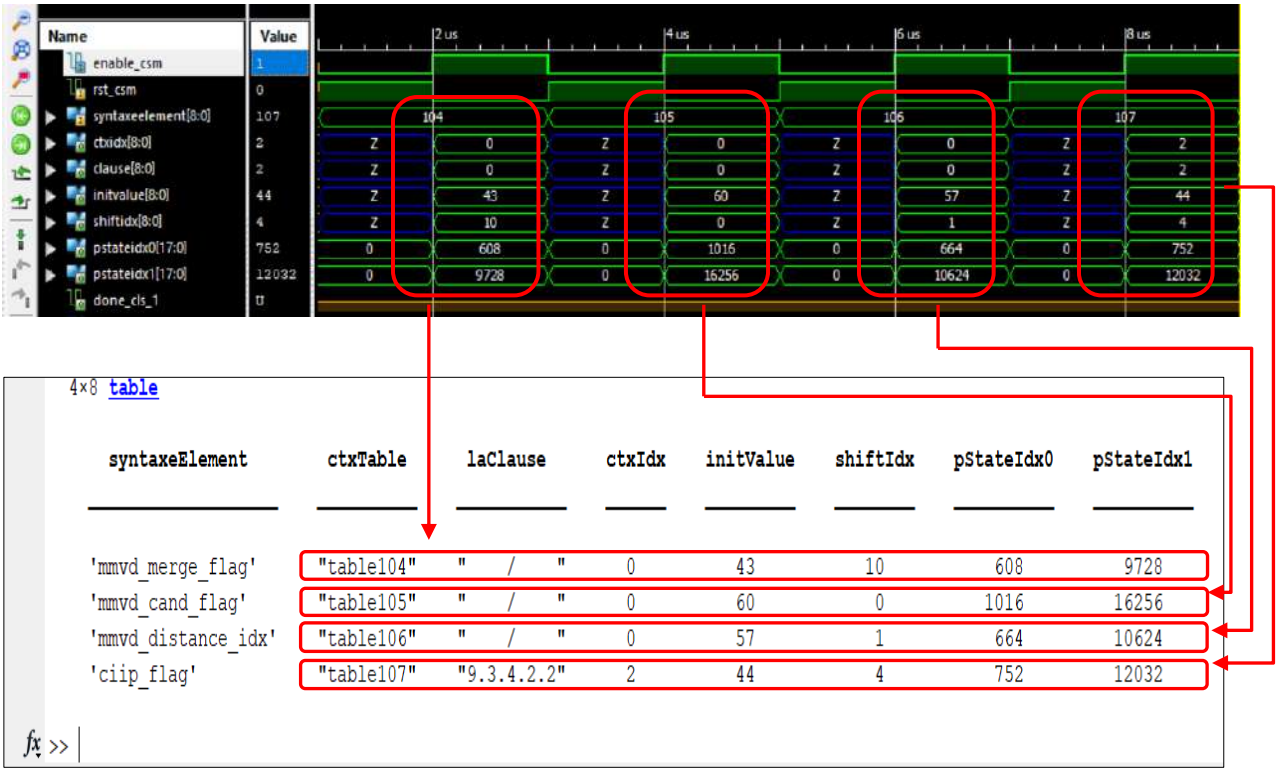


Figure IV.27– Comparaison de résultats des implémentations matérielle et logicielle des clauses selon leurs tables du contexte

Pour la CSM globale on a aussi comparer les résultats obtenus par ModelSim avec celles de Matlab pour les quatres éléments de syntaxes suivants :

- *mmvd_merge_flag ;*
- *mmvd_cand_flag ;*
- *mmvd_distance_idx ;*
- *clip_flag.*

Ces SEs utilisent les tables du contextes suivantes : 104, 105, 106 et 107. La validation des résultats matériels et logiciels est vérifier selon la figure IV.27.

IV.5 Conclusion

Dans ce de dernier chapitre nous avons présentés toutes les architectures proposées des 11 clauses, ainsi que l'architecture globale de la CSM. De plus, une synthèse suivi d'une émulation sous ModelSim est effectuée. A la fin du chapitre, une validation des résultats est réalisée en se basant sur les résultats matériels et logiciels.

Conclusion générale

Conclusion générale

La compression des vidéos est un processus indispensable avant les transmettre ou diffuser sur Internet, télévision, téléphones portables…etc.

L'objectif de notre travail était d'étudier, de concevoir et d'implémenter sur FPGA la partie sélection et modélisation du contexte pour un décodeur entropique CABAC dans la norme de compression vidéo VVC/H.266. Le décodeur CABAC VVC est basé sur trois blocs : BAD (décodage arithmétique binaire), CSM (sélection et modalisation de contexte) et la DB (Dé-binarisation).

Notre travail est organisé principalement en trois parties :

Dans la première partie, on a présenté en détaille les normes de compression vidéo récentes et les techniques de compression VVC. Suivi par une description de quelques notions de base sur le système de communication et la théorie de l'information. Par la suite, nous mettons en évidence le codage entropique et ses types plus précisément la partie sélection et modélisation du contexte pour un décodeur entropique CABAC VVC.

Dans la deuxième partie, on a conçu les organigrammes des codes du décodeur entropique CABAC avec leurs implémentations sous Matlab. Ce passage de l'implémentation logicielle sous Matlab nous a permis de maitriser les algorithmes de la partie sélection et modélisation du contexte pour un décodeur entropique CABAC utilisés dans la norme de compression vidéo VVC avant de passer à l'implémentation en langage VHDL sous l'ISE.

Dans la troisième partie, nous avons utilisé le logiciel ISE de Xilinx version 14.7 pour synthétiser notre architecture du CSM/CABAC VVC proposée décrite en langage de description matérielle HDL. Ensuite, nous avons utilisé le ModelSim pour vérifier l'architecture et confirmer les résultats obtenus avec celles de Matlab. L'architecture CSM fonctionne avec une fréquence de 156 MHz, un chemin critique de 4.871 ns et un nombre de ressource acceptable.

Comme perspectives nous proposons d'implémenter un décodeur CABAC complet avec un chemin critique optimisé et un débit (Throughput) améliorer en se basant sur des techniques de parallélisme et de pipeline afin d'assurer les contraintes temps réels.

Référence

[1] Cisco Visual networking Index. Forecast and methodology, 2018-2023, white paper. San Jose, CA, USA, 1, 2018.

[2] B. Bross, J. Chen, J. R. Ohm, G. J. Sullivan, and Y.-K. Wang. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). Proceedings of the IEEE, page 1–31, January 2021.

[3] A. Wieckowski, G. Hege, C. Bartnik, C. Lehmann, C. Stoffers, B. Bross, and D. Marpe. Towards a live software decoder implementation for the upcoming versatile video coding (vvc) codec. IEEE International Conference on Image Processing (ICIP), page 3124–3128, October 2020.

[4] Sze, V., Budagavi, M., Sullivan, G.J.: 'High efficiency video coding (hevc)', Integrated Circuit and Systems, Algorithms and Architectures Springer, 2014, 39, pp. 40.

[5] KTHIRI, M., 'Etude et Implantation d'Algorithmes de Compression Vidéo optimisés H.264/AVC dans un Environnement Conjoint Matériel et Logiciel', thèse de Doctorat, Université de Bordeaux 1, France, 4 Avril 2012.

[6] 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.

[7] PASCAL NSAM, Modélisation d'un système de transmission d'images animées sur réseau sans fil,Université du Québec à Trois-Rivières,1997. 151515

[8] Information technology—digital compression and coding of continuous-tone still images—part 1: Requirements and guidelines. In Recommendation ITU-T T.81 and ISO/IEC 10918-1. ITU-T and ISO/IEC JTC 1, 1992.

[9] Anthony Nasrallah. Nouvelles techniques de compression pour le codage vidéo prochaine-génération. Image Processing [eess.IV]. Institut Polytechnique de Paris, 2021. English. NNT : 2021IPPAT043.tel-03559752

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

[11]https://www.nextinpact.com/lebrief/43243/13017-le-h-266-officiellement-presente-des-fichiers-encore-moins-gros-a-qualite-constante

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

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

[14] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan. The joint exploration model (jem) for video compression with capability beyond hevc. IEEE Trans. Circuits Syst. Video Technol., 30(5):1208–1225, 2020.

[15] C. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423

[16] D. Marpe, H. Schwarz, T. Wiegand, Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard, IEEE Trans. Circuits Syst. Video Technol. 13 (7) (2003) 620–637.

[17] S. Golomb, Run-length encodings, IEEE Trans. Inf. Theory 12 (3) (1966) 399–401.

[18] D. Taubman, M. Marcelin, JPEG2000 Image Compression Fundamentals, Standards and Practice, Kluwer, 2002.

[19] David R. Bull Fan Zhang, INTELLIGENT IMAGE AND VIDEO COMPRESSION Communicating Pictures SECOND EDITION, 2021.

[20] DAMIAN KARWOWSKI, Precise Probability Estimation of Symbols in VVC CABAC Entropy Encoder, Faculty of Computing and Telecommunications, Poznan University of Technology, 60-965 Poznań, Poland, May 6, 2021.

[21] NEJI, N., '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.

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

[23] ITU-T, Telecommunication Standardization Sector, SERIES H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services – Coding of moving video, H.266 (08/2020).