REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Électronique

thème :

Implémentation sur FPGA d'un décodeur LDPC pour les communications sans fils

Idris ACHOURI

Sous la direction de M. Mohamed TAGHI

Présenté et soutenue publiquement le (21/06/2018)

Composition du Jury :

Président	M .Boualem BOUSSEKSOU,	Cc	ENP
Promoteur	M .Mohamed Oussaïd TAGHI,	Mr	ENP
Examinateur	M .Lies SAADAOUI,	Dr	ENP

ENP 2018

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Électronique

thème :

Implémentation sur FPGA d'un décodeur LDPC pour les communications sans fils

Idris ACHOURI

Sous la direction de M. Mohamed TAGHI

Présenté et soutenue publiquement le (21/06/2018)

Composition du Jury :

Président	M .Boualem BOUSSEKSOU,	Pr	USTHB
Promoteur	M .Mohamed Oussaïd TAGHI,	Mr	ENP
Examinateur	M .NLies SAADAOUI,	Dr	ENP

ENP 2018

Dédicace

Ce travail est dédié à nos chers parents,

A nos professeurs,

A nos familles,

A nos frères et sœurs,

Et à nos amis,
A tous ceux que je n'ai pas cités et que je n'oublierai jamais leurs soutiens et leurs aides.

Remerciements

En premier lieu, nous remercions Dieu le tout puissant de nous avoir donné le courage, la volonté et la patience pour réaliser ce travail.

Nous remercions nos parents, qui nous ont soutenus tout au long de nos études.

Nos remerciements les plus vifs s'adressent particulièrement à Monsieur Mohamed TAGHI, notre promoteur à l'Ecole Nationale Polytechnique pour son aide, sa confiance et sa disponibilité, ainsi que les membres du jury d'avoir accepté l'examination de ce mémoire.

Nous remercions aussi tous nos amis qui nous ont soutenus dans les moments difficiles.

Ce travail, fruits de notre cursus et notre volonté, n'a été possible que grâce au tous nos enseignants dont nous louons les efforts qu'ils ont consentis durant toutes ces années.

ملخص

تمثل رموز اختيار التكافؤ منخفض الكثافة (LDPC) رموز مصححات الخطأ الأكثر فعالية لأنها تسمح بالحصول على نتائج قريبة من تلك المعرفة ب نهاية شانون (SHANON) للرموز ذات المجموعات الواسعة جدا فيما يخص جودة تصحيح الخطأ.هذه الدراسة خصصناها لتصميم تخطيط نصف متوازي مرن لجهاز فك الرموز بالاعتماد على نظام الحلول الحسابية (خوارزمية) لفك الترميزMin-Sum. تم تأكيد كفاءة هذه الخوارزمية عن طريق المحاكاة على جهاز الكمبيوتر. تخطيط جهاز فك الرموز تم دمجه في بطاقة FPGA بعد تقليص حجم رموز ال (LDPC) الخاص بهذا العمل بسبب متطلبات الاختبار. في الأخير تم ربط نتائج التركيب مع الأنماط البيانية المولدة من لغة وصف الأجهزة HDL الخاصة بهذا التخطيط.

كلمات مفتاحيه : اختيار التكافؤ منخفض الكثافة (LDPC) , جهاز فك الرموز HDL , بطاقة FPGA ,التخطيط ,لفك الترميز, .Min-Sum

Abstract :

Low-density parity-check (LDPC) are among the most powerful forward error correcting codes since they achieve error correction performance very close to the Shannon limit for large block lengths. LDPC block. In this thesis, we investigate into the design architecture of an array type LDPC code based on min-sum algorithm. The performance of the decoding algorithm was first validated via simulations. The detailed design of the decoding architecture was implemented on a field-programmable gate array (FPGA) kit with a short block length as an example. The schematics generated have been documented along with the synthesis results.

Key words : LDPC, FPGA, HDL, Min-Sum, Decoder, Architecture, Implementation.

Résumé :

Codes de contrôle de parité à faible densité(LDPC) codes font partie des codes correcteurs d'erreur les plus performant, puisque ils permettent d'atteindre une performance de correction d'erreur très proche de la limite de Shanon pour des codes en block très larges. Nous avons consacré notre travail à la conception d'une architecture semi parallèle, flexible d'un décodeur LDPC basée sur l' de décodage Min-sum. Les performances de cet algorithme de décodage ont été validé dans un premier temps par le biais d'une simulation. La conception de l'architecture du décodeur a été ensuite implémentée sur la carte FPGA après réduction de la taille du code LDPC considéré pour ce travail, à cause des exigences de test. Les schémas générés par la description HDL de cette architecture ont été associé aux résultats de synthèse.

Mots Clés : LDPC, FPGA, HDL, Min-Sum, Décoder, Architecture, Implémentation.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

APP A Posteriori Probability

AWGN Additive White Gaussian Noise

BCH Bose-Chaudhuri-Hocquenghem

BEC Binary Erasure Channel

BER Bit Error Rate

BPSK Binary Phase Shift Keying

BP Belief Propagation

BSC Binary Symmetric Channel

DVB-S2 2nd Generation Digital Video Broadcast Spatial

DVB-T Digital Video Broadcast Terrestrial

FER Frame Error Rate

GSM Groupe Signaux Multidimensionnels

i.i.d indépendant identiquement distribué

IEEE Institute of Electrical and Electronics Engineers

LDPC Low Density Parity Check

LLR Log Likelihood Ratio

MAP Maximum A Posteriori

NRNSC Non Recursive Non Systematic Code

NSC Non Systematic Code

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RSC Recursive Systematic Code

SNR Signal to Noise Ratio

SP Sum Product

WiFi Wireless Fidility

WIMAX Worldwide Interoperability for Microwave Access

Liste des notations

$sign(.)$ La fonction signe

$|\ .\ |$ La valeur absolue

$(.)^t$ La transposée d'un vecteur ou d'une matrice

$(.)^H$ L'opérateur Hermitien c.à.d le conjugué de la transposée d'une matrice

$(.)^*$ Le conjugué

$\oplus$ Somme modulo-2

$LLR(.)$ Le log-rapport de vraisemblance

$\prod$ La fonction d'entrelacement

$\prod^{-1}$ La fonction de dés-entrelacement

c Le vecteur de la séquence d'information

$M_{y/x}$ Matrice de transition d'un canal

C La capacité du canal

d_s Le débit symbole de la source

x Le vecteur du mot de code

K La taille de la séquence d'information

N La taille du mot de code

M Le nombre de bits de redondance ou de parité

R Rendement de codage de canal

F_q Corps de Galois de q éléments

d_H La distance de Hamming

w_H Le poids de Hamming

d_{min} La distance minimale

G Matrice génératrice de code

H Matrice de contrôle de parité

w_r Le poids des lignes de la matrice H

w_c Le poids des colonnes de la matrice H

E_b/N_0 Le rapport de l'énergie transmise par bit d'information sur la densité de puissance du bruit

E_s/N_0 Le rapport de l'énergie transmise par symbole d'information sur la densité de puissance du bruit

Introduction Générale

De nos jours, nous vivons dans un monde extrêmement numérique où les technologies des communications sans fils jouent un rôle primordial. Ainsi que dans ces deux dernières décennies, le monde a connu une énorme évolution notamment avec le développement et l'explosion de l'électronique moderne et l'apparition des nouvelles technologies et techniques avancées des systèmes de communications numériques. Cet extraordinaire essor que connaît le monde est principalement dû aux développements considérables dans le domaine de télécommunication.

Ce développement technologique a donné naissance à une forte expansion de la communication numérique. Ainsi, le codage du canal est devenu un élément essentiel et indispensable car il permet d'assurer la protection de l'information transmise en contrôlant les erreurs de transmission. Son principe est de découper le message à transmettre en blocs de k bits, qui sont alors traités séparément par le codeur. Les codes classiques ne permettent pas d'atteindre la limite de Shannon. On a donc développé d'autres systèmes de codages appelé codes convolutifs. Le principe de ces codes, inventés par Peter Elias en 1954, est de le considérer comme une séquence semi-infinie de symboles qui passent à travers une succession de registres à décalage, dont le nombre est appelé mémoire du code.

Les codes LDPC ont été découverts par Gallager au début des années 1960. Cette découverte remarquable a été largement ignorée par les chercheurs pendant près de 20 ans, jusqu'au travail de Tanner en 1981, dans lequel il a fourni une nouvelle interprétation des codes LDPC d'un point de vue graphique. Le travail de Tanner a également été ignoré par les théoriciens pendant environ 14 ans jusqu'à la fin des années 1990, lorsque certains chercheurs en codage ont commencé à étudier les codes graphiques et le décodage itératif. Leurs recherches ont mené à la redécouverte des codes de Gallager. Ils ont montré qu'un long code LDPC avec un décodage itératif basé sur la propagation de croyance permet une erreur de performance représentant seulement une fraction de décibel de la limite de Shannon [4] [6]. Cette découverte fait des codes LDPC de puissants concurrents par rapport aux codes turbo pour le contrôle des erreurs lorsqu'une haute fiabilité est requise. Les codes LDPC ont l'avantage des codes turbo, il ne nécessite pas un long entrelacement pour atteindre une bonne performance d'erreur. Ainsi, en 2004, un code LDPC a d'abord été standardisé dans une émission satellite DVB-S2.

En 2004, les codes LDPC ont été introduits pour la première fois dans la norme de

télédiffusion numérique par satellite(DVB-S2)[1], et à partir de cette année, les applications de ces codes ne cessent d'augmenter surtout dans les communications sans fils, en effet, ils ont été adopté quelques années plu tard par plusieurs standard récents, tels que le WiMAX WLAN802.16e[1], le G.hn/G.9960 pour les réseaux domestiques filaire[2], et le 10GBASE-T standard pour les 10Giga bit Ethernet(802.3an)[3]. L'intégration des techniques de codage dites avancées, telles que les codes LDPC se généralise donc dans les standards de communications. Dans ce contexte l'objectif de notre travail est d'étudier en largeur et en profondeur les codes LDPC notamment les algorithmes de décodage et les différentes techniques architecturale et matérielle existantes, afin de proposer une architecture d'un décodeur LDPC.

Organisation du document :

L'objectif principale de ce projet est d'implémenter sur la carte Xilinx FPGA BASYS-2 un décodeur LDPC pour les communications sans fils. Pour cela, ce mémoire est organisé comme suit :

Le **premier chapitre**, décrit brièvement les concepts généraux liés à la théorie de l'information et au codage canal, une description de la chaîne de communication est initialement introduite en suite détaillée, en indiquant le rôle de chaque élément, suivit d'une définition générale des codes correcteur d'erreurs et finalement une comparaison de performance entre les codes correcteurs d'erreurs les plus connus.

Dans le **second chapitre**, une large présentation des codes LDPC est proposée, incluant les notations et outils mathématiques indispensables à la compréhension, particulièrement au sujet d'encodage situé dans la dernière section, Ce chapitre traite également la classe des codes LDPC, et de la construction des codes LDPC afin de discuter les différents types d'architectures possible lors de l'implémentation d'un code LDPC, ce chapitre introduira de plus, la représentation de ces codes et en fin le concept des codes cycliques et leur avantage à augmenter la flexibilité du décodeur.

Le **chapitre troisième** est divisé en deux parties, dans un premier temps, une profonde description des algorithmes de décodage des codes LDPC dérivée de la théorie du chapitre précédent et d'une logique mathématique qui finit par une déduction de l'algorithme Min-sum, qui fera l'objet de notre architecture. tandis que la deuxième partie est consacrée pour une étude des performances des différents algorithmes de décodage conçu pour différents types de codes LDPC.

Une description architecturale des différentes implémentations des codes LDPC est présentée au **chapitre quatre**, en déduisant les avantages et inconvénients de chacune d'elles, afin de faire une comparaison des différents types de conceptions.

Enfin dans le **dernier chapitre5**, Une présentation de la carte FPGA-BASYS-2 est donnée et sur laquelle l'implémentation de notre travaille à été effectuée, nous proposons deux types d'architecture basée sur l'algorithme Min-sum sur avec lequel nous avons travaillé tout au long de notre projet, Le détail de conception, une présentation des différents blocs généré par la plate-forme Xilinx-ISE Spartane-3E sera exposés, et enfin une simulation accompagnée d'une discussion des résultats obtenus.

Une **conclusion** et quelques perspectives sont finalement données à la fin de ce document.

Chapitre 1

Généralités sur le codage canal et codes correcteurs d'erreurs

Ce premier chapitre a pour objectif de présenter des principes et les concepts fondamentaux qui seront utiles pour la compréhension des codes LDPC en commençant par illustrer et présenter la chaîne de communication en expliquant brièvement le rôle et l'intérêt de chaque bloc de la chaîne, avant de donner quelques rappels sur la théorie du codage. Puis, nous introduisons les différents types des codes correcteurs d'erreurs en focalisant sur la famille des codes linéaires en blocs, et finir par une comparaison de leur performance.

1.1 Introduction

Dans un système de communication numérique l'objectif est toujours de faire passer un message transmis en un point à partir d'un autre point. mais le problème qui se pose, est que le canal est généralement soumis à des perturbation telles que le bruit et les interférences. Alors, comment faire pour établir une communication fiable dans ces conditions ?

Les premières tentatives d'évaluations de performance datent des contributions de Nyquist en 1924[4] et de l'ingénieur Américain R.Hartetly en 1928[5], mais l'étape décisive fut franchie en 1948 par Claude E.Shannon, lorsque parut son article fondateur de la théorie de l'information, intitulé *"The Mathematical Theory of Communication"*[6] .

A la croisée de la théorie de l'information, des mathématiques et de l'électronique, le codage correcteur d'erreurs (codage de canal) a connu de nombreux développements depuis les travaux fondateurs de Shannon. Du simple code de Hamming (1950) aux récents turbo codes(1993) en passant par les codes LDPC (1962), le codage de canal a considérablement évolué et a intégré des concepts de plus en plus sophistiqués, en particulier le traitement probabiliste de l'information.

1.2 La chaîne de communication numérique

Une chaîne de communication numérique modélise les différentes étapes autorisant le transfert de l'information d'une source vers un destinataire. Une chaîne de transmission peut différer grandement selon la nature du système de transmission, qui peut aussi bien être un système de stockage de données qu'un système de télécommunication. Néanmoins certaines étapes sont applicables dans tous les contextes. Ainsi, le couple codeur décodeur de source peut être employé pour réduire la taille du message à transmettre par une opération de compression-décompression réalisée avec ou sans perte d'information. Par contre, le couple codeur-décodeur de canal augmente la taille du message à transmettre afin d'accroître la qualité de l'information transmise sur le canal.

Le canal, justement, comprend toutes les étapes entre la sortie du codeur et l'entrée du décodeur. Cela intègre le lieu physique de la transmission, dans lequel le signal transmis est bruité d'une façon aléatoire, et le couple modulateur-démodulateur, qui est utilisé dans la plupart des systèmes pour transformer l'information numérique en un signal continu compatible avec le milieu de transmission et vice-versa.

On désigne par le paradigme de Shannon (voir Figure 1.1) le schéma fondamental d'une communication numérique. Une source engendre un message à l'intention d'un destinataire. La source et le destinataire sont deux entités séparées, éventuellement distantes, qui sont reliées par un canal qui est le support de communication d'une part, mais qui d'autre part est le siège des perturbations[7].

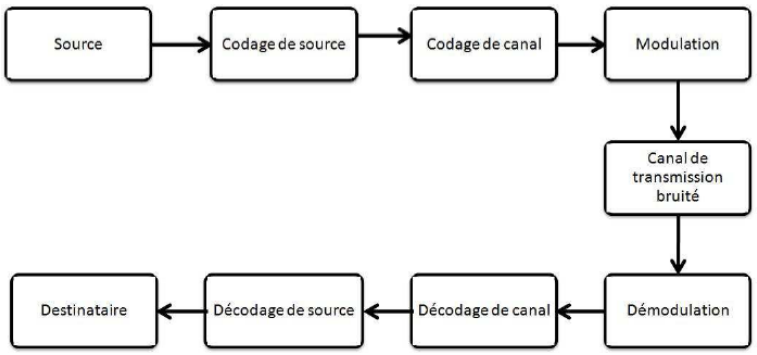

FIGURE 1.1 – Schéma fondamental d'une communication numérique : le paradigme de Shannon

La suite de cette section illustre un peu plus en détail chacune des étapes d'une chaîne de communication en se contraignant au cas de la chaîne de communication sans fil présentée dans la figure au dessus.

1.2.1 Le codage de source

Le codage de source vise à la concision maximale. L'usage d'un canal coûte d'autant plus cher que le message est long, le verbe "coûter" devant s'étendre ici en un sens très général, celui d'exiger l'emploi de ressources limitées telles que le temps, la puissance et la bande passante. Pour diminuer ce coût, on cherche à représenter le message avec le moins débits possibles, c'est-à-dire le compresser. Pour ce faire, on essaye d'éliminer la redondance contenue dans le message transmis par la source[8]. Un point essentiel dans le codage de source est le critère de Fidélité. Ce critère varie selon l'application et sur tout selon l'importance du domaine d'utilisation. Les applications où la compression de données doit se faire sans perte, utilisent un codage de source appelé réversible [1], tandis que, les applications où les pertes sont tolérables, utilisent un codage dit non-réversible [2][9].

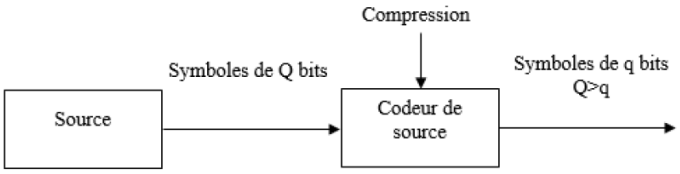

FIGURE 1.2 – Schéma simplifié d'un codeur de source

1.2.2 Codage canal

Alors que le codage de source élimine la redondance indésirable dans les informations à envoyer, la finalité du codage de canal est de protéger le message contre les erreurs et les perturbations du canal pouvant résulter des imperfections et du bruit du canal par l'ajout d'une redondance à l'information utile du message à transmettre. La redondance et l'information utile sont liées par une loi donnée. La sortie du codeur de canal est un mot de code à partir d'un code de canal, A la réception, le décodeur de canal exploite la redondance produite par le codeur dans le but de détecter, puis de corriger si c'est possible les erreurs introduites lors de la transmission[10] . Ce point sera détaillé encore plus dans les sections suivantes.

1.2.3 La modulation

La modulation consiste à effectuer un codage dans l'espace euclidien, espace généralement adapté aux canaux rencontrés en pratique.Pour une modulation M-aire, on associe à chaque mot de L bits un signal $x_i(t)$, i = 1,...,M de durée T choisi parmi les M = 2L

1. Réversible :le message envoyé sera exactement restitué au niveau du récepteur (sans perte ou distorsion).

2. Par exemple, dans la norme MPEG4, la compression des données multimédia est faite avec une distorsion tolérable par l'observateur (critère de fidélité), et cela à cause des défauts de l'œil (persistance rétinienne) et de l'oreille humaine(l'effet de masquage).

signaux. Quant à la démodulation, son rôle est d'extraire les échantillons et de décider en faveur des symboles les plus probablement émis[11]. Les données fournies par l'unité de démodulation seront traitées par ce que l'on appelle le décodeur. On distingue deux types de décodeurs, le premier est appelé décodeur à décision dure(Hard decision), car il fonctionne à partir des données fermes('0'ou'1'). Le second type est appelé décodeur à décision pondérée(Soft decision), car le démodulateur fournie au décodeur une valeur ferme accompagnée d'une mesure de fiabilité[12].

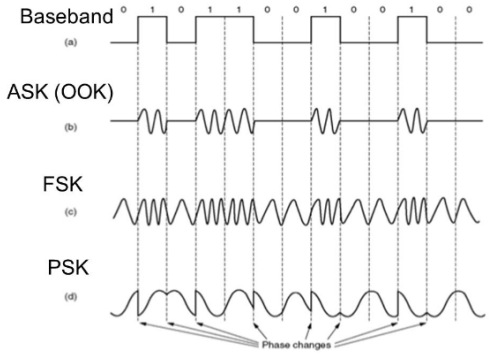

FIGURE 1.3 – Exemple de modulations numériques

1.2.4 Le canal de communication

Le canal de communication est le support physique permettant d'acheminer un message entre une source et un ou plusieurs destinataires. Au sens des communications numériques, et comme représenté dans la Figure(1.1), inclut le milieu de transmission (support physique entre l'émetteur et le récepteur : câble, fibre, espace libre,...), le bruit (bruit de fond, bruit impulsif, parasite, évanouissement, distorsion, défaut, panne), et les interférences (provenant des autres utilisateurs du milieu de transmission, de brouilleurs intentionnels ou non). Il existe plusieurs type de canaux, mais en théorie d'information, les canaux les plus utilisés sont appelés canaux discrets [3]. Un canal discret est un système stochastique acceptant en entrée des suites de symboles définies sur un alphabet de sortie Y, reliés par une loi de transition $P_{Y|X}$ i.e. une matrice stochastique $M_{Y|X}$.

$$M_{Y|X} = \begin{pmatrix} P_{y_1|x_1} & \cdots & P_{y_j|x_1} \\ \vdots & \ddots & \vdots \\ P_{y_1|x_k} & \cdots & P_{y_j|x_k} \end{pmatrix} \tag{1.1}$$

Le canal sera donc défini par :

$$T = (X, Y, M_{Y|X}) \tag{1.2}$$

3. Pour plus de détails sur les autres types de canaux, le lecteur peut se reporter aux références suivantes :[13],[11],[12]et[14].

Le canal est dit sans mémoire si le symbole courant de sortie ne dépend que du symbole courant d'entrée et il ne dépend pas des précédents ni des suivants.

Les canaux binaires BEC et BSC

Le canal binaire symétrique(BSC), présenté par le schéma de droite de la Figure1.4, est le canal le plus simple qu'on puisse imaginer. Ce canal est caractérisé par des alphabets d'entrée et de sortie binaires, une probabilité d'erreur p et une matrice de transition $M_{Y|X}$ donnée par la relation. L'autre canal, présenté par le schéma de droite de la Figure 1.4, est appelé canal binaire avec effacement(BEC), ce canal est binaire à l'entrée, mais ternaire en sortie (aux deux symboles notés '0' et '1' est adjoint un troisième, noté ε). Ce canal est caractérisé par une probabilité d'erreur p et une matrice de transition $M_{Y|X}$ donnée par la relation.

$$M_{Y|X(BSC)} = \begin{pmatrix} 1-P & P \\ P & 1-P \end{pmatrix} \tag{1.3}$$

$$M_{Y|X(BEC)} = \begin{pmatrix} 1-P & P & 0 \\ 0 & P & 0 \end{pmatrix} \tag{1.4}$$

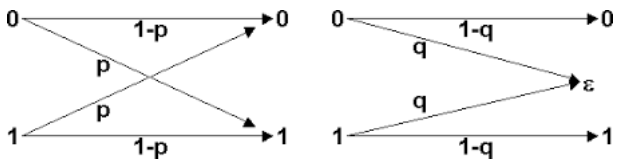

FIGURE 1.4 – (a) Le canal binaire avec effacement BEC.(b) Le canal binaire symétrique.

Le canal à bruit Blanc Additif et Gaussien

Parmi les canaux stationnaires le plus utilisé, celui sur lequel l'évaluation des performances des systèmes de communications est aussi la plus simple, est le canal à Bruit Blanc Additif et Gaussien (BABG ou AWGN en anglais : Additive White Gaussian Noise en anglais).Ce dernier est le modèle de bruit de base le plus utilisé et le plus rencontré en théorie de l'information pour rendre compte des nombreux processus stochastiques qui se produisent dans la nature. Il a les caractéristiques suivantes :

Additif : car il est ajouté au bruit intrinsèque du système.
Blanc : car sa puissance est uniforme sur toute la bande fréquentielle du système. C'est une analogie avec la couleur blanche, qui correspond une émission uniforme sur toutes les

fréquences du spectre visible.
Gaussien : car il a une distribution normale dans le domaine temporel avec une moyenne nulle.

Le modèle mathématique le plus simple pour un canal de communication est le canal à bruit additif. Dans ce modèle, le signal transmis *s(t)* est corrompu par un processus de bruit aléatoire additif *n(t)*. Physiquement, le processus de bruit additif peut provenir de composants électroniques et d'amplificateurs au récepteur du système de communication, ou des interférences rencontrées dans la transmission, comme dans le cas de la transmission du signal radio. Si le bruit est introduit principalement par des composants électroniques et des amplificateurs au récepteur, il peut être caractérisé comme un bruit thermique.
Ce type de bruit est caractérisé statistiquement comme un processus de bruit gaussien. Par conséquent, le modèle mathématique résultant pour le canal est habituellement appelé le canal additif de bruit gaussien. L'atténuation des canaux est facilement incorporée dans le modèle. Lorsque le signal subit une atténuation lors de la transmission à travers le canal, le signal reçu est :

$$r(t) = as(t) + n(t) \tag{1.5}$$

tel que a représente le facteur atténuation.

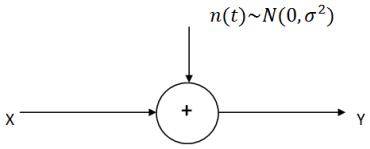

FIGURE 1.5 – Le canal binaire avec effacement BEC.(b) Le canal binaire symétrique.

Le bruit introduit dans ce canal est modélisé par un signal aléatoire *n(t)*, dont la distribution de probabilité suit la loi Gaussienne :

$$f_N(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(n-\mu)^2}{2\sigma^2} \tag{1.6}$$

tel que

$$\mu = E[n(t)] = 0$$

et

$$\sigma^2 = E[n(t) - \mu]^2 = E[n^2(t)]$$

représentent respectivement la moyenne et la variance.

Dans la suite, nous allons introduire les problèmes techniques que peut rencontrer une chaîne de transmission. ce qui nous permet de bien comprendre par la suite les différents concepts.

Trajet multiples (Multipath)

La Propagation multi-trajets apparait comme conséquence de réflexion, dispersions, atténuation de l'énergie du signal et diffractions par différents obstacles des ondes électromagnétiques émises. Cela a pour conséquence des retards à la réception, des changements de phases et des atténuations différentes[15]. Ces phénomènes sont causés par les obstacles naturels qui s'opposent à la propagation du signal radio comme les montagnes, les bâtiments,..etc. Ces réflexions obligent le signal à suivre des chemins différents pour arriver au récepteur, ce qui donne naissance au phénomène de trajets multiples. La propagation multi-trajets apparait lorsqu'un signal émis par un émetteur prend plusieurs chemin pour arriver aux récepteurs. Le récepteur reçoit des versions retardées du signal émis et les voit comme des échos avec différents temps d'arrivé et d'amplitudes.

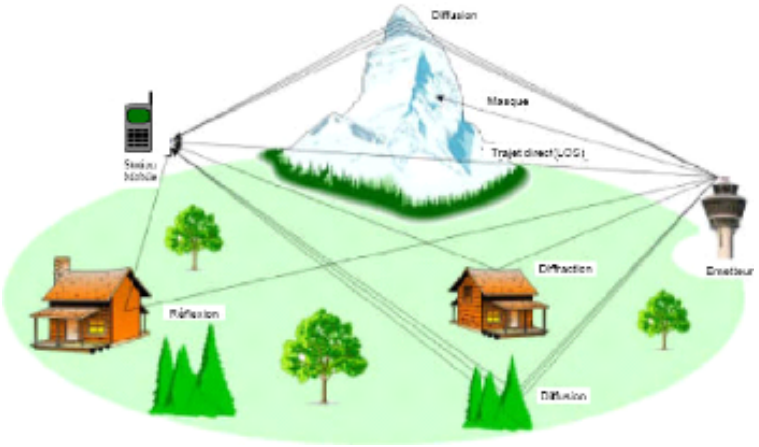

FIGURE 1.6 – Propagation par trajets multiples

Effet Doppler

Quand la source et le récepteur se déplacent l'une par rapport à l'autre, la fréquence du signal reçue au récepteur n'est pas identique à celle de la source, la distance entre l'émetteur et le récepteur varie au cours du temps, on obtient donc un décalage fréquentiel[8]. L'effet Doppler représente ce décalage de fréquence. Cette différence entre la fréquence émise et reçue appelée fréquence Doppler peut s'écrire sous la forme :

$$f_d = \frac{\upsilon f_c cos(\alpha)}{c} \tag{1.7}$$

tel que

f_d : Fréquence Doppler.

υ : est la vitesse de déplacement du récepteur.

c : est la vitesse de propagation de l'onde électromagnétique dans le vide.
α : est l'angle entre v (vitesse de déplacement) et k (direction de propagation du champ).

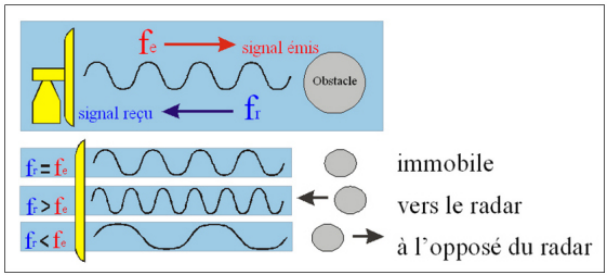

FIGURE 1.7 – Effet doppler

Effet de masquage (shadowing)

L'effet de masquage est un phénomène aléatoire plus local (sur quelques centaines de longueur d'onde), causé par l'obstruction des ondes qui se propagent, par de grands obstacles, tels que des collines, des édifices, des murs, des arbres...etc, ce qui cause une atténuation, plus ou moins importante, de la force du signal. Sa variation due à l'effet de masque est appelée évanouissement lent (slowfading) et peut être décrite par une distribution log-normale[15]. Les variations de la puissance reçue dues aux pertes par parcours et à l'effet de masque peuvent être neutralisées d'une manière efficace par le contrôle de puissance. Dans ce qui suit,on ne prendra en considération que l'évanouissement rapide.

Pertes par parcours (pathloss)

Les pertes par parcours représentent l'atténuation que subit la puissance moyenne du signal transmis le long de la distance entre l'émetteur et le récepteur. En espace libre la puissance moyenne du signal est inversement proportionnelle au carré de la distance r^2. Ce pendant dans un canal radio mobile ou, en générale, il n'a pas de visibilité (no line of sight), la puissance moyenne est inversement proportionnelle à L(tel que $r^3 < L < r^5$) [15]. la figure 1.8 résume tous les types d'évanouissement.

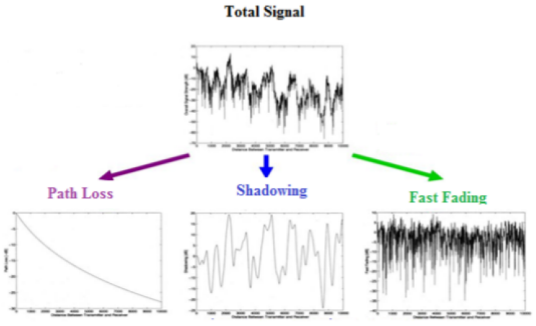

FIGURE 1.8 – schéma récapitulatif des différents types d'évanouissement

1.2.5 La capacité d'un canal

On définit la capacité d'un canal (exprimée en bit/s) comme le maximum de l'information mutuelle moyenne *I(X ; Y)*, elle représente donc la plus grande quantité d'information pouvant transiter entre l'émetteur et le récepteur[11], comme le montre la figure 1.9.

$$C = \max_{P(X)} I(X, Y) \tag{1.8}$$

On désigne par *H(X)* l'entropie de la source qui représente la surprise moyenne ou la quantité d'information délivrée par une source d'information. Par contre, *H(X| Y)* représente l'information requise pour supprimer l'ambiguïté sur l'entrée.

$$H(x) = \sum_{i}^{N} P_i \log_2 P_i \tag{1.9}$$

P_i : la probabilité d'apparition des lettres de l'alphabet de la source.
La capacité C s'exprime en Shannon par symbole ou bit par symbole. Il est également possible de l'exprimer en Shannon par seconde, on parle alors de capacité par unité de temps [13].Pour la distinguer de la capacité par symbole, on trouve généralement dans la littérature la notation suivante :

$$C_s = Cd_s \tag{1.10}$$

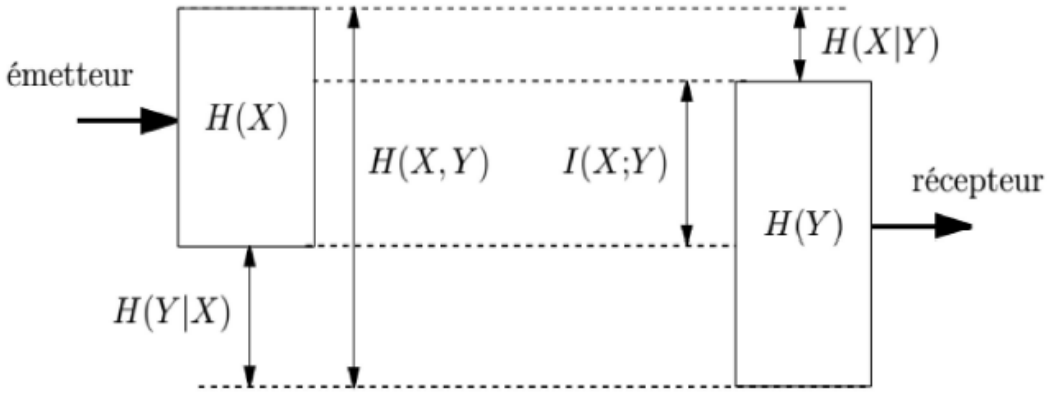

FIGURE 1.9 – schéma représentatif de l'information mutuelle

Dans le cas d'une transmission sur deux voies en quadrature sur un canal gaussien de largeur $W\ (Hz)$, et pour un rapport signal sur bruit par symbole E_s/N_0, où E_s est l'énergie par symbole émis et $N_0/2$ la densité spectrale de puissance du bruit, la capacité maximale est donnée par :

$$C = W \log_2(1 + \frac{E_s}{N_0})(bit/s) \tag{1.11}$$

La figure 1.10 illustre l'évolution de cette capacité dans le cas de quelques formes d'onde M-aires en fonction du rapport E_b/N_0 pour un canal Gaussien.

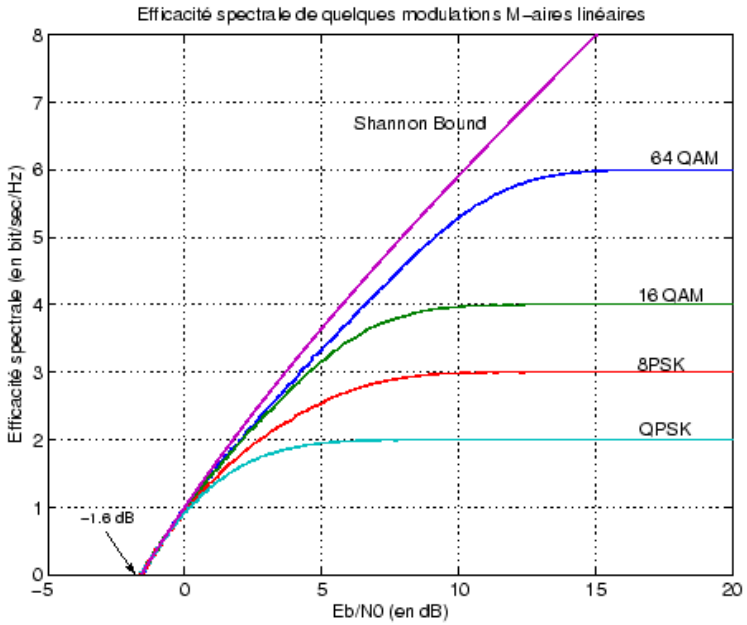

FIGURE 1.10 – Capacité d'un Canal Gaussien de quelques modulations M-aires linéaires en fonction du E_b/N_0

1.2.6 Le théorème fondamental du codage canal

En 1948, Shannon a annoncé dans son article intitulé *A Mathematical Theory of Communication communication* [6], le théorème fondamental de la théorie de l'information (Deuxième théorème de Shannon) :

"Pour une source à débit d'information R_D et un canal de capacité C, si $R_D < C$, il existe un code ayant des mots d'une longueur N tel que sa probabilité d'erreur soit arbitrairement petite".

une autre façon d'annoncer le théorème :

"Tout canal de transmission admet un paramètre C, appelé capacité du canal, tel que pour tout $C > 0$ et pour tout $R < C$, il existe un code de taux R permettant la transmission du message avec un taux d'erreurs binaire de R".

En d'autres termes, nous pouvons obtenir des transmissions aussi fiables que l'on veut, en utilisant des codes de taux plus petits que la capacité du canal.

Ce théorème affirme l'existante de codes dont la probabilité de décodage erroné est arbitrairement petite, mais ne montre pas comment ces codes peuvent être construits. Ce théorème affirme également une chose tout à fait surprenante, à savoir que, quelque soit le niveau des perturbations d'un canal, on peut toujours y passer des messages avec une probabilité d'erreur aussi faible que l'on veut. Ce théorème a causé un énorme développement dans la théorie de codage de canal [14].

Cependant, ce théorème n'indique pas le moyen de construire de tels codes, nous cherchons donc à construire des codes ayant un taux le plus élevé possible (pour des raisons de temps et de coût) et permettant une fiabilité arbitrairement grande. Les codes classiques ne permettent pas d'atteindre cette limite. Nous avons donc développé d'autres systèmes de codages.

1.3 Les codes correcteur d'erreur

Il existe une grande variété de codes correcteurs d'erreurs, dont les performances et les applications sont variables. Mais, le principe de base reste le même : ajouter de la redondance intelligemment et utiliser cette sur information pour déterminer la Fiabilité du message(détection d'erreur), puis, si c'est possible reconstruire le message d'origine au mieux (correction d'erreur). Mais en revanche, l'ajout de la redondance dans le message à transmettre,entraîne une perte d'efficacité du système. En effet, les bits de redondance introduits ne véhiculent pas de l'information utile. Cependant, cette perte est à mettre en balance avec le gain de qualité obtenu par l'utilisation du codage[16].

1.3.1 Définitions et notation

• Le codeur de canal permet de générer un mot de code X de N bits à partir d'un mot d'information c de K bits. Ce code engendre donc M bits de redondance, avec $M = N-K$, appelés bits de parité, que nous noterons par le vecteur p (voir la Figure 1.11)

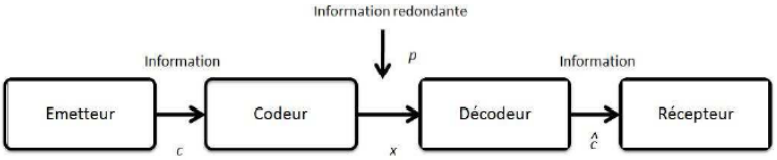

FIGURE 1.11 – Schéma simplifié d'un codeur/décodeur de canal.

• Un code est dit systématique si les symboles de c apparaissent explicitement dans x. On appelle aussi rendement de codage R, le rapport entre le nombre de bits d'information et le nombre de bits du mot de code transmis :

$$R = \frac{K}{N} \tag{1.12}$$

• Les symboles du message information c et du mot de code x prennent leurs valeurs dans un corps fini F_q à q éléments, appelé corps de *Galois* et dont les principales propriétés sont illustrées dans la référence[16]. Pour la majorité des codes, les symboles sont binaires

et prennent leur valeur dans le corps F_2 à deux éléments. Les opérations élémentaires d'addition et de multiplication dans le corps F_2 sont données dans le Tableau 1.1.

a	b	a+b	$a \times b$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLE 1.1 – addition et multiplication dans le corps de Galois F_2

• On appelle distance de Hamming entre deux codes x_i et x_j , le nombre de composantes où les deux codes sont différents, on la note par $d_H(x_i; x_j)$. On appelle poids de Hamming,noté $w_H(x)$, le nombre d'éléments non nuls présents dans un mot de code.

• On appelle distance minimale d_{min} la plus petite distance de Hamming entre deux mots de code x_i et x_j.

$$d_{min} = \min_{x_i, x_j \in Z} d_H(x_i, x_j) \tag{1.13}$$

où Z représente l'ensemble des mots de code possibles.

• Le pouvoir de détection et de correction d'un code est déterminé par sa distance minimale d_{min}. Pour détecter e erreurs pouvant intervenir dans le mot de code, il faut que :

$$d_{min} = e + 1 \tag{1.14}$$

Pour la correction de e erreurs pouvant intervenir dans le mot de code, il faut que :

$$d_{min} = 2e + 1 \tag{1.15}$$

1.3.2 Mesure des performances d'un code correcteur d'erreur

On appelle gain du codage, l'écart d'énergie par bit utile entre deux systèmes pour un taux d'erreur donné (voir Figure 1.12). Dans le cas de l'utilisation de techniques de codage avancées, l'évolution de la performance du code peut se diviser en trois régions comme l'illustre la Figure1.12. La première région correspond à un comportement où le décodage ne converge pas pour des SNR faibles, le décodage dégrade les performances par rapport à un système non codé ,on parle alors de la région de non convergence. A partir d'un certain rapport signal sur bruit SNR, appelé seuil de convergence, le décodage rentre dans une phase où la probabilité d'erreur diminue très rapidement avec le SNR, on parle de la région du Waterfall.

Enfin, il existe une région où la probabilité d'erreur diminue de manière moins rapide que la région du Waterfall. Ce comportement est spécifique de la région du plancher d'erreur ou error floor.

La performance d'un couple codeur/décodeur se juge en premier lieu en termes d'erreurs résiduelles à la sortie du décodeur, lorsqu'on s'est fixé un cadre bien précis d'évaluation : type de perturbation, longueur de message, taux de redondance ou rendement de codage etc. D'autres aspects, comme la complexité du décodage, les latences introduites par le codeur et le décodeur, le degré de flexibilité du code (en particulier son aptitude à se conformer à différentes longueurs de message et/ou à différents rendements de codage) sont également à considérer de plus ou moins près suivant les contraintes propres au système de communication. Les erreurs résiduelles que le décodeur n'est pas parvenu à corriger se mesurent à l'aide de deux paramètres. Le taux d'erreurs binaires (TEB) est le rapport entre le nombre d'erreurs binaires résiduelles et le nombre total de bits d'information transmis. Le taux d'erreurs de mots, de blocs ou de paquets (TEP) est le nombre de mots de code mal décodés (au moins un des bits d'information est faux) ramené au nombre total de mots de code émis. Le rapport entre TEB et TEP est la densité d'erreurs moyenne δ_e dans la partie systématique d'un mot mal décodé :

$$\delta_e = \frac{TEB}{TEP} \tag{1.16}$$

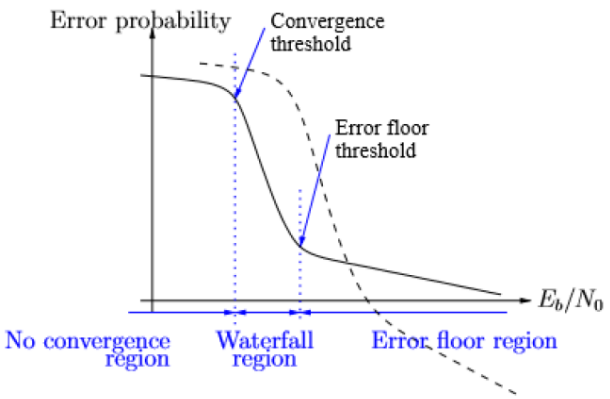

FIGURE 1.12 – Illustration des régions caractérisant les performances d'un code correcteur d'erreurs.

Remarque :

Dans les systèmes de communications, le critère utilisé pour évaluer les performance est donné en taux d'erreur binaire BER (Bit Error Rate en anglais) en fonction du SNR (rapport signal sur bruit), mais pour avoir des résultats plus exacts, sur tout quand on compare des codes de rendements différents, on utilise un autre critère qui donne le BER

en fonction de E_b/N_0 [13]. avec :
E_b : L'énergie transmise dans un bit d'information.
N_0 : La densité spectrale du bruit.

1.3.3 Concaténation de codes

La concaténation de codes consiste à combiner plusieurs codes élémentaires de taille raisonnable, de telle sorte que le code global (résultant) possède un pouvoir de correction élevé(d_{min} élevée) et qu'il soit aisément dé-codable. Le premier schéma de codage composite, appelé concaténation de codes, à été introduit par *Forney* dans son travail de thèse en 1965(voir Figure 1.13)[17]. Ce schéma est constitué d'un premier codeur, dit codeur extérieur, permettant de fournir un mot de code à un deuxième codeur, dit codeur intérieur ,pour générer un code concaténé. Si les deux codes sont systématiques, le code concaténé est lui-même systématique.

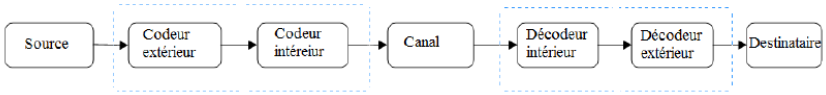

FIGURE 1.13 – Concaténation de deux codes correcteurs d'erreurs.

1.3.4 les classes des codes correcteurs d'erreurs

Un code est dit linéaire si la fonction de codage est une application linéaire, si non il est dit non-linéaire. Lorsque les traitements requis pour obtenir les propriétés de détection ou de correction ont lieu par bloc de N symboles, on dit qu'on a affaire à un code en bloc. Lorsque les symboles générés par la source ne sont pas traités par des blocs, mais de manière continue, on dit qu'on a affaire à un code convolutif. Pour la suite de ce chapitre, nous allons parler uniquement des codes en bloc et des codes convolutifs, ainsi que notre étude sera focalisée sur les codes linéaire en bloc.

1.4 Les types des codes correcteurs d'erreurs

1.4.1 les codes en bloc

Une grande famille de codes correcteurs d'erreurs est constituée des codes par blocs. Dans notre projet nous ne traiterons que ces types des codes. Pour ces codes l'information est d'abord coupée en blocs de taille constante et chaque bloc est transmis indépendamment des autres, avec une redondance qui lui est propre. La plus grande sous-famille de ces codes rassemble ce que l'on appelle les codes linéaires (Figure 1.14).

Le but de l'opération de codage en bloc est d'associer à chaque mot d'information composé de K symboles q-aire un mot de code composé de N symboles *q-aire*. Cette

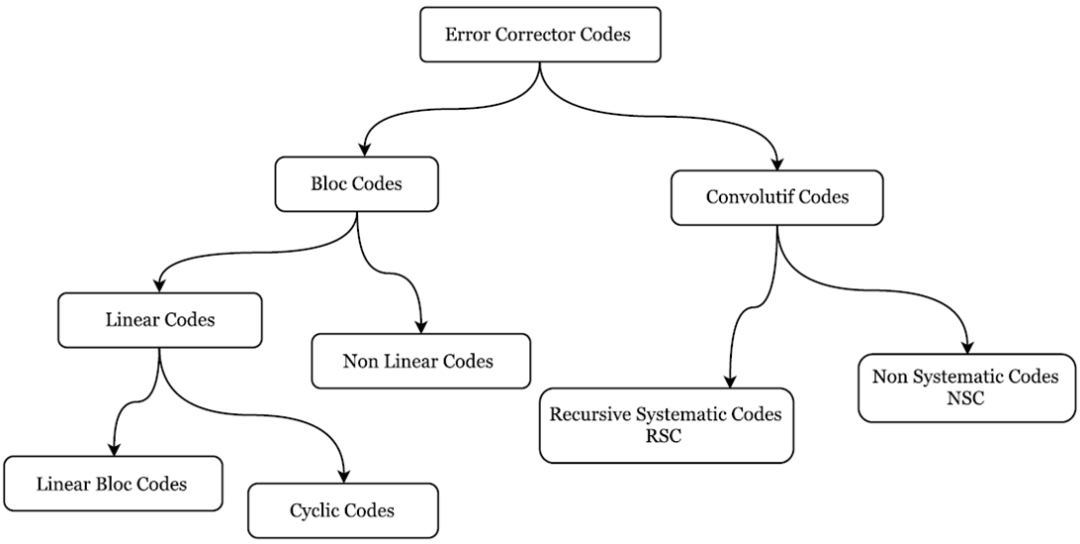

FIGURE 1.14 – la hiérarchie des codes correcteurs d'erreurs.

opération peut être représentée par une application g de l'ensemble F_K^q vers l'ensemble F_N^q

$$g : F_K^q \longrightarrow F_N^q$$
$$c \longrightarrow x = g(c)$$

L'information de la source est mise en trames de longueur fixe que nous devons transmettre c'est le message ou le mot initial. Le codage prend ce message pour en faire un mot de code : Nous appelons mot de code, la suite de n bits obtenue après un codage *(k, n)*. Le nombre n de bits qui composent un mot du code est appelé la longueur du code. La dimension k étant la longueur initiale des mots. Le message est constitué de k caractères soit 2^k messages possibles. Le mot de code utilisé sera lui aussi de longueur fixe de n caractères soit 2^n mots de code possibles. Avec $n > k$ il y'aura donc $n - k$ caractères du mot de code qui sont redondants et serviront à traiter les erreurs éventuelles. Par ailleurs, $2^n > 2^k$ donc un certain nombre de mots de code ne correspondent pas à un message mais seulement à des erreurs de transmission. On parle ainsi d'un code de bloc *(n , k)* et du rapport ou le rendement du code défini par le rapport entre k et n.

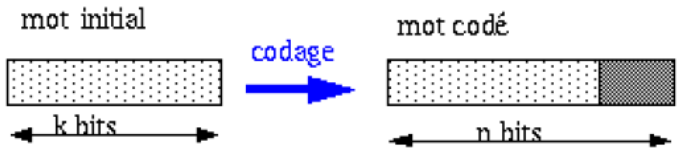

FIGURE 1.15 – le codage d'un message.

Un cas particulier des codes en blocs est celui des codes où le message apparaît explicitement sur ses k caractères c'est à dire "en clair". A côté de ces k caractères seront donc

ajoutés *n-k* caractères redondants. Nous obtenons alors un code dit code systématique.

Selon le classement donné par la Figure 1.14, Les codes linéaires se divisent en deux grandes parties :

Les codes cycliques :

sont ceux où les mots de codes ont considérés comme étant des éléments dans unea lgèbre, à savoir des polynômes[18].

Remarque :

Pour la suite, nous ne nous intéresserons qu'aux codes en bloc linéaires binaires *(q = 2)*.

Les codes linéaires en bloc :

sont ceux où les mots de code sont considérés comme étant des éléments dans un espace vectoriel

Remarque :

Pour la suite, nous ne nous intéresserons qu'aux codes en bloc linéaires binaires *(q = 2)*.

Les codes linéaires en bloc sont caractérisés par une matrice G de taille *(K,N)* appelée la ***matrice génératrice***[11]. Cette matrice transforme un message d'information c de K bits en un mot de code x de taille N *(N > K)* par l'opération matricielle suivante :

$$x = c.G \tag{1.17}$$

avec

$$G = \begin{pmatrix} g_{11} & \cdots & g_{1N} \\ \vdots & \ddots & \vdots \\ g_{K1} & \cdots & g_{KN} \end{pmatrix} \tag{1.18}$$

Chaque mot de code est une combinaison linéaire des vecteurs G_i de G. Ainsi donc, un code en bloc linéaire peut être défini comme un sous espace vectoriel à $K < N$ dimensions construit suivant la relation 1.17[13]. Pour faciliter l'opération de codage, il est toujours possible de mettre la matrice G sous la forme systématique, en combinant les lignes entre elles [4].

$$G_{syst} = \left[R^t | I_K\right] \tag{1.19}$$

4. On désigne par $(.)^t$ la transposé d'une matrice ou d'un vecteur.

$$I_K = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \tag{1.20}$$

et

$$R^t = \begin{pmatrix} r_{11} & \cdots & r_{1M} \\ \vdots & \ddots & \vdots \\ r_{K1} & \cdots & r_{KM} \end{pmatrix} \tag{1.21}$$

Les codes linéaires en bloc sont aussi caractérisés par une autre matrice H de taille *(N,M)* appelée matrice de contrôle de parité[11]. La propriété principale [5] de cette matrice est :

$$H.x^t = 0 \tag{1.22}$$

$$H.G^t = 0 \tag{1.23}$$

Dans le cas symétrique la matrice H devient comme suit :

$$H_{syst} = \Big[I_M | R\Big] \tag{1.24}$$

avec

I_M : la matrice identité d'ordre M.

et

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1K} \\ \vdots & \ddots & \vdots \\ r_{M1} & \cdots & r_{MK} \end{pmatrix} \tag{1.25}$$

Une fois l'opération de codage est terminée, le message x sera transmis à travers un canal qui est généralement bruité, un bruit n s'ajoute à ce dernier. A la réception, le message reçu r sera donné par la relation suivante :

$$r = x + n = c.G + n \tag{1.26}$$

A partir de 1.22 et 1.26, on aura :

$$H.r^t = H.x^t + H.n^t = H.n^t \tag{1.27}$$

5. En remplaçant la relation 1.17 dans la relation 1.22 on obtient $H.(c.G)^t = H.G^t.c^t = 0$, et comme cette relation est valable pour n'importe quelle séquence d'information c, donc $H.G^t = 0$.

On appelle le produit $H.r^t$ ***un syndrome*** , si le résultat de ce produit est un vecteur nul, alors r est un mot de code, si non le vecteur r contient des bits erronés.

Le calcul de syndrome est la méthode utilisée par la plupart des codes en bloc, pour détecter la présence d'erreur, puis en fonction de l'algorithme de décodage, corriger si c'est possible ces erreurs[18].

Exemples de codes en bloc

Les premiers codes en bloc sont les codes de *Hamming*, introduits en 1950 par *Richard Hamming*[19]. Ces codes donnaient des résultats médiocres par rapport aux critères de *Varshamov* et *Gilbert* [20], c'est la raison pour laquelle de nouveaux codes correcteurs d'erreurs ont été développés, on peut citer par exemple : Les code *Reed-Solomon* qui sont une classes particulière des codes cycliques BCH. Ces codes, développés par *I.S.Reed* et *G.Solomon*[21], sont largement utilisés pour la correction d'erreurs groupées dans la plupart des supports de données numériques comme les CD,DVD, blu-ray Discs, et dans de nombreux standards comme DVB-T[22]. Il existe beaucoup d'autres classes de codes en bloc, qu'on ne va pas détailler dans ce manuscrit comme : les codes de *Goppa* [23] qui sont très utilisés dans les crypto-systèmes de McEliece et Niederreiter, les codes *Reed-Muller* [24], les codes *Golay*[25]...etc. La classe de codes en bloc la plus puissante jusqu'à présent, est appelée les codes **LDPC** *(Low Density Parity Check)*. Cette famille sera l'objet de notre étude par la suite, qu'on présentera dans le chapitre 2.

1.4.2 Les codes convolutifs

Les codes convolutifs, inventés en 1954 par *Peter Elias*[26], constituent une famille de codes correcteurs d'erreurs, dont la simplicité de codage et de décodage sont à l'origine de leur succès. Le principe est non plus de découper le message en blocs finis, mais de le considérer comme une séquence semi-infinie $a_0a_1...a_n$ de symboles qui passent à travers une succession de registres à décalage, dont le nombre d'étages m est appelé mémoire du code et 2^m le nombre d'états possibles. La quantité $\mu = m + 1$ est appelée longueur de contrainte du code et le rapport $R = \frac{K}{N}$ est appelé le rendement de codage.

Pour illustrer le principe des codes convolutifs, voici un exemple présenté par la Figure 1.16, pour $K = 1$, $m = 2$ et $N = 2$. a_t parvient au codeur à l'instant t, les bits de sortie X et Y sont calculés par les relations suivantes :

$$\begin{cases} X = a_t + a_{t-1}a_{t-2} \\ Y = a_t + a_{t-2} \end{cases}$$

Supposons que le codeur reçoive le message 1011, les registres étant initialement tous les deux à 0. A la sortie on obtient la séquence codée suivante 11100001, et les registres seront finalement à l'état 11. Le diagramme en Treillis (voir Figure1.18 (b)) est une représentation utile pour l'algorithme de *Viterbi*, l'algorithme de décodage le plus utilisé pour

les codes convolutifs[27],[28].

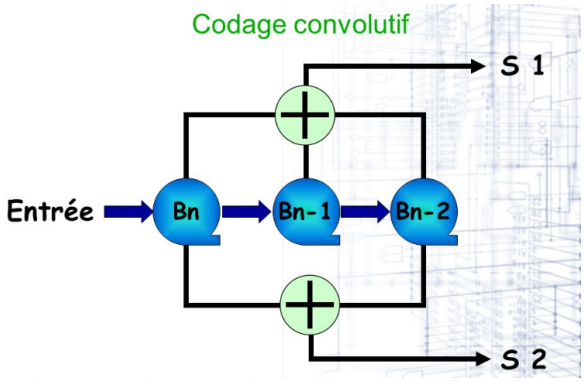

FIGURE 1.16 – Exemple d'un code convolutif.

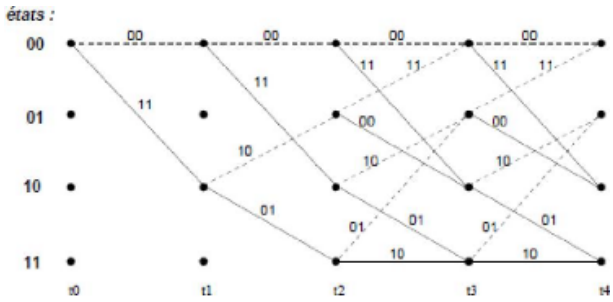

FIGURE 1.17 – Exemple d'un diagramme en Treillis.

Les codes NSC et RSC

On désigne par NSC les codes non-systématiques (Non-Systematic Code en anglais) et par RSC les codes récursifs et systématiques (Recursive Systematic Code en anglais). Un code convolutif est dit récursif si la séquence passante dans les registres à décalages est alimentée par le contenu de ses registres (voir la Figure 1.18(b)). Si les K symboles d'information à l'entrée du codeur se retrouvent explicitement dans le code, alors le code est dit systématique, si non il est dit non-systématique [12] (voir la Figure1.18(a)). Les codes non-systématiques et non-récursifs présentent,pour des SNR élevés, des performances meilleures qu'un code systématique et non-récursive et l'inverse pour les SNR faibles[29], [30]. Pour cette raison, les codes NSC ont été principalement étudiés et utilisés jusqu'au début des années 1990. On rappelle que la puissance d'un code (capacité de correction d'erreurs) et la complexité du décodeur augmentent avec l'augmentation de la mémoire m de ce code. Quant aux codes récursifs systématiques(RSC), ils sont utilisés par les turbo-codes, car ils sont les seuls susceptibles d'atteindre la limite de *Shannon*[29].

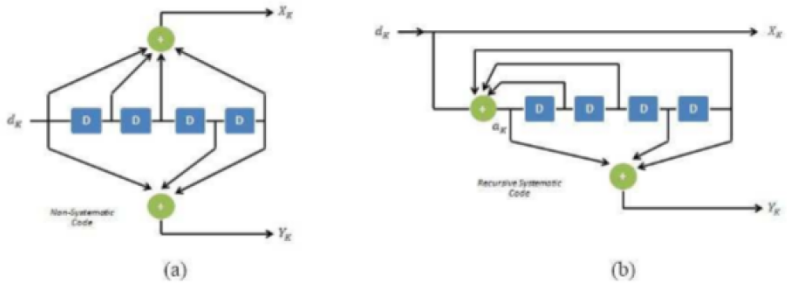

FIGURE 1.18 – (a) Un code non-systématique (NSC) (b) Un code récursif systématique (RSC).

Les turbo-codes

Le plus célèbre des codes convolutifs est sans doute le turbo-code inventé par *C.Berro*, *A.Glavieuxet*, *P.Thitimajshima* en 1993[6]. Ces codes et les codes LDPC forment ce que l'on appelle les techniques de codage avancées. Le turbo-code utilise deux (ou plusieurs) encodeurs de type convolutifs. La Figure 1.19 montre le cas d'une concaténation parallèle, constituée de deux codes convolutifs récursifs systématiques identiques et un entre laceur pseudo-aléatoire. A chaque mot d'information c, on associe une redondance p, qui peut être divisée en une redondance p_0 issue du premier encodeur et une redondance p_1 issue du deuxième encodeur. Pour la première fois, un décodeur itératif est introduit. L'idée, très simple en soi, consiste en un décodeur comportant deux sous-ensembles de décodages échangeant de l'information. Le principe de ce récepteur est illustré dans la Figure 1.20. Pour expliquer le fonctionnement d'un tel décodeur, la notion d'information extrinsèque fut introduite. C'est cette information qui est échangée entre les décodeurs au cours des itérations. Après un certain nombre d'itérations, la décision ferme est prise sur l'information a posteriori. Cette information regroupe à la fois l'information issue de l'observation du canal et les informations extrinsèques issues des différents décodeurs[16].

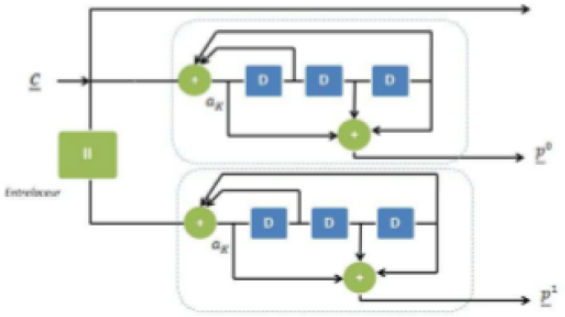

FIGURE 1.19 – Schéma de principe d'un turbo-code.

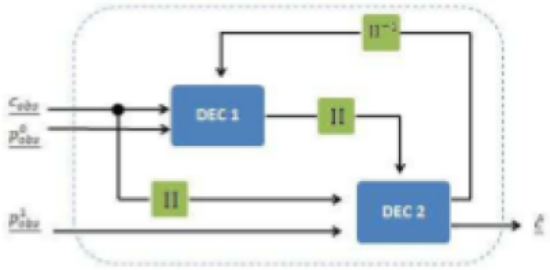

FIGURE 1.20 – Schéma de principe d'un turbo-decode.

1.4.3 Comparaison des performances entre quelques codes correcteurs d'erreurs

Avant de clôturer ce chapitre, nous allons donner une comparaison des performances entre les codes correcteurs les plus utilisés. Pour un canal à bruit blanc additif et gaussien AWGN, la Figure1.21 montre que le code de *Golay* donne des performances meilleures par rapport au code de *Hamming* et le code de parité. Pour une probabilité d'erreur égal à 10^{-5}, Le code *Golay (23,12)* apporte un gain de codage de *3,8 dB*, et le code de *Hamming (7,4)* apporte *1,8 dB*, tandis que le code de parité apporte uniquement *1 dB* par rapport au cas sans codage. Malgré cela, les performances de ces codes restent toujours médiocres par rapport aux critères de *Varshamovet Gilbert*. C'est pour cette raison que ces codes sont généralement utilisés dans des applications simples, par exemple dans le télétexte : on utilise le code de *Hamming* étendu *(8,4)*[13].

Par contre, la Figure 1.22 montre dans un ordre chronologique, l'évolution des performances des codes correcteurs. A première vue, on constate que seules les techniques de codage avancées peuvent approcher la limite de *Shannon*, par exemple pour une probabilité d'erreur égal à 10^{-5}, un code LDPC irrégulier de taille $N = 107$ et de rendement $R = 1/2$ apporte un gain de $9,6dB$ et il approche la limite de *Shannon* de $0,04dB$ (Les détails sur les codes LDPC seront illustrées dans le chapitre suivant). Quant au Turbocode avec un entre laceur de taille$N = 65536$, le gain apporté est de $9dB$ et il fonctionne à moins de $0,5dB$ de la limite de *Shannon*. Pour les autres codes : le code convolutif avec 214 états, le code R-Set le code convolutif avec 64 états, ils apportent respectivement des gains de $6.5,6$ et $5dB$ pour un taux d'erreur égal à 10^{-5}. La figure 1.23 présente une comparaison des performances des codes correcteurs d'erreurs les plus utilisées en communication. La courbe est donné présente la variation du rendement de codage en fonction de SNR.

En analysant ce résultat, on constate que seules les techniques de codage avancées comme les codes LDPC peuvent fonctionner très proche de la capacité du canal et atteindre la limite de Shannon dans des rapports signal / bruit très faibles. C'est la raison pour laquelle les codes LDPC sont utilisés dans plusieurs standard.

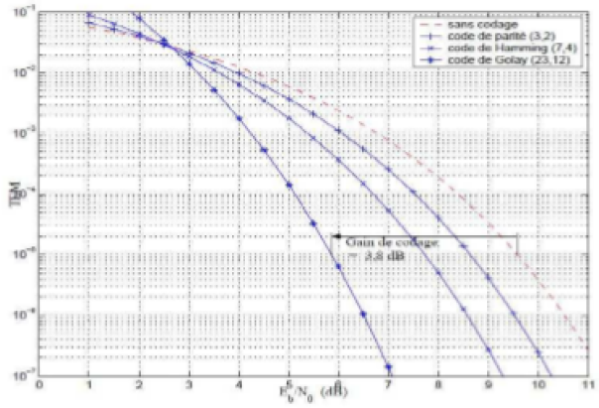

FIGURE 1.21 – Comparaison entre les performances des codes de parité, de *Hamming* et de *Golay* (Courbes reproduites de la référence[13]).

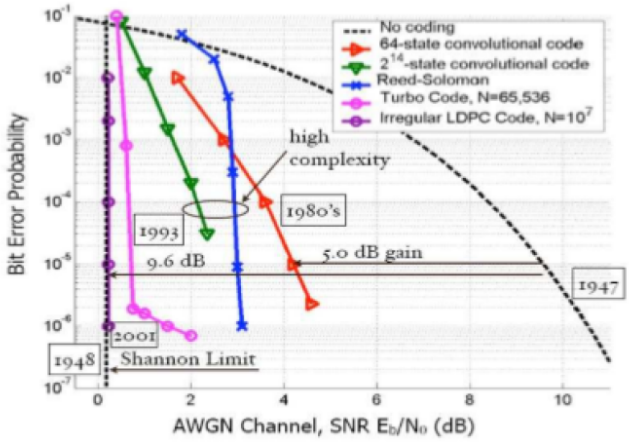

FIGURE 1.22 – Comparaison entre les performances des codes convolutifs, *Reed-Solomon* et les techniques de codage avancées (LDPC et Turbo-codes)[2].

1.5 conclusion

Dans ce chapitre, nous avons introduit, dans un premier temps, le schéma fondamental d'une communication numérique, en expliquant brièvement chacune de ses parties.

Ensuite, nous avons présenté quelques notions fondamentales sur le codage de canal, en introduisant le théorème fondamental de codage de canal et les différentes classes de code correcteur d'erreurs.

A la fin de ce chapitre, nous avons présenté quelques résultats sur les performances des codes correcteurs les plus connus. En analysant ces résultats, nous avons constaté que seules les techniques de codage avancées (turbo-codes et codes LDPC) peuvent fonctionner très proche de la capacité du canal et atteindre la limite de *Shannon*. Pour ce qui suit, notre étude sera focalisée sur des codes LDPC.

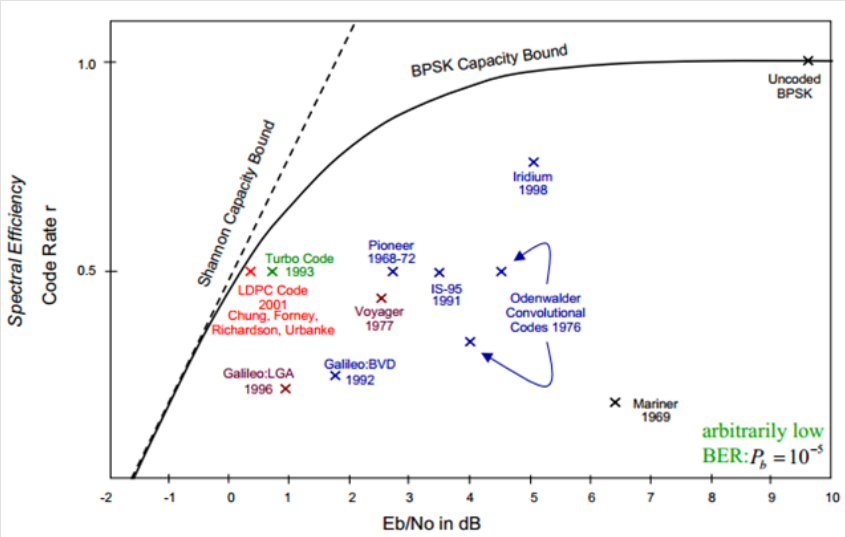

FIGURE 1.23 – Comparaison des performances des codes correcteur d'erreurs

Chapitre 2

Codage LDPC

Dans ce chapitre, nous entamerons la première partie du projet qui est l'étude des codes LDPC. Pour cela, nous commencerons d'abord par un bref historique sur les codes LDPC, ensuite, nous présenterons les différentes classes des codes LDPC ainsi que leurs présentations matricielle et graphique et de comprendre les différents concepts qui nous permettent d'entamer les chapitres qui se suivent.

2.1 Historique

Les codes LDPC ont été découverts par Gallager [21] au début des années 1960. Cette découverte remarquable a été largement ignorée par les chercheurs pendant près de 20 ans, jusqu'au travail de Tanner en 1981, dans lequel il a fourni une nouvelle interprétation des codes LDPC d'un point de vue graphique. Le travail de Tanner a également été ignoré par les théoriciens pendant environ 14 ans jusqu'à la fin des années 1990, lorsque certains chercheurs en codage ont commencé à étudier les codes graphiques et le décodage itératif. Leurs recherches ont mené à la redécouverte des codes de Gallager. Ils ont montré qu'un long code LDPC avec un décodage itératif basé sur la propagation de Believe permet une erreur de performance représentant seulement une fraction de décibel de la limite de Shannon[6]. Cette découverte fait des codes LDPC de puissants concurrents par rapport aux codes turbo pour le contrôle des erreurs lorsqu'une haute fiabilité est requise. Les codes LDPC ont l'avantage des codes turbo, il ne nécessite pas un long entrelacement pour atteindre une bonne performance d'erreur. Ainsi, en 2004, un code LDPC a d'abord été standardisé dans une émission satellite DVB-S2 [1].

Les codes «Low-density parity-check (LDPC)» sont des codes correcteurs d'erreurs, Proposés par *Gallager* pour l'obtention de sa thèse de doctorat à MIT. à cet époque l'incroyable potentiel de ces codes demeurait inexploré des calculs fastidieux qu'il nécessite pour la simulation, dans une ère où les transistors à tubes venaient juste d'être remplacés par les premiers transistors. Il demeuraient cependant négligés plus de 35 ans. Entre temps le domaine des codes correcteurs d'erreurs étaient largement dominé par les algébrique en bloc et conventionnel. Malgé les nombreux succès de ces codes, leur performances etaient

loin de répondre aux exigences théoriques des limites établies par *Shannon* dans sa publication majeur en 1948. Malgré des décennies de tentatives, les chercheurs ont échoué semble-t-il à sur passer ces contraintes pratique-théoriques.

Cette relative étude propre au domaine de codage, serait ultérieurement transformée par l'introduction des turbo-codes proposés par *Berrou*, *Glavieux* et *Thitimajshima* en 1993, où, tous les moyens de succès des codes correcteurs d'erreurs ont été remplacés : les turbo-codes impliquent peu d'algèbres, emploient des méthodes récursives, des algorithmes distribués, se focalisent sur la performance moyenne (plutot que sur le cas le plus défavorable), et reposent sur (les probabiliste) les informations extraites du canal de transmission. Du jour au lendemain, le gap de la limite de *Shannon* fut résolu en utilisant des décodeurs à complexité gérable.

Au moment où les chercheurs luttent pour comprendre pourquoi les turbo-codes ne fonctionnaient pas aussi bien qu'ils le faisaient au préalable, deux chercheurs, *McKay* et *Neal* ont introduit une nouvelle classe de codes en bloc destinés à traiter plusieurs caractéristiques de ces nouveaux turbo-codes. Il s'est vite avéré que ces codes en bloc ne sont qu'une redécouverte des codes LDPC développés quelques années plutôt par *Gallager*. En effet l'algorithme utilisé pour décoder les turbo-code n'était qu'un cas particulier du décodage des codes LDPC proposés par *Gallager* quelques années en arrière.

Une nouvelle généralisation des codes LDPC de *Gallager* a été faite par de nombreux chercheurs y compris *Luby*, *Mitzenmacher*, *Shokrollahi*, *Spielman*, *Richardson* et *Urbanke* en produisant les nouveaux codes LDPC irréguliers qui dépassent les meileurs turbo-codes, ainsi qu'offrir quelques avantages pratiques et sans doute une meilleur adaptation aux résultats théoriques, aujourd'hui, les techniques architecturales des codes LDPC sont nombreuses et permettent la construction de codes approche la capacité de *Shannon* à des centaines de décibels. Aussi rapide soit le progrès dans le domaine de la théorie de codage qu'aujourd'hui les nombreuses manières de codages ont quasiment méconnaissable par rapport à leur états une décennie en arrière. En plus de l'intérêt puissant dans les codes LDPC, de tels codes ont déjà été adoptés dans nombreux standards de communications.

2.2 Définitions et notations

Un code LDPC *(N,K)* est un code linéaire en bloc, dont les N bits du mot de code doivent satisfaire $M = N - K$ équations de parité ou équations de contrôles déduites à partir de la relation (1.22) présentée dans la section 1.4.1. La particularité de ce type de code est que sa matrice de contrôle de parité H est de faible densité, c'est-à-dire que le rapport entre le nombre d'éléments non-nuls "1" et le nombre d'éléments nuls "0" tend vers zéro. De ce fait, le processus de décodage et de calcul de syndromes (décrit dans la section 1.4.1) peuvent être effectués plus rapidement si l'on exploite la propriété de la matrice creuse. Par contre, la matrice de codage G devient dense et entraîne une complexité pour

le codeur.

Pour illustrer comment obtenir les équations de parité, voici l'exemple d'un code $x = [x_0, ..., x_6]$ de rendement $R = 3/7$, caractérisé par sa matrice de contrôle de parité $H(4, 7)$.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.1}$$

Les équations de parité associées à cette matrice et au mot de code x sont :

$$\begin{cases} x_0 \oplus x_2 \oplus x_4 \oplus x_5 = 0 \\ x_1 \oplus x_3 \oplus x_5 = 0 \\ x_1 \oplus x_2 \oplus x_6 = 0 \\ x_0 \oplus x_3 \oplus x_4 \oplus x_6 = 0 \end{cases} \tag{2.2}$$

Un code LDPC est aussi caractérisé par le nombre d'éléments non nuls présents dans les lignes et les colonnes de la matrice H. On note par w_r le poids[1] d'une ligne et par w_c le poids d'une colonne de la matrice H.

Pour que la matrice H soit de faible densité, il faut que :

$$\begin{cases} w_r \ll N \\ w_c \ll M \end{cases}$$

Selon les résultats présentés dans la référence [22], il faut que $w_c \geq 3$ pour avoir de bonnes performances du code LDPC.

2.3 Les classes de codes LDPC

Les codes LDPC se divisent en deux grandes familles, à savoir les codes réguliers et les codes irréguliers.

2.3.1 Les codes réguliers

Les codes réguliers ont été introduits par *R. Gallager* en 1962 [21]. La régularité de ces codes est spécifiée par le nombre constant de *"1"* dans les lignes et les colonnes de la

1. On rappelle que le poids d'un vecteur est égal au nombre d'éléments non nuls.

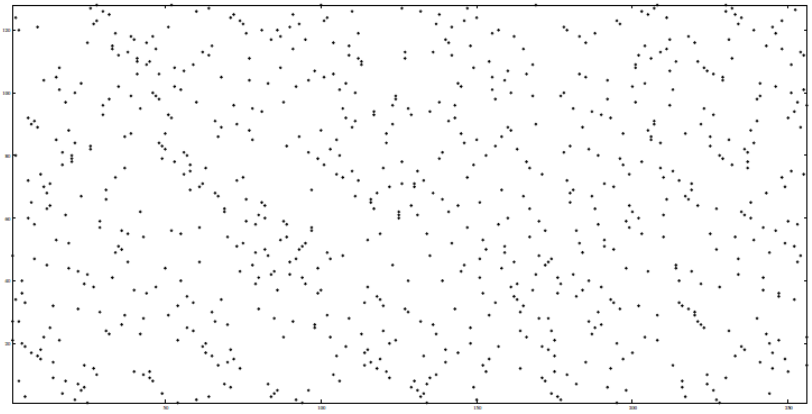

FIGURE 2.1 – Matrice de contrôle d'un code LDPC régulier *(3,6)* de taille $n = 256$ et de rendement $R = 0,5$[20].

matrice H, c'est-à-dire que w_r et w_c sont constants et reliés par la relation suivante :

$$w_r = w_c \frac{N}{M} \tag{2.3}$$

Les codes LDPC réguliers sont alors paramétrés par (N, w_r, w_c) représentant respectivement, la longueur du mot de code, le poids des lignes et le poids des colonnes. Il est clair que w_r (respectivement w_c) sont des entiers très petits devant N (respectivement M) de telle sorte que H soit clairsemée.

Dans le cas des codes LDPC réguliers, le rendement R (qu'on a défini dans la section 1.3.1) peut s'écrire de la manière suivante :

$$R = 1 - \frac{w_c}{w_r} \tag{2.4}$$

2.3.2 Les codes irréguliers

Dans le cas où la distribution des éléments non nuls est non uniforme, ces codes LDPC seront appelés codes irréguliers. L'irrégularité de ces codes n'est pas paramétré par w_c et w_r, mais plutôt par deux polynômes qu'on illustrera dans la section suivante.

La matrice présentée ci-dessous, représente le cas d'une matrice H pour un code LDPC irrégulier.

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

2.3.3 Comparaison entre les codes réguliers et irréguliers

Selon les travaux de *Ludy.* présentés dans [24] et [25], les performances d'un code LDPC irrégulier bien construit dépasse celle d'un code régulier. Les Figures 2.2 et 2.3 montrent une comparaison de performances entre un code régulier et un code irrégulier de taille *N=16000* et de rendement $R = 1/2$ et $R = 1/4$. Le canal considéré est un canal à bruit blanc additif et gaussien (AWGN).

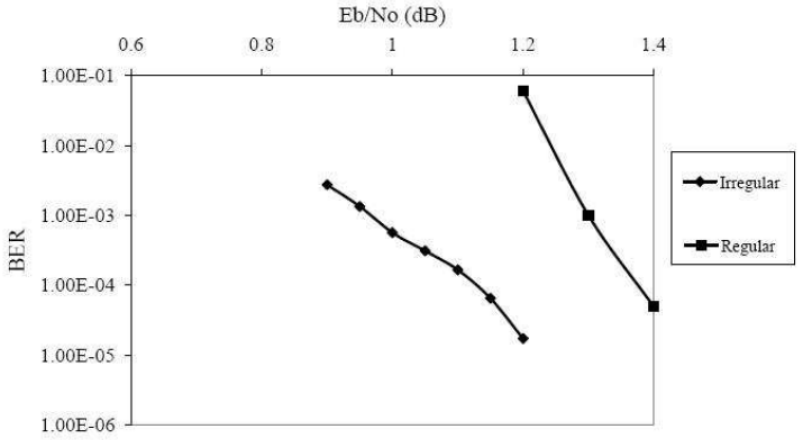

FIGURE 2.2 – Comparaison de performances entre les codes LDPC réguliers et irréguliers de taille *N=16000* et de rendement $R = 1/2$ (Graphes reproduits de la référence [24]) .

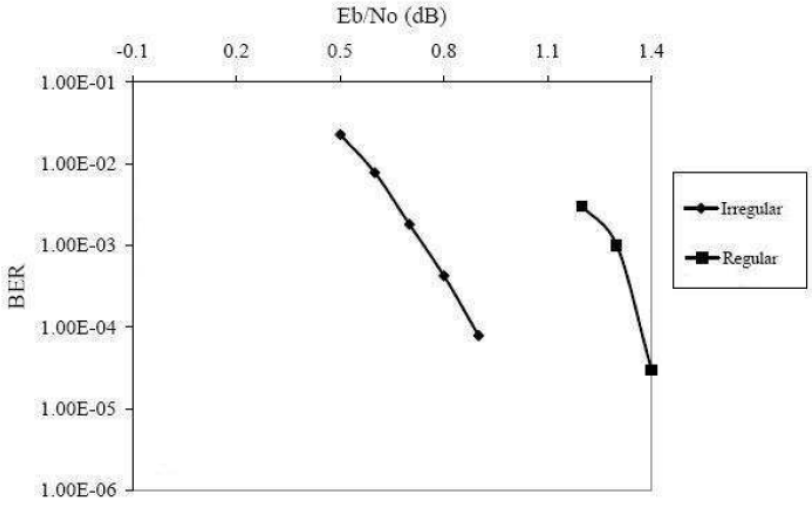

FIGURE 2.3 – Comparaison de performances entre les codes LDPC réguliers et irréguliers de taille *N=16000* et de rendement $R = 1/4$ (Graphes reproduits de la référence [3])

2.4 Construction des codes LDPC

La construction des codes LDPC binaire est équivalent à attribuer un très petit nombre de '1' à une matrice nulle tel que les lignes et les colonnes respectent un certain degré de distribution.

Le code original présenté par *Gallager* est régulier définit par une structure à bande sur *H*. les lignes de la matrice de contrôle de parité H sont divisés sur w_c ensembles avec M/w_c lignes dans chaque ensemble. Le premier ensemble de ligne contient w_r consécutives '1' ordonnés de gauche vers la droite. alors que les autres ensembles ont aléatoirement choisies selon une permutation des colonnes du premier ensemble. par conséquence chaque colonne de H a un seul '1' dans chacun des w_c ensemble.

Pour structurer les étapes avec lesquelles on conçoit les matrices de contrôle de parité, *Gallager* [21], [23] a proposé la technique suivante :

1. Définition des paramètres du code : N, K, w_c et w_r.
2. Construction d'une sous-matrice H_1 de $(N-K)/w_c$ lignes et N colonnes.
3. Construction des autres sous-matrices par une permutation pseudo-aléatoire des colonnes de H_1, qu'on note $\pi_i(H_1)$.
4. Entassement des w_c sous matrices pour construire une matrice H creuse et régulière.

$$H = \begin{pmatrix} \pi_1(H_1) \\ \pi_2(H_1) \\ \vdots \\ \pi_{w_c}(H_1) \end{pmatrix} \tag{2.5}$$

Exemple 1 : Supposons que $N = 20$, $K = 10$, $w_c = 2$ et $w_r = 4$, il s'ensuit que :

$$H_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{2.6}$$

Après la réalisation des étapes 3 et 4, on obtient une matrice de contrôle de parité H (voir (2.7)) avec un rendement de codage $R = 1 - \dfrac{2}{4} = \frac{1}{2}$

$$H_1 = \left(\begin{array}{cccccccccccccccccc} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ \hline 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{array}\right) \tag{2.7}$$

Exemple 2 : Supposons que $N = 12$, $K = 3$, $w_c = 3$ et $w_r = 4$, il s'ensuit que :

$$H = \left(\begin{array}{cccccccccccc} 1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1 \\ \hline 1&0&1&0&0&1&0&0&0&1&0&0 \\ 0&1&0&0&0&0&1&1&0&0&0&1 \\ 0&0&0&1&1&0&0&0&1&0&1&0 \\ \hline 1&0&0&1&0&0&1&0&0&1&0&0 \\ 0&1&0&0&0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&0&1&0&0&1 \end{array}\right) \tag{2.8}$$

Une autre construction très répandue des codes LDPC est la méthode proposée par *MacKay* et *Neal*.dans celle-ci les colonnes de H sont ajoutées une par une du gauche vers la droite. le poids de chaque colonne est choisis de façon à obtenir le correct degré de distribution, et la position des '1' dans chaque colonnes est choisie de façon aléatoire parmi les lignes qui ne sont pas encore remplie. Si à moment donnée il ya des lignes avec plus de positions libre que le nombre de colonnes restantes devant s'ajouter,H ne sera pas correct. le processus pourrait recommencer ou revenir en arrière de quelques colonnes, jusqu'à ce que les degrés de distribution des lignes seront obtenus.

$$H = \begin{pmatrix} 1&0&0&0&0&1&0&1&0&1&0&0 \\ 1&0&0&1&1&0&0&0&0&0&1&0 \\ 0&1&0&0&1&0&1&0&1&0&0&0 \\ 0&0&1&0&0&1&0&0&0&0&1&1 \\ 0&0&1&0&0&0&1&1&0&0&0&1 \\ 0&1&0&0&1&0&0&0&1&0&1&0 \\ 1&0&0&1&0&0&1&0&0&1&0&0 \\ 0&1&0&0&0&1&0&1&0&1&0&0 \\ 0&0&1&1&0&0&0&0&1&0&0&1 \end{pmatrix} \tag{2.9}$$

Une méthode de construction des codes LDPC appelés *repeat-accumulate* possède un poids2-colonne à travers un modèle en escalier pour les m dernières colonnes de H. cette

structure fait que ces codes LDPC sont systématiques et leur permet d'être codés facilement.

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{2.10}$$

2.5 Représentation graphique des codes LDPC

Un code LDPC peut également être représenté, en plus de sa matrice de contrôle de parité, par un graphe bipartite appelé graphe de Tanner [31], ou plus généralement graphe factoriel [32]. Ce graphe contient deux types de nœuds, les nœuds de données *(variable-nodes)* représentant le mot de code et les nœuds fonctionnels ou nœuds de contrôle *(check-nodes)* correspondant aux contraintes de parité. Un nœud de données i est relié à un nœud fonctionnel j par une branche, si et seulement si, l'élément correspondant à la i^{eme} ligne et la j^{eme} colonne de la matrice de contrôle de parité est non nul ($H_{ij} = 1$). Par convention, les nœuds de données seront représentés par des cercles et les nœuds de contrôle par des carrés.

On peut définir un graphe de Tanner par les concepts suivants :

- Nœuds de variables : associés au bits du mot de codes.
- Nœuds de parité : associés au équations de parités.
- Branches : lien entre nœuds de variables et nœuds de parité.

Un noeud de variable n sera connecté au nœud de parité m si $h_{mn} = 1$ dans la matrice.

Exemple 1 :

La figure 2.4 est une représentation graphique de la matrice de contrôle de parité suivante :

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{2.11}$$

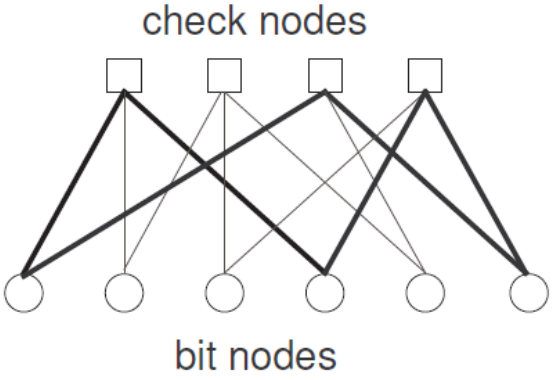

FIGURE 2.4 – Graphe de Tanner de la matrice de contrôle de parité 2.11. Un 6-cycle est affiché en gras.

Exemple 2 :

La figure 2.5 est une représentation graphique de la matrice de contrôle de parité suivante :

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2.12}$$

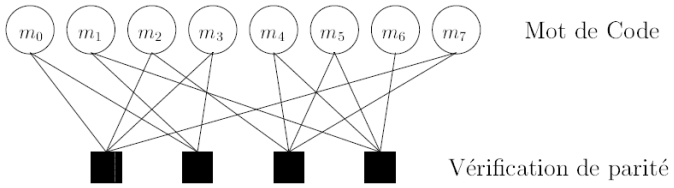

FIGURE 2.5 – Graphe bipartite dit de Tanner de la matrice 2.12

2.5.1 Le profil d'irrégularité des nœuds de données et des nœuds de contrôle

Quand le nombre de branches connectées aux différents types de nœuds est constant, on parle alors d'un code LDPC régulier. Par conséquent chaque bit du mot de code participe au même nombre d'équations de parité. De même chacune des équations de parité utilise le même nombre de bits. Par extension, les codes LDPC irréguliers sont les codes dont le nombre de branches connectées aux différents types de nœuds varie de façon irrégulière. Pour décrire ces codes, il est d'usage de spécifier l'irrégularité d'un code à travers deux

polynômes (Polynômes associés aux nœuds de variables et de parité) $\lambda(x)$ et $\sigma(x)$ [16] :

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \tag{2.13}$$

$$\sigma(x) = \sum_i \sigma_i x^{i-1} \tag{2.14}$$

Où λ_i (respectivement σ_i) caractérise la proportion du nombre de branches connectées aux noeuds de données (respectivement aux nœuds de contrôle) de degré i par rapport au nombre total de branches. Le degré est défini comme le nombre de branches connectées à un nœud.

On peut relier le profil d'irrégularité du code au rendement de codage R de la façon suivante :

$$R \geq 1 - \frac{\sum_i \frac{\lambda_i}{i}}{\sum_i \frac{\sigma_i}{i}} \tag{2.15}$$

2.5.2 La notion de cycle

On dit qu'un graphe de Tanner contient un *cycle*, s'il existe un chemin pour quitter et revenir à un nœud sans passer par les mêmes branches. Le nombre de branches traversées détermine la longueur du cycle. Un graphe sans cycle est dit graphe en arbre [16]. Le

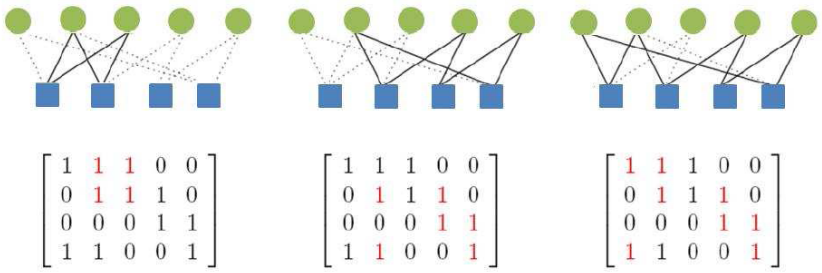

FIGURE 2.6 – Exemples de cycles de longueur 4, 6 et 8.

graphe de Tanner est une représentation graphique simple des codes LDPC. Ce graphe permet notamment d'illustrer les algorithmes de décodage, que nous présenterons dans la section suivante.

2.6 Les codes quasi-cycliques

Un code est dit quasicyclique si pour tout décalage rotatif d'un mot de code par c positions, le mot résultant reste toujours un mot de code, cependant un code cyclique est un code quasi-cyclique avec $c = 1$.

Le code quasi-cyclique le plus simple qu'il soit et un code à rotation de lignes qui est décrit par la matrice de contrôle de parité suivante :

$$H = \begin{pmatrix} A_1 & A_2 & \cdots & A_l \end{pmatrix}$$

Où $A_1, .., A_l$ est une matrice binaire circulaire de taille $v * v$.

A condition que l'une des matrices circulaires est inversible (disons tous). La matrice génératrice du code peut être construite sous la forme systématique suivante :

$$G = \begin{pmatrix} & & A_l^{-1}A_1 \\ & & A_l^{-1}A_1 \\ & I_{v(l-1)} & : \\ & & : \\ & & A_l^{-1}A_{l-1} \end{pmatrix} \tag{2.16}$$

G est donc une matrice quasi-cyclique de longueur *v.l* et de dimension *v(l-1)*. Comme une des matrices circulaire est inversible, la construction de la matrice génératrice de cette façon impose un rang complet à la matrice *H*.

2.7 Opérations d'encodage

Nous avons précédemment souligné que la matrice génératrice pour un code avec sa matrice de contrôle de parité H peut être trouvé en appliquant l'élimination de *Gauss-Jordan* pour l'obtenir sous la forme standard suivante :

$$H = \begin{bmatrix} A & I_{n-k} \end{bmatrix}$$

où : A est une matrice binaire de dimension $(n-k) \times k$ et I_{n-k} est la matrice identité de dimension $(n \times k)$, la matrice génératrice serait donc :

$$G = \begin{bmatrix} I_k & A^T \end{bmatrix}$$

Ici, dans ce chapitre, nous allons plus en détail à travers l'exemple suivant :

Exemple :

Nous souhaitons encoder le code LDPC suivant :

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2.17}$$

Premièrement, nous mettons H sous forme lignes échelonnées (i.e.tel que pour chaque deux lignes successive non nulles,le '1' leader de la ligne inférieur apparaît directement à

droite du '1' leader de la ligne supérieur).
La matrice H est mise sous cette forme en appliquant des opérations élémentaires sur les lignes de la matrice dans l'espace GF(2). ce qui semble à inter-changer les lignes deux à deux en additionnant une ligne à une autre. les propriétés de l'algèbre linéaire nous assurent un même ensemble de mots de code en utilisant uniquement des opération élémentaires sur les lignes.
La première et la deuxième colonne de la matrice H ont déjà un '1' dans la diagonal et les '1' de ces colonnes au-delà de la diagonal seront enlevé en remplaçant la 4ème ligne par la somme modulo-2 de la ligne 4 avec la ligne 1.

$$H_r = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{2.18}$$

Ensuite, la matrice de contrôle de parité est mise sous la forme échelonnée réduite (i.e chaque colonne possédant un '1' leader a des zéros par tout ailleurs). La première colonne est déjà correct et le '1' de la deuxième colonne est éliminé en remplaçant la première ligne par la somme modulo-2 de la première et deuxième colonne. De la même manière le '1' de la troisième colonne au dessus de la diagonal est éliminé par remplacer la deuxième ligne par la somme modulo-2 de la deuxième et troisième ligne. pour effacer le '1' de la quatrième colonne, la première ligne est remplacée par la somme modulo-2 de la première et la quatrième ligne. En finissant avec la 4ème et la 5ème colonne, nous obtenons la matrice H_{rr} qui est sous la forme échelonnée réduite :

$$H_{rr} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{2.19}$$

Pour terminer, quelques permutations des colonnes permettent de mettre la matrice de contrôle de parité sous la forme standard (les "m" dernières colonnes de H_{std} sont les m colonnes de H_{rr} qui contiennent les '1' leader) :

$$H_{std} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.20}$$

Dans cette dernière étape, la permutation des colonnes utilisé pour avoir la matrice H_{std} engendre une inversion du mot de code correspondant à la matrice H, une solution est

mémoriser les permutations des colonnes comme dans ce cas :

$$\pi = \begin{pmatrix} 6 & 7 & 8 & 9 & 10 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Et d'appliquer une permutation inverse pour chaque mot de code obtenu de H_{std} avant qu'il ne soit transmis. Si non lorsque le canal est sans mémoire, l'ordre dans le mot de code n'est plus important, une méthode très simple est d'appliquer π à la matrice de contrôle de parité original H pour avoir H_p :

$$H^p = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2.21}$$

Ayant les mêmes propriétés que H mais partageant le même ordre de bit dans le mot de code que celui de la matrice H_{std}.
Finalement, une génératrice G pour le code avec comme matrice de contrôle de parité H_{std} et H est donnée par :

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{2.22}$$

Toutes ces opérations peuvent être faite hors-ligne, seulement les matrices G et H^p seront fournies à la fois à l'encodeur et au décodeur respectivement. cependant, l'inconvénient de cette approche est que, contrairement à H, la matrice G sera très probablement non creuse, ce qui fait que multiplication matricielle :

$$c = uG \tag{2.23}$$

Au niveau de l'encodeur aura une complexité de l'ordre de n^2. et comme n est large pour les codes LDPC, de quelques milliers à des centaines de milliers de bits, l'encodeur peut devenir extrêmement complexe.

2.8 conclusion

Ce chapitre a présenté une étude détaillée des codes LDPC, il en ressort que nous retenons ce qui suit :
— le codage peut être réalisé de manière linéaire en temps car la matrice génératrice peut être réalisée à laide de simples registre à décalage.

— Représentation de H simplifié par utilisation conjointe de la matrice de base et des polynômes associés à l'extension.
— Le codage peut être réalisé de manière fortement parallélisé, ces codes ont en générale un bon compromis complexité/performance.

Chapitre 3

Algorithmes de décodage LDPC

Ce chapitre expose les principaux algorithmes de décodage LDPC notamment l'algorithme *"Min-Sum"* retenu solution du décodage et qu'on va l'implémenter par la suite. Puis, les performances des différents techniques de codage et algorithmes de décodage LDPC sont étudiés et des comparaisons par rapport aux différents paramétrés de codage sont établies.

3.1 Introduction

Le décodage optimal des codes LDPC met en œuvre des algorithme itératifs qui propagent les informations extrinsèques et les informations a priori dans le graphe bipartite de *Tanner* reliant les variables aux équations de parité. Nous proposons dans ce chapitre tout d'abord une illustration de ces algorithmes ainsi des exemples d'application de l'algorithme *"Min-Sum"* pour comprendre le déroulement de ce dernier afin de nous donner une prévision sur le choix de l'architecture du décodeur qu'on va le voir dans les chapitres suivantes.

La classe des algorithmes de décodage utilisés pour décoder les codes LDPC sont communément(collectivement) appelés message-passing algorithmes puisque leur opération peut être illustré(expliqué) par le passage de l'information(message)à décoder à travers les nœuds du graphe de Tanner.2.3
Chaque nœud appartenant au graphe de Tanner travaille en isolation,ne pouvant par conséquent avoir accès qu'aux informations contenu dans les messages provenant des autres nœuds avec les quelles il est connecté.
Les message-passing algorithmes sont aussi connu sous le terme d'algorithmes itératives du moment ou les messages "basculent" des nœuds de bits aux nœuds de contrôle itérativement jusqu'à ce que le résultat soit trouvé ou que l'algorithme arrive à sa fin. Entre autre, pour cette même classe, les noms des algorithmes qui y appartiennent héritent leur noms soit des types des messages qui circules dans le graphe de Tanner ou des types des opérations qui s'exécutent au niveau des nœuds, nous citons comme exemple : bit flipping, Min-Sum, qui seront présentés dans ce chapitre.

Dans certains types d'algorithmes, tel que "bit flipping decoding" le message est binaire mais il est probabiliste dans "belief propagation" öù la valeur de probabilité attribué aux messages représente la quantité de croyance pour le bit du code word. dans cette situation, il est souvent convenable de représenter la probabilité par le rapport devrai- semblance, et une fois encore cet algorithme "belief propagation decoding" est souvent appelé "Sum-Product decoding" vu qu'on aura a appliquer uniquement les opérations d'addition et de multiplication au niveau des noeuds de bits et ceux de contrôles.

Un graphique Tanner est un graphique bipartite composé de deux ensembles de nœuds nommés nœuds variables (VN) et nœuds de contrôle (CN). Les codes LDPC sont décodés itérativement avec le passage de messages entre VN et CN en fonction des connexions du graphique de Tanner.

Le décodage des codes LDPC est effectué itérativement à l'aide du Graphique de Tanner. Deux demi-itérations sont répétées jusqu'à mot de code est corrigé ou jusqu'à un nombre maximum d'itérations est atteint. Les VN et les CN échangent des probabilités représentant un niveau de croyance sur la valeur de chaque bit de mot de code et il est souvent pour représenter ces probabilités en tant que logarithme du rapport de vraisemblance (LLR). Dans la première demi-itération, chaque nœud de contrôle j calcule un message extrinsèque à chaque nœud variable i connecté par un bord dans le graphique de Tanner. Ce message s'appelle E_{ij} et représente le LLR de la probabilité que le bit i provoque la parité vérifier l'équation j à satisfaire.

3.2 Algorithme bit flipping

L'algorithme "basculement de bits" (bit flipping en anglais) est un algorithme de passage de message (message-passing) à décision . Une décision binaire (dure) sur chaque bit reçu est faite par le détecteur et ceci est passé au décodeur. Pour l'algorithme de retournement de bit les messages transmis le long des bords du graphe de *Tanner* sont également binaires : un nœud envoie un message déclarant si c'est un "1" ou un "0", et chaque nœud de contrôle envoie un message à chaque nœud de bit connecté, déclarant quelle valeur le bit est basé les informations disponibles pour le nœud de contrôle. Le nœud de vérification détermine que son équation de contrôle de parité est satisfaite si la somme modulo-2 du bit entrant les valeurs sont nulles. Si la majorité des messages reçus par un nœud sont différents à partir de sa valeur reçue, le nœud de bits change (retourne) sa valeur actuelle. Ce processus est répété jusqu'à ce que toutes les équations de contrôle de parité soient satisfaites, ou jusqu'à ce que un nombre maximum d'itérations du décodeur est atteint.

Le décodeur de bit-flipping peut être immédiatement terminé chaque fois qu'un valide mot de code a été trouvé en vérifiant si toutes les équations de contrôle de parité sont satisfaites. Cela est vrai pour tous les décodages de messages LDPC. Deux avantages importants, premièrement, des itérations supplémentaires sont évitées une fois qu'une so-

lution a été trouvé, et d'autre part un échec à converger vers un mot de code est toujours détecté.

L'algorithme de basculement de bits est basé sur le principe qu'un bit de mot de code impliqué dans un grand nombre d'équations de contrôle incorrect est susceptible d'être incorrect lui-même. L'éparpillement de H aide à étaler les bits dans les contrôles de sorte que les équations de contrôle de parité ne contiennent probablement pas le même ensemble de bits de mot de code.

La figure 3.1[34] montre une présentation de l'algorithme "Bit flipping". L'entrée est la décision difficile sur le vecteur reçu, $y = [y_1, ..., y_n]$, et la sortie est $M = [M_1, ..., M_n]$.

```
1:  procedure DECODE(y)
2:
3:      I = 0                                              ▷ Initialization
4:      for i = 1 : n do
5:          M_i = y_i
6:      end for
7:
8:      repeat
9:          for j = 1 : m do                          ▷ Step 1: Check messages
10:             for i = 1 : n do
11:                 E_{j,i} = Σ_{i'∈B_j, i'≠i} (M_{i'}  mod 2)
12:             end for
13:         end for
14:
15:         for i = 1 : n do                              ▷ Step 2: Bit messages
16:             if the messages E_{j,i} disagree with y_i then
17:                 M_i = (r_i + 1  mod 2)
18:             end if
19:         end for
20:
21:         for j = 1 : m do                          ▷ Test: are the parity-check
22:             L_j = Σ_{i'∈B_j} (M_{i'}  mod 2)              ▷ equations satisfied
23:         end for
24:         if all L_j = 0 or I = I_max then
25:             Finished
26:         else
27:             I = I + 1
28:         end if
29:     until Finished
30: end procedure
```

FIGURE 3.1 – Algorithme de décodage bit flipping[34]

La figure 3.2 présente un décodage "bit flipping" de la chaîne reçue $y = [101001]$. Chaque sous-figure indique la décision prise à chaque étape de l'algorithme de décodage sur la base des messages de l'étape précédente. Une croix représente que le contrôle de parité n'est pas satisfait alors qu'une coche indique qu'il est satisfait. Pour les messages, une flèche pointillée correspond au "bit messages = 0 "alors qu'une flèche pleine correspond à "bit = 1".

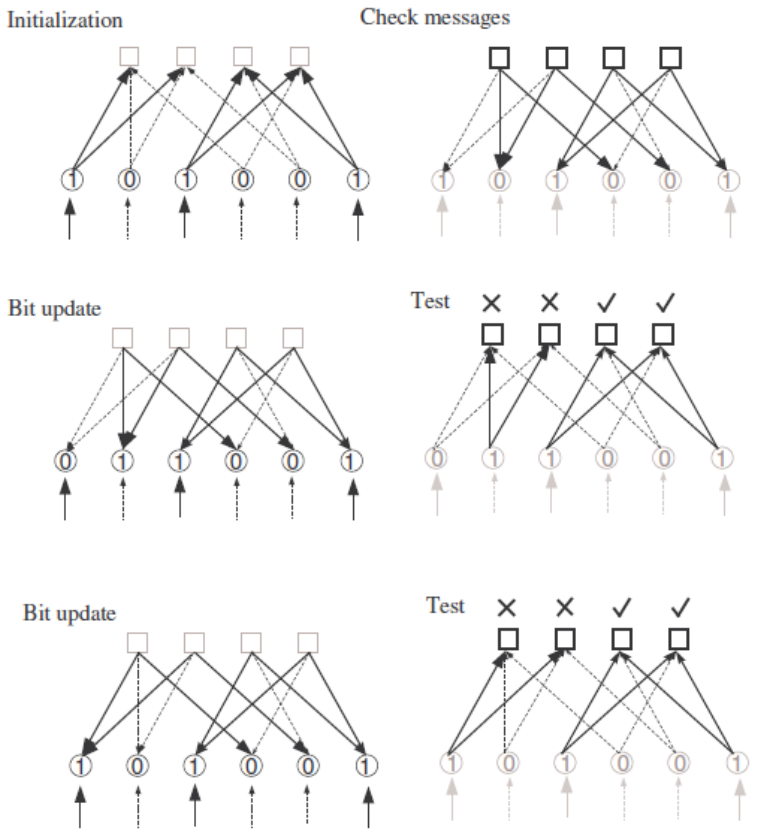

FIGURE 3.2 – Illustration de l'algorithme de décodage Bit flipping

3.3 Algorithme Sum-Product

L'algorithme sum-product est un algorithme à décision pondérée(soft) message passing algorithme, similaire à bit-flipping algorithme décrit dans la section précédente mais avec des probabilités pour représenter chaque décision prise au niveau des nœuds, en effet, les bits reçu en entrée du décodeur sont désormais des probabilités, et sont appelés a priori probabilités car ils ne seront connu qu'après l'exécution de l'algorithme. les probabilités des bits renvoyés(retournés) par le décodeur sont eux aussi appelés a posteriori probabilités. Pour ce type d'algorithme les probabilités sont exprimés en maximum de vraisemblance.

pour une variable binaire x il est facile à trouver $p(x = 1)$ connaissant $p(x = 0)$, selon :

$$p(x = 1) = 1 - p(x = 0) \tag{3.1}$$

Nous avons seulement besoin de stocker une valeur de probabilité pour x. Les rapports de vraisemblance log servent à représenter les métriques d'une variable binaire par une seule

valeur :

$$L(x) = \log \frac{p(x=0)}{p(x=1)} \tag{3.2}$$

où nous utilisons *log* pour signifier log_e.
Si $p(x = 0) > p(x = 1)$ alors $L(x)$ est positif et plus la différence entre $p(x = 0)$ et $p(x = 1)$ est grande, plus la valeur positive pour $L(x)$ est grande. Inversement, si $p(x = 1) > p(x = 0)$ alors $L(x)$ est négatif et plus la différence est grande entre $p(x = 0)$ et $p(x = 1)$, plus la valeur négative de $L(x)$ est grande. Ainsi le signe de $L(x)$ fournit la décision difficile sur x et la grandeur $|L(x)|$ est la fiabilité de cette décision.

Pour traduire les rapports de vraisemblance logarithmique probabilités, nous notons que (en utilisant 3.1 et 3.2) :

$$p(x=1) = \frac{e^{-L(x)}}{1+e^{-L(x)}} \tag{3.3}$$

Et

$$p(x=0) = \frac{e^{L(x)}}{1+e^{L(x)}} \tag{3.4}$$

L'avantage de la représentation logarithmique des probabilités est le passage de la multiplication à l'addition, par conséquent réduire la complexité d'implémentation.

Le but du décodage somme-produit est de calculer le maximum a posteriori probabilité (MAP) pour chaque bit de mot de code, $P_i = p(c_i), i = 1, N$ qui est la probabilité que le i-ème bit du mot de code est un 1 conditionnel à l'événement N que tous les contraintes de parité-vérification sont satisfaites. Les informations supplémentaires sur le bit que j'ai reçu à partir des contrôles de parité est appelée information extrinsèque pour le bit i.

Cet algorithme calcul itérativement une approximation de la valeur MAP pour chaque bit du code.Pourtant, les probabilités a posteriori retournées par ce décodeur sont exacte si seulement le graphe de *Tanner* est un cycle libre. brièvement, l'information extrinsèque obtenu des équations de contrôle de parité dans lors de la première itération est indépendante de la probabilité a priori pour un bit donnée (mais elle dépend évidemment de celle des autres bits). L'information extrinsèque fournie au bit i lors d'une itération ultérieure (subséquente) reste indépendante de la probabilité a priori original pour le bit i jusqu'à ce que cette dernière soit retournée à travers un cycle dans le graphe de *Tanner*. Finalement la corrélation de l'information extrinsèque avec la probabilité a priori original du bit i est la raison pour laquelle la probabilité a posteriori ne soit pas toujours exact.

Dans cet algorithme Sum-Product le message extrinsèque provenant du nœud de contrôle j vers le nœud de bit i, E_{ji}, est le LLR de la probabilité que le bit i engendre la satisfaction de l'équation de contrôle de parité j.

La probabilité que l'équation de contrôle de parité soit satisfaite si le bit i est à '1' est :

$$P_{j,i}^{ext} = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2\ P_{i'}^{int}) \tag{3.5}$$

Où $P_{i'}^{int}$ est l'estimation actuelle disponible au niveau du noeud de contrôle j, de la probabilité que le bit $i^{0'}$ est un 1. la probabilité que l'équation de contrôle de parité soit satisfaite si le bit i est un 0 est par conséquent $1 - P_{j,i}^{ext}$, en l'exprimant par le rapport de maximum de vrai semblance il devient :

$$E_{j,i} = LLR\ (P_{j,i}^{ext}) = \log\ (\frac{1 - P_{j,i}^{ext}}{P_{j,i}^{ext}}) \tag{3.6}$$

Et en substituant (3.5), l'expression (3.6) devient :

$$E_{j,i} = \log\ (\frac{\frac{1}{2} + \frac{1}{2} \prod\limits_{i' \in B_j, i' \neq i} (1 - 2\ P_{i'}^{int})}{\frac{1}{2} - \frac{1}{2} \prod\limits_{i' \in B_j, i' \neq i} (1 - 2\ P_{i'}^{int})}) \tag{3.7}$$

En utilisant cette formule dans l'expression (3.7) :

$$tanh\ (\frac{1}{2}\ log(\frac{1-p}{p})) = 1 - 2p \tag{3.8}$$

On obtient :

$$E_{j,i} = log\ (\frac{1 + \prod\limits_{i' \in B_j, i' \neq i} tanh(M_{j,i'}/2)}{1 - \prod\limits_{i' \in B_j, i' \neq i} tanh(M_{j,i'}/2)}) \tag{3.9}$$

Tel que

$$M_{j,i'} = LLR(P_{j,i'}^{int}) = log(\frac{1 - P_{j,i'}^{int}}{P_{j,i'}^{int}}) \tag{3.10}$$

En utilisant également cette formule dans (3.9) :

$$2\ tanh(p)^{-1} = log(\frac{1+p}{1-p}) \tag{3.11}$$

(3.9) peut être écrit de manière équivalente comme :

$$E_{j,i} = 2\ tanh^{-1}(\prod_{i' \in B_j, i' \neq i} tanh(M_{j,i'/2})) \tag{3.12}$$

Chaque bit a accès à l'entrée a priori LLR, r_i, et les LLR de chaque nœud de vérification connecté. Le LLR total du i-ème bit est la somme de ces LLR :

$$L_i = LLR\ (P_i^{it}) = r_i + \sum_{j \in A_i} E_{j,i} \tag{3.13}$$

Cependant, les messages envoyés par les nœuds de bits aux nœuds de contrôle, $M_{j,i}$, sont pas la valeur LLR complète pour chaque bit. Pour éviter de renvoyer à chaque nœud de

procedure DECODE(**r**)

  $I = 0$ ▷ Initialization
  for $i = 1 : n$ **do**
    for $j = 1 : m$ **do**
      $M_{j,i} = r_i$
    end for
  end for

  repeat
    for $j = 1 : m$ **do** ▷ Step 1: Check messages
      for $i \in B_j$ **do**
        $E_{j,i} = \log\left(\frac{1+\prod_{i' \in B_j, i' \neq i} \tanh(M_{j,i'}/2)}{1-\prod_{i' \in B_j, i' \neq i} \tanh(M_{j,i'}/2)}\right)$
      end for
    end for

    for $i = 1 : n$ **do** ▷ Test
      $L_i = \sum_{j \in A_i} E_{j,i} + r_i$
      $z_i = \begin{cases} 1, & L_i \leq 0 \\ 0, & L_i > 0. \end{cases}$
    end for
    if $I = I_{\max}$ or $H\mathbf{z}^T = 0$ **then**
      Finished
    else
      for $i = 1 : n$ **do** ▷ Step 2: Bit messages
        for $j \in A_i$ **do**
          $M_{j,i} = \sum_{j' \in A_i, j' \neq j} E_{j',i} + r_i$
        end for
      end for
      $I = I + 1$
    end if
  until Finished
end procedure

FIGURE 3.3 – Illustration de l'algorithme de décodage "Sum-Product[41]

contrôle d'informations dont il dispose déjà, le message du i-ème nœud au j check-node est la somme en (3.13) sans le composant $E_{j,i}$, je ne reçu du j-ème nœud de contrôle :

$$M_{j,i} = \sum_{j' \in A_i, j' \neq j} E_{j',i} + r_i \tag{3.14}$$

Une illustration de l'algorithme "sum-product" est montré dans la figure 3.3.

Exemple d'application :

Soit le code LDPC donné par sa matrice de contrôle :

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{3.15}$$

soit le mot code suivant :

$$c = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

est envoyé via un canal AWGN BPSK avec $E_s/N_0 = 1,25$ (ou 0,9691dB) et le signal reçu est :

$$y = \begin{bmatrix} -0.1 & 0.5 & -0.8 & 1.0 & -0.7 & 0.5 \end{bmatrix}$$

Si une décision difficile est prise sans décodage, il y a deux bits d'erreur dans cette vecteur reçu, les bits 1 et 6. Pour un canal AWGN les LLR a priori sont donnés par :

$$r_i = 4y_i \frac{E_s}{N_0}$$

Nous avons donc :

$$r = \begin{bmatrix} -0.5 & 2.5 & -4.0 & 5.0 & -3.5 & 2.5 \end{bmatrix}$$

En appliquant l'algorithme présenté au-dessus nous allons avoir les résultats suivants :

Itération 1 :

$r = \begin{bmatrix} -0.5 & 2.5 & -4.0 & 5.0 & -3.5 & 2.5 \end{bmatrix}$

$$E = \begin{bmatrix} 2.4217 & -0.4930 & . & -0.4217 & . & . \\ . & 3.0265 & -2.1892 & . & -2.3001 & . \\ -2.1892 & . & . & . & -0.4217 & 0.4696 \\ . & . & 2.4217 & -2.3001 & . & -3.6869 \end{bmatrix}$$

$L = \begin{bmatrix} -0.2676 & 5.0334 & -3.7676 & 2.2783 & -6.2217 & -0.7173 \end{bmatrix}$

$z = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

$Hz^T = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \Longrightarrow continue$

$$M = \begin{bmatrix} -2.6892 & 5.5265 & . & 2.6999 & . & . \\ . & 2.0070 & -1.5783 & . & -3.9217 & . \\ 1.9217 & . & . & . & -5.8001 & -1.1869 \\ . & . & -6.1892 & 4.5783 & .2.9696 & \end{bmatrix}$$

Itération 2 :

$$E = \begin{bmatrix} 2.6426 & -2.0060 & . & -2.6326 & . & . \\ . & 1.4907 & -1.8721 & . & -1.1041 & . \\ 1.1779 & . & . & . & -0.8388 & -1.9016 \\ . & . & 2.7877 & -2.9305 & \Delta & -4.3963 \end{bmatrix}$$

$L = \begin{bmatrix} 3.3206 & 1.9848 & -3.0845 & -0.5630 & -5.4429 & -3.7979 \end{bmatrix}$

$z = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

$Hz^T = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \Longrightarrow continue$

$$M = \begin{bmatrix} 0.6779 & 3.9907 & . & 2.0695 & . & . \\ . & 0.4940 & -1.2123 & . & -4.3388 & . \\ 2.1426 & . & . & . & -4.6041 & -1.8963 \\ . & . & -5.8721 & 2.3674 & .0.5984 & \end{bmatrix}$$

Itération 3 :

$$E = \begin{bmatrix} 1.9352 & 0.5180 & .0.6515 & . & . & \\ . & 1.1733 & -0.4808 & . & -0.2637 & . \\ 1.8332 & . & . & . & -1.3362 & -2.0620 \\ . & . & 0.4912 & -0.5948 & . & -2.3381 \end{bmatrix}$$

$$L = \begin{bmatrix} 3.2684 & 4.1912 & -3.9896 & 5.0567 & -5.0999 & -1.9001 \end{bmatrix}$$

$$z = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$Hz^T = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \Longrightarrow termier$$

Le décodeur "Sum-Product" converge vers le mot de code correct après trois itérations.

3.4 Algorithme Min-Sum

En raison de certaines complexités d'implémentation, L'algorithme Sum-Product (SPA) peut être modifié pour réduire la complexité de l'implémentation du décodeur.
L'algorithme Min-Sum ou Propagation des croyances (BP) a été établi pour réduire la complexité de l'algorithme Log-BP (SPA) en utilisant une approximation de l'équation 3.12. En effet, dans l'algorithme Min-Sum, la mise à jour d'un CN est effectué par l'équation 3.13.
L'algorithme Min-Sum permet donc de simplifier le décodeur en éliminant les tableaux de correspondances nécessaires à la mise en œuvre des fonctions exponentielles et logarithmes et en minimisant le nombre d'opérations arithmétiques.

$$E_{j,i} = 2tanh^{-1}\Big(\prod_{i' \in B_j, i' \neq i} tanh(M_{j,i'/2})\Big) \tag{3.16}$$

Pour pouvoir remplacer le produit par la somme. Tout d'abord pour plus de simplicité dans l'écriture nous écrirons :

$$\prod_{i' \in B_j, i' \neq i} \equiv \prod_{i'}$$

Dans le reste de cette section. $M_{j,i'}$ doit s'écrire :

$$M_{j,i'} = \alpha_{j,i'} \beta_{j,i'}$$

où :

$$\alpha_{j,i'} = sign(M_{j,i'})$$
$$\beta_{j,i'} = |M_{j,i'}|$$

En utilisant cette notation, nous allons avoir :

$$\prod_{i'} tanh(M_{j,i'}/2) = \prod_{i'} \alpha_{j,i'} \prod_{i'} tanh(\beta_{j,i'}/2) \tag{3.17}$$

L'équation 3.12 devient :

$$E_{j,i'} = \prod_{i'} \alpha_{j,i'} 2tanh^{-1}(\prod_{i'} tanh(\beta_{j,i'}/2)) \tag{3.18}$$

Cette dernière équation peut être maintenant réarrangée pour remplacer le produit par la somme :

$$E_{j,i} = \prod_{i'} \alpha_{j,i'} 2tanh^{-1} log^{-1} log(\prod_{i'} tanh(\beta_{j,i'}/2)) \tag{3.19}$$

$$= \prod_{i'} \alpha_{j,i'} 2tanh^{-1} log^{-1}(\sum_{i'} log(tanh(\beta_{j,i'}/2)) \tag{3.20}$$

Ensuite, nous définissons :

$$\phi(x) = log(tanh(\frac{x}{2})) = log\ \frac{e^x + 1}{e^x - 1} \tag{3.21}$$

Notez également que :

$$\phi(\phi(x)) = log\ \frac{e^{\phi(x)} + 1}{e^{\phi(x)} - 1} = x$$

Nous avons $\phi^{-1} = \phi \Longrightarrow \phi$ est bijective.

Finalement, l'équation 3.20 devient :

$$E_{j,i} = (\prod_{i'} \alpha_{j,i'})\ \phi\ (\sum_{i'} \phi(\beta_{j,i'})) \tag{3.22}$$

Le graphe de la fonction $\phi(x)$ est représenté sur la figure 3.4. Puisque le terme correspon-

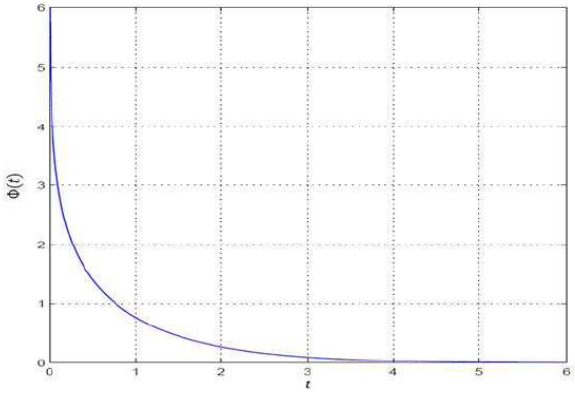

FIGURE 3.4 – représentation graphique de $\phi(x)$

dant au plus petit domine, nous avons :

$$\phi\ (\sum_{i'}\phi(\beta_{j,i'})) \approx \phi\ (\phi(min(\beta_{j,i'}))) = min(\beta_{j,i'}) \tag{3.23}$$

Le produit des signes peut être calculer par des opération "XOR" appliquer aux bits de signes sur chaque $M_{j,i'}$, tandis que la fonction $\phi(x)$ serait facilement implémentée en se réfèrent a sa tabulation. En d'autres termes, l'algorithme Min-Sum, simplifie le calculde de l'équation 3.12, sachant que le terme qui domine le produit correspond à la plus petite valeur de $M_{j,i'}$ et la relation devient approximativement égale à :

$$E_{j,i} = \prod_{i'} sign\ (M_{j,i'}) \times Min\ |M_{j,i'}| \tag{3.24}$$

Finalement, cette formule permet d'exécuter uniquement l'opération de l'addition et de rechercher les minimums.

Exemple d'application :

pour bien comprendre les étapes voici un organigramme de l'algorithme de décodage "Min-Sum" présenté dans la figure 3.5. Soit le code LDPC donné par sa matrice de contrôle

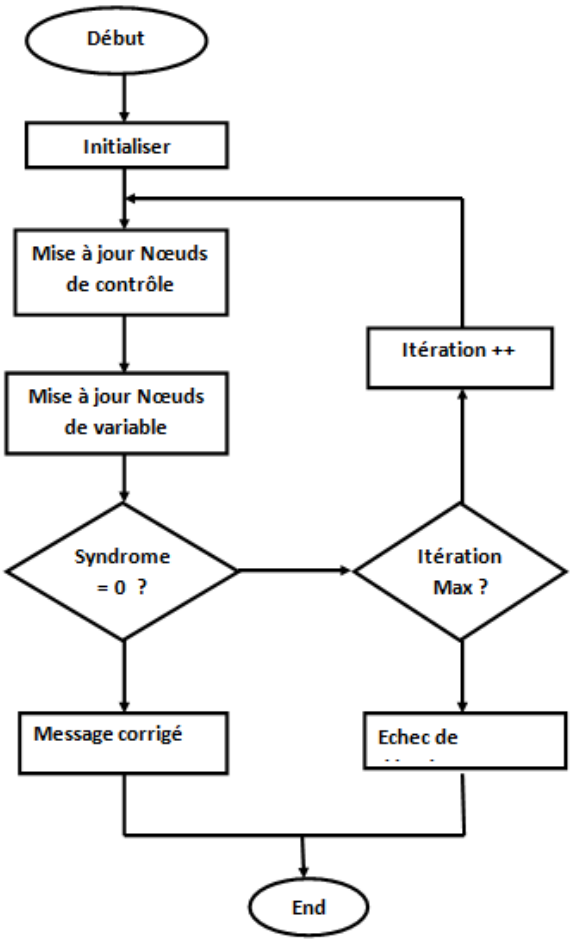

FIGURE 3.5 – Organigramme de l'algorithme de décodage

H suivante :

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{3.25}$$

Considérons une trame [1 0 1 1 0 0 1] envoyée sur le canal à l'aide de la modulation bi-polaire. Soit la trame envoyée est [-1 1 -1 -1 1 1 -1].

Supposons que le canal ajoute un bruit AWGN avec un variance de $0,5$ pour que les LLR de la trame reçue soient $[-8\ -6\ -11\ -5\ 8\ 9\ -12]$. Le deuxième reçu l'échantillon a une erreur. Les étapes de décodage sont illustrées sur le graphe de Tanner par la figure 3.6.

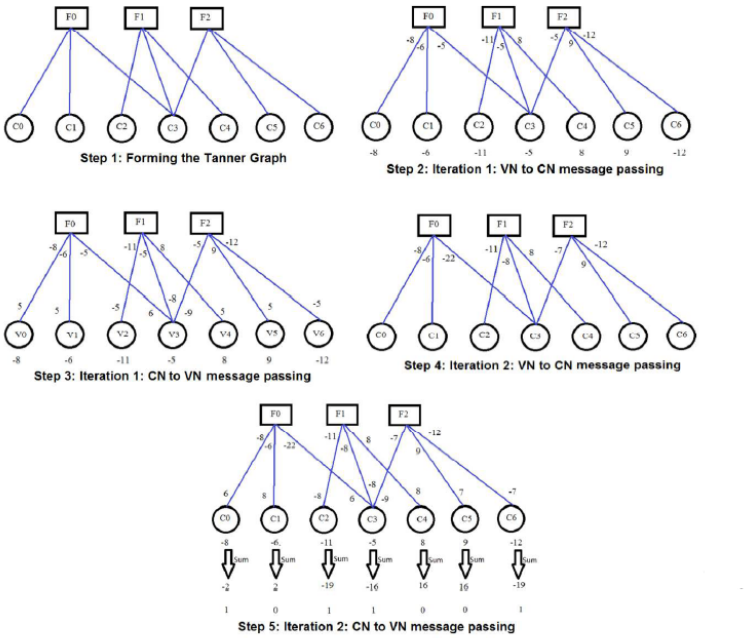

FIGURE 3.6 – Exemple d'algorithme Min-Sum expliqué avec un graphique Tanner où l'erreur la correction se produit dans la deuxième itération[35].

3.4.1 Avantages de l'algorithme Min-Sum

1) L'algorithme Min-Sum se rapproche de l'algorithme de Sum-Product avec une vérification simple des opérations de mise à jour de nœud, ce qui réduit considérablement la complexité de la mise en œuvre VLSI.

2) Les calculs exponentiels ou logarithmiques complexes sont évités, par conséquence réduction de la complexité de l'implémentation sur le matériel.

3.4.2 Inconvénients de l'algorithme Min-Sum

Le nombre d'itérations requis est souvent supérieur au SPA, et donc on perd en performances et vitesse de convergence.

3.5 Étude des performances des codes LDPC

Dans cette section, une évaluation des performances des codes LDPC est présentée. Pour cela, nous avons utilisé le logiciel MATLAB pour programmer toute la chaîne de simulation et cela à cause de la simplicité de programmation sous MATLAB. Mais au cours des simulations, il s'est avéré nécessaire de basculer vers le langage C afin de faire quelques simulations et cela pour des raisons qu'on définira plus tard.

L'organisation du reste de la section est donnée sous la forme suivante :

Présentation de la chaîne de simulation.
Définition des conditions de simulation et les hypothèses.
Présentation des résultats.
Interprétations et conclusions.

3.5.1 Présentation de la chaîne de simulation

Le modèle de simulation est schématisé dans la Figure 3.7. Un générateur pseudo aléatoire génère des blocs de données binaires de taille K, ces blocs sont codés par un encodeur LDPC, puis modulés et transmis via un canal AWGN. A la réception, le processus de détection est réalisé pour pouvoir estimer les bits transmis, on compare les bits reçus et les bits transmis afin de déterminer le taux d'erreur binaire BER (Bit Error Rate en anglais), qu'on le définit par la relation suivante :

$$BER = \frac{Nombre\ de\ bits\ errones}{Nombre\ de\ bits\ transmis} \tag{3.26}$$

Les principaux paramètres et hypothèses utilisés dans la chaîne de simulation sont résumés dans le Tableau 3.1 :

3.5.2 Comparaison entre les différents algorithmes de décodage LDPC

La Figures 3.8 représente les différentes implémentations de l'algorithme de décodage BP (Belief Propagation) avec un processus de décodage de 10 itérations et un code LDPC irrégulier de taille N = 1024 et de rendement R= 1/2.

Paramètres	Valeurs et caractéristiques
Types du code LDPC	Code irrégulier
La matrice H	Pseudo aléatoire, sans cycle de longueur 4
Algorithme de décodage	Algorithme BP
Le poids des colonnes	$w_c = 5$
Type du canal	Canal Gaussien à bruit blanc additif (AWGN)
Estimation de la variance du bruit σ^2	Parfaite
Type de modulation	BPSK
Le critère d'évaluation des performances	BER en fonction de E_0/N_0

TABLE 3.1 – Les paramètres de simulation utilisés pour évaluer les performances des codes LDPC.

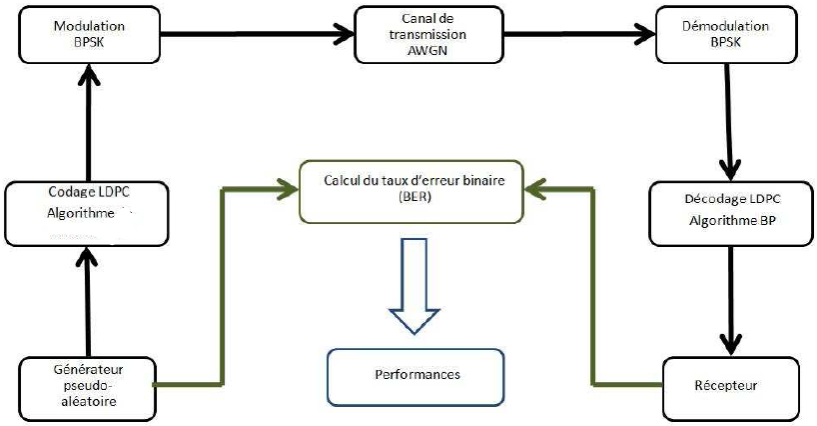

FIGURE 3.7 – Le modèle de simulation utilisé pour l'évaluation des performances des codes LDPC.

Comme attendu, on voit bien que le décodage Hard donne des performances médiocres par rapport au décodage Soft, d'ailleurs on remarque que le décodage Hard présente une région de non convergence d'environ 7.8 dB et un seuil de convergence situé à 6 dB, tandis que dans le décodage Soft et pour le cas de l'algorithme Log-Domain, la région de non convergence est 2 dB uniquement et un seuil de convergence situé à 1.5 dB. Si on se fixe comme référence à un BER = 10^{-5}, on voit bien que le décodage Hard apporte uniquement un gain de 1 dB par rapport au cas sans codage, alors que le cas Soft apporte au moins un gain de 5.5 dB. Delà, on peut dire que tout ces résultats confirment la raison pour laquelle tous les standards introduisant les codes LDPC utilisent le décodage Soft au lieu du décodage Hard.

Pour le cas Soft, Les deux courbes données en vert et en bleu dans la Figure 3.8, permettent d'évaluer l'influence de l'approximation donnée par la relation (3.24). On voit bien que pour un BER = 10^{-4}, ces deux algorithmes de décodage (Log-Domain et Min-Sum) donnent respectivement des gains de 4.5 dB et 4.25 dB par rapport au cas sans codage. En comparant les performances de ces deux algorithmes, nous avons constaté que

l'algorithme sous-optimal Min-Sum est une bonne approximation de l'algorithme Log Domain. Nous avons aussi constaté, au cours des simulations, que l'exécution de l'algorithme Min-Sum met beaucoup moins de temps par rapport à l'exécution de l'algorithme Log Domain.

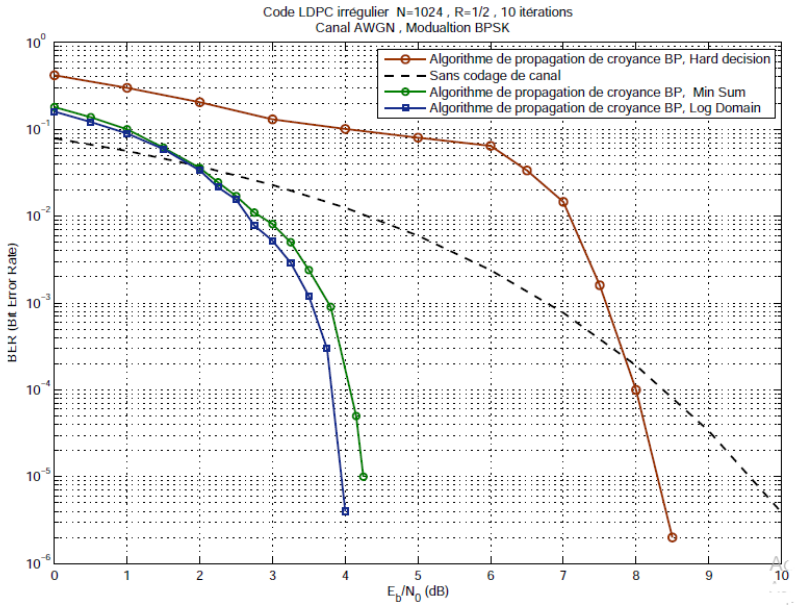

FIGURE 3.8 – Comparaison des performances entre les différents algorithmes de décodage pour l'algorithme BP.

3.5.3 Influence de la taille du code sur les performances

Le but de cette simulation est de voir l'influence de la taille N sur les performances des codes LDPC. Pour cela, on fixe R=1/2 et le nombre d'itérations du processus de décodage à 10.

D'après les résultats présentés dans la Figure 3.9, on remarque qu'avec l'augmentation de N(taille du mot code), il résulte une amélioration de performances, par exemple pour un BER égal à 10^{-3}, le code LDPC irrégulier de taille $N = 1024$ apporte un gain de $0.8dB$ par rapport à celui de $N = 200$, et de $0.3dB$ par rapport à celui de $N = 500$. On remarque aussi qu'en augmentant N, le seuil de convergence reste presque constant et c'est les pentes des graphes qui augmentent avec l'augmentation de N. Par contre dans les Figures 3.10 et 3.11 qui correspondent respectivement au décodage Soft Log-Domain et Min-Sum, on voit clairement l'amélioration apportée par l'augmentation de N. Par exemple la Figure 3.10 montre que pour un BER égal à 10^{-5}, le code LDPC irrégulier de taille $N = 1024$ apporte un gain de $0.6dB$ par rapport à celui de $N = 500$ et un gain de $1.2dB$ par rapport à celui de $N = 200$. D'autre part, nous avons constaté que plus N est grand plus le seuil de convergence décale vers les E_b/N_0 faibles et les pentes des graphes deviennent encore plus grandes. Par exemple pour $N = 1024$ et $E_b/N_0 = 4.5dB$, le taux d'erreur binaire est de l'ordre de 10^{-8}, cela signifie 1 bit erroné parmi cent millions de bits transmis. Ce résultat montre clairement la puissance des codes LDPC.

En augmentant encore plus la taille du code N ($N \geq 1000$), nous avons constaté que le logiciel MATLAB prend beaucoup de temps, car il manipule des matrices de taille très grande [1]. Pour contourner ce problème, il s'est avéré nécessaire de passer au langage C.

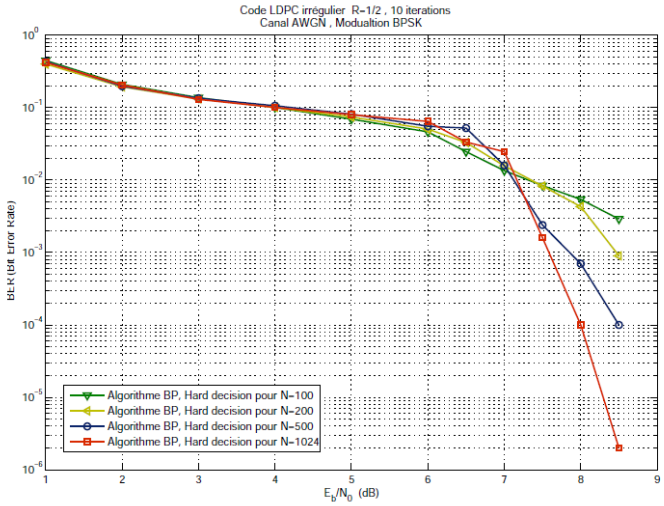

FIGURE 3.9 – Influence de la taille du code sur les performances pour un décodage Hard.

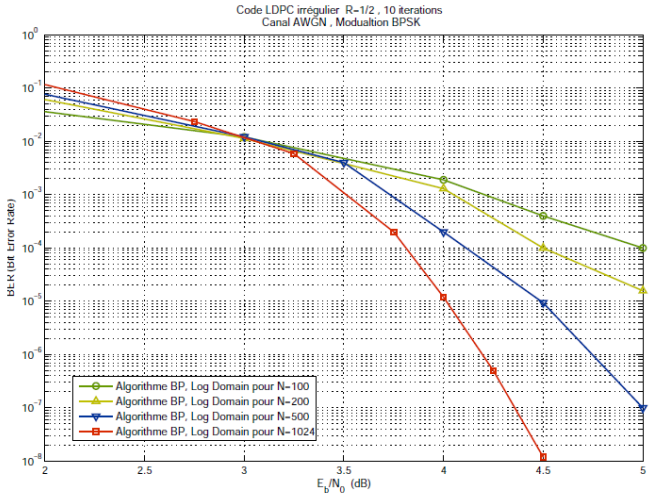

FIGURE 3.10 – Influence de la taille du code sur les performances pour l'algorithme de décodage Log-Domain.

Pour cela nous avons utilisé des fonctions prédéfinies dans la bibliothèque de Radford Neal disponibles sur son site [36].

La Figure 3.12 montre le cas des codes LDPC de taille $N = 1000, 5000$ *et* 10000 décodés par l'algorithme Log-Domain (10 itérations). Dans cette figure, on voit encore plus clairement l'effet de la taille du code N sur les performances. D'ailleurs, on voit bien que pour $N = 10000$, le taux d'erreur binaire $BER = 10^{-7}$ uniquement pour $E_b/N_0 = 1.8dB$. Ces résultats confirment la puissance des codes LDPC irréguliers surtout pour N

1. A titre d'exemple, dans le cas N=1000 et en utilisant un ordinateur Core 2 Duo 3 Go de RAM, les simulations ont mis tout un week-end pour obtenir un point de $BER = 10^{-6}$.

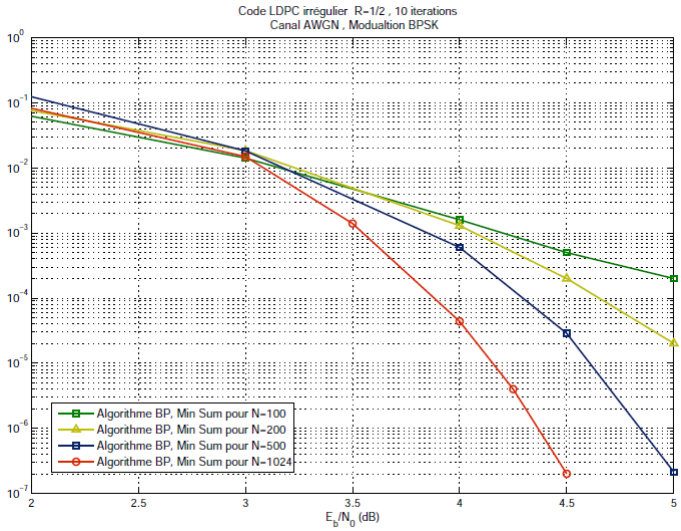

FIGURE 3.11 – Influence de la taille du code sur les performances pour l'algorithme de décodage Min-Sum.

très grand. Les résultats obtenus dans cette partie justifient pourquoi tous les standards

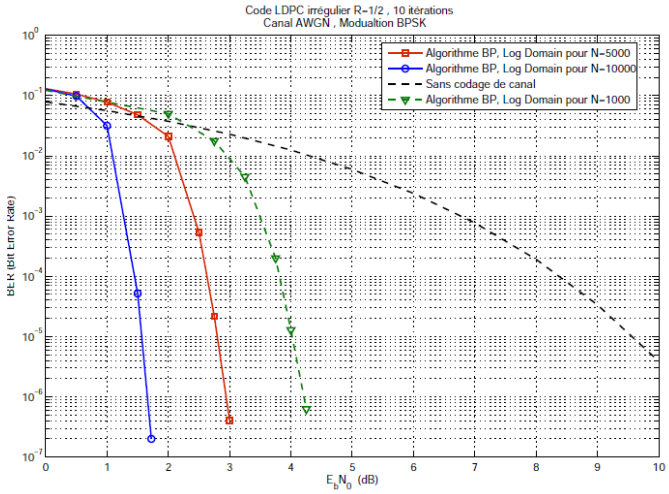

FIGURE 3.12 – Influence de la taille du code sur les performances pour l'algorithme de décodage Log-Domain.

de communication utilisant des codes LDPC choisissent des tailles N très grandes, par exemple $N = 68000$ pour le standard DVB-S2 et $N = 8176$ pour le standard CCSDS.

3.5.4 Influence du nombre d'itérations du processus de décodage sur les performances

Dans cette simulation, nous étudions l'effet du nombre d'itérations sur les performances. Pour cela, nous allons considérer l'algorithme Min-Sum et cela à cause de sa rapidité d'exécution par rapport à l'algorithme Log-Domain. La Figure 3.13 montre l'effet du nombre d'itérations sur les performances pour $N = 200$. On remarque qu'il y a une

amélioration à chaque fois qu'on augmente le nombre d'itérations, par exemple entre 2 et 10 itérations, il y a un gain de 0.8 dB pour un $BER = 10^{-2}$. Mais à partir de 15 à 20 itérations, on ne voit pas d'amélioration, c'est-à-dire que les courbes commencent à se saturer.

La Figure 3.14 montre un comportement identique pour $N = 1024$, c'est-à-dire que l'augmentation du nombre d'itérations améliore les performances du décodage jusqu'à un nombre maximum d'itérations où l'algorithme de décodage converge.

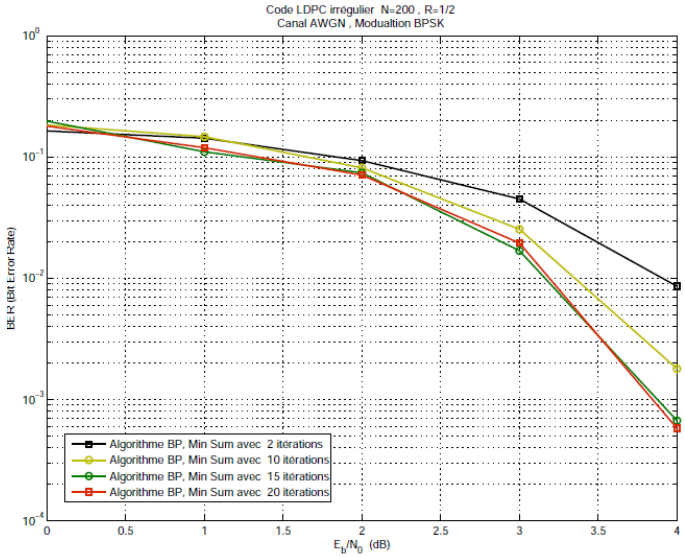

FIGURE 3.13 – Influence du nombre d'itérations sur les performances des codes LDPC pour N=200 et un décodage Min-Sum .

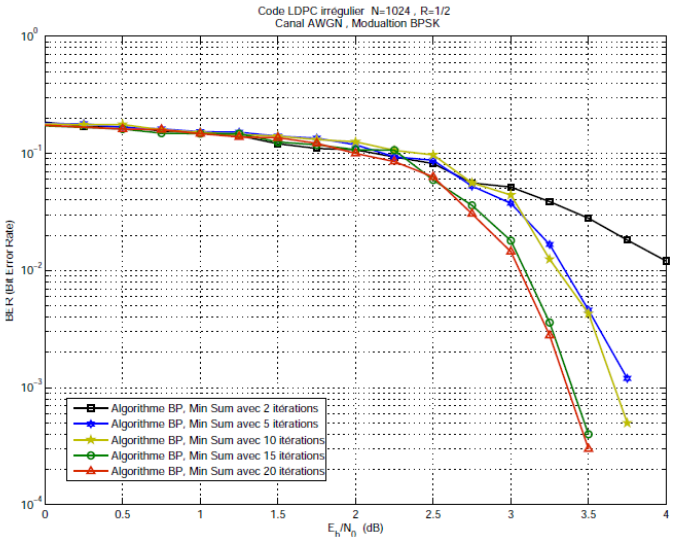

FIGURE 3.14 – Influence du nombre d'itérations sur les performances des codes LDPC pour N=1024 et un décodage Min-Sum .

3.5.5 Influence du rendement de codage sur les performances

Dans cette dernière partie, nous étudions l'effet du rendement de codage R sur les performances des codes LDPC irréguliers. Pour cela, on prend *N=500* et on considère un décodage Min-Sum avec *20* itérations.

La Figure 3.15 montre une comparaison des performances entre les codes LDPC irréguliers de rendement R égal respectivement à 1/4, 1/2 et 4/5 en terme de *BER* en fonction du rapport signal sur bruit *SNR*. D'après les résultats obtenus, on voit bien que le code LDPC irrégulier de rendement $R = 1/4$ est le plus performant, puis vient celui avec $R = 1/2$ et enfin $R = 4/5$. Pour un $BER = 10^{-3}$, le gain apporté en SNR (et non

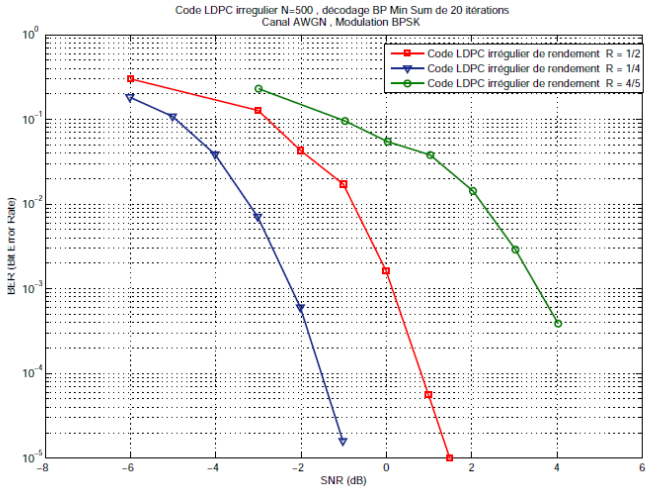

FIGURE 3.15 – Influence du rendement R sur les performances des codes LDPC (BER = f(SNR)).

pas en E_b/N_0) par le code de rendement $R = 1/4$ est de 2 dB par rapport à celui de $R = 1/2$ et de 5.9 dB par rapport à celui de $R = 4/5$. On peut interpréter ces résultats par le fait que pour $R = 1/4$, on associe 3 bits de redondance à chaque bit de données, et pour R=1/2, à chaque bit de données est associé un bit de redondance. Alors que, pour R=4/5, pour 4 bits de données, on associe 1 seul bit de redondance. Delà, on constate que plus on augmente le nombre de bits redondants, plus les performances sont meilleures. Mais d'un côté, on perd en débit et en efficacité spectrale du système.

On voit bien que le critère BER en fonction du SNR ne montre pas cet inconvénient. C'est pour cette raison qu'on préfère le critère BER en fonction de E_b/N_0.

On rappelle que :

$$\frac{E_b}{N_0} = \frac{SNR}{R} \tag{3.27}$$

La Figure 3.16 montre une comparaison en terme de BER en fonction de E_b/N_0. Cette fois-ci, c'est le code LDPC irrégulier de rendement $R = 1/2$ qui montre de meilleures performances. D'après ces résultats, il s'avère qu'un code LDPC de rendement $R = 1/2$ représente un bon compromis entre la protection contre les erreurs et l'efficacité spectrale, cela veut dire qu'un seul bit redondant pour chaque bit informatif est suffisant pour assurer

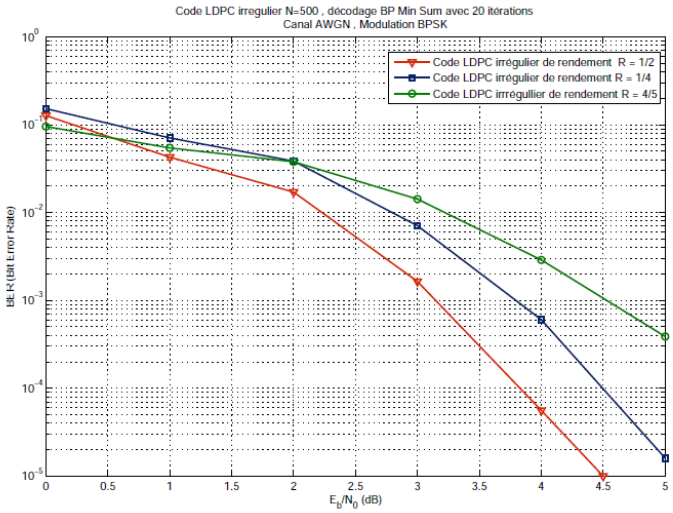

FIGURE 3.16 – Influence du rendement R sur les performances des codes LDPC (BER=f(E_b/N_0)).

une bonne protection contre les erreurs et une bonne efficacité spectrale du système. Ce résultat affirme pourquoi tous les standards utilisant les codes LDPC n'introduisent jamais les codes de rendement $R = 1/4$ (Voir le Tableau 3.2).

Le standard	La technique utilisée	La taille N	Le rendement R
DVB-S2	Repeat-Accumulate	$N = 68000$	$R = 1/2$
CCSDS	Quasi-Cyclique Codes	$N = 8176$	$R = 7/8$
802.16e WIMAX	Repeat-Accumulate	$N = 2304$	R variable

TABLE 3.2 – Exemples de techniques de codage utilisées par quelques standards de télécommunication.

3.6 Conclusion

les techniques de codage et décodages LDPC ont fait l'objet d'une étude détaillé dans ce chapitre. L'algorithme de propagation de croyance et ses algorithmes dérivés (sous optimaux) sont développés ainsi que l'algorithme de Radford Neal. Après avoir présenté tous ces algorithmes, les simulations permettant d'évaluer les performances des codes LDPC dans un canal gaussien sot effectués .les résultats obtenues nous permettent de tirer les conclusions suivantes :

• Le décodage Soft est beaucoup plus performant que le décodage Hard.
• L'algorithme sous optimal Min-Sum représente une très bonne approximation de l'algorithme de décodage Log-Domain.
• L'implémentation de l'algorithme Min-Sum dans des applications en temps réel représente une bonne solution, contrairement à l'algorithme Log-Domain.

• L'augmentation de la taille du ode LDPC entraîne une amélioration de performances, cette amélioration est remarquable surtout pour N ¸ 1000.
• L'augmentation du nombre d'itérations améliore les performances de décodage jusqu'à un nombre d'itérations maximum où l'algorithme de décodage converge.
• Plus R est petit plus la protection contre les erreurs est meilleure, mais à partir d'un certain seuil, le système commence à perdre en efficacité spectrale.

ces résultats démontrent la puissance des codes LDPC irréguliers sur tout pour les grades tailles. Leur puissance ainsi que leur efficacité sont à l'origine de la multitude d'applications de ces codes dans les systèmes de communication émergeant.

Chapitre 4

Architecture du décodeur LDPC

Ce chapitre, est une introduction de l'implémentation de notre algorithme Min-Sum sur FPGA et dans lequel nous allons voir et discuter de la conception du décodeur sur le matériel. Pour cela, Nous allons commencer par présenter les différentes types d'architectures pour le décodage des codes LDPC afin de connaitre les choix de conception d'un décodeur LDPC qui va nous aider dans l'implémentation, en suite nous allons présenter et expliquer le rôle des deux unités principales Check Node Unit *(CNU)* et Variable Node Unit *(VNU)* du décodeur LDPC avant de finir par une conclusion.

4.1 Architecture des codes LDPC

Cette section traite des différents aspects d'implementations d'architectures pour les décodeurs LDPC, le décodage LDPC présente une particularité inhérente qui est le parallélisme. Chaque noeud de contrôle et de donnée pourront être exécuté indépendamment. cette caractéristique peut être exploré différemment selon l'ordonnancement choisie. l'indépendance de traitement des noeuds représente un grand avantage pour l'implementation des codes LDPC. Quelques éléments sont éventuellement nécessaire pour la description de n'importe quel architecture LDPC :

• Unités fonctionnels servants aux traitement des noeuds de données et des noeuds de contrôles.
• Réseaux de permutation pour supporter le transport des messages entre les noeuds de données et les noeuds de contrôle en respectant le graphe de *Tanner* du code en question.
• Mémoires pour stocker les entrées LLRs et les messages échangés par les noeuds de données et les noeuds de contrôle durant le processus de décodage itératif.

Cependant, chaque élément dépend directement des contraintes relatives aux standards sur les quelles l'architecture sera portée. dans la majorité des cas, la flexibilité est l'une des plus importante critère lorsqu'un décodage à haut débit est exigé. Par exemple, un décodeur à ultra large-bande(UWB) exige un maximum de débit de $1,024 Mb/s$ tandis qu'un GMR-1 décodeur exige un débit inférieur à $1 Mb/s$, mais avec un maximum de longueur du code de 64,800 bits. cependant différentes solutions ont dérivés pour chacun

de ses standard, le coût et la faisabilité de chaque solution dépendra des exigences de stockage, du débit et de la flexibilité du décodeur.

4.2 Choix de conception du décodeur

La conception architecturale qui reflète les possibilités d'implémentation d'un décodeur LDPC peut être divisée en plusieurs niveaux. du point de vue "high level" ou le plus haut niveau d'abstraction, le nombre de branche du graphe de *Tanner* traité par cycle d'horloge détermine l'architecture de base, En effet, au plus haut niveau d'abstraction, il existe seulement trois possibilités de base :

• Décodeur entièrement parallèle : toute les branches du graphe de *Tanner* sont traitées en un seul cycle d'horloge.
• Décodeur partiellement parallèle : P branches sont traitées en un seul cycle d'horloge.
• Décodeur à architecture série : un noeud fonctionnel est traité par cycle. Celle-ci pourrait aboutir à des changement, toute fois, le nombre de branche traitée reste très petit.

Se référant à ces techniques de bases, différentes solutions existent pour la distribution, l'acheminement, le stockage et la réalisation des unités de traitement. Ces trois techniques de base sont présentées dans la section suivante.

4.2.1 Architecture entièrement parallèle

Dans une réalisation d'un décodeur entièrement parallèle, le graphe de Tanner est directement câblé sur hardware. tout les noeuds de données et noeuds de contrôlé sont instanciés et la connections entre eux est câblée. ce type a été montré dans[44] pour un code LDPC irrégulier $R = 1/2$ avec un mot de code de longueur N = 1024. Le chemin de donnée d'une architecture entièrement parallèle d'un décodeur LDPC est montrée dans la figure 4.1.
Cette figure montre deux nœuds fonctionnels, représentant un nœud de données de degré 3, et un nœud de contrôle de degré 4 respectivement. On constate que les deux nœuds sont implémentés de façon parallèle et acceptent nécessairement en entrée la totalité des messages autrement dit tout les messages sont traités à la fois. Avec ce chemin de donnée nous avons besoin d'un seul cycle d'horloge pour une seul itération. le chemin critique de ce type d'architecture est très long, du moment où les messages passent des nœuds de contrôle aux nœuds de données et vice versa. le débit résultant du décodeur dans [44] est de 1 Gbit/s en considérant 64 itérations. L'architecture parallèle est la plus rapide de toutes les autres architectures puisque les nœuds, par conséquent, les branches du graphe de Tanner sont traités simultanément. Ainsi l'avantage d'une implémentation parallèle est le haut débit. En particulier, dans les systèmes de transmission optique, cette avantage est de grand intérêt où un code LDPC ayant un taux ($R > 0.8$) pourrait être établis. le problème majeur pour l'implémentation Hardware est la complexité du routage qui

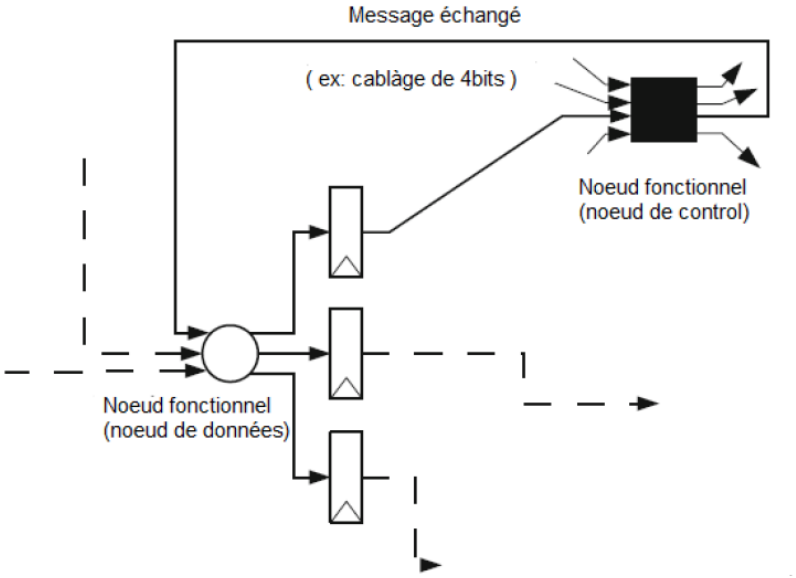

FIGURE 4.1 – Chemin de donnée d'une architecture entièrement parallèle. cette figure est dérivée de[37]

devient prohibitif pour de larges mots de code. Par ailleurs, cette approche est seulement convenable pour les applications ou un seul code fixe est utilisé.

4.2.2 Architecture partiellement parallèle

Un décodeur partiellement parallèle traite P branches en un seul cycle d'horloge, où P est très petit par rapport à la longueur du bloc N. Souvent ce type d'architecture est nécessaire pour fournir plus de flexibilité au décodeur, où seulement un certain sous-ensemble de nœuds fonctionnels sont instanciés. Les nœuds de contrôle et nœuds de données sont traités essentiellement sur ces unités de traitement. un nœud de traitement peut être instancié pour traiter une seul ou plusieurs branche à la fois(tout dépend du degré de distribution) par cycle d'horloge, Par conséquent, bien que le nombre de branches P traités et le nombre de nœuds fonctionnels instancié peut différer, le débit est seulement déterminé par le nombre de branche P traités par cycle d'horloge. La principale question qui se pose pour la cartographie est l'allocation d'un quelconque nœud dans le graphe de *Tanner* qui soit physiquement réalisée en Hardware, autrement la question qui se pose est quel nœud devant être traité sur quelle unité fonctionnelle et à quel instant ? une des procédure de mapping pour le cas de l'architecture semi-parallèle est montrée dans la figure 4.2, celle-ci considère qu'une chaque unité fonctionnelle traite une seule branche par cycle d'horloge, et c'est d'ailleurs la procédure pour laquelle nous avons opté(citer des ref pour code 21x28). Les messages échangés entre les unités fonctionnelles noeuds de données (VNFU) et noeuds de contrôle (VNCN) devraient ètre stockés en mémoires (montrer la figure de la RAM). Un réseau à permutation doit être instauré pour implémenter les connectivités désirées pour un graphe de *Tanner* donné. Cela nécessite des mémoires additionnel afin de stocker le modèle de connectivité. L'avantage majeure d'une architecture semi-parallèle est la possibilité de fournir une flexibilité, en produisant une

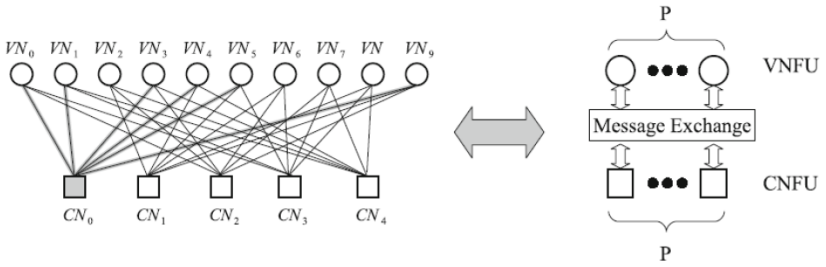

FIGURE 4.2 – Mapping du décodeur semi-parallèle à partir du graphe de *Tanner*

architecture générique pouvant adopter des codes de longueur de bloc de code rate variable. Tandis que son inconvénient est la limitation du débit qui est plus petit par rapport à l'implémentation d'un décodeur à architecture entièrement parallèle.

4.2.3 Architecture série

Parmi tout les types d'architectures, l'architecture série est la plus simple. Seulement un bloc fonctionnel pour chacun des nœud de donnée et des nœud de contrôle est instancié avec un traitement séquentielle de tout les nœuds de données et nœuds de contrôle. les messages traités ainsi que la structure du code sont stockés en mémoires. l'avantage d'une telle architecture est la simplicité de distribution des messages, sa flexibilité, tandis que son inconvénient est le faible débit.

Une architecture série d'un décodeur LDPC est présenté Dans [36]. Pour augmenter le débit, on a suggéré de mettre en parallèle plusieurs décodeur, mais cela se traduit par une grande latence et de grandes exigence de mémoires. l'instantiation de plusieurs décodeur LDPC en parallèle est toujours possible, mais la surface global occupée par l'architecture augmente linéairement avec le nombre de décodeurs instanciés, toute fois cette solution est rarement choisie dans le cas où la latence serait un facteur critique.

Dans la suite, seulement les architectures parallèles seront prises en compte puisqu'ils fournissent une meilleur flexibilité par rapport à une réalisation entièrement parallèle, et débit plus élevé que l'implémentation série. La question de flexibilité détermine le style d'implémentation des noeuds fonctionnels qui sera présenté dans la section suivante.

4.3 Comparaison des deux conceptions

Malgré les nombreux recherche dans le thème des codes LDPC et leur implémentation, la conception d'un décodeur à haut débit avec faible consommation d'énergie et une petite surface de silicone est encore un défi.

Étant donné que le nombre d'opérations réalisables par cycle est plus grand avec les décodeurs entièrement parallèle, leur efficacité énergétique et leur débit sont théoriquement meilleur [18,51]. La figure 4.3 montre le débit de quelques conceptions (mesuré ou

post-layout implementation) en fonction de l'année pour les deux types de décodeurs, les graduations sur le coté droit indiquent l'exigence du débit maximale pour les cinq normes de communication les plus connus. Tout les décodeurs des Normes standards DVB-S2, 802.16e et 802.11n qui nécessitent du matériel reconfigurable pour soutenir différents rendements et longueur de codes sont de types semi-parallèle.

Bien qu'il n'existe pas de nombreuse implémentations de décodeurs parallèles déclarées, leur débit est en générale supérieur à celui des décodeurs semi-parallèle et séries. Mais ces décodeurs parallèles nécessitent un grand nombre de noeuds de traitements, par conséquent, ils souffrent d'une dominance du câblage et de liaison au niveau hardware.

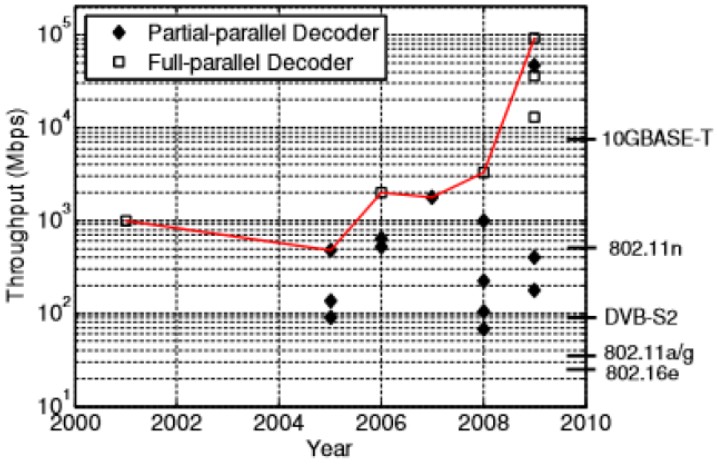

FIGURE 4.3 – Exigence et variation du débit pour quelques décodeur LDPC parallèle et semi-parallèle en fonction de l'année[38]

4.4 Architecture non stratifiée

4.4.1 Architecture de base du décodeur :

La figure 4.4 montre un système comprenant un décodeur LDPC. Il comprend généralement un émetteur et un récepteur. Le récepteur comprend un port d'E / S, un processeur, une mémoire et un décodeur LDPC. L'émetteur transmet le signal codé en utilisant le codage LDPC pour fournir une correction d'erreur. L'émetteur peut être sans fil ou filaire. Le port E / S détecte le signal de l'émetteur et peut avoir besoin protocoles pour recevoir les données. Ce signal est ensuite fourni au décodeur LDPC. le Le décodeur LDPC détecte et tente de corriger les erreurs introduites dans le signal. Le processeur contrôle les opérations du port d'E / S. La mémoire peut être composée de n'importe quel type de éléments de stockage, tels que DRAM, SRAM ou mémoire flash.

4.4.2 Check Node Unit

L'unité de nœud de contrôle de la figure 4.5 émule les calculs effectués sur un nœud de contrôle . Il envoie le minimum des valeurs reçues et le signe de chaque valeurs.

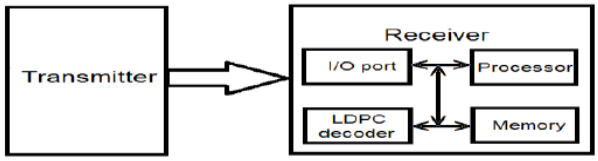

FIGURE 4.4 – Architecture de base de niveau supérieur[35].

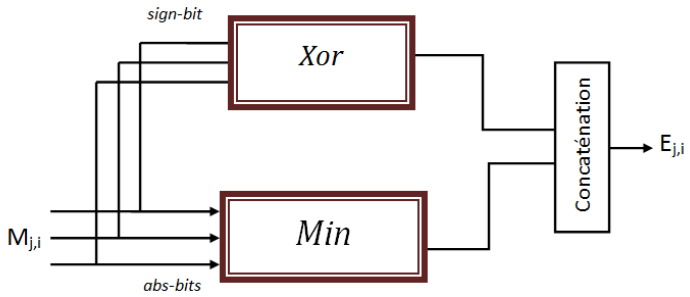

FIGURE 4.5 – Schéma général d'une unité du nœud de contrôle .

4.4.3 Variable Node Unit

L'unité de nœud variable sur la figure 4.6 émule les calculs qui sont effectués sur un nœud variable d'un graphique de Tanner. L'unité de noeud variable a un bloc conditionneur qui ajoute toute la valeur log-vraisemblance de rapport (LLR) qu'il reçoit et un soustracteur qui soustrait la valeur qu'il reçoit d'un nœud de contrôle à partir de la somme partielle et renvoyez-le au nœud de contrôle. Une décision difficile est également calculé à partir du signe de la somme.

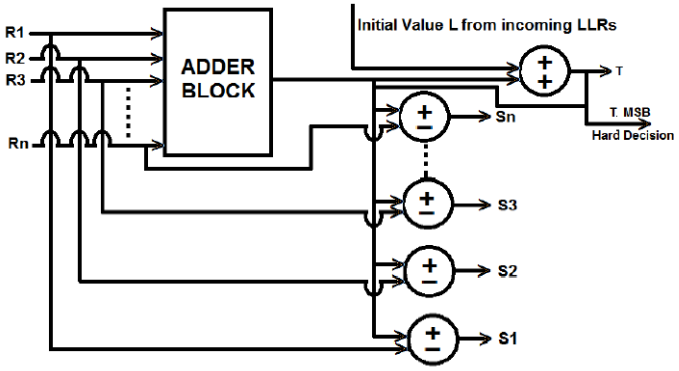

FIGURE 4.6 – Schéma général d'une unité du nœud de variable[42].

Chapitre 5

Implémentation sur FPGA d'une architecture basée sur l'algorithme Min-Sum

Dans ce dernier chapitre, nous présentons une implémentation sur FPGA d'une architecture d'un décodeur LDPC basé sur l'algorithme Min-Sum. Pour cela, nous avons choisit une architecture entièrement parallèle. Le code sélectionné est un code LDPC régulier (4, 6). Nous allons également donner et expliquer en détail, les blocs CNU, VNU et Syndrome Unit, ces Unités ont été implémentés sur le circuit BASYS-2-FPGA via la plate-forme Xilinx-ISE. Le langage de description utilisé est le langage VHDL. L'outil ModelSim est utilisé pour faire les simulations nécessaires des différents blocs avant l'implémentation. Par la suite nous présentons l'architecture des différents blocs obtenus après la synthèse. Finalement, nous terminons par une discussion du rapport sythèse généré par ISE-Xilinx.

5.1 Description de la carte FPGA-BASYS-2

Nous avons l'intention d'implémenter un décodeur LDPC basé sur l'algorithme Min-Sum sur la carte FPGA-BASYS-2, Pour cela il est important de connaître les caractéristiques de la carte FPGA, qui va être le support de notre architecture.

Caractéristiques :

• Famille : 100,000-gate Xilinx Spartan 3E-100 CP132 FPGA

• Configuration : Atmel AT90USB2 Full-speed USB2 port providing board power and programming/data transfer interface

• Mémoire : Xilinx Platform Flash ROM to store FPGA configurations

• Réseau et communication : PS/2 port and 8-bit VGA port, Four 6-pin header expansion connectors

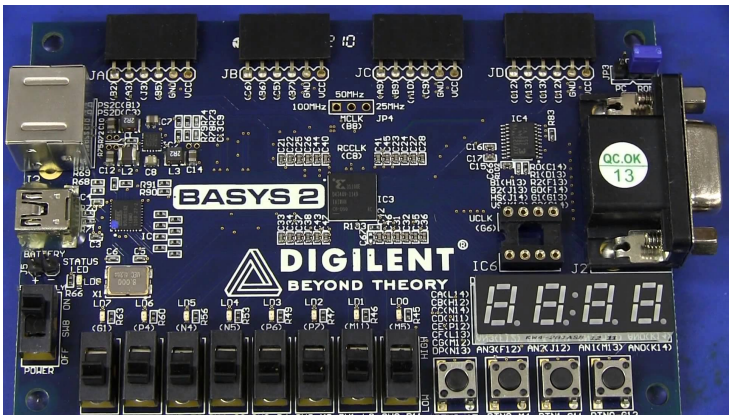

FIGURE 5.1 – La carte FPGA Basys2

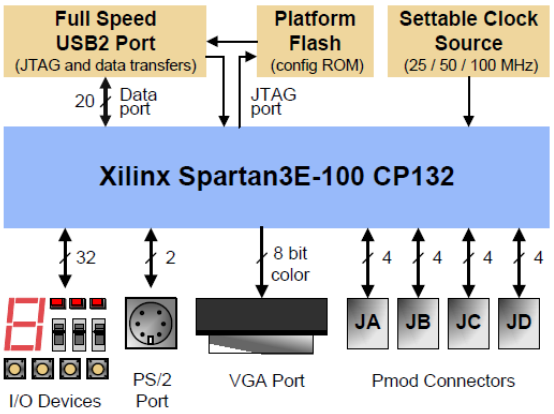

FIGURE 5.2 – Diagramme de bloc de la carte FPGA-Basys2

• Afficheur et I/O et contrôle : 8 LEDs, 4-digit 7-segment display, 4 buttons, 8 slide switches, ESD and short-circuit protection on all I/O signals.

• Horloges : User-settable clock (25/50/100MHz), plus socket for 2^{nd} clock

5.2 Architecture du décodeur implémenté sur l'FPGA

Dans cette première partie nous allons implémenter le code représenté par sa fameuse matrice de contrôle mais avec une taille réduite (4,6).

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{5.1}$$

Le type de conception utilisé pour son implémentation est le type entièrement parallèle. Le schéma synoptique général complet du décodeur est donné par la figure 5.4.

L'architecture contient 3 blocs de base :

- Check Node Unit.
- Variable Node Unit.
- Syndrome Unit.

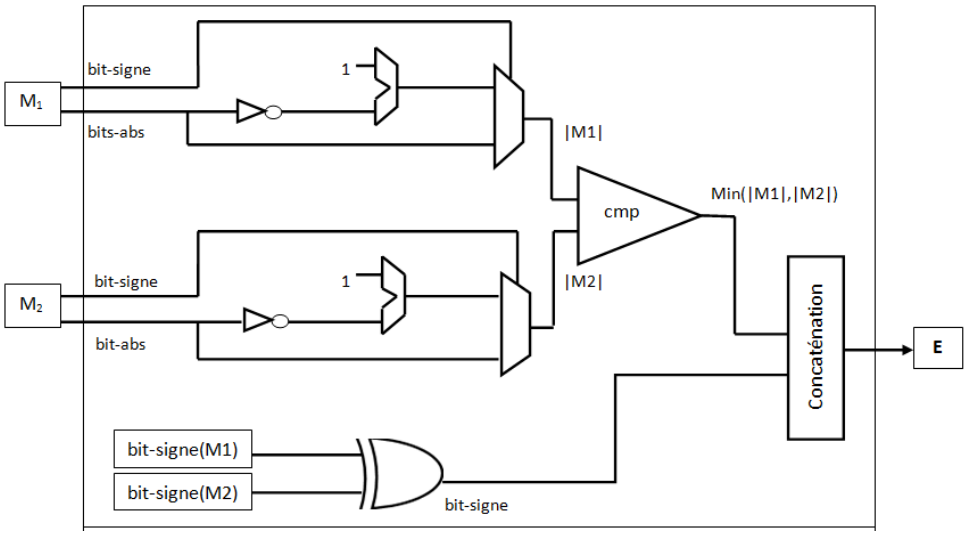

FIGURE 5.3 – Schéma de l'unité de contrôle élémentaire

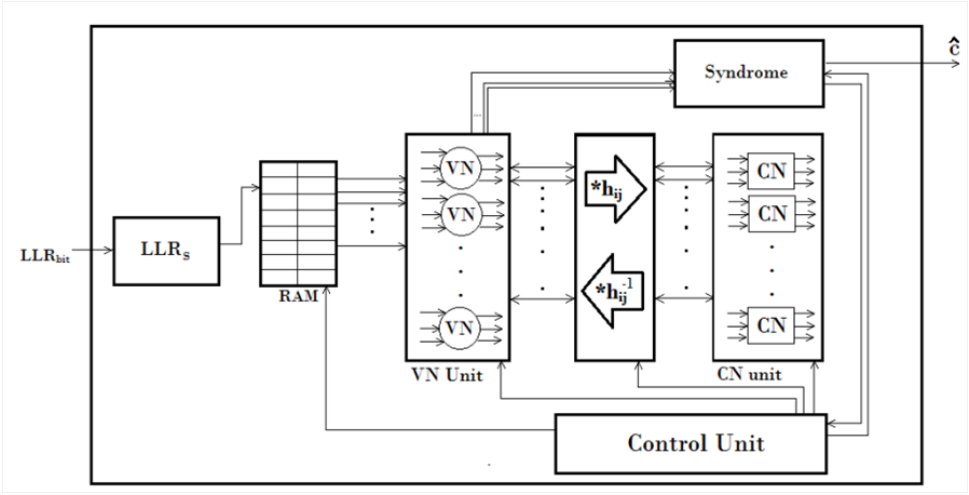

FIGURE 5.4 – Schéma du décodeur LDPC entièrement parallèle

5.3 Les blocs générés dans Xilinx ISE

5.3.1 Unité du nœud de contrôle (CNU)

Ce bloc est l'élément responsable à fournir les informations extrinsèques aux nœuds de données afin de mettre à jour les nouveaux LLRs. Il reçoit les messages en série lus du bloc de RAM et recherche les deux minimum, ces derniers doivent passer par deux états de transitions et finalement l'état final lorsque toute les données séries ont traités, une opération de XOR permet de déterminer le signe des valeurs de sortie, ces dernières sont

elles aussi acheminées en série.
Le schéma RTL donné par la figure 5.4 montre le bloc du CNU tel qu'il est généré dans Xilinx ISE. Il a comme signal d'entré la matrice M et en sortie la matrice E qui va être le signal d'entré du VNU par la suite.

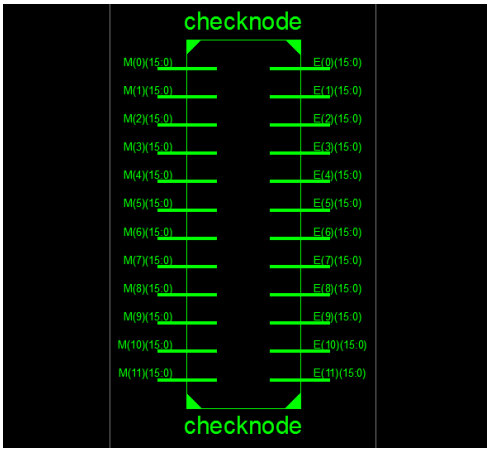

FIGURE 5.5 – Bloc de l'unité du nœud de contrôle généré par Xilinx-ISE.

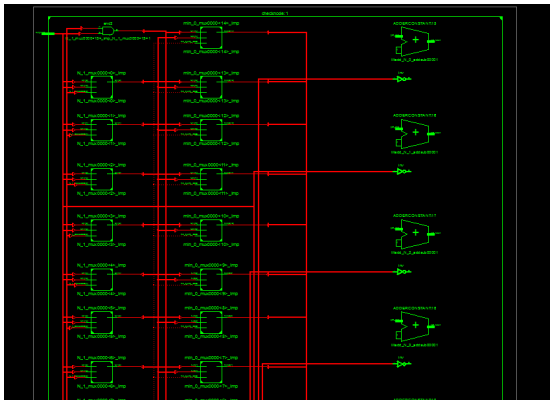

FIGURE 5.6 – Schématique du bloc noeud de contrôle généré par Xilinx-ISE.

5.3.2 Unité du nœud de variable (VNU)

Ce bloc est l'élément responsable à fournir les informations extrinsèques aux nœuds de contrôle afin de mettre à jour la matrice M qui va être à l'entré de l'unité de contrôle, ainsi le vecteur de séquence z qui va être l'entré de l'unité de syndrome pour voir si c'était un mot de code(sans erreurs) donc fin de calcule, sinon l'unité de variable va mettre à jour la matrice M pour une nouvelle itération. Le schéma RTL donné par la figure 5.6 montre le bloc du VNU tel qu'il est généré dans Xilinx ISE. Il a comme signal d'entré la matrice E, H et le vecteur de données r et en sortie la matrice M(si le mot n'est pas encore corrigé) et la séquence binaire z qui va être le signal d'entré du VNU par la suite.

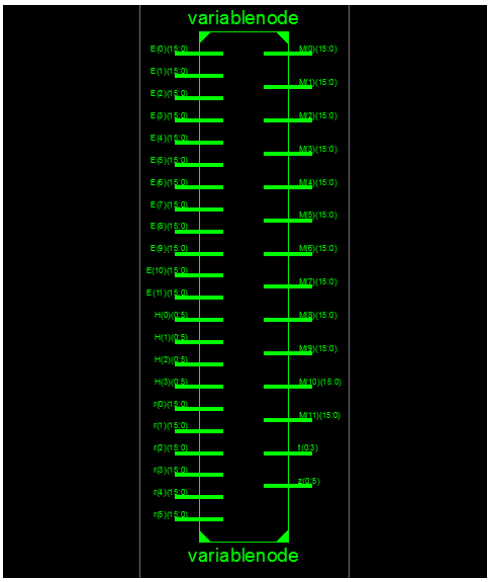

FIGURE 5.7 – Bloc de l'unité du nœud de variable généré par Xilinx-ISE.

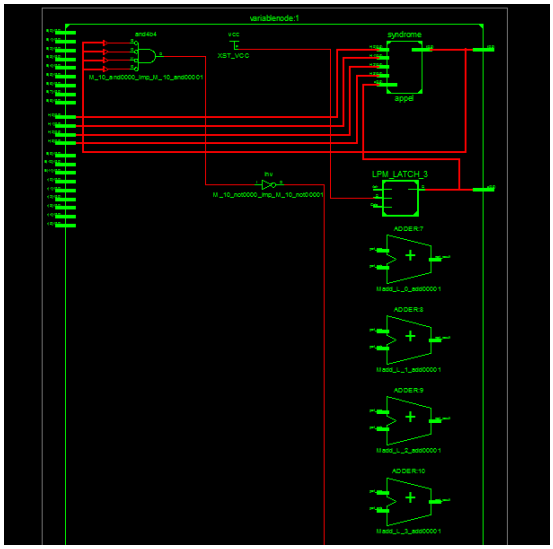

FIGURE 5.8 – Schématique du bloc noeud de variable généré par Xilinx-ISE

5.3.3 Syndrome Unit

Ce bloc est l'élément responsable qui calcule le produit matricielle Hz^t qui représente le résultat du syndrome [1]. Si le résultat est nul, le calcule s'arette et envoie en sortie le mot de code corrigé z sinon, on envoie la nouvelle matrice M au nœud de contrôle pour une autre itération.

5.3.4 Unité du décodeur

La synthèse du code vhdl qui décrit l'architecture parallèle du code (4,6) que nous avons proposé est présentée dans la figure 5.12. Nous pouvons directement remarquer

1. produit scalaire entre les lignes de H qui représentent les mots erreurs et le mot corrigé z. Le vecteur z sera un mot de code si le résultat du syndrome est le vecteur nul.

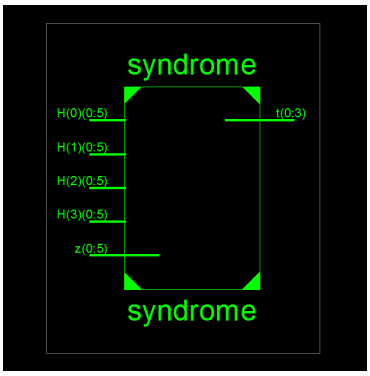

FIGURE 5.9 – Bloc de l'unité du Syndrome généré par Xilinx-ISE.

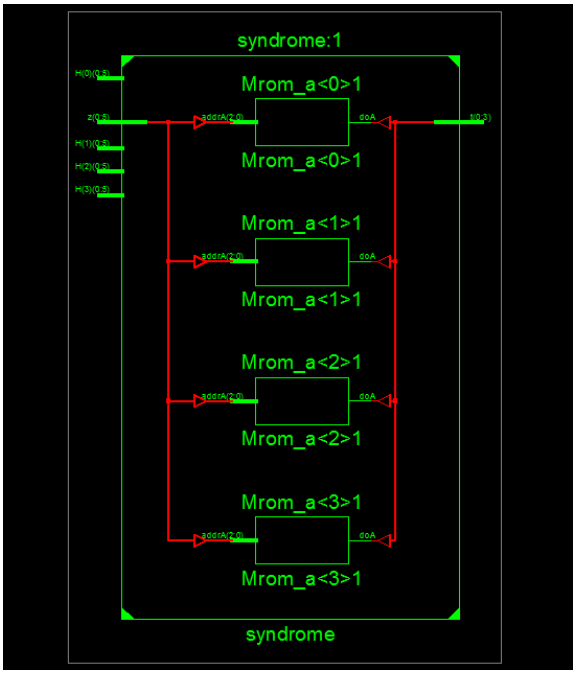

FIGURE 5.10 – Schématique du Syndrome généré par Xilinx-ISE.

à l'œil nu l'immense complexité de l'architecture, même si la taille du code est réduite. Cette conception est celle qui garantie le meilleur débit possible, où deux cycles d'horloges suffisent pour réaliser une itération. mais d 'un point de vue complexité, l'architecture doit établir un réseau de câblage complexe pour relier tout les blocs et assurer le chemin de donnée. Par exemple pour le code LDPC du standards dvbs2 et pour une conception entièrement parallèle, 13.000 liaisons entre chaque 2 blocs adjacents doivent être établies.

5.4 Simulation du décodeur

Dans la simulation, toutes les données seront représentées en complément à deux(CA2) et sur 16 Bits, 11 bits pour la partie fractionnaire(car on a besoin d'une précision de l'ordre de 10^{-4} dans certains cas), 4 bits pour la partie entière(on a pas besoin d'une grande valeur dans tous les cas), 1 bit de signe.

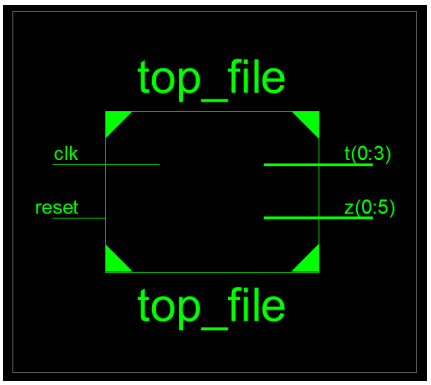

FIGURE 5.11 – Bloc général de l'unité du décodeur généré par Xilinx-ISE.

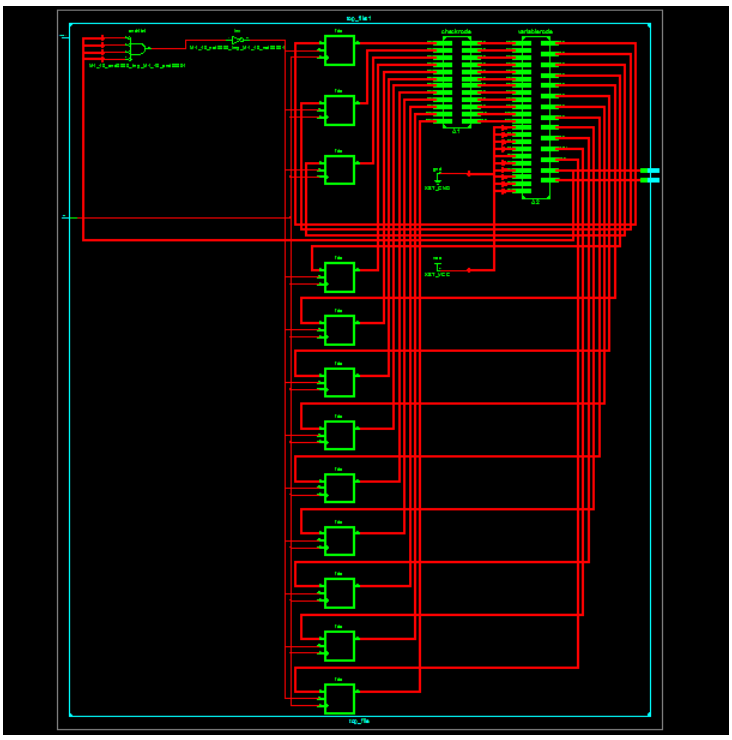

FIGURE 5.12 – Schéma de l'architecture du décodeur généré par Xilinx-ISE

5.4.1 Simulation fonctionnelle

Après avoir décrire les différentes unités en langage VHDL, nous avons simulé les blocs à l'aide du logiciel *ModelSim*.

En utilisant le code LDPC donné par sa matrice de contrôle 5.1 et on choisie comme mot de code le suivant :

$$c = [0\ 0\ 1\ 0\ 1\ 1]$$

Ce dernier est envoyé via un canal $AWGN$ avec une modulation $BPSK$ et $SNR = E_s/N_0 = 1,25$ (ou 0,9691 dB) et le signal reçu est

$$y = [-0.1\ 0.5\ \ -0.8\ 1.0\ \ -0.7\ 0.5]$$

Si la décision est prise sans décodage, on aura deux bits qui sont erronés dans le vecteur reçu, les bits 1 et 6.

Pour un canal AWGN les LLR a priori sont donnés par la formule suivante :

$$r_i = 4y_i \frac{E_s}{N_0} \tag{5.2}$$

Nous avons donc

$$r = [-0.5\ 2.5\ -4.0\ 5.0\ -3.5\ 2.5]$$

Les résultats de la simulation fonctionnelle sur *ModelSim* des 3 unités et le top-file(programme global du décodeur) sont représentés sur les figures 5.12, 5.13, 5.14, 5.15

En analysant les résultats obtenus, les informations extrinsèques ont été calculés avec précision et l'erreur a été détectée puis corrigée et le décodeur "Min-Sum" converge vers le mot de code correct après trois itérations.

5.4.2 Simulation temporelle

Après s'être assuré que le circuit conçu est fonctionnellement correct, nous devons maintenant effectuer la simulation de temporelle en utilisant le simulateur *Isim* défini sur Xilins-ISE pour voir comment il va se comporter quand il est réellement implémenté dans le périphérique FPGA choisi. donc la simulation temporelle a un objectif pour vérifier le comportement du circuit en tenant compte des temps de réponses et les retards des composants et des contraintes de temps.

Pour vérifier si le circuit conçu est temporellement correct, on doit imposer une horloge égale à celle générée par l'FPGA (50Mhz) et voir si les composants de l'FPGA ont assez de temps pour faire les opérations et les changements d'états nécessaires. Ce retard est dû aux retards de propagation dans l'élément logique et les fils dans le dispositif FPGA.

Test sur l'FPGA

Pour pouvoir regarder les résultats obtenus sur la carte FPGA, un "Diviseur de fréquence" a été injecté dans le programme général(Top-file) afin de visualiser les étapes de décodage dans les LEDs de la carte et voir également le résultat du "Syndrome" sur les 7-Segments de cette dernière.

La figure 5.18 présente le résultat final du décodage où la séquence binaire souhaité est affichée sur les LEDs.

Calcul du débit :

Ci-dessous les différentes opérations qui s'exécutant à des cycles d'horloges différents sont :

Chaque itération nécessite 1 seul cycle d'horloge pour s'effectuer. Dans le cas ou 4 itérations sont imposés au décodeur(suffisamment large pour ce code),le nombre de cycle d'horloges à consommer est égale à 4. le calcul théorique du débit est donné ci-dessous :

Supposons que la moyenne ça prend n nombre d'itérations pour chaque trame. chaque trame est codé sur 6 bits. la fréquence de l'horloge est de 50MHz. comme chaque itération nécessite 1 cycle d'horloge, nous obtenons 6 bits en sortie, en d'autres termes,6 bits chaque $20 \times n$ nano secondes, Cependant, pour une seconde nous obtenons $6/20 \times n$, i.e, $300Mbp/ns$. Par exemple si $n = 4$ itérations, le débit donc serait égale à $75Mbp/s$. Tout de mème le plus grand avantage de cette architecture est la flexibilité et la vitesse de transmission.
La figure 5.16 montre le rapport de synthèse de cette architecture qui affiche des informations pour faciliter l'analyse de l'utilisation des périphériques.

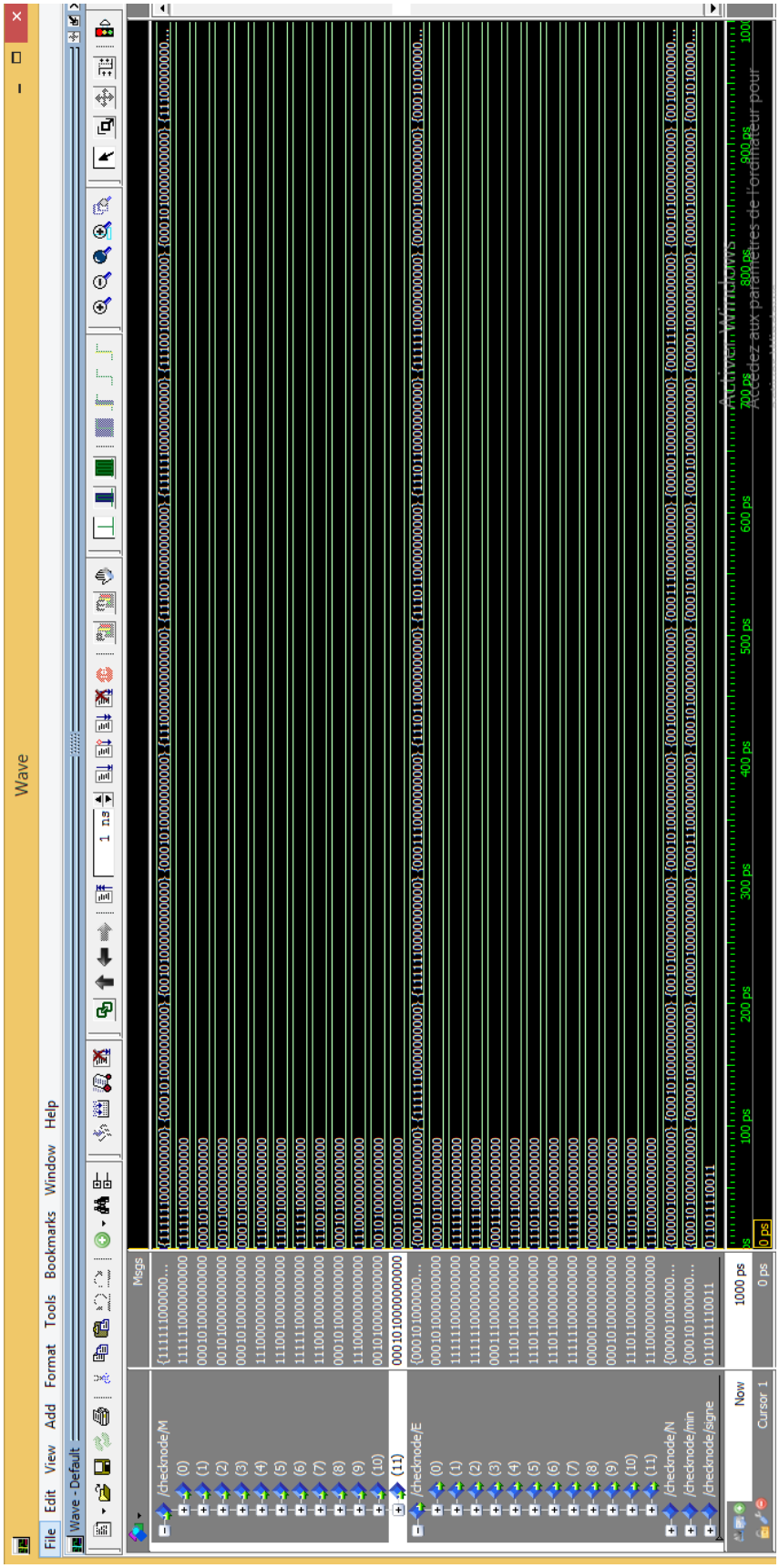

FIGURE 5.13 – Résultat de la simulation fonctionnelle du CNU sur ModelSim

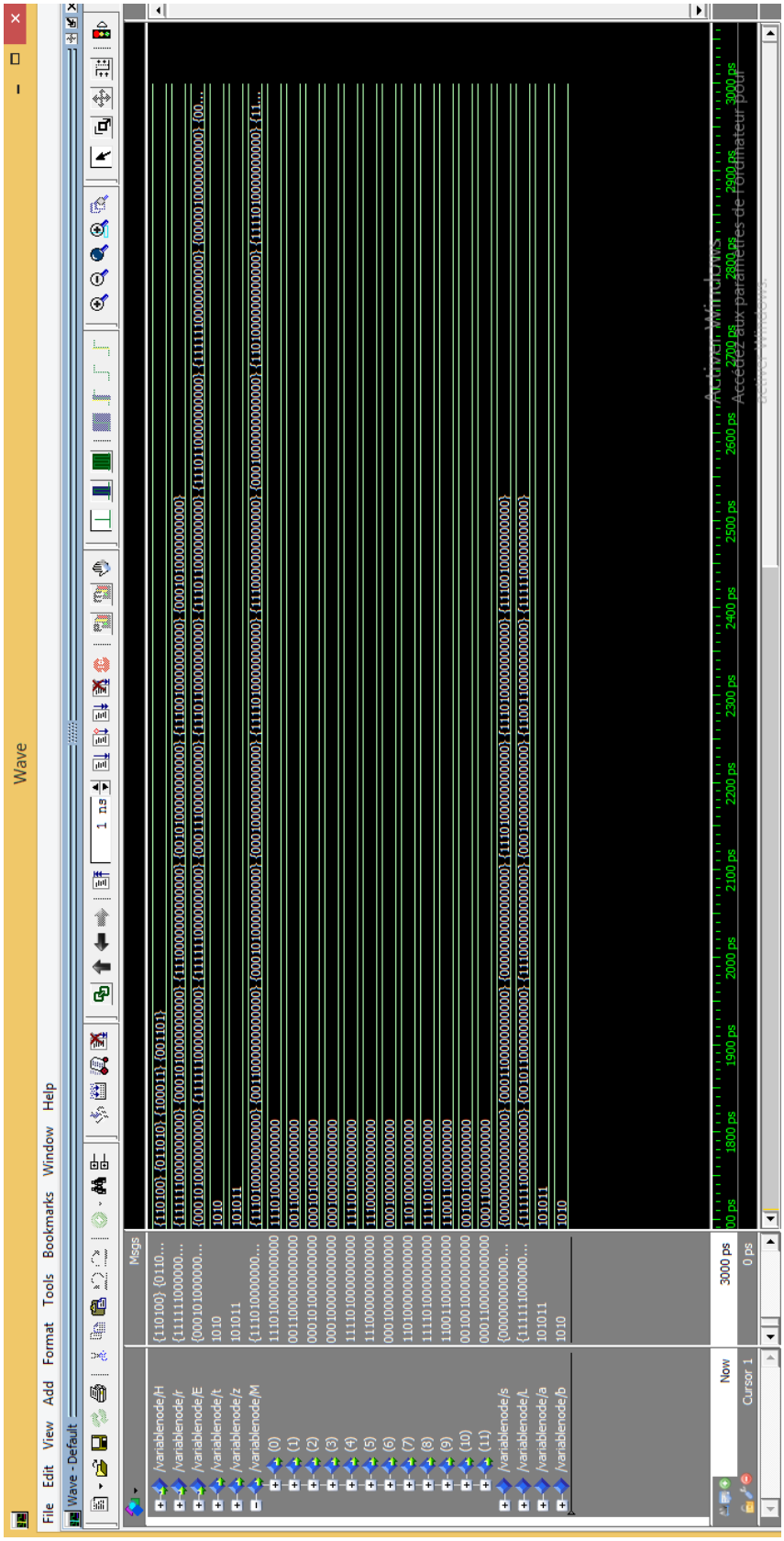

FIGURE 5.14 – Résultat de la simulation fonctionnelle du VNU sur ModelSim

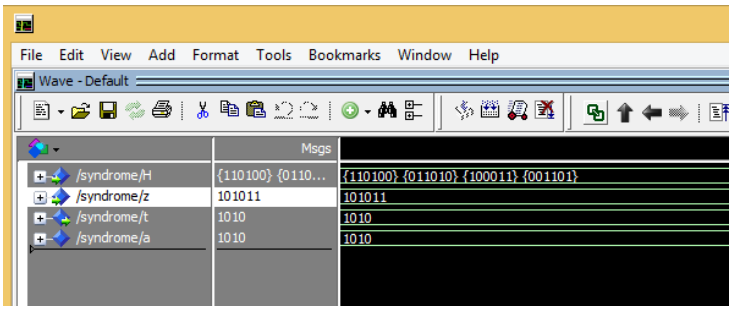

FIGURE 5.15 – Résultat de la simulation fonctionnelle de l'unité de syndrome sur ModelSim

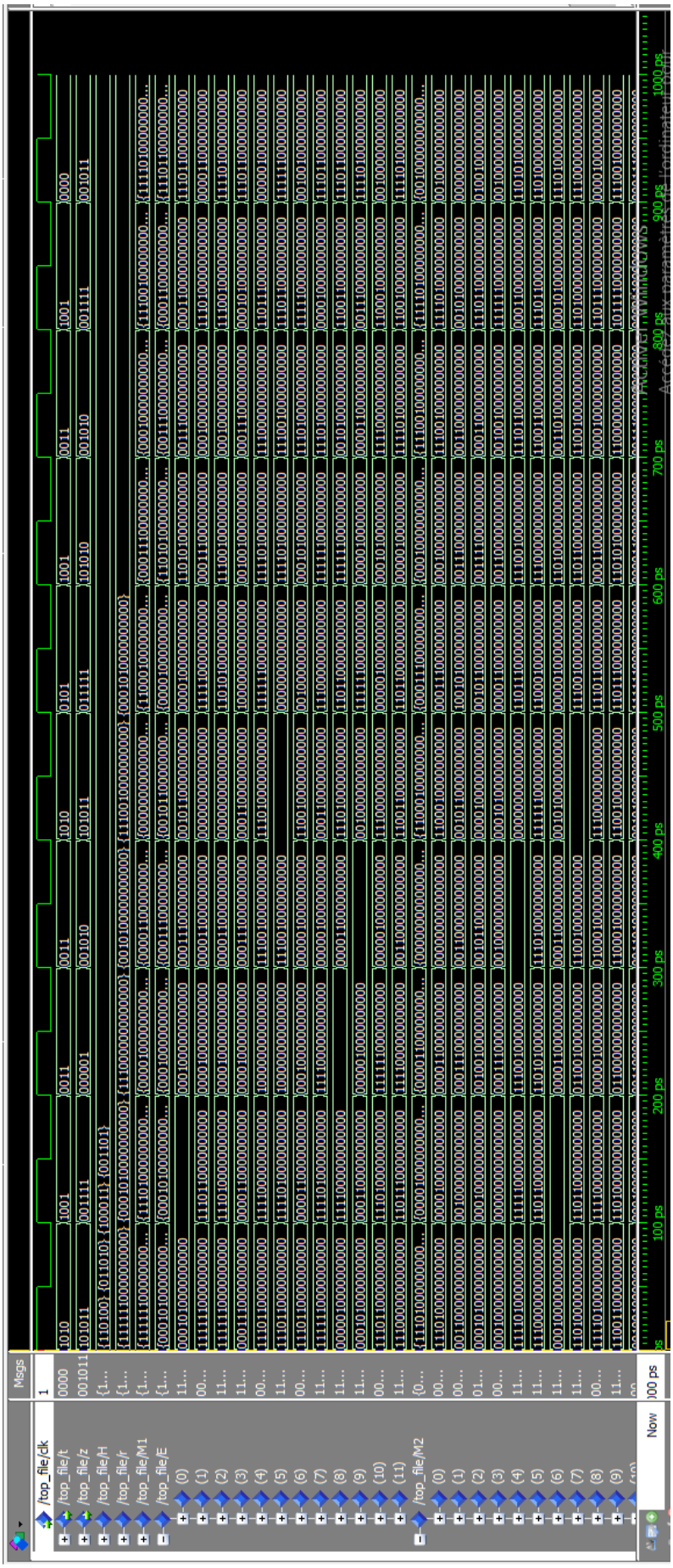

FIGURE 5.16 – Résultat de la simulation fonctionnelle du décodeur sur ModelSim

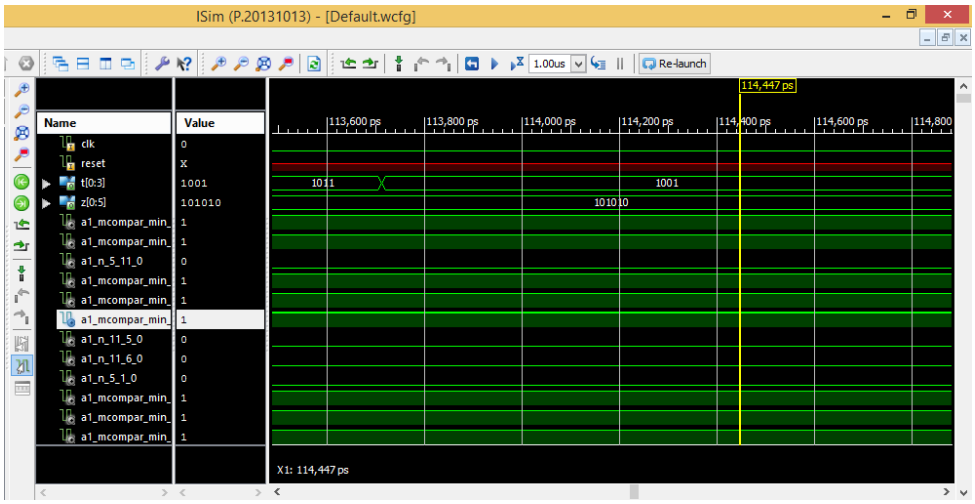

FIGURE 5.17 – Illustration de la simulation temporelle du décodeur sur Xilinx-ISE

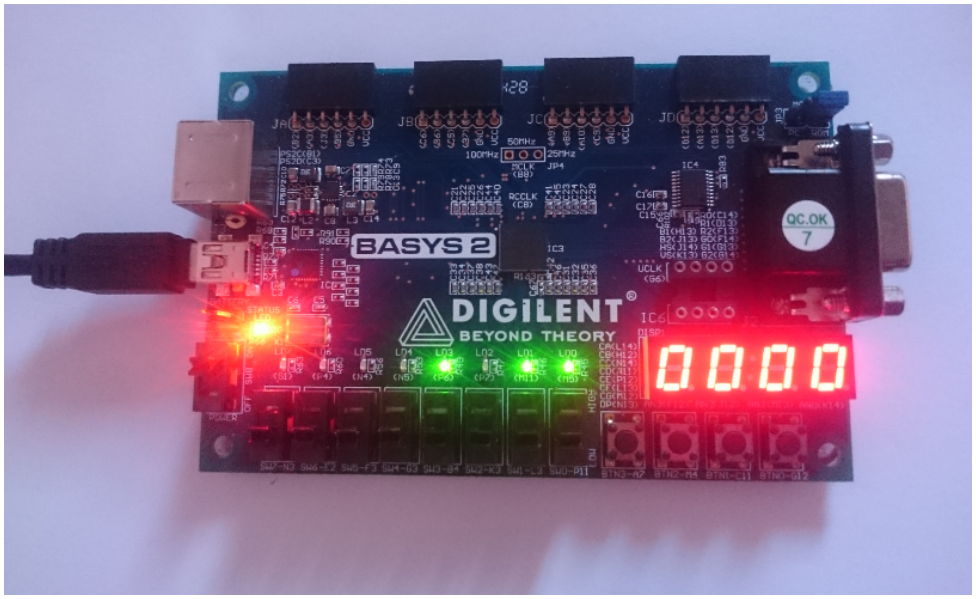

FIGURE 5.18 – Résultat du test obtenus sur la carte BASYS-2-FPGA

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	578	960	60%
Number of 4 input LUTs	1116	1920	58%
Number of bonded IOBs	384	83	462%

FIGURE 5.19 – Rapport de synthèse

Conclusion générale

Ce projet de fin d'étude a été consacré à l'implémentation sur FPGA d'un décodeur LDPC dans le cadre des communications sans fils. Pour ce faire nous avons commencé notre travail par donner une description détaillée de la chaîne de communication numérique, tout en mettant l'accent sur le canal de communication sans fil, qui constitue le support physique de l'information à traiter dans notre démarche.

En second lieu, nous avons présenté quelques résultats sur les performances des codes correcteurs d'erreurs les plus connus. En analysant ces résultats, nous avons constaté que seules les techniques de codage avancées tel que les codes LDPC peuvent atteindre la limite de *Shannon*, ce qui justifie par la suite notre choix pour ce type de codes correcteurs d'erreurs.

Pour la bonne compréhension de la partie pratique de notre travail, nous avons étudié de façon détaillée les différents algorithmes de décodages puis le choix supporté pour des raisons de simplicités sur l'algorithme Min-sum dont les performances sont très proches de la puissance de l'algorithme de décodage Sum-Product.

L'étude des performances établies dans le troisième chapitre montre que l'algorithme choisi min-sum est une solution correcte en terme de complexité, flexibilité et rapidité.

Concernant la partie pratique de notre travail, il s'agissait au départ d'implémenter une architecture d'un décodeur LDPC basée sur l'algorithme Min-sum. Pour cela, nous avons pris un code réduit de taille(4, 6). Les résultats d'implémentation ont montré de très bonnes performances en débit, mais des performances défavorables en flexibilité et en complexité de l'architecture. Les résultats et les détails d'implémentations sont documentés dans ce rapport.

A la fin de ce projet nous pouvons conclure, que les codes LDPC donnent des bonnes performances de correction d'erreur en se référant sur les résultats obtenus dans le dernier chapitre.

Ce travail non exhaustif offre quelques perspectives que nous présenterons ci-dessous :

•Considérer pour l'implémentation du décodeur LDPC Min-Sum une architecture basée sur les codes quasi-cyclique.

•Conception d'un décodeur LDPC utilisant le code irréguliers.

•Étudier la faisabilité de l'extension de l'étude d'un code LDPC non-binaire.

• Implémentation d'un décodeur pour les codes LDPC irréguliers en utilisant les réseaux de *Benes* (Benes Network).

Bibliographie

[1] T.T.S.I, *digital video broadcasting (DVB) second generation framing structure for broadband satellite applications*. url : http ://www.dvb.org.

[2] G.hn/G.9960. *next generation standard for wired home network*. url : http :// www.itu.int/ITU-T.

[3] IEEE P802.3AN, *10GBASE-T taskforce*. url : http ://www.ieee802.org/3/an.

[4] H. Nyquist. *"Certain factors affecting telegraph speed"*. In : *Bell System Technical Journal* 3 (1924),p.324–346.

[5] R. Hartley. *"Transmission of information"* .In : *Bell System Technical Journal* 3 (juil. 1928).

[6] C. E. Shannon. *"Mathematical theory of communication"*. In : *Bell System Technical Journal* 27 (oct.1948).

[7] K.L. Du et M.N.S. Swamy. *"Wireless Communication Systems"*. In : *Cambridge University Press, NewYork* (fév. 2009).

[8] A. Gersho et R.M. Gray. *Vector Quantization and Compression*. Kluwer Academic Publishers Norwell, MA, USA, 1991.

[9] M. A. Khalighi. *"Iterative decoding and detection (turbo methods), principles and related algorithms"*. In : *Report, INPG University, Gronoble, France* (sept. 2002).

[10] G. Battail. *"Théorie de l'information, Application aux techniques de communication"*. In : *Collection Pédagogique de télécommunication*. Masson (1997).

[11] J. G. Proakis. *Digital Communication*. 5 edition.McGraw Hill,2008.

[12] A. Galvieux. *Codage de canal des bases théoriques aux turbo codes*. Lavoisier,2005.

[13] D. Le Ruyet. *Theorie de l'information, Codage Source-Canal*. Ecole Supérieure de Conception et de Production industrielles, France.Jan2007.

[14] J. Oswald et G. Battail. *Théorie de l'information, Analyse diacritique des systèmes*. Edition Masson, 2006.

[15] S.Kaiser et K.Fazel. *Multi-carrier and spread spectrum systems : From OFDM and MCCDMA toLTE and WiMAX*. deuxième édition.Wiley, G Bretagne,2008.

[16] J. B. Doré. *"Optimisation de codes LDPC et leur architectures de décodage et mise enoeuvre sur FPGA"*. Thèsededoct. INSA de Rennes,Octobre2007.

[17] G. D. Forney. *"Concatenated Codes"*.Thèse de doct.Cambridge,1966.

[18] A. Spataru. *Fondements de la théorie de la transmission de l'information.* 1991.

[19] W. Peterson et E.J. Welden. *"Error-correcting codes".* In : MIT Press, Cambridge (1961).

[20] C. Berrou. *"Codes et Turbo-Codes".* In : Springer, 2007. Chap. 3 limites théoriques.

[21] R. G. Gallager, *"Low-Density Parity-Check Codes"*, Cambridge, MA : M.I.T. Press, (1963).

[22] J. Sun, An introduction to low density parity check codes, *Wireless Communication Research Laboratory Lane Department of Computer Science and Electrical Engineering, West Virginia University*, June 2003.

[23] B.Kurkoski. *"Introduction to low density parity check codes,"* in : (2007).

[24] M. Luby, M. Mitzemmacher, A. Shokrollahi, and D. Spielman, *"Improved Low Density Parity Check codes using irregular graphs and belief propagation"*, IEEE Transactions on Information Theory, Apr. 1998.

[25] M. Luby, M. Mitzemmacher, and A. Shokrollahi, *"Analysis of low density parity check codes and improved design using irregular graphs"*, STOC, 1998.

[26] I. S. Reed et G. Solomon. *"Polynomial codes over certain fnite felds".*In : *Journal of theSIAM* (juin 1960),p.300–304.

[27] ESTI Standard,Digital Video Broadcasting (DVB) :*Framing structure,channel coding and modulation for digital terrestrial television, Digital Video Broadcasting*, 2004.

[28] V. D. Goppa. *"A new class of linear error correcting codes".In : Problemy Peredachi Informatsi*,6 (sept.1970),p.207–212.

[29] B. Sakkour. *"Etude du décodage des codes Reed-Muller et application à la cryptographie".* Thèse de doct. UMA-ENSTA, 2007.

[30] M. Kanemasu. "Golaycodes".In : *MIT Under graduate Journal of Mathematics* (2000).

[31] R. Tanner. *"A recursive approach to low complexity codes".* In : IEEE Transactions on Information Theory 27 (sept.1981).

[32] B. Sakkour. *"Etude du décodage des codes Reed-Muller et application à la cryptographie".* Thèse de doct.UMA-ENSTA,1996.

[33] Kschischang F.R. et B.J. Frey. *"Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models".* In : Journal on Selected Areasin Communications 16 (1998),p.219–230.

[34] Sarah J. Johnson *Introducing Low-Density Parity-Check Codes* School of Electrical Engineering and Computer Science The University of Newcastle Australia.

[35] Arijit Mondal. *Design of a min-sum LDPC decoder for error correction.* INDIAN INSTITUTE OF SCIENCE BANGALORE, INDIA JUNE 2014

[36] Softwares for LDPC Codes, software, *http ://www.cs-toronto.edu/ radford/ldpc.*

[37] Blanksby.A, Howlan et C.J. "*A690-mW1-Gb/s,Rate-1/2 low density parity check code decoder".*In : IEEE J.Solide-StateCirc.37(3) 42 (2002),p.404–412.

[38] Cocco.M, Dielissen.J, Heijlligers.M, Heksta.A et Huisken.J. *"A scalable architecture for LDPC decoding"* .In : In :proceeding of 2004 Design, Automation and test in Europe 42 (mar.2004), p.429–455.