THESE / UNIVERSITE DE BRETAGNE-SUD
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE DE BRETAGNE-SUD
Mention : STIC
Ecole doctorale SICMA

présentée par
Oussama ABASSI
Préparée à l'UMR 6285
Université de Bretagne-Sud
Lab-STICC

Etude des décodeurs LDPC non-binaires

Thèse soutenue le 27 juin 2014
devant le jury composé de :

Jean-François HELARD
Directeur de la recherche – INSA Rennes / *président*

Charly POULLIAT
Professeur des Universités – INP-ENSEEIHT / *rapporteur*
Christophe JEGO
Professeur des Universités – IPB/ENSEIRB-MATMECA / *rapporteur*

Pierre PENARD
Ingénieur Recherche et Développement – Orange Labs / *examinateur*

Laura CONDE-CANENCIA
Co-directeur de thèse
Emmanuel BOUTILLON
Directeur de thèse

à ma chère Maman
à mon cher Papa

Résumé

Les codes binaires à matrices creuses (Low-Density Parity-Check, LDPC) et les turbo-codes ont une capacité s'approchant de la limite de Shannon pour des trames de grande taille. Cependant, ces codes ont le désavantage d'être moins efficaces pour les faibles tailles. De plus, l'association de codes binaires avec des modulations d'ordre élevé oblige à une étape de marginalisation pour passer d'un symbole aux fiabilités des bits associés à ce symbole. Ce calcul s'accompagne d'une perte d'information qui peut être récupérée par démodulation itérative au prix d'une plus grande complexité. Les codes LDPC définis sur des corps de Galois d'ordre $q > 2$ permettent de résoudre ces problèmes. Toutefois, les décodeurs optimaux associés ont une complexité très importante qui rend leur utilisation problématique.

L'objectif de cette thèse est de valoriser les codes LDPC non binaires en proposant d'une part une architecture d'un décodeur à complexité réduite et en montrant d'autre part l'intérêt de les associer à des modulations d'ordre élevé.

Dans la première partie de notre thèse, nous proposons de simplifier l'algorithme de décodage Extended Min-Sum (EMS) en considérant un nombre limité $n_\alpha \ll n_m$ des fiabilités intrinsèques lors de la mise à jour des messages par les nœuds de variable. Cette approche permet de réduire la taille de la mémoire dédiée au stockage des messages intrinsèques. De plus, pour améliorer l'efficacité des nœuds de parité nous proposons une variante simplifiée de l'algorithme L-Bubble Check et l'architecture associée. Enfin, nous montrons par l'intermédiaire d'un prototype sur une carte FPGA (Field Programmable Gate Array) que notre décodeur possède une faible complexité en le comparant avec un ancien décodeur EMS conçu par notre laboratoire de recherche dans le cadre du projet européen DAVINCI.

Dans la deuxième partie, nous étudions l'association des codes LDPC non binaires avec une modulation par décalage cyclique de code (Cyclic Code-shift Keying, CCSK) de même ordre. Nous avons choisi cette modulation pour ses propriétés qui permettent de réduire la complexité du démodulateur. En effet, nous montrons qu'il est possible dans le cas d'un système de transmission mono-porteuse avec préfixe cyclique de fusionner le démodulateur et l'égaliseur dans un même bloc comportant une seule transformée de Fourier rapide et une seule transformée de Fourier rapide inverse. Les simulations montrent que ce système possède des performances comparables à un système de transmission multi-porteuses de type OFDM (Orthogonal Frequency-Division Multiplexing). Elles montrent aussi que la modulation CCSK donne des performances meilleures que la modulation de Hadamard dans un canal en environnement intérieur sélectif en fréquence. Enfin, les simulations montrent que les codes LDPC non binaires sont nettement plus efficaces avec la modulation CCSK que les codes LDPC binaires même en considérant une démodulation itérative.

Mots-clés : codes LDPC non binaires, algorithme de décodage EMS, modulations à étalement de spectre, modulation CCSK, égalisation fréquentielle.

Abstract

Binary Low-Density Parity-Check (LDPC) codes and turbo-codes are known to have near-capacity performance for long code lengths. However, these codes are less efficient for short and moderate code lengths. In addition, the combination of binary codes with high-order modulations requires a marginalization step to extract bits reliabilities from symbols reliablities. Thus, binary demodulation suffers from a loss of information that can be recovered using iterative demodulators at the expense of higher complexity. LDPC codes defined over finite fields of order $q > 2$ can be considered as a solution to these problems. Nevertheless, optimal decoding of non-binary LDPC codes suffers from extremely high complexity which almost prevents practical implementation.

In this thesis we aim at proving the feasibility of using non-binary LDPC codes in modern communication systems by proposing on the one hand a low-complexity decoder architecture based on a sub-optimal decoding algorithm, and showing on the other hand the advantages of combining such codes with high-order modulations.

In the first part of our thesis, we propose to simplify the Extended Min-Sum (EMS) algorithm by considering a limited number $n_\alpha \ll n_m$ of intrinsic reliabilities when updating messages at the variable nodes. This approach reduces the memory size required to save intrinsic messages. Furthermore, to improve the efficiency of the parity-check nodes, we propose a simplified variant of the L-Bubble Check algorithm with its architecture. Finally, we show through an FPGA (Field Programmable Gate Array) prototype that our decoder has significantly lower complexity when compared with a former EMS decoder designed by our research center within the framework of the European DAVINCI project.

In the second part, we focus on the combination of non-binary LDPC codes with Cyclic Code-Shift Keying (CCSK) modulation of the same order. We decided to study this modulation technique due to its properties that enable reducing the receiver complexity. Indeed, we show that it is possible, in the case of a single-carrier system with cyclic-prefix, to merge the demodulator and the equalizer in a single block comprising one Fast Fourier Transform (FFT) and one inverse FFT only. The simulations show that this single-carrier system has similar performance as the multi-carrier Orthogonal Frequency-Division Multiplexing (OFDM) system. They also show that CCSK modulation offers better performance than Hadamard modulation in a frequency-selective indoor channel. Finally, simulations demonstrate that non-binary LDPC codes are much more efficient when combined with CCSK signalling than binary LDPC codes even if we consider an iterative demodulation.

Index Terms : non-binary LDPC codes, EMS decoding algorithm, spread spectrum modulation, CCSK signalling, frequency-domain equalizer.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

$\mathbb{C}$ Ensemble des nombres complexes
$\mathbb{N}$ Ensemble des entiers naturels
$\mathbb{R}$ Ensemble des nombres réels
$\mathbb{Z}$ Ensemble des entiers relatifs
3GPP The 3rd Generation Partnership Project
APP A Posteriori Probability
ASIC Application-Specific Integrated Circuit
ASK Amplitude-Shift Keying
AWGN Additive White Gaussian Noise
BCGR Bahl Cocke Jelinek Raviv
BICM Bit-Interleaving Coded Modulation
BICM-ID Bit-Interleaving Coded Modulation with Iterative Decoding
BP Belief Propagation
BPSK Binary Phase-Shift Keying
CAM Content Adressable Memory
CCSK Cyclic Code-Shift Keying
CN Check Node
CNP Check Node Processor
CP Cyclic Prefix
DAVINCI Design And Versatile Implementation of Non-binary wireless Communications based on Innovative LDPC codes
DSSS Direct-Sequence Spread Spectrum
DVB Digital Video Broadcast
ECN Elementary Check Node
EMS Extended Min-Sum
FDE Frequency-Domain Equalization
FFT Fast Fourier Transform
FHSS Frequency-Hopping Spread Spectrum
FIFO First-In First-Out
FPGA Field Programmable Gate Array
GF Galois Field
GNSS Global Navigation Satellite Systems
HSPA High Speed Packet Access
IFFT Inverse Fast Fourier Transform
JTIDS Joint Tactical Information Distribution System

Lab-STICC Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance
LDPC Low-Density Parity-Check
LFSR Linear Feedback Shift Register
LLR Log-Likelihood Ratio
LTE Long Term Evolution
LUT Look Up Table
MAP Maximum A posteriori Probability
ML Maximum Likelihood
MLSE Maximum Likelihood Sequence Estimation
MMSE Maximum Mean Square Error
OFDM Orthogonal Frequency-Division Multiplexing
OM Orthogonal Modulation
PAPR Peak-to-Average Power Ratio
PSK Phase-Shift Keying
QAM Quadrature Amplitude Modulation
RAM Random Access Memory
ROM Read Only Memory
SC Single Carrier
TCM Treillis Coded Modulation
TEB Taux d'Erreur Binaire
TEP Taux d'Erreur Paquet
TES Taux d'Erreur Symbole
VN Variable Node
VNP Variable Node Processor
WiFi Wireless Local Area Network
WiMAX Worldwide Interoperability for Microwave Access
ZF Zero Forcing

Introduction générale

La transmission fiable de l'information sur des canaux bruités est l'une des exigences de base des systèmes de communication. En raison de cette exigence, les systèmes de communication modernes (DVB, 3GPP, WiMAX...) [4, 5, 6] s'appuient fortement sur les codes de correction d'erreurs pour détecter et corriger les erreurs de transmission causés par le bruit et les interférences dans les canaux de communication ou des imperfections dans les supports de stockage. Ces codes sont utilisés pour assurer une transmission de données robuste et fiable à travers des canaux imparfaits. Shannon a démontré dans ses travaux pionniers sur la théorie de communication que le codage de correction d'erreurs n'est possible qu'en rajoutant de la redondance au message à transmettre [7].

Nous distinguons deux grandes familles de codes de correction d'erreurs selon la manière avec laquelle la redondance est ajoutée : les codes en blocs et les codes convolutifs [8]. Dans le premier cas, le message est d'abord divisé en blocs de données et le codeur traite ces blocs séparément. Par conséquent, l'encodeur doit attendre la réception d'un bloc pour démarrer. Dans le deuxième cas, l'encodeur traite le message de façon continue et génère séquentiellement les symboles de redondance sans avoir besoin du message complet. L'essentiel des standards de communication actuels se situent dans l'une de ces deux grandes familles de codes de correction d'erreurs. Néanmoins, exploiter ce type de codes, compte tenu des contraintes applicatives fortes en termes de débit de transmission et de taux d'erreur, n'est pas chose aisée et demande beaucoup d'expertise. En effet, les concepteurs doivent parvenir à développer des architectures capables d'atteindre les performances applicatives voulues à moindre coût en termes de temps de conception, surface matérielle et consommation d'énergie.

Codes en blocs

Dans le codage en bloc, le flux de données est réparti en segments, ou blocs, de symboles. Chaque bloc de message noté ζ contient K symboles d'information et il en résulte 2^K mots de codes possibles. La fonction de codage consiste à ajouter de la redondance au message d'information ζ afin de générer un mot de code noté c de taille N symboles avec $N > K$. Pour utiliser ce code en bloc dans la pratique, il est nécessaire que les 2^K mots de codes soient différents. Transformer le code en bloc en 2^K mots de codes est une opération qui consomme une grande partie des ressources mémoire de l'encodeur puisqu'il faut stocker les 2^K mots de codes. Afin de réduire ce besoin, les applications utilisent dans la pratique les codes en blocs linéaires.

Un code en bloc linéaire est une classe de codes en bloc dans lesquels la somme de deux mots de code *modulo* 2 est également un mot de code. Dans cette classe de codes, les mots de codes sont générés par une matrice génératrice notée G de dimensions $K \times N$. Les K

lignes de G sont des mots de codes linéairement indépendants. Un mot de code c est le résultat de la multiplication d'un message d'information ζ par la matrice génératrice : $C = \zeta \cdot G$. L'implémentation de l'encodeur peut être simplifiée en introduisant une structure systématique durant la construction des codes en bloc linéaires. Dans cette structure, le mot de code est divisé en deux parties : la première correspond au message d'information ζ et la seconde correspond aux symboles de parités. Ce type de codes est appelé *code en bloc linéaire systématique*. La matrice génératrice est dans ce cas construite par la concaténation d'une matrice aléatoire notée P de dimensions $K \times (N - K)$ et d'une matrice identité notée I_d de dimension K : $G = P|I_d$. La sous-matrice I sert à reproduire le message d'information ζ à la fin du mot de code tandis que la sous-matrice P sert à générer les symboles de redondances. Un exemple d'une matrice génératrice d'un code systématique de taille $N = 7$ bits à partir d'un message d'information de taille $K = 4$ bits est donné ci-dessous :

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Soit $\zeta = (\zeta_0, \zeta_1, \zeta_2, \zeta_3)$ le message à encoder. Le mot de code $c = (c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ est obtenu comme suit :

$$\begin{cases} c_0 & = \zeta_0 \oplus \zeta_2 \oplus \zeta_3 \\ c_1 & = \zeta_0 \oplus \zeta_1 \oplus \zeta_2 \\ c_2 & = \zeta_1 \oplus \zeta_2 \oplus \zeta_3 \\ c_3 & = \zeta_0 \\ c_4 & = \zeta_1 \\ c_5 & = \zeta_2 \\ c_6 & = \zeta_3 \end{cases}$$

où $\oplus$ est l'opérateur logique *XOR*.

En exploitant ces équations, l'encodeur peut être implanté, comme illustré dans la figure 0.1, par quatre registres servant à stocker les bits d'informations, trois registres servant à stocker les bits de redondances et trois portes logiques de type *XOR*.

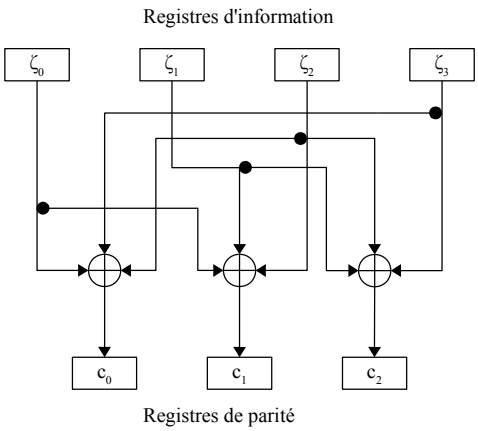


FIGURE 0.1 – Architecture d'un encodeur en bloc systématique

Soit C l'ensemble de tous les mots de codes et $C^{\perp}$ le dual de C définie par $C^{\perp} = \{u \in \{0,1\}^N$ tel que $\forall c \in C, \langle u|c\rangle = \sum_{i=0}^{N-1} u_i \cdot c_i = 0\}$. La matrice génératrice du dual $C^{\perp}$ notée H est appelée la matrice de parité du code en bloc linéaire C. La matrice de parité est de dimensions $(N-K) \times N$ et est reliée à la matrice G par $G \cdot H^T = 0$, H^T étant la matrice transposée de H. A partir de la relation précédente nous pouvons établir la relation servant de base à la détection d'erreurs : $c \cdot H^T = \zeta \cdot G \cdot H^T = 0$. Le décodeur va utiliser cette relation pour détecter les erreurs. Si le mot de code reçu Y est entaché d'erreur nous pouvons l'exprimer sous la forme $y = c + \eta$ avec η l'erreur de transmission. Le syndrome de l'erreur notée S_H est définie par $S_H = y \cdot H^T = \eta \cdot H$. Si $S_H \neq 0$ le décodeur détecte la présence d'erreurs dans le mot de code reçu.

Codes LDPC

Les codes LDPC (Low-Density Parity-Check), également appelés les codes de Gallager, sont une classe de codes en bloc linéaires inventés en 1962 [9]. Un code LDPC est caractérisé par une matrice de parité creuse, c'est-à-dire ne contenant qu'un très faible nombre d'éléments non nuls. Malgré ses très bonnes performances, cette classe de codes a été écartée pendant trois décennies en raison de sa complexité de décodage dépassant la capacité des systèmes électroniques de l'époque. Le procédé de décodage des codes LDPC sera entièrement décrit dans le chapitre 1.

Ce furent Mackay et al. qui ont tiré de l'oubli les codes LDPC au milieu des années 1990 [10, 11]. L'importance du codage LDPC a encore pris de l'ampleur en poussant sa capacité dans [12] à seulement $0.0045dB$ de la limite de Shannon, ce qui en fait le meilleur code de correction d'erreurs connu jusqu'ici. Cependant, pour atteindre ces performances il faut considérer des codes de grande taille (de l'ordre de 10^6 bits) au prix d'une augmentation considérable de la surface d'implantation des décodeurs.

Il est montré dans [13] que généraliser la définition des codes LDPC sur des corps de Galois d'ordre strictement supérieur à 2 permet d'améliorer considérablement les performances des codes de petites tailles. Ces codes sont dits non binaires. Toutefois, la complexité importante des décodeurs associés rend leur utilisation problématique. Récemment, le nombre de publications proposant des décodeurs LDPC non binaires de plus en plus efficaces ne cesse de croître ce qui montre l'intérêt que porte la communauté scientifique pour cette famille de codes. De plus, des chercheurs s'intéressent à la bonne adéquation qui existe entre les codes LDPC non binaires et les modulations d'ordre élevé ce qui permet de réaliser des gains importants.

Dans cette thèse, nous proposons dans un premier temps un décodeur LDPC non binaire à faible complexité et nous étudions dans un deuxième temps les avantages d'associer les codes LDPC non binaires à la modulation CCSK.

Organisation du manuscrit

Ce manuscrit de thèse est composé de trois chapitres :

(a) Le chapitre 1 a pour but de présenter le concept des codes LDPC non binaires et des algorithmes de décodage itératifs associés. Dans un premier temps, nous présentons les pré-requis à la compréhension de la théorie des ensembles finis (théorie de

Galois). Ensuite, nous donnons une description détaillée des codes LDPC construits sur des corps de Galois d'ordre $q = 2^m$, $m > 0$. Nous détaillons aussi les principaux algorithmes de décodages associés à cette classe de codes correcteurs d'erreurs. Enfin, nous consacrons la dernière section de ce chapitre à la description de l'architecture matérielle du décodeur LDPC non binaire développée dans le cadre du projet européen DAVINCI.

(b) Dans le chapitre 2 nous proposons une architecture optimisée du décodeur LDPC non binaire conçu dans le cadre du projet européen DAVINCI. Nos améliorations ont donné lieu à une architecture plus efficace en termes de surface et de débit. Dans la dernière section de ce chapitre nous présentons les résultats de synthèse d'un prototype FPGA (Field Programmable Gate Array) de notre décodeur et nous comparons ces résultats avec ce qui existe dans l'état de l'art.

(c) Dans le Chapitre 3 nous étudions l'association des codes LDPC non binaires avec la modulation CCSK (Cyclic Code-Shift Keying). Dans un premier temps, nous présentons les inconvénients d'associer les modulations d'ordre élevé à des codes binaires, ce qui nous permet de montrer l'intérêt d'associer directement ces modulations à des codes non binaires de même ordre. Nous nous intéressons ensuite à l'étude de l'association des codes LDPC non binaires à la modulation CCSK. Nous proposons d'abord les architectures des démodulateurs associés dans différents systèmes de transmission. Par la suite, nous analysons les résultats de simulations obtenus sur des modèles de canaux sans fil.

Nous terminons ce manuscrit par un bilan de nos contributions et les perspectives pour la suite de nos travaux.

Chapitre 1

Les codes LDPC non binaires : concepts et architectures

Sommaire

Ce chapitre est une introduction aux décodage LDPC non binaire. Dans la Section 1.1, nous introduisons la notion de corps de Galois nécessaire à la définition des codes LDPC non binaires de la Section 1.2. La Section 1.3 est consacrée à l'étude des principaux algorithmes de décodage des codes LDPC non binaires. Enfin, dans la Section 1.4 nous décrivons l'architecture du décodeur LDPC non binaire conçue dans le cadre du projet européen DAVINCI.

1.1 Introduction aux corps de Galois

L'algèbre moderne se caractérise par un niveau d'abstraction élevée. En effet, l'algèbre classique étudie des ensembles, de type $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ et $\mathbb{C}$, construits avec des opérations arithmétiques telles que l'addition et la multiplication. Quant à l'algèbre moderne, la notion d'*opération* (ou *loi de composition*) prend une dimension plus complexe et se définie comme étant une application qui, dans des ensembles généralisés, associe à deux ou plusieurs *symboles* un autre symbole. La théorie de codage a tiré profit de cette abstraction afin d'étendre la définition des codes correcteurs d'erreurs à des ensembles autres que les ensembles classiques cités ci-dessus. Dans ce manuscrit de thèse, nous nous intéressons en particulier au cas des codes LDPC non binaires définis sur les corps de Galois (Galois Fields en anglais). Dans le but de donner une définition complète des corps de Galois, nous commençons par décrire les structures algébriques de base munies de lois de composition interne. Le contenu de cette section a été principalement extrait de [14, 15, 16, 17].

1.1.1 Les structures algébriques

Soient E et K deux ensembles.

Définition 1.1. *Une loi de composition interne sur E est une application qui associe à un couple (x, y) dans $E \times E$ un élément z dans E.*

Définition 1.2. *Une loi de composition externe sur E est une application qui associe à un couple (x, y) dans $K \times E$ un élément z dans E.*

Une loi de composition est généralement notée « $*$ ». Nous distinguons particulièrement la loi additive notée « $+$ » et la loi multiplicative notée « $\cdot$ ».

Nous appelons *composé* d'un élément x par un élément y, l'unique élément $x * y$ associé par la loi « $*$ » au couple (x, y).

Définition 1.3. *Une structure algébrique de base est un ensemble muni d'une ou plusieurs lois de composition interne.*

Définition 1.4. *Une structure algébrique S est dite finie si elle contient un nombre fini d'éléments. Le nombre d'éléments de S est alors noté $|S|$ et est appelé ordre de la structure algébrique.*

1.1.2 Les groupes

Définition 1.5. *Un groupe est un ensemble G muni d'une loi de composition interne « $*$ » telle que :*
◇ *« $*$ » est associative : $\forall a, b, c \in G, (a * b) * c = a * (b * c)$*
◇ *« $*$ » admet un élément neutre $e \in G$: $\forall a \in G, a * e = e * a = a$*
◇ *$\forall a \in G$, a possède un élément symétrique $b \in G$: $a * b = b * a = e$*
Le groupe G est dit abélien (en l'honneur de Niels Abel) si de plus « $*$ » est commutative : $\forall a, b \in G, a * b = b * a$.

L'élément neutre e est unique. De plus, $\forall a \in G$, son symétrique b est unique. L'associativité de la loi de composition garantit que l'expression $a_1 * a_2 * \cdots * a_n$ possède un sens puisqu'elle représente un élément unique de G indépendamment de la position des parenthèses.

Un groupe G est dit additif si nous utilisons la notation additive de la loi de composition. L'élément symétrique de a (ou l'opposé de a) est alors noté $-a$ et l'élément neutre est noté 0. Dans le cas où nous utilisons la notation multiplicative, le groupe est dit multiplicatif, l'élément symétrique de a (ou l'inverse de a) est noté a^{-1} et l'élément neutre est noté 1.

Nous utilisons les conventions suivantes pour indiquer le composé de n-fois d'un élément x avec lui même :
◇ Notation additive : $nx = x + x + \cdots + x$
◇ Notation multiplicative : $x^n = x \cdot x \cdot \ \cdots \ \cdot x$

Le tableau 1.1 donne quelques règles conventionnelles relatives aux deux notations multiplicative et additive.

TABLE 1.1 – Règles conventionnelles des deux notations multiplicative et additive

Notation multiplicative	Notation additive
$a^0 = 1$	$0a = 0$
$a^{-n} = (a^{-1})^n$	$(-n)a = n(-a)$
$a^{n+m} = a^n \cdot a^m$	$(n+m)a = na + ma$
$a^{nm} = (a^n)^m$	$(nm)a = n(ma)$

Les opérations de soustraction et de division sont définies comme fonction de l'élément symétrique :
- ◇ Soustraction : $a - b = a + (-b)$
- ◇ Division : $\frac{a}{b} = a \cdot b^{-1}$

1.1.3 Les anneaux

Définition 1.6. *Un anneau $(A, +, \cdot)$ est un ensemble muni de deux lois de composition interne « + » et « · » telles que :*
- ◇ *A muni de « + » est un groupe abélien.*
- ◇ *« · » est associative : $\forall a, b, c \in A, (a \cdot b) \cdot c = a \cdot (b \cdot c)$*
- ◇ *« · » est distributive par rapport à « + » : $\forall a, b, c \in A, (a + b) \cdot c = a \cdot c + b \cdot c$ et $c \cdot (a + b) = c \cdot a + c \cdot b$*
- ◇ *« · » possède un élément neutre.*
- ◇ *A est dit commutatif si « · » est commutative : $\forall a, b \in A, a \cdot b = b \cdot a$*

L'élément neutre de « + » est noté 0 et celui de « · » est noté 1. Nous utilisons par convention les notations « + » et « · » pour indiquer que les deux lois de composition interne d'un anneau satisfont certaines des propriétés de l'addition et la multiplication des nombres entiers relatifs. Cependant, nous devons toujours garder en esprit la définition d'une loi de composition donnée dans la sous-section 1.1.1.

1.1.4 Congruence et arithmétique modulaire dans $\mathbb{Z}$

Définition 1.7. *Soient a et b des entiers quelconques, et n un entier strictement positif.*
- ◇ *Nous disons que a est congru à b* modulo *n si n divise $a - b$. Nous utilisons la notation $a \equiv b \pmod{n}$*
- ◇ *$a \equiv b \pmod{n}$ signifie d'une manière équivalente que b est le reste de la* division euclidienne *de a par n. Nous utilisons la notation $b = a \bmod n$*
  Nous obtenons alors les équivalences suivantes :
  $a \equiv b \pmod{n} \Leftrightarrow \exists k \in \mathbb{Z}$ *tel que* $a = b + k \cdot n \Leftrightarrow b = a \bmod n$
- ◇ *L'opération d'addition de a et b* modulo *n est par définition :*
  $a \oplus_n b = (a + b) \bmod n$
- ◇ *L'opération de multiplication de a et b* modulo *n est par définition :*
  $a \otimes_n b = (a \cdot b) \bmod n$

En particulier, comme nous pouvons le constater des tableaux 1.2 et 1.3, les opérations d'addition et de multiplication dans l'ensemble $\mathbb{Z}_2 = \{0, 1\}$ correspondent aux deux fonctions logiques *XOR* et *AND*.

TABLE 1.2 – Addition *modulo* 2

$\oplus_2$	0	1
0	0	1
1	1	0

TABLE 1.3 – Multiplication *modulo* 2

$\otimes_2$	0	1
0	0	0
1	0	1

Il est simple de vérifier que d'une manière générale l'ensemble $\mathbb{Z}_n = \{0, 1, 2, \cdots, n-1\}$ muni des deux lois de composition interne $\oplus$ et $\otimes$ forme un anneau commutatif.

1.1.5 Les corps de Galois

Définition 1.8. *Un corps $(C, +, \cdot)$ est un ensemble muni de deux lois de composition interne « + » et « · » telles que :*
- ◇ *C muni de « + » est un anneau commutatif.*
- ◇ *Tout élément non nul de C possède un inverse : $\forall a \in C, \exists a^{-1} \in C$ tel que $a \cdot a^{-1} = 1$*

Définition 1.9. *Un corps fini est un corps ayant un nombre fini d'éléments. Un corps fini est généralement appelé* corps de Galois *et est noté $\mathbb{GF}$. L'ordre (ou cardinal) d'un corps de Galois est le nombre de ses éléments.*

Il est facilement démontrable que l'anneau commutatif $(\mathbb{Z}_2, \oplus_2, \otimes_2)$ forme également un corps de Galois d'ordre 2. D'une manière générale, nous pouvons démontrer que pour un nombre premier p quelconque, l'anneau commutatif $(\mathbb{Z}_p, \oplus_p, \otimes_p)$ est également un corps de Galois d'ordre p et est noté $\mathbb{GF}(p)$.

Définition 1.10. *Soient C un corps et K un sous-ensemble de C. Si K muni des lois de composition interne de C est aussi un corps alors nous disons que K est un* sous-corps *de C. D'une manière équivalente, C est appelé* extension *du corps K.*

Nous pouvons montrer que pour un quelconque nombre premier p et un quelconque entier positif m, l'anneau commutatif $(\mathbb{Z}_{p^m}, \oplus_{p^m}, \otimes_{p^m})$ forme un corps fini. $\mathbb{Z}_{p^m}$ est une extension du corps de Galois $\mathbb{GF}(p)$. Il est également appelé corps de Galois d'ordre $q = p^m$ et est noté $\mathbb{GF}(q)$. En particulier, les corps de Galois d'ordre $q = 2^m$, avec m un entier positif, suscitent un grand intérêt en pratique notamment en théorie de codage. En effet, comme nous allons le montrer ultérieurement, un élément appartenant à un corps de Galois $\mathbb{GF}(2^m)$ peut être représenté d'une manière unique sous forme d'un symbole binaire de m bits.

Définition 1.11. *Nous disons qu'un ensemble est clos pour une opération si cette opération appliquée à un élément quelconque de l'ensemble produit toujours un élément de l'ensemble.*

Un corps de Galois $\mathbb{GF}(q)$ est clos pour ses deux lois de composition interne $\oplus_q$ et $\otimes_q$.

1.1.6 Les polynômes définis sur $\mathbb{GF}(q)$

Définition 1.12. *Un polynôme f défini sur le corps de Galois $\mathbb{GF}(q)$ est une expression de la forme :*

$$f(X) = \alpha_n X^n + \alpha_{n-1} X^{n-1} + \cdots + \alpha_1 X + \alpha_0$$

où les coefficients α_i, $i = 0, 1, \cdots, n$, sont éléments de $\mathbb{GF}(q)$ et X est un symbole formel appelé indéterminée du polynôme. L'entier positif n est appelé le degré du polynôme et il est noté $deg(f)$.

Définition 1.13. *Considérons les deux polynômes $f(X) = \sum_{i=0}^{n} a_i X^i$ et $g(X) = \sum_{i=0}^{m} b_i X^i$ avec $m \leq n$. Le polynôme $g(X)$ peut aussi être écrit sous la forme $g(X) = \sum_{i=0}^{n} b_i X^i$ en considérant que les coefficients b_i sont nuls pour tout i supérieur à m. Nous obtenons les définitions suivantes :*

- ◇ *f=g si et seulement si $\forall i \in \{0, 1, \cdots, n\}$, $a_i = b_i$*
- ◇ *L'opération d'addition définie sur les polynômes est :*

$$f(X) + g(X) = \sum_{i=0}^{n} (a_i \oplus_q b_i) X^i$$

- ◇ *L'opération de multiplication définie sur les polynômes est :*

$$f(X) \cdot g(X) = \sum_{k=0}^{n+m} c_k X^i \quad avec \quad c_k = \bigoplus_{\substack{i+j=k \\ i \in \{0,1,\cdots,n\} \text{ et } j \in \{0,1,\cdots,m\}}} a_i \otimes_q b_j$$

- ◇ *L'ensemble $F_q[X]$ des polynômes à indéterminée X et à coefficients dans $\mathbb{GF}(q)$ muni des opérations d'addition et de multiplication est un anneau.*

Le théorème de la division euclidienne peut être généralisé sur les polynômes. Ainsi, si g est un polynôme non nul dans $F_q[X]$ alors pour tout polynôme f de $F_q[X]$ il existe deux polynômes q et r dans $F_q[X]$ tels que :

$$f(X) = q(X) \cdot g(X) + r(X)$$

avec $deg(f) < deg(g)$.

Définition 1.14. *Soient deux polynômes f et $g \in F_q[X]$.*

- ◇ *Nous disons que g est un diviseur de f s'il existe un polynôme $q \in F_q[X]$ tel que*

$$f(X) = q(X) \cdot g(X)$$

- ◇ *f est dit irréductible dans $F_q[X]$ si $deg(f) > 0$ et f ne peut pas être factorisé en produit de deux polynômes chacun de degré strictement supérieur à 0. Autrement dit, si $f = q \cdot g$ alors $deg(q) = 0$ ou $deg(g) = 0$.*
- ◇ *Un polynôme irréductible f de degré m est dit primitif si $X^n + 1 = f(X) \cdot g(X)$ implique que $n \geq 2^m$.*
- ◇ *Un élément $\alpha \in \mathbb{GF}(q)$ est appelé racine du polynôme $f \in F_q[X]$ si $f(\alpha) = 0$. D'une manière équivalente, nous pouvons démontrer que α est une racine de f si le polynôme $(X - \alpha)$ est un diviseur de f.*

1.1.7 Construction des corps de Galois $\mathbb{GF}(2^m)$

Soit p un polynôme primitif de degré m et à coefficients dans $\mathbb{GF}(2)$. Ce polynôme n'a pas de racine dans $\mathbb{GF}(2)$. Cependant, en algèbre abstraite, nous pouvons imaginer qu'il possède une racine α dans un autre ensemble (par analogie aux polynômes à coefficients dans $\mathbb{R}$ qui peuvent avoir une ou plusieurs racines dans $\mathbb{C}$). Nous considérons les deux éléments 0 et 1 de $\mathbb{GF}(2^m)$ et le nouveau élément α. Définissons l'opération de multiplication noté « $\cdot$ » comme suit :

- ◇ 0 est l'élément absorbant de la multiplication : $0 \cdot \alpha = \alpha \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$
- ◇ 1 est l'élément neutre de la multiplication : $1 \cdot \alpha = \alpha \cdot 1 = \alpha$ et $1 \cdot 1 = 1$
- ◇ Le composé n-fois de l'élément α avec lui même est noté $\alpha^n = \alpha \cdot \alpha \cdot \cdots \cdot \alpha$. Par convention $\alpha^0 = 1$
- ◇ $\forall i, j \in \mathbb{N}, \quad \alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j}$

p étant un polynôme primitif de degré m implique :

$$X^{2^m-1} + 1 = q(X) \cdot p(X)$$

En remplaçant X par α, nous obtenons

$$\alpha^{2^m-1} + 1 = q(\alpha) \cdot p(\alpha) = q(\alpha) \cdot 0 = 0$$

Il en découle $\alpha^{2^m-1} = 1$. Par conséquent, l'ensemble $F = \{0, 1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}\}$ muni de la loi « $\cdot$ » est un ensemble fini d'ordre 2^m.

A travers cette section, nous allons montrer que l'ensemble F, muni de la loi multiplicative « $\cdot$ » et d'une loi d'addition « $+$ » à définir, forme un corps de Galois d'ordre 2^m.

Nous commençons par définir la loi d'addition de sorte que $(F, +)$ forme un groupe abélien. Pour cela, nous observons que chaque élément α^i de F peut être représenté d'une façon unique par un polynôme non nul de degré strictement inférieur à m. En effet, la division euclidienne du monôme X^i, $i = 0, 1, \cdots, 2^m - 2$, par p donne $X^i = q_i(X) \cdot p(X) + a_i(X)$, avec $a_i(X) = a_{i0} + a_{i1}X + a_{i2}X^2 + \cdots + a_{i(m-1)}X^{m-1}$ et les coefficients $a_{ij} \in \{0, 1\}$. Les polynômes $a_i(X)$ sont forcément non nuls du fait que X^i et p sont premiers entre eux. De plus, il est facile de montrer que $a_i(X) \neq a_j(X)$ si $i \neq j$. Comme α est racine de p alors $\alpha^i = a_i(\alpha)$, $i = 0, 1, \cdots, 2^m - 2$. Nous venons de montrer que chaque élément non nul de F est représenté par un polynôme $a_i(X)$. Par convention, l'élément 0 de F est représenté par le polynôme nul. Chaque élément de F possède aussi une représentation binaire en ne considérant que les coefficients de sa représentation polynomiale. Nous définissons la loi d'addition comme suit :

- ◇ $0 + 0 = 0$
- ◇ 0 est l'élément neutre de l'addition : $0 + \alpha^i = \alpha^i + 0 = \alpha^i$, $i = 0, 1, \cdots, 2^m - 2$
- ◇ $\alpha^i + \alpha^j = a_i(\alpha) + a_j(\alpha) = \sum_{k=0}^{m-1} (a_{ik} \oplus_2 b_{jk}) X^i, 0 \leq i, j \leq 2^m - 2$

Il est facile de montrer que l'ensemble $(F, +, \cdot)$ est un anneau commutatif. De plus, nous observons que $\alpha^i \cdot \alpha^j = \alpha^{(i+j) mod (2^m-1)}$. Il en découle que chaque élément non nul α^i de F possède un inverse égale à α^{2^m-1-i}. En conclusion $(F, +, \cdot)$ forme un corps de Galois d'ordre 2^m. Dans le reste de ce manuscrit, nous considérons cette définition du corps de Galois et nous utilisons la notation $\mathbb{GF}(q = 2^m)$. Tous les résultats de simulation et de synthèse de ce manuscrit sont obtenus avec le corps de Galois $\mathbb{GF}(q = 2^6)$ décrit dans

le tableau A.1 de l'annexe A. Enfin, pour simplifier les notations mathématiques dans le reste du manuscrit, β_i dénote, lorsqu'elle est utilisée, le i-ème symbole de $\mathbb{GF}(q = 2^m)$, c'est-à-dire, $\beta_0 = 0, \beta_1 = 1, \beta_2 = \alpha, \cdots, \beta_{q-1} = \alpha^{q-2}$.

1.2 Les codes LDPC non binaires définis sur les corps de Galois

Il a fallu attendre l'invention des Turbo-Codes [18] par Berrou et .al en 1993 pour que nous puissions parler réellement de codes de correction d'erreurs pratiques permettant de s'approcher de la limite de Shannon [19]. Cependant, quelques années plus tard, une ancienne classe de codes refait surface pour s'imposer fortement sur la scène scientifique et réussir à entrer en compétition avec les Turbo-Codes. Il s'agit de la classe des codes en bloc linéaires à matrices de parité creuses, connue sous l'abréviation LDPC. Pourtant inventés en 1963 par Gallager dans son manuscrit de thèse [9], les travaux de recherche de Mackay et Neal [10, 11], publiés dans la deuxième moitié des années 1990, marquent un réel tournant dans l'histoire des codes LDPC. C'est ainsi que ces deux chercheurs ont montré que les performances des codes LDPC peuvent s'approcher considérablement de la limite de Shannon et surpasser les performances des Turbo-Codes.

Comme nous venons de mentionner ci-dessus, un code LDPC fait parti de la famille des codes en bloc linéaires [20] avec la particularité d'être défini par une matrice de parité creuses de dimensions $M \times N$, c'est-à-dire une matrice ne contenant qu'un faible nombre d'éléments non nuls. La matrice de parité est généralement notée H. Son nombre de lignes noté M correspond au nombre de contraintes de parité du code, et son nombre de colonnes noté N correspond à la longueur des mots de code. Un mot de code se compose de K symboles appartenant au message d'information initial et $M = N - K$ symboles de redondance ajoutés par l'encodeur. Les contraintes de parité de la matrice H doivent être par construction respectées par les mots de codes. Ainsi, un message c de longueur N est un mot de code si et seulement si $C \cdot H^T = 0$, où H^T désigne la matrice transposée de H. Prenons l'exemple de la matrice de parité H suivante de dimensions 4×6 :

$$H = \begin{pmatrix} h_{0,0} & h_{0,1} & h_{0,2} & 0 & 0 & 0 \\ 0 & h_{1,1} & 0 & h_{1,3} & h_{1,4} & 0 \\ h_{2,0} & 0 & 0 & h_{2,3} & 0 & h_{2,5} \\ 0 & 0 & h_{3,2} & 0 & h_{3,4} & h_{3,5} \end{pmatrix}$$

Par conséquent, un mot de code $C = [c_0, c_1, c_2, c_3, c_4, c_5]$ satisfait les quatre équations suivantes :

$$h_{0,0} \cdot c_0 + h_{0,1} \cdot c_1 + h_{0,2} \cdot c_2 = 0 \tag{1.1}$$

$$h_{1,1} \cdot c_1 + h_{1,3} \cdot c_3 + h_{1,4} \cdot c_4 = 0 \tag{1.2}$$

$$h_{2,0} \cdot c_0 + h_{2,3} \cdot c_3 + h_{2,5} \cdot c_5 = 0 \tag{1.3}$$

$$h_{3,2} \cdot c_2 + h_{3,4} \cdot c_4 + h_{3,5} \cdot c_5 = 0 \tag{1.4}$$

Outre la présentation matricielle, un code LDPC peut être présenté à l'aide d'un graphe biparti (ou graphe de Tanner) [21]. Les graphes bipartis fournissent une description complète de la structure du code et aident également à décrire les algorithmes de décodage

comme expliqué dans la Section 1.3. Un graphe bi-parti est un graphe composé par deux ensembles de nœuds tels que deux nœuds du même ensemble ne sont connectés qu'à travers un nœud de l'autre ensemble. Dans le cas d'un code LDPC, nous parlons de l'ensemble des nœuds de parité (Check Node ou CN en anglais) et l'ensemble des nœuds de variable (Variable Node ou VN en anglais). Un CN représente une ligne de la matrice du parité du code (ou d'une manière équivalente une contrainte de parité) et un VN représente une colonne (ou d'une manière équivalente un symbole du mot de code). Par conséquent, le graphe biparti associé à un code LDPC représenté par une matrice de parité H de dimensions $M \times N$ est composé de M CNs et N VNs. Un CN p_i est lié à un VN v_j si l'élément de la i-ème ligne et j-ème colonne de la matrice de parité est non nul (ou d'une manière équivalente, si le j-ème symbole du mot de code participe à la i-ème contrainte de parité). Ainsi la matrice de l'exemple précédant peut être présentée par le graphe biparti de la figure 1.1.

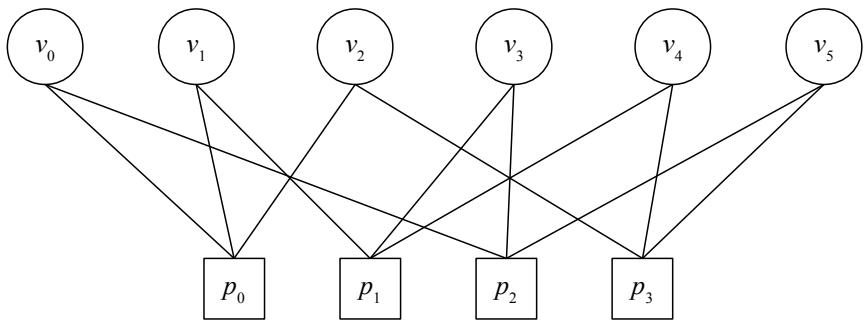


FIGURE 1.1 – Représentation graphique d'un code LDPC

Le nombre de symboles non nuls dans chaque colonne de la matrice de parité est noté d_v et le nombre de symboles non nuls dans chaque ligne est noté d_c. Un code LDPC est dit régulier si d_v est constant pour toutes les colonnes de la matrice et $d_c = \frac{N}{M} \cdot d_v$ est constant pour toutes les lignes de la matrice. Dans le cas contraire, le code est dit irrégulier. Bien que les codes irréguliers possèdent de meilleurs performances grâce à leur structure hautement aléatoire, les codes réguliers sont généralement des codes structurés qui permettent des implémentations matérielles efficaces. Il est possible de repérer la régularité d'un code à l'aide de son graphe biparti. Le code est régulier si le nombre d_v d'arêtes sortantes de chaque VN et le nombre d_c d'arêtes sortantes de chaque CN sont constants. Par conséquent, d_v et d_c sont appelés respectivement les degrés de connectivité des VNs et des CNs. Dans le cas d'un code LDPC régulier défini par une matrice de rang plein (aucune ligne de la matrice est combinaison linéaire d'autres lignes), le rendement R du code peut être exprimé en fonction de d_v et d_c comme suit :

$$R \triangleq \frac{K}{N} = \frac{N-M}{N} = 1 - \frac{d_v}{d_c} \tag{1.5}$$

Dans ce manuscrit, nous considérons le cas des codes LDPC définis sur les corps de Galois $\mathbb{GF}(q = 2^m)$, $m \geq 1$. Les codes LDPC dont les symboles des mots de code appartiennent au corps de Galois binaire (m=1) sont dits binaires, tandis que les codes LDPC dont les symboles des mots de code appartiennent à un corps de Galois d'ordre $q > 2$ sont dits non binaires. La notation $\mathbb{GF}(q)$-LDPC est utilisée dans ce manuscrit pour désigner un code

LDPC défini sur le corps de Galois d'ordre q. Les éléments d'une matrice de parité d'un code $\mathbb{GF}(q)$-LDPC appartiennent à un corps de Galois $\mathbb{GF}(q = 2^m)$, $m \geq 2$ et les produits matriciels des équations de parité sont effectués en utilisant les lois de composition internes du corps de Galois. Il est alors préférable d'ajouter au graphe biparti de la figure 1.1 une nouvelle famille de nœuds appelés les nœuds de permutation qui servent à modéliser la multiplication des symboles du mot de code par les éléments non nuls de la matrice de parité. La figure 1.2 illustre la graphe biparti partiel de l'équation 1.2 en ajoutant les nœuds de permutation qui correspondent aux éléments $h_{1,1}$, $h_{1,3}$ et $h_{1,4}$.

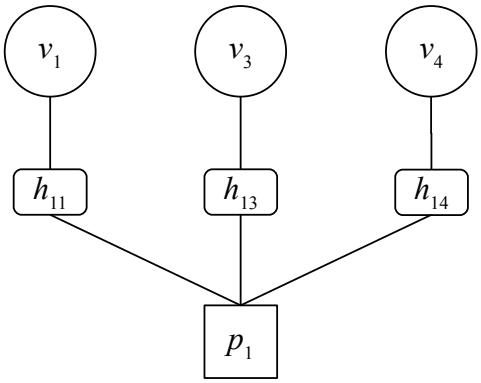


FIGURE 1.2 – Représentation graphique d'une équation de parité dans le cas non binaire

Les codes LDPC binaires possèdent des performances asymptotiques s'approchant de la limite de Shannon [10, 11]. Cependant, pour des mots de code de petite ou moyenne taille, les performance des codes LDPC binaires se dégradent considérablement. Il a été montré dans [13] que cette perte peut être compensée en utilisant des codes $\mathbb{GF}(q)$-LDPC de grande cardinalité. De plus, la grande cardinalité des codes assure une meilleure résistance aux erreurs par paquet [22]. Cette amélioration des performances peut être expliquée intuitivement par le fait que plusieurs bits sont regroupés dans un seul symbole non binaire. Par conséquent, les bits erronés sont confinés dans moins de symboles non binaires, et par la suite les contraintes de parité sont touchées par moins d'erreurs. Néanmoins, l'amélioration des performances par l'augmentation de l'ordre du corps de Galois s'accompagne d'une augmentation exorbitante de la complexité du décodage qui constitue un frein à l'exploitation pratique des codes $\mathbb{GF}(q)$-LDPC.

1.3 Le décodage itératif des codes $\mathbb{GF}(q)$-LDPC

La représentation graphique des codes $\mathbb{GF}(q)$-LDPC peut être exploitée pour la mise en œuvre d'algorithmes dont l'efficacité a été montrée sur des modèles de graphe tel que l'algorithme *Propagation de Croyance* notée généralement BP (Belief Propagation en anglais). L'algorithme BP est un algorithme itératif qui fait parti de la classe des algorithmes avec passage de messages. Ils sont appelés ainsi car, à chaque itération de l'algorithme, des messages sont transmis des CNs à leurs VNs connexes, et vice versa. Nous distinguons deux types de messages :

- ◇ Les messages intrinsèques : ce sont des messages d'information *a priori* calculés uniquement à partir des observations du canal. Ces messages sont dit intrinsèques car ils contiennent de l'information qui ne dépend que du canal. A l'étape d'initialisation, ces messages sont directement envoyés à l'ensemble des CNs.
- ◇ Les messages extrinsèques : chaque branche d'un noeud du graphe fait circuler un message entrant et un message sortant. Ces messages sont dits extrinsèques car l'information sortante sur une branche donnée d'un nœud n'est fonction que des messages entrants sur les autres branches de ce noeud. Les messages sortants des VNs sont fonction des messages extrinsèques des CNs et des messages intrinsèques. Les messages sortants des CNs sont fonctions des messages extrinsèques des VNs et sont calculés à partir des contraintes de parité locales.

Le décodeur doit pouvoir converger vers un mot de code valide au bout d'un nombre fini d'itérations. En pratique, l'algorithme de décodage peut être stoppé selon deux critères. Le plus simple est de fixer le nombre d'itérations indépendamment de la convergence du décodeur. Le deuxième critère, qui permet de réduire la latence du décodeur, consiste à arrêter le décodage dès qu'il converge vers un mot de code valide (un mot de code estimé $\hat{C}$ est valide s'il satisfait le syndrome $\hat{C} \cdot H^T = 0$). Cependant, pour éviter une exécution à l'infinie au cas où le décodeur ne réussit pas à converger vers un mot de code valide, un nombre maximal d'itérations est fixé.

Dans l'algorithme BP, les messages échangés sont des *probabilités a posteriori* calculées sur les symboles du mot de code. Cependant, l'algorithme BP tel qu'il est proposé dans [13] souffre d'une complexité calculatoire prohibitive, dominée par $O(q^2)$, qui provient essentiellement des calculs effectués lors de la mise à jour des contraintes de parité. En remarquant que la mise à jour des CNs peut être modélisée par des produits de convolution, Barnault et al. ont proposé dans [23] l'algorithme FFT-BP dans lequel les mises à jour des contraintes de parité se font dans le domaine fréquentiel afin de transformer les produits de convolution en de simples multiplications. Ainsi, des opérations supplémentaires de transformée de Fourier, directe et inverse, sont ajoutées entre les VNs et les CNs pour assurer la passage du domaine des probabilités vers le domaine fréquentiel, et vice versa. Bien que la complexité de l'algorithme FFT-BP est considérablement réduite à l'ordre de $O(q \log(q))$, un grand nombre de multiplications reste nécessaire pour effectuer la mise à jour des nœuds du graphe. L'algorithme log-BP [24] est un algorithme dans lequel les quatre étapes de décodage s'effectuent dans le domaine logarithmique pour permettre une implantation matérielle moins sensible aux erreurs de quantification, et par conséquent mieux adaptée à une arithmétique en virgule fixe. Cependant, la mise à jour des CNs nécessite toujours une grande quantité de calcul et la complexité du décodeur reste dominée par $O(q^2)$. Une combinaison directe des algorithmes FFT-BP et log-BP n'est pas avantageuse parce que le calcul de la transformée de Fourier est très complexe dans le domaine logarithmique. Pour profiter simultanément des avantages des algorithmes FFT-BP et log-BP, Song et al. ont proposé dans [22] l'algorithme log-BP-FFT. Dans cet algorithme, les VNs sont traités dans le domaine logarithmique. Les messages extrinsèques des VNs subissent une double transformation pour passer du domaine logarithmique vers le domaine des probabilités et du domaine des probabilités vers le domaine fréquentiel dans lequel seront traités les CNs. Les messages extrinsèques des CNs subissent à leur tour une double transformation pour repasser au domaine logarithmique des VNs. Toutefois, l'algorithme log-BP-FFT nécessite des tables de correspondances LUTs (Look Up Tables en anglais) pour assurer la conversion entre le domaine des probabilités et le domaine logarithmique. Ces tables ont l'inconvénient de consommer beaucoup de ressources mé-

moires, une consommation qui augmente avec le degré de parallélisme du décodeur. La figure 1.3 illustre les étapes des différents algorithmes de décodage cités ci-avant.

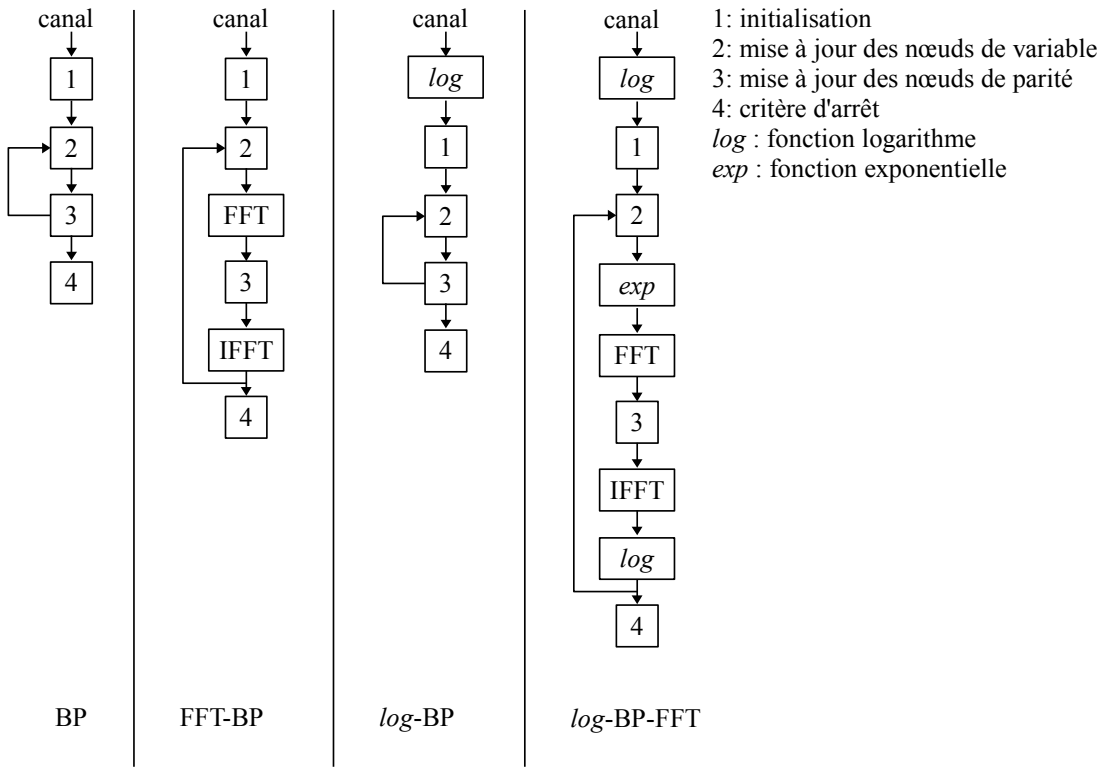


FIGURE 1.3 – Les principaux algorithmes de décodage optimal des codes $\mathbb{GF}(q)$-LDPC

Les algorithmes BP, FFT-BP, log-BP et log-BP-FFT sont en effet des algorithmes de décodage dit optimaux parce qu'ils n'utilisent aucune approximation mathématique pour réduire la complexité du décodage. L'algorithme BP et ses variantes garantissent des performances de décodage optimales mais ils ne sont pas d'un grand intérêt pour une implantation matérielle. Par conséquent, d'autres algorithmes reposant sur des approximations de l'algorithme BP ont été proposés dans le but d'assurer un compromis performance/complexité raisonnable. Nous citons principalement l'algorithme Min-Sum [24] et sa variante EMS (Extended Min-Sum) [25, 1]. Une comparaison détaillée des algorithmes optimaux et sous-optimaux cités ci-dessus se trouve dans [26]. Outre l'algorithme EMS, il existe aussi l'algorithme Min-Max [27] qui peut être considéré comme une approximation de l'algorithme Min-Sum, et qui par conséquent fournit de moins bonnes performances.

Dans ce qui suit, nous détaillons les algorithmes BP, log-BP, Min-Sum, EMS et Min-Max. Nous adoptons la convention mathématique suivante : Soit $V = [v_0, v_1, \cdots, v_n]$ un vecteur quelconque composé par n éléménts. Si i est un entier positif ou nul, la notation $V(i)$ indique l'élément de position i dans V. Si $\beta \in \mathbb{GF}(2^m)$, la notation $V[\beta]$ indique l'élément associé au symbole β dans V.

1.3.1 L'algorithme BP

Soit $c = [c_0, c_1, \cdots, c_{N-1}]$, $c_i \in \mathbb{GF}(q = 2^m)$, le mot de code transmis. Le rôle du décodage est de converger vers un mot de code valide $\hat{c} = [\hat{c}_0, \hat{c}_1, \cdots, \hat{c}_{N-1}]$ à partir d'un signal

bruité $y = [y_0, y_1, \cdots, y_{N-1}]$, y_i étant la version bruitée de c_i. Le décodage est dit réussi si $\hat{C} = C$.

Dans l'algorithme BP, l'information intrinsèque d'un VN v_i est un vecteur de q probabilités *a posteriori* défini par l'équation 1.6

$$I_i = [p(v_i = \beta_0|y_i), p(v_i = \beta_1|y_i), \cdots, p(v_i = \beta_{q-1}|y_i)] \tag{1.6}$$

où $p(a|b)$ est la probabilité conditionnelle de a sachant b.

Soient $i = 0, 1, \cdots, M-1$ et $j = 0, 1, \cdots N-1$. Si l'élément h_{ij} de la matrice de parité H est non nul alors $M_{v_j p_i}$ désigne le message envoyé par le VN v_j au CN p_i et $M_{p_i v_j}$ le message envoyé par le CN p_i au VN v_j. L'ensemble des étapes de l'algorithme BP sont :

(a) Initialisation : Chaque VN v_j envoie son information intrinsèque à l'ensemble de ces CNs connexes.

(b) Mise à jour des VNs : Un VN v_j reçoit d_v messages $\tilde{M}_{p_i v_j}$ et fournit d_v messages $\tilde{M}_{v_j p_i}$. Les messages sortant de v_j sont calculés par l'équation 1.7. Chaque message sortant est fonction de tous les messages entrants à v_j sauf celui de p_i.

$$M_{v_j p_i}[\beta] = \mu_{v_j p_i} \cdot I_j[\beta] \cdot \prod_{\substack{s \neq i \\ h_{s,j} \neq 0}} \tilde{M}_{p_s v_j}(\beta) \quad \beta \in \mathbb{GF}(q) \tag{1.7}$$

avec $\mu_{v_j p_i}$ est un facteur de normalisation tel que $\sum_{\beta \in \mathbb{GF}(q)} M_{v_j p_i}[\beta] = 1$.

(c) Permutation : Avant d'entrer au CN p_i, le message $M_{v_j p_i}$ est multiplié par l'élément non nul h_{ij} de la matrice de parité. Le message résultant $\tilde{M}_{v_j p_i}$ est obtenu par l'équation 1.8.

$$\tilde{M}_{v_j p_i}[\beta] = M_{v_j p_i}[\beta \cdot h_{ij}^{-1}] \quad \beta \in \mathbb{GF}(q) \tag{1.8}$$

(d) Mise à jour des CNs : La mise jour du CN p_i est effectuée par l'équation 1.9.

$$M_{p_i v_j}[\beta] = \sum_{\substack{\sum_{s \neq j} \theta_s = \beta \\ h_{is} \neq 0}} \prod_{\substack{s \neq j \\ h_{is} \neq 0}} \tilde{M}_{v_s p_i}[\theta_s] \tag{1.9}$$

avec β et θ_s des variables appartenant à $\mathbb{GF}(q)$. La mise à jour de p_i consiste à calculer la probabilité de toutes les combinaisons de symboles vérifiant l'équation de parité.

(e) Permutation inverse : Avant d'entrer au VN v_j, le message $M_{p_i v_j}$ est divisé par l'élément non nul h_{ij} de la matrice de parité. Le message résultant $\tilde{M}_{p_i v_j}$ est obtenu par l'équation 1.10.

$$\tilde{M}_{p_i v_j}[\beta] = M_{p_i v_j}[\beta \cdot h_{ij}] \quad \beta \in \mathbb{GF}(q) \tag{1.10}$$

(f) Estimation du mot de code : Au terme de chaque itération, chaque VN v_j met à jour un vecteur de probabilités *a priori* noté APP_j comme dans l'équation 1.11.

$$\mathrm{APP}_j[\beta] = \mu_{v_j} \cdot I_j[\beta] \cdot \prod_{h_{s,j} \neq 0} \tilde{M}_{p_s v_j}(\beta) \quad \beta \in \mathbb{GF}(q) \tag{1.11}$$

avec μ_{v_j} est un facteur de normalisation tel que $\sum_{\beta \in \mathbb{GF}(q)} \mathrm{APP}_j[\beta] = 1$. Ensuite, la décision se fait par l'équation 1.12 qui consiste à sélectionner le symbole qui possède

la plus grande probabilité dans APP_j.

$$\hat{c}_j = \underset{\beta \in \mathbb{GF}(q)}{\text{argmax}}\{\text{APP}_j[\beta]\} \quad j = 0, 1, \cdots N-1 \tag{1.12}$$

Si l'ensemble des symboles $\hat{c}_j$ forment un mot de code valide alors le décodage prend fin.

1.3.2 L'algorithme log-BP

La fiabilité d'un symbole peut aussi être mesurée par le rapport de vraisemblance logarithmique (Log-Likelihood Ratio ou LLR en anglais) défini par l'équation 1.13.

$$\text{LLR}(\beta) = \ln \frac{p(v_j = \beta | y_j)}{p(v_j = 0 | y_j)} \quad \beta \in \mathbb{GF}(q) \tag{1.13}$$

Remplacer les probabilités par des LLRs dans les équations 1.7, 1.9 et 1.11 permet d'une part de transformer les opérations de multiplication en des opérations d'additions et de l'autre part de réduire les erreurs de quantification. Ainsi, dans l'algorithme log-BP, l'information intrinsèque d'un VN v_j est définie par l'équation 1.14.

$$I_j = [0, \ln \frac{p(v_j = \beta_1 | y_j)}{p(v_j = \beta_0 | y_j)}, \cdots, \ln \frac{p(v_j = \beta_{q-1} | y_j)}{p(v_j = \beta_0 | y_j)}] \tag{1.14}$$

Les messages qui circulent sur le graphe biparti sont composés par des LLRs. L'algorithme log-BP garde les mêmes étapes de décodage de l'algorithme BP tout en modifiant les équations de mise à jour. En effet, la mise à jour d'un VN v_j se fait par l'équation 1.15.

$$M_{v_j p_i}[\beta] = I_j[\beta] + \sum_{\substack{s \neq i \\ h_{s,j} \neq 0}} \tilde{M}_{p_s v_j}(\beta) \quad \beta \in \mathbb{GF}(q) \tag{1.15}$$

La mise à jour d'un CN p_i se fait par l'équation 1.16.

$$M_{p_i v_j}[\beta] = \ln \sum_{\substack{\sum_{\substack{s \neq j \\ h_{is} \neq 0}} \theta_s = \beta}} \exp\Big(\sum_{\substack{s \neq j \\ h_{is} \neq 0}} \tilde{M}_{v_s p_i}[\theta_s] \Big) \tag{1.16}$$

Enfin, la mise à jour de l'information *a priori* se fait par l'équation 1.17.

$$\text{APP}_j[\beta] = I_j[\beta] + \sum_{h_{s,j} \neq 0} \tilde{M}_{p_s v_j}(\beta) \quad \beta \in \mathbb{GF}(q) \tag{1.17}$$

1.3.3 L'algorithme Min-Sum

L'algorithme Min-Sum a été proposé dans [25] pour réduire la complexité de l'algorithme log-BP à l'aide d'une approximation de l'équation 1.16. En effet, dans l'algorithme Min-Sum, la mise à jour d'un CN p_i est effectuée par l'équation 1.18.

$$M_{p_i v_j}[\beta] \approx \max_{\substack{\sum_{\theta_s=\beta} \\ s\neq j \\ h_{is}\neq 0}} \{ \sum_{\substack{s\neq j \\ h_{is}\neq 0}} \tilde{M}_{v_s p_i}[\theta_s] \} \tag{1.18}$$

L'algorithme Min-Sum permet donc de simplifier le décodeur en supprimant les tables de correspondances nécessaires à l'implantation des fonctions exponentielles et logarithmes et en minimisant le nombre des opérations arithmétiques.

1.3.4 L'algorithme EMS

Pour simplifier encore le décodeur Min-Sum, les auteurs de [25] ont introduit l'algorithme EMS dans lequel les messages qui circulent sur le graphe biparti sont tronqués en en sélectionnant les n_m symboles les plus fiables parmi les q symboles possibles, avec $n_m \ll q$. Toutefois, la valeur de n_m doit être minutieusement choisie pour que les performances de décodage ne subissent pas une dégradation significative.

Dans le cas de l'algorithme log-BP, les messages sont des vecteurs composés de q valeurs de fiabilité non triées. De plus, il n'est pas nécessaire d'indiquer explicitement la valeur du symbole associé à chacune des fiabilités puisqu'elle peut être facilement déduite par sa position dans le message. En raison de la troncature, les messages de l'algorithme EMS doivent être triés et les valeurs des symboles doivent être explicitement mentionnées. Les messages qui circulent sur le graphe sont par conséquent de la forme $M = [(\mathrm{LLR}(\theta_k), \theta_k)]_{0\leq k<n_m}$, avec θ_k une variable dans $\mathbb{GF}(q)$ et $\mathrm{LLR}(\theta_{k'}) \geq \mathrm{LLR}(\theta_{k''})$ si $k^{'} < k^{''}$. Dans ce qui suit, $M.GF$ désigne le message partiel contenant l'ensemble des symboles du message M et $M.L$ désigne le vecteur contenant l'ensemble des LLRs du message M. Le symbole le plus fiable dans M est $M.GF(0)$ et le symbole le moins fiable est $M.GF(n_m - 1)$.

La troncature des messages entraîne une dégradation des performances qui peut être compensée en utilisant une valeur de fiabilité constante notée γ pour les symboles non retenus lors de la troncature. La valeur de γ est calculée de la façon suivante :

$$\gamma = M.L(n_m - 1) - \text{offset} \tag{1.19}$$

où offset est un scalaire déterminé par simulation de façon à obtenir le meilleur BER possible.

Les étapes de l'algorithmes EMS peuvent être résumées comme suit :

(a) Initialisation : chaque VN v_j envoie l'information intrinsèque des n_m symboles les plus fiables à l'ensemble de ses CNs connexes.

(b) Mise à jour des VNs : un VN v_j reçoit d_v messages $\tilde{M}_{p_i v_j}$. Un scalaire de compensation γ_i est associé à chaque message $\tilde{M}_{p_i v_j}$. La valeur de γ_i est déterminé par l'équation 1.19. Le message sortant $M_{v_j p_i}$ contient les n_m symboles les plus fiables en combinant l'information intrinsèque avec les messages entrants sauf celui de p_i. La fiabilité d'un symbole $M_{v_j p_i}.GF(k)$, $k = 0, 1, \cdots, n_m$ s'obtient par l'équation 1.20.

$$M_{v_j p_i}.L(k) = I_i[M_{v_j p_i}.GF(k)] + \sum_{\substack{s\neq j \\ h_{s,i}\neq 0}} W_s(k) \tag{1.20}$$

avec

$$W_s(k) = \begin{cases} \tilde{M}_{p_s v_j}[M_{v_j p_i}.GF(k)] & \text{si } M_{v_j p_i}.GF(k) \in \tilde{M}_{p_s v_j} \\ \gamma_s & \text{sinon} \end{cases}$$

(c) Permutation : chaque symbole de $M_{v_j p_i}$ est multiplié par l'élément $h_{ij} \neq 0$ de la matrice de parité.

$$\tilde{M}_{v_j p_i}.GF(k) = h_{ij} \cdot M_{v_j p_i}.GF(k) \quad k = 0, 1, \cdots, n_m - 1 \tag{1.21}$$

(d) Mise à jour des CNs : la fiabilité d'un symbole d'un message sortant est calculée comme dans l'équation 1.18. Les messages sortants $M_{p_i v_j}$ contiennent les n_m symboles les plus fiables. Le CN peut être implanté d'une manière efficace par une récursion *Avant-Arrière* (Forward-Backward en anglais) proposée dans [24]. Cette récursion consiste à construire les messages sortants par un ensemble d'opérations élémentaires permettant à la fois de ne pas refaire les mêmes calculs et de réduire la latence de traitement. Ces opérations élémentaires sont réalisées par des nœuds de parité élémentaires (Elementary Check Node ou ECN en anglais). La figure 1.4 illustre l'architecture Forward-Backward d'un CN p de degré $d_c = 6$. Pour avoir plus de clarté, les messages entrants du CN sont notés par $\tilde{M}_{v_j p}$ et les messages sortants sont notés par M_{pv_j}, $j = 0, 1, \cdots, 5$.

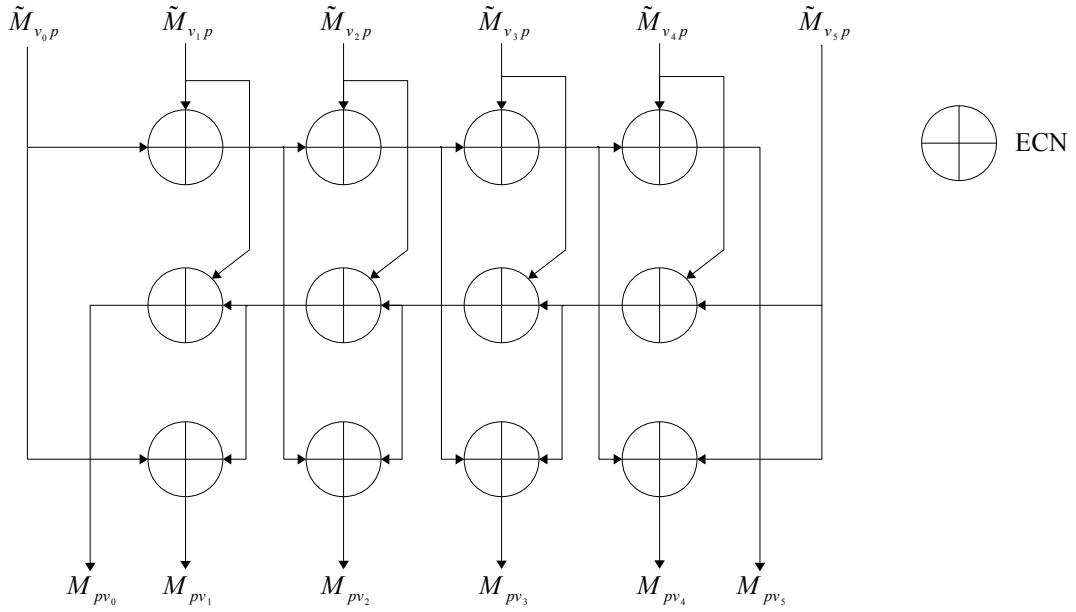


FIGURE 1.4 – Architecture Forward-Backward d'un CN de degré $d_c = 6$

Chaque ECN reçoit deux messages triés $M_e^{'}$ et $M_e^{''}$ et génère un message trié M_s (chaque message contient n_m symboles et n_m fiabilités). Le message M_s est construit en sélectionnant les n_m symboles les plus fiables parmi toutes les combinaisons possibles de $M_e^{'}$ et $M_e^{''}$. La fiabilité d'un symbole $M_s.GF(k)$, $k = 0, 1, \cdots, n_m$, est déterminée par l'équation 1.22.

$$M_s.L(k) \approx \max_{\substack{M_e^{'}.GF(k^{'}) + M_e^{''}.GF(k^{''}) = M_s.GF(k) \\ k^{'}, k^{''} = 0,1,\cdots,n_m}} \left\{ M_e^{'}.L(k^{'}) + M_e^{''}.L(k^{''}) \right\} \tag{1.22}$$

La réalisation d'un CN de degré d_c nécessite l'implantation de $3(d_c - 2)$ ECNs répartis

sur 3 couches. Dans notre cas de figure, le message sortant M_{pv_5} est généré par la première couche d'ECNs qui additionne les messages entrants dans le sens *Avant* (Forward). Le message sortant M_{pv_0} est généré par la deuxième couche d'ECNs qui additionne les messages entrants dans le sens *Arrière* (Backward). Les autres messages sortants sont générés par la troisième couche qui combine les deux messages entrants $\tilde{M}_{v_0p}$ et $\tilde{M}_{v_5p}$ avec les messages intermédiaires des deux couches supérieures.

D'une manière exhaustive, un ECN doit sélectionner les n_m meilleurs couples (fiabilité, symbole) parmi n_m^2 couples possibles. Cependant, effectuer un tel traitement s'avère très complexe car il faut d'abord déterminer les n_m^2 combinaisons et ensuite procéder à un tri pour déterminer les symboles les plus fiables. Dans [1], Voicila et al. ont proposé une implémentation moins complexe en modélisant le problème sous forme d'une matrice T_M construite en calculant la somme de $M_e^{'}$ et $M_e^{''}$:

$$\begin{cases} T_M.L(k^{'},k^{''}) = M_e^{'}.L(k^{'}) + M_e^{''}.L(k^{''}) \\ T_M.GF(k^{'},k^{''}) = M_e^{'}.GF(k^{'}) + M_e^{''}.GF(k^{''}) \end{cases} \qquad k^{'},k^{''} = 0,1,\cdots,n_m \tag{1.23}$$

La matrice T_M contient n_m^2 candidats. Le but de l'algorithme proposé dans [1] est d'explorer de façon efficace T_M pour déterminer itérativement les n_m symboles les plus fiables. Puisque $M_e^{'}$ et $M_e^{''}$ sont déjà triés dans l'ordre décroissant les plus grands LLRs dans T_M se situent dans le triangle supérieur gauche. Par conséquent, il suffit d'effectuer un balayage horizontal de la matrice. A la première itération, un comparateur de n_m éléments est initialisé par les valeurs de la première colonne de T_M. Ensuite, à chaque itération, le comparateur sort le couple le plus fiable et le remplace par le couple se trouvant sur la même ligne et sur la colonne d'après dans la matrice T_M. Nous devons noter que puisque un même symbole peut avoir plusieurs occurrences dans T_M le nombre d'itérations n_{oper} nécessaire pour trouver les n_m symboles les plus fiables varient entre n_m dans le meilleur des cas et $\frac{n_m^2}{2}$ dans le pire des cas. La figure 1.5 illustre l'algorithme décrit ci-dessus.

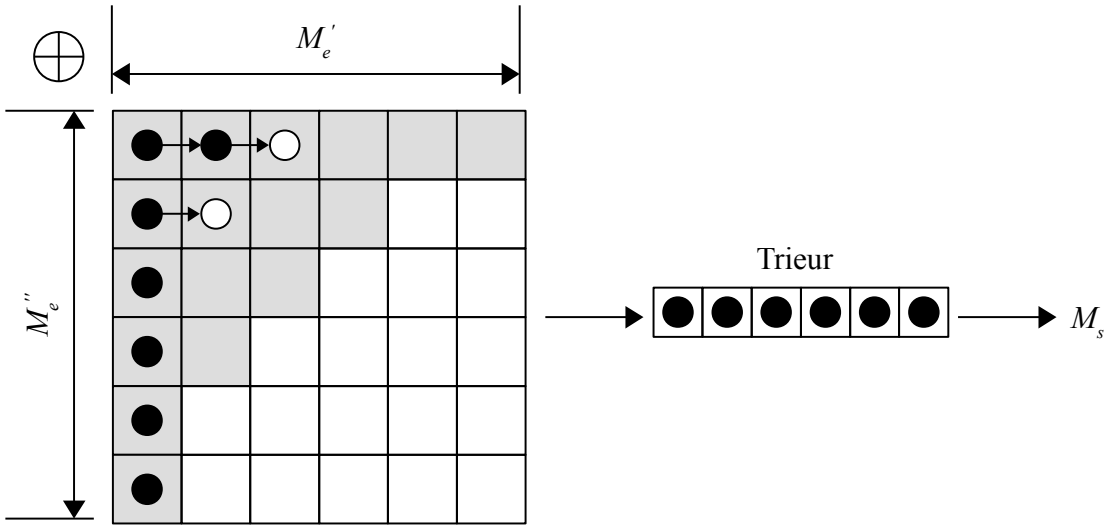


FIGURE 1.5 – Un ECN tel que proposé dans [1]

Le CN peut aussi être modélisé sous la forme d'un treillis. Chaque colonne du treillis

représente une entrée du CN et chaque ligne représente un symbole du corps de Galois. Le poids de chaque nœud du treillis est le LLR du symbole correspondant. La mise à jour du CN consiste donc à trouver les chemins les plus fiables sur le treillis.

La figure 1.6 illustre un treillis obtenu pour $\mathbb{GF}(4)$, $d_c = 6$ et $n_m = 2$. Les n_m nœuds les plus fiables de chaque message sont encadrés. Dans cet exemple, le message sortant M_{pv_5} contient les $n_m = 2$ couples les plus fiables parmi toutes les configurations possibles des messages entrants $\tilde{M}_{v_0p}$, $\tilde{M}_{v_1p}$, $\tilde{M}_{v_2p}$, $\tilde{M}_{v_3p}$ et $\tilde{M}_{v_4p}$. Ces couples sont $(130, \alpha^2)$ et $(122, 0)$.

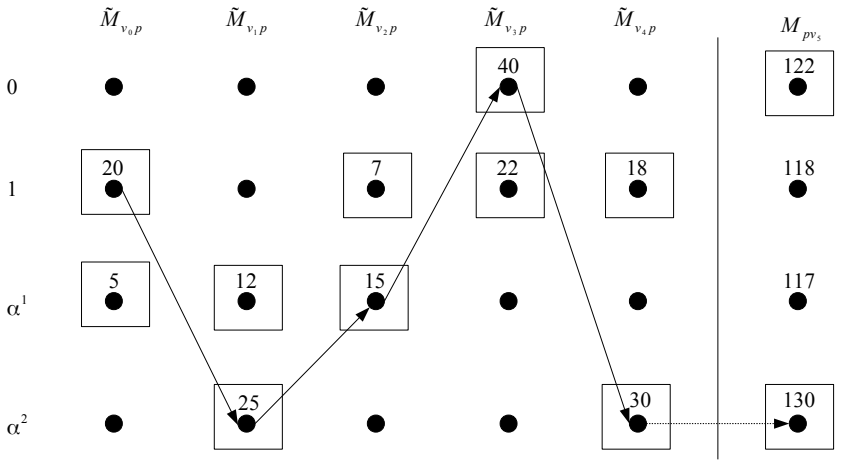


FIGURE 1.6 – Représentation en treillis d'un CN

L'algorithme T-EMS, proposé dans [28, 29], est une variante de l'algorithme EMS qui se base sur la forme en treillis des CNs. Dans cet algorithme, le treillis est obtenu en utilisant des messages différentiels. Ces messages sont construis en soustrayant l'élément le plus fiable dans chaque message entrant au reste de ses éléments. Toutefois, une implantation matérielle de l'algorithme T-EMS n'est efficace que pour les corps de Galois de petite cardinalité.

(e) Permutation inverse : chaque symbole de $M_{p_iv_j}$ est divisé par l'élément $h_{ij} \neq 0$ de la matrice de parité.

$$\tilde{M}_{p_iv_j}.GF(k) = h_{ij}^{-1} \cdot M_{p_iv_j}.GF(k) \quad k = 0, 1, \cdots, n_m - 1 \tag{1.24}$$

(f) Estimation du mot de code : au terme de chaque itération, chaque VN v_j met à jour un vecteur de LLRs *a priori* APP_j comme dans l'équation 1.25.

$$APP_j[\beta] = I_j[\beta] + \sum_{h_{s,j} \neq 0} W_s(\beta) \quad \beta \in \mathbb{GF}(q) \tag{1.25}$$

avec

$$W_s[\beta] = \begin{cases} \tilde{M}_{p_sv_j}.L[\beta], & \text{si } \beta \in \tilde{M}_{p_sv_j}.GF \\ \gamma_s, & \text{sinon} \end{cases}$$

Ensuite la décision se fait comme dans l'équation 1.12.

1.3.5 L'algorithme Min-Max

Le rapport de vraisemblance logarithmique tel que défini dans les sections 1.3.1 et 1.3.3 peut prendre des valeurs négatives. Toutefois, ce serait plus simple de ne traiter que des valeurs positives. Par conséquent, l'auteur de [27] a proposé de définir les LLRs comme suit :

$$LLR(\beta) = -\ln \frac{p(x=\beta|y)}{\max\limits_{\theta \in \mathbb{GF}(2^m)} \{p(x=\theta|y)\}} \quad \beta \in \mathbb{GF}(2^m) \tag{1.26}$$

où $y = (y_0, y_1, \cdots, y_{m-1})$ est l'observation du canal et $x = (x_0, x_1, \cdots, x_{m-1})$ est le symbole transmis.

Dans cette définition, la normalisation se fait par la probabilité du symbole le plus fiable. Il en découle que le LLR de ce symbole vaut toujours zéro et les LLRs des autres symboles sont positifs.

Dans le même papier, l'auteur a proposé son algorithme Min-Max qui permet de simplifier les traitements au niveau des nœuds de parité en remplaçant la somme dans l'équation 1.18 par l'opérateur max :

$$M_{p_i v_j}[\beta] \approx \min_{\substack{\sum\limits_{\substack{s \neq j \\ h_{is} \neq 0}} \theta_s = \beta}} \Big\{ \max_{\substack{s \neq j \\ h_{is} \neq 0}} \tilde{M}_{v_s p_i}[\theta_s] \Big\} \tag{1.27}$$

Les messages dans l'algorithme Min-Max peuvent également être tronqués comme dans l'algorithme EMS.

1.4 Architecture du décodeur EMS du projet DAVINCI

Nous consacrons cette section pour présenter les travaux menés par notre laboratoire de recherche dans le cadre du projet DAVINCI [30]. Ces travaux sont antérieurs à notre thèse et ont abouti au développement d' une architecture efficace d'un décodeur EMS basé sur des matrices de parités ultra-creuses ($d_v = 2$) [31, 32, 33]. Cette classe de matrices permet d'avoir de très bonnes performances et de simplifier l'implantation des VNs. Des résultats de synthèse obtenus pour $\mathbb{GF}(64)$ sont publiés dans [2]. Dans la limite de notre connaissance, il s'agit de la première implantation dans l'état de l'art d'un décodeur LDPC avec une cardinalité supérieure à 32. Toutefois, il faut noter qu'une architecture qui possède certaines similitudes avec celle de [2] a été publiée dans [3] durant la même période.

1.4.1 Architecture globale du décodeur

L'architecture qui a été proposée dans le cadre du projet DAVINCI est une architecture en série avec un ordonnancement à permutation horizontale (Horizontal schuffle scheduling en anglais) [34, 35]. Le principe de ce type d'ordonnancement est illustré dans la figure 1.7 et consiste à traiter les CNs un à un dans chaque itération. Lorsqu'un CN est traité, il

transmet immédiatement ces nouveaux messages extrinsèques à ses VNs connexes. Ceux-ci mettent à jour leurs messages extrinsèques avant de passer à la mise à jour du CN d'après. Par conséquent, l'ordonnancement horizontal permet d'accélérer la convergence du décodeur puisque dans une itération un CN p_j bénéficie de l'information des itérations précédentes et de l'information de l'itération en cours en provenance des CNs p_i qui le précèdent ($i < j$).

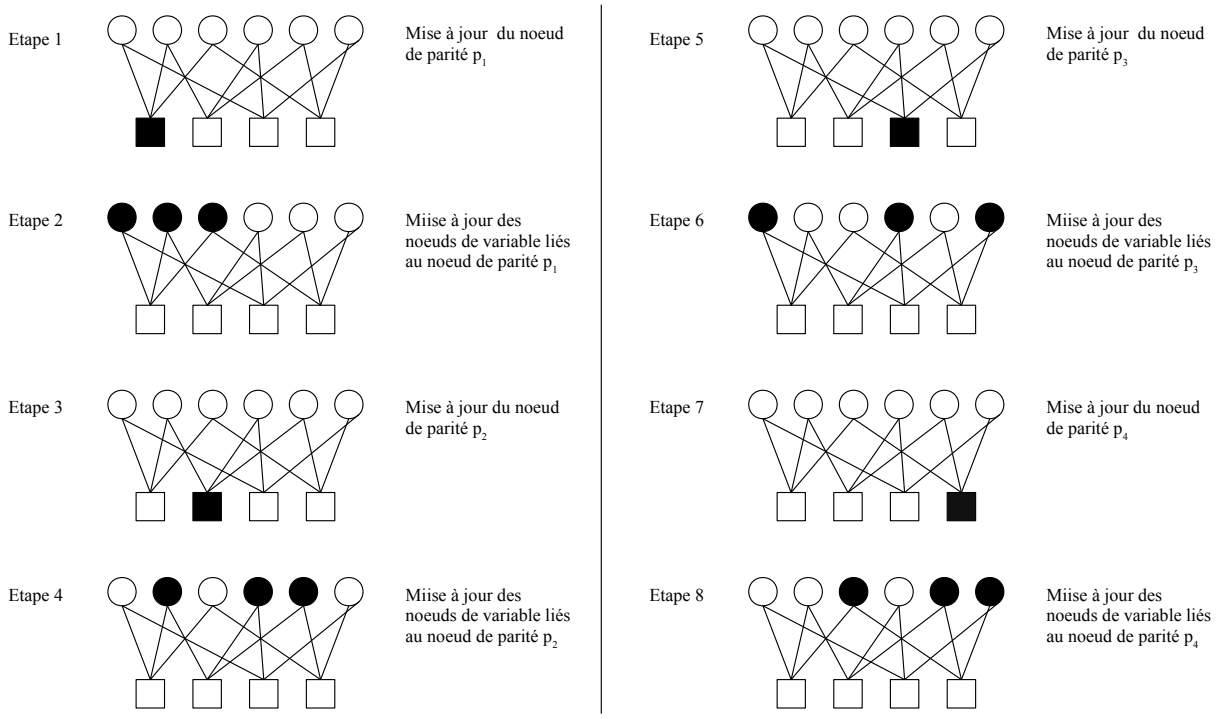


FIGURE 1.7 – Les étapes d'une itération selon l'ordonnancement à permutation horizontale

L'architecture globale du décodeur est illustrée sur la figure 1.8 et est composée d'un unique processeur de nœud parité (Check Node Processor ou CNP en anglais) connecté à d_c processeurs de nœud de variable (Variable Node Processor ou VNP en anglais). L'architecture comporte aussi un système de mémorisation contenant d_c mémoires vives (Random Access Memory ou RAM en anglais) $RAMM_{vp}$ pour sauvegarder les messages extrinsèques des VNPs et d_c mémoires vives $RAMy$ pour sauvegarder l'information du canal notée Y. A l'étape d'initialisation, les VNPs calculent les fiabilités intrinsèques en utilisant l'information du canal Y et sauvegardent les résultats dans les RAMs M_{vp}. A chaque itération, le CNP reçoit en parallèle d_c messages M_{vp}^{old} et fournit après traitement d_c messages M_{pv} qui seront immédiatement traités par les VNPs pour générer les nouveaux messages M_{vp}^{new} qui vont remplacer les anciens messages M_{vp}^{old} dans les RAMs M_{vp}. Ce processus de décodage est répété n_{iter} fois. Durant chaque itération, M mises à jour de CN et $N = d_c \times M$ mises à jour de VN sont effectuées. A la dernière itération, les VNPs déterminent séquentiellement la valeur du mot de code. Nous devons noter que puisque $d_v = 2$, ce processus de décision se fait lors de la mise à jour de la deuxième arête de chaque VN.

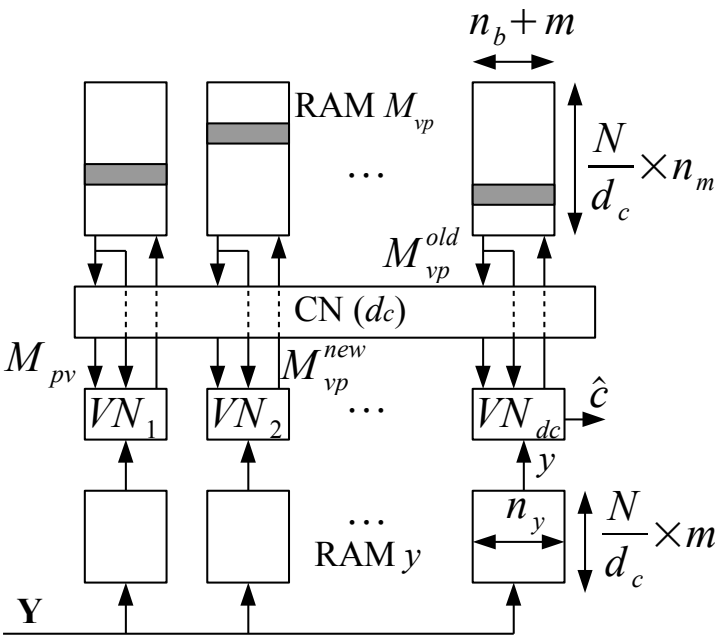


FIGURE 1.8 – Architecture globale du décodeur DAVINCI [2]

1.4.2 Système de mémorisation

Dans l'architecture proposée, les d_c VNPs fonctionnent en parallèle. Pour éviter les conflits d'accès à la mémoire, les données relatives à chaque VNP sont mémorisées dans un banc mémoire séparé. Par conséquent, chaque banc mémoire ($RAM\ y$ ou $RAMM_{vp}$) contient les données associées à $\frac{N}{d_c}$ VNs.

Les observations du canal y sont sauvegardées dans les bancs mémoires $RAM\ y$. Elles sont ensuite exploitées par les VNPs pour générer les LLRs intrinsèques. Chaque observation est associée à un symbole du mot du code (c'est-à-dire à un VN du graphe biparti). Chaque symbole du corps de Galois est codé sur m bits et l'information du canal relative à un bit du symbole est codée sur n_y bits. Les observations des bits d'un même symbole sont sauvegardées dans m adresses mémoires consécutives. Il en découle que la taille de chaque banc $RAM\ y$ est égale à $(\frac{N}{d_c} \times m) \times n_y$ bits.

Chaque message M_{vp} se compose de n_m couples $(LLR, symbole)$. Les données d'un même message sont sauvegardées dans n_m adresses mémoires consécutives. Puisque les LLRs sont codés sur n_b bits et les symboles sur m bits, la taille d'un banc $RAMM_{vp}$ est égale à $\frac{N}{d_c} \times n_m \times (m + n_b)$ bits.

La répartition des N VNs entre les bancs $RAMy$ et $RAMM_{vp}$ est un problème de coloriage dont l'objectif est d'éviter les conflits d'accès à la mémoire. Dans ce problème, d_c couleurs différentes sont assignées aux d_c bancs mémoires et les messages extrinsèques des d_c VNs liés au même CN doivent être stockés chacun dans un banc mémoire séparé. La figure 1.9 montre un exemple de répartition dans le cas d'une matrice de parité d'un code $\mathbb{GF}(64)$-LDPC de taille $N = 192$ symboles, de rendement $R = \frac{2}{3}$ et de degrés de connectivité $d_v = 2$ et $d_c = 6$. Cette répartition est faite en utilisant la méthodologie de mapping mémoire proposée dans [36]. Chaque élément de la matrice dessinée sur la figure 1.9 est le numéro d'une colonne dans la matrice de parité (i.e. l'indice d'un VN ou autrement l'indice de position d'un symbole du mot de code).

38	39	94	122	133	179
46	47	71	84	146	176
41	42	78	128	150	153
51	52	71	124	139	181
54	55	75	99	142	161
62	63	64	105	130	175
50	51	88	112	147	182
45	46	80	106	163	165
47	48	87	99	145	152
25	26	79	118	129	147
39	40	76	105	160	168
19	20	67	96	133	186
15	16	82	126	127	142
49	50	70	107	140	174
0	1	91	110	135	138
36	37	79	120	153	164
13	14	91	101	132	185
29	30	65	116	151	189
7	8	73	113	136	186
17	18	95	104	155	174
2	3	90	127	137	182
42	43	81	104	161	166
22	23	66	113	157	177
9	10	85	122	138	178
34	35	76	101	149	173
27	28	69	110	158	169
10	11	95	115	139	177
56	57	92	119	150	170
6	7	83	102	145	167
31	32	66	117	143	168
0	1	75	103	134	191
58	59	72	123	132	155
1	2	89	121	148	156
12	13	73	107	144	187
28	29	96	109	140	190
14	15	67	106	143	180
30	31	77	124	134	187
61	62	83	125	158	181
32	33	69	87	141	171
3	4	64	123	165	184
48	49	78	121	157	175
35	36	77	102	156	183
21	22	74	112	135	188
37	38	84	108	144	169
55	56	68	115	163	190
16	17	74	108	151	154
18	19	93	103	159	176
33	34	81	97	137	189
8	9	82	98	128	173
23	24	93	119	149	184
52	53	89	111	136	171
57	58	88	94	141	183
40	41	80	116	162	167
5	6	92	100	154	172
26	27	86	126	159	162
4	5	85	117	129	166
11	12	90	114	160	172
59	60	98	109	146	188
43	44	86	111	130	185
24	25	65	114	152	180
60	61	68	118	148	179
53	54	72	120	131	178
44	45	70	100	164	191
20	21	97	125	131	170

Banc 0
Banc 1
Banc 2
Banc 3
Banc 4
Banc 5

FIGURE 1.9 – Partition des VNs dans les bancs mémoires

Chaque banc mémoire est composé de $\frac{N}{d_c}$ cases mémoire. Dans le cas des $RAMs\ y$, une case est composée de m cellules. Dans le cas des $RAMM_{vp}$, une case est composée de n_m cellules. La figure 1.10 montre l'architecture des bancs mémoires dans le cas de la matrice de la figure 1.9.

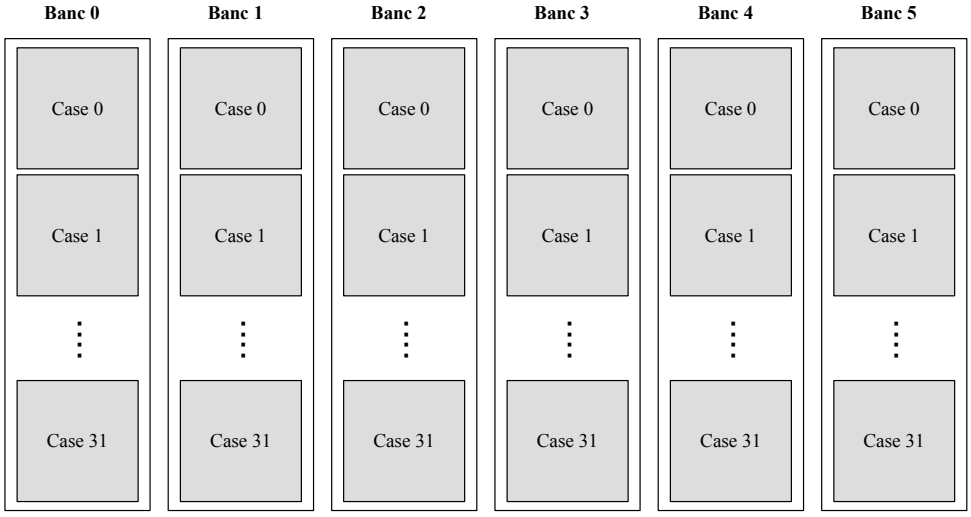


FIGURE 1.10 – Configuration du système de mémorisation

Le tableau 1.4 indique la case attribuée à chaque VN du banc 0. La première case est attribuée au VN ayant le plus petit indice, et ainsi de suite. Les autres bancs sont configurés selon le même principe.

TABLE 1.4 – Configuration du banc 0

VN	Case	VN	Case
1	0	33	16
3	1	35	17
5	2	37	18
7	3	39	19
9	4	41	20
11	5	43	21
13	6	45	22
15	7	47	23
17	8	49	24
19	9	51	25
21	10	53	26
23	11	55	27
25	12	57	28
27	13	59	29
29	14	61	30
31	15	63	31

Toutes les informations relatives à la répartition des VNs entre les bancs mémoires $RAM\ y$ et $RAMM_{vp}$ ainsi que les valeurs des symboles non nuls de la matrice de parité sont stockées dans des mémoires mortes (Read Only Memory ou ROM en anglais) qui, associées à un ensemble de compteurs, constituent la partie contrôle du décodeur.

1.4.3 Définition des LLRs

Dans ce qui suit, nous retenons la définition des LLRs présentée dans la section 1.3.5. Avec cette définition, le LLR du symbole le plus fiable vaut toujours zéro et les LLRs des autres symboles sont positifs. Par conséquent, les messages doivent être triés par ordre croissant ce qui entraîne une modification des équations 1.19 et 1.22 de l'algorithme EMS qui deviennent respectivement :

$$\gamma = M.L(n_m - 1) + \text{offset} \tag{1.28}$$

où $offset$ est un scalaire strictement positif.

$$M_s.L(k) \approx \min_{\substack{M_e^{'}.GF(k^{'})+M_e^{''}.GF(k^{''})=M_s.GF(k) \\ k^{'},k^{''}=0,1,\cdots,n_m}} \left\{M_e^{'}.L(k^{'}) + M_e^{''}.L(k^{''})\right\} \tag{1.29}$$

De même, la décision se fait par l'équation 1.30 :

$$\hat{c}_j = \underset{\beta \in \mathbb{GF}(q)}{\text{argmin}}\{\text{APP}_j[\beta]\} \quad j = 0, 1, \cdots N-1 \tag{1.30}$$

1.4.4 Architecture du processeur de nœud de variable

Les VNs ont un degré de connectivité $d_v = 2$. La figure 1.11 montre les messages échangés entre un VN v et ses deux CNs connexes p_0 et p_1.

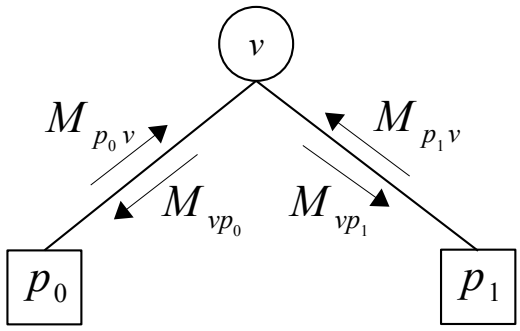


FIGURE 1.11 – VN de degré $d_v = 2$

L'architecture d'un VNP est donnée par la figure 1.12. Elle fonctionne selon trois modes :

- ◇ Génération des LLRs intrinsèques : ce mode précède les itérations de décodage et consiste à générer les messages intrinsèques par le Module de Génération Intrinsèque *iLLR*.
- ◇ Mise à jour du VN : dans ce mode, tous les modules du VNP, à part le module *Decision*, sont activés. Le VNP reçoit et met à jour un message M_{pv} en provenance du CNP. Il génère par la suite un message M_{vp}^{new} qui sera sauvegardé dans une $RAMM_{vp}$.
- ◇ Décision : à la dernière itération, le module *Decision* est activé pour déterminer le mot de code.

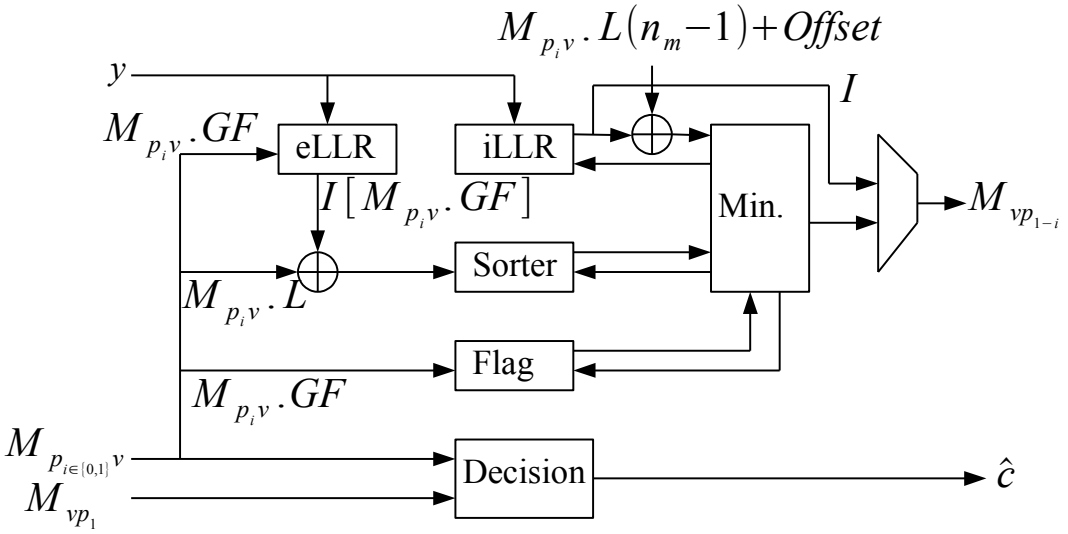


FIGURE 1.12 – Architecture du VNP

L'étape de Mise à jour du VN s'effectue par deux processus parallèles. Le premier consiste à générer, par le module *iLLR*, le message $\tilde{I}$ contenant n_m couples (LLR intrinsèque, symbole) triés par ordre croissant. Les LLRs de $\tilde{I}$ sont ensuite mis à jour en leur ajoutant la valeur $\gamma = M_{p_i v}.L(n_m - 1) + offset$.

Le deuxième processus consiste à mettre à jour les LLRs du message $M_{p_i v}$ comme suit :

$$M_{p_i v}.L[x]_{x\in\mathbb{GF}(q)} = M_{p_i v}.L[x] + I[x] \tag{1.31}$$

où $I[x]$ est généré par le module *eLLR* et correspond au LLR intrinsèque du symbole x. Le nouveau message $M_{p_i v}$ est réordonné par ordre croissant par le module *Sorter*. Le module *Flag* est un registre de q bits qui sert à indiquer la présence d'un symbole donné dans $M_{p_i v}.GF$. En effet, chaque bit est associé à un symbole du corps de Galois. Si $x \in M_{p_i v}.GF$ alors $Flag[x] = 1$ sinon $Flag[x] = 0$.

Les deux nouveaux messages $\tilde{I}$ et $M_{p_i v}$ sont stockés respectivement dans deux mémoires $FIFO_I$ et $FIFO_M$ de type premier-entrant premier-sortant (First-In First-Out ou FIFO en anglais). Le module *Min* permet de comparer les sorties des FIFOs et de générer le messages $M_{vp_{1-i}}$ en sélectionnant à chaque cycle d'horloge le symbole ayant le LLR le plus petit. Ainsi, en supposant qu'à un cycle donné les entrées du module *Min* prennent les valeurs $M_{p_i v}.L(j)$ et $\tilde{I}.L(k)$, $j \in \{0, 1, \cdots, n_m - 1\}$ et $k \in \{0, 1, \cdots, n_m - 1\}$, alors trois cas se présentent :

(a) Si $Flag[\tilde{I}.GF(k)] = 1$ alors aucune comparaison n'est faite et $FIFO_I$ s'incrémente par une position pour fournir $\tilde{I}.L(k+1)$ au comparateur. Ce procédé permet de ne pas avoir des redondances dans le message $M_{vp_{1-i}}$.

(b) Si $Flag[\tilde{I}.GF(k)] = 0$ et $M_{p_i v}.L(j) \leq \tilde{I}.L(k)$ alors $M_{p_i v}(j)$ est ajoutée à $M_{vp_{1-i}}$ et la sortie de $FIFO_M$ prend la valeur $M_{p_i v}.L(j+1)$.

(c) Si $Flag[\tilde{I}.GF(k)] = 0$ et $\tilde{I}.L(k) < M_{p_i v}.L(j)$ alors $\tilde{I}(k)$ est ajoutée à $M_{vp_{1-i}}$ et la sortie de $FIFO_I$ prend la valeur $\tilde{I}.L(k+1)$.

Architectures des modules *iLLR* et *eLLR*

Pour plus de simplicité, nous adoptons dans ce qui suit la notation $p(x|y)$ pour indiquer la probabilité conditionnelle d'avoir transmis le symbole x sachant l'observation du canal y. Par conséquent, l'équation 1.26 devient :

$$LLR(x) = -\ln \frac{p(x|y)}{\max\limits_{\theta \in \mathbb{GF}(2^m)} \{p(\theta|y)\}} \tag{1.32}$$

En utilisant le théorème de Bayes nous obtenons :

$$LLR(x) = -\ln \frac{\frac{p(y|x)p(x)}{p(y)}}{\max\limits_{\theta \in \mathbb{GF}(2^m)} \left\{\frac{p(y|\theta)p(\theta)}{p(y)}\right\}} \tag{1.33}$$

En supposant que les symboles de la source sont équiprobables nous obtenons :

$$LLR(x) = -\ln \frac{p(y|x)}{\max\limits_{\theta \in \mathbb{GF}(2^m)} \{p(y|\theta)\}} \tag{1.34}$$

En présence de bruit indépendant, $p(y|x)$ s'écrit :

$$p(y|x) = \prod_{i=0}^{m-1} p(y_i|x_i) \tag{1.35}$$

En considérant une modulation par changement de phase binaire (Binary Phase-Shift Keying ou BPSK en anglais) en émission et un canal à bruit blanc gaussien $N(0, \sigma)$, nous obtenons :

$$p(y_i|x_i) = \sqrt{\frac{1}{2\pi\sigma}} \cdot e^{-\frac{(y_i - \mathrm{BPSK}(x_i))^2}{2\sigma^2}} \tag{1.36}$$

où $\mathrm{BPSK}(x_i) = 2x_i - 1$. Soit $s_i \in \{-1, 1\}$, $i = 0, 1, \cdots, m-1$, le signe de y_i.

D'après l'équation 1.36, nous obtenons :

$$\begin{aligned} \ln p(y|x) &= m \ln \sqrt{\frac{1}{2\pi\sigma}} - \frac{1}{2\sigma^2} \cdot \sum_{i=0}^{m-1} \big(y_i - \mathrm{BPSK}(x_i)\big)^2 \\ &= m \ln \sqrt{\frac{1}{2\pi\sigma}} - \frac{1}{2\sigma^2} \cdot \sum_{i=0}^{m-1} y_i^2 + 1 + 2|y_i|s_i(1 - 2x_i) \end{aligned} \tag{1.37}$$

où $|\cdot|$ est l'opérateur valeur absolue.

Soit $\tilde{x}$ le symbole ayant la plus grande probabilité. Sa valeur est déterminée par :

$$\begin{aligned}
\tilde{x} &= \underset{x\in\mathbb{GF}(2^m)}{\operatorname{argmax}}\{p(y|x)\} \\
&= \underset{x\in\mathbb{GF}(2^m)}{\operatorname{argmax}}\{\ln p(y|x)\} \\
&= \underset{x\in\mathbb{GF}(2^m)}{\operatorname{argmax}}\{m\ln\sqrt{\frac{1}{2\pi\sigma}} - \frac{1}{2\sigma^2}\cdot\sum_{i=0}^{m-1} y_i^2 + 1 + 2|y_i|s_i(1-2x_i)\} \\
&= \underset{x\in\mathbb{GF}(2^m)}{\operatorname{argmin}}\{\sum_{i=0}^{m-1}\big(y_i^2 + 1 + 2|y_i|s_i(1-2x_i)\big)\} \\
&= \underset{x\in\mathbb{GF}(2^m)}{\operatorname{argmin}}\{\sum_{i=0}^{m-1}|y_i|s_i(1-2x_i)\} \\
&= \big(\frac{1+s_0}{2}, \frac{1+s_1}{2}, \cdots, \frac{1+s_{m-1}}{2}\big) \qquad (1.38)
\end{aligned}$$

D'une manière équivalente, nous avons :

$$\tilde{x_i} = \begin{cases} 0, & \text{si } s_i = -1 \\ 1, & \text{si } s_i = 1 \end{cases} \qquad i = 0, 1, \cdots, m-1$$

et

$$\ln p(y|\tilde{x}) = m\ln\sqrt{\frac{1}{2\pi\sigma}} - \frac{1}{2\sigma^2}\cdot\sum_{i=0}^{m-1}\big(|y_i| - 1\big)^2 \qquad (1.39)$$

L'équation 1.32 devient :

$$LLR(x) = \frac{1}{2\sigma^2}\cdot\sum_{i=0}^{m-1} 2|y_i|[s_i(1-2x_i)+1] \qquad (1.40)$$

Soit $s'_i = \frac{1-s_i}{2}$ le bit de signe de y_i ($s'_i \in \{0,1\}, i = 0, 1, \cdots, m-1$). En remplaçant dans l'équation 1.40 s_i par son expression en fonction de s'_i, nous obtenons :

$$\begin{aligned}
LLR(x) &= \frac{2}{\sigma^2}\cdot\sum_{i=0}^{m-1}|y_i|[1-(x_i+s'_i)]^2 \\
&= \frac{2}{\sigma^2}\cdot\sum_{i=0}^{m-1}|y_i||1-(x_i+s'_i)| \\
&= \frac{2}{\sigma^2}\cdot\sum_{i=0}^{m-1}|y_i|\Delta_i \qquad (1.41)
\end{aligned}$$

où

$$\Delta_i = \begin{cases} 0, & \text{si } s'_i \neq x_i \\ 1, & \text{si } s'_i = x_i \end{cases} \quad i = 0, 1, \cdots, m-1$$

Pour simplifier cette équation, nous supprimons le coefficient $\frac{2}{\sigma^2}$. La dégradation des performances qui en résulte peut être corrigée en calibrant l'*offset*. L'expression des LLRs devient :

$$LLR(x) = \sum_{i=0}^{m-1} |y_i| \Delta_i \tag{1.42}$$

L'architecture du module *iLLR* est donnée par la figure 1.13. Elle repose sur une approche systolique de génération des LLRs intrinsèques proposée dans [37].

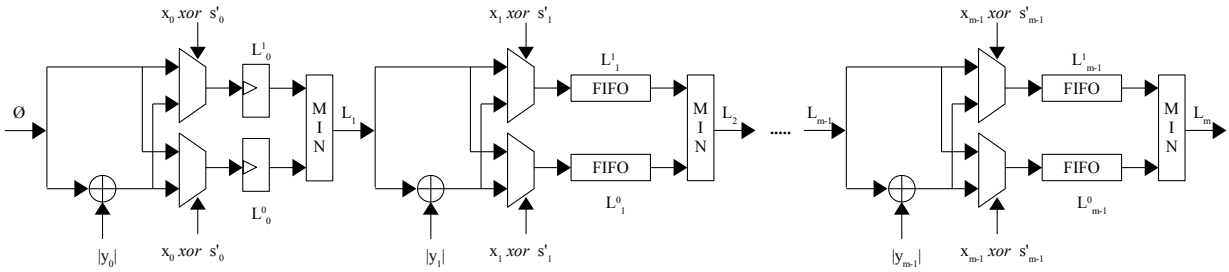


FIGURE 1.13 – Architecture du module *iLLR*

Cette architecture permet de réduire la latence de génération des LLRs intrinsèques grâce à l'utilisation de m étages fonctionnant en mode *pipeline*. Nous appelons symbole partiel d'ordre j un symbole composé de j bits, $j = 1, 2, \cdots, m$. Nous définissons le LLR partiel d'ordre j d'un symbole x par :

$$LLR_j(x) = \sum_{i=0}^{j} |y_i| \Delta_i \quad 0 \leq j < m \tag{1.43}$$

Selon la valeur de Δ_0, le premier étage génère une liste triée L_1 contenant les deux LLRs partiels qui correspondent aux bits $x_0 = 0$ et $x_0 = 1$. Le i-ème étage, $i = 1, 2, \cdots, m-1$, reçoit une liste triée L_i contenant 2^i LLRs partiels associés à 2^i symboles partiels chacun composé de i bits. Il commence par générer deux sous-listes triées L_i^0 et L_i^1. Pour $x_i \in \{0, 1\}$, les symboles partiels de $L_i^{x_i}$ sont construits en concaténant x_i aux symboles de L_i. Les LLRs partiels de $L_i^{x_i}$ sont déterminés comme suit :

$$L_i^{x_i}(k) = \begin{cases} L_i(k) + |y_i|, & \text{si } x_i = s'_i \\ L_i(k), & \text{sinon} \end{cases} \quad k = 0, 1, \cdots, 2^i - 1 \tag{1.44}$$

L_i^0 et L_i^1 sont ensuite stockées dans deux FIFOs pour être finalement comparées et générer la liste L_{i+1}.

Le fonctionnement du module *eLLR* est plus simple puisqu'il ne génère que le LLR du symbole donné à son entrée. Par conséquent, son architecture est une implantation directe de l'équation 1.42 comme le montre la figure 1.14.

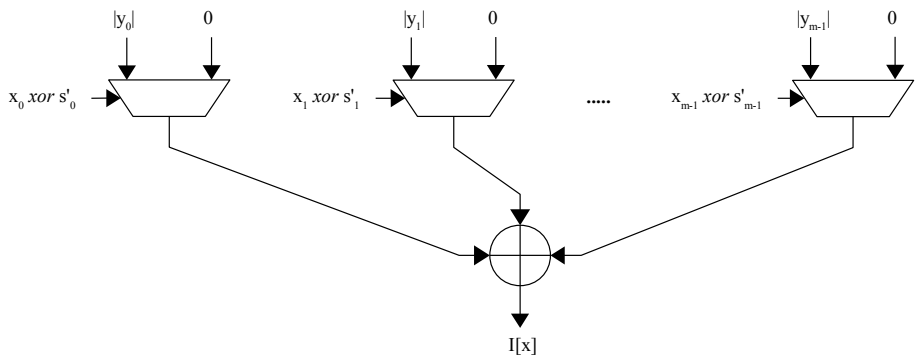


FIGURE 1.14 – Architecture du module *eLLR*

Architecture du trieur

Le module *Sorter* sert à trier par ordre croissant le message $M_{p_i v}$ après sa mise à jour par les LLRs intrinsèques générés à l'aide du module *eLLR*. L'architecture du module *Sorter* est illustrée sur la figure 1.15 et se compose de $\lceil \log_2 n_m \rceil$ étages fonctionnant en mode *pipeline* ($\lceil \cdot \rceil$ étant l'opérateur partie entière supérieure).

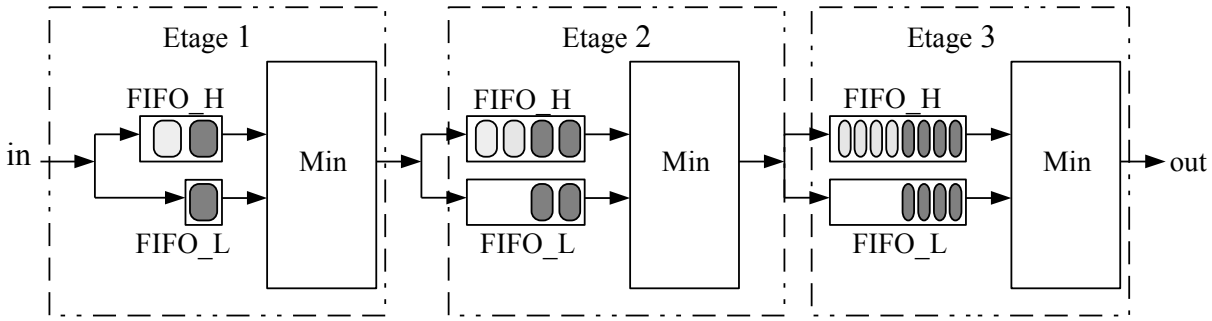


FIGURE 1.15 – Architecture du module *Sorter*

Le premier étage compare les données une par une pour générer une liste de données triées deux par deux. Le deuxième étage reçoit cette liste, compare les données deux à deux pour générer une liste de données triées quatre par quatre, et ainsi de suite jusqu'à trier toutes les données au niveau du dernier étage. Autrement dit, le i-ème étage, $i = 0, 1, \cdots, \lceil \log_2 n_m \rceil - 1$, reçoit une liste dont chaque 2^i éléments consécutifs sont déjà triés. Chaque deux sous-listes consécutives (chacune comportant 2^i éléments triés) sont respectivement stockées dans deux FIFOs *FIFO_H* et *FIFO_L*. Les éléments de ces deux sous-listes sont comparés pour générer une sous-liste comportant 2^{i+1} éléments triés par ordre croissant.

Architecture du module *Decision*

A la dernière itération, la décision se fait dès la réception du message $M_{p_1 v}$ de la deuxième arête. Soient I le vecteur des LLRs intrinsèques de tous les symboles de $\mathbb{GF}(q = 2^m)$. La méthode optimale pour faire la décision nécessite le détermination du vecteur APP comme suit :

$$APP[x]_{x\in\mathbb{GF}(q)} = I[x] + W_0 + W_1 \tag{1.45}$$

où

$$W_{i=0,1} = \begin{cases} M_{p_i v}[x], & \text{si } x \in M_{p_i v}.GF \\ M_{p_i v}(n_m - 1) + offset, & \text{sinon} \end{cases}$$

La décision se fait en sélectionnant le symbole ayant le LLR le plus faible.

$$\hat{c} = \underset{x\in\mathbb{GF}(2^m)}{\text{argmin}} \{APP[x]\} \tag{1.46}$$

Nous considérons les ensembles $\Lambda_1 = M_{vp_1}.GF \cap M_{p_1 v}.GF$, $\Lambda_2 = M_{vp_1}.GF - \Lambda_1$ et $\Lambda_3 = M_{p_1 v}.GF - \Lambda_1$. Le cardinal de $\Lambda_1 \cup \Lambda_2 \cup \Lambda_3$ est compris entre n_m et $2n_m$. L'équation 1.45 peut être modifiée sans perte de performance de la manière suivante :

$$APP[x]_{x\in\Lambda_1\cup\Lambda_2\cup\Lambda_3} = \begin{cases} M_{vp_1}.L[x] + M_{p_1 v}.L[x], & \text{si } x \in \Lambda_1 \\ M_{vp_1}.L[x] + M_{p_1 v}.L(n_m - 1) + offset, & \text{si } x \in \Lambda_2 \\ M_{p_1 v}.L[x] + M_{vp_1}.L(n_m - 1) + offset, & \text{sinon} \end{cases} \tag{1.47}$$

L'équation 1.46 se transforme en :

$$\hat{c} = \underset{x\in\Lambda_1\cup\Lambda_2\cup\Lambda_3}{\text{argmin}} \{APP[x]\} \tag{1.48}$$

L'implantation matérielle de l'équation 1.48 est complexe puisqu'il faut effectuer une recherche exhaustive ou bien utiliser une mémoire adressable par contenu (Content Adressable Memory ou CAM en anglais) de taille n_m pour déterminer les ensembles Λ_1, Λ_2 et Λ_3. Afin de réduire la taille de la mémoire CAM, une méthode de décision sous-optimale a été proposée. Elle ne considère que les $n_s \ll n_m$ symboles les plus fiables de M_{vp_1}. Le vecteur APP est alors construit comme suit :

$$APP[x]_{x\in M_{p_1 v}.GF} = \begin{cases} M_{vp_1}.L[x] + M_{p_1 v}.L[x], & \text{si } x \in \{M_{vp_1}.GF(i)\}_{i=1,\cdots,n_s-1} \\ M_{p_1 v}.L[x] + M_{vp_1}.L(n_s - 1), & \text{sinon} \end{cases} \tag{1.49}$$

La décision se fait par :

$$\hat{c} = \underset{x\in M_{p_1 v}.GF\cup M_{vp_1}.GF(0)}{\text{argmin}} \{M_{vp_1}.L(0), APP[x]\} \tag{1.50}$$

L'architecture associée à cette algorithme est donnée dans la figure 1.16. Elle utilise une CAM de taille n_s au lieu de n_m.

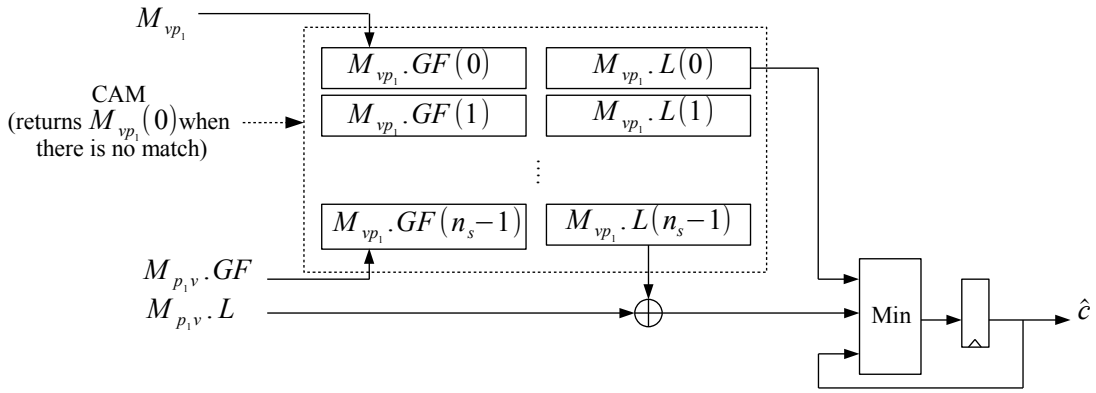


FIGURE 1.16 – Architecture du module *Decision*

1.4.5 Architecture du processeur de nœud de parité

Le CNP reçoit d_c messages M_{v_ip} et génère d_c messages M_{pv_i} qui seront transmis aux VNPs, $i = 0, 1, \cdots, d_c - 1$. L'architecture du noeud de parité utilise la récursion Forward-Backward de la figure 1.4. Elle se compose de $d_c - 2$ ECNs. Un ECN reçoit deux messages triés $M_e^{'}$ et $M_e^{''}$ comprenant chacun n_m éléments triés dans le sens croissant (le symbole le plus fiable étant celui ayant la plus petite valeur LLR) et génère un message trié M_s. Le rôle de l'ECN est d'extraire les n_m symboles les plus fiables (ayant les plus petites valeurs LLR) de la matrice T_M.

Contrairement à la solution illustrée dans la figure 1.5 qui consiste à explorer toute la matrice et utiliser un trieur de taille n_m éléments, l'ECN développé dans le cadre du projet DAVINCI utilise l'algorithme L-Bubble Check [38, 39] qui repose sur la supposition que les n_m symboles plus fiables sont essentiellement distribués sur les deux premières lignes et les deux premières colonnes. De cette manière la complexité du problème est réduite de l'ordre de $\sqrt{n_m}$. L'algorithme L-Bubble ne nécessite qu'un trieur de taille 4 éléments quelque soit la valeur prise par n_m. La mise à jour des candidats utilise 4 parcours de sélection comme le montre la figure 1.17. Pour palier à la présence de symboles redondants dans T_M, le nombre d'opérations maximal est fixé à $n_{oper} \geq n_m$.

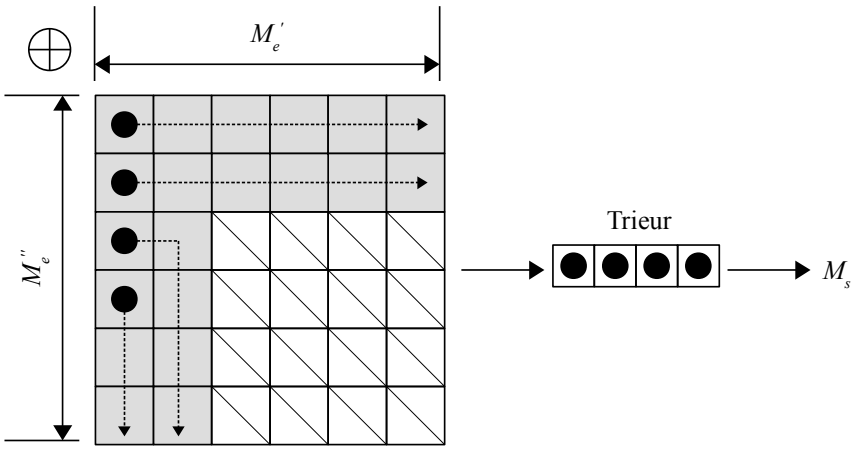


FIGURE 1.17 – Principe de l'algorithme L-Bubble Check

L'architecture proposée pour implanter l'algorithme L-Bubble Check est donnée dans la

figure 1.18. Elle se compose essentiellement de deux mémoires RAM pour stocker les deux messages d'entrée $M_e^{'}$ et $M_e^{''}$, une table d'indices pour sauvegarder la position dans la matrice T_M des derniers candidats introduits dans le trieur. Le Trieur contient quatre registres R_0, R_1, R_2 et R_3 pour mémoriser les candidats, quatre multiplexeurs et un comparateur *Min*. Les quatre candidats introduits dans *Min* sont appelé bulles (Bubble en anglais) et sont notés B_0, B_1, B_2, B_3. La mise à jour des candidats se fait par le module *Update*.

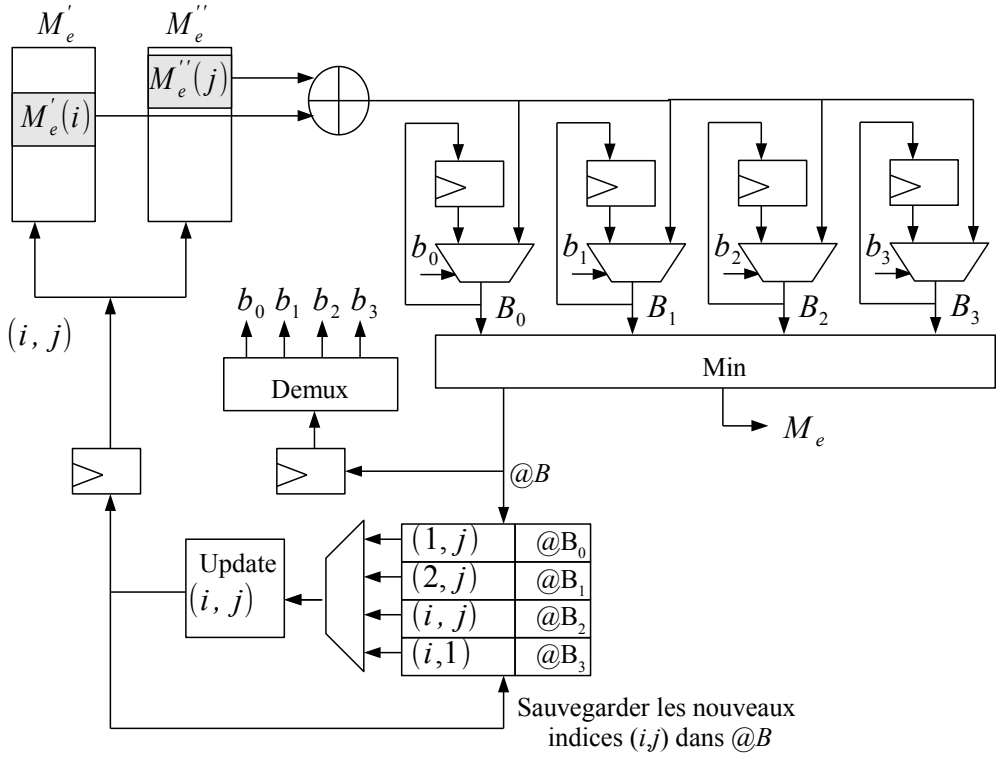


FIGURE 1.18 – Architecture L-Bubble Check

L'ECN fonctionne de la manière suivante :

(a) Extraire $M_e^{'}(i)$ et $M_e^{''}(j)$ des deux RAMs.

(b) Calculer le nouveau candidat $T_M(i,j) = M_e^{'}(i) + M_e^{''}(j)$. Ce candidat alimente le trieur pour remplacer le candidat sélectionné par *Min* au cycle précédant. Le nouveau candidat est introduit directement dans le comparateur et le registre correspondant est ignoré afin de gagner un cycle d'horloge.

(c) Le comparateur *Min* détermine l'indice du candidat le plus fiable $@B = \underset{k=0,1,2,3}{\text{argmin}}\{B_k\}$. Le candidat sélectionné est ajouté au message M_s. Si le nouveau candidat n'a pas été sélectionné alors il est mémorisé dans le registre correspondant.

(d) L'indice du candidat sélectionné $@B$ est fourni à la table d'indices et au module *Update* pour déterminer les indices du candidat qui va remplacer $@B$ dans le cycle prochain. La règle de mise à jour est la suivante :

$$(i,j) = \begin{cases} (i, j+1), & \text{si } @B = 0 \text{ ou } 1 \\ (3, 2), & \text{si } @B = 3 \text{ et } j = 1 \\ (i+1, j), & \text{sinon} \end{cases}$$

Nous signalons que l'algorithme L-Bubble Check et l'architecture associée ont fait l'objet d'un dépôt de brevet [40].

1.5 Conclusion

Dans ce chapitre, nous avons commencé par rappeler quelques notions mathématiques de base qui ont servi pour généraliser la définition des codes LDPC sur les corps de Galois $\mathbb{GF}(q = 2^m)$. Nous avons ensuite discuté des algorithmes de décodage LDPC non binaire en décrivant en détail les algorithmes BP, log-BP et EMS. Enfin, nous avons présenté une architecture d'un décodeur LDPC non binaire qui a été conçue par notre laboratoire de recherche dans le but de réduire le coût d'implantation de l'algorithme EMS. Le chapitre suivant présente la première contribution de la thèse. Il est consacré à la description de l'architecture du décodeur EMS que nous avons développé en partant d'une analyse des points faibles de l'architecture du projet DAVINCI.

Chapitre 2

Optimisation de l'architecture DAVINCI

Sommaire

L'architecture du décodeur proposé dans le cadre du projet DAVINCI a été décrite en détail dans la section 1.4. C'est une architecture en série qui se compose essentiellement d'un seul processeur de nœud de parité CNP, d_c processeurs de nœud de variable VNPs et un système de mémorisation (stockage des données et entrelacement des nœuds de variables). En ré-examinant cette architecture dans tous ses aspects, nous trouvons que des améliorations peuvent être apportées sur des parties du décodeur notamment le processeur de nœud de parité et le processeur de nœud de variable. Dans la première section de ce chapitre, nous identifions les aspects qui nous semblent nécessiter des améliorations. Dans la seconde section nous proposons une nouvelle architecture en se basant sur les conclusions retenues lors de la première section.

2.1 Analyse de l'architecture du projet DAVINCI

2.1.1 Système de mémorisation

Dans l'architecture du projet DAVINCI, les nœuds de variables sont partitionnés sur d_c bancs de mémoire comme dans la figure 1.10. Chaque banc de mémoire contient $\frac{N}{d_c}$ cases. Chaque case est associée à un unique nœud de variable. Une case d'un banc mémoire RAM y contient m cellules pour stocker l'observation du canal d'un nœud de variable. De même, une case d'un banc mémoire RAM M_{vp} contient n_m cellules pour stocker les couples (*LLR*, *symbole*) générés par un nœud de variable. Le partitionnement des nœuds de variable en utilisant la technique de coloriage illustrée par la figure 1.9 nous semble un choix judicieux puisqu'elle permet de résoudre les conflits d'accès à la mémoire en évitant de mettre dans le même banc deux nœuds de variable liés au même nœud de parité. Cependant, la répartition des données sur plusieurs cellules implique que le stockage et l'extraction de ces données doivent se faire sur plusieurs cycles d'horloge. Par exemple,

le CNP nécessite n_m cycles d'horloges pour extraire un message M_{vp}. Pendant ce temps, la mémoire reste occupée et les VNPs ne peuvent pas avoir accès pour stocker les nouveaux messages M_{vp}. Cette configuration rend difficile l'ordonnancement en *pipeline* du CNP et des VNPs, à moins d'utiliser une mémoire à double accès (Dual-Port Memory en anglais). Puisque utiliser une mémoire à un seul accès (Single-Port Memory en anglais) est moins coûteux, nous modifions la structure des cases en compactant tous les éléments d'un message M_{vp} dans une seule cellule accessible en un seul cycle d'écriture ou de lecture.

2.1.2 Processeur de nœud de parité

La complexité des décodeurs LDPC non binaires provient en grande partie du CNP. Le décodeur du projet DAVINCI utilise un CNP implanté sous la forme de plusieurs ECNs connectés entre eux selon un schéma *Avant-Arrière*. La recherche des candidats se fait par l'algorithme L-Bubble Check qui permet de réduire la complexité des ECNs en considérant d'une part uniquement les deux première lignes et les deux premières colonnes de la matrice T_M et en utilisant un comparateur de 4 éléments. Cependant, pour mettre à jour la liste des candidats, il faut attendre d'abord la décision du comparateur pour connaître l'indice du candidat retenu. Pour remplacer ce candidat, il faut ensuite déterminer les adresses des deux éléments de $M_e^{'}$ et $M_e^{''}$ qui servent à calculer le nouveau candidat. Ce mécanisme de *feedback* s'accompagne d'une réduction de la fréquence maximale du circuit due à l'augmentation de son chemin critique. Pour couper en deux ce chemin critique nous proposons de supprimer le *feedback* en séparant le processus de génération des candidats de celui de la comparaison. Cette séparation se fait en empilant dans quatre mémoires FIFO les candidats déterminés au fur et à mesure de la réception des messages d'entrée $M_e^{'}$ et $M_e^{''}$. Ce calcul à la volée permet de supprimer les deux mémoires RAMs (bien que remplacées par quatre mémoires FIFO) utilisées pour stocker les deux messages $M_e^{'}$ et $M_e^{''}$. De plus, il permet de réduire la complexité des ECNs puisque nous n'avons pas à déterminer à chaque fois la position dans la matrice T_M du nouveau candidat. Finalement, comme le montre la figure 1.17, l'algorithme L-Bubble Check définit un chemin sous forme de « L » pour mettre à jour le troisième candidat de la liste. Pour se débarrasser de cette ambiguïté inutile nous proposons une nouvelle variante de l'algorithme L-Bubble Check que nous baptisons Straight-Bubble Check ou tout court S-Bubble Check. Cette variante et l'architecture associée sont détaillées dans la section 2.2.

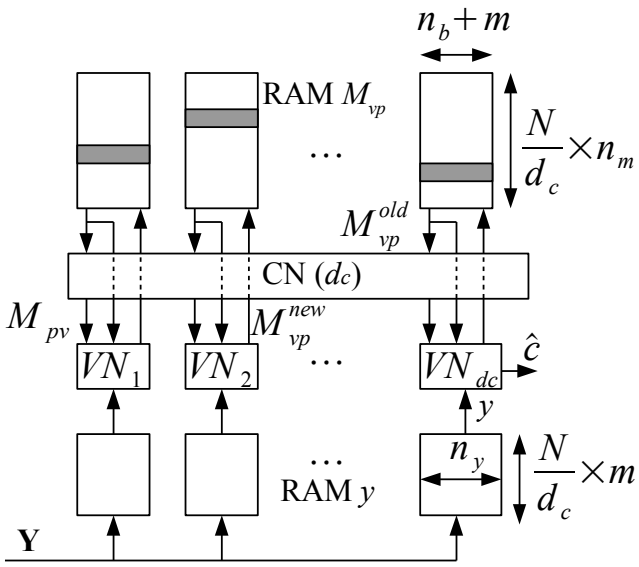
2.1.3 Processeur de nœud de variable

Les résultats du projet DAVINCI montrent que les VNPs possèdent aussi un coût d'implantation élevé. En analysant l'architecture, nous trouvons qu'il est possible de réduire ce coût en supprimant le module *iLLR* qui peut être remplacé par le module *eLLR* pour assurer la génération des messages intrinsèques lors de la phase d'initialisation du décodeur. Ces messages doivent être sauvegardés dans une mémoire RAM pour servir ensuite à la mise à jour des VNs. Pour optimiser la taille de la mémoire, il suffit de sauvegarder uniquement les symboles puisque les LLRs associés peuvent être retrouvés grâce au module *eLLR*. Un autre moyen permettant de réduire encore plus la complexité des VNPs consiste à modifier l'architecture du module *Sorter* pour pouvoir supprimer les deux mémoires FIFOs utilisées en amont du comparateur. Enfin, nous trouvons que l'algorithme

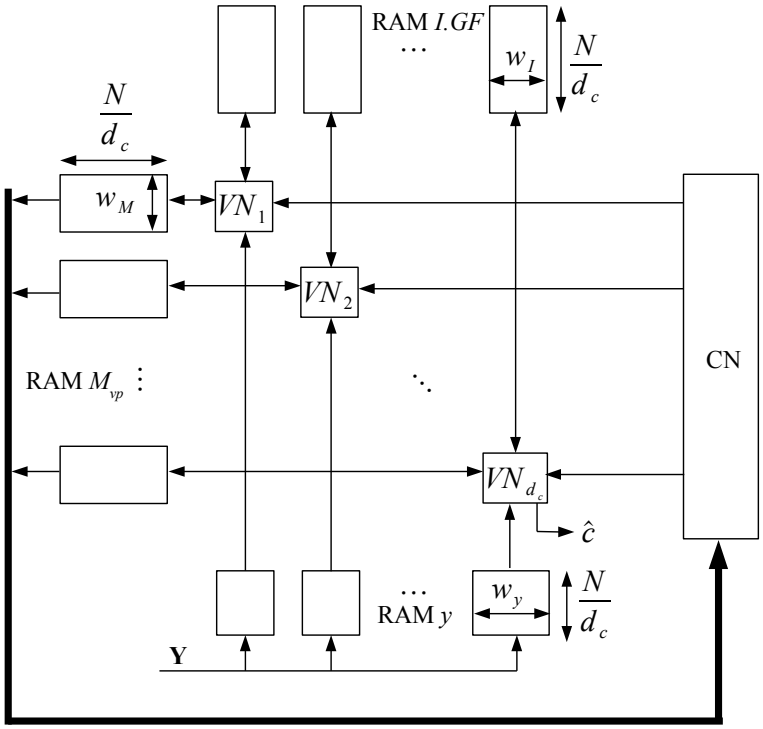
de décision utilisé par les VNPs n'est pas clairement justifié et nécessite une rectification. Par exemple, le choix de considérer dans l'équation 1.50 l'élément $M_{vp_1}(0)$ sans le mettre à jour n'est pas la meilleure solution.

2.2 Architecture du nouveau décodeur

La figure 2.1b montre l'architecture du décodeur que nous proposons.



(a) Architecture du projet DAVINCI [2]



(b) Nouvelle architecture du décodeur

FIGURE 2.1 – Architecture du décodeur

Pour simplifier l'architecture des VNPs nous proposons en premier lieu de supprimer le module *iLLR* et de sauvegarder les messages intrinsèques dans des bancs mémoire dédiés. De plus, nous remarquons qu'il suffit de sauvegarder uniquement les symboles $I.GF$ des messages intrinsèques puisque les LLRs associés peuvent être régénérés par le module *eLLR*. La taille des mots de la mémoire est alors divisée par $\frac{m+n_b}{m}$. Afin de réduire encore cette taille, nous proposons, en mode de mise à jour des VNs, d'utiliser uniquement les $n_\alpha \ll n_m$ symboles les plus fiables du message intrinsèque. Pour cela, nous modifions l'architecture globale du décodeur comme illustrée dans la figure 2.1. Par rapport à l'architecture du projet DAVINCI nous ajoutons d_c bancs mémoire RAM $I.GF$ pour sauvegarder les n_α symboles les plus fiables de chaque message intrinsèque. Tous les bancs mémoire sont configurés comme dans la figure 1.10 et le partitionnement des VNs est identique à celui de la figure 1.9. Chaque case d'un banc mémoire RAM M_{vp} contient la totalité d'un message extrinsèque soit $w_M = (m + n_b) \cdot n_m$ bits, chaque case d'un banc mémoire RAM $I.GF$ contient n_α symboles d'un message intrinsèque soit $w_I = m \cdot n_\alpha$ bits et chaque case d'un banc mémoire RAM y contient l'observation du canal associée à un VN soit $w_y = n_y \cdot m$ bits.

Le décodeur fonctionne selon 4 modes :

(a) Initialisation : le décodeur commence par recevoir les observations du canal et les sauvegarder dans les bancs mémoire RAM y.

(b) Génération des messages intrinsèques : en utilisant les observations du canal, les d_c VNPs sont activés pour générer les messages intrinsèques qui seront sauvegardés dans les bancs mémoire RAM M_{vp}. Les n_α symboles les plus fiables de chaque message intrinsèque sont également mémorisés dans les bancs mémoire RAM $I.GF$.

(c) Décodage : le décodeur effectue n_{iter} itérations de décodage. Dans une itération, les CNs sont mis à jour en série un par un. Le CNP reçoit en parallèle d_c messages M_{vp} et génère d_c messages M_{pv}. Les messages M_{pv} sont directement transmis aux d_c VNPs pour calculer les nouveaux messages M_{vp} qui seront mémorisés dans les bancs mémoire RAM M_{vp} à la place des anciens messages. Dans la dernière itération, étant donné que le degré de connectivité des VNs est limité à $d_v = 2$, un VNP qui reçoit son deuxième message M_{vp} effectue une opération de décision.

(d) Décision : Le décodeur fournit une décision $\hat{c}$.

2.2.1 Architecture du processeur de nœud de variable

Le VNP fonctionne selon trois modes. Dans le but de simplifier la description de l'architecture du VNP un schéma est donné pour chaque mode dans lequel les modules non utilisés sont omis.

Mode de génération des LLRs intrinsèques

Dans ce mode, chaque VNP génère les vecteurs de LLRs intrinsèques de $\frac{N}{d_c}$ VNs. Un compteur permet de générer les q symboles du corps de Galois. Ensuite, en utilisant l'observation du canal le module *eLLR* génère le LLR intrinsèque de chaque symbole. Le module *Sorter* permet d'obtenir un message intrinsèque I trié par ordre croissant. Les n_m couples (symbole, LLR) les plus fiables de I (ceux ayant les plus petits LLRs)

sont sauvegardés dans la mémoire RAM M_{vp} tandis que la mémoire RAM $I.GF$ sauvegarde seulement les n_α symboles les plus fiables (les LLRs ne sont pas sauvegardés puisqu'ils peuvent être régénérés par le module $eLLR$ ce qui permet de réduire la taille de la mémoire). La figure 2.2 décrit l'architecture du VNP en mode de génération des LLRs intrinsèques.

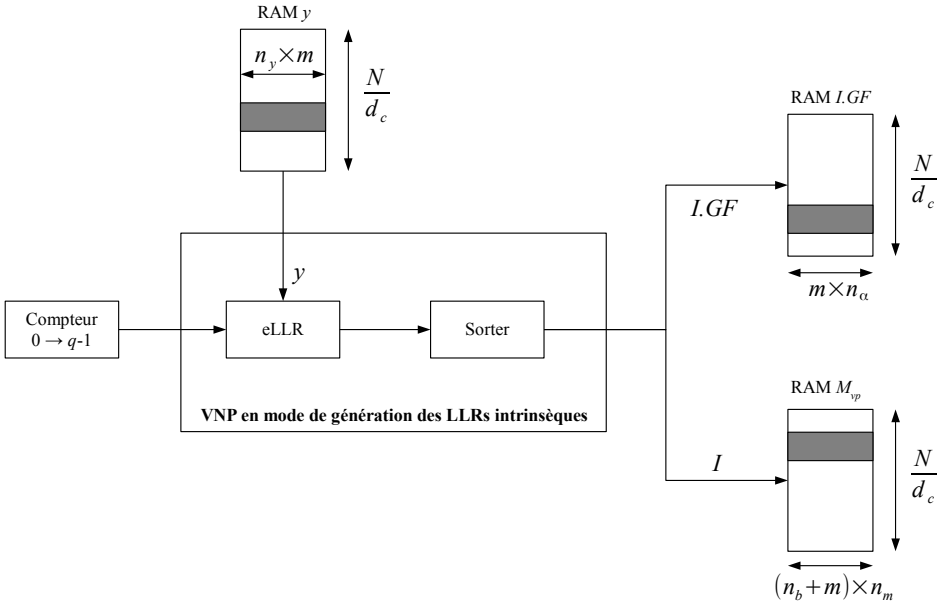


FIGURE 2.2 – Architecture du VNP en mode génération des LLRs intrinsèques

La figure 2.3 montre le diagramme de temps du VNP en mode de génération des LLRs intrinsèques. Pour chaque VN, le VNP commence par extraire l'information du canal correspondante pendant un cycle d'horloge. Le VNP reçoit aussi les q symboles du corps de Galois en série pendant q cycles d'horloge. La latence du VNP est $L_{\text{VNP},0} = q+2$ cycles d'horloge.

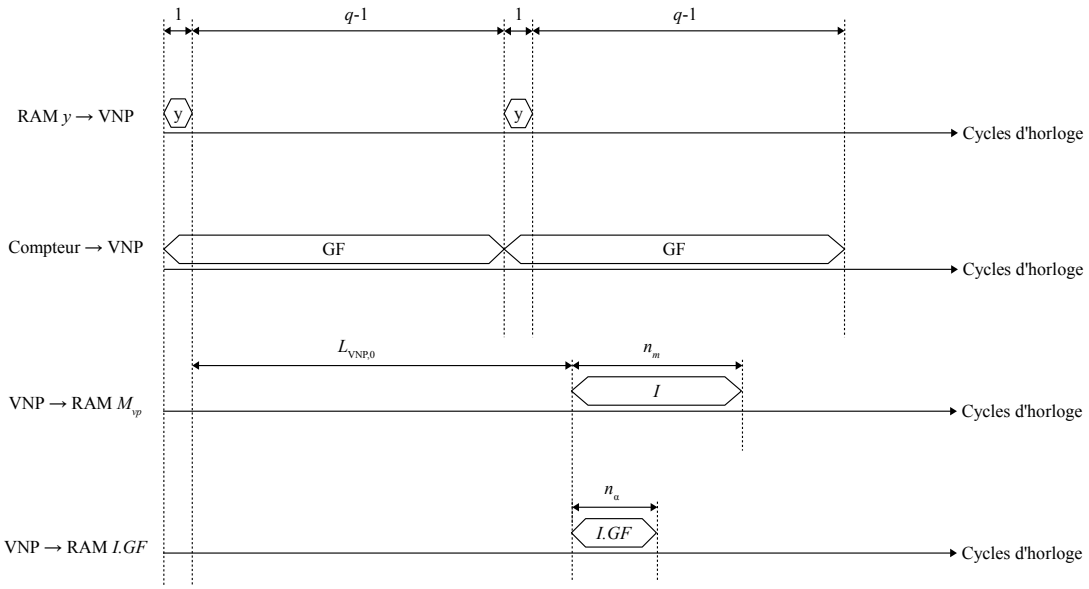


FIGURE 2.3 – Diagramme de temps du VNP en mode de génération des LLRs intrinsèques

Mode de mise à jour des VNs

L'architecture du VNP en mode de mise à jour des VNs est représentée par la figure 2.4.

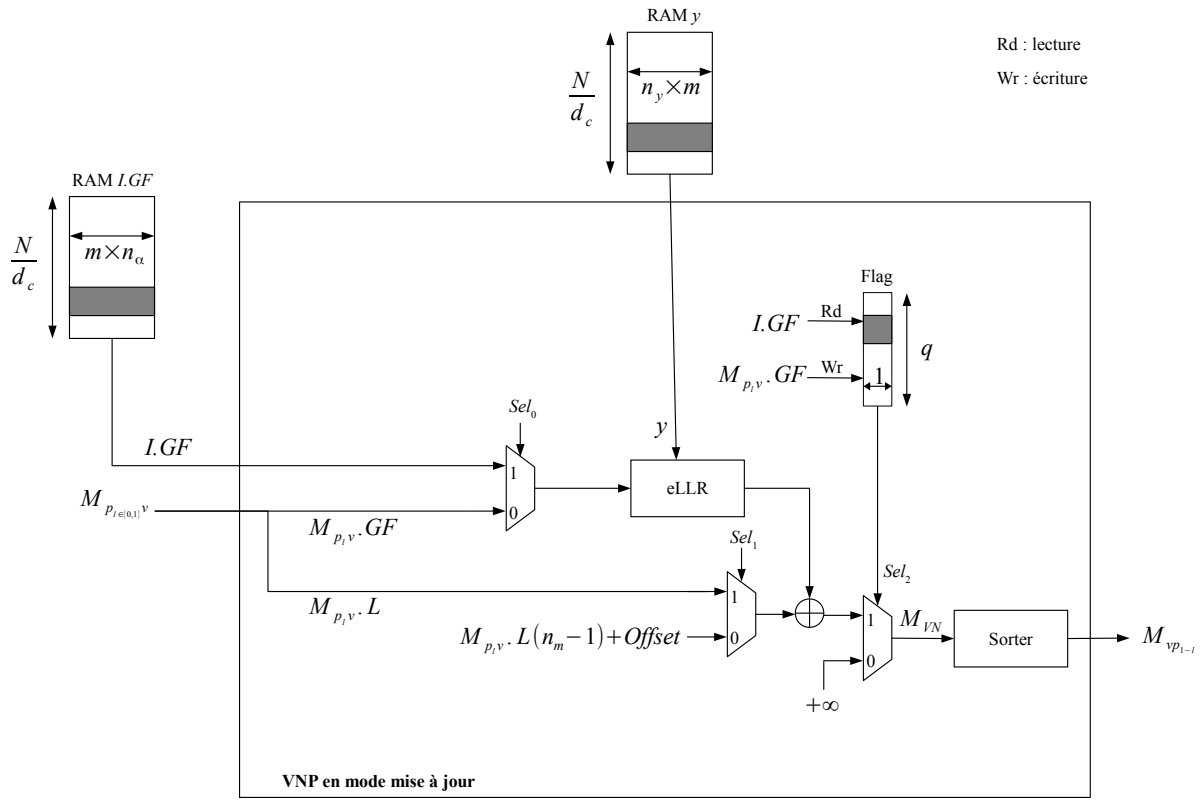


FIGURE 2.4 – Architecture du VNP en mode mise à jour

La mise à jour d'un VN se déroule en deux étapes. Dans un premier temps, le VNP met à jour le message extrinsèque $M_{p_l v}$ en provenance du CNP. Par conséquent, le signal Sel_0 prend la valeur 0 et les signaux Sel_1 et Sel_2 prennent la valeur 1. La mise à jour du message $M_{p_l v}$ se fait par l'équation 2.1 :

$$\begin{cases} M_{\mathrm{VN}}.GF(i) = M_{p_l v}.GF(i), \\ M_{\mathrm{VN}}.L(i) = M_{p_l v}.L(i) + I.LLR[M_{p_l v}.GF(i)], \end{cases} \quad i = 0, 1, \cdots, n_m - 1 \tag{2.1}$$

Les LLRs intrinsèques utilisés pour la mise à jour des LLRs du messages $M_{p_l v}$ sont au fur et à mesure générés par le module *eLLR*. Le module *Flag* est un registre de q bits (chaque bit correspond à un symbole du corps de Galois). Les bits de ce registre sont mis à jour par l'équation 2.2 :

$$Flag[M_{p_l v}.GF(i)] = 1, \quad i = 0, 1, \cdots, n_m - 1 \tag{2.2}$$

Dans un second temps, le VNP met à jour les LLRs des n_α symboles du message $I.GF$. Par conséquent, le signal Sel_0 prend la valeur 1 et le signal Sel_1 prend la valeur 0. Si un symbole du message $I.GF$ appartenait aussi au message $M_{p_l v}$ (c'est-à-dire si $Flag[I.GF(i)] = 1$, $i = 0, 1, \cdots, n_\alpha - 1$) alors le signal Sel_2 prend la valeur 0 et le LLR associé à ce symbole est saturé à la valeur $+\infty$ pour ne pas avoir des symboles en double dans le message sortant du VNP. L'équation de mise à jour du message $I.GF$ est 2.3 :

$$
\begin{aligned}
M_{\mathrm{VN}}.GF(n_m+i) &= I.GF(i) \\
M_{\mathrm{VN}}.L(n_m+i) &= \begin{cases} I.LLR(i) + M_{p_l v}.GF(n_m-1) + \mathit{offset} & \text{si Flag}[I.GF(i)] = 0 \\ +\infty & \text{sinon} \end{cases} \\
i &= 0, 1, \cdots, n_\alpha - 1
\end{aligned}
\tag{2.3}
$$

Le message M_{VN} de taille (n_m+n_α) est au fur et à mesure introduit dans le module *Sorter* pour être trié par ordre croissant. Le message $M_{vp_{1-l}}$ est obtenu en sélectionnant les n_m premiers éléments sortants du module *Sorter*.

La figure 2.5 montre le diagramme de temps de notre VNP en mode de mise à jour des VNs. La latence du VNP est $L_{\mathrm{VNP},1} = n_m + n_\alpha + 2$.

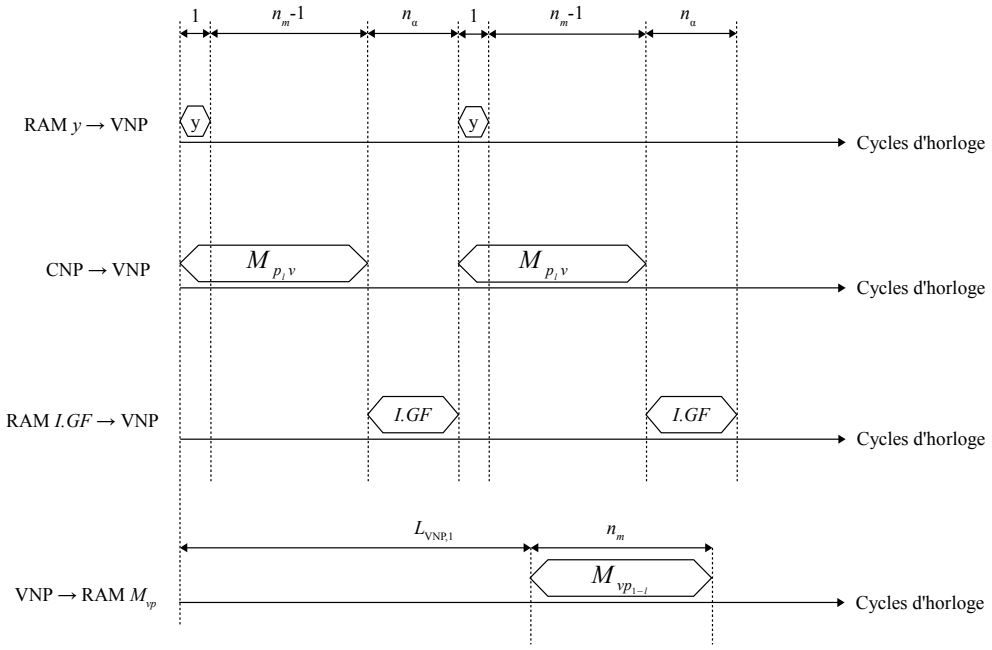


FIGURE 2.5 – Diagramme de temps du VNP en mode de mise à jour des VNs

Mode de décision

Dans le mode de décision, nous considérons uniquement les n_s premiers éléments du message M_{vp_1} afin de réduire la taille de la mémoire CAM. Par conséquent, nous définissons les ensembles Λ_1, Λ_2 et Λ_3 comme suit :

$\Lambda_1 = \{M_{vp_1}.GF(i)\}_{i=0,1,\cdots,n_s-1} \cap M_{p_1 v}.GF$

$\Lambda_2 = \{M_{vp_1}.GF(i)\}_{i=0,1,\cdots,n_s-1} - \Lambda_1$

$\Lambda_3 = M_{p_1 v}.GF - \Lambda_1$

Le calcul de l'information *a priori* APP se fait par l'équation 2.4 :

$$APP[x]_{x\in\Lambda_1\cup\Lambda_2\cup\Lambda_3} = \begin{cases} M_{vp_1}.L[x] + M_{p_1v}.L[x], & \text{si } x \in \Lambda_1 \\ M_{vp_1}.L[x] + M_{p_1v}.L(n_m-1) + offset, & \text{si } x \in \Lambda_2 \\ M_{p_1v}.L[x] + M_{vp_1}.L(n_s-1) + offset, & \text{sinon} \end{cases} \tag{2.4}$$

Ensuite, la décision se fait par l'équation 2.5 :

$$\hat{c} = \underset{x\in\Lambda_1\cup\Lambda_2\cup\Lambda_3}{\operatorname{argmin}} \{APP[x]\} \tag{2.5}$$

Nous proposons d'implanter l'algorithme de décision à l'aide de l'architecture illustrée par la figure 2.6.

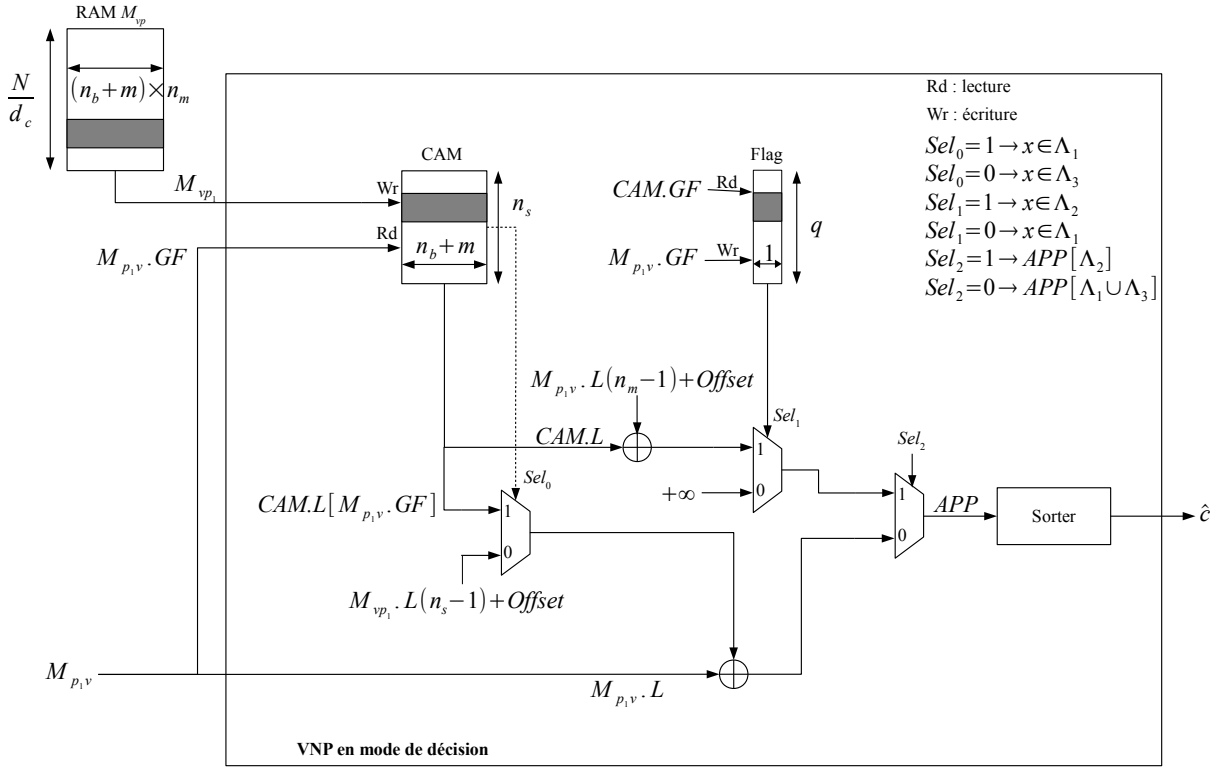


FIGURE 2.6 – Architecture du VNP en mode décision

Le VNP reçoit d'abord les n_s premiers éléments du message M_{vp_1} et les mémorise dans la CAM. Ensuite, il calcule le message APP en deux phases :

- ◇ *Phase* 1 : Durant cette phase, le VNP calcule les valeurs APP associées à l'ensemble $\Lambda_1 \cup \Lambda_3$ (les symboles du message M_{p_1v}). Le signal Sel_2 prend alors la valeur 0. Les symboles de l'ensemble Λ_1 sont repérés par la mémoire CAM. A chaque fois qu'un symbole de Λ_1 est repéré le signal Sel_0 prend la valeur 1 sinon il prend la valeur 0. Le registre *Flag* mémorise les symboles de l'ensemble $\Lambda_1 \cup \Lambda_3$ ($Flag[M_{p_1v}.GF(i)] = 1$, $i = 0, 1, \cdots, n_m - 1$) pour pouvoir les repérer durant la deuxième phase.
- ◇ *Phase* 2 : Durant cette phase, le VNP calcule les valeurs APP de l'ensemble Λ_2 (les symboles mémorisés dans la CAM sauf ceux de l'ensemble Λ_1). Le signal Sel_2 prend alors la valeur 1. Les symboles de la CAM appartenant à l'ensemble Λ_1 sont repérés par le registre *Flag*. En présence de ces symboles le signal Sel_1 prend la valeur 0 et les valeurs APP associées sont forcées à $+\infty$ pour ne pas être traitées par le module *Sorter*.

Durant les deux phases, les valeurs APP sont au fur et à mesure introduites dans le module *Sorter*. La décision est prise en sélectionnant le symbole ayant la plus petite valeur APP.

La figure 2.7 illustre le diagramme de temps du VNP en mode de décision. La latence du VNP est $L_{\text{VNP},2} = n_m + n_s + 2$.

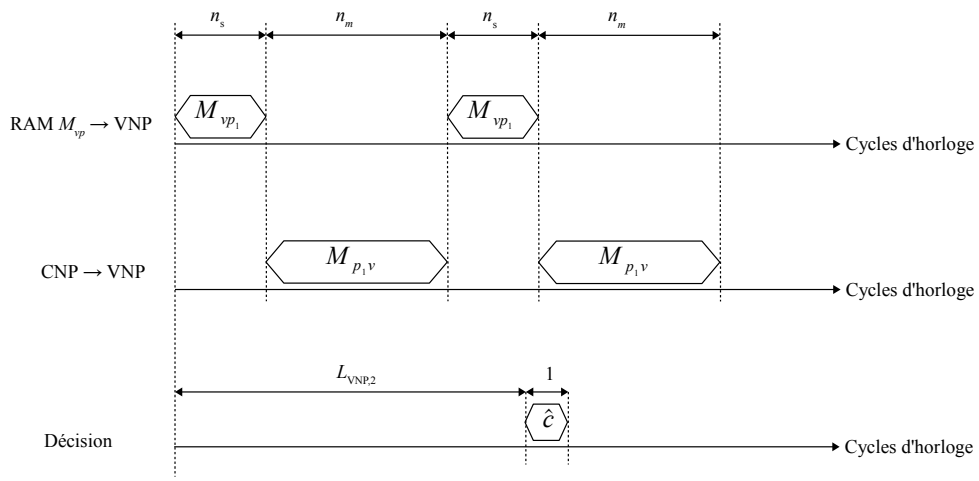


FIGURE 2.7 – Diagramme de temps du VNP en mode de décision

Après avoir décrit l'architecture du VNP pour chaque mode, nous devons déterminer les valeurs de n_s et n_α de façon à garder des performances quasi-identiques au cas optimal $n_s = n_\alpha = n_m$. Pour cela, nous avons lancé des simulations de Monte-Carlo sur un canal à bruit blanc gaussien additive (Additive White Gaussian Noise ou AWGN en anglais) en utilisant deux codes LDPC définis sur le corps de Galois $\mathbb{GF}(q = 64)$. Le décodage se fait par des messages de taille $n_m = 12$, un scalaire de correction *offset* $= 1$ et un nombre d'itérations $n_{iter} = 8$.

Dans un premier lieu, nous proposons de simuler les performances pour différentes valeurs $n_\alpha \leq n_m$ et une valeur fixe $n_s = n_m$. La figure 2.8 donne les courbe de taux d'erreur par paquet (TEP) d'un code de taille $N = 1152$ bits et de rendements $R = \frac{1}{2}$. Nous observons que pour $n_\alpha \leq 4$ les performances commencent à se dégrader considérablement. Cependant, pour $5 \leq n_\alpha \leq 12$ les performances sont très proches. Pour valider ces observations, nous avons lancé les mêmes simulations en modifiant le rendement du code à $R = \frac{2}{3}$. Les résultats sont illustrés sur la figure 2.9. Nous constatons une fois de plus que la valeur qui permet de réduire au maximum la taille des bancs mémoire RAM M_{vp} tout en gardant de bonnes performances est $n_\alpha = 5$.

L'étape suivante consiste à déterminer la valeur de n_s pour une valeur fixe $n_\alpha = 5$. La figure 2.10 donne les courbes de simulation en faisant varier n_s et en utilisant un code de rendement $R = \frac{1}{2}$. Ces courbes montrent que lorsque $n_s \leq 3$ les performances commencent à se dégrader. Pour valider cette constatation nous considérons les courbes de performance de la figure 2.11 obtenues avec un code de rendement $R = \frac{2}{3}$. Compte tenu de ces résultats, nous fixons la taille de la mémoire CAM à $n_s = 4$ (il est à noter que la valeur choisie dans [2] est $n_s = 3$).

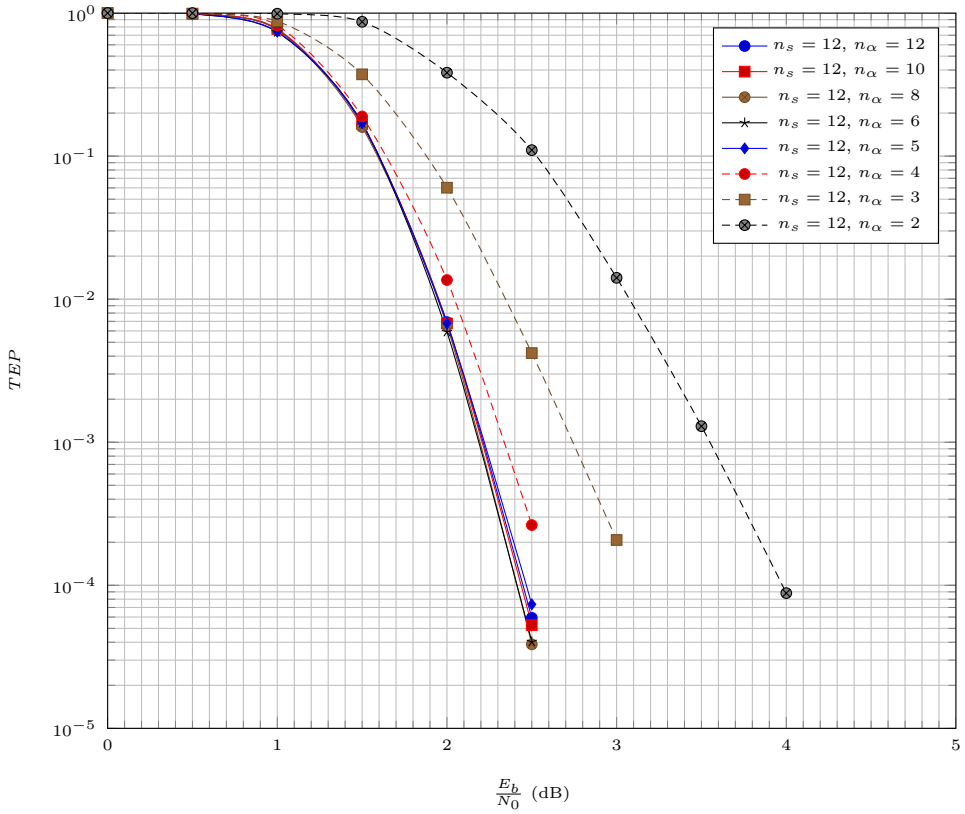


FIGURE 2.8 – Performance sur un canal AWGN de l'algorithme EMS pour $q = 64$, $N = 1152$ bits, $R = \frac{1}{2}$, $n_m = 12$, *offset* $= 1$, $n_{iter} = 8$, $n_s = n_m$ et $n_\alpha \leq n_m$

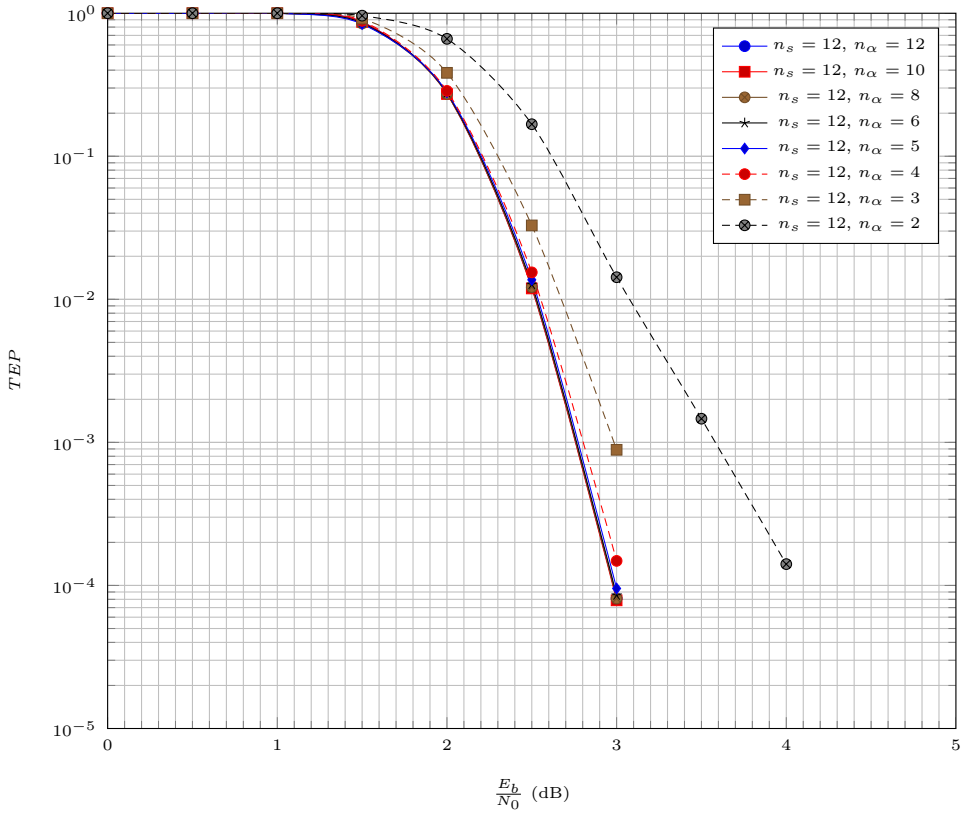


FIGURE 2.9 – Performance sur un canal AWGN de l'algorithme EMS pour $q = 64$, $N = 1152$ bits, $R = \frac{2}{3}$, $n_m = 12$, *offset* $= 1$, $n_{iter} = 8$, $n_s = n_m$ et $n_\alpha \leq n_m$

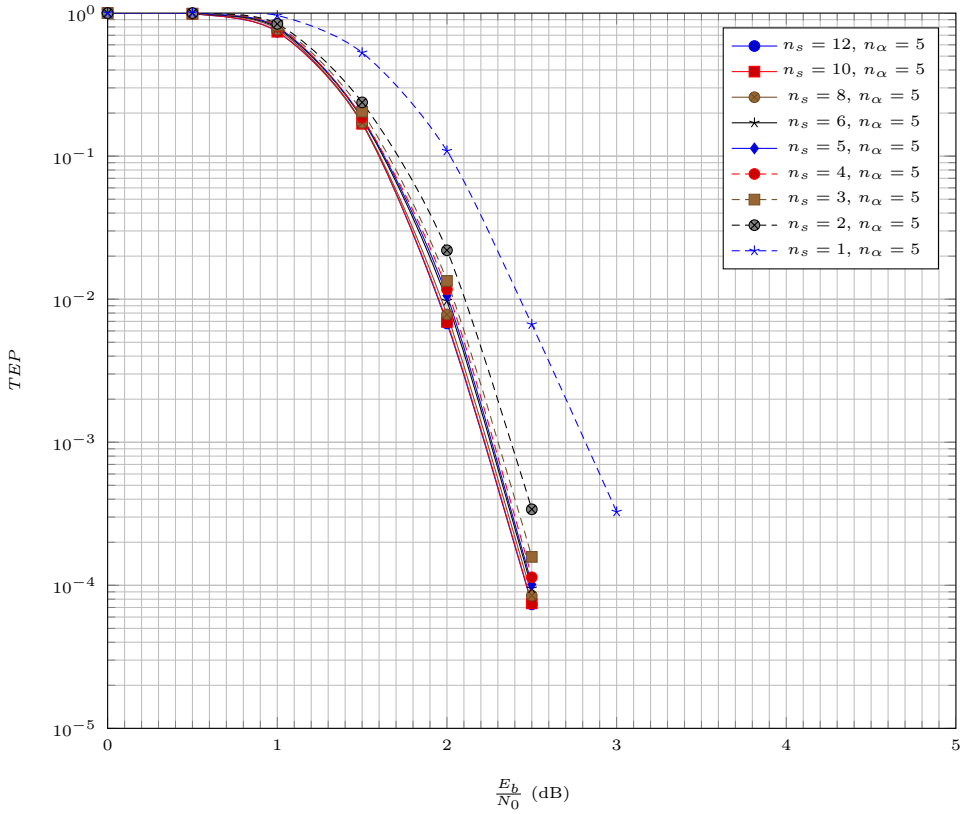


FIGURE 2.10 – Performance de l'algorithme EMS pour un canal AWGN, $q = 64$, $N = 1152$ bits, $R = \frac{1}{2}$, $n_m = 12$, *offset* $= 1$, $n_{iter} = 8$, $n_\alpha = 5$ et n_s variable

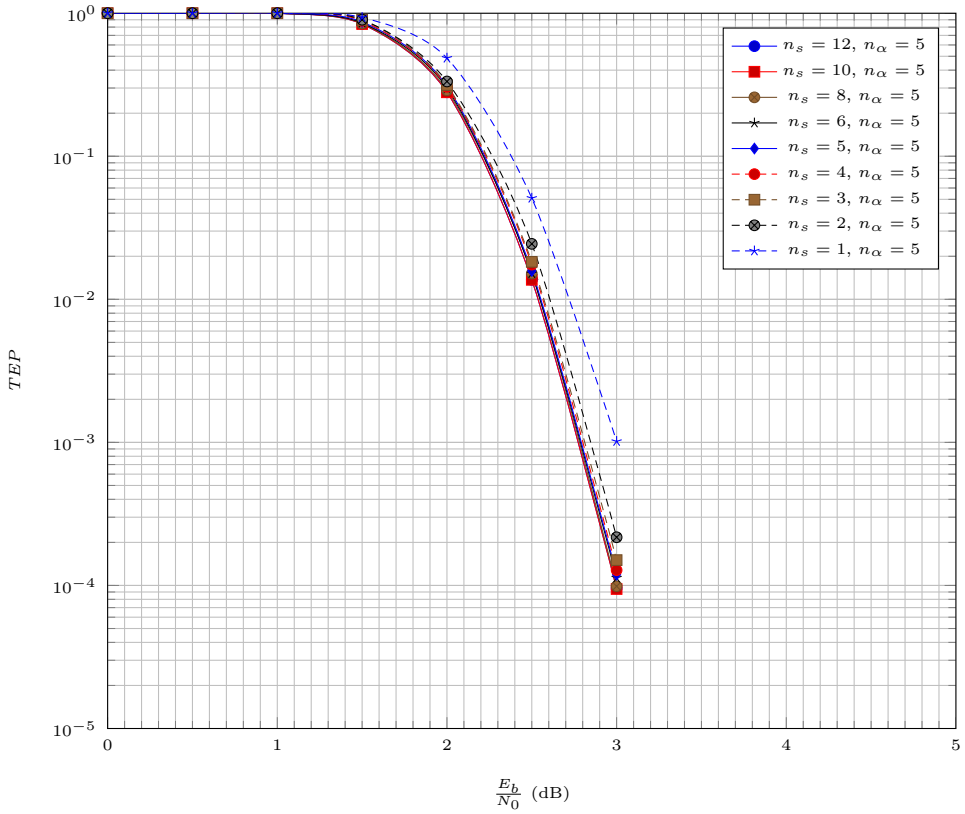


FIGURE 2.11 – Performance de l'algorithme EMS pour un canal AWGN, $q = 64$, $N = 1152$ bits, $R = \frac{2}{3}$, $n_m = 12$, *offset* $= 1$, $n_{iter} = 8$, $n_\alpha = 5$ et n_s variable

Pour l'intégrité scientifique, il faut noter que les auteurs de [3] proposent aussi de considérer l'ensemble des n_α éléments les plus fiables du message intrinsèque pour faire la mise à jours de VNs. Cependant, il existe plusieurs différences fondamentales avec notre architecture :

- ◇ Dans [3] tous les symboles du message M_{pv} qui n'appartiennent pas à l'ensemble $\{I.GF(0), I.GF(1), \cdots, I.GF(n_\alpha)\}$ sont mis à jour par $I.\text{LLR}(n_m - 1)$. Par conséquent, notre façon de mettre à jour le message M_{pv} est meilleure puisque nous considérons la totalité du message intrinsèque (le LLR intrinsèque de chaque symbole est obtenu par le module *eLLR*).
- ◇ La détermination des symboles de l'ensemble $M_{pv} \cap \{I.GF(0), I.GF(1), \cdots, I.GF(n_\alpha - 1)\}$ se fait dans [3] par une mémoire CAM contenant le message M_{pv} tandis qu'elle se fait dans notre cas par un simple registre *Flag* de q bits. De plus, notre VNP ne nécessite aucune mémoire pour sauvegarder le message M_{pv} puisqu'il est traité à la volée.
- ◇ Le VNP de [3] dispose d'une mémoire RAM pour sauvegarder les n_m éléments les plus fiables du message intrinsèque (bien que seulement $n_\alpha \ll n_m$ éléments sont utilisés). Notre VNP reçoit uniquement n_α symboles (sans LLR) du message intrinsèque sauvegardés dans une mémoire externe.
- ◇ Dans [3], le message reçu par le trieur peut contenir des symboles en double ce qui entraîne une dégradation des performances du décodeur. Notre VNP se sert d'un registre *Flag* pour détecter et supprimer les duplicatas.

La figure 2.12 donne les performances de décodage en utilisant notre algorithme et l'algorithme de [3].

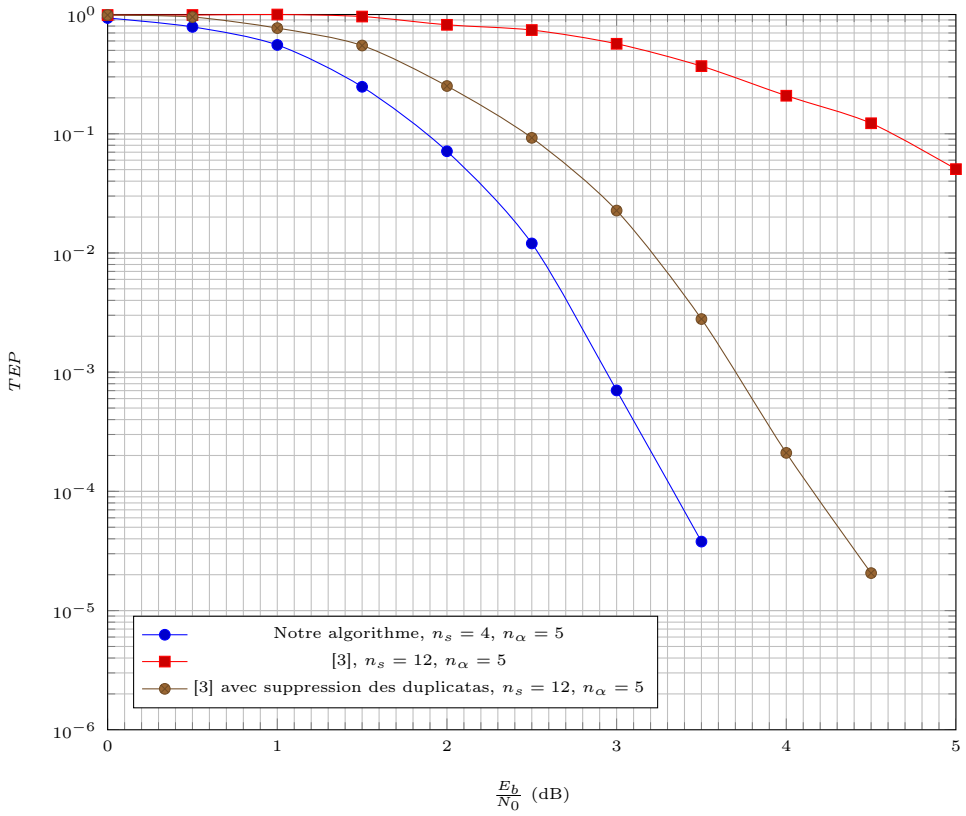


FIGURE 2.12 – Comparaison des performances sur un canal AWGN de notre algorithme de mise à jour des VNs avec celui de [3], $q = 64$, $N = 288$ bits, $R = \frac{1}{2}$, $n_m = 12$, $offset = 1$, $n_{iter} = 8$, $n_\alpha = 5$

Les simulations sont faites avec les paramètres $q = 64$, $R = \frac{1}{2}$, $N = 288$ bits, $n_m = 12$, $n_\alpha = 5$, $n_s = 4$ dans notre cas et $n_s = n_m$ dans le cas de [3]. Les résultats montrent un gain considérable en notre faveur. Nous pouvons améliorer les performances de [3] en supprimant les duplicatas dans les messages extrinsèques. Toutefois, notre algorithme donne toujours de meilleures performances (nous observons un écart de 0.8 dB à TEP = 10^{-4}).

Architecture du module *Sorter*

Pour améliorer encore le VNP, nous proposons de modifier l'architecture du trieur. En effet, ce module permet de trier trois types de listes :

- ◇ des listes contenant q éléments en mode de génération des LLRs intrinsèques.
- ◇ des listes contenant $n_m + n_\alpha$ éléments en mode de mise à jour des VNs.
- ◇ des listes contenant $n_m + n_s$ éléments en mode de décision.

L'architecture du module *Sorter* de la figure 1.15 n'est pas flexible et présente peu d'intérêt dans notre cas étant donné que sa complexité augmente avec la taille des listes à trier. En effet, puisqu'il faut considérer le pire cas, le trieur doit comporter $n_{\text{stage}} = \log_2(q)$ étages. De plus sa latence est égale à $L_{\text{sorter}} = n_e + n_{\text{stage}}$ cycles d'horloge, où n_e est la taille de la liste d'entrée. Par exemple, pour trier une liste contenant $q = 64$ éléments, nous obtenons $n_{\text{stage}} = 6$ et $L_{\text{sorter}} = 70$ cycles. Par conséquent, nous avons opté pour l'architecture présentée dans la figure 2.13. Cette architecture comporte toujours n_m étages indépendamment de la taille de la liste d'entrée puisque ce qui compte est la taille de la liste de sortie. De plus, la latence de cette architecture est inférieure à celle du décodeur DAVINCI et vaut $L_{\text{sorter}} = n_e$ cycles. Ainsi pour trier une liste contenant $q = 64$ éléments, nous obtenons $L_{\text{sorter}} = 64$ cycles.

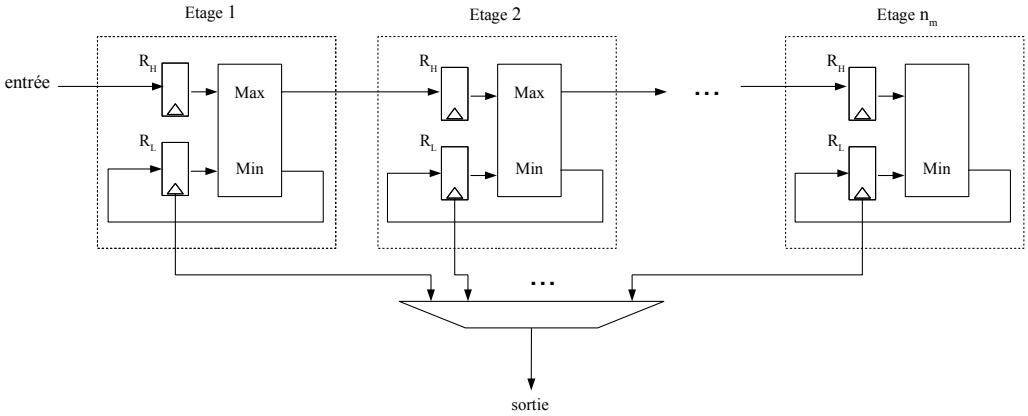


FIGURE 2.13 – Nouvelle architecture du module *Sorter*

Chaque étage du trieur comporte deux registres R_H et R_L et un comparateur. Le registre R_H contient le dernier élément fournit à l'entrée de l'étage et le registre R_L contient le petit élément de la comparaison. Un étage fournit le plus grand élément de comparaison à l'étage suivant. Par conséquent, les n_m registres R_L, lus de gauche à droite, forment une liste triée par ordre croissant. A l'arrivée du dernier élément de la liste d'entrée, le résultat de comparaison du premier étage à gauche est fourni à la sortie du trieur. Le cycle suivant, la sortie du trieur reçoit le résultat de comparaison de l'étage suivant, et ainsi de suite.

2.2.2 Architecture du processeur de nœud de parité

Dans un décodeur LDPC, Le CNP est le composant le plus coûteux en termes de surface d'implémentation. Comme nous avons déjà mentionné dans la section 1.3.3, une manière efficace d'implanter un CNP est de le diviser en un ensemble d'ECNs qui peuvent être connectés avec la récursion Forward-Backward de la figure 1.4 ou avec une récursion parallèle comme dans la figure 2.14. La parallélisation du CNP a été proposée dans [41]. Elle nécessite d'une part moins d'éléments de mémorisation pour sauvegarder les messages intermédiaires entre ECNs et permet d'autre part de réduire la latence du CNP par un cycle d'horloge. Par exemple, si le CNP est de degré $d_c = 6$, la génération d'un message sortant nécessite quatre ECNs en configuration Forward-Backward et trois ECNs en configuration parallèle. Néanmoins, la récursion Forward-Backward permet de concevoir une architecture générique pour tout rendement de code ; ce qui n'est pas possible avec la récursion parallèle.

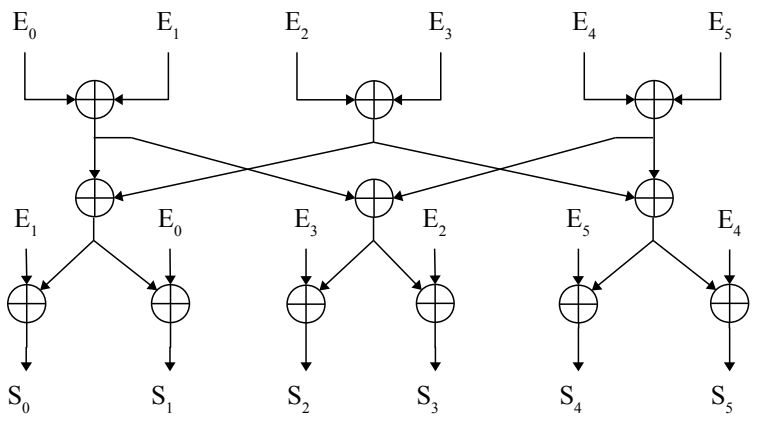


FIGURE 2.14 – Architecture parallèle d'un CN de degré $d_c = 6$

Un ECN effectue des opérations de comparaison pour extraire les n_m plus fiables symboles parmi les n_m^2 possibilités obtenues en additionnant ces deux messages entrants. L'algorithme L-Bubble Check permet de réduire la complexité de ce problème en décomposant la matrice T_M en deux zones :

- ◇ La première, dite zone d'exclusion notée par *Excl*, est exclue de la zone d'exploration de la matrice T_M, c'est-à-dire qu'il est déterminé au préalable qu'aucun des éléments de cette zone ne sera un candidat susceptible d'être intégré au trieur.
- ◇ La deuxième, dite zone de sélection notée par *Sel*, est le complémentaire de *Excl* et contient l'ensemble des candidats susceptibles d'être intégrés au trieur. La zone de sélection contient la premier ligne, la moitié de la deuxième ligne, la première colonne et la moitié de la deuxième colonne de la matrice T_M. Cependant, à cause de la possibilité de présence de duplicatas il est préférable de considérer la totalité de la deuxième ligne et la deuxième colonne.

Cependant, comme expliqué dans la section 2.1.2, l'implantation matérielle de cet algorithme souffre d'un long chemin critique due au mécanisme de *feedback* utilisé pour la mise à jour de la liste des candidats. Pour résoudre ce problème, nous proposons de modifier l'algorithme L-Bubble Check pour d'une part séparer le processus de génération des candidats du processus de comparaison, et d'autre part simplifier l'exploration de la matrice T_M. Pour cela, nous commençons par décomposer la zone de sélection *Sel* en quatre

sous-ensembles :

- ◇ $Sel_0 = T_M(0, i)$, $i = 0, 1, \cdots, n_m - 1$.
- ◇ $Sel_1 = T_M(i, 0)$, $i = 1, 2, \cdots, n_m - 1$.
- ◇ $Sel_2 = T_M(1, i)$, $i = 1, 2, \cdots, n_m - 1$
- ◇ $Sel_3 = T_M(i, 1)$, $i = 2, 3, \cdots, n_m - 1$

Ces quatre sous-ensembles sont illustrés dans la figure 2.15.

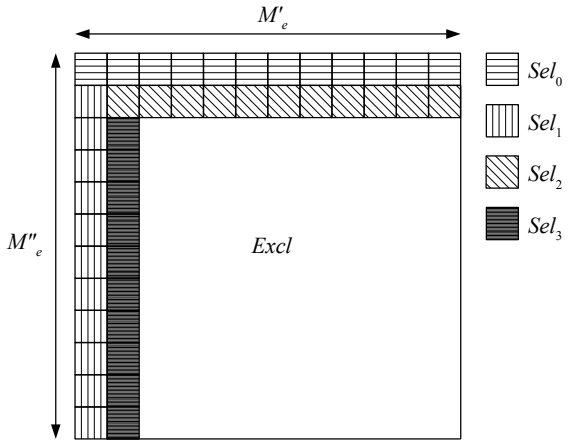


FIGURE 2.15 – Les sous-ensembles de la zone de sélection

Puisque les messages $M_e^{'}$ et $M_e^{''}$ sont déjà triés par ordre croissant, nous obtenons alors la propriété suivante :

$$Sel_k(i) \leq Sel_k(j); 0 \leq k \leq 3 \;\; et \;\; 0 \leq i \leq j < Cardinal\{Sel_k\}$$

Autrement dit, les sous-ensembles sont déjà triés par ordre croissant et peuvent être construits au fur et à mesure de l'arrivée des éléments de $M_e^{'}$ et $M_e^{''}$. Par conséquent, pour obtenir le message sortant M_s il suffit de sélectionner un candidat de chaque sous-ensemble. Dans ce cas, les quatre parcours de sélection sont décrits par la figure 2.16.

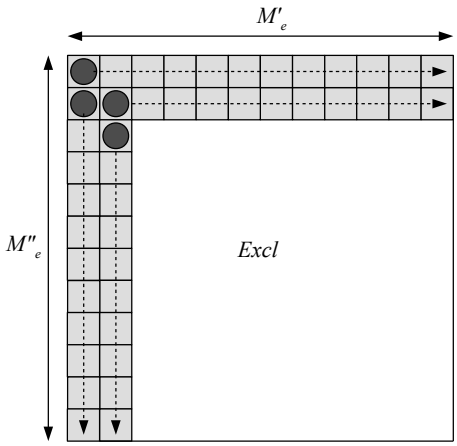


FIGURE 2.16 – Les parcours de sélection

En se basant sur ces observations, nous proposons l'algorithme S-Bubble Check détaillé ci-dessous.

Algorithme 2.1 S-Bubble Check

Entrées : $M_e^{'}$ et $M_e^{''}$
Sorties : M_s

Calculer Sel_0 :
pour $i = 0$ ***à*** $n_m - 1$ **faire**
  si $M_e^{'}(i)$ *valide* **alors**
    $Sel_0.LLR(i) := M_e^{'}.LLR(i) + M_e^{''}.LLR(0)$;
    $Sel_0.GF(i) := M_e^{'}.GF(i)$ **xor** $M_e^{''}.GF(0)$;
  sinon
    $Sel_0(i) := \emptyset$; `/* donnée non valide */`
  fin
fin

Calculer Sel_1 :
pour $i = 1$ ***à*** $n_m - 1$ **faire**
  si $M_e^{''}(i)$ *valide* **alors**
    $Sel_1.LLR(i-1) := M_e^{''}.LLR(i) + M_e^{'}.LLR(0)$;
    $Sel_1.GF(i) := M_e^{''}.GF(i)$ **xor** $M_e^{'}.GF(0)$;
  sinon
    $Sel_1(i-1) := \emptyset$;
  fin
fin

Calculer Sel_2 :
pour $i = 1$ ***à*** $n_m - 1$ **faire**
  si $M_e^{'}(i)$ *valide et* $M_e^{''}(1)$ *valide* **alors**
    $Sel_2.LLR(i-1) := M_e^{'}.LLR(i) + M_e^{''}.LLR(1)$;
    $Sel_2.GF(i-1) := M_e^{'}.GF(i)$ **xor** $M_e^{''}.GF(1)$;
  sinon
    $Sel_2(i-1) := \emptyset$;
  fin
fin

Calculer Sel_3 :
pour $i = 2$ ***à*** $n_m - 1$ **faire**
  si $M_e^{''}(i)$ *valide et* $M_e^{'}(1)$ *valide* **alors**
    $Sel_3.LLR(i-2) := M_e^{''}.LLR(i) + M_e^{'}.LLR(1)$;
    $Sel_3.GF(i-2) := M_e^{''}.GF(i)$ **xor** $M_e^{'}.GF(1)$;
  sinon
    $Sel_3(i-2) := \emptyset$;
  fin
fin

Initialisation des indices :
$ind_0 := 0$;
$ind_1 := 0$;
$ind_2 := 0$;
$ind_3 := 0$;

suite de l'algorithme 2.1 S-Bubble Check

pour $i = 0$ ***à*** $n_m - 1$ **faire**
  Mise à jour des candidats :
  $cand_0 := Sel_0(ind_0)$; $cand_1 := Sel_1(ind_1)$; $cand_2 := Sel_2(ind_2)$; $cand_3 := Sel_3(ind_3)$;
  Comparaison :
  si $\text{cand}_0 = \emptyset$ *ou* $\text{cand}_1 = \emptyset$ *ou* $\text{cand}_2 = \emptyset$ *ou* $\text{cand}_3 = \emptyset$ **alors**
    $M_s(i) := \emptyset$;
    pour $j = 0$ ***à*** 3 **faire**
      si $\text{cand}_j = \emptyset$ **alors**
        $ind_j := ind_j + 1$;
      fin
    fin
  sinon si $cand_0.LLR = \min\{cand_0.LLR, cand_1.LLR, cand_2.LLR, cand_3.LLR\}$ **alors**
    $M_s(i) := cand_0$;
    $ind_0 := ind_0 + 1$;
  sinon si $cand_1.LLR = \min\{cand_0.LLR, cand_1.LLR, cand_2.LLR, cand_3.LLR\}$ **alors**
    $M_s(i) := cand_1$;
    $ind_1 := ind_1 + 1$;
  sinon si $cand_2.LLR = \min\{cand_0.LLR, cand_1.LLR, cand_2.LLR, cand_3.LLR\}$ **alors**
    $M_s(i) := cand_2$;
    $ind_2 := ind_2 + 1$;
  sinon
    $M_s(i) := cand_3$;
    $ind_3 := ind_3 + 1$;
  fin
fin

Détection des symboles en double :
$Indicateur(i)_{i=0,1,\cdots,q-1} := 0$;
pour $i = 0$ ***à*** $n_m - 1$ **faire**
  si $\text{Indicateur}[M_s(i).GF] = 1$ **alors**
    $M_s(i) := \emptyset$; `/* donnée non valide */`
  fin
  $Indicateur[M_s(i).GF] := 1$; `/* marque la présence de` $M_s(i).GF$ `dans` M_s `*/`
fin

Dans notre algorithme, les duplicatas (en entrée et en sortie) sont considérés comme des données non valides et sont marquées par $\emptyset$. La comparaison ne doit pas se faire avec des candidats non valides pour éviter le risque de générer un message non trié. Dans un tel cas, le résultat de comparaison est forcé à $\emptyset$ et les candidats non valides doivent être remplacés.

Nous proposons d'implanter l'algorithme S-Bubble Check avec l'architecture de la figure 2.17. Dans cette architecture, les parcours de sélection sont empilés chacun dans une mémoire FIFO. Dans un premier cycle, le comparateur commence par sortir $Sel_0(0)$ (étant donné que $Sel_0(0)$ est nécessairement valide et son LLR vaut toujours zéro) et $cand_0$ prend la valeur $Sel_0(1)$. Dans un deuxième cycle, le comparateur choisit entre $cand_0$ et

$cand_1$ et met à jour le candidat sélectionné en incrémentant l'adresse de la mémoire FIFO correspondante. Dans les cycles suivants, la comparaison se fait en considérant les quatre candidats. Le module *Flag* est un registre de q bits qui joue le rôle d'un indicateur servant à détecter la présence de duplicatas dans le message sortant.

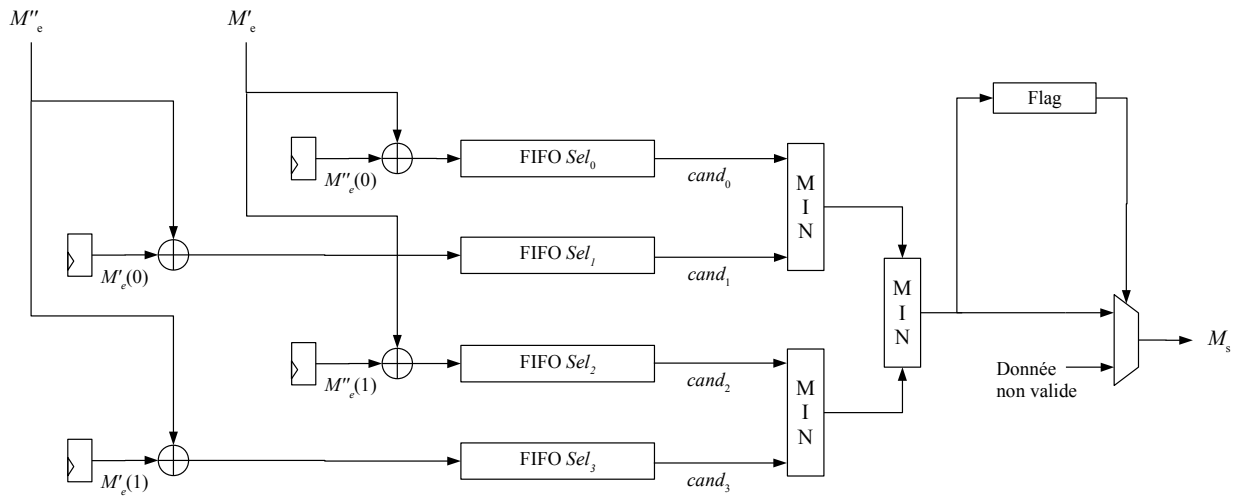


FIGURE 2.17 – Architecture S-Bubble Check

Nous avons implanté notre CNP en se basant sur l'architecture parallèle de la figure 2.14. Les symboles des messages entrants du CNP doivent être multipliés (permutés) par les éléments non nuls de la matrice de parité. De même, les symboles des messages sortants du CNP doivent être divisés par les mêmes éléments non nuls. Pour cela, nous utilisons l'architecture du multiplicateur présentée dans [2]. La figure 2.18 montre le diagramme de temps de notre CNP.

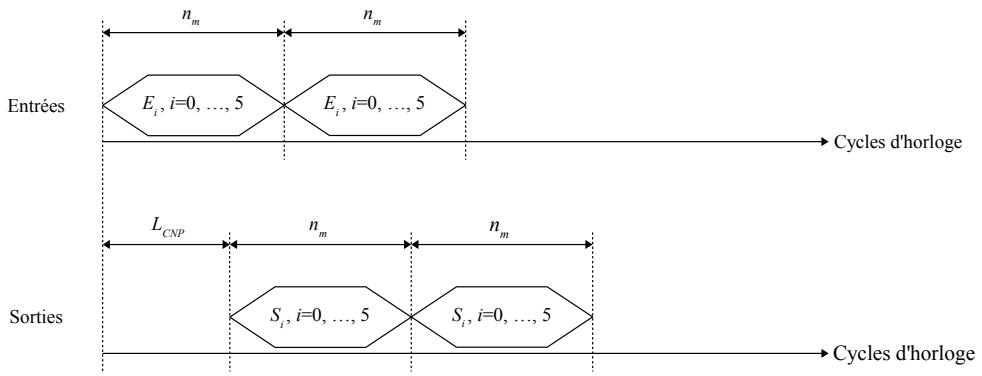


FIGURE 2.18 – Diagramme de temps du CNP

La latence de notre CNP vaut $L_{CNP} = 8$ cycles d'horloge tandis que la latence du CNP développé dans le cadre du projet DAVINCI vaut $L_{CNP} = 11$ cycles d'horloge. De plus, notre CNP peut être complétement *pipeliné* contrairement à celui du projet DAVINCI où deux message d'entrée consécutifs doivent être séparés par un intervalle de 6 cycles d'horloges.

2.2.3 Résultats de synthèse

Nous avons consacré une partie de notre thèse à l'implantation d'un prototype de notre décodeur sur la carte ZedBoard (Zynq Evaluation and Development Board) [42]. Cette carte intègre sur une même puce un microprocesseur à deux cœurs ARM Cortex-A9 et un FPGA Zynq XC7Z020-CLG484-1.

La figure 2.19 illustre le diagramme de temps du prototype développé. Les quatre modes de fonctionnement du décodeur sont déjà décrits dans la section 2.2.

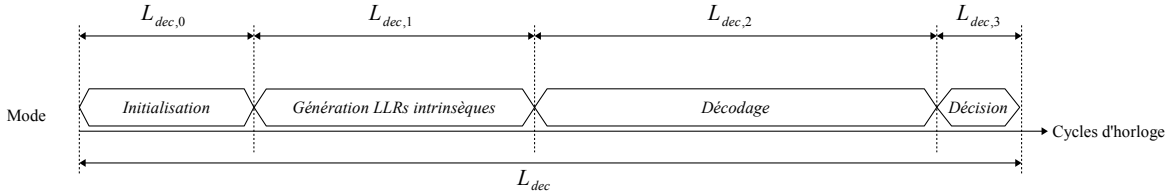


FIGURE 2.19 – Diagramme de temps du décodeur

La figure 2.20 donne le diagramme de temps du prototype en mode de décodage.

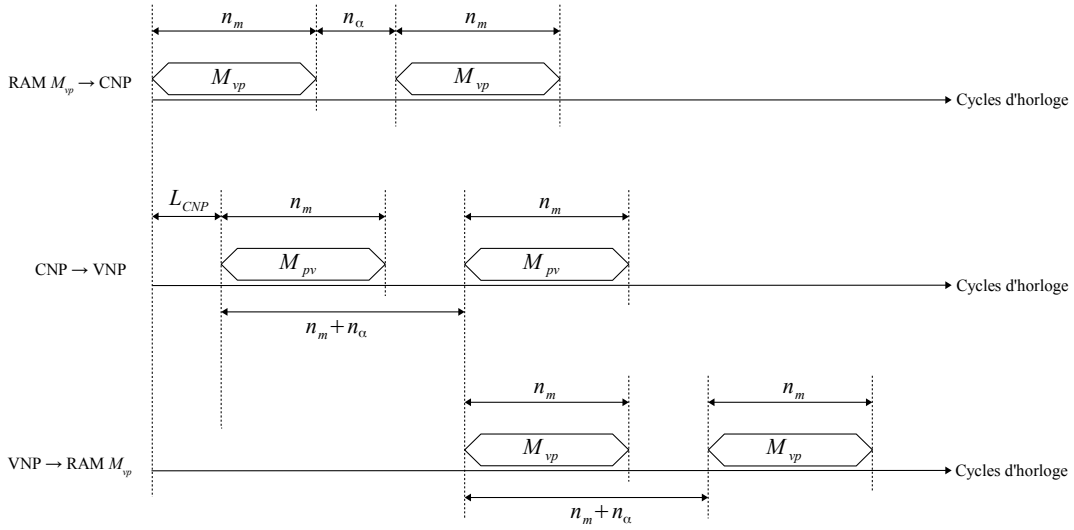


FIGURE 2.20 – Diagramme de temps du prototype en mode de décodage

Le *pipelining* des opérations est contraint par la longueur des messages traités par les VNPs. En effet, lorsque le décodeur est en mode de décodage, les VNPs peuvent fonctionner selon deux modes :

- ◇ Mode de mise à jour : chaque VNP traite d'abord un message M_{pv} pendant n_m cycles d'horloge ensuite un message I.GF pendant n_α cycles d'horloge. Par conséquent, les VNPs ne peuvent recevoir des nouveaux messages M_{pv} que tous les $n_m + n_\alpha$ cycles.
- ◇ Mode de décision : chaque VNP traite d'abord un message M_{pv} pendant n_m cycles d'horloge ensuite un message M_{vp} pendant n_s cycles d'horloge. Par conséquent, les VNPs ne peuvent recevoir des nouveaux messages M_{pv} que tous les $n_m + n_s$ cycles.

Pour simplifier l'implantation du décodeur, nous avons décidé de séparer les messages M_{pv} par un intervalle régulier indépendamment du mode de fonctionnement des VNPs. Autrement dit, chaque VNP ne reçoit un nouveau message M_{pv} que tous les $n_m + max\{n_\alpha, n_s\}$ cycles d'horloge (soit $n_m + n_\alpha$ dans notre cas).

Nous avons développé le prototype en se basant sur une matrice de parité d'un code défini sur le corps de Galois $\mathbb{GF}(q = 64)$, de taille $N = 192$ symboles et de rendement $R = \frac{2}{3}$. Le degré du CNP est $d_c = 6$ et celui des VNPs est $d_v = 2$. Les symboles du corps de Galois sont codés sur $m = 6$ bits. Chaque observation du canal est quantifiée sur $n_y = 6$ bits. Les LLRs sont codés sur $n_b = 6$ bits et les paramètres de l'algorithme EMS sont $n_m = 12$, $offset = 1$, $n_s = 4$, $n_\alpha = 5$ et $n_{iter} = 8$. Avec ces valeurs, nous obtenons :

$L_{dec,0} = N \cdot n_y = 1152$ cycles

$L_{dec,1} = (\frac{N}{d_c} - 1) \cdot q + L_{vnp,0} + n_m = 2062$ cycles

$L_{dec,2} = L_{CNP} + M \cdot n_{iter} \cdot (n_m + n_\alpha) + n_m = 8724$ cycles

$L_{dec,3} = N = 192$ cycles

Dans ce qui suit, nous commençons par donner les résultats de synthèse des principaux composants de notre décodeur sur FPGA Zynq XC7Z020-CLG484-1. Ensuite, nous donnons les résultats de synthèse du système complet. Le prototype du projet DAVINCI a été développé pour fonctionner sur les FPGAs de la famille VIRTEX 5 et n'est pas synthétisable sur les FPGAs de la famille Zynq. Par conséquent, pour pouvoir faire des comparaisons équilibrées, nous donnons aussi les résultats de synthèse de notre décodeur sur FPGA VIRTEX XC5VLX330T-2FF1738.

Les tableaux 2.1, 2.2 et 2.3 donnent les résultats de synthèse du module *Sorter* pour des listes d'entrée contenant respectivement 16, 17 et 64 éléments.

TABLE 2.1 – Résultat de synthèse du trieur pour une liste d'entrée contenant 16 éléments

	Notre trieur		Trieur de [2]
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de slices occupés	220	139	229
Fréquence maximale en MHz	298.597	250.740	234.522

TABLE 2.2 – Résultat de synthèse du trieur pour une liste d'entrée contenant 17 éléments

	Notre trieur		Trieur de [2]
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de slices occupés	227	146	446
Fréquence maximale en MHz	298.597	250.740	234.241

TABLE 2.3 – Résultat de synthèse du trieur pour une liste d'entrée contenant 64 éléments

	Notre trieur		Trieur de [2]
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de slices occupés	232	181	895
Fréquence maximale en MHz	298.597	250.740	205.137

Nous observons que la complexité de notre architecture varie peu en fonction de la longueur des listes à trier alors que celle de l'architecture DAVINCI augmente considérablement. Cette augmentation provient de la taille des mémoires FIFO qui augmente avec la position de l'étage et vaut 2^{pos}, $pos = 0, 1, \cdots n_{\text{stage}} - 1$ étant la position de l'étage.

Le tableau 2.4 donne les résultats de synthèse de notre VNP. Les résultats de synthèse du VNP de l'architecture du projet DAVINCI sont obtenus pour $n_s = 3$ et $n_\alpha = n_m$. Nous

observons que notre architecture divise par deux la surface tout en doublant la fréquence maximale.

Table 2.4 – Résultat de synthèse du VNP

	Notre VNP		VNP de [2]
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de slices occupés	367	423	776
Fréquence maximale en MHz	202.634	240.955	94.705

Le tableau 2.5 donne les résultats de synthèse de notre ECN. Nous déduisons que l'algorithme S-Bubble Check et l'architecture associée permettent de diviser approximativement la surface par deux tout en doublant la fréquence maximale.

	S-Bubble Check		L-Bubble Check
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de Slices occupés	451	179	370
Fréquence maximale en MHz	150.291	209.842	90.075

Table 2.5 – Résultats de synthèse de l'algorithme S-Bubble Check

Le tableau 2.6 donne les résultats de synthèse de notre CNP. Les résultats montrent de nouveau que l'algorithme S-Bubble Check permet de réduire par la moitié la surface tout en doublant la fréquence maximale.

	Notre CNP		CNP de [2]
Récursion	Parallèle		Forward-Backward
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de Slices occupés	3631	2547	4287
Fréquence maximale en MHz	160.183	209.842	91.041

Table 2.6 – Résultats de synthèse du CNP

Avant de donner les résultats de synthèse du décodeur complet, il est convenable de trouver un paramètre qui permet de mesurer l'efficacité de notre architecture afin de se comparer avec les décodeurs de l'état de l'art. Intuitivement, une architecture est plus efficace si elle permet de traiter plus de données par unité de temps et unité de surface de fabrication. Par conséquent, nous retenons la définition suivante de l'efficacité matérielle :

$$Eff = \frac{Quantité\ de\ données}{Surface \times Temps\ de\ traitement} \tag{2.6}$$

Étant donné que nos résultats de synthèse sont obtenus sur un FPGA Zynq XC7Z020-CLG484-1, la surface du décodeur peut être mesurée en nombre de blocs logiques occupés (*Slice* en anglais). Un Slice Zynq contient 4 LUTs de 6 entrées et 8 bascules D Flip-Flop. Les LUTs servent à implanter les équations logiques et peuvent être utilisés pour implanter des mémoires distribuées. Les bascules permettent de mémoriser les états ou synchroniser les signaux. D'autre part, le temps de traitement peut être mesuré par la durée T_{clk} d'un cycle d'horloge multipliée par la latence du décodeur L_{dec} (la latence étant le nombre de cycles d'horloge nécessaires pour décoder un mot de code). Enfin, la quantité de données traitées correspond à la taille des mots de code. La définition de l'efficacité devient :

$$
\begin{aligned}
Eff &= \frac{N(\text{en bits})}{\textit{Nombre de Slices occupés} \times L_{\text{dec}}} \times f_{\max} \\
&= \frac{D}{\textit{Nombre de Slices occupés}}
\end{aligned} \tag{2.7}
$$

avec $f_{\max} = \frac{1}{T_{\text{clk}}}$ est la fréquence maximale et D (en bits par seconde) est le débit d'entrée du décodeur.

Il est à noter que nous n'avons pas pris en considération le paramètre efficacité énergétique de l'architecture bien que cela serait d'une grande importance pour plusieurs applications pratiques notamment la téléphonie mobile ou la communication par satellite.

Dans la limite de notre connaissance, en plus du décodeur du projet DAVINCI, le meilleur décodeur EMS de l'état de l'art est celui présenté dans [3]. De plus, il a montré dans [2] que les deux décodeurs possèdent le même coût d'implantation. Il en découle, selon les résultats du tableau 2.7, que nous avons amélioré l'efficacité du décodeur EMS par un facteur de 7. Il faut noter que pour faciliter la comparaison le compilateur a été forcé de synthétiser les bancs mémoires sous forme de LUTs (mémoire distribuée).

	Notre décodeur		Décodeur de [2]
FPGA	Zynq	VIRTEX-5	VIRTEX-5
Nombre de Slices occupés	11195	12592	26597
f_{max} (MHz)	116.150	89.519	38.380
D (Mbps)	11.063	8.527	2.479
Eff	761.681	677.175	93.206

TABLE 2.7 – Résultats de synthèse du décodeur

Toutefois, nous pouvons améliorer le débit d'entrée de notre décodeur en commençant la réception des observations du canal et la génération des LLRs intrinsèques d'un nouveau mot de code pendant le décodage du mot de code actuel. Pour cela, nous considérons un système de mémorisation en *ping pong* où chaque banc mémoire est dupliqué (un banc mémoire sert au décodage du mot de code actuel et l'autre sert à la préparation du nouveau mot de code et inversement). De plus, la génération des LLRs intrinsèques ne se fait plus par les VNPs et peut être implantée séparément en utilisant l'architecture systolique de la figure 1.13. Par conséquent nous obtenons le nouveau diagramme de temps illustré dans la figure 2.21.

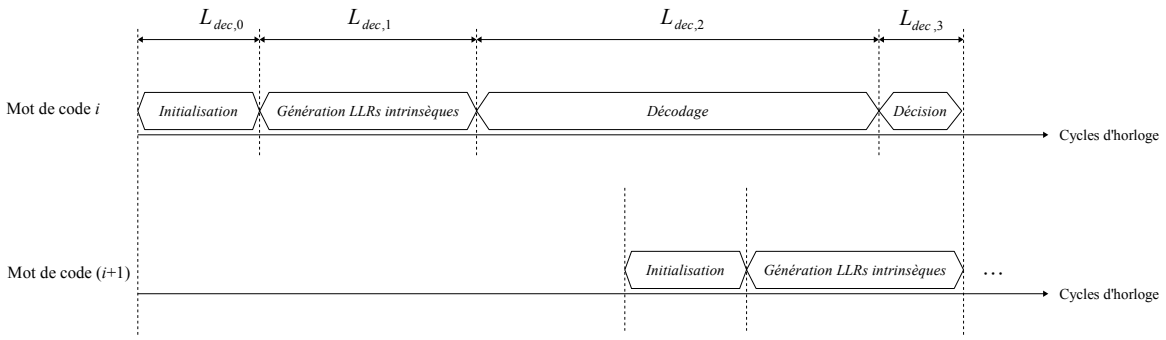


FIGURE 2.21 – Diagramme de temps du décodeur *pipeliné*

Avec cette configuration, le débit d'entrée devient :

$$D = \frac{N}{L_{\text{dec},2} + L_{\text{dec},3}} \cdot f_{\max} \tag{2.8}$$

Par conséquent, nous estimons le débit maximal sur FPGA VIRTEX 5 à $D = 9.561$ Mbps et sur FPGA Zynq à $D = 12.405$ Mbps.

2.2.4 Comparaison avec les architectures de l'état de l'art

Depuis les travaux pionniers de Davey et Mackay [13], le nombre annuel de publications scientifiques proposant des implantations matérielles de décodeurs LDPC non binaires ne cessent d'augmenter rapidement. Toutefois, il est difficile de comparer notre architecture avec tous ces travaux à cause de la différence des paramètres utilisés (ordre du corps de Galois, taille des codes, rendement des codes...) et des technologies ciblées. Par conséquent, les comparaisons faites dans cette section sont approximatives. De plus, n'ayant pas accès à des outils de conception de circuits intégrés à application spécifique (Application-Specific Integrated Circuit ou ASIC en anglais), nous nous comparons uniquement aux implantations ciblant des composants FPGA. Il faut noter tout de même que la plupart des travaux ciblent les technologies ASIC. Nous citons à titre d'exemple les architectures proposées dans [29, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55].

Le tableau 2.8 donne une comparaison des résultats de synthèses présentés dans [56, 57, 58] et de notre approche. Le tableau indique à chaque fois la technologie qui a servi pour l'implantation. Il faut noter que l'ordre q du corps de Galois ainsi que l'algorithme de décodage diffèrent d'une implantation à l'autre.

	Notre décodeur	[56]	[57]	[58]
FPGA	Zynq	VIRTEX-2P	VIRTEX-4	VIRTEX-2P
q	64	8	4	32
N (bits)	1152	2160	1944	3720
n_b (bits)	6	8	-	5
Ordonnancement	série	série	3-parallèle	31-parallèle
Algorithme	EMS	*log*-BP-FFT	EMS	Min-Max
n_iter	8	10-20	20	15
Nombre de Slices	11195	4660	46190	47341
f_{max} (MHz)	116.150	99.73	131.411	106.2
D (Mbits/s)	11.063	1.09	50	9.3
Eff (bits/(slice $\cdot$ s))	761.681	223.905	1082.485	196.447

TABLE 2.8 – Comparaison des résultats de synthèse présentés dans l'état de l'art et de notre approche

Nous déduisons d'abord que notre architecture est beaucoup plus efficace que l'architecture du décodeur optimal présentée dans [56] bien que celle-ci se limite au corps de Galois d'ordre 8. De plus, étant donné que la complexité de l'algorithme BP est dominée par $O(q^2)$, migrer vers le corps de Galois d'ordre 64 revient à multiplier la surface du décodeur optimal par un facteur de 64. S'ajoute à cela le fait que l'écart de performance entre les algorithmes BP et EMS est seulement d'environ 0.5 dB pour un code de taille

$N = 192$ symboles. Il en découle que le gain en performance réalisé par le décodeur optimal ne compense pas sa complexité exorbitante.

Pour pouvoir se comparer avec le décodeur EMS présenté dans [57], il faut multiplier sa surface par un facteur de 5 puisque la complexité de l'algorithme EMS est théoriquement dominée par $O(n_m \log n_m)$ [26]. Par conséquent, notre décodeur est nettement plus efficace même en supposant que le débit d'entrée dans [57] reste constant malgré l'augmentation de la cardinalité du corps de Galois.

A ce jour, la meilleure architecture d'un décodeur Min-Max est celle proposée dans [50]. En partant d'une implantation ASIC, il a été montré qu'elle permet de multiplier par un facteur de 6 l'efficacité du décodeur présenté dans [45] et par conséquent nous estimons qu'elle permet de multiplier par un facteur de 12 l'efficacité du décodeur présenté dans [58]. Toutefois, cette amélioration a été effectuée au détriment d'une dégradation des performances par environ 0.1 dB à cause d'une simplification des calculs dans le nœud de parité. De plus, nous estimons que les performances de notre décodeur sont largement meilleures étant donné que ces décodeurs sont basés sur un corps de Galois d'ordre $q = 32$.

2.3 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle architecture d'un décodeur EMS en se basant sur des observations faites sur le décodeur qui a été proposé par notre laboratoire de recherche dans le cadre du projet européen DAVINCI. Nous avons simplifié le processeur de nœud de variable en utilisant uniquement $n_\alpha \ll n_m$ éléments de l'information intrinsèque pour effectuer la mise à jour des messages extrinsèques et en proposant une nouvelle architecture pour le module de tri. Pour réduire la complexité du processeur de nœud de parité, nous avons proposé d'une part une nouvelle variante de l'algorithme L-Bubble Check en enlevant le parcours sous forme de « L » dans la matrice T_M et d'autre part une architecture d'un nœud de parité élémentaire dans laquelle la création des candidats et la comparaison se traitent séparément. Les résultats de synthèse d'un prototype du décodeur défini sur un corps de Galois d'ordre $q = 64$ montrent que nous avons amélioré significativement l'efficacité en terme de débit d'entrée sur la surface d'implantation. Cependant, pour pouvoir se comparer avec les architectures de l'état de l'art, il est nécessaire de généraliser notre prototype sur d'autres corps de Galois (en particulier $\mathbb{GF}(32)$). De plus, un prototype ASIC permettra de déterminer avec exactitude l'efficacité de notre décodeur puisque les résultats que nous avons donné dans le tableau 2.8 dépendent de l'architecture du composant FPGA considéré (par exemple, la fréquence maximale sur Zynq est supérieure à celle sur VIRTEX-5). Enfin, nous étions contraints de développer une architecture en série car le décodeur devait tenir dans notre carte FPGA. Toutefois, ce serait intéressant de développer une architecture parallèle de notre décodeur.

Chapitre 3

Codage LDPC non binaire et forme d'onde CCSK

Sommaire

Récemment, des études telles que [59, 60] ont mis l'accent sur la bonne adéquation qui existe entre les codes LDPC non binaires et les modulations d'ordre élevé. Cette propriété permet de réaliser des gains importants de performances qui justifieraient l'utilisation des codes LDPC non binaires dans des systèmes de communication à forte contrainte énergétique au niveau de l'émetteur et où nous pouvons fermer les yeux sur la grande complexité de décodage au niveau du récepteur. Dans ce même état d'esprit, nous avons consacré une partie de notre thèse à l'étude des performances de l'association des codes LDPC non binaires avec des modulations de même ordre, en particulier la modulation par décalage cyclique de code (Cyclic Code-Shift Keying ou CCSK en anglais). Toutefois, nous commençons par rappeler quelques notions mathématiques indispensable à la compréhension de la suite de ce chapitre.

3.1 Rappel mathématique

Définition 3.1. Soit a un entier. Si $a \geq 0$, l'opération de division de a par q peut toujours s'écrire d'une manière unique sous la forme euclidienne :

$$a = k \cdot q + b \tag{3.1}$$

avec k et b deux entiers tels que $k \geq 0$ et $0 \leq b < q$. b est appelé le reste de la division euclidienne de a par q. Par abus de langage, nous disons que b est égale à a *modulo* q et nous utilisons la notation $b = (a)_q$. Si $a < 0$, nous définissons l'opération $(a)_q$ comme suit :

$$(a)_q = (a + k \cdot q)_q \tag{3.2}$$

avec k un entier positif tel que $0 \leq a + k \cdot q < q$.

Définition 3.2. Considérons deux vecteurs de nombres complexes $x = [x_i]_{0 \leq i < q}$ et $y = [x_i]_{0 \leq i < q}$. La corrélation de x et y est définie par :

$$c = \sum_{j=0}^{q-1} x^*[j]y[j] \tag{3.3}$$

où x^* est le conjugué de x. La corrélation donne une mesure du taux de ressemblance de x et y. La corrélation croisée (ou inter-corrélation) de x et y, notée par $\star$, est donnée par :

$$x \star y[i] = \sum_{j=0}^{q-1} x^*[j]y[(j+i)_q] \tag{3.4}$$

La corrélation croisée peut se calculer par transformée de Fourier rapide (Fast Fourier Transform ou FFT) et transformée de Fourier rapide inverse (Inverse Fast Fourier Transform ou IFFT) comme suit :

$$x \star y[i] = \mathrm{IFFT}(\mathrm{FFT}^*(x) \times \mathrm{FFT}(y))[i] \tag{3.5}$$

L' auto-corrélation de x est obtenue par $x \star x$

Définition 3.3. Considérons deux vecteurs de nombres complexes $x = [x_i]_{0 \leq i < q}$ et $y = [x_i]_{0 \leq i < q}$. La convolution circulaire de x et y est définie par :

$$x \odot y[i] = \sum_{j=0}^{q-1} x[j]y[(i-j)_q] \tag{3.6}$$

La transformée de Fourier discrète de la convolution circulaire est le produit des transformées de Fourier discrètes de x et y. Par conséquent, il est plus rapide de calculer la convolution circulaire par FFT et IFFT :

$$x \odot y[i] = \mathrm{IFFT}(\mathrm{FFT}(x) \times \mathrm{FFT}(y))[i] \tag{3.7}$$

3.2 Modulation des codes LDPC

Pour améliorer l'efficacité spectrale, les systèmes de communication utilisent des modulations numériques d'ordre élevé (M-QAM, M-ASK, M-PSK ...). Le mot de code est par conséquent groupé en symboles de $\log_2 M$ bits et modulé par la suite au niveau symbole contrairement à la modulation BPSK où les bits du mot de code sont directement modulés. Cependant pour respecter le seuil maximal de la puissance de transmission il faut rapprocher les points de constellations au prix d'une dégradation des performances. Les modulations codées en treillis [61, 62, 63] (Treillis Coded Modulations TCM) permettent de résoudre ce problème en considérant la modulation et le codage simultanément. En effet, dans les systèmes classiques le code de correction d'erreur est conçu séparément de la modulation. L'objectif est alors d'augmenter la distance de Hamming du code. Cependant, la probabilité d'erreur des modulations d'ordre élevé est fonction de la distance

Euclidienne minimale qui n'est pas forcément proportionnelle à la distance de Hamming du code. L'idée clé d'une modulation codée est d'effectuer la modulation avant le codage. Autrement dit, le codage est effectué dans l'espace des signaux de modulation et l'optimisation du code correcteur d'erreur se fait par conséquent dans l'objectif d'augmenter la distance Euclidienne minimale.

Les modulations codées ont été largement utilisées dans les transmission filaires où les canaux peuvent être assimilés à des canaux AWGN. Cependant, dans le cas d'un canal à évanouissement les performances dépendent plutôt de la diversité du code, autrement dit de la distance de Hamming. Les premières approches pour améliorer les performances des modulations codées reposaient sur l'ajout d'un entrelacement symbole pour augmenter l'ordre de diversité [64]. Néanmoins, il a été montré dans [65, 66] qu'utiliser un entrelacement bit à l'émetteur et un décodage souple au récepteur permet de réaliser un meilleur gain que la modulation TCM sur un canal de Rayleigh. Cette technique est connue sous le nom de la modulation codée à bits entrelacés (Bit-Interleaving Coded Modulation ou BICM en anglais) et a connu une grande réussite grâce à sa faible complexité comparée à la technique TCM.

Dans un schéma BICM, l'information est d'abord codée. Ensuite, les bits du mot de code sont entrelacés avant d'être modulés. La démodulation nécessite une étape de *demapping* symbole-bit. Dans le cas d'une démodulation souple, le demapping consiste à marginaliser les probabilités symbole afin d'obtenir des probabilités bit. Soit $x = (x_0, x_1, \cdots, \log_2 M)$ un symbole transmis. L'ensemble des symboles x forme un corps de Galois $\mathbb{GF}(M)$. Soit χ_i^b, $i = 0, 1, \cdots, \log_2 M$, l'ensemble des symboles dont le bit x_i vaut b, $b \in \{0, 1\}$. Dans le cas de la modulation M-QAM, un symbole x est transmis sur deux porteuses I et Q en quadrature de phase. L'amplitude de la porteuse en phase I est modulée avec l'abscisse du point de constellation et l'amplitude de la porteuse en quadrature de phase Q est modulée par l'ordonnée du point de constellation. Nous obtenons ainsi un signal complexe $s = s_I + js_Q$. Si ce signal passe par un canal AWGN complexe $N(0, \sigma^2)$, le démodulateur reçoit un signal bruité $y = y_I + jy_Q$. La démodulation se fait avec l'équation 3.8 qui consiste à marginaliser les probabilités a posteriori des symboles afin de calculer les LLRs des bits x_i.

$$LLR(x_i) = \ln\left(\frac{\sum\limits_{x \in \chi_i^0} e^{\frac{-\|y-s\|^2}{2\sigma^2}}}{\sum\limits_{x \in \chi_i^1} e^{\frac{-\|y-s\|^2}{2\sigma^2}}}\right) \tag{3.8}$$

Cependant, les probabilités symbole ne peuvent pas être recalculées à partir des probabilités bit. Autrement dit, la marginalisation entraîne une perte d'information due à la corrélation des probabilités des bits. De plus, dans les schémas BICM la diversité est augmentée en dépit d'une réduction de la distance Euclidienne minimale entraînant une dégradation des performances sur les canaux Gaussiens sans évanouissement. Ces inconvénients peuvent être résolus au prix d'une augmentation considérable de la complexité du récepteur par un décodage itératif (Bit-Interleaving Coded Modulation with Iterative Decoding ou BICM-ID en anglais) [67] qui consiste à échanger itérativement des informations extrinsèques entre le décodeur et le démodulateur jusqu'à arriver à la convergence.

Pour effectuer une démodulation sans perte d'information, il est possible d'utiliser un code non binaire de même ordre que la modulation considérée. Ce type d'association est simple à concevoir et permet d'avoir des meilleures performances que le décodage BICM-ID. Par exemple, il a été montré dans [59] que l'association dans un canal AWGN d'un

code $\mathbb{GF}(64)$-LDPC avec une modulation 64-QAM réalise, à $TEB = 10^{-4}$, un gain de 1 dB en comparaison avec le décodage BICM-ID. En effet, il est possible d'identifier une connexion plus étroite entre les codes NB-LDPC et les schémas de modulation d'ordre élevé. Contrairement à un décodeur binaire, un décodeur non binaire utilise des LLRs symbole qui sont calculés directement en déterminant la distance entre le signal reçu et un point de constellation unique. Par conséquent, les LLRs symbole ne sont pas corrélés ce qui améliore intuitivement les performances des algorithmes de décodage. Néanmoins, une telle association code-modulation n'est pas a priori adaptée pour la transmission dans des canaux à évanouissement. La robustesse de canaux à évanouissement peut être améliorée par l'utilisation des modulations à étalement de spectre qui permettent d'introduire de la diversité dans le système de transmission. Le principe de la technique d'étalement de spectre est expliqué dans la section suivante.

3.3 Principe des modulations à étalement de spectre

Une modulation à étalement de spectre est une technique qui utilise un spectre de transmission plus large que le spectre du signal d'origine. L'utilisation des modulations à étalement de spectre trouve ses origines dans les applications militaires qui cherchent à masquer le signal transmis afin de réduire la probabilité d'interception. Les modulations à étalement de spectre sont également très répandues dans les applications de communication civile sans fil grâce à leur immunité aux interférences et la possibilité d'établir des liens de communications à accès multiple en utilisant des séquences d'étalement orthogonales. Les modulations à étalement de spectre peuvent être classées en deux catégories :

- ◇ Les modulations à étalement de spectre par saut de fréquence (Frequency-Hopping Spread Spectrum ou FHSS en anglais). Dans ce cas, le signal est commuté en répétition entre plusieurs sous-porteuses générées par un synthétiseur de fréquences commandé par un générateur de séquences pseudo-aléatoires. Un récepteur ne connaissant pas la séquences des fréquences ne peut pas intercepter la communication.
- ◇ Les modulation à étalement de spectre par séquence directe (Direct-Sequence Spread Spectrum ou DSSS en anglais). Dans ce cas, chaque bit de donnée est transmis sous forme d'une séquence pseudo-aléatoire de n chips tel que $T_b = nT_c$, où T_b est la durée d'un bit et T_c est la durée d'un chip. Par conséquent la bande de transmission est plus large que la bande de données puisque $T_c < T_b$.

Les applications modernes opèrent dans des bandes de fréquences limitées avec des débits de transmission très élevés ce qui rend les modulations à étalement de spectre pas adaptées du fait de leur faible efficacité spectrale. Pour résoudre ce problème certains systèmes de communication utilisent des modulations à étalement de spectre qui opèrent au niveau symbole. Par exemple, la norme IS-95 (souvent appelée CDMAone) [68] utilise une modulation DSSS non binaire, dite la modulation orthogonale k-aire (k-ary Orthogonal Modulation ou k-ary OM en anglais) [69], qui consiste à transmettre des symboles de k bits à l'aide de 2^k séquences de Walsh-Hadamard [70]. Chaque séquence de Walsh-Hadamard contient 2^k chips. La démodulation se fait par un banc de filtres adaptés qui permet de calculer la corrélation du signal reçu avec les séquences orthogonales.

La modulation CCSK [71] est une alternative à la modulation orthogonale qui permet de simplifier les traitements en utilisant 2^k séquences construites à partir d'une unique séquence pseudo-aléatoire dite séquence fondamentale. Chaque séquence est obtenue avec

un ou plusieurs décalages cycliques de la séquence fondamentale. Il en découle que l'émetteur ne nécessite qu'un seul registre à décalage. Les registres à décalage permettent de générer des séquences ayant de bonnes propriétés d'auto-corrélation. La réception se fait en calculant la corrélation croisée du signal reçu et la séquence fondamentale. Ce calcul peut se faire dans le domaine des fréquences par transformée de Fourier [72] ce qui réduit la complexité du récepteur. De plus, il a été montré dans [72] que les performances de la modulation CCSK sont comparables à celles de la modulation k-ary OM pour un canal AWGN et un TEB $\leq 10^{-4}$. Les bonnes performances et la simplicité de la modulation CCSK justifient son utilisation dans le *Joint Tactical Information Distribution System* (JTIDS).

Nous consacrons le reste de ce chapitre à l'étude de l'association des codes LDPC non binaires avec la modulation CCSK. Ce choix est motivé par la simplicité de cette association et l'intérêt que peut présenter la modulation CCSK pour différentes applications telles que les systèmes de navigation par satellite (Global Navigation Satellite Systems ou GNSS en anglais) [73]. Il faut noter que l'association des codes LDPC non binaires avec la modulation k-ary OM a été déjà étudiée dans [60, 74]. Par conséquent, des comparaisons entre les performances de la modulation CCSK et la modulation k-ary OM sont faites.

3.4 Association CCSK et codes LDPC non binaires

Nous proposons dans cette section d'étudier la complexité et les performances de l'association d'un code LDPC non binaire et d'une modulation CCSK dans les canaux radio mobiles [75, 76]. Dans cet environnement, les ondes électromagnétiques émises peuvent suivre différents trajets en raison des phénomènes de réflexion, de diffraction et de diffusion causés par les obstacles tels que les bâtiments, les voitures et les arbres. De ce fait, le signal reçu par une antenne est souvent la superposition de plusieurs répliques retardées, atténuées et déphasées du signal émis. En fonction de la phase de chaque réplique, cette superposition peut être constructive ou destructive. Par conséquent, la puissance du signal reçu se caractérise par de fortes fluctuations spatiales.

En fonction des délais observés nous distinguons deux types d'évanouissement. Si les trajets multiples arrivent dans une courte fraction de la période d'un symbole alors l'évanouissement est dit plat ou bien non-sélectif en fréquence. Dans un tel scénario, la propagation peut être modélisée par un seul trajet multiplié par une loi statistique appropriée telle que la loi de Rayleigh, la loi de Rice ou la loi log-normale. Toutefois, les systèmes de communication modernes opèrent avec des débits de transmission très élevés (c'est-à-dire avec des symboles de courtes périodes). Par conséquent, les retards des trajets multiples sont souvent non négligeables par rapport à la période d'un symbole et nous obtenons un évanouissement sélectif en fréquence. Dans cette condition, les symboles s'étalent dans le temps et provoquent une distorsion du signal.

Nous considérons dans notre étude les deux types de canaux en choisissant d'une part le modèle de Rayleigh d'un canal non sélectif en fréquence [77] et d'autre part le modèle exponentiel d'un canal en environnement intérieur sélectif en fréquence [78]. Même si ce sont deux modèles très simples pour modéliser avec précision les canaux du monde réel, ils nous sont utiles pour comparer les différentes approches étudiées.

3.4.1 Transmission dans un canal de Rayleigh non sélectif en fréquence

La figure 3.1 montre le schéma fonctionnel d'un système de communication associant, dans un canal de Rayleigh, un code $\mathbb{GF}(q = 2^m)$-LDPC à une modulation CCSK de même ordre.

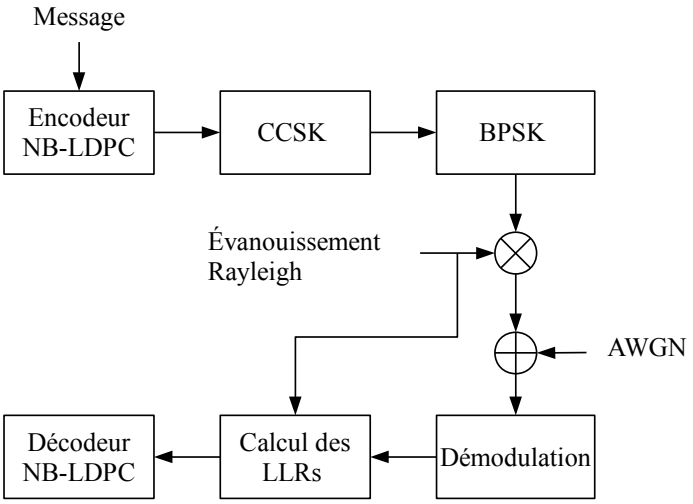


FIGURE 3.1 – Schéma fonctionnel de l'association d'un code LDPC non binaire et d'une modulation CCSK

Le tableau 3.1 montre un exemple de mapping CCSK dans le corps de Galois $\mathbb{GF}(8)$.

TABLE 3.1 – Exemple de mapping CCSK

symbole de $\mathbb{GF}(16)$	séquence CCSK
0000	0001110110010101
0001	1000111011001010
0010	0100011101100101
0011	1010001110110010
0100	0101000111011001
0101	1010100011101100
0110	0101010001110110
0111	0010101000111011
1000	1001010100011101
1001	1100101010001110
1010	0110010101000111
1011	1011001010100011
1100	1101100101010001
1101	1110110010101000
1110	0111011001010100
1111	0011101100101010

Comme nous pouvons le constater, la modulation CCSK est construite à partir d'une séquence fondamentale notée pn et de longueur q chips. Cette séquence peut être générée par un registre à décalage à rétroaction linéaire (Linear Feedback Shift Register ou LFSR

en anglais). Chaque séquence pn_j de la modulation CCSK, $j = 0, 1, \cdots, q-1$, est obtenue en décalant circulairement la séquence fondamentale par j positions :

$$pn_j[i] = pn[(i+j)_q] \quad i = 0, 1, \cdots, q-1 \tag{3.9}$$

L'association d'un code LDPC non binaire et d'une modulation CCSK n'ajoute aucune complexité au récepteur puisqu'elle se fait directement en attribuant une séquence CCSK à chaque symbole du mot de code. Ainsi, à un symbole β_j du corps de Galois nous attribuons la séquence pn_j.

La forme d'onde est obtenue ensuite en appliquant une modulation BPSK. Dans ce qui suit, $\hat{pn}_j$ désigne la séquence pn_j modulée en BPSK :

$$\hat{pn}_j(i) = 1 - 2pn_j(i) \quad i = 0, 1, \cdots, q-1 \tag{3.10}$$

La démodulation CCSK consiste à calculer les LLRs symbole par corrélation croisée qui permet de déterminer le taux de ressemblance entre le signal reçu et les séquences de la modulation CCSK. Nous considérons dans nos calculs le modèle d'un canal de Rayleigh non sélectif en fréquence. C'est un modèle simple considéré comme raisonnable pour simuler la propagation dans un environnement avec un très grand nombre de réflexions et sans visibilité directe tel que les zones urbaines fortement denses. Cependant, ce modèle manque de flexibilité et il est souvent difficile de l'appliquer à d'autre canaux du monde réel.

Dans un canal de Rayleigh, le signal reçu $z = [z_i]_{0 \leq i \leq q-1}$ s'exprime par :

$$z_i = r_i \hat{pn}_j(i) + \eta_i \tag{3.11}$$

où $\eta = [\eta_i]_{0 \leq i \leq q-1}$ est un bruit AWGN $N(0, \sigma^2)$ et $r = [r_i]_{0 \leq i \leq q-1}$ est l'enveloppe du signal reçu obéissant à la loi statistique de Rayleigh :

$$p(r_i) = 2r_i e^{-r_i^2} \quad r_i \in \mathbb{R}_+^* \tag{3.12}$$

Un canal de Rayleigh en bloc est un canal où les coefficients de Rayleigh restent constants sur un ensemble de n_r chips. Dans le cas d'un entrelacement parfait chaque chip est affecté par un coefficient de Rayleigh indépendant et nous obtenons $n_r = 1$. Un canal AWGN peut être modélisé sous la forme d'un canal de Rayleigh sans évanouissement (autrement dit $r_i = 1 \ \forall i$).

Si le récepteur dispose d'une information complète sur l'état du canal (Channel State Information ou CSI en anglais) alors le rapport de vraisemblance logarithmique d'un symbole β_j s'obtient par :

$$\begin{aligned} LLR(\beta_j) &= \ln\left(\frac{p\big(z|(r,\hat{pn}_j)\big)}{p\big(z|(r,\hat{pn}_0)\big)}\right) \\ &= \ln\left(\frac{\prod_{i=0}^{q-1} p\big(z_i|(r_i,\hat{pn}_j[i])\big)}{\prod_{i=0}^{q-1} p\big(z_i|(r_i,\hat{pn}_0[i])\big)}\right) \end{aligned} \tag{3.13}$$

Sachant :

$$p\big(z_i|(r_i,\hat{pn}_j[i])\big) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_i - r_i \hat{pn}_j[i])^2}{2\sigma^2}} \tag{3.14}$$

l'équation 3.13 devient :

$$LLR(\beta_j) = \frac{1}{\sigma^2}\sum_{i=0}^{q-1}(r_i z_i \hat{pn}_j[i]) - \frac{1}{\sigma^2}\sum_{i=0}^{q-1}(r_i z_i \hat{pn}_0[i]) \tag{3.15}$$

En supprimant le coefficient $\frac{1}{\sigma^2}$ et en tenant compte de l'équation 3.9 nous obtenons :

$$\begin{aligned} LLR(\beta_j) &\propto \sum_{i=0}^{q-1}(w_i \hat{pn}[(i+j)_q]) - \sum_{i=0}^{q-1}(w_i \hat{pn}[i]) \\ &\propto w \star \hat{pn}[j] - w \star \hat{pn}[0] \end{aligned} \tag{3.16}$$

avec $w = [w_i = r_i z_i]_{0 \le i \le q-1}$. Il en découle que le calcul des LLRs peut se faire dans le domaine fréquentiel. Cette propriété permet de passer d'une complexité de calcul de l'ordre de q^2 vers une complexité de l'ordre de $q \log_2 q$ grâce à l'algorithme de transformée de Fourier rapide. La figure 3.2 montre l'architecture du circuit de calcul des LLRs. Les LLRs sont normalisés par le terme $w \star \hat{pn}[0]$ mémorisé dans un registre.

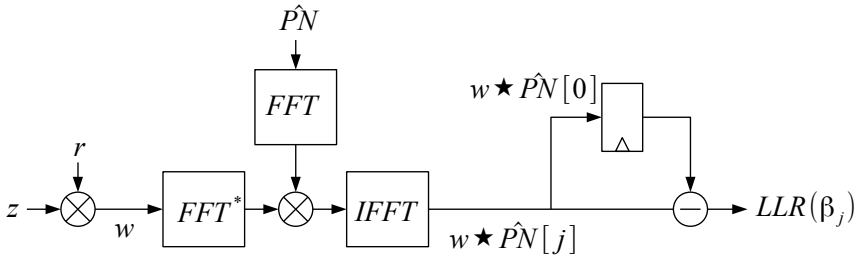


FIGURE 3.2 – Circuit d'un démodulateur CCSK associé à un décodeur non binaire

Pour déterminer les performances de l'association des codes LDPC non binaires avec la modulation CCSK, nous avons lancé une série de simulations de Monte-Carlo. Toutes les simulations de ce chapitre considèrent des codes $\mathbb{GF}(64)$-LDPC. Le décodage est réalisé avec l'algorithme EMS ou bien l'algorithme Min-Sum (les performances de l'algorithme

Min-Sum étant très proches des performances de l'algorithme optimal BP). Des comparaisons sont faites avec un code LDPC binaire construit avec l'algorithme PEG [79, 80] et décodé par l'algorithme Min-Sum [81]. Le tableau 3.2 présente les différents paramètres utilisés.

TABLE 3.2 – Paramètres des codes LDPC utilisés dans les simulations

	$\mathbb{GF}(2)$-LDPC	$\mathbb{GF}(64)$-LDPC	
N en bits	1024	144, 1008, 1152, 1440	
R	$\frac{506}{1024}$	$\frac{1}{2}$	
Algorithme de décodage	Min-Sum	Min-Sum	EMS
n_m	-	64	12
offset	-	-	1
n_{iter}	1000	100	100

La modulation CCSK est construite par une séquence pseudo-aléatoire générée par un LFSR dont le polynôme primitif est $P[X] = X^6 + X^5 + X^4 + X + 1$. Cet LFSR permet de générer une séquence périodique de 63 chips à laquelle nous ajoutons un chip 0 au prix d'une légère dégradation de la fonction d'auto-corrélation. La figure 3.3 illustre la fonction d'auto-corrélation de notre séquence fondamentale après modulation BPSK.

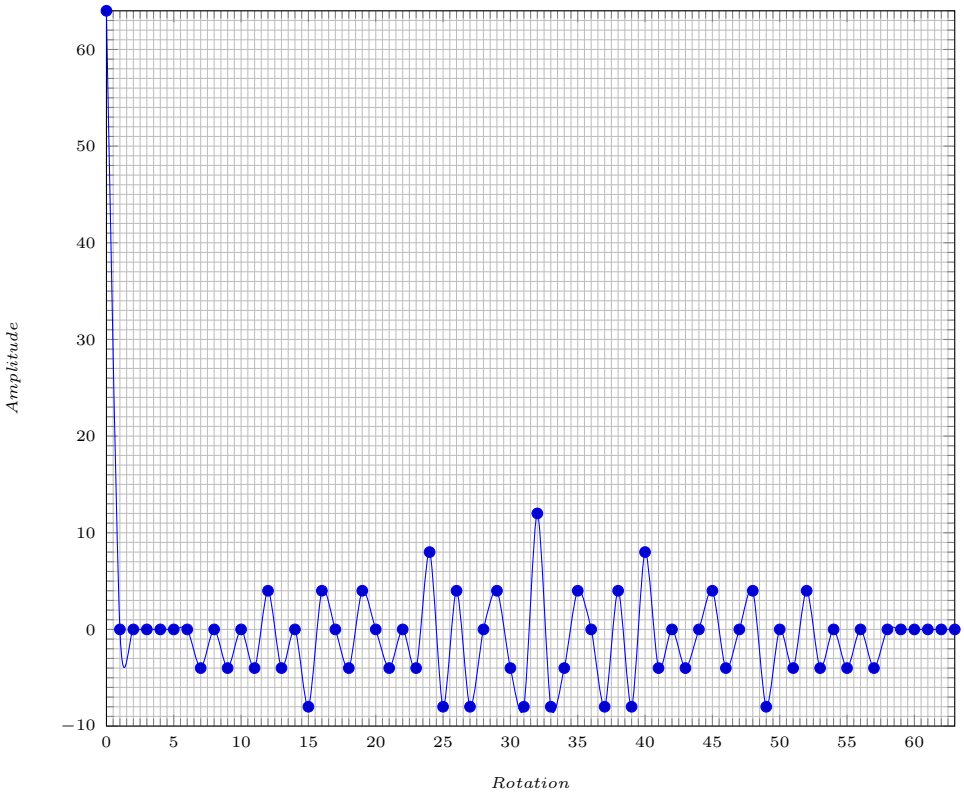


FIGURE 3.3 – Fonction d'auto-corrélation de la séquence fondamentale

La modulation CCSK d'un code LDPC binaire se fait en associant une séquence CCSK à chaque groupe de $m = 6$ bits consécutifs du mot de code. Les LLRs symboles sont calculés en utilisant l'équation 3.16. Ensuite, les LLRs bit sont obtenus en marginalisant les LLRs symbole. Si $x = [x_i]_{0 \leq i < m}$ désigne un groupe de m bits consécutifs du mot de code alors l'information intrinsèque du décodeur est obtenue par l'équation 3.17.

$$\begin{aligned} LLR(x_i) &= \ln \frac{p(x_i = 0|z)}{p(x_i = 1|z)} \\ &= \ln \left(\frac{\sum\limits_{\beta_j \in \chi_i^0} e^{LLR(\beta_j)}}{\sum\limits_{\beta_j \in \chi_i^1} e^{LLR(\beta_j)}} \right) \end{aligned} \tag{3.17}$$

où χ_i^0 (respectivement χ_i^1), désigne l'ensemble des symboles dont le bit de position i vaut 0 (respectivement vaut 1). Pour compenser l'information perdue par la marginalisation, nous considérons aussi le cas d'un récepteur avec démodulation itérative BICM. Dans ce cas, le décodeur renvoie au démodulateur l'information a priori $L(x_i) = \ln \frac{p(x_i=0)}{p(x_i=1)}$ qui permettra de recalculer l'information intrinsèque comme suit :

$$\begin{aligned} LLR(x_i) &= \ln \frac{p(x_i = 0|z)}{p(x_i = 1|z)} \\ &= \ln \frac{\sum\limits_{\beta_j \in \chi_i^0} p(x = \beta_j|z)}{\sum\limits_{\beta_j \in \chi_i^1} p(x = \beta_j|z)} \\ &= \ln \frac{\sum\limits_{\beta_j \in \chi_i^0} p(z|x = \beta_j)p(x = \beta_j)}{\sum\limits_{\beta_j \in \chi_i^1} p(z|x = \beta_j)p(x = \beta_j)} \\ &= \ln \frac{\sum\limits_{\beta_j \in \chi_i^0} p(z|x = \beta_j)e^{\sum\limits_{k=0,k\neq i}^{m-1} f(\beta_j,k)L(x_k)}}{\sum\limits_{\beta_j \in \chi_i^1} p(z|x = \beta_j)e^{\sum\limits_{k=0,k\neq i}^{m-1} f(\beta_j,k)L(x_k)}} \end{aligned} \tag{3.18}$$

avec

$$f(\beta_j, k) = \begin{cases} 1, & \text{si} \quad \beta_j(k) = 0 \\ 0, & \text{sinon} \end{cases} \tag{3.19}$$

Outre la modulation CCSK, les simulations sont faites aussi avec la modulation 64-OM construite par une matrice de Sylvester-Hadamard d'ordre 64 [82].

La figure 3.4 montre, dans un canal AWGN, le seuil théorique de Shannon pour des mots de code de taille finie ainsi que les performances de codes $\mathbb{GF}(64)$-LDPC associés à la modulation CCSK. Le seuil de Shannon est calculé par l'outil délivré par TELECOM Bretagne dans [83] en tenant compte du rendement effectif obtenu après modulation CCSK. D'une autre manière, les rendements considérés dans le calcul du seuil de Shannon sont égaux aux rendements des codes NB-LDPC multipliés par le rendement de la modulation CCSK. Pour un TEP $= 10^{-4}$, les courbes tracées montrent un écart approximatif de 1 dB entre le seuil de Shannon et les performances de l'algorithme Min-Sum.

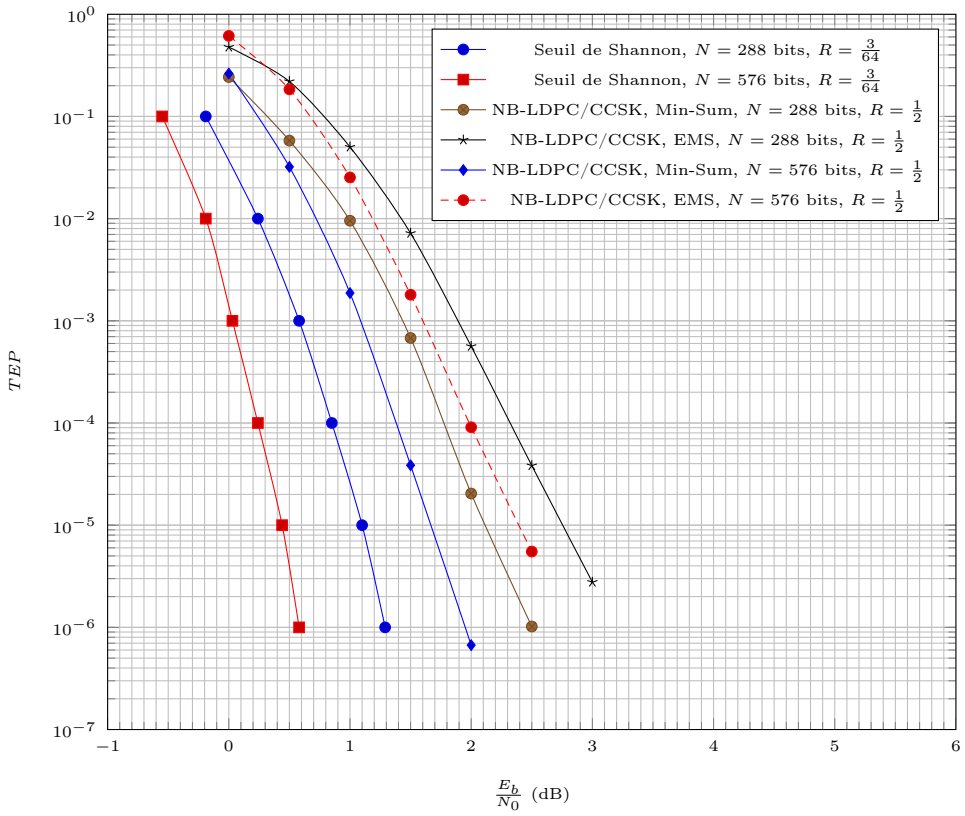


FIGURE 3.4 – Comparaison sur un canal AWGN du TEP de l'association $\mathbb{GF}(64)$-LDPC/CCSK par rapport au seuil théorique de Shannon

Les figures 3.5 et 3.6 permettent de comparer, respectivement dans un canal AWGN et un canal de Rayleigh, les performances des modulations BPSK, CCSK ou 64-OM associées à un code $\mathbb{GF}(64)$-LDPC. Dans le cas de la modulation BPSK, n_r désigne le nombre de bits affectés par un même coefficient de Rayleigh. Dans le cas des modulations CCSK et 64-OM, n_r désigne le nombre de chips affectés par un même coefficient de Rayleigh. Pour obtenir une comparaison équilibrée, les trois techniques de modulation doivent bénéficier de la même diversité du canal. Par exemple, si nous considérons un canal avec un entrelacement idéal, chaque bit d'un mot de code modulé en BPSK est multiplié par un coefficient de Rayleigh indépendant ($n_r = 1$ bit). En modulation CCSK ou 64-OM, un bit du mot de code correspond à $\frac{64}{6}$ chips. Ainsi, pour garder le même ordre de diversité que la modulation BPSK, chaque $\frac{64}{6}$ chips doivent être multipliés par le même coefficient de Rayleigh. Cependant, 64 n'étant pas divisible par 6 nous considérons $n_r = 11$ chips. Il en découle qu'un mot de code de taille $N = 1008$ bits subit 1008 coefficients de Rayleigh dans le cas de la modulation BPSK et 978 coefficients de Rayleigh dans le cas des modulations CCSK et 64-OM. En analysant les résultats, nous constatons que les modulations CCSK et 64-OM possèdent des performances identiques. De plus, nous observons dans le canal AWGN et à TEP $= 10^{-4}$ un écart approximatif de 0.5 dB en faveur des modulations CCSK et 64-OM par rapport à la modulation BPSK. Cet écart s'élargit dans le canal de Rayleigh pour atteindre 1.8 dB. Ce gain énorme s'explique par la diversité apportée par les modulations à étalement de spectre. Cette diversité se traduit par une augmentation de la distance Euclidienne minimale des symboles non binaires transmis (la distance Euclidienne minimale des symboles transmis vaut $\epsilon = 2$ pour la modulation BPSK, $\epsilon = 11.31$ pour la modulation 64-OM et $\epsilon = 12.32$ pour la modulation CCSK).

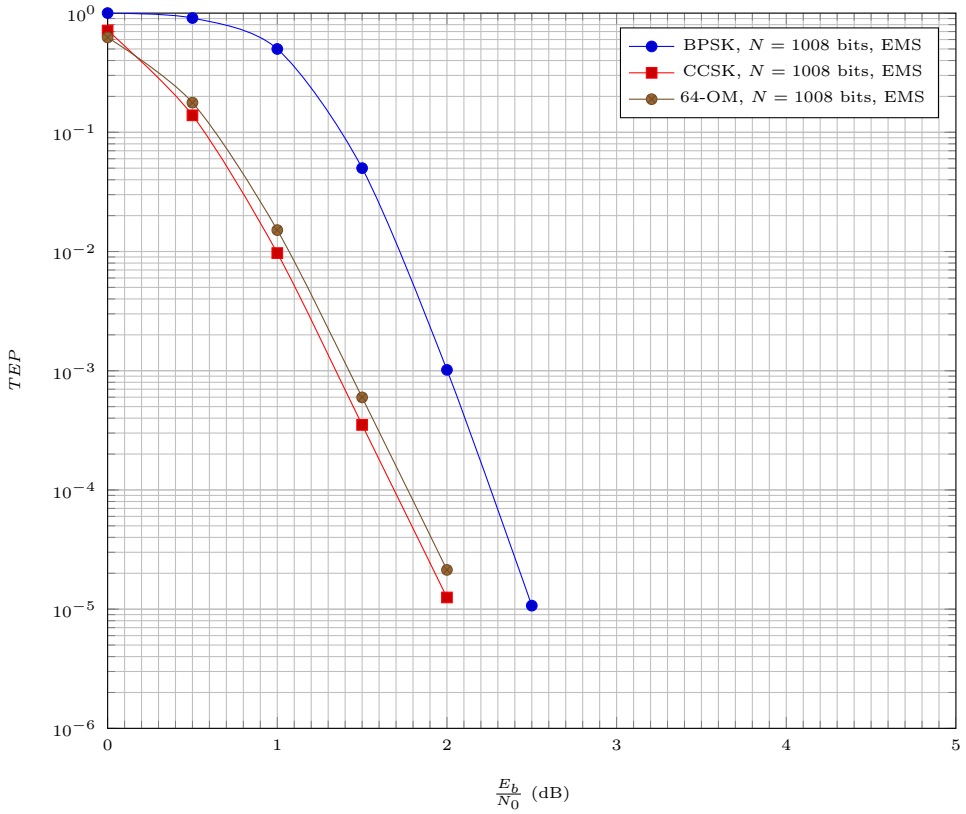


FIGURE 3.5 – Comparaison dans un canal de AWGN du TEP des modulations BPSK, CCSK et 64-OM associées à un code $\mathbb{GF}(64)$-LDPC de rendement $R = \frac{1}{2}$

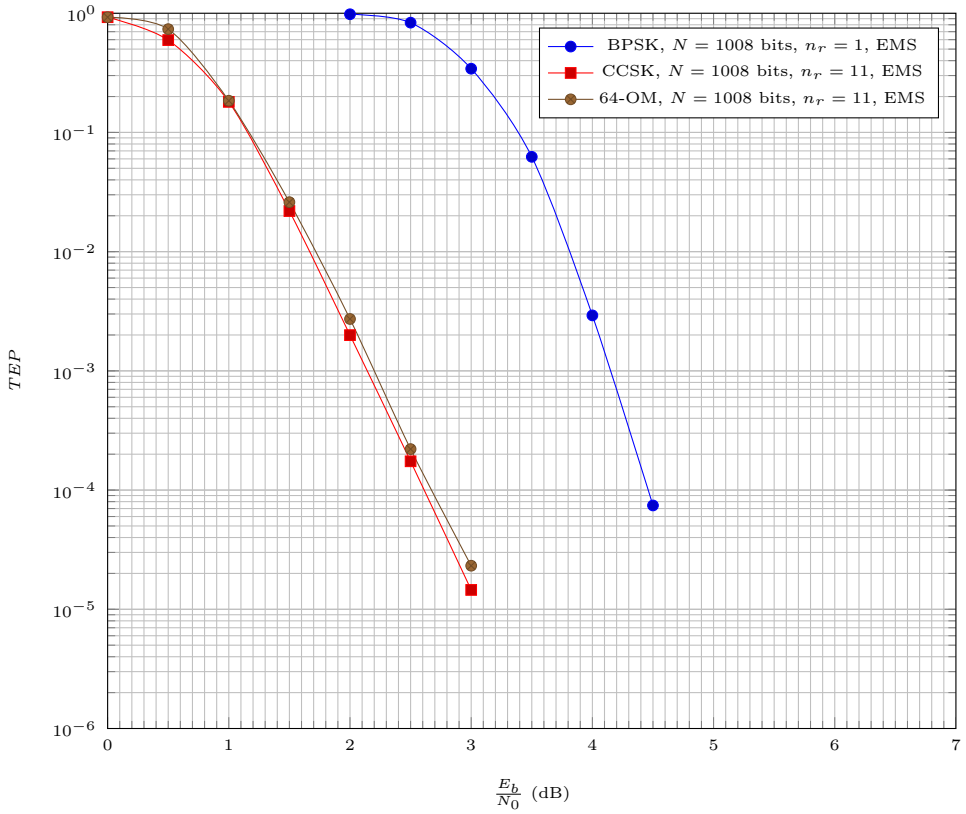


FIGURE 3.6 – Comparaison dans un canal de Rayleigh du TEP des modulations BPSK, CCSK et 64-OM associées à un code $\mathbb{GF}(64)$-LDPC de rendement $R = \frac{1}{2}$

Les figures 3.7 et 3.8 servent à comparer les performances d'un code $\mathbb{GF}(64)$-LDPC et un code LDPC binaire de taille comparable associés à la modulation CCSK. Puisque la démodulation binaire souffre d'une perte d'information, nous considérons aussi les performances obtenues par décodage BICM-ID. Nous constatons que l'association de la modulation CCSK avec le code LDPC non binaire permet d'obtenir des performances nettement meilleures que le code LDPC binaire même en utilisant un décodage BICM-ID. Le gain réalisé par le décodeur non binaire s'explique par sa capacité de bénéficier pleinement de la diversité ajoutée par la modulation CCSK. D'autre part, nous observons que les performances de la modulation CCSK associée au code binaire sont moins bonnes dans le canal AWGN que la modulation BPSK associée au même code. Cette observation change dans le canal de Rayleigh où la modulation CCSK donne de meilleures performances que la modulation BPSK pour $\frac{E_b}{N_0} < 5$ dB. En effet, cette observation s'explique par le fait que la diversité ajouté par les séquences CCSK dans le canal AWGN n'est pas suffisante pour compenser l'information perdue par marginalisation. Par contre, cette diversité couplée à la diversité innée du canal de Rayleigh (puisque une séquence CCSK est multipliée par plusieurs coefficients de Rayleigh destructifs) permet de compenser la perte d'information et obtenir de meilleures performances que la modulation BPSK. D'une autre manière, en BPSK, les bits sont démodulés séparément et par conséquent la démodulation d'un bit effacé ne peut pas bénéficier de l'information apportée par le reste des bits. En CCSK, la démodulation se fait par symbole et par conséquent la démodulation d'un bit effacé bénéficie de l'information apportée par les autres bits du même symbole.

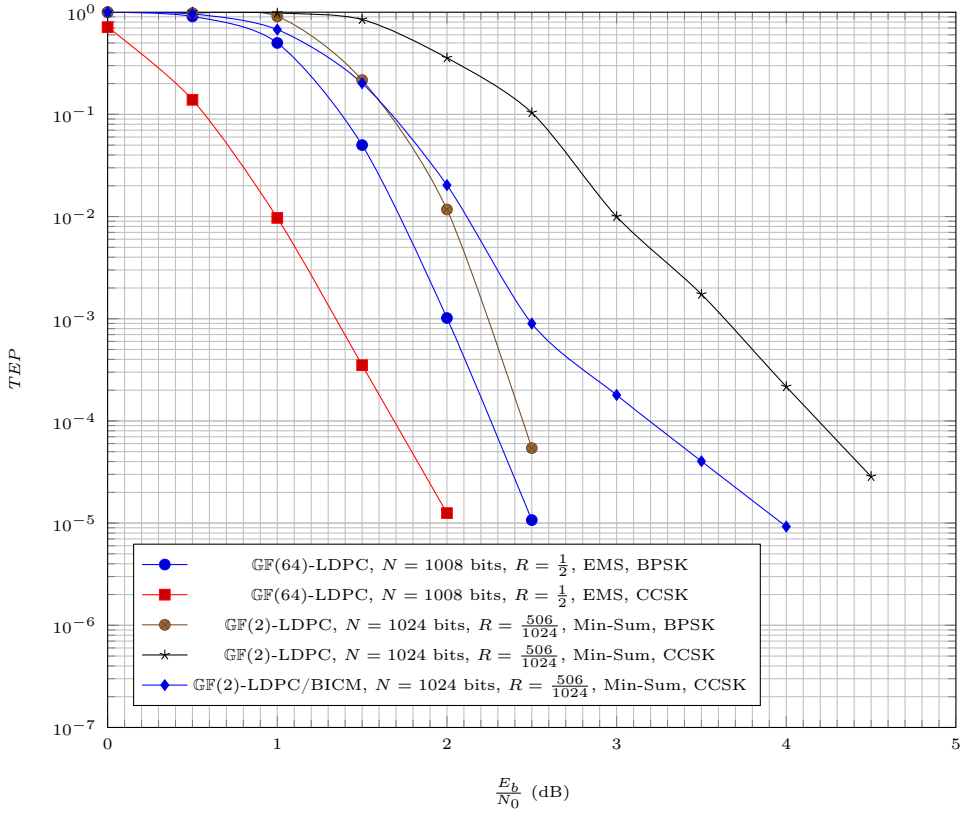


FIGURE 3.7 – Comparaison dans un canal AWGN du TEP d'un code LDPC non binaire et un code LDPC binaire associés à une modulation CCSK

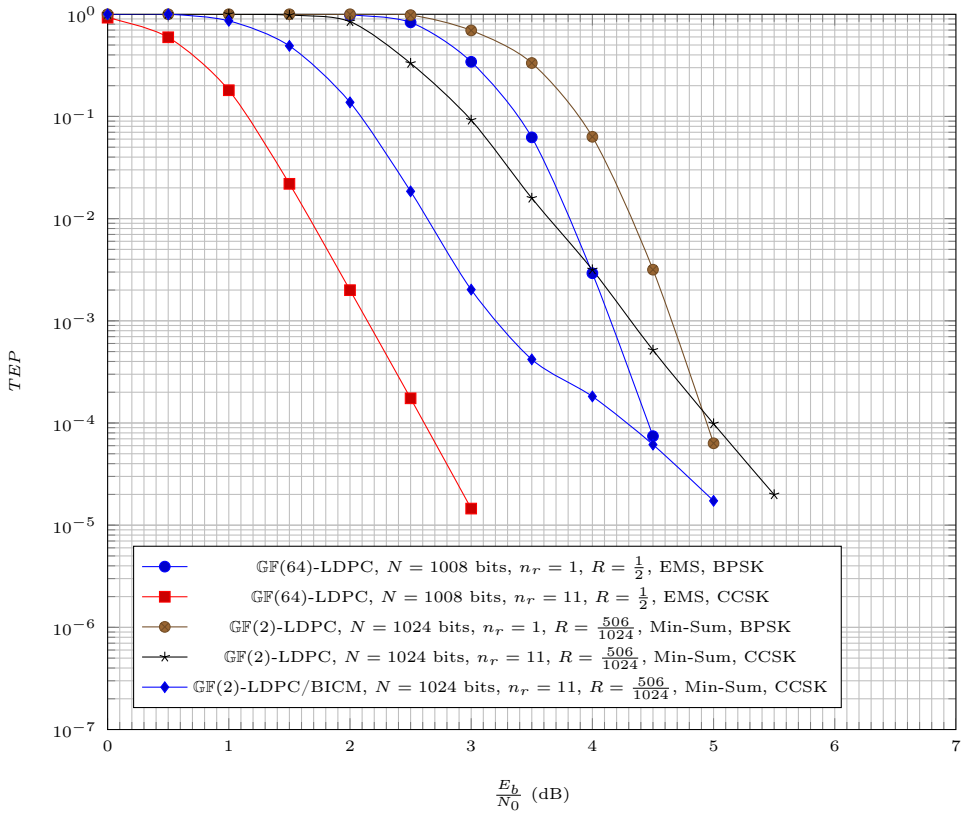


FIGURE 3.8 – Comparaison dans un canal de Rayleigh du TEP d'un code LDPC non binaire et un code LDPC binaire associés à une modulation CCSK

3.4.2 Transmission dans un canal sélectif en fréquence

Dans un canal sélectif en fréquence, le signal reçu est la convolution du signal émis avec la réponse impulsionnelle du canal $h(j) = \sum_{i=0}^{N_{\text{tap}}-1} h_i \delta(i-j)$, où N_{tap} est le nombre de coefficients complexes du canal. La fonction de transfert du canal, notée H, est par définition la transformée de Fourier de sa réponse impulsionnelle. Dans ce type de canaux, le signal subit un étalement temporaire qui provoque de l'interférence inter-symbole. Cette distorsion du signal peut être contrée par deux approches :

- ◇ Une approche optimale qui consiste à minimiser la probabilité d'erreur. Il s'agit d'utiliser un algorithme de détection probabiliste qui repose soit sur la maximisation de la probabilité a postériori (Maximum A posteriori Probability ou MAP en anglais) ou sur l'estimation de la séquence au maximum de vraisemblance (Maximum Likelihood Sequence Estimation ou MLSE en anglais). La technique MAP vise à minimiser la probabilité d'erreur symbole tandis que la technique MLSE, qui peut être implantée avec l'algorithme de Viterbi, vise à minimiser la probabilité d'erreur séquence. Les techniques MAP et MLSE offrent des performances comparables et possèdent une complexité qui s'accroît exponentiellement avec la longueur de la réponse impulsionnelle du canal.
- ◇ Une approche sous optimale appelée égalisation qui consiste à récupérer la forme du signal transmis en inversant la distorsion du canal. Lorsque cette opération est réalisée par un filtre adapté l'égalisation est dite linéaire. Un exemple simple d'égalisation linéaire est la technique de forçage à zéro (Zero Forcing ou ZF en anglais) qui consiste à considérer la fonction réciproque de la réponse impulsionnelle du canal. Cependant en présence de bruit dans le canal, l'égalisation ZF entraîne une amplification du bruit et

par la suite une dégradation des performances. Ce problème peut être résolu en optimisant le filtre d'égalisation selon le critère de l'erreur quadratique moyenne (Minimum Mean Square Error ou MMSE en anglais) ce qui permet de trouver un compromis entre la réduction des interférences inter-symbole et du bruit.

Une description complète des approches citées ci-dessus se trouvent dans [84, 85, 86, 87].

L'égalisation canal peut se faire dans le domaine temporel. Dans ce cas, il s'agit de calculer le produit de convolution du signal reçu avec la réponse impulsionnelle du filtre. Cependant, la complexité matérielle de cette opération augmentent rapidement avec le débit de transmission des données et devient prohibitive pour les hauts débits. Une approche qui permet de contourner ce problème est d'utiliser une transmission OFDM (Orthogonal Frequency-Division Multiplexing). L'idée clé consiste à transmettre l'information sur plusieurs sous-porteuses de bandes étroites. L'évanouissement du canal vue par chaque sous-porteuse devient plat, et par conséquent l'égalisation se transforme en une simple multiplication. La technique OFDM peut être efficacement implantée en utilisant une IFFT en émission et une FFT en réception, ce qui réduit encore la complexité du système. De plus, l'effet de l'interférence inter-symbole peut être confiné dans un intervalle de garde qui sépare chaque deux symboles OFDM consécutifs. Afin de mieux simplifier le récepteur OFDM, l'intervalle de garde peut être défini sous forme de préfixe cyclique (Cyclic Prefix ou CP en anglais). Le préfixe cyclique est construit en copiant les derniers échantillons du symbole OFDM dans sa partie avant. Ce principe permet de transformer la convolution du canal en une convolution circulaire. Par conséquent, en supprimant le préfixe cyclique au récepteur, les symboles OFDM peuvent être traités séparément. Toutefois, l'utilisation du préfixe cyclique dans un système de transmission mono-porteuse permet aussi de réduire la complexité du récepteur en transformant l'égalisation dans le domaine fréquentielle. Ce schéma de transmission est connu en littérature sous le nom de transmission mono-porteuse avec égalisation fréquentielle (Single Carrier Frequency-Domain-Equalization ou SC-FDE en anglais). Globalement, nous pouvons considérer qu'un système SC-FDE possède une complexité comparable à celle d'un système OFDM compte tenu des faits suivant :

- ◇ L'introduction du préfixe cyclique dans une transmission mono-porteuse permet de réduire considérablement la complexité de l'égaliseur en effectuant les traitements bloc par bloc dans le domaine fréquentielle.
- ◇ Les deux techniques SC-FDE et OFDM utilisent une FFT et une IFFT même si c'est pour réaliser des traitements différents. En effet, dans le système OFDM, l'opération FFT et l'opération IFFT servent à implémenter respectivement la démodulation et la modulation. En contrepartie, dans un système SC-FDE, ces deux opérations sont utilisées pour migrer du domaine temporel au domaine fréquentielle et vice versa afin d'effectuer l'égalisation dans le domaine fréquentielle.
- ◇ La technique OFDM possède un facteur de crête très élevé (Peak-to-Average Power Ratio ou PAPR en anglais) et nécessite par la suite des amplificateurs hautement linéaires. De plus, la technique OFDM est très sensible aux problèmes de synchronisation.

Finalement, il a été montré que sous certaines conditions les systèmes SC-FDE peuvent avoir des meilleures performances que les systèmes OFDM [88, 89]. Dans le reste de ce chapitre, nous allons montrer que dans un système SC-FDE l'association d'un code LDPC non binaire et d'une modulation CCSK permet à la fois de réduire la complexité du récepteur et améliorer les performances de décodage.

Démodulation CCSK et détection ML dans un système de transmission mono-porteuse avec préfixe cyclique

La figure 3.9 montre le diagramme en blocs d'un système de transmission mono-porteuse avec préfixe cyclique.

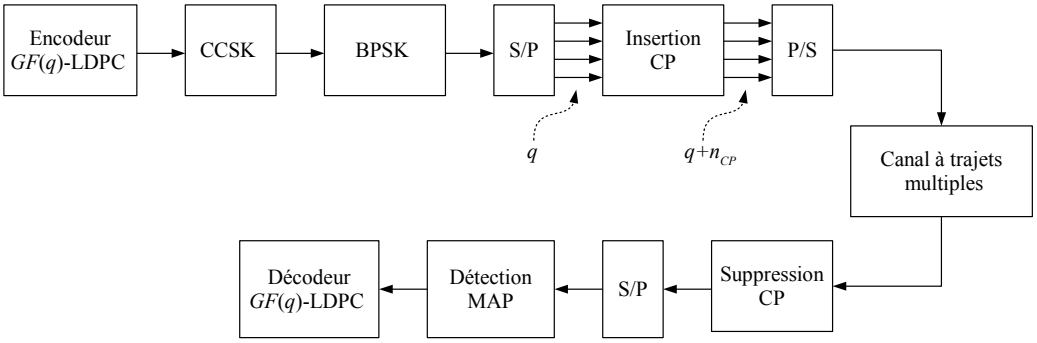


FIGURE 3.9 – Diagramme en bloc d'un système de transmission mono-porteuse avec préfixe cyclique et détection ML

L'émetteur commence par effectuer le mapping CCSK qui consiste à recevoir un symbole $\beta_j \in \mathbb{GF}(q)$ généré par l'encodeur LDPC et le convertir en une séquence pn_j. Ensuite, il applique une modulation BPSK pour d'obtenir la forme d'onde $\hat{pn}_j$. L'étape suivante consiste à insérer un préfixe cyclique de longueur n_{cp} au début de chaque forme d'onde transmise. Le préfixe cyclique permet d'éliminer l'interférence inter-symbole et transformer la convolution linéaire du canal en une convolution circulaire. Le récepteur commence par supprimer le préfixe cyclique. Après cette opération, le signal reçu $z = [z_i]_{0 \leq i \leq q-1}$ prend la forme :

$$\begin{aligned} z[i] &= h \odot \hat{pn}_j[i] + \eta_i \\ &= \sum_{k=0}^{q-1} h[k] \cdot \hat{pn}_j[(i-k)_q] + \eta_i \end{aligned} \tag{3.20}$$

où $h[k] = 0 \; \forall k \geq N_{tap}$ et η_i est un échantillon d'un bruit additive blanc gaussien complexe $N(0, \sigma^2)$. Une variable aléatoire $\eta = \eta_{re} + j\eta_{im}$ est dite une variable complexe gaussienne si η_{re} et η_{im} sont deux variables gaussiennes dé-corrélées et possédant la même variance σ^2. Dans notre cas, les deux variables possèdent une distribution normale. Par conséquent la distribution gaussienne de la variable complexe η est donnée par :

$$p(\eta_{re}, \eta_{im}) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{\frac{-(\eta_{re}^2 + \eta_{im}^2)}{2\sigma^2}} \tag{3.21}$$

Intuitivement, la démodulation optimale consiste à trouver la séquence CCSK ayant le maximum de vraisemblance et ce en calculant la corrélation du signal reçu avec les séquences obtenues par convolution de la réponse impulsionnelle du canal et des séquences CCSK. Pour démontrer cette hypothèse, nous considérons la définition des LLRs de l'équation 3.22 en supposant que les symboles transmis sont équiprobables et que le récepteur

dispose d'une connaissance parfaite de l'état du canal.

$$\begin{aligned} LLR(\beta_j) &= \ln\left(\frac{p\big(z|(h,\hat{pn}_j)\big)}{p\big(z|(h,\hat{pn}_0)\big)}\right) \\ &= \ln\left(\frac{p\big(z_{re}|(h_{re},\hat{pn}_j)\big)\cdot p\big(z_{im}|(h_{im},\hat{pn}_j)\big)}{p\big(z_{re}|(h_{re},\hat{pn}_0)\big)\cdot p\big(z_{im}|(h_{im},\hat{pn}_0)\big)}\right) \end{aligned} \tag{3.22}$$

Si nous tenons aussi compte des équations 3.20 et 3.21 alors les LLRs s'expriment par :

$$LLR(\beta_j) = \mathrm{Re}\left(\frac{1}{\sigma^2}\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}_j\big)[i]\right) - \mathrm{Re}\left(\frac{1}{\sigma^2}\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}_0\big)[i]\right) \tag{3.23}$$

où $\mathrm{Re}(\cdot)$ désigne la partie réel d'un nombre complexe. En supprimant le facteur en commun $\frac{1}{\sigma^2}$ nous obtenons :

$$LLR(\beta_j) \propto \mathrm{Re}\left(\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}_j\big)[i]\right) - \mathrm{Re}\left(\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}_0\big)[i]\right) \tag{3.24}$$

En considérant l'équation 3.9, l'expression des LLRs prend la forme :

$$\begin{aligned} LLR(\beta_j) &\propto \mathrm{Re}\left(\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}\big)[(i+j)_q]\right) - \mathrm{Re}\left(\sum_{i=0}^{q-1} z^*[i]\cdot\big(h\odot\hat{pn}\big)[i]\right) \\ &\propto \mathrm{Re}\Big(z\star\big(h\odot\hat{pn}\big)[j]\Big) - \mathrm{Re}\Big(z\star\big(h\odot\hat{pn}\big)[0]\Big) \end{aligned} \tag{3.25}$$

Il en découle que les LLRs symboles s'obtiennent par corrélation croisée de z et $h\odot\hat{pn}$. Par conséquent, la démodulation ML peut être efficacement implantée par transformée de Fourier discrète ce qui simplifie considérablement la complexité du récepteur. La figure 3.10 montre un exemple d'architecture pour la démodulation ML.

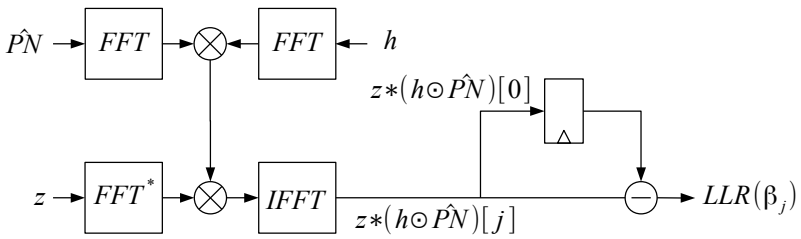


FIGURE 3.10 – Démodulation ML pour une transmission mono-porteuse utilisant un préfixe cyclique, un codage non binaire et une modulation CCSK

Démodulation CCSK et égalisation fréquentielle du canal dans un système de transmission mono-porteuse avec préfixe cylcique

La figure 3.11 montre un système de transmission mono-porteuse avec préfixe cyclique dans lequel nous avons remplacé la détection ML par une égalisation fréquentielle de type MMSE.

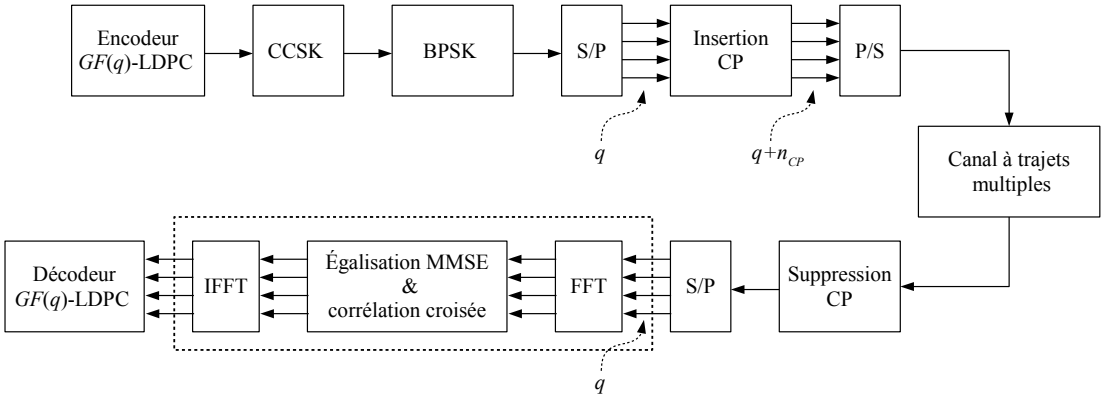


FIGURE 3.11 – Diagramme en bloc d'un système de transmission mono-porteuse avec préfixe cyclique et égalisation MMSE

Dans un système de transmission mono-porteuse avec préfixe cyclique, l'égalisation peut se faire dans le domaine fréquentiel et nécessite une FFT et une IFFT. De même, l'utilisation de la modulation CCSK en émission permet de transformer le calcul des LLRs en un calcul de corrélation croisée qui peut aussi s'effectuer par une FFT et une IFFT. Par conséquent, il est possible de fusionner les calculs et utiliser une seule FFT et une seule IFFT dans le récepteur. En effet, dans le domaine fréquentiel, les coefficients de l'égaliseur MMSE s'obtiennent par :

$$\xi_{\mathrm{SC}}[i] = \frac{H^*[i]}{\|H[i]\|^2 + \sigma^2} \quad i = 0, 1, \cdots, q-1 \tag{3.26}$$

Si Z désigne la FFT d'ordre q du signal z, l'égalisation s'obtient en multipliant les coefficients $\xi_{\mathrm{SC}}[i]$ avec les points $Z[i]$. Si de plus $\hat{PN}$ désigne la FFT d'ordre q de la séquence $\hat{pn}$, la corrélation croisée de la séquence fondamentale $\hat{pn}$ avec le signal égalisé se calcule dans le domaine fréquentiel par :

$$\Psi_{\mathrm{SC}}[i] = \hat{PN}[i]\xi_{\mathrm{SC}}[i]Z[i] \quad i = 0, 1, \cdots, q-1 \tag{3.27}$$

Si ψ_{SC} désigne la IFFT d'ordre q de Ψ_{SC} alors les LLRs s'expriment par :

$$LLR(\beta_i) = \mathrm{Re}\big(\psi_{\mathrm{SC}}[i]\big) - \mathrm{Re}\big(\psi_{\mathrm{SC}}[0]\big) \quad i = 0, 1, \cdots, q-1 \tag{3.28}$$

Démodulation CCSK et égalisation fréquentielle du canal dans un système OFDM

La figure 3.12 montre un système OFDM utilisant un codage LDPC non binaire et une modulation CCSK.

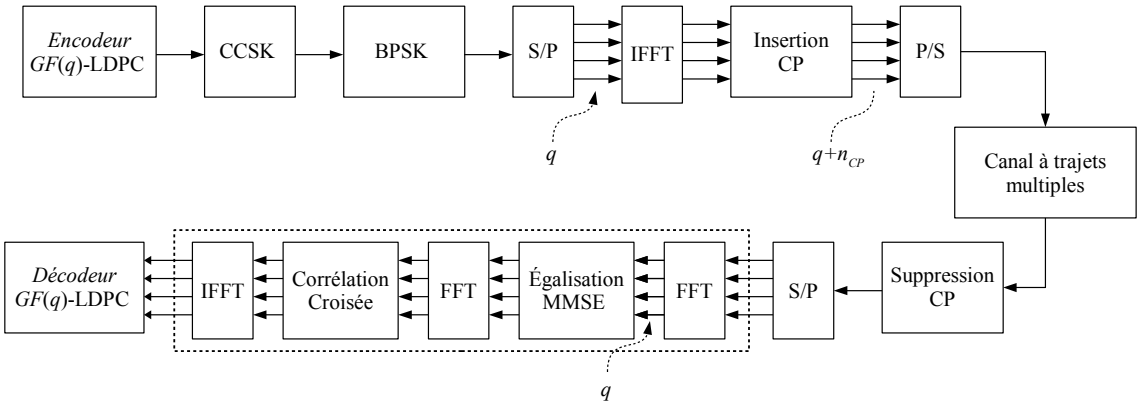


FIGURE 3.12 – Diagramme en bloc d'un système OFDM avec égalisation MMSE

En émission, un symbole OFDM est obtenu en appliquant une IFFT à la séquence CCSK. Ensuite, un préfixe cyclique est inséré à l'entête du symbole OFDM. En réception, le préfixe cyclique est supprimé et ensuite une FFT est appliquée au signal reçu pour séparer les sous-porteuses. En traversant le canal, chaque sous-porteuse subit un évanouissement plat puisqu'elle occupe une bande de fréquence étroite. Par conséquent, en réception l'égalisation de la i-ième sous porteuse se réduit à une division par le coefficient ξ_{OFDM} optimisé sur la base du critère MMSE comme indiqué par l'équation 3.29.

$$\xi_{\text{OFDM}}[i] = \frac{H^*[i]}{\|H[i]\|^2 + q\sigma^2} \quad i = 0, 1, \cdots, q-1 \tag{3.29}$$

Contrairement au système SC-MMSE, l'égalisation dans le cas du système OFDM génère directement la version temporelle du signal égalisé et ne nécessite pas une IFFT à la sortie. Par conséquent, l'égalisation et le calcul de la corrélation croisée ne peuvent pas être fusionnées car il est nécessaire d'insérer une FFT à la sortie de l'égaliseur pour pouvoir effectuer la corrélation croisée dans le domaine fréquentielle. Formellement, la sortie du circuit d'égalisation est donnée par :

$$\phi_{\text{OFDM}}[i] = \xi_{\text{OFDM}}[i] Z[i] \quad i = 0, 1, \cdots, q-1 \tag{3.30}$$

Si Φ_{OFDM} désigne la FFT d'ordre q de ϕ_{OFDM}, la sortie du circuit de calcul de la corrélation croisée s'obtient par :

$$\Psi_{\text{OFDM}}[i] = \hat{PN}[i] \Phi_{\text{OFDM}}[i] \quad i = 0, 1, \cdots, q-1 \tag{3.31}$$

Enfin, si ψ_{OFDM} désigne la IFFT d'ordre q de Ψ_{OFDM}, les LLRs se calculent par :

$$LLR(\beta_i) = \text{Re}\big(\psi_{\text{OFDM}}[i]\big) - \text{Re}\big(\psi_{\text{OFDM}}[0]\big) \quad i = 0, 1, \cdots, q-1 \tag{3.32}$$

Le tableau 3.3 résume le nombre de FFT et IFFT utilisées par les récepteurs SC-ML, SC-MMSE et OFDM en supposant que la FFT de la séquence fondamentale est déterminée à l'avance et par conséquent peut être stockée dans une mémoire. Nous constatons que l'association d'un code LDPC non binaire et d'une modulation CCSK dans un système de transmission mono-porteuse permet de réduire la complexité du récepteur ML qui possède

désormais la même complexité que le récepteur MMSE. Cependant, le récepteur OFDM nécessite une FFT supplémentaire et possède ainsi une plus grande complexité.

TABLE 3.3 – Complexité des récepteurs SC-ML, SC-MMSE et OFDM

	SC-ML	SC-MMSE	OFDM
Nombre de FFT	2	2	3
Nombre d'IFFT	1	1	1

Simulation des performances de l'association d'un code LDPC non binaire et d'une modulation CCSK dans un environnement intérieur

Les simulations sont faites par le modèle exponentiel d'un canal à trajets multiples en environnement intérieur (indoor channel model en anglais) [78]. Dans ce modèle, la puissance moyenne du canal décroit exponentiellement. Autrement dit, les coefficients de la réponse impulsionnelle du canal suivent un profil de puissance moyenne des retards (Power Delay Profile ou PDP en anglais) qui décroît exponentiellement. Chaque coefficient de la réponse impulsionnelle est multiplié par un évanouissement de Rayleigh indépendant. Dans nos simulations nous considérons le cas d'un canal avec 11 trajets dont le profil de puissance moyenne des retards est donné par la figure 3.13. La puissance moyenne du trajet direct est normalisée de façon à obtenir en réception une puissance moyenne totale égale à l'unité. La taille du préfixe cyclique est égale à $n_{cp} = 11$ chips afin de supprimer complètement l'interférence inter-symbole.

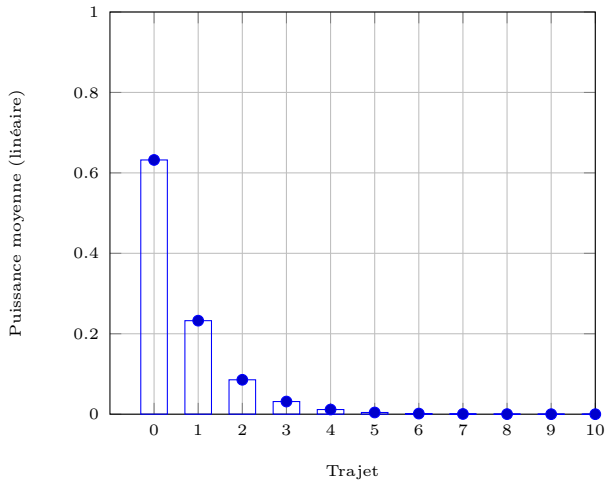


FIGURE 3.13 – Profil de la puissance moyenne des retards

La figure 3.14 illustre les performances des systèmes SC-ML, SC-MMSE et OFDM-MMSE en considérant un code $\mathbb{GF}(64)$-LDPC associé à une modulation CCSK de même ordre. Nous constatons en premier que dans ces conditions la transmission mono-porteuse avec préfixe cyclique permet d'obtenir des performances quasi-identiques à celles de la transmission OFDM. Nous constatons également que l'égalisation MMSE est optimale dans le sens où elle permet d'obtenir les mêmes performances de la démodulation ML. En fait, nous pouvons identifier une certaine symétrie entre ces deux techniques puisque d'une part la démodulation ML procède en faisant subir la distorsion du canal à la séquence fondamentale de la modulation CCSK et d'autre part l'égalisation MMSE procède en enlevant cette distorsion du signal reçu.

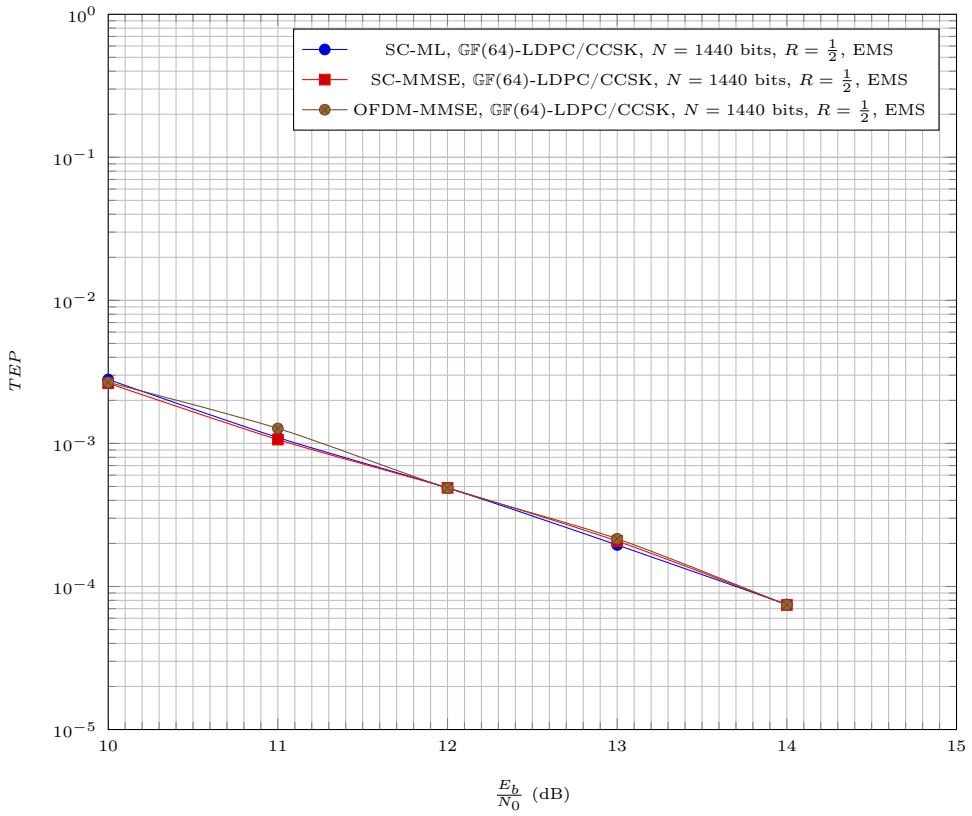


FIGURE 3.14 – Performance des systèmes SC-ML, SC-MMSE et OFDM-MMSE dans un canal sans fil en environnement intérieur

La figure 3.15 illustre les performances des modulations CCSK et 64-OM associées à un code $\mathbb{GF}(64)$-LDPC. Nous constatons que dans le cas d'un système OFDM-MMSE, les deux types de modulations possèdent les mêmes performances. Par contre, dans un système SC-MMSE la modulation CCSK réalise à TEP $= 10^{-4}$ un gain d'environ 2 dB par rapport à la modulation 64-OM. Une méthode formelle pour justifier ce résultat surprenant consiste à calculer pour chaque modulation l'entropie des LLRs intrinsèques afin de mesurer l'information fournie au décodeur. Toutefois, ce calcul fait apparaître des équations très complexes à résoudre. Pour simplifier la tâche, nous proposons de contourner le problème en étudiant statistiquement la dynamique des LLRs intrinsèques. Nous menons donc l'expérience décrite par l'algorithme 3.1. Le nombre de valeurs δ négatives permet de mesurer le taux d'erreur symbole à la sortie des démodulateurs. Ainsi, pour améliorer les performances de décodage, il faut minimiser ce nombre. La figure 3.16 illustre les histogrammes obtenus par l'algorithme 3.1 en considérant le même canal de la figure 3.15 et en choisissant un rapport signal à bruit égal à 15 dB. Nous constatons que l'histogramme de la modulation CCSK possède la plus petite surface négative ce qui explique le gain réalisé en sa faveur. Nous pensons que la robustesse des séquences CCSK provient de leur aspect aléatoire comparées aux séquences de Hadamard où le séquencement des 1 et des -1 est produit d'une façon structurée. Pour valider cette hypothèse, nous considérons un canal constant à deux trajets en opposition de phase modélisé par la réponse impulsionnelle $h = [1, -1]$. Nous calculons ainsi la distance Euclidienne minimale de chaque modulation après lui avoir appliqué la convolution circulaire de canal. Le tableau 3.4 liste les valeurs trouvées pour différents ordres de modulation. Les polynômes primitifs utilisés pour générer les séquences CCSK sont X^3+X^2+1, X^4+X^3+1, X^5+X^3+1 et $X^6+X^5+X^4+X+1$. Nous constatons que la distance Euclidienne minimale reste constante dans le cas de la

modulation orthogonale et augmente avec l'ordre de la modulation CCSK. Il en découle que contrairement à la modulation orthogonale, la résistance de la modulation CCSK à la distorsion du canal augmente avec la taille des séquences. Cette propriété de la modulation CCSK permet d'améliorer considérablement la dynamique des LLRs intrinsèques comme le montrent les histogrammes de la figures 3.17 tracés avec des séquences de taille 64 chips.

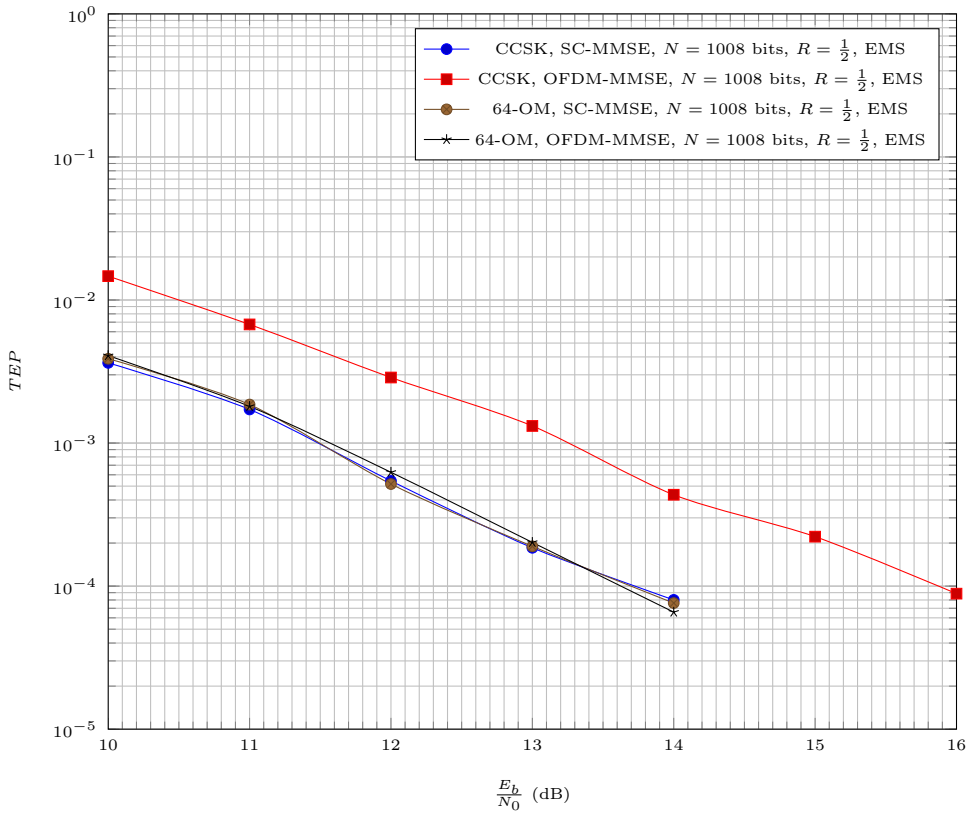


FIGURE 3.15 – Performances des modulations CCSK et 64-OM associées à un code $\mathbb{GF}(64)$-LDPC dans un canal sans fil en environnement intérieur

Algorithme 3.1 – Etude statistique de la dynamique des LLRs intrinsèques obtenus par démodulation CCSK ou 64-OM

$\Delta = [\,]$;
pour $i = 0$ ***à*** 63 **faire**
  pour $j = 1$ ***à*** 10000 **faire**
    Générer la séquence CCSK (ou bien la séquence 64-OM) associée au symbole β_i ;
    Générer la forme d'onde par modulation BPSK ;
    Ajouter un préfixe cyclique ;
    Transmettre le signal obtenu sur une réalisation aléatoire du canal ;
    Supprimer le préfixe cyclique du signal reçu ;
    Faire une égalisation MMSE ;
    Calculer les fiabilités intrinsèques $\mathrm{LLR}(\beta_k)$, $k = 0, 1, \cdots, 63$;
    Calculer $\delta = \mathrm{LLR}(\beta_i) - \max\limits_{\beta_k \neq \beta_i} \{\mathrm{LLR}(\beta_k)\}$;
    Mettre à jour $\Delta = \Delta \cup \delta$;
  fin
fin
Tracer l'histogramme de Δ ;

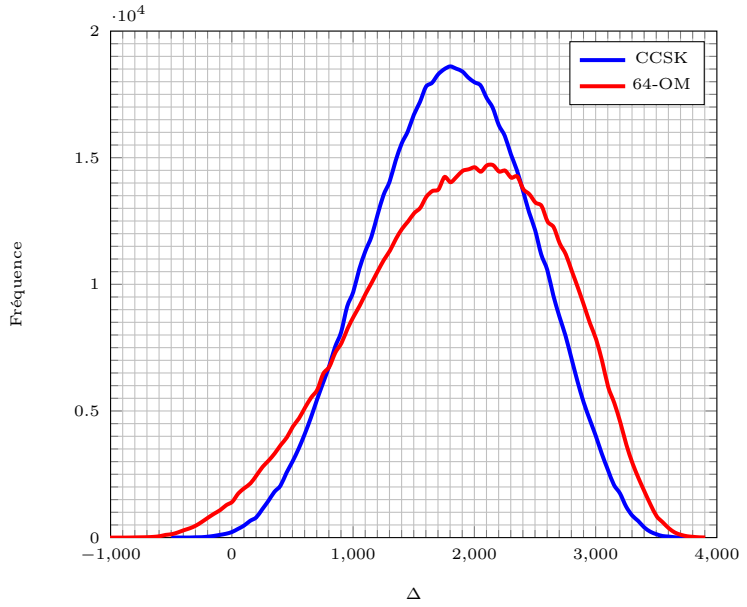


FIGURE 3.16 – Histogrammes de la démodulation CCSK et 64-OM dans un canal à 11 trajets avec $\frac{E_b}{N_0} = 15$ dB

TABLE 3.4 – Effet de la distorsion d'un canal à deux trajets en opposition de phase sur la distance Euclidienne minimale des modulations CCSK et k-ary OM

Ordre de la modulation	CCSK	Hadamard
8	4	2.28
16	5.65	2.28
32	9.38	2.28
64	14.42	2.28

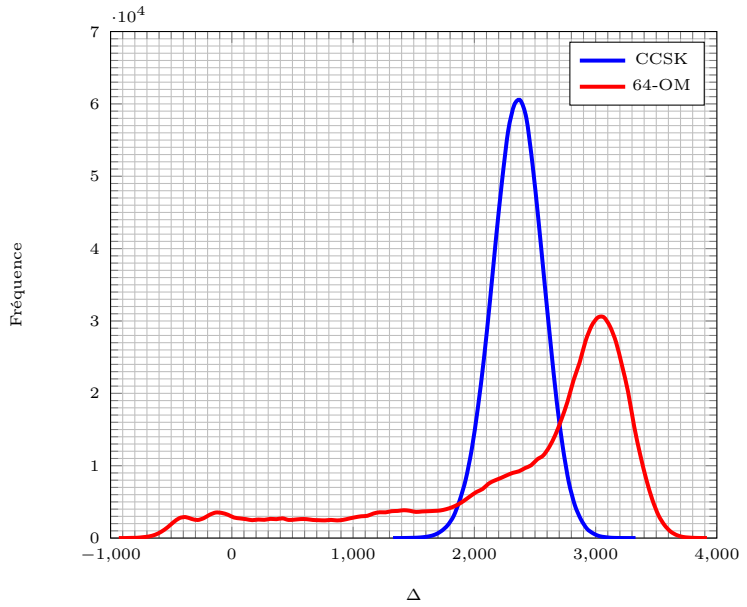


FIGURE 3.17 – Histogrammes de la démodulation CCSK et 64-OM dans un canal à 2 trajets en opposition de phase avec $\frac{E_b}{N_0} = 15$ dB

La figure 3.18 montre les performances d'un code $\mathbb{GF}(64)$-LDPC et d'un code $\mathbb{GF}(2)$-LDPC associés à une modulation CCSK d'ordre 64 dans un système de transmission mono-porteuse avec préfixe cyclique. Nous constatons d'abord que l'utilisation d'un décodage itérative BICM-ID permet d'améliorer les performances du code binaire par environ 1 dB à $TEP = 10^{-4}$. Toutefois, les performances du code $\mathbb{GF}(64)$-LDPC restent nettement meilleures avec un gain d'environ 1.5 dB à $TEP = 10^{-4}$.

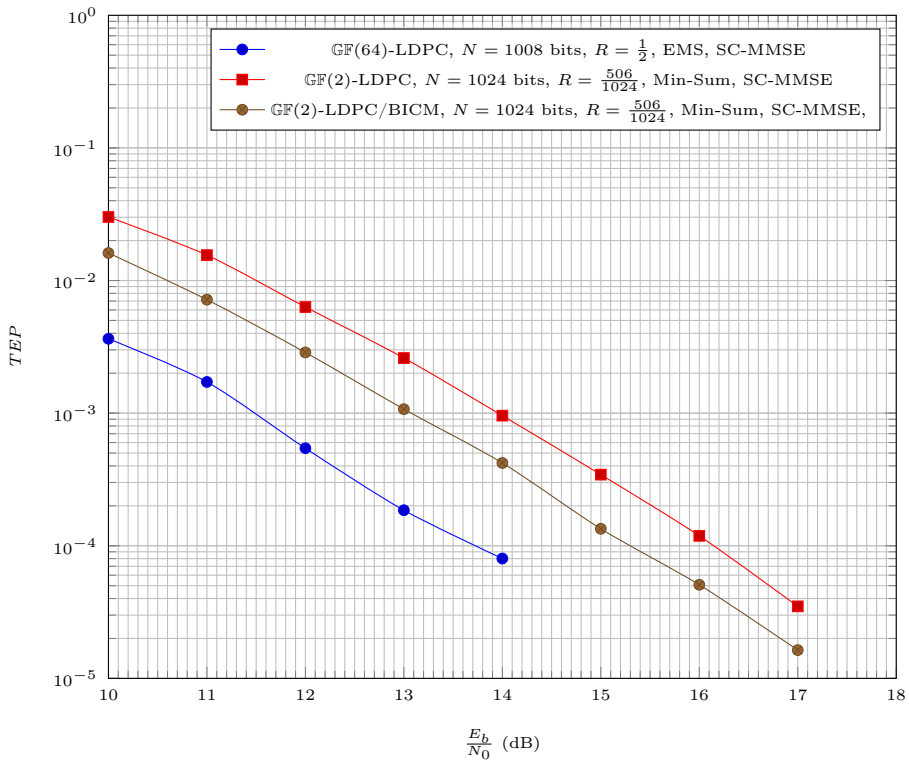


FIGURE 3.18 – Comparaison des performances d'un code $\mathbb{GF}(64)$-LDPC et d'un code $\mathbb{GF}(2)$-LDPC associés à une modulation CCSK d'ordre 64 dans un canal sans fil en environnement intérieur

3.5 Conclusion

Dans ce chapitre, nous avons montré que l'association d'un code LDPC non binaire et d'une modulation CCSK de même ordre permet de réduire la complexité de la démodulation tout en améliorant les performances de décodage. En effet, les propriétés de la modulation CCSK permettent de générer efficacement les LLRs intrinsèques par des opérations de corrélation. De plus, les décodeurs non binaires ont la capacité de profiter pleinement de la diversité apportée par la modulation CCSK. Par contre, les décodeurs binaires souffrent d'une perte d'information due à la marginalisation des probabilités des symboles. En considérant le cas d'une transmission dans un canal de Rayleigh, nous avons proposé d'implanter la démodulation CCSK par de simples opérations de FFT et FFT inverse. Les simulations sur ce canal ont montré que la modulation CCSK donnent des performances comparables à la modulation orthogonale. En considérant le cas d'un canal sélectif en fréquence, nous avons montré qu'il est possible de réduire la complexité d'un système SC-FDE en fusionnant les fonctions d'égalisation et de démodulation CCSK. De plus, les simulations ont montré que l'association d'un code LDPC non binaire et d'une

modulation CCSK donnent les mêmes performances dans les systèmes SC-FDE et OFDM. Enfin, nous avons montré que les codes LDPC non binaires sont largement plus efficaces en les associant à la modulation CCSK que les codes LDPC binaires même en ayant recours à une démodulation itérative pour compenser la perte d'information.

Conclusion et perspectives

Sommaire

3.1 Décodeur EMS

Dans la première partie du manuscrit, nous avons proposé une architecture d'un décodeur $\mathbb{GF}(q = 2^m)$-LDPC basé sur l'algorithme de décodage EMS. Nous avons choisi cet algorithme en raison de sa faible complexité en le comparant à l'algorithme BP. Toutefois, un choix judicieux de la taille n_m des messages tronqués permet de garder des performances proches du décodeur optimal. Dans nos travaux, nous avons considéré une ancienne architecture développée par notre laboratoire comme point de départ.

Nous avons proposé dans un premier temps de réduire à $n_\alpha \ll n_m$ le nombre des valeurs intrinsèques utilisées pour la mise à jour des messages par les nœuds de variable. Cette modification permet à la fois de réduire la taille de la mémoire dédiée au stockage des LLRs intrinsèques et de réduire la latence des nœuds de variable à $n_\alpha + n_m$ cycles d'horloges au lieu de $2n_m$ cycles. Pour l'étape de décision, nous considérons aussi un nombre limité $n_s \ll n_m$ de valeurs du premier message extrinsèque généré par le noeud de variable lors de la dernière itération de décodage. Cette approche permet de réduire la taille physique de la mémoire CAM connue pour son coût de fabrication élevé. Enfin, nous avons proposé une nouvelle architecture du trieur plus adaptée au fonctionnement de notre nœud de variable.

Nous avons proposé dans un second temps une nouvelle architecture pour les nœuds de parité élémentaires basée sur l'algorithme L-Bubble Check. Toutefois, cet algorithme présente un parcours sous forme de « L » qui ajoute de la complexité inutile. Par conséquent, nous avons proposé de modifier l'algorithme L-Bubble Check en définissant quatre parcours d'exploration directs. Les candidats de chaque parcours sont empilés dans une mémoire FIFO. Un circuit de comparaison permet ensuite de trouver le candidat le plus fiable parmi les sorties des quatre mémoires FIFO. Cette approche permet de séparer les processus de génération des candidats et de comparaison dans le but de couper le chemin critique du circuit et augmenter ainsi sa fréquence de fonctionnement.

Les résultats de synthèse d'un prototype FPGA montrent que notre décodeur est sept fois plus efficace que le prototype du décodeur développé dans le cadre du projet DAVINCI.

De plus, notre décodeur est plus efficace que les prototypes FPGA des décodeurs que nous avons réussi à trouver dans l'état de l'art.

Travaux futurs

L'architecture du VNP que nous proposons présente une dépendance à la cardinalité du corps de Galois due au module *Flag* utilisé pour détecter les symboles en double. Pour s'affranchir de cette dépendance, il est possible de modifier l'architecture du VNP en associant le module *Flag* à la mémoire associative CAM (c'est-à-dire en associant chaque bit du registre *Flag* à un élément de la mémoire CAM). Par conséquent, il faut adapter la mémoire CAM pour sauvegarder aussi les symboles du message $I.GF$ en mode de mise à jour des VNs. Ainsi, elle peut servir à détecter les symboles en commun des messages $I.GF$ et $M_{p_l v}$. La CAM prend alors la taille $(n_b + m) \times n_\alpha$ bits au lieu de $(n_b + m) \times n_s$ bits. En contrepartie, nous réduisons la taille du registre *Flag* à n_α bits. Une autre évolution possible de notre décodeur consiste à adapter l'algorithme S-Bubble Check pour développer des CNPs basés sur l'algorithme Min-Max.

Nous avons consacré une partie de la thèse à implanter notre décodeur sur cible FPGA. Cependant, il est difficile avec un tel prototype de se comparer aux décodeurs de l'état de l'art. En effet, les résultats de synthèse sur un FPGA dépendent de ses ressources et par conséquent diffèrent d'une famille à l'autre. De plus, la plupart des architectures de l'état de l'art sont prototypés sur des cibles ASIC qui permettent à la fois de déterminer avec précision la surface des décodeurs et leur fréquence de fonctionnement. D'autre part, nos résultats ont été obtenus en considérant uniquement le cas d'un corps de Galois d'ordre $q = 64$. Toutefois, la complexité des algorithmes de décodage est fonction de l'ordre du corps de Galois. Ainsi, nous proposons de poursuivre nos travaux en développant des prototypes ASIC pour les corps de Galois d'ordres 32, 64 et 256 fréquemment utilisés dans l'état de l'art. Il serait également possible de concevoir une architecture parallèle de notre décodeur et de la comparer avec les architectures parallèles existantes.

3.2 Association d'un code LDPC non binaire et d'une modulation CCSK

Dans la deuxième partie du manuscrit, nous avons étudié les performances de l'association d'un code LDPC non binaire et d'une modulation CCSK. Cette approche permet de conserver l'information intrinsèque du canal contrairement à l'association de la modulation CCSK avec un code binaire qui oblige à une étape de marginalisation pour passer des LLRs des symboles aux LLRs des bits. De plus, nous avons pris la décision d'étudier la modulation CCSK pour ses propriétés qui permettent de réduire la complexité du démodulateur.

En considérant dans un premier temps le cas d'une transmission sur un canal de Rayleigh, nous avons montré qu'il est possible d'effectuer la démodulation par des opérations de FFT et FFT inverse. Les résultats de simulations ont montré que la modulation CCSK donne des performances équivalentes à la modulation de Hadamard. D'autre part, nous constatons que la modulation CCSK est beaucoup plus efficace avec les codes non binaires même en utilisant une démodulation itérative pour les codes binaires.

En considérant ensuite le cas d'une transmission sur un canal sélectif en fréquence, nous avons montré que dans un système de transmission mono-porteuse avec préfixe cyclique la démodulation optimale ML peut aussi s'effectuer par des opérations de FFT et FFT inverse. En considérant une étape d'égalisation, nous avons montré qu'il est possible de fusionner le démodulateur CCSK et l'égaliseur dans un seul bloc. Les simulations obtenues avec le modèle exponentiel d'un canal indoor indiquent que les performances de l'égalisation MMSE sont identiques à la démodulation ML. De plus, nous constatons que les performances d'une transmission mono-porteuse avec préfixe cyclique sont comparables à celles d'une transmission OFDM. Finalement, les simulations indiquent que la modulation CCSK est plus performante que la modulation de Hadamard dans le cas d'une transmission mono-porteuse.

Travaux futurs

Une étude récente a déjà montré l'intérêt de considérer la modulation CCSK pour améliorer l'efficacité spectrale des systèmes de navigation par satellite. Toutefois, il est possible d'étendre cette étude en prenant compte du codage de canal. D'autre part, nous trouvons que les performances de l'association de la modulation CCSK et des codes LDPC non binaires présentent un intérêt pour les systèmes de communication simplex où l'on cherche à réduire la puissance d'émission indépendamment de la complexité du récepteur. Par exemple, une piste intéressante à suivre est celle des réseaux de capteurs. Enfin, nos simulations ont été réalisées par des modèles simples de canaux sans fil. Pour confirmer nos résultats, il est nécessaire de considérer d'autres modèles plus élaborés tels que les modèles des canaux LTE.

Publications de l'auteur relatives aux travaux de la thèse

Nos contributions sont publiées dans deux conférences internationales et ont fait l'objet de deux dépôts de brevet :

- O. Abassi, L. Conde-Canencia, M. Mansour, and E. Boutillon. Non-Binary Low-Density Parity-Check coded cyclic Code-Shift keying. in *2013 IEEE Wireless Communications and Networking Conference (WCNC) : PHY (IEEE WCNC 2013 - PHY)*, Shanghai, P.R. China, Apr. 2013, pp.2544–2548
- O. Abassi, L. Conde-Canencia, M. Mansour, and E. Boutillon. Non-Binary coded CCSK and Frequency-Domain equalization with simplified LLR generation. in *2013 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications : Fundamentals and PHY Track (PIMRC'13 - Fundamentals and PHY Track)*, London, United Kingdom, Sep. 2013.
- E. Boutillon, O. Abassi ; L. Conde-Canencia. Procédé de transmission de mots de code correcteur d'erreur non binaire avec modulation CCSK, signal et dispositif correspondant. Brevet FR1251334, Feb. 2012
- E. Boutillon, O. Abassi ; L. Conde-Canencia. Architecture pour l'algorithme Bubble Check. Brevet FR 1450086, Jan. 2014

Une publication sur l'architecture de notre décodeur est à soumettre.

Annexe A

Le corps de Galois $\mathbb{GF}(q = 2^6)$

TABLE A.1 – Le corps de Galois $\mathbb{GF}(q = 2^6)$ construit par $p(X) = 1 + X + X^6$

Représentation exponentielle	Représentation polynomiale	Représentation binaire
0	0	000000
1	1	000001
α	α	000010
α^2	α^2	000100
α^3	α^3	001000
α^4	α^4	010000
α^5	α^5	100000
α^6	$\alpha + 1$	000011
α^7	$\alpha^2 + \alpha + 1$	000110
α^8	$\alpha^3 + \alpha^2$	001100
α^9	$\alpha^4 + \alpha^3$	011000
α^{10}	$\alpha^5 + \alpha^4$	110000
α^{11}	$\alpha^5 + \alpha + 1$	100011
α^{12}	$\alpha^2 + 1$	000101
α^{13}	$\alpha^3 + \alpha$	001010
α^{14}	$\alpha^4 + \alpha^2$	010100
α^{15}	$\alpha^5 + \alpha^3$	101000
α^{16}	$\alpha^4 + \alpha + 1$	010011
α^{17}	$\alpha^5 + \alpha^2 + \alpha$	100110
α^{18}	$\alpha^3 + \alpha^2 + \alpha + 1$	001111
α^{19}	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha$	011110
α^{20}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$	111100
α^{21}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$	111011
α^{22}	$\alpha^5 + \alpha^4 + \alpha^2 + 1$	110101
α^{23}	$\alpha^5 + \alpha^3 + 1$	101001
α^{24}	$\alpha^4 + 1$	010001
α^{25}	$\alpha^5 + \alpha$	100010
α^{26}	$\alpha^2 + \alpha + 1$	000111
α^{27}	$\alpha^3 + \alpha^2 + \alpha$	001110
α^{28}	$\alpha^3 + \alpha^2 + \alpha$	001110
α^{29}	$\alpha^5 + \alpha^4 + \alpha^3 +$	111000
α^{30}	$\alpha^5 + \alpha^4 + \alpha + 1$	110011
α^{31}	$\alpha^5 + \alpha^2 + 1$	100101
α^{32}	$\alpha^3 + 1$	001001
α^{33}	$\alpha^4 + \alpha$	010010
α^{34}	$\alpha^5 + \alpha^2$	100100
α^{35}	$\alpha^3 + \alpha + 1$	001011
α^{36}	$\alpha^4 + \alpha^3 + \alpha$	011010
α^{37}	$\alpha^3 + \alpha^2 + 1$	001101
α^{38}	$\alpha^4 + \alpha^3 + \alpha + 1$	011011
α^{39}	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha$	110110
α^{40}	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$	101111
α^{41}	$\alpha^4 + \alpha^3 + \alpha^2 + 1$	011101
α^{42}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha$	111010
α^{43}	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$	110111
α^{44}	$\alpha^5 + \alpha^3 + \alpha^2 + 1$	101101
α^{45}	$\alpha^4 + \alpha^3 + 1$	011001
α^{46}	$\alpha^5 + \alpha^4 + \alpha$	110010
α^{47}	$\alpha^5 + \alpha^2 + \alpha + 1$	100111
α^{48}	$\alpha^3 + \alpha^2 + 1$	001101
α^{49}	$\alpha^4 + \alpha^3 + \alpha$	011010
α^{50}	$\alpha^5 + \alpha^4 + \alpha^2$	110100
α^{51}	$\alpha^5 + \alpha^3 + \alpha + 1$	101011
α^{52}	$\alpha^4 + \alpha^2 + 1$	010101
α^{53}	$\alpha^5 + \alpha^3 + \alpha$	101010
α^{54}	$\alpha^4 + \alpha^2 + \alpha + 1$	010111
α^{55}	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha$	101110
α^{56}	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$	011111
α^{57}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$	111110
α^{58}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$	111111
α^{59}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$	111101
α^{60}	$\alpha^5 + \alpha^4 + \alpha^3 + 1$	111001
α^{61}	$\alpha^5 + \alpha^4 + 1$	110001
α^{62}	$\alpha^5 + 1$	100001

Bibliographie

[1] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard. Low-complexity, low-memory EMS algorithm for non-binary ldpc codes. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 671–676, 2007.

[2] E. Boutillon, L. Conde-Canencia, and A. Al Ghouwayel. Design of a GF(64)-LDPC decoder based on the EMS algorithm. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 60(10) :2644–2656, 2013.

[3] Yaoyu Tao, Youn Sung Park, and Zhengya Zhang. High-throughput architecture and implementation of regular (2, dc) nonbinary LDPC decoders. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 2625–2628, May 2012.

[4] The Digital Video Broadcasting Project. `http://www.dvb.org/`.

[5] The 3rd Generation Partnership Project. `http://www.3gpp.org/`.

[6] Jeffrey G Andrews, Arunabha Ghosh, and Rias Muhamed. *Fundamentals of WiMAX : understanding broadband wireless networking*. Prentice Hall, Upper Saddle River, NJ, 2007.

[7] Claude Elwood Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1964.

[8] Shu Lin. *Error control coding : fundamentals and applications*. Pearson-Prentice Hall, Upper Saddle River, N.J, 2004.

[9] R. G. Gallager. *Low-density parity-check codes*. PhD thesis, MIT, Cambridge, Mass., September 1960.

[10] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electron. Lett.*, 32(18) :1645–1646, August 1996.

[11] D. J C MacKay. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2) :399–431, 1999.

[12] Sae-Young Chung, Jr. Forney, G.D., T.J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *Communications Letters, IEEE*, 5(2) :58–60, 2001.

[13] M.C. Davey and D. MacKay. Low-density parity check codes over $GF(q)$. *Communications Letters, IEEE*, 2(6) :165–167, 1998.

[14] Rudolf Lidl and Harald Niederreiter. *Finite fields*. Cambridge University Press, Cambridge, 2008.

[15] Jean-Pierre Deschamps and Gustavo D Sutter. *Hardware implementation of finite-field arithmetic*. McGraw-Hill, New York, 2009.

[16] John M. Howie. *Fields and Galois Theory (Springer Undergraduate Mathematics Series)*. Springer, 2007.

[17] David Forney. Introduction to finite fields. `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-451-principles-of-digital-communication-ii-spring-2005/lecture-notes/chap7.pdf`.

[18] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding : Turbo-codes. 1. In *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064–1070 vol.2, 1993.

[19] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423 and 623–656, 1948.

[20] Jorge Moreira. *Essentials of error-control coding.* John Wiley & Sons, West Sussex, England, 2006.

[21] R.M. Tanner. A recursive approach to low complexity codes. *Information Theory, IEEE Transactions on*, 27(5) :533–547, 1981.

[22] Hongzin Song and J.R. Cruz. Reduced-complexity decoding of q-ary LDPC codes for magnetic recording. *Magnetics, IEEE Transactions on*, 39(2) :1081–1087, 2003.

[23] L. Barnault and D. Declercq. Fast decoding algorithm for LDPC over $GF(2^q)$. In *Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, pages 70–73, 2003.

[24] H. Wymeersch, H. Steendam, and M. Moeneclaey. Log-domain decoding of LDPC codes over $GF(q)$. In *Communications, 2004 IEEE International Conference on*, volume 2, pages 772–776 Vol.2, 2004.

[25] D. Declercq and M. Fossorier. Decoding Algorithms for Nonbinary LDPC Codes Over $GF(q)$. *Communications, IEEE Transactions on*, 55(4) :633–643, 2007.

[26] L. Conde-Canencia, E. Boutillon, and A. Al-Ghouwayel. Complexity comparison of non-binary LDPC decoders. In *proceedings of ICT Mobile Summit*, Spain, June 2009.

[27] V. Savin. Min-Max decoding for non binary LDPC codes. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 960–964, 2008.

[28] Erbao Li, K. Gunnam, and D. Declercq. Trellis based Extended Min-Sum for decoding nonbinary LDPC codes. In *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, pages 46–50, Nov 2011.

[29] Erbao Li, D. Declercq, and K. Gunnam. Trellis-Based Extended Min-Sum Algorithm for Non-Binary LDPC Codes and its Hardware Structure. *Communications, IEEE Transactions on*, 61(7) :2600–2611, July 2013.

[30] Design And Versatile Implementation of Non-binary wireless Communications based on Innovative LDPC codes. `http://www.ict-davinci-codes.eu/`.

[31] A. Mourad and I. Gutierrez. System level evaluation of DAVINCI non-binary LDPC codes. In *Future Network and Mobile Summit, 2010*, pages 1–9, 2010.

[32] I. Gutierrez, G. Bacci, J. Bas, A. Bourdoux, H. Gierszal, A. Mourad, and S. Pleftschinger. DAVINCI non-binary LDPC codes : Performance and complexity assessment. In *Future Network and Mobile Summit, 2010*, pages 1–8, 2010.

[33] W. Chen, C. Poulliat, D. Declercq, L. Conde-Canencia, A. Al-Ghouwayel, and E. Boutillon. Non-binary LDPC codes defined over the general linear group : Finite length design and practical implementation issues. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5, 2009.

[34] M.M. Mansour and N.R. Shanbhag. Low-power VLSI decoder architectures for LDPC codes. In *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, pages 284–289, 2002.

[35] M.M. Mansour and N.R. Shanbhag. High-throughput LDPC decoders. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(6) :976–996, 2003.

[36] C. Chavet and P. Coussy. A memory mapping approach for parallel interleaver design with multiples read and write accesses. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3168–3171, 2010.

[37] A.A. Ghouwayel and E. Boutillon. A systolic LLR generation architecture for non-binary LDPC decoders. *Communications Letters, IEEE*, 15(8) :851–853, 2011.

[38] E. Boutillon and L. Conde-Canencia. Bubble check : a simplified algorithm for elementary check node processing in extended min-sum non-binary ldpc decoders. *Electronics Letters*, 46(9) :633–634, 2010.

[39] E. Boutillon and L. Conde-Canencia. Simplified check node processing in nonbinary ldpc decoders. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 201–205, 2010.

[40] Emmanuel Boutillon and Laura Conde-Canencia. Procédé de commande d'une unité de calcul, tel qu'un nœud de parité élémentaire dans un décodeur de code LDPC non binaire, et unité de calcul correspondante. patent no. FR0952988, May 2009.

[41] M. Awais, A. Singh, E. Boutillon, and G. Masera. A novel architecture for scalable, high throughput, multi-standard LDPC decoder. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 340–347, Aug 2011.

[42] Zynq Evaluation and Development Board. `http://www.zedboard.org/`.

[43] Xinmiao Zhang and Fang Cai. Partial-parallel decoder architecture for quasi-cyclic non-binary LDPC codes. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 1506–1509, March 2010.

[44] Xinmiao Zhang and Fang Cai. An efficient architecture for iterative soft reliability-based majority-logic non-binary LDPC decoding. In *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pages 885–888, Nov 2011.

[45] Xinmiao Zhang and Fang Cai. Reduced-complexity decoder architecture for non-binary LDPC codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(7) :1229–1238, July 2011.

[46] Weiguo Tang, Jie Huang, Lei Wang, and Shengli Zhou. A Nonbinary LDPC Decoder Architecture With Adaptive Message Control. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(11) :2118–2122, Nov 2012.

[47] Yeong-Luh Ueng, Kuo-Hsuan Liao, Hsueh-Chih Chou, and Chung-Jay Yang. A High-Throughput Trellis-Based Layered Decoding Architecture for Non-Binary LDPC Codes Using Max-Log-QSPA. *Signal Processing, IEEE Transactions on*, 61(11) :2940–2951, June 2013.

[48] J. Lin and Z. Yan. An Efficient Fully Parallel Decoder Architecture for Nonbinary LDPC Codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99) :1–1, 2013.

[49] F. Garcia-Herrero, M.J. Canet, and J. Valls. High-speed NB-LDPC decoder for wireless applications. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, pages 215–220, Nov 2013.

[50] Fang Cai and Xinmiao Zhang. Relaxed min-max decoder architectures for nonbinary low-density parity-check codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(11) :2010–2023, Nov 2013.

[51] F. Garcia-Herrero, M.J. Canet, and J. Valls. Nonbinary LDPC Decoder Based on Simplified Enhanced Generalized Bit-Flipping Algorithm. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99) :1–1, 2013.

[52] Jun Lin and Zhiyuan Yan. Efficient Shuffled Decoder Architecture for Nonbinary Quasi-Cyclic LDPC Codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(9) :1756–1761, Sept 2013.

[53] F. Garcia-Herrero, E. Li, D. Declercq, and J. Valls. Multiple-Vote Symbol-Flipping Decoder for Nonbinary LDPC Codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99) :1–1, 2014.

[54] Youn Sung Park, Yaoyu Tao, and Zhengya Zhang. A 1.15Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, pages 422–423, Feb 2013.

[55] Xiaoheng Chen and Chung-Li Wang. High-Throughput Efficient Non-Binary LDPC Decoder Based on the Simplified Min-Sum Algorithm. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 59(11) :2784–2794, Nov 2012.

[56] C. Spagnol, E.M. Popovici, and W.P. Marnane. Hardware Implementation of $\mathbb{GF}(2^m)$ LDPC Decoders. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 56(12) :2609–2620, Dec 2009.

[57] Yue Sun, Yuyang Zhang, Jianhao Hu, and Zhongpei Zhang. FPGA implementation of nonbinary quasi-cyclic LDPC decoder based on EMS algorithm. In *Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on*, pages 1061–1065, July 2009.

[58] Xinmiao Zhang and Fang Cai. Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 58(2) :402–414, Feb 2011.

[59] D. Declercq, M. Colas, and G. Gelle. Regular $GF(2^q)$-LDPC coded modulations for higher order QAM-AWGN channel. In *Proc. ISITA*, Parma, Italy, October 2004.

[60] Yu zhen Huang, Yun peng Cheng, Yu ming Zhang, Guo hai Yu, and Jin Chen. Combine non-binary LDPC codes with m-ary orthogonal spread spectrum modulation. In *Wireless Communications and Signal Processing (WCSP), 2010 International Conference on*, pages 1 –4, oct. 2010.

[61] G. Ungerboeck. Channel coding with multilevel/phase signals. *Information Theory, IEEE Transactions on*, 28(1) :55–67, 1982.

[62] G. Ungerboeck. Trellis-coded modulation with redundant signal sets part i : Introduction. *Communications Magazine, IEEE*, 25(2) :5–11, 1987.

[63] G. Ungerboeck. Trellis-coded modulation with redundant signal sets part ii : State of the art. *Communications Magazine, IEEE*, 25(2) :12–21, 1987.

[64] S. Hamidreza Jamali and Tho Le-Ngoc. *Coded-modulation techniques for fading channels.* Kluwer, New York, 1994.

[65] E. Zehavi. 8-PSK trellis codes for a Rayleigh channel. *Communications, IEEE Transactions on*, 40(5) :873–884, 1992.

[66] G. Caire, Giorgio Taricco, and Ezio Biglieri. Bit-interleaved coded modulation. *Information Theory, IEEE Transactions on*, 44(3) :927–946, 1998.

[67] Xiaogdong Li and J.A. Ritcey. Bit-interleaved coded modulation with iterative decoding using soft feedback. *Electronics Letters*, 34(10) :942–943, 1998.

[68] Qualcomm Inc. *An Overview of the Application of Code Division Multiple Access (CDMA) to Digital Cellular Systems and Personal Cellular Networks*. May 1992.

[69] A. J. Viterbi. *CDMA : Principles of Spread Spectrum Communication*. Addison-Wesley, 1995.

[70] D.V. Sarwate and M.B. Pursley. Crosscorrelation properties of pseudorandom and related sequences. *Proceedings of the IEEE*, 68(5) :593–619, 1980.

[71] A. Y.-C. Wong and V. C. M. Leung. Code-phase-shift keying : a power and bandwidth efficient spread spectrum signalling technique for wireless local area network applications. In *Proc. IEEE Canadian Conf. Elect. Comput. Eng.*, volume 2, pages 478–481, May 1997.

[72] G. M. Dillard, M. Reuter, J. Zeidler, and B. Zeidler. Cyclic code shift keying : a low probability of intercept communication technique. *IEEE Tans. Aerosp. Electron. Syst.*, 39(3), July 2003.

[73] A. G. Peña, M.-L. Boucheret, C. Macabiau, J.-L. Damidaux, L. Ries, S. Corazza, and A.-C. Escher. Implementation of code shift keying signalling technique in galileo e1 signal. In *Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC), 2010 5th ESA Workshop on*, pages 1 –8, dec. 2010.

[74] Shu Lin, Michael R. Masse, Michael B. Pursley, Thomas C. Royster, and Shumei Song. Frequency-hop antijam communications with nonbinary error-control coding. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1 –7, oct. 2007.

[75] James Cavers. *Mobile Channel Characteristics (The Springer International Series in Engineering and Computer Science)*. Springer, 2013.

[76] Matthias Pätzold. *Mobile Fading Channels : Modelling, Analysis, & Simulation*. Wiley, 2002.

[77] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[78] J. Medbo and P. Schramm. Channel models for hiperlan/2. *ETSI/BRAN doc. No. 3ERI085B*, 1998.

[79] Xiao-Yu Hu, E. Eleftheriou, and D.-M. Arnold. Progressive edge-growth tanner graphs. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 2, pages 995 –1001 vol.2, 2001.

[80] D. J. C. MacKay. Source code for progressive edge growth parity check matrix construction. `http://www.inference.phy.cam.ac.uk/mackay/PEG\_ECC.html`.

[81] Sarah Johnson. *Iterative error correction : turbo, low-density parity-check and repeat-accumulate codes*. Cambridge University Press, Cambridge, UK New York, 2010.

[82] R. Craigen and H. Kharaghani. Hadamard matrices and hadamard designs. In C. J. Colbourn and J. H. Dinitz, editors, *Handbook of Combinatorial Designs*. Chapman & Hall/CRC, second edition, 2007.

[83] Emeric Maury. Theoretical performance limit evaluation. `http://departements.telecom-bretagne.eu/data/elec/turbo/LIMIT/`.

[84] John Proakis. *Digital communications.* McGraw-Hill, Boston, 2008.

[85] K. Abend and B.D. Fritchman. Statistical detection for communication channels with intersymbol interference. *Proceedings of the IEEE*, 58(5) :779–785, 1970.

[86] G.D. Forney. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *Information Theory, IEEE Transactions on*, 18(3) :363–378, 1972.

[87] John Barry. *Digital communication.* Kluwer Academic Publishers, Boston, 2004.

[88] D. Falconer, S.L. Ariyavisitakul, A. Benyamin-Seeyar, and B. Eidson. Frequency domain equalization for single-carrier broadband wireless systems. *Communications Magazine, IEEE*, 40(4) :58–66, 2002.

[89] F. Pancaldi, G.M. Vitetta, R. Kalbasi, N. Al-Dhahir, M. Uysal, and H. Mheidat. Single-carrier frequency domain equalization. *Signal Processing Magazine, IEEE*, 25(5) :37–56, 2008.