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 Electronique
Thème :

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

---

Oualid MOUHOUBI

Sous la direction de :
M. Mohamed TAGHI

Présenté et soutenu publiquement le 23/06/2016

Composition du Jury :

Président	M. Zidane TERRA	Dr	ENP
Rapporteur	M. Mohamed TAGHI	Pr	ENP
Examinateur	Mme. Aicha MOUSSAOUI	Dr	ENP

Promotion : Juin 2016

Dédicace

" Je dédie ce travail à toute ma famille. À mes chers parents pour tous leurs sacrifices consentis pour mon éducation et ma formation, sans eux je ne serais jamais arrivé à ce niveau. À mon frère et ma sœur pour leurs précieux soutiens et encouragements. À tous mes amis et camarades, à toutes les personnes qui m'ont enseigné tout au long de mon parcours. À tous ceux que je n'ai pas cités et que je n'oublierai jamais leurs soutiens et leurs aides. "

Remerciement

Je remercie M. M.TAGHI pour son soutien continu, sa disponibilité, son aide et ses précieux conseils.

je tien à remercier chaleureusement monsieur M. Z.TERRA pour m'avoir fait l'honneur de présider le jury de ce mémoire, ainsi que Mme. A.MOUSSAOUI pour voir accepté d'examiner mon travail.

je remercie toute personne qui, d'une manière ou d'une autre a contribué à l'élaboration de ce travail. Enfin, je tien à remercier tous mes amis et camarades ainsi que tous les enseignants du département d'Électronique de l'ENP.

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

Low-density parity-check (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	BinaryErasure Channel
BER	Bit Error Rate
BLAST	Bell LayeredSpace Time
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	BinarySymmetric Channel
CCSDS	The Consultative Committer for Space Data Systems
CNRS	Centre National de Recherche Scientifique
DVB-S2	2ndGéeneration Digital Video Broadcast Spatial
DVB-T	Digital VideoBroadcastTerrestrial
ENSPM	Ecole Nationale Supèrieure de Physique de Marseille
FER	Frame Error Rate
FSO	Free SpaceOptie
GSM	Goupe Signaux Multidimentionnels
GUI	Graphic User Interface
i.i.d	indépendant identiquement didtribué
IEEE	Institute of Electrical and ElectronicsEngineers
LD	Linear Dispersion
LDPC	LowDensityParity Check
LLR	Log Likelihhod Ratio
MAP	Maximum A Posteriori
MIT	Massachusetts InstituteofTechnology
MMSE	Minimum Mean Square Eroor
MP	Message Passing
MUX	Schéma de Multiplixage Spatial

Liste des notations

sign(.) La fonction signe

abs(.) La valeur absolue

$(.)^{t}$ La transposé d'un vecteur ou d'une matrice

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

$(.)^{*}$ Le conjugué

L(.) Le log-rapport de vraisemblance

II La fonction d'entrelacement

$\mathbf{II}^{-1}$ La fonction de désentrelacement

C Le veteur de la séquence d'information

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

C La capacité d'un canal

d_x Le débit symbole de la source

x Le vecteur du mot de code

P Le vecteur de Parité

K La taille de la séquence d'information

N La taille du mot de code

M Le nombre de bits de redondance ou parité

R Rendement de codage de canal

v Longueur de contrainte d'un code convulitif

m Mémoire d'un code convulitif

F_q Corps de galois et de q éléments

d_H La distance de Hamming

w_H Le poid de Hamming

d_{min} La distance minimale

X La matrice du code espace-temps

N La matrice de bruit

Y La matrice de récepton

R_{STC} Rendemenr de codage espace-temps

Introduction générale

Le monde vit ces 2 dernières décennies l'ère du tout numérique où la technologie des communications sans fils se taille la part du lion. Cet extraordinaire essor que connaît cette technologie est principalement dû aux avancées considérables dans domaine de la micro(nano) électronique ainsi qu'à la demande de plus en pressante de moyens de communications souples, plus rapides , et plus efficaces.

Les Techniques développées récemment dans ce secteur ont permis d'atteindre des débits de transmission très élevés de l'ordre du Gbps Les dispositifs et circuits de corrections d'erreurs qui sont une brique de base dans les systèmes de communications ont été l'objet de recherches intenses dans le but d'augmenter la fiabilité de transmission et de permettre des débits très élevés.

La découverte dans les années 90 des Turbo-codes et, plus généralement du principe itératif appliqué au traitement du signal, a révolutionné la manière d'appréhender un système de communications numériques. Cette avancée notable a permis la redécouverte des codes correcteurs d'erreurs inventés par R. Gallager en 1963, appelés codes Low Density Parity Check (LDPC).
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 WLAN 802.16e [1], le G.hn/G.9960 pour les réseaux domestiques filaire [2], et le 10GBASE-T standard pour les 10 Gigabit 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, d'un un code prédéfini choisi à partir du standards de communications Wi-max, alliant performance et flexibilité.

Organisation du document

Ce mémoire est organisé comme suit :

Le **chapitre 1**, 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 introduit ensuite détaillée, en indiquant le rôle de chaque élément la constituant, 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 **chapitre 2**, une large présentation des codes LDPC est proposée, incluant les notations et outils mathématiques indispensable à la compréhension, particulièrement au sujet d'encodage situé dans la dernière section ; Ce chapitre traite de la classe des codes LDPC, de la construction des codes LDPC et du concept d'ordonnancement qui sera d'une grande utilité pour le chapitre **chapitre 5** 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 enfin le concept des codes cycliques et leur avantage à augmenter la flexibilité du décodeur.

Le **chapitre 3** 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 de performance des différents algorithmes de décodage conçu pour différents types de codes LDPC.

Quand au **chapitre 4**, il fera l'objet d'une description architecturale de différentes implémentations des codes LDPC en déduisant les avantages et inconvénient de chacune d'elles, et enfin une comparaison des différents types de conceptions

Enfin dans le **chapitre 5**, est entamé par une présentation de la carte FPGA ML501 sur laquelle l'implémentation de notre travaille à été effectuée, nous proposons deux types d'architecture basée sur l'algorithme Min-sum sur lesquelles nous avons travaillé tout au long de notre projet, pour chacune d'entre elles, un détail de conception, une présentation des différents blocs ainsi généré par la plate-forme Xilinx ISE Virtex 5 XC5VLX110, et enfin une simulation accompagnée d'une discussion des résultats obtenus est proposée.

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

Chapitre 1

Généralités sur le codage correcteur d'erreur

Ce premier chapitre a pour objectif de présenter des principes et des concepts fondamentaux qui seront utiles pour la compréhension et l'implémentation des codes LDPC. Ce chapitre commence par illustrer la chaîne de communication et expliquer brièvement le rôle et l'intérêt de chaque bloc de la chaîne, avant d'introduire les différents types des codes correcteurs d'erreurs, et finir par une comparaison de leur performance.

1.1 Introduction

L'objectif fondamental d'un système de communication, est de reproduire en un point decla chaîne de communication un message transmis à partir d'un autre point, mais le problèmecqui se pose, est que le canal est généralement soumis à des perturbations, 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 chaine de communication numérique

On désigne par le paradigme de Shannon (voir Figurec1.1) 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].

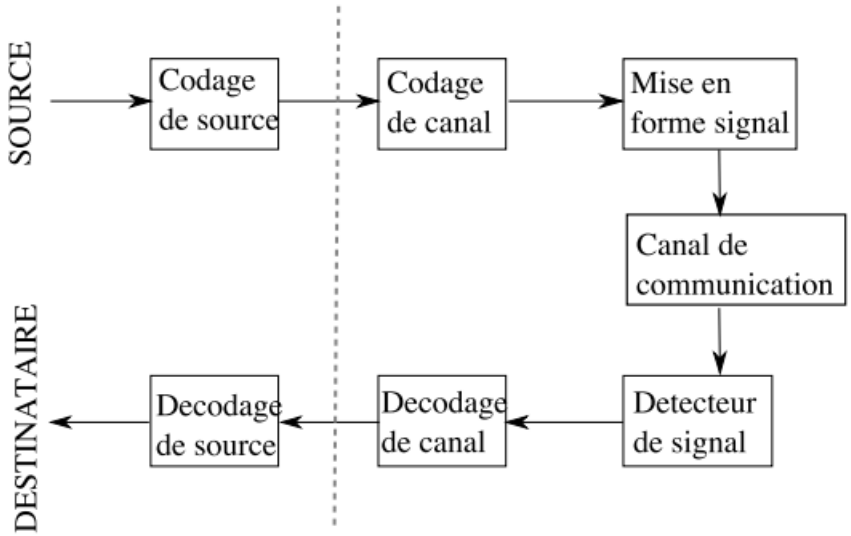

FIGURE 1.1 – schéma fonctionnel d'une communication numérique

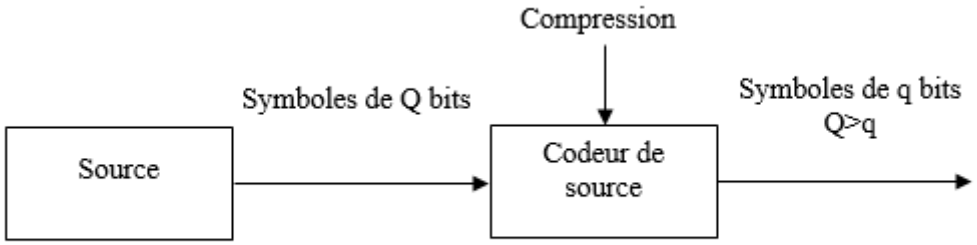

FIGURE 1.2 – Schéma simplifié d'un codeur de source

1.2.1 codage 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 de bits 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 surtout 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].

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'oeil (persistance rétinienne) et de l'oreille humaine (l'effet de masquage).

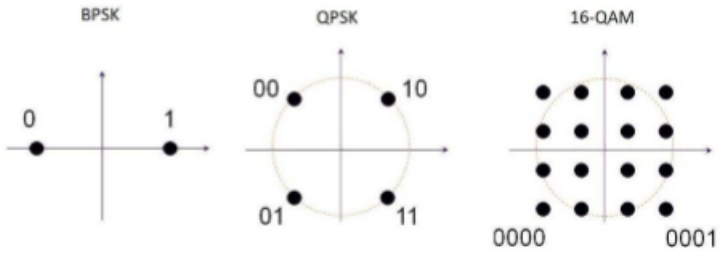

FIGURE 1.3 – Exemple de modulation numérique

1.2.2 Codage canal

La finalité du codage de canal est de protéger le message contre les perturbations du canal, et cela, en introduisant une redondance à l'information utile dans le message à transmettre. La redondance et l'information utile sont liées par une loi donnée. 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 $M = 2^L$ 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].

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. 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 défnies sur un alphabet X, et émettant en sortie des suites de symboles défnies sur un alphabet de sortie Y, reliés par une loi de transition $P_{Y|X}$ i.e. une matrice stochastique $M_{Y|X}$.

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

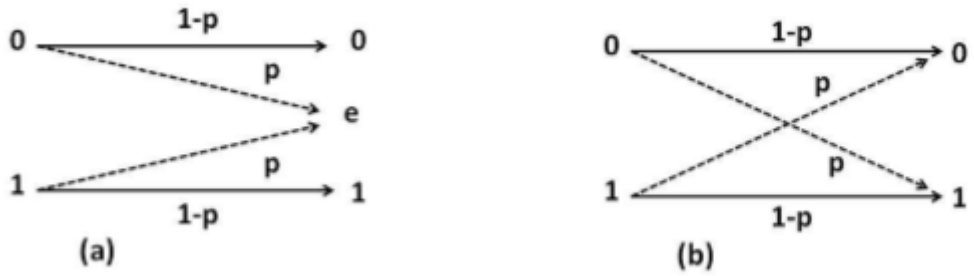

FIGURE 1.4 – (a) Le canal binaire avec effacement BEC. (b) Le canal binaire symétrique BSC

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

Le canal sera donc défini par :

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

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 Figure 1.4, 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 gauche 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 e). Ce canal est caractérisé par une probabilité d'erreur p et une matrice de transition $M_{Y|X}$ donnée par la relation.

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

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

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

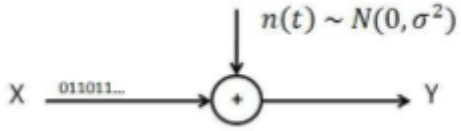

FIGURE 1.5 – Canal à entrée binaire perturbée par l'addition d'un bruit blanc gaussien

Additif et Gaussien ou AWGN (Additive White Gaussian Noise en anglais). Ce canal de transmission est rencontré dans les transmissions par faisceaux hertziens à faible débit ou dans les liaisons entre des satellites ou des sondes spatiales et des stations terriennes, c'est l'ensemble des transmissions radio électriques en espace libre.
Le bruit introduit dans ce canal est modélisé par un signal aléatoire $n(t)$, dont la distribution de probabilité suit la loi Gaussienne :

$$f_N(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(n-\mu)^2}{2\sigma^2} \tag{1.5}$$

ou :

$$\mu = En(t) = 0 \tag{1.6}$$

et

$$\sigma^2 = E[n(t) - \mu]^2 = En(t)^2 \tag{1.7}$$

représentent respectivement la moyenne et la variance.

1.2.5 Le canal radio mobile

La connaissance et la compréhension des caractéristiques du support de communication est indispensable pour aborder des travaux dans ce domaine. Les canaux radio mobiles sont considérés en particulier comme étant des canaux souffrant de nombreuses imperfections comme le multi-trajet, l'effet Doppler, l'atténuation par parcours (Path Loss) et l'effet de masque (Shadowing). Ces facteurs perturbateurs peuvent affecter les informations transmises. Le signal reçu sera donc la somme de répliques atténuées, réfléchies, réfractées et diffractées du signal transmis. Donc l'optimisation de notre système de communication en prenant en compte ces imperfections devient vraiment primordiale [15].

A Bruit dans un canal radio mobile

Le terme bruit est généralement utilisé pour dénommer des signaux non désirés qui perturbent ou corrompent la transmission et le traitement des signaux dans les systèmes de communications, et que l'on ne peut maitriser ou contrôler comme on le souhaiterait.

On distingue trois types de bruit dans les systèmes de communications :

- Le bruit de fond qui existe même en l'absence de tout signal porteur d'information, il peut être causé par les phénomènes d'agitation thermique, par les perturbations atmosphériques, bruit propre aux composants actifs...etc.
- Le bruit du a l'auto-perte, il provoque une distorsion dans la voie de transmission, il est causé par le phénomène de non-linéarité, l'échantillonnage avec filtre imparfait...etc.
- La diaphonie(crosstalk) qui représente l'influence indésirable entre signaux utiles transmis sur des voies voisines dans l'espace, en fréquence ou dans le temps .

Le bruit diminue par conséquent le rapport signal sur bruit (SNR) à la réception et de ce fait, limitera l'efficacité spectrale du système. Les bruits présents dans les systèmes de communication peuvent être modélisés avec précision en utilisant un bruit blanc additif gaussien (AWGN).

B Impérfection dans le canal radio mobile

Dans le canal radio mobile, le signal transmis souffre de plusieurs effets, dont les plus importants sont les suivants :

— Multi-trajets.

— Effet Doppler.

— Atténuation par parcours (Path Loss).

— Effet de masque (Shadowing).

Évanouissement à petite-échelle (small-scale fading) :

Le phénomène d'évanouissement à petite échelle est un phénomène très local, c'est-à-dire qu'il se produit quand le mobile se déplace d'une faible distance (fast fading).

- **Propagation multi trajets :**
  La Propagation multi-trajets apparait comme conséquence de réflexion, dispersions 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].

- **L'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 :

$$f_d = \frac{v f_c \cos(\alpha)}{c} \tag{1.8}$$

Où

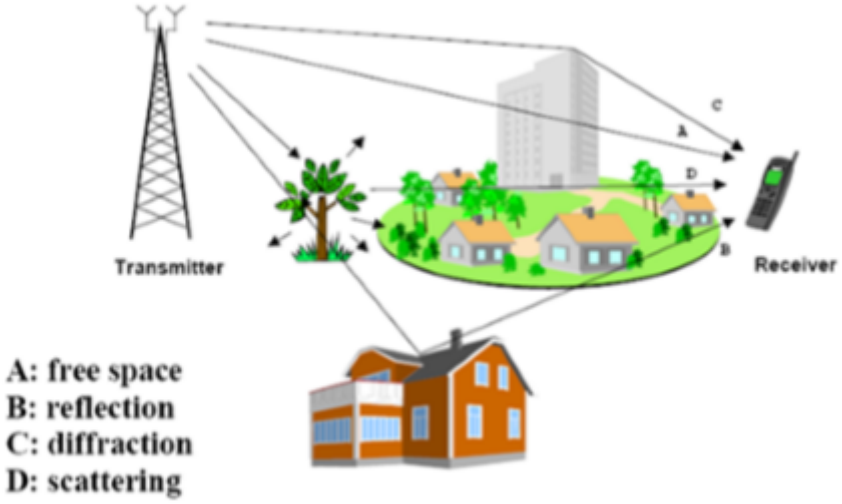

FIGURE 1.6 – Propagation par trajets multiples [15]

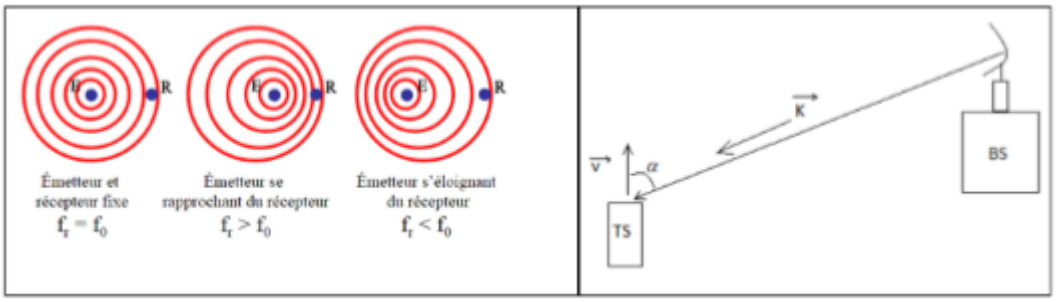

FIGURE 1.7 – Principe de l'effet Doppler [15]

v : 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).

On remarque d'après l'équation 1.8 et la figure 1.7 que la fréquence Doppler est proportionnelle à la vitesse du déplacement. De plus, si 1'on se déplace dans la direction de 1' arrivée de l'onde, le décalage Doppler est positif, donc la fréquence du signal reçu augmente, ce qui donne un étalement du spectre du signal reçu. Quand on s'éloigne de la direction de 1' arrivée de 1' onde, la fréquence Doppler est négative, donc la fréquence du signal reçu diminue.

L'effet Doppler provoque des changements de phases et d'amplitudes des signaux se propageant dans le canal, ce qui rend la propagation multi-trajets variable dans le temps. Même des mouvements de l'ordre de la longueur d'onde peuvent affecter grandement la superposition des signaux à la réception. La variation de la force du signal, due à la propagation multi-trajets variable dans le temps, est appelée évanouissement rapide (fast fading) [15].

Évanouissement à grande échelle (large-scale fading) :

Il y a deux types d'évanouissement à grande échelle :

— Pertes par parcours (pathloss).

— Effet de masquage (shadowing).

- **Pertes par parcours (pathloss) :** Les pertes par parcours représente 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). Cependant 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].

- **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 (slow fading) 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. la figure 1.8 résume tous les types d'évanouissement :

1.2.6 La capacité d'un canal

On définit la capacité d'un canal 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.

$$C = \max_{P(X)} I(X;Y) \tag{1.9}$$

L'information mutuelle I(X ;Y) est définie par la relation suivante :

$$a + b \tag{1.10}$$

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.

$$H(x) = -\sum_{i}^{N} P_i \log_2 P_i \tag{1.11}$$

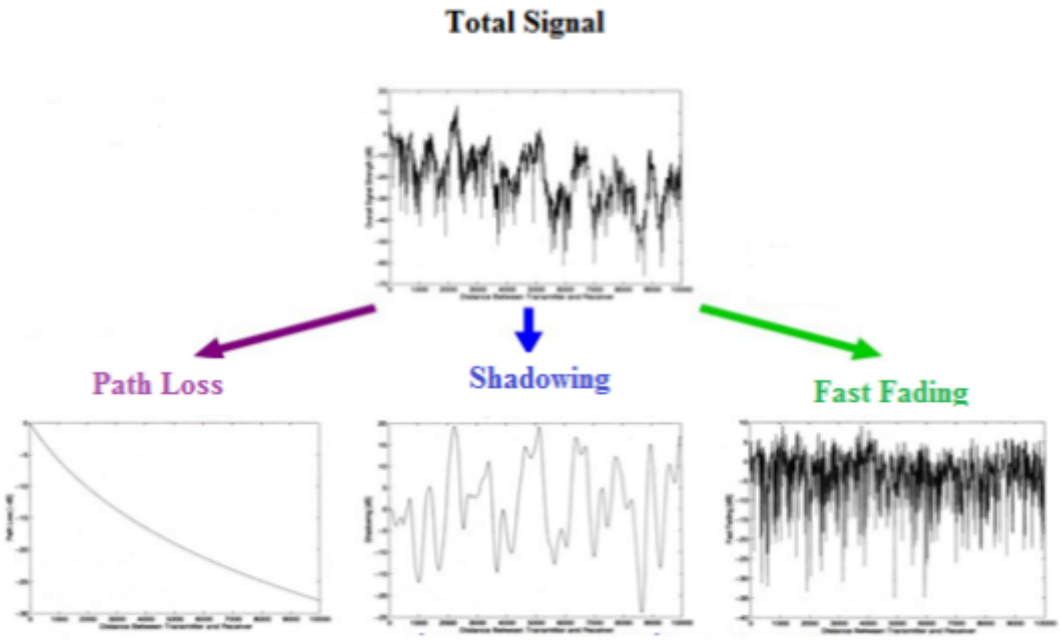

FIGURE 1.8 – schéma récapitulatif des différents types d'évanouissement [15]

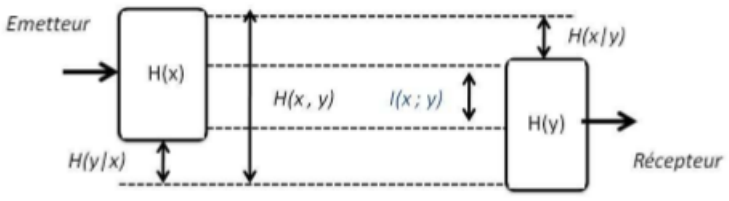

FIGURE 1.9 – schéma représentatif de l'information mutuelle

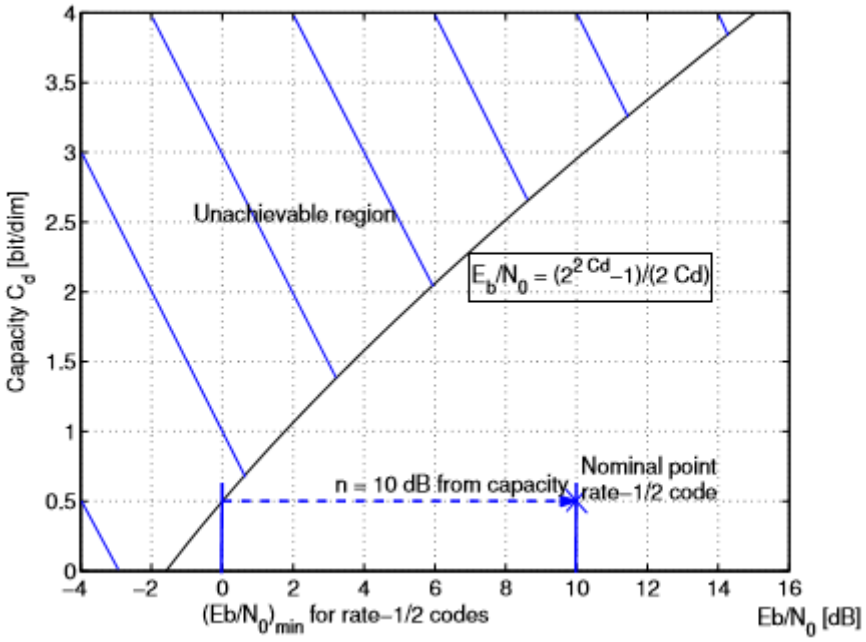

FIGURE 1.10 – variation de la capacité du canal AWGN en fonction du SNR

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 :

$$C_s = C.d_s \tag{1.12}$$

où d_s, représente la probabilité d'apparition des lettres de l'alphabet de la source.

Si on considère le cas d'un canal à entrée binaire perturbé par l'addition d'un bruit balnc et gaussien (AWGN) de densité spectrale N_0, et on suppose que les signaux occupent une bande passante B et que leur puissance reçue est P. La capacité de ce canal en Shannon par symbole est donnée par la relation suivante :

$$C_{AWGN} = 1/2log_2(1 + P/N) \tag{1.13}$$

où $N = N_0B$ est la puissance du bruit [4]. L'allure de la capacité C en fonction du rapport signal sur bruit SNR est illustré dans la Figure 1.10.

1.2.7 Le théorème fondamental du codage canal

En 1948, Shannon a annoncé dans son article intitulé " A Mathematical Theory of Communication [6].", le théorème fondamental de la théorie de l'information : *Pour une*

4. La démonstration complète de la relation 1.13 est disponible dans la référence [11].

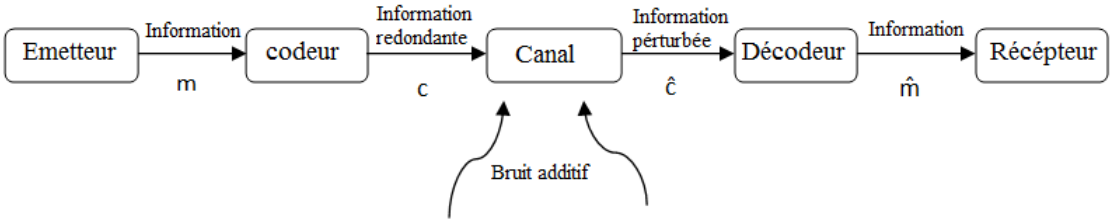

FIGURE 1.11 – Schéma simplifié d'un codeurdecodeur de canal

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

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)
- 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 :

$$R = K/N \tag{1.14}$$

- 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 $\oplus$ b	a $\odot$ 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 corp de Galois F2

- 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 $d_H(xi, xj)$. 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 dmin la plus petite distance de Hamming entre deux mots de code x_i et x_j.

$$d_{min} = \min_{\{x_i, x_j \in Z\}} d_H(x_i, x_j) \tag{1.15}$$

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 :

$$d_{min} = e + 1 \tag{1.16}$$

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

$$d_{min} = 2e + 1 \tag{1.17}$$

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 Figure 1.12. 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.

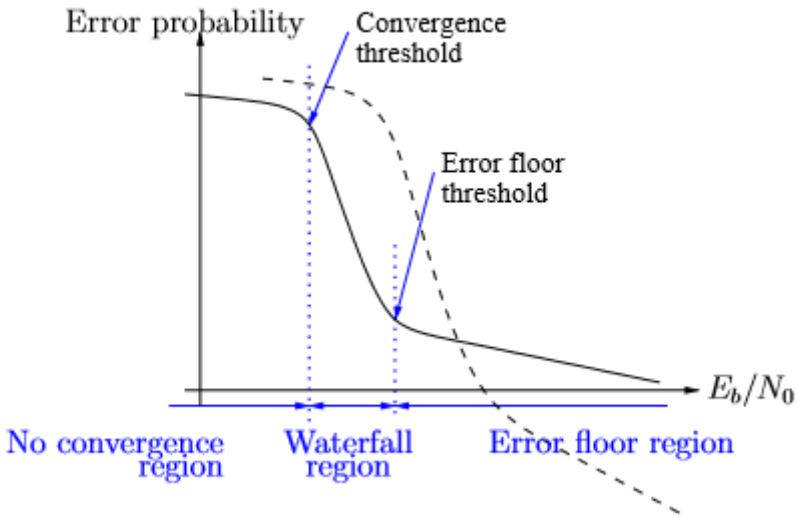

FIGURE 1.12 – Illustration des régions caractérisant les performances d'un code correcteur d'erreurs.

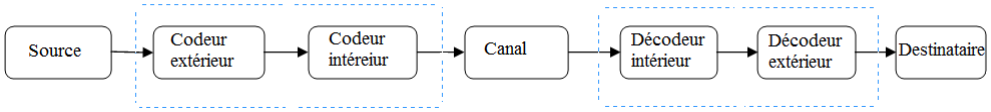

FIGURE 1.13 – Concaténation de deux codes correcteurs d'erreurs.

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, mais pour avoir des résultats plus exacts, surtout quand on compare des codes de rendements différents, on utilise un autre critère qui donne le BER en fonction de $Eb/N0$ [13]. avec :
Eb : L'énergie transmise dans un bit d'information.
$N0$: 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 décodable.

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.

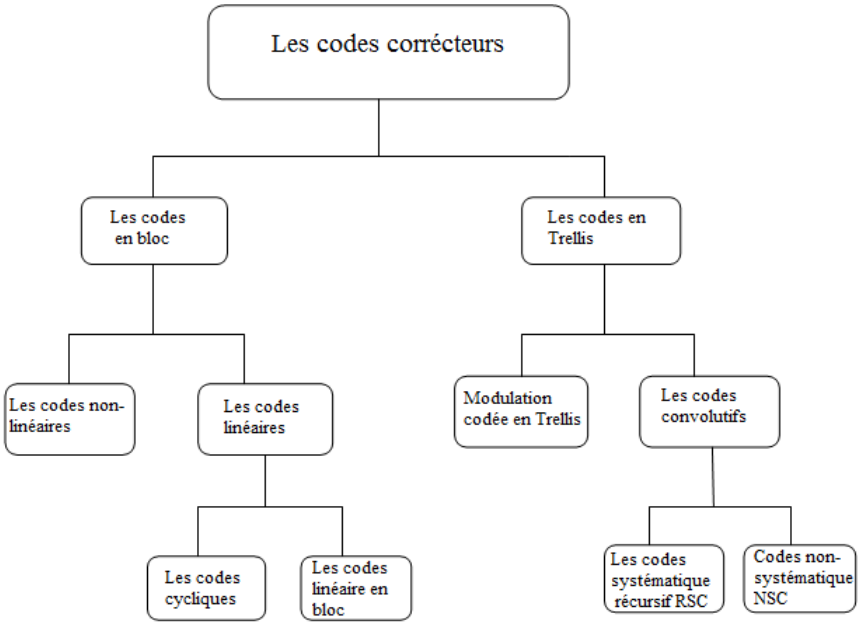

FIGURE 1.14 – Classification codes corrécteurs d'erreurs.

1.3.4 les classes de code correcteurs d'erreurs

Un code est dit linéaire si la fonction de codage est une application linéaire, sinon 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.

1.4 Les types des codes correcteurs d'rreurs

1.4.1 les codes en bloc

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

$$g : F_K^q \rightarrow F_N^q \tag{1.18}$$
$$c \rightarrow x = g(c) \tag{1.19}$$

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

Les codes linéaires en bloc : sont ceux où les mots de code sont considérés comme étant des éléments dans un espace vectoriel

Les codes cycliques : sont ceux où les mots de code sont considérés comme étant des éléments dans une algèbre, à savoir des polynômes [18].

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

A Les codes linéaire en bloc

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 :

$$x = c.G \tag{1.20}$$

avec :

$$G = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \ddots & \vdots & \\ g_{k1} & g_{k2} & \cdots & g_{kN} \end{bmatrix} \tag{1.21}$$

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 1.20 [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 [5].

$$G_{syst} = [R^t | I_K] \tag{1.22}$$

$$I_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{1.23}$$

et

$$R^t = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \ddots & \vdots & \\ r_{k1} & r_{k2} & \cdots & r_{kM} \end{bmatrix} \tag{1.24}$$

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 [6] de cette matrice est :

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

6. En remplaçant la relation 1.20 dans la la relation 1.25 on obtient $H.(cG)^t = H.G^t.c^t = 0$, et comme cette relation est valable pour n'importe quelle séquence d'information c, donc $H.G^t = 0$.

$$H.x^t = 0 \tag{1.25}$$

$$H.x^t = 0 H.G^t = 0 \tag{1.26}$$

Dans le cas symétrique la matrice H devient comme suit :

$$H_{syst} = [I_M|R] \tag{1.27}$$

$$I_M = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{1.28}$$

et

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ r_{21} & r_{22} & \cdots & r_{2k} \\ \vdots & \ddots & \vdots & \\ r_{M1} & r_{M2} & \cdots & r_{Mk} \end{bmatrix} \tag{1.29}$$

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 :

$$r = x + n = c.G + n \tag{1.30}$$

A partir de 1.30 et 1.25, on aura :

$$H.r^t = H.x^t + H.n^t = H.n^t \tag{1.31}$$

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, sinon le vecteur r contient des bits erronés.
le calcul de syndrome est la méthodes 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].

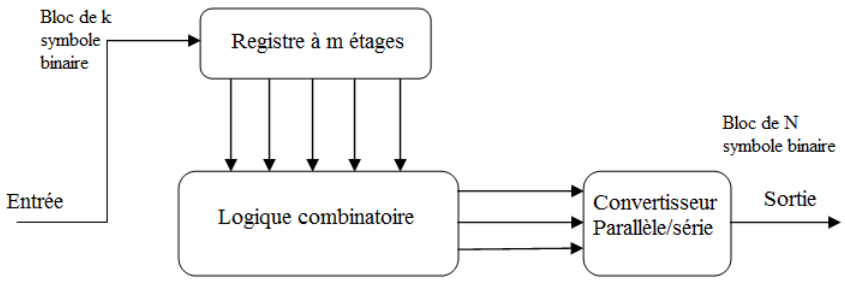

FIGURE 1.15 – Schéma de principe d'un codeur convolutif de rendement R et de mémoire m.

B 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 représente l'objet de notre étude, 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 $a_0a_1a_2...a_n$ de symboles qui passe à travers une succession de registres à décalage,dont le nombre d'étages m est appelé mémoire du code et2^m le nombre d'états possibles. La quantité $\mu = m + 1$ est appelée longueur de contrainte du code et le rapport $R = K/N$ 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 $N = 2$. at 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

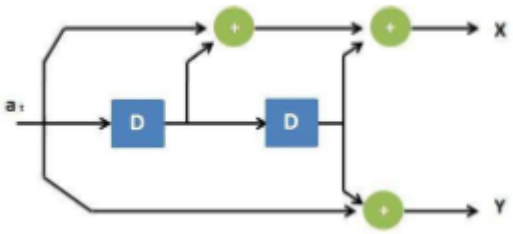

FIGURE 1.16 – Exemple d'un code convolutif de rendement $R = 1/2$.

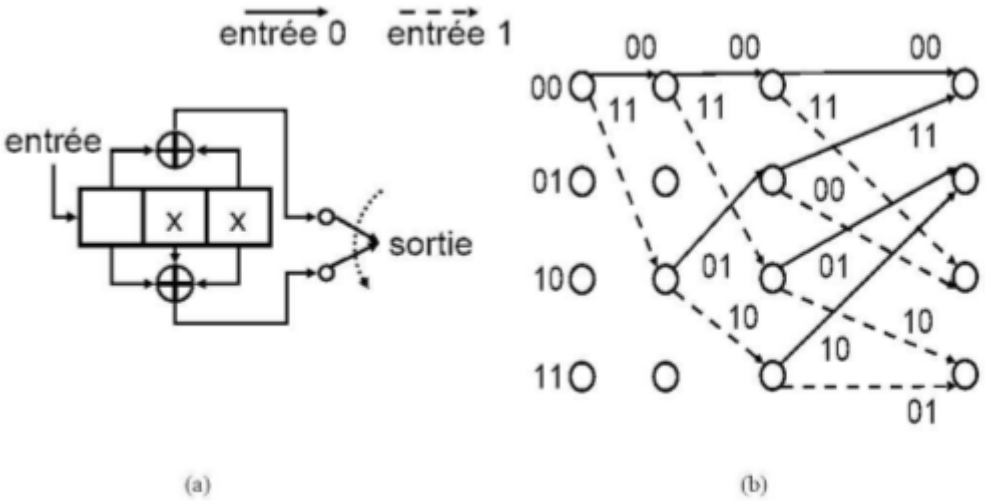

FIGURE 1.17 – (a) Un code convolutif de R=1/2. (b) Un diagramme en Treillis.

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 Figure 1.17(b))est une représentation utile pour l'algorithme de Viterbi, l'algorithme de décodage le plus utilisé pour les codes convolutifs [27], [28].

A 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 passant 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, sinon il est dit *non-systématique* [12] (voir la Figure 1.18 (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].

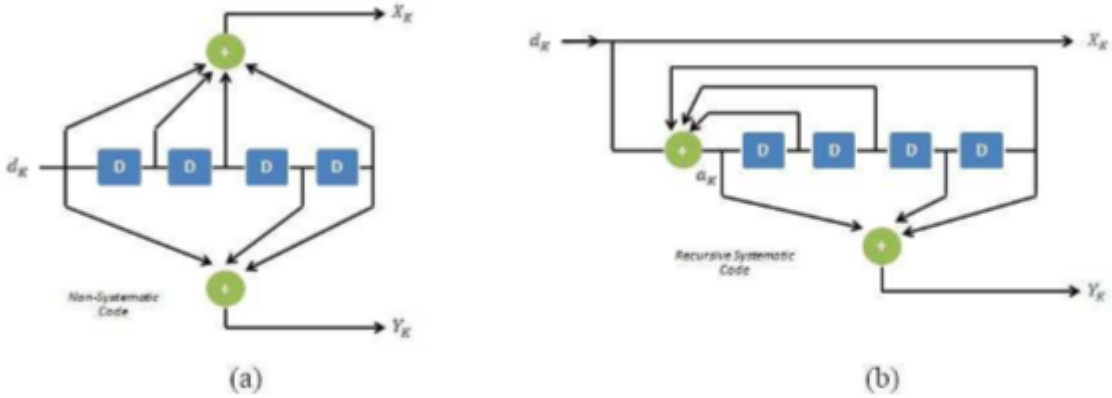

FIGURE 1.18 – (a) Un code non-systématique (NSC) (b) Un code récursif systématique (RSC).

B Les turbo-codes

Le plus célèbre des codes convolutifs est sans doute le turbo-code7 inventé par *C.Berro, A.Glavieux et 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) en codeurs de type convolutifs. La Figure 1.19 montre le cas d'une concaténation parallèle, constituée de deux codes convolutifs récursifs systématiques identiques et un entrelaceur 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 décodage s'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].

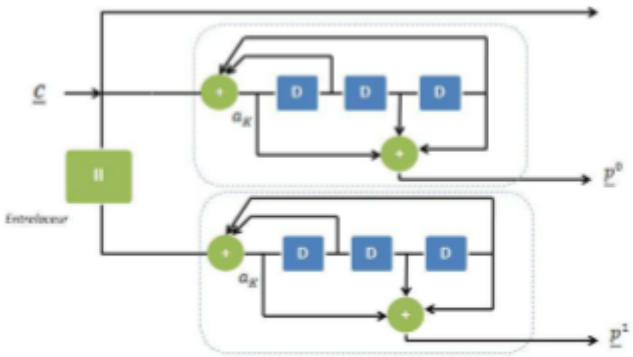

FIGURE 1.19 – Schéma de principe d'un turbo-code.

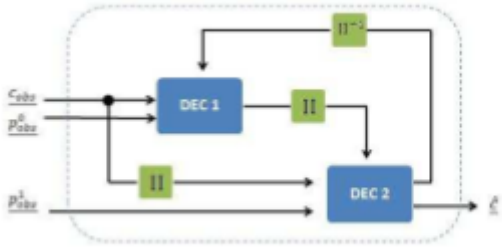

FIGURE 1.20 – Schéma de principe d'un turbo-decode.

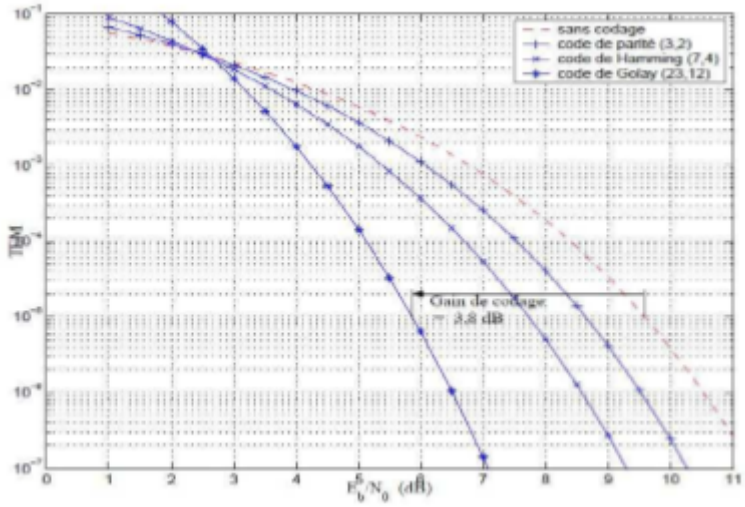

FIGURE 1.21 – Comparaison entre les performances des codes de parité, de Hamming et de Golay (Courbes reproduites de la référence [13]).

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 Figure 1.21 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 $3.8dB$, et le code de *Hamming* (7,4) apporte $1.8dB$, tandis que le code de parité apporte uniquement $1dB$ par rapport au cas sans codage. Malgré cela, les performances de ces codes restent toujours médiocres par rapport aux critères de *Varshamov et 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 Turbo-

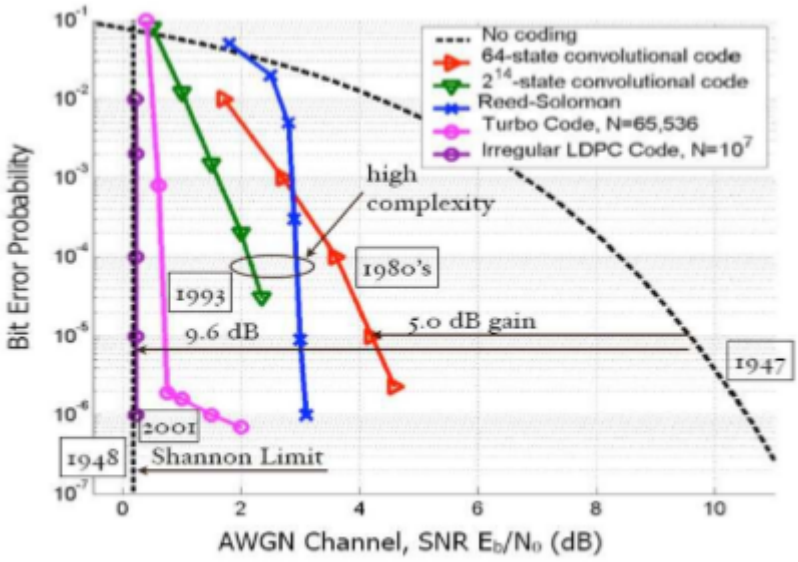

FIGURE 1.22 – Comparaison entre les performances des codes convolutifs, Reed-Solomon et les techniques de codage avancées (LDPC et Turbo-codes) (Courbes reproduites de laréférence[2]).

code avec un entrelaceur 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 2^{14} états, le code R-S et 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.

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 atteindre la limite de Shannon. Pour ce qui suit, notre étude sera focalisée sur des codes LDPC.

Chapitre 2

Codage LDPC

2.1 Historique

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 calcul 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 convolutionnel. Malgé les nombreux succès de ces codes, leur performances etaient loin de répondre aux exigences théoriques des limites établies par *Shanon* dans sa publication majeur en 1948. Malgré des décennies de tentatives, les chercheurs ont échoué semble-t-il à surpasser ces contraintes pratique-théoriques.
Cette relative quiétude propre au domaine de codage, serait ultérieurement transformée par l'introduction des turbo-codes proposés par *Berrou, Glavieux et Thitimajshima* en 1993, où, touts les moyens de succès des codes correcteur 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 information 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 lutter pour comprendre pourquoi les turbo-codes ne fonctionnaient pas aussi bien qu'il 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 code LDPC de *Gallager* a été faite par de nombreux chercheur 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 technique 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

nombreuse manières de codage sont 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 standard de communications.

2.2 Fondement théorique et mathématique

Comme le suggère leur nom, les codes LDPC sont une classe des codes en bloc, avec des matrices de contrôle de parité ne contenant qu'un nombre très petit de '1'. c'est donc sa particularité d'être "creuse" ou " sparse " qui garantie à la fois une complexité de décodage qui augmente linéairement avec longueur du code, et une distance minimale qui elle aussi augmente linéairement avec la longueur du code.
A part le critère de " sparsness " sur la matrice H, un code LDPC est similaire à n'importe code en bloc. En effet les codes en bloc existants peuvent être utilisés avec succès, par les algorithmes de décodage itératifs relatifs aux code LDPC s'ils peuvent être représentés par des matrices creuses. Mais en réalité il est rarement possible de trouver une matrice de contrôle de parité vérifiant le critère de sparsness pour un code existant. Mais plutôt les codes LDPC sont conçus de manière à construire une matrice de contrôle de parité creuse pour déterminer ensuite la matrice génératrice permettant de réaliser l'opération d'encodage.
La différence majeur entre les codes LDPC et les codes en bloc classiques, est la manière dont ils sont décodés. ces dernières sont généralement décoder via l'algorithme de décodage ML et sont souvent court, et conçu algébriquement pour rendre la tâche moins complexe. cependant les codes LDPC sont décodés itérative-ment se basant sur la représentation graphique de leur matrice de contrôle de parité, ainsi leur conception est basée sur les propriétés de leur matrice H.

2.3 La classe des codes LDPC

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

2.3.1 Codes réguliers

Les codes réguliers ont été les premiers à être introduits lorsque R. Gallager a introduit les codes LDPC en 1962 [31]. La régularité de ces codes est spécifiée par le nombre constant de "1" dans les lignes et les colonnes de la matrice H, c'est-à-dire que w_r et w_c sont constants et reliés par la relation suivante :

$$w_r = w_c \frac{N}{M} \tag{2.1}$$

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 (creuse, peu dense).
Dans le cas des codes LDPC réguliers, le rendement R (qu'on a défini dans la section 1.3.1

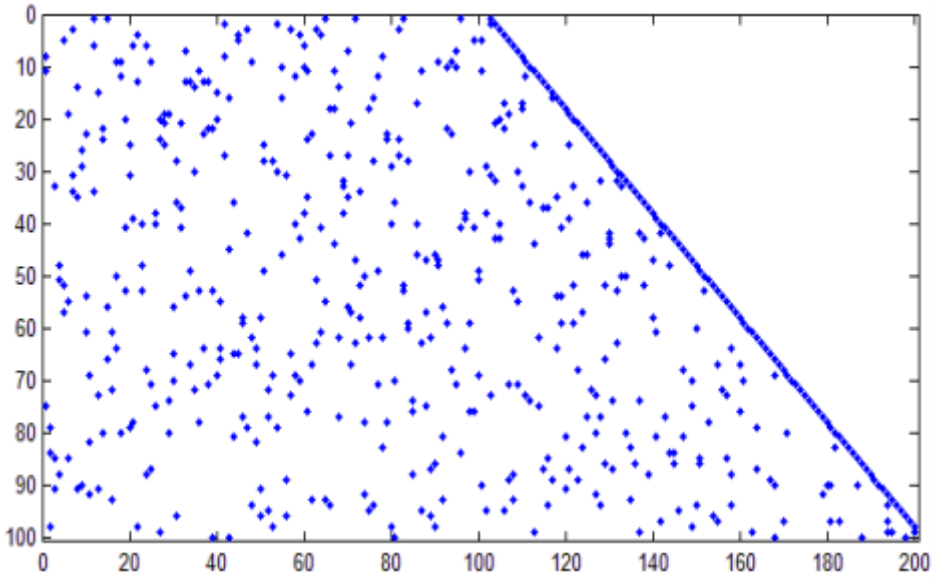

FIGURE 2.1 – Représentation d'une matrice irrégulière $N = 2000$

peut s'écrire de la manière suivante :

$$R = 1 - \frac{w_c}{w_r} \tag{2.2}$$

Un exemple d'une matrice régulière est la matrice de R.Gallager 2.3

2.3.2 Codes irréguliers

Dans le cas où la distribution des éléments non nuls est non uniforme, ces codes LDPC sont 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 **??**.
Un exemple d'un code irrégulier est illustré par la matrice repeat-accumulate 2.5 et la figure 2.1

Selon les travaux de Ludy et al. présentés dans [32] et [**reference49**], la performance d'un code LDPC irrégulier bien construit dépasse celle d'un code régulier. La Figures **??** montre une comparaison de performances entre un code régulier et un code irrégulier

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 matrices de contrôle de parité sont divisés sur trois 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 ensemble sont 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

$$H = \left[\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{2.3}$$

Une autre construction très répandue des codes LDPC est la méthode proposée par MacKay et Neal. dans celle-ci le colonnes de H sont ajoutées une par une du gauche vers la droite. le poids de chaque colonne est choisis de façon à obtenir la 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 y a 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 degré de distribution des lignes seront obtenus

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \mathit{1} & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \mathit{1} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & \mathit{1} & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Une méthode de construction des codes LDPC appelés *repeat-accumulate* possède un poids 2-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.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.5}$$

La figure 2.2 montre la distribution des '1' pour quelques méthodes de constructions.

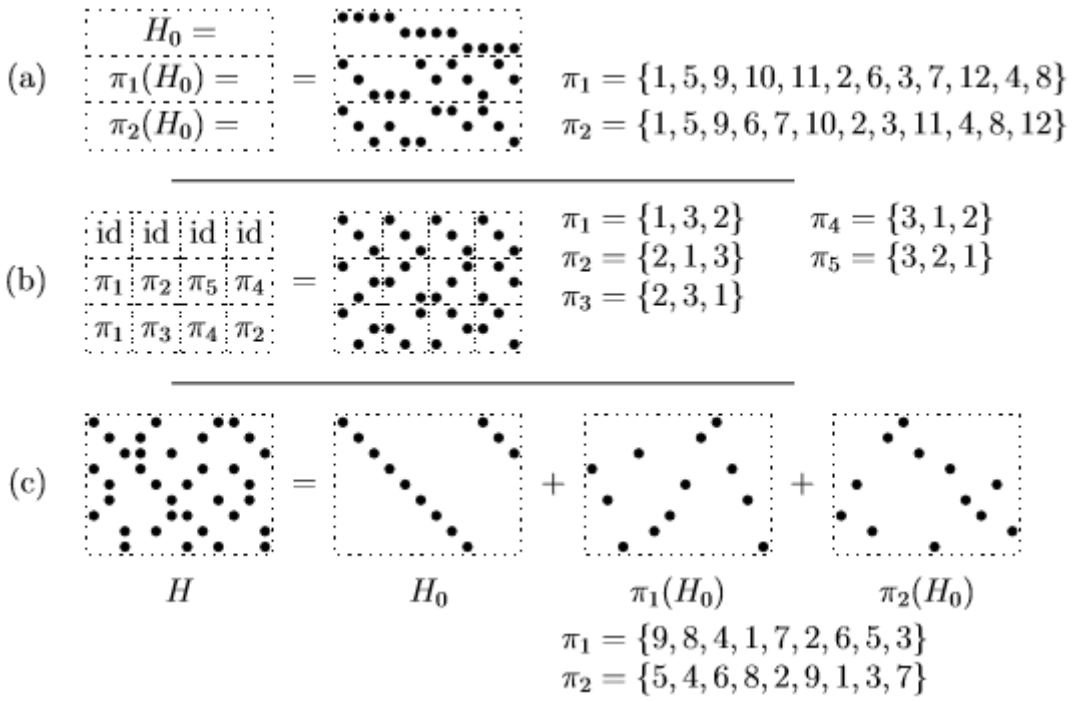

FIGURE 2.2 – Quelques constructions aléatoire de codes réguliers basées sur la méthode de Gallager (a) celle de McKay (b,c). Exemple de code régulier de taille $N = 12$. Les permutation peuvent aussi par rapport aux colonnes que par rapport aux lignes (a,b).

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 [33], ou plus généralement graphe factoriel [34]. Ce graphe contient deux types de noeuds, les noeuds de données (variable-nodes) représentant le mot de code et les noeuds fonctionnels ou noeuds de contrôle (check-nodes) correspondant aux contraintes de parité. Un noeud de données i est relié à un noeud 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 noeuds de données seront représentés par des cercles et les noeuds de contrôle par des carrés.
La figure **??** est une représentation graphique de la matrice de contrôle de parité 2.6

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{2.6}$$

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

Quand le nombre de branches connectées aux différents types de noeuds est constant, on parle alors d'un code LDPC régulier. Par conséquent chaque bit du mot de code

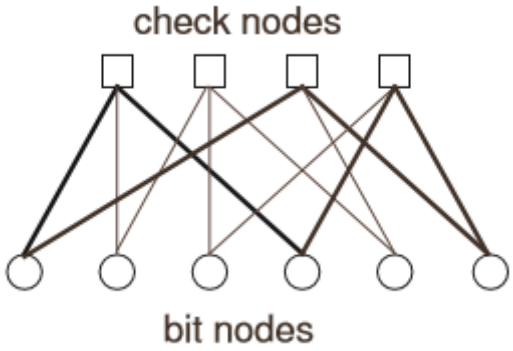

FIGURE 2.3 – Représentation graphique d'une matrice irrégulière $N = 2000$

participe à un 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 noeuds varie de façon irrégulière. Pour décrire ces codes,il est d'usage de spécifier l'irrégularité d'un code à travers deux polynômes $\lambda(x)$ et $\sigma(x)$ [16] :

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \tag{2.7}$$

$$\sigma(x) = \sum_i \sigma_i x^{i-1} \tag{2.8}$$

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

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 noeud 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].

La figure 2.4 est la représentation d'un cycle-4 par l'intermédiaire du graphe de Tanner.

Le graphe de Tanner est une représentation graphique simple des codes LDPC. Ce graphe permet notamment d'illustrer les algorithmes de décodage présentés dans le chapitre suivant.

2.6 Les codes quasi-cycliques

Un code est dit quasi cyclique 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

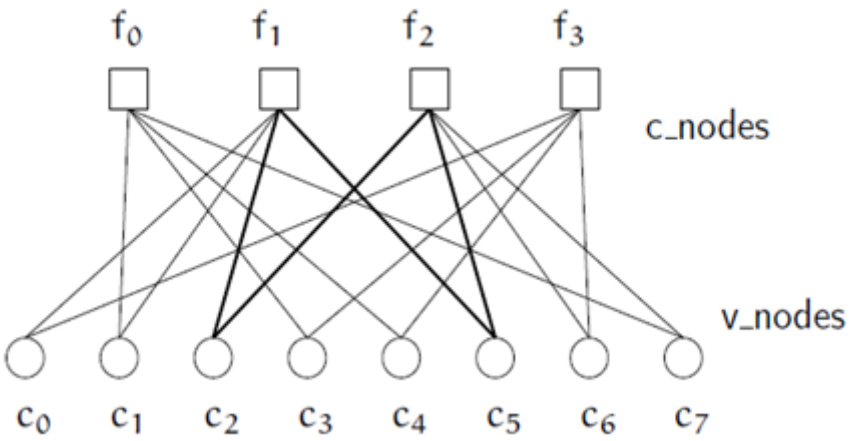

FIGURE 2.4 – Représentation graphique d'un cycle-4 par le graphe de Tanner

décrit par la matrice de contrôle de parité suivante :

$$H = \begin{bmatrix} A1 & A2 & \cdots & Al \end{bmatrix} \tag{2.9}$$

Où $A1, ..., Al$ est une matrice binaire circulaire de taille $v * v$.

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

$$G = \begin{bmatrix} & & Al^{-1}A1 \\ & I_{v(l-1)} & Al^{-1}A2 \\ & & \vdots \\ & & Al^{-1}Al_{-1} \end{bmatrix} \tag{2.10}$$

G est donc une matrice quasi-cyclique de longueur vl 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.
La figure 2.5 montre le graphe de Tanner de la matrice cyclique 2.11 :

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

2.7 Ordonnancement

2.8 Opérations d'encodage

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

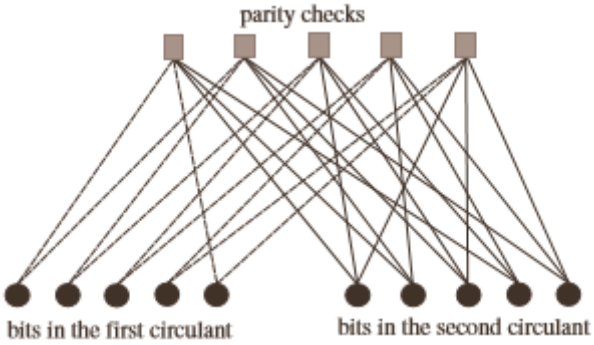

FIGURE 2.5 – Graphe de Tanner du code quasi-cyclique représenté par 2.11

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

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

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

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

exemple

Nous souhaitons encoder le code LDPC $(N = 10, rate - 1/2)$

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

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 élémentaire 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 élémentaire 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

$$H_r = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

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 partout 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 :

$$H_{rr} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

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) :

$$H_{std} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

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 :

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

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

$$H^p = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.16}$$

Ayant les même 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 :

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.17}$$

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 :

$$c = uG \tag{2.18}$$

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.

$$H_t = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{2.19}$$

2.9 conclusion

Ce chapitre a fait l'objet d'une étude détaillée des codes LDPC nous retenons ce qui suit :

- le codage peut être réalisé de manière linéaire en temps car la matrice génératrice peut être réalisée à l aide de simples registre à décalage (figure 5.8).
- 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

Principaux algorithmes de décodage

La classe des algorithmes de décodage utilisés pour décoder les codes LDPC sont communément(collectivement) appelés message-passing algorithmes puisque leur opération peut ètre illustré(expliqué) par le passage de l'information(message) à décoder à travers les noeuds du graphe de Tanner. 2.3
Chaque noeud appartenant au graphe de Tanner travaille en isolation, ne pouvant par conséquent avoir accès qu'aux informations contenu dans les messages provenant des autres noeuds 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 noeuds de bits aux les noeuds 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 noeuds, nous citons comme exemple : bit flipping, Min-Sum, qui serons présentés plus loin dans ce chapitre.
Dans certains types d'algorithmes, tel que " bit flipping decoding " le message est binaire mais il est probabiliste dans belief propagationöù la valeur de probabilité attribué aux messages représente la quantité de croyance pour le bit du codeword. dans cette situation, il est souvent convenable de représenter la probabilité par le rapport de vraisemblance, et une fois encore cette 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 .

3.1 Algorithme bit flipping

Bit-flipping algorithme est un*message-passing* algorithme à décision dure. Ainsi, une décision dure est appliqué sur chaque bit reçu par le détecteur avant qu'elle ne soit transmise au décodeur.
De la même manière, le messages circulant le long du graphe de Tanner sont binaire : les noeuds de bits envoient un message déclarant si c'est zéro ou un, a leur tour les noeuds de contrôle renvoient un autre message pour chacun des noeuds avec lesquelles sont reliés, rapportant la valeur binaire en se basant sur les information disponible a leur niveau. Comme les noeuds de contrôles représentent les équations de contrôle de parité, la détermination de la valeur de bit à renvoyer à chacun des noeuds de bits se fait en calculant la somme modulo-2 de tout les bits qui y arrivent dans chaque itération autrement dit, les bits qui constituent cette équation.

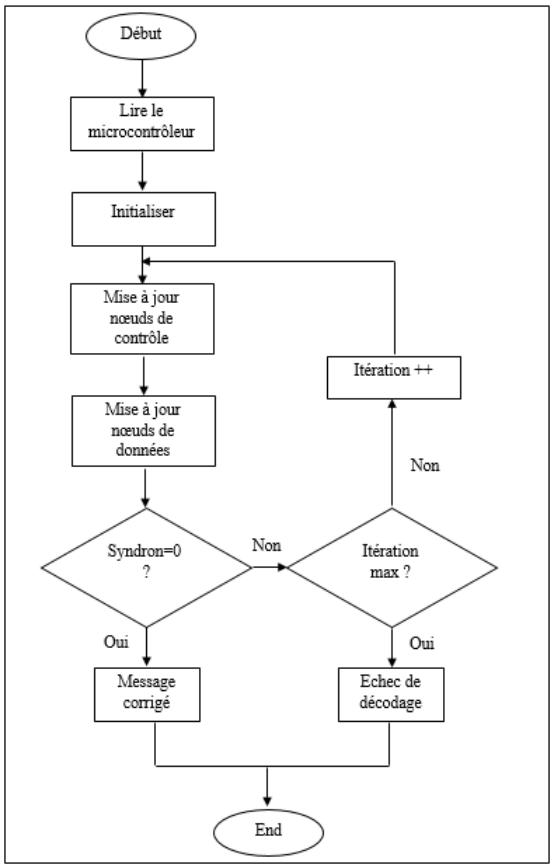

FIGURE 3.1 – Organigramme de l'algorithme de décodage " bit-flippin "

si la majorité des valeur binaires retournant aux noeuds de bits sont différent de la valeur préalablement envoyé par ces derniers i.e à l'itération i, ils changent aisi leur valeur, jusqu'à ce que toutes les équation de contrôle de parité seront satisfaite i.e la somme modulo-2 soit égale à zéro ou que le maximum d'itérations permises est atteint et l'algorithme s'arrète.
Le décodeur *bit flipping* l'interrompe quelque soit le code valide trouvé en vérifiant que toutes les équations de contrôle de parité soit satisfaite. ceci est vrai pour tout les décodeur LDPC *message-passing* et présente à cet effet deux importants avantages : premièrement, éviter toute itération additionnelle, une fois la solution est trouvée. deuxièmement, toute tentative ne convergeant par vers un mot de code est toujours détectée.
Cet algorithme est conçu sur le principe que si un bit est impliqué dans un grand nombre d'équation de contrôle de parité incorrectes il serait susceptible que ce bit soit lui même incorrect. la SPARSENESS (faible densité) de la matrice de contrôlé de parité permet de disperser les bits sur les noeuds de contrôle pour que celles-ci ne vont probablement pas contenir le même ensemble de bit d'un même mot de code.
Un organigramme de cet algorithme est illustré dans la figure 3.1.

3.2 Algorithme sum-Product (log-domain)

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 probabilité pour représenter chaque décision prise au niveau des noeuds, en ef-

fet, les bit 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 :

$$p(x = 1) = 1 - p(x = 0)$$

la fonction de maximum de vraisemblance est utilisée pour représenter le métrique d'une variable binaire par une seul valeur :

$$L(x) = \log\left(\frac{P(x = 0)}{P(x = 1)}\right) \tag{3.1}$$

Où log signifie $\log_e$.

si $p(x = 0) > p(x = 1)$ alors $L(x)$ est positive, et plus la différence entre $p(x = 0)$ et $p(x = 1)$ plus $L(x)$ est grand, et réciproquement. cependant, le signe de L(x) fourni la décision dure (hard) sur x tandis que l'amplitude de $|L(x)|$ représente la fiabilité de cette décision.
Pour obtenir la probabilité à partir de la fonction de maximum de vraisemblance on note :

$$P(x = 1) = \frac{\frac{P(x=1)}{P(x=0)}}{\frac{1+p(x=1)}{p(x=0)}} = \frac{\exp(-L(x))}{1 + \exp(-L(x))} \tag{3.2}$$

Et

$$P(x = 0) = \frac{\frac{P(x=0)}{P(x=1}}{\frac{1+p(x=0)}{p(x=1)}} = \frac{\exp(L(x))}{1 + \exp(L(x))} \tag{3.3}$$

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

Le but du décodeur *sum-product* est de calculer le *maximum a posteriori probability MAP* pour chaque bit du cmot de code, $P_i = P\{C_i = 1|N\}$, qui est la probabilité que le ième bit soit égale à 1 sachant l'événement N où toutes les équations de contrôle de parités soit satisfaite. l'extra-information sur le i-ème bit est appelée *information extrinsèque*.
cet algorithme calcul itérativement une approximation de la valeur MAP pour chaque bit du code. Pourtant, les probabilités a posteriori retournée 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 lui 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 noeud de contrôle

j vers le noeud de bit i, $E_{j,i}$, est le LLR de la probabilité que le bit i engendre la satisfaction de l'équation de contrôle de parité j.

$$P_{j,i}^{ext} = 1/2 - 1/2 \prod_{i' EB_j, i'=i} (1 - 2P_{i'}^{int}) \tag{3.4}$$

Où $P_{i'}^{int}$ est l'estimation actuelle disponible au niveau du noeud de contrôle j, de la probabilité que le bit i' 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 la fonction de maximum de vraisemblance il devient :

$$E_{j,i} = LLR(P_{j,i}^{ext}) = log(\frac{1 - P_{j,i}^{ext}}{P_{j,i}^{ext}}) \tag{3.5}$$

et en remplaçant par 3.4 :

$$E_{j,i} = log(\frac{1/2 + 1/2 \prod_{i' EB_j, i'=i} (1 - 2P_{i'}^{int})}{1/2 - 1/2 \prod_{i' EB_j, i'=i} (1 - 2P_{i'}^{int})}) \tag{3.6}$$

s'appuyant sur la relation suivante :

$$\tanh(1/2 \log(\frac{1-p}{p})) = 1 - 2p \tag{3.7}$$

on obtient :

$$E_{j,i} = \log(\frac{1 + \prod_{i' EB_{j,i'=i}} \tanh(M_{j,i'}/2)}{1 - \prod_{i' EB_{j,i'=i}} \tanh(M_{j,i'}/2)}) \tag{3.8}$$

où

$$M_{j,i'} = LLR(P_{j,i'}^{int}) = \log(\frac{1 - P_{j,i'}^{int}}{P_{j,i'}^{int}}) \tag{3.9}$$

Autrement, s'appuyant sur la relation suivante :

$$2 \tanh^{-1}(p) = \log(\frac{1+p}{1-p}) \tag{3.10}$$

On peut écrire :

$$E_{j,i} = 2 \tanh^{-1}(\prod_{i' EB_{j,i'=i}} (\tanh(M_{j,i'}/2))) \tag{3.11}$$

3.3 Algorithme Min-Sum

L'algorithme *Sum-Product* peut être modifié pour réduire la complexité de l'implémentation du décodeur.

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

Pour pouvoir remplacer le produit par la somme. pour plus de simplicité dans l'écriture nous considérons :

$$\prod_{i'} = \prod_{i'EB_{j,i'}i'=i} \tag{3.13}$$

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

$$M_{j,i'} = a_{j,i'}b_{j,i'}$$

où :

$$a_{j,i'} = \text{sign}(\text{M}_{\text{j,i'}})b_{j,i'} = |M_{j,i'}| \tag{3.14}$$

En utilisant cette notation il vient :

$$\prod_{i}^{'}(\tanh(M_{j,i'}/2) = \prod_{i'}(a_{j,i'}) \prod_{i'}(b_{j,i'}/2) \tag{3.15}$$

Donc, l'équation 28 devient :

$$E_{j,i} = 2\tanh^{-}1(\prod_{i'}(a_{j,i'}) \prod_{i'}(\tanh(b_{j,i'}/2))) = (\prod_{i'}(a_{j,i'}))2\tanh^{-}1(\prod_{i'}(\tanh(b_{j,i'}/2))) \tag{3.16}$$

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

$$E_{j,i} = (\prod_{i'}(a_{j,i'}))2\tanh^{-}1\log^{-}1\log(\prod_{i'}\tanh(b_{j,i'}/2)) \tag{3.17}$$

$$= (\prod_{i}^{'}(a_{j,i'}))2\tanh^{-}1\log^{-}1(\sum_{i}^{'}\log\tanh(b_{j,i'}/2)) \tag{3.18}$$

Nous définissons :

$$fi(x) = -\log\tanh(x/2) = \log\frac{\exp(x)+1}{\exp(x)-1} \tag{3.19}$$

Et puisque la fonction $fi(x)$ est bijective (figure 3.2 :

$$fi(fi(x)) = \log\frac{\exp(fi(x))+1}{\exp(fi(x))-1} = x \tag{3.20}$$

Nous avons $fi^{-1} = fi$, finalement l'équation 213 devient :

$$E_{j,i} = (\prod_{i'}(a_{j,i'}))fi(\sum_{i}^{'}(fi(b_{j,i'}))) \tag{3.21}$$

Le produit des signes peut être calculer par le moyen de l'addition modulo-2 de la décision dure (hard) sur chaque $M_{j,i'}$ tandis que la fonction fi serait facilement implémentée en se réfèrent a sa tabulation.
En d'autres termes, l'algorithme *Min-Sum*, simplifie le calcul de l'équation 28, 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 :

$$E_{j,i} = (\prod_{i'}(\mathrm{sign(M_{j,i'})}))\min(|M_{j,i'}|) \tag{3.22}$$

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

3.4 Ordonnancement

L'ordonnancement d'un algorithme de décodage est l'ordre dans lequel les messages devraient se propager dans le graphe de Tanner. deux type d'ordonnancement sont distingués dans cette section [35].

- Ordonnancement bi-directionnel : c'est un ordonnancement orienté en série, où seulement les messages pertinent sont traités et envoyés aux noeuds avec lesquels ils sont reliés.
- Ordonnancement à flux : est un ordonnancement orienté en parallèle, où tout les noeuds sont traités en même temps, dans ce cas ce cas les messages arrivés se comporte comme des déclencheur pour les processeur des noeuds.

Pour une exploitation pratique des codes LDPC, l'ordonnancement à flux est souvent utilisé, le comportement de chaque processeur des noeuds devient plus simple. Mais la question qui se pose déjà est dans quel ordre les processeurs des noeuds calculent les messages de sortie ? ce choix peut engendrer des cycles inutiles dans le graphe de Tanner et provoquer une divergence au niveau du décodage, cependant trois familles d'ordonnancement existent dans la littérature que nous allons décrire :

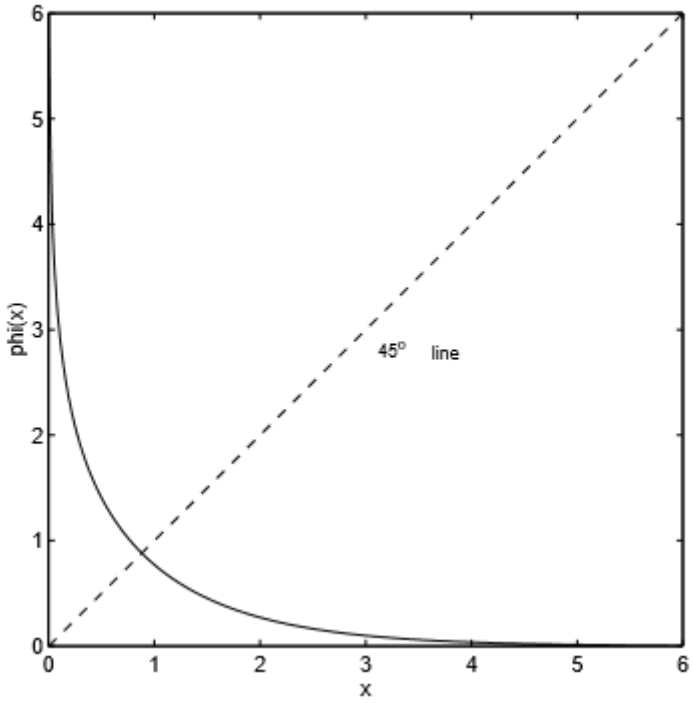

FIGURE 3.2 – Graphe de la fonction fi

Ordonnancement à écoulement

Ce type désigne l'au-delà de la façon classique d'ordonnancement des algorithmes BP. le sens d'écoulement, peut avoir été changé depuis son original sens [35]. dans ce type d'ordonnancement, les noeuds du même genre sont mise à jours dans un premier temps ensuite l'autre types de noeuds. (Voir figure 3.3 a). La mise à jour d'un type de noeud peut être faite noeud après noeud (en série) ou tous à la fois : cela n'affectera pas les valeurs de sorties

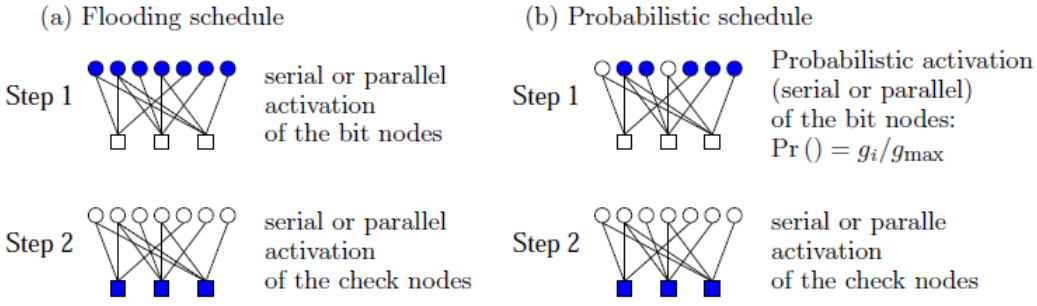

FIGURE 3.3 – Comparaison des performances entre les différents algorithmes de décodage pour N=1024.

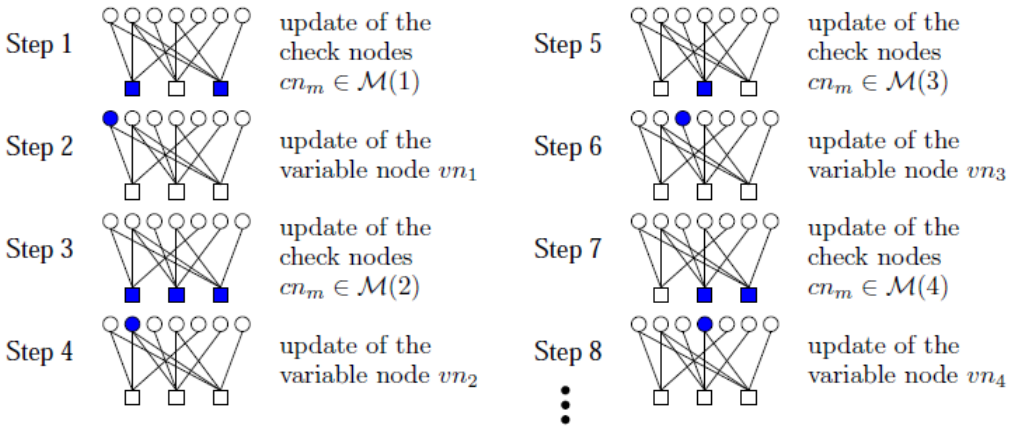

FIGURE 3.4 – Les première étape de décodage dans le cas de l'ordonnancement horizontale lors 'une itération i

Ordonnancement probabiliste

L'auteur de [36] décrit l'ordonnancement probabiliste. La première idée est de se débarrasser de l'auto confirmation des messages causée par les cycles du graphe de Tanner : parfois cela est évité par la non activation de quelques noeuds, qui, sont censé l'être dans le cas de l'ordonnancement à écoulement. Voir figure 3.3 b).

Ordonnancement verticale

L'auteur de [37] a proposé " shuffle " algorithme BP qui converge plus rapidement qu'un algorithme BP classique. L'idée est de mettre à jour les informations dés que les messages soient calculés, pour permettre au prochain processeur de noeud d'utiliser plus d'informations avant qu'il ne se mette à jour. Cet ordonnancement opère le longs des noeuds de données, ce qui signifie que tout les noeuds de données sont traités l'un après l'autre.
Figure 3.4 montre un exemple des premières étapes prises pour une itération i. l'ordonnancement d'une itération i est en série : les noeuds de contrôle impliqués dans le premier noeud de données sont traités (étape 1) ce noeud de données est ensuite mis à jour (étape 2). Ensuite les noeuds de contrôle impliquant le deuxièmes noeud de donnée sont traités (étape 3), ce dernier est mis à jour (étape 4), ainsi de suite, jusqu'à mettre à jour tout les noeuds de données.

Ordonnancement horizontale

L'auteur de [38] a proposer une architecture en série s'appuyant sur un ordonnancement qui consiste à traiter les processeurs des équations de contrôle de parité en série. l'information envoyée au noeuds de contrôle désigné par le traitement, disons CNm prend en compte l'information de l'itération précédente et celle de l'itération actuelle qui a été mise à jour par tout les autre noeuds de contrôle précédent CNm0, $m0 < m$. cet ordonnancement est souvent associé par l'algorithme APP, car les résultats obtenus avec l'algorithme BP sont plus ou moins médiocre.

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 l'environnement MATLAB/Simulink pour implémenter toute la chaîne de simulation.
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 3.5. 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) et le tracer en fonction du SNR, il est définit par la relation suivante :

$$BER = \frac{Nombre \quad de \quad bits \quad errons}{Nombre \quad de \quad bits \quad transmis} \tag{3.23}$$

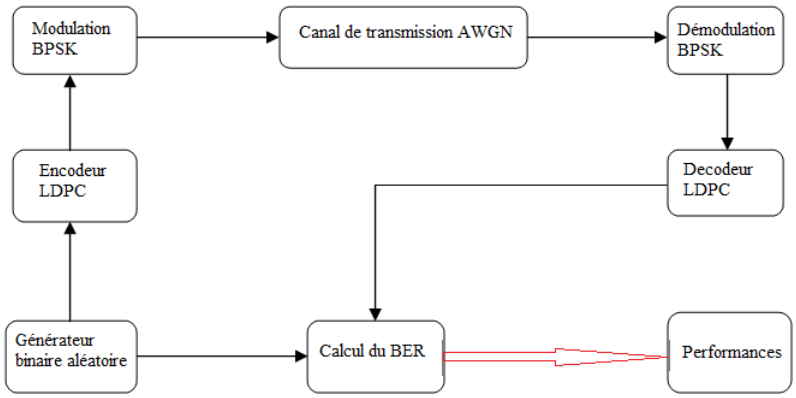

FIGURE 3.5 – Le modèle de simulation utilisé pour l'évaluation des performances des codes LDPC.

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

La matrice de contrôle de parité du standard de communication DVB-S2 est irrégulière est possédant les degrés de distribution suivants (Tableau 3.2, Tableau 3.3) :

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

Type du code LDPC	Irrégulier
Matrice de contrôle de parité	dvbs2ldpc du standard DVB-S2
Type du canal	Canal gaussien à bruit blanc additif (AWGN)
Type de modulation	BPSK (Binary Phase Shift Keying)
Critère d'évaluation des performances	BER en fonction de Eb/N0 (SNR)
Estimation de la variance du bruit σ	parfait

TABLE 3.2 – Degré de distribution des lignes de la matrice H

Lignes	Nombre de 1 par lignes
1	6
2 à 32400	7

3.5.2 Comparaison entre les différents algorithmes de décodage

La Figures 3.6 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. 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 dure (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 pondéré (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 pondéré(Soft) au lieu du décodage dure (Hard).
Pour le cas Soft, Les deux courbes données en vert et enbleu dans la Figure 3.6, permettent d'évaluer l'influence de l'approximation donnée par la relation 3.22. 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.

TABLE 3.3 – Degré de distribution des colonnes de la matrice H

Colonnes	Nombre de 1 par colonnes
1 à 12960	8
12961 à 32400 à 32400	6

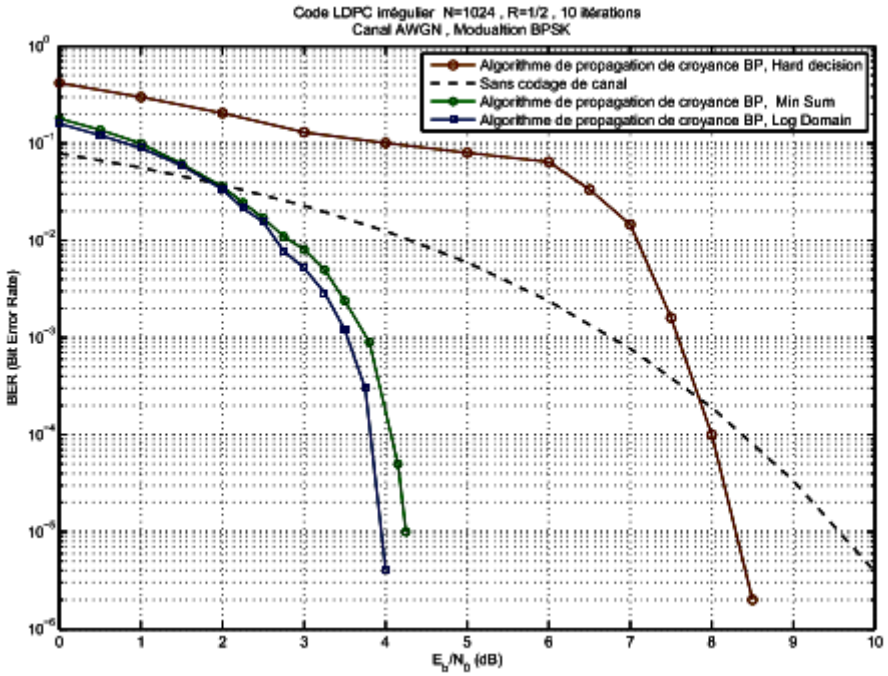

FIGURE 3.6 – Comparaison des performances entre les différents algorithmes de décodage pour N=1024.

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 3.7, on remarque qu'avec l'augmentation de N, il résulte une amélioration de performances, par exemple pour un BER égale à 10^{-3}, le code LDPC irrégulier de taille N=1024 apporte un gain de 0.8 dB par rapport à celui de N=200, et de 0.3 dB 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 3.8 et 3.9 qui correspondent respectivement au décodage pondéré (Soft) Log Domain et Min-Sum, on voit clairement l'amélioration apportée par l'augmentation de N. Par exemple la Figure 3.17 montre que pour un BER égal à 10^{-5}, le code LDPC irrégulier de taille N=1024 apporte un gain de 0.6 dB 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 $Eb/N0$ faibles et les pentes des graphes deviennent encore plus grandes. Par exemple pour $N = 1024$ et $Eb/N0 = 4.5dB$, 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.

La Figure 3.10 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 $Eb/N0 = 1.8dB$. Ces résultats confirment la puissance des codes LDPC irréguliers surtout pour N très

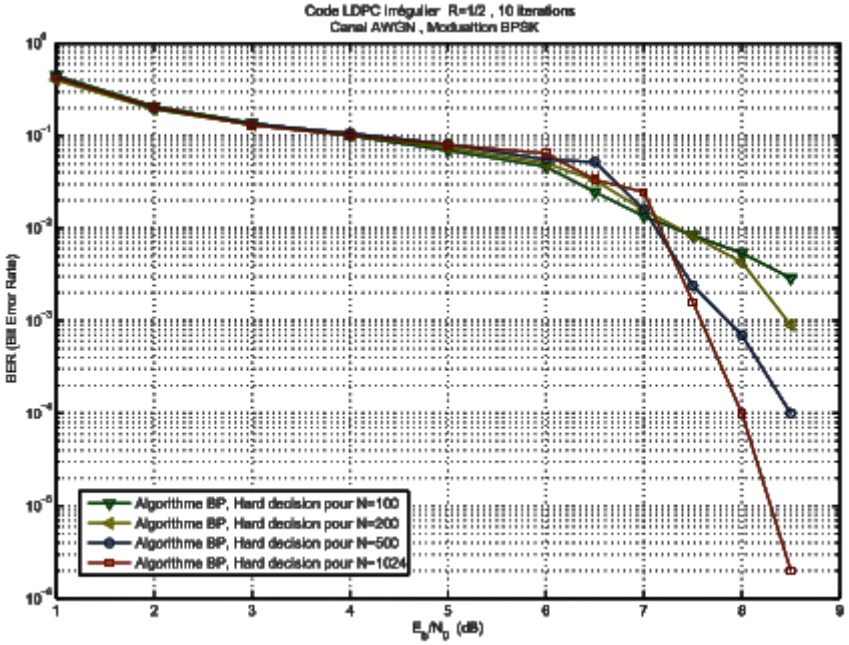

FIGURE 3.7 – Influence de la taille du code sur les performances pour un décodage dure (Hard)

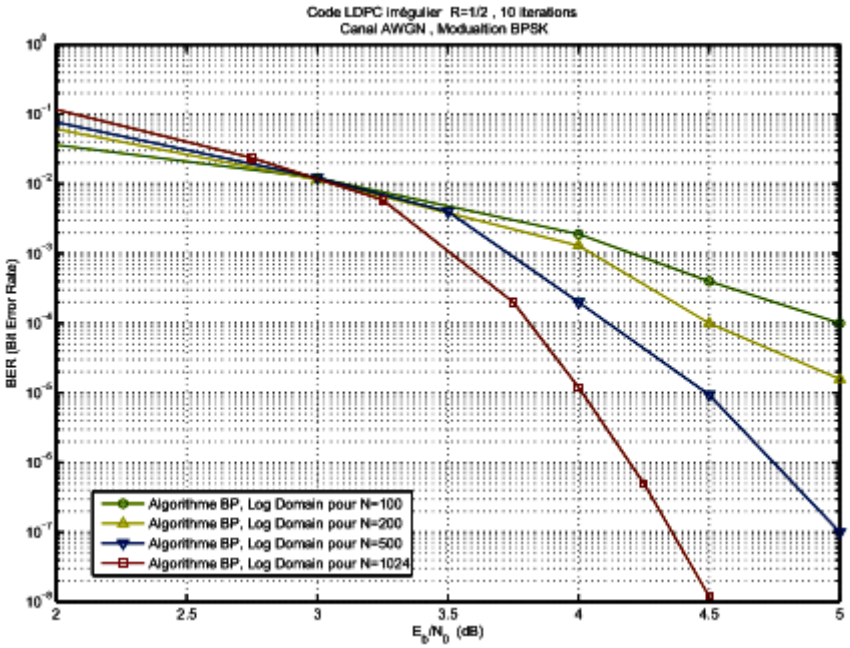

FIGURE 3.8 – Influence de la taille du code sur les performances pour l'algorithme de décodage Log Domain

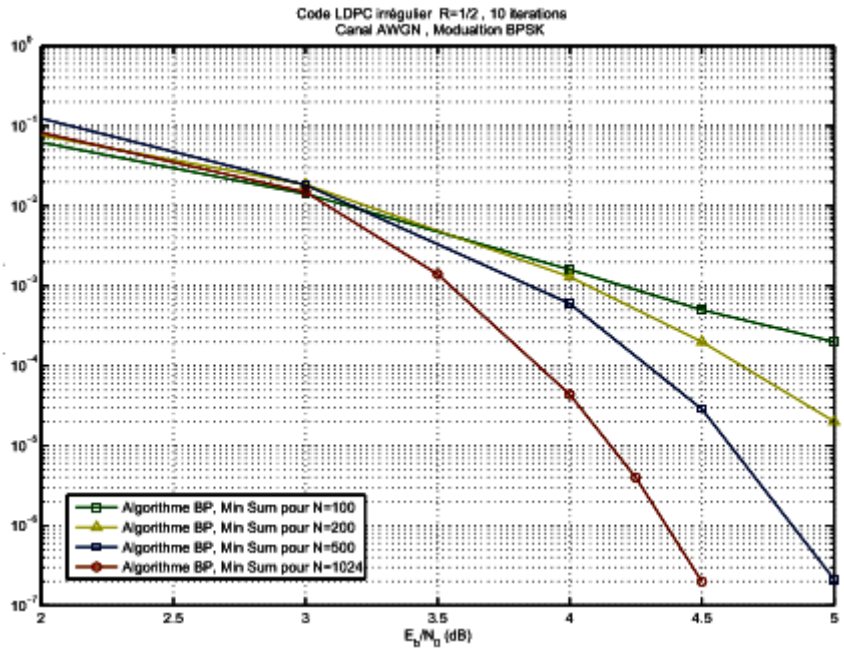

FIGURE 3.9 – Infuence de la taille du code sur les performances pour l'algorithme de décodage Min-Sum.

grand.
Les résultats obtenus dans cette partie justifient pourquoi tous les standards de communication utilisant des codes LDPC choisissent des tailles N très grandes, par exemple $N = 64800$ pour le standard DVB-S2 et $N = 8176$ pour le standard CCSDS [39] [40].

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 3.11 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 3.12 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.

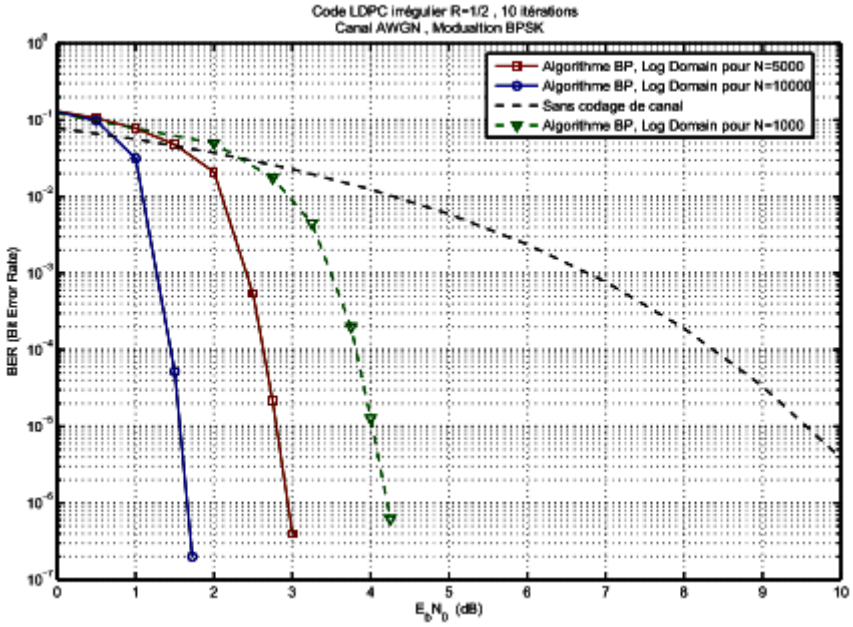

FIGURE 3.10 – Influence de la taille du code sur les performances pour l'algorithme de décodage Log-Domain.

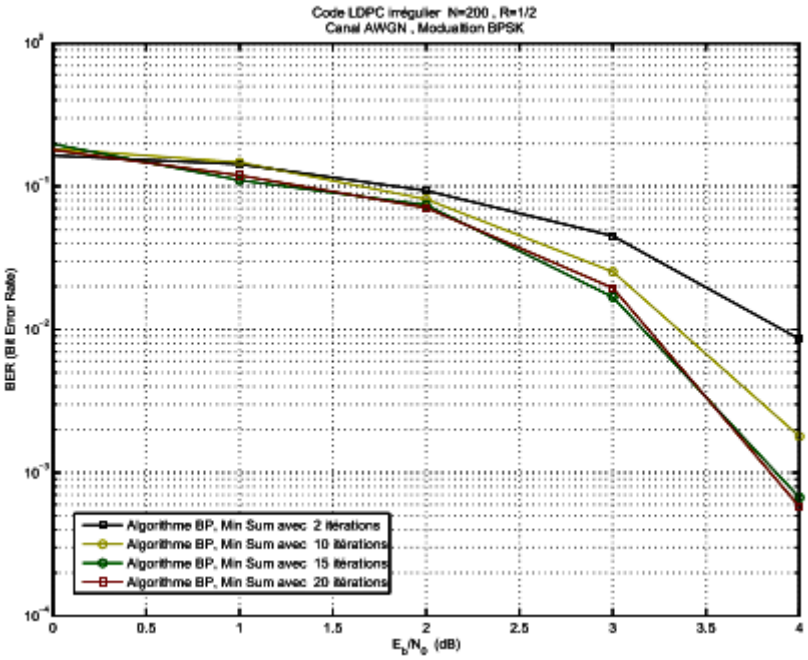

FIGURE 3.11 – Influence du nombre d'itérations sur les performances des codes LDPC pour N=200 et un décodage Min-Sum. [41]

3.5.5 Infuence 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 3.13 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 pas en $Eb/N0$) 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 $Eb/N0$. On rappelle que :

$$Eb/N0 = SNR/R \tag{3.24}$$

La Figure 3.14 montre une comparaison en terme de BER en fonction de $Eb/N0$. 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 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

3.6 conclusion

Après avoir présenté Les différents algorithmes de décodage, nous avons traités quelques exemple de simulation pour évaluer les performances des codes LDPC dans un canal gaussien, et à partir des résultats de simulation ces conclusions suivantes sont tirées :

- 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 Sum-product (Log-Domain).
- L'augmentation de la taille du ode LDPC entraîne une amélioration de performances, cette amélioration est remarquable surtout pour $N \geq 1000$.
- 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.

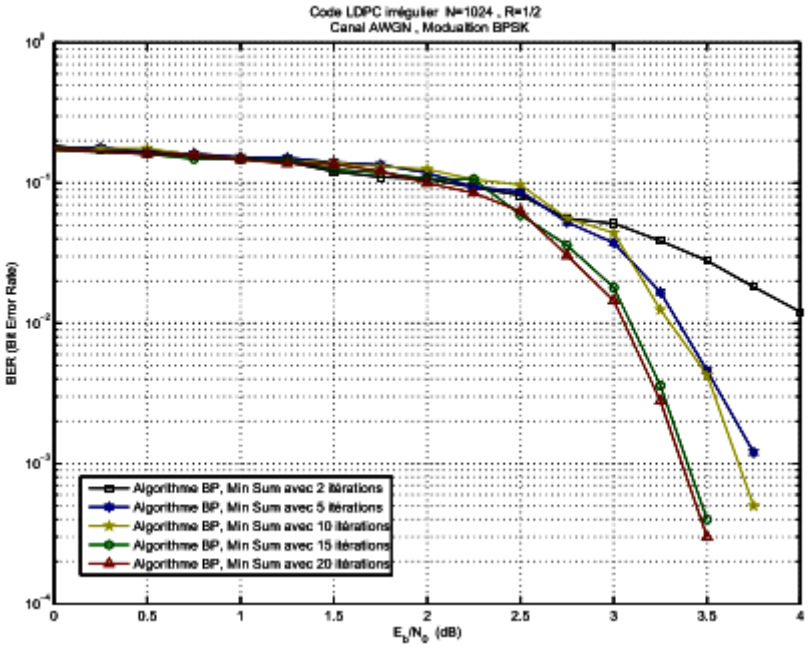

FIGURE 3.12 – Influence du nombre d'itérations sur les performances des codes LDPC pour N=1024 et un décodage Min-Sum.

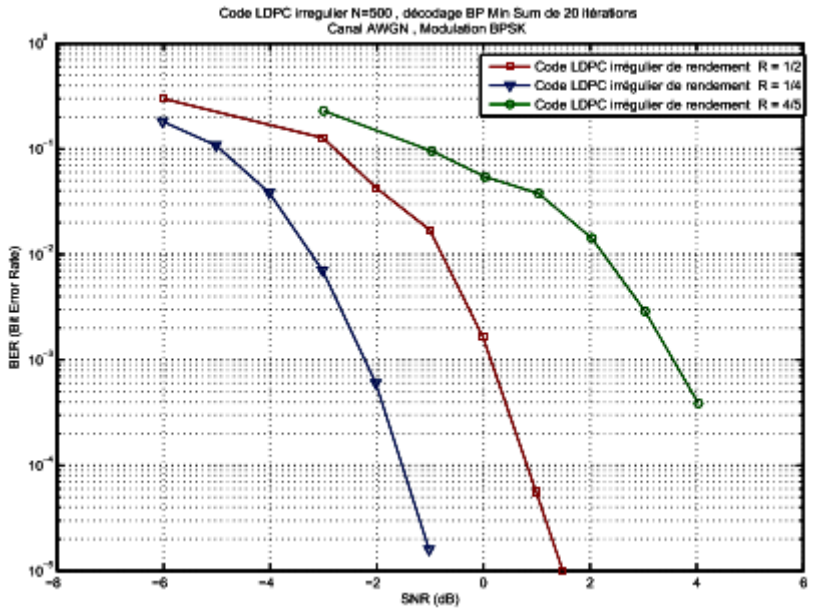

FIGURE 3.13 – Influence du rendement R sur les performances des codes LDPC (BER=f(SNR)) [41]

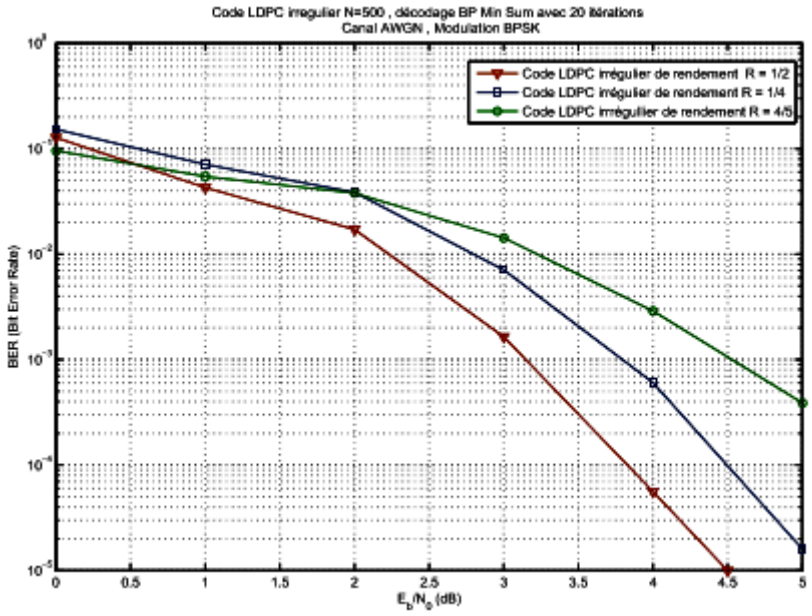

FIGURE 3.14 – Influence du rendement R sur les performances des codes LDPC (BER =f(Eb/N0)).

Chapitre 4

Les choix de conception d'un décodeur LDPC

introduction

Dans ce chapitre, nous allons définir les différents types d'architectures pour le décodage des codes LDPC basés sur l'algorithme min-sum traité dans le chapitre

4.1 Architecture des codes LDPC

Cette section traite des différents aspects d'implementations d'architectures pour les décodeurs LDPC,(section 2.7), tel que expliqué auparavant, le décodage LDPC présente une particularité inhérente qui est le parallélisme. Chaque noeud de contrôla 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 LDPC architecture :

- 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 graph 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 une le processus de décodage itératif.

Cependant, chaque élément dépend directement des contraintes relatives aux standards sur lesquelles l'architecture sera portée. dans la majorité des cas, la flexibilité est l'une des plus importants critères lorsqu' un décodage à haut débit est exigé. Par exemple, un décodeur à ultra large-bande (UWB) exige un maximum de débit de $1,024 Mbs/s$ tandis qu'un GMR-1 décodeur exige un débit inférieur à $1 Mbs/s$, mais avec un maximum de longueur du code de 64,800 bits. cependant différentes solution sont 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 spatiale (architecturale) qui reflète les possibilité d'implémentation d'un décodeur LDPC peut être divisée en plusieurs niveaux Une liste améliorée de possibilité de conception peut être trouvée dans [42]. 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, toutefois, le nombre de branche traitée reste très petit.

Se référant à ces techniques de bases, différentes solutions existe 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 montrer dans [43] pour un code LDPC irrégulier "$R = 1/2$ " avec un mot de code de longueur $N = 1.024$. Le chemin de donnée d'une architecture entièrement parallèle d'un décodeur LDPC est montrée dans la figure 4.1.

Cette figure montre deux noeuds fonctionnel, représentant un noeud de données de degré 3, et un noeud de contrôle de degré 4 respectivement. On constate que les deux noeuds 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 noeuds de contrôle aux noeuds de données et vice versa. le débit résultant du décodeur dans [43] est de 1 Gbit/s en considérant 64 itérations.
L'architecture parallèle est la plus rapide de toutes les autres architectures puisque les noeuds, 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 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

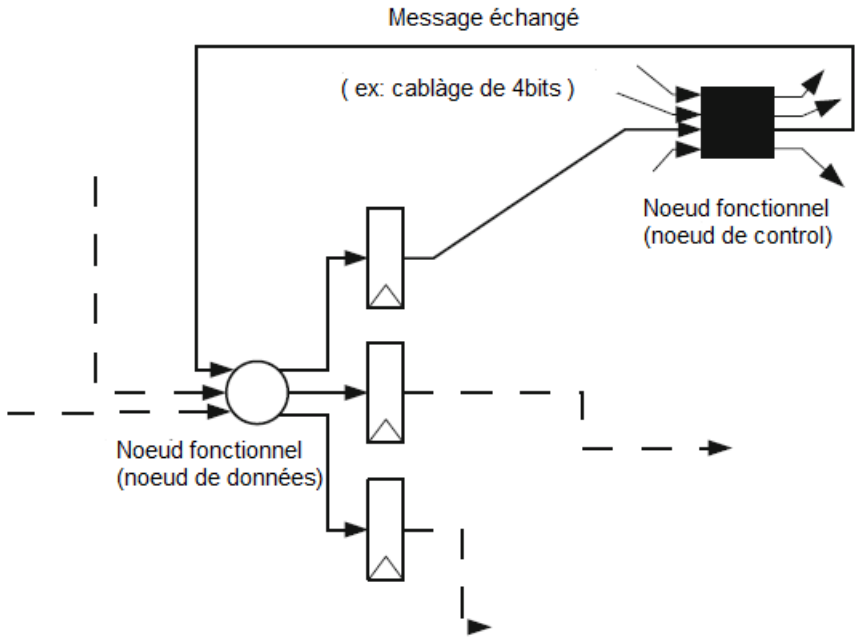

FIGURE 4.1 – Chemin de donnée d'une architecture entièrement parallèle. cette figure est dérivée de [43]

est nécessaire pour fournir plus de flexibilité au décodeur, où seulement un certain sous-ensemble de noeuds fonctionnels sont instanciés. Les noeuds de contrôle et noeuds de données sont traités essentiellement sur ces unité de traitement. un noeud 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 noeuds fonctionnel instancié peut différer, le débit est seulement détermine 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 noeud dans le graphe de Tanner qui soit physiquement réalisée en Hardware, autrement la question qui se pose est quel noeud devant être traité sur quelle unité fonctionnel 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 code21x28).

Les messages échangés entre les unités fonctionnelles noeuds de données (VNFU) et noeuds de contrôle (VNCN) devraient être stockés en mémoires (montrer la figure de la RAM). Un réseau à permutaion doit être instauré pour implémenter les connectivitées désirées pour un graphe de Tanner donné (voir figure **??**. Cela nécessite des mémoires additionnel afin de stocker le modèle de connectivitée. L'avantage majeure d'une architecture semi-parallèle est la possibilité de fournir une flexibilité, en produisant une 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.

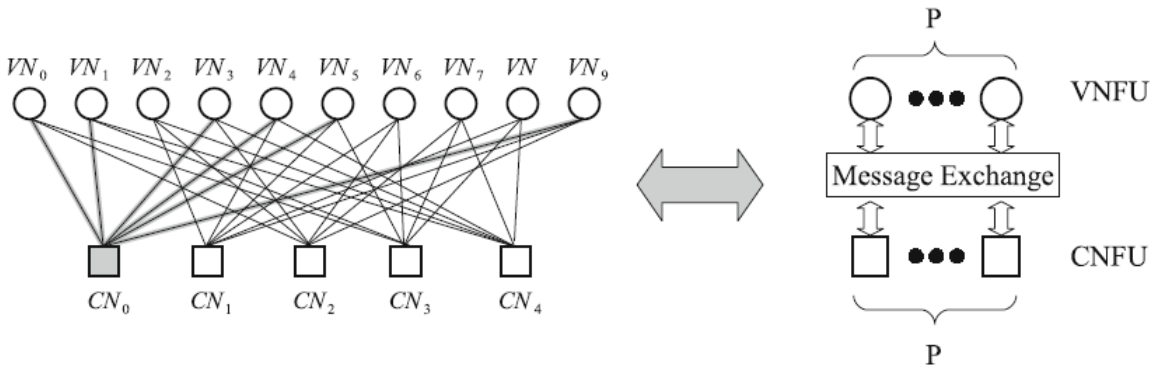

FIGURE 4.2 – Mapping du décodeur semi-parallèle à partir du graphe de Tanner

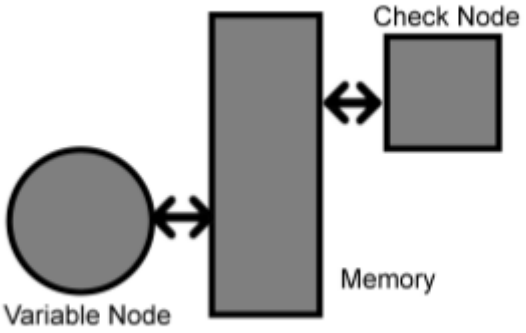

Figure 2.1: Serial Decoder Architecture.

FIGURE 4.3 – Exemple d'une architecture série

4.2.3 Architecture série

Parmi tout les types architectures, l'architecture série est la plus simple. Seulement un bloc fonctionnel pour chacun des noeud de donnée et des noeud de contrôle est instancié avec un traitement séquentielle de tout les noeuds de données et noeuds 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 [44]. 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, toutefois cette solution est rarement choisie dans le cas où la latence serait un facteur critique.

Dans la suite, seulement les architectures partiellement parallèles seront pris en compte puisqu'ils fournissent une meilleur flexibilité par rapport à une réalisation entièrement parallèle, et débit plus élevé que implémentation série. La question de flexibilité

détermine le style d'implémentation des noeuds fonctionnels qui sera présentée dans la section suivante.(chercher une figure à mettre).

4.3 Comparaison des deux conceptions

Malgré les nombreux recherche dans le thème des codes LDPC et leur implémentation en VLSI, la conception d'un décodeur à haut débit avec faible consommation d'énergie et une petite surface de silicon 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 4.5 montre le débit de quelques conceptions (mesuré ou post-layout implementation) en fonction de l'année pour les deux types de décodeurs, les graduation 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 nécessite du matériel reconfigurable pour soutenir différents rendements et longueur de codes sont de types semi-parallèle comme le montre la figure 4.5. 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. La figure **??** montre une comparaison de deux architecture l'une parallèle et l'autre série.

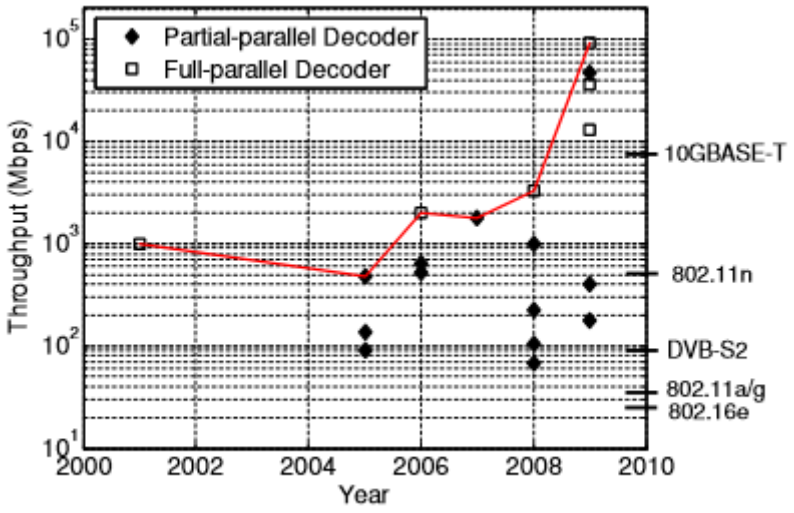

FIGURE 4.4 – Exigence et variation du débit pour quelques décodeur LDPC parallèle et semi-parallèle en fonction de l'année [45]

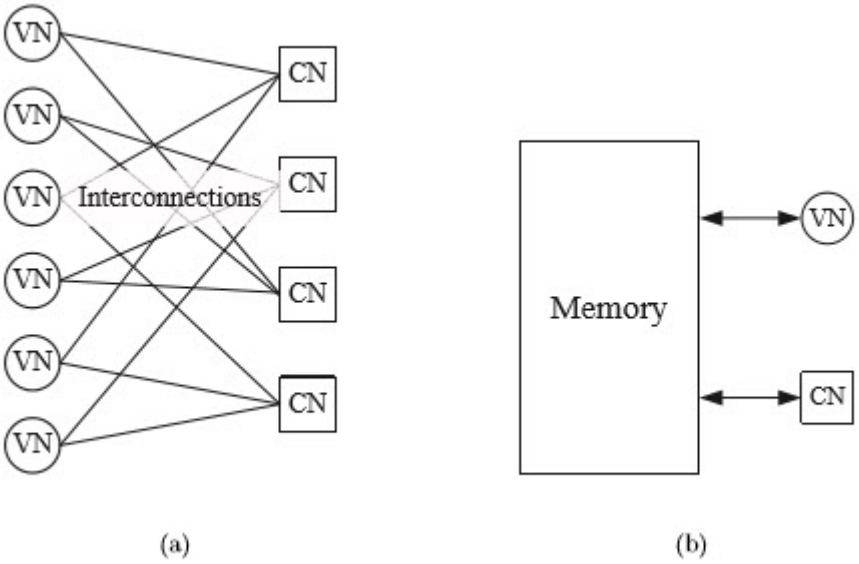

FIGURE 4.5 – Illustration d'un décodeur parallèle a), et d'un décodeur série b).

Chapitre 5

Implémentation d'une architecture LDPC basée sur l'algorithme min-sum

Ce chapitre traite de l'implémentation sur Virtex-5 FPGA ML501 kit, de deux architectures différentes l'une appartient à la catégorie de la conception entièrement parallèle à titre comparatif et l'autre appartient à la catégorie de conception semi-parallèle. L'ensemble des deux l'architectures est expliqué en détail, les bloc tels que CNU, VNU, Déca leurs, convertisseur complément à 2 signé syndrome RAM et l'unité de contrôle ont été implémentés sur la plat-forme Xilinx ISE en utilisant le langage de description VHDL. Après la synthèse de ces codes, les schémas de l'architecture de la description top level jusqu'à la description unitaire sont générés et quelques uns sont illustrés dans ce chapitre. De même les blocs sont vérifiés et testés par l'intermédiaire de l'outil testbench, les résultats y aussi documentés. et finalement le rapport de synthèse tel que généré par Xilinx ISE est présenté et discuté à la fin de ce chapitre.

5.1 Description de la carte FPGA Vertex5 ML501

Nous avons l'intention d'implémenter un décodeur LDPC basé sur l'algorithme Min-Sum sur la carte FPGA Virtex5 ML501 5.1, Pour cela il est important de connaître les caractéristiques de la carte FPGA, qui est le support de notre architecture :

5.2 Architecture parallèle du code régulier 20(3,4)

Dans cette première partie nous allons implémenter le code représenté par la fameuse matrice de R.Gallager (section 2.3) mais avec une taille réduite 20(3,4).
Le type de conception utilisé pour son implémentation est le type entièrement parallèle. La figure **??** montre le schéma générale de l'architecture

5.2.1 Détails de la conception architecturale sur FPGA

L'architecture est composée de 4 blocs principaux :

- Processeur noeuds de donnée

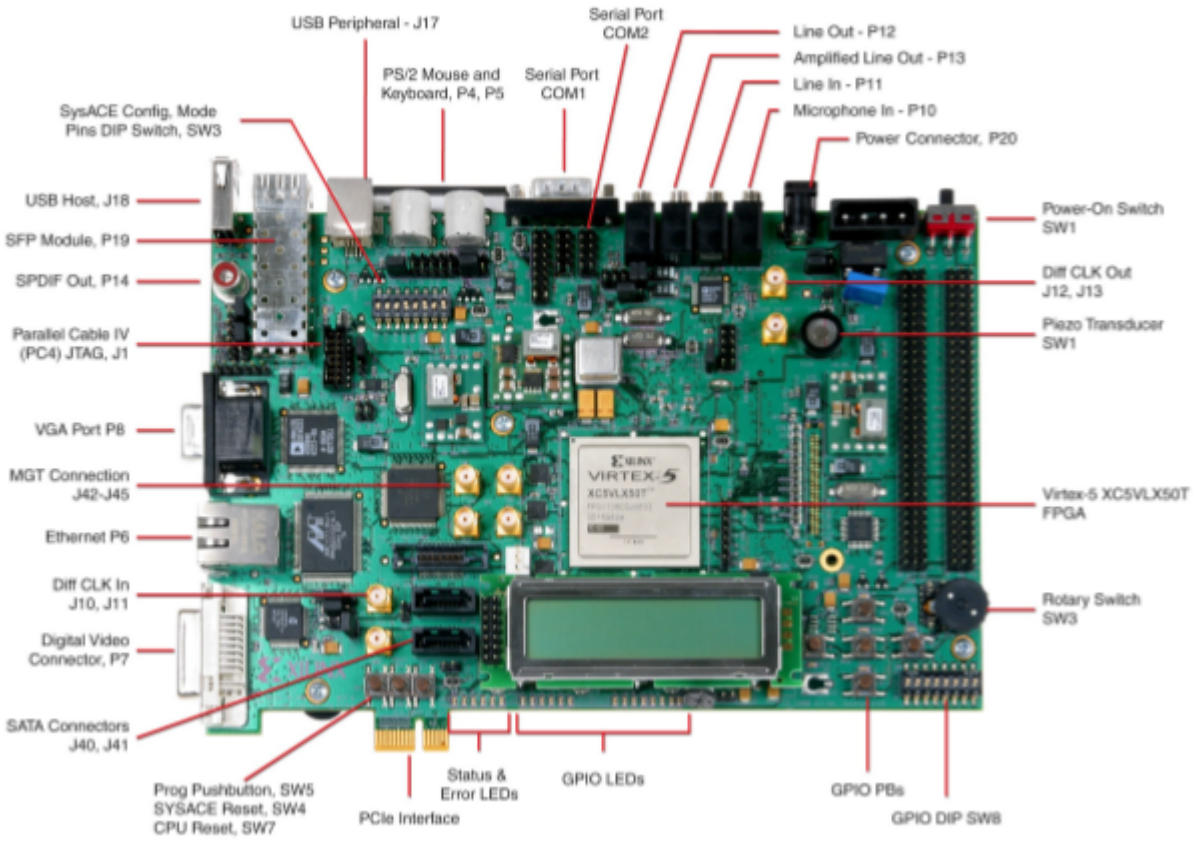

FIGURE 5.1 – Photo de la carte de développement ML501

TABLE 5.1 – Caractéristiques de la carte FPGA Vertex5 ml501

Configuration	• On-board JTAG permettant la configuration via USB
Mémoire	• 16MB Quad SPI Flash
Réseau & communication	• Un pont UART vers USB • Gigabit Ethernet GMII, RGMII et SGMII
Afficheur	• Afficheur LCD 2x16 • 7 LEDs
Horloges	• oscillateur fixe 200Mhz avec une sortie différentielle • 3 horloges asymètrique 27Mhz, 33Mhz et 100Mhz
I/O & control	• 5x bouttons poussoires • 8x switchs • 7 I/O pins disponible pour le LCD

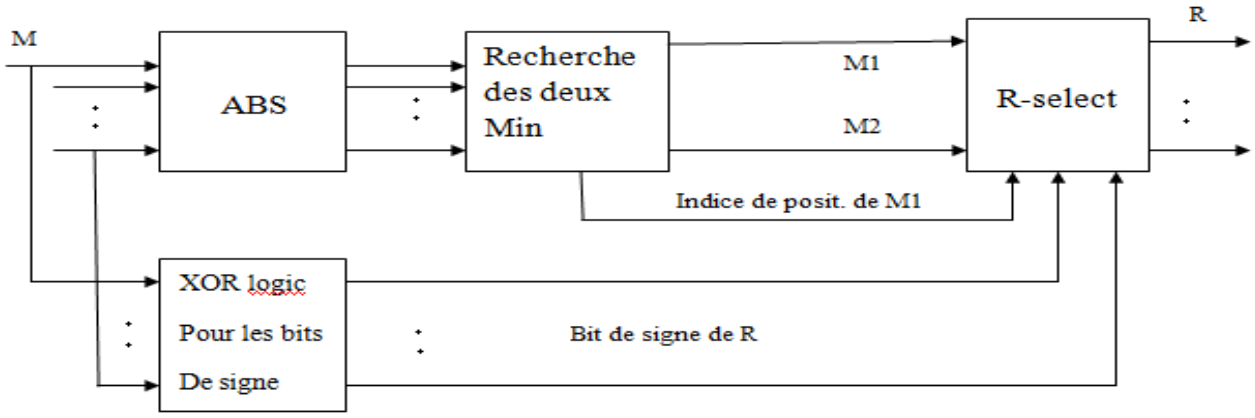

FIGURE 5.2 – Schéma du noeud de contrôle parallèle

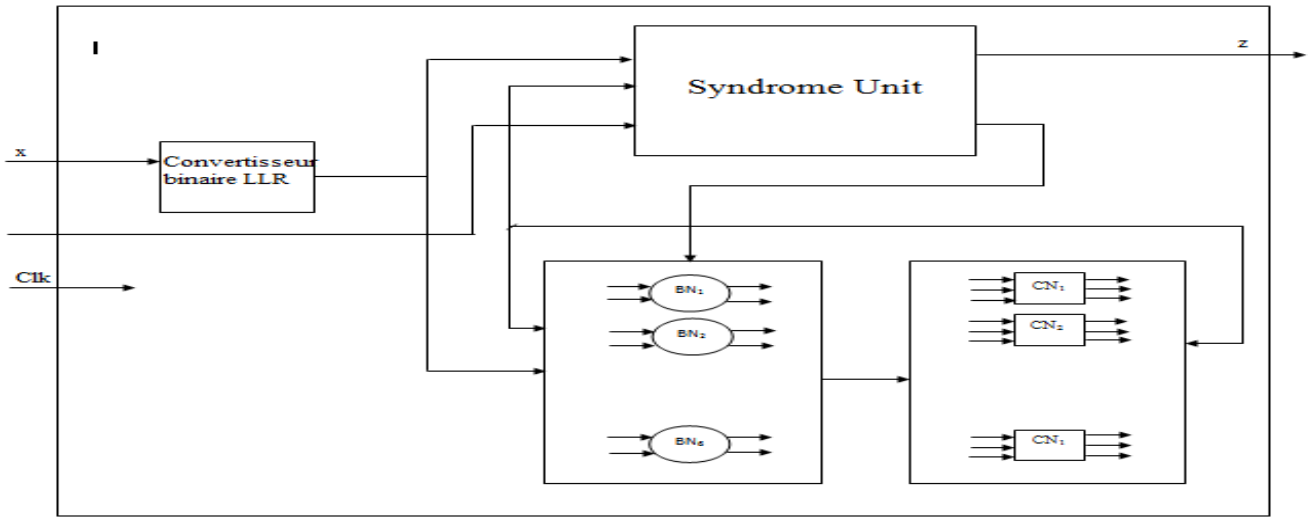

FIGURE 5.3 – Schéma du décodeur LDPC entièrement parallèle

- processeur noeuds de contrôle (voir figure 5.2).
- unité syndrome
- bloc de contrôle

5.2.2 Bloc du décodeur ainsi généré sous xilinx ISE

A Schéma globale de l'architecture

La synthèse du code vhdl qui décrit l'architecture parallèle du code 20(3,4) que nous avons proposé a généré l'architecture de la figure **??**. Nous pouvons directement remarquer à l'oeil 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

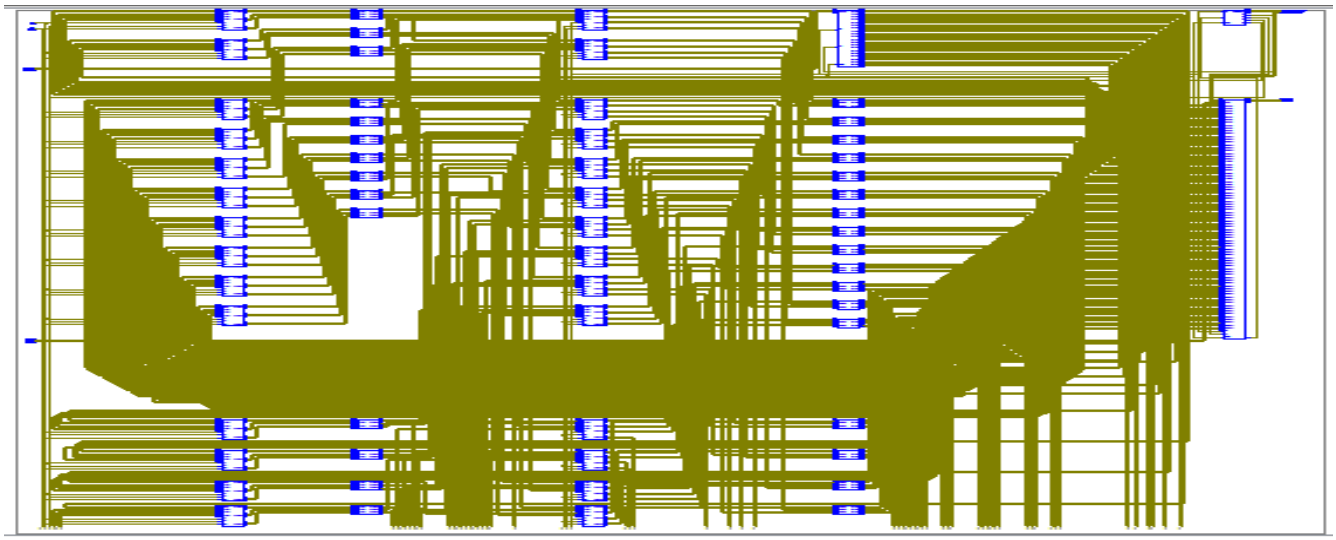

FIGURE 5.4 – Architecture du décodeur parallèle généré par Xilinx ISE

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.

5.3 Architecture Semi-parallèle du code cyclique LDPC

Choix du code LDPC approprié

Le code adopté dans cette section pour l'implémentation d'une architecture semi-parallèle est code cyclique définit par trois paramètres : un entier p et deux entiers k (dégrée de distribution des lignes) et j (degré de distribution des colonnes), tel que $j, k <= p$ et sont traduit par :

$$H = \begin{vmatrix} I & I & I & \cdots & I \\ I & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ I & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ I & \alpha^{(j-1)} & \alpha^{2(j-1)} & \cdots & \alpha^{(j-1)(k-1)} \end{vmatrix} \tag{5.1}$$

Où I est matrice identité $(P * P)$ et α est une permutation traduite par un décalage vers la droite de la matrice identité I, l'exposant de α dans la matrice H est appelé coefficient de décalage, et représente le nombre de décalage cyclique sur la matrice identité I.
On a vu que les codes cycliques donnent des performances assez raisonnable pour les algorithmes de décodage itératifs. Cependant, la construction de codes performants est un défi souvent géré par des méthodes de théorie des graphes.

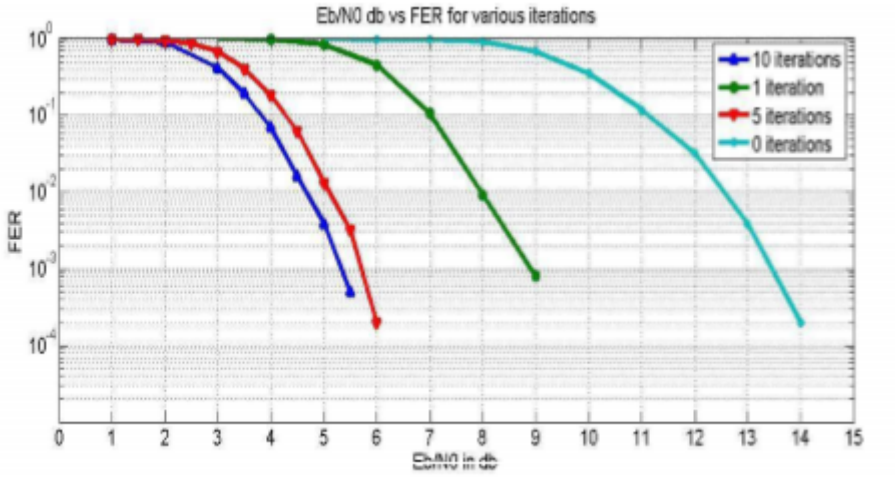

FIGURE 5.5 – FER en fonction du rapport $Eb/N0$ pour 0,1,5 et 10 itérations respectivement

Simulation du code choisi

Le code cyclique 5.2 a été utilisé pour $P = 37$, $N = 259$ et $k = 148$ donnant la matrice H suivante :

$$H = \begin{vmatrix} I & I & I & I & I & I & I \\ I & P^1 & P^2 & P^3 & P^4 & P^5 & P^6 \\ I & P^2 & P^4 & P^6 & P^8 & P^10 & P^12 \\ I & P^3 & P^6 & P^9 & P^12 & P^15 & P^18 \end{vmatrix} \tag{5.2}$$

I est donc la matrice identité de taille $37*37$, P est construit en décalant ces colonnes vers la droite à partir de la gauche ; le rendement du code devient égale à 3/7. Lors de la simulation on envoie une trame contenant 259 $1's$ à travers une modulation BPSK et en lui ajoutant un bruit blanc gaussien (AWGN). les LLR's des valeurs reçu sont envoyés au décodeur pour vérifier l'opération de décodage. FER (Frame Error Rate) est tracé enfin en fonction de $Eb/N0$ pour 0 itération, 1 itération, 5 itérations et 10 itérations respectivement. Figure 5.5

Interprétation des résultats de simulation

Nous constatons que :

- Avec 1 itération nous obtenons un gain de 4.5 dB.
- Avec 5 itérations nous obtenons amélioration en gain proche de 3dB par rapport à 1 itération.
- Avec 10 itérations seulement un gain de 0.3 dB est obtenu.
- Après un certain nombre d'itération le gain en SNR reste a peu prés constant, ce qui fait qu'il faut bien choisir le nombre d'itération adéquat pour établir un compromis entre la latence et le gain en SNR.

5.3.1 Détails de la conception architecturale sur FPGA

Le bloc top level diagramme du décodeur est illustré par la figure 5.6.

Pour des raison de simulations et de vérifications du concept nous avons choisi la matrice réduite du code 5.1 de taille $21 * 28$, tout de même l'architecture reste extensible pour le code 5.2. le message circulant entre les différents blocs de l'architecture est de taille égale à 11 bits avec 6 bits de précision réservés à la partie factionnaire. L'architecture se compose de 11 blocs de base :

- Convertisseur binaire-LLR
- Un bloc de processeur de noeuds de donnée
- Un bloc de processeur de noeuds de contrôle
- un multiplexeur 14 :7
- une file d'attente FIFO
- Un convertisseur complément à 2 vers signe et module
- Un convertisseur signe et module vers complément à 2
- Un bloc de RAM servant à stocker les donnée en sortie des processeurs des noeuds de données
- Un bloc de RAM servant à stocker les donnée en sortie des processeurs des noeuds de contrôle
- deux décaleurs
- un bloc syndrome
- un bloc de contrôle

Quelque blocs parmi ceux cité ci-dessus sont détaillés dans la suite de ce chapitre.

Les valeur des LLRs entrante sont stockées dans un bloc constitué de 7 files d'attentes FIFO parallèle. Tel que montré dans la figure 5.6 FIFO 1 contient la 1ère, 8, 15 et 22ème valeur FIFO 2 contient la 2, 9, 16 et 23ème valeur ainsi de suite. Les valeurs sont stockées de cette façon pour respecter l'ordonnancement que nous avons choisi et qui consiste à traiter les données en trame de 7 bits en parcourant la matrice H de gauche à droite.
Les valeurs des LLRs sont ensuite fournies au blocs des noeuds de données en quatre étapes et seront stockés dans des registres buffers. Lorsque les messages sont traités par les processeur des noeuds de données les sorties seront acheminés vers un registre à décalage permettant d'organiser le stockage des nouvelles données dans le premier bloc de RAMs dans des adresses successives générés par le bloc de contrôle. En parallèle l'unité syndrome reçoit les informations en sortie du bloc des noeuds de données, pour tester si le mot de code est valide ou le nombre d'itération a atteint le maximum.
Après écriture dans le premier bloc de RAMs les données sont lues avec un saut de 4 cases mémoires toujours pour respecter l'ordonnancement choisi, et sont envoyés au bloc des noeuds de contrôle 5.7, ce derniers recherche les deux premiers minimum et le signe correspondant a chacune des valeur de sorties avec l'ordre d'apparition du 2ème minimum. Les données sont enfin envoyés au deuxième bloc de RAMs pour nourrir les processeurs des noeuds de données dans la prochaine itération.

Bloc des noeuds de contrôle

Ce bloc est l'élément responsable à fournir les informations extrinsèques 3.22au noeud de données afin de mettre à jour les nouveaux LLRs. Il reçoit les messages en série lus du

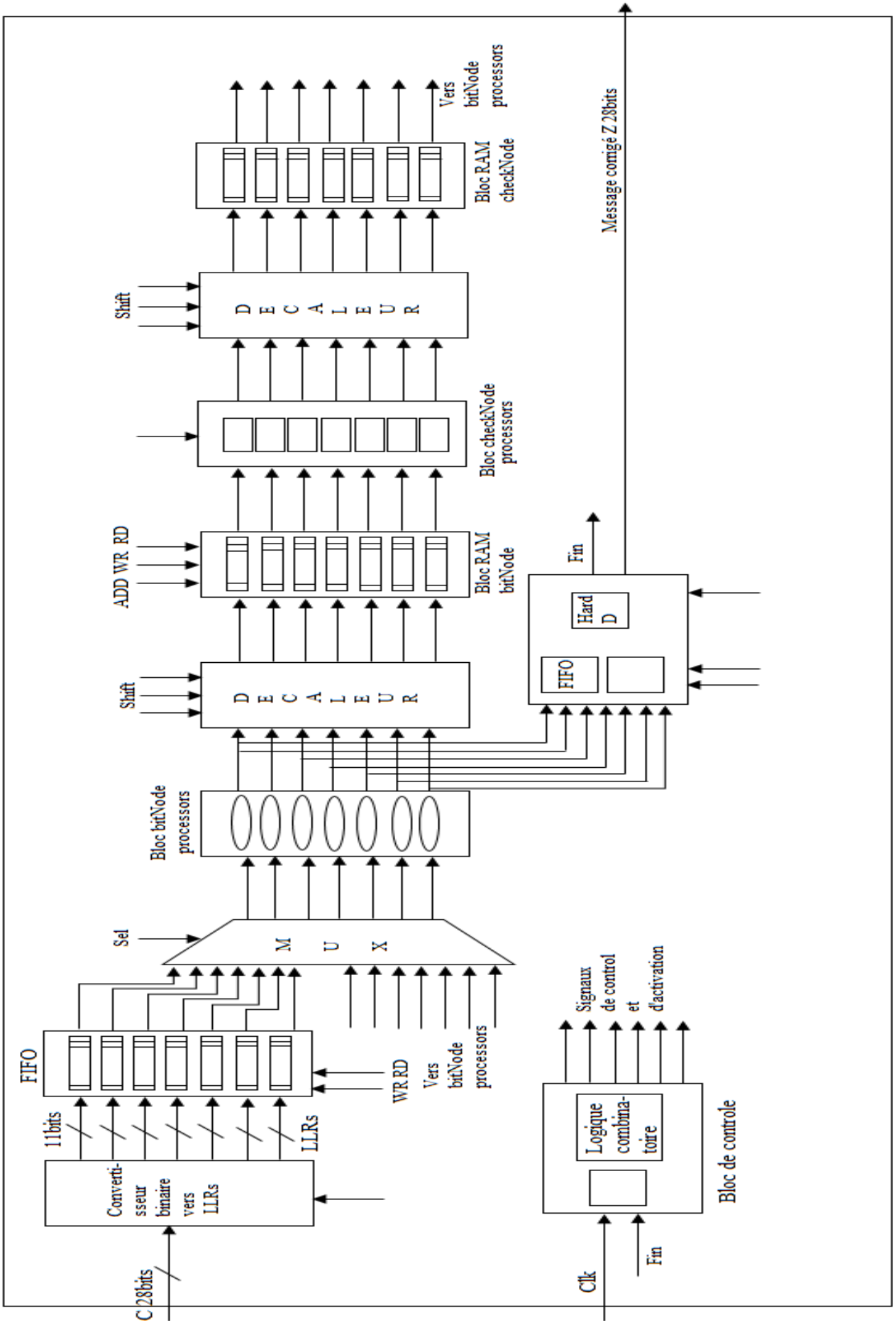

FIGURE 5.6 – Aperçu du bloc architectural du décodeur

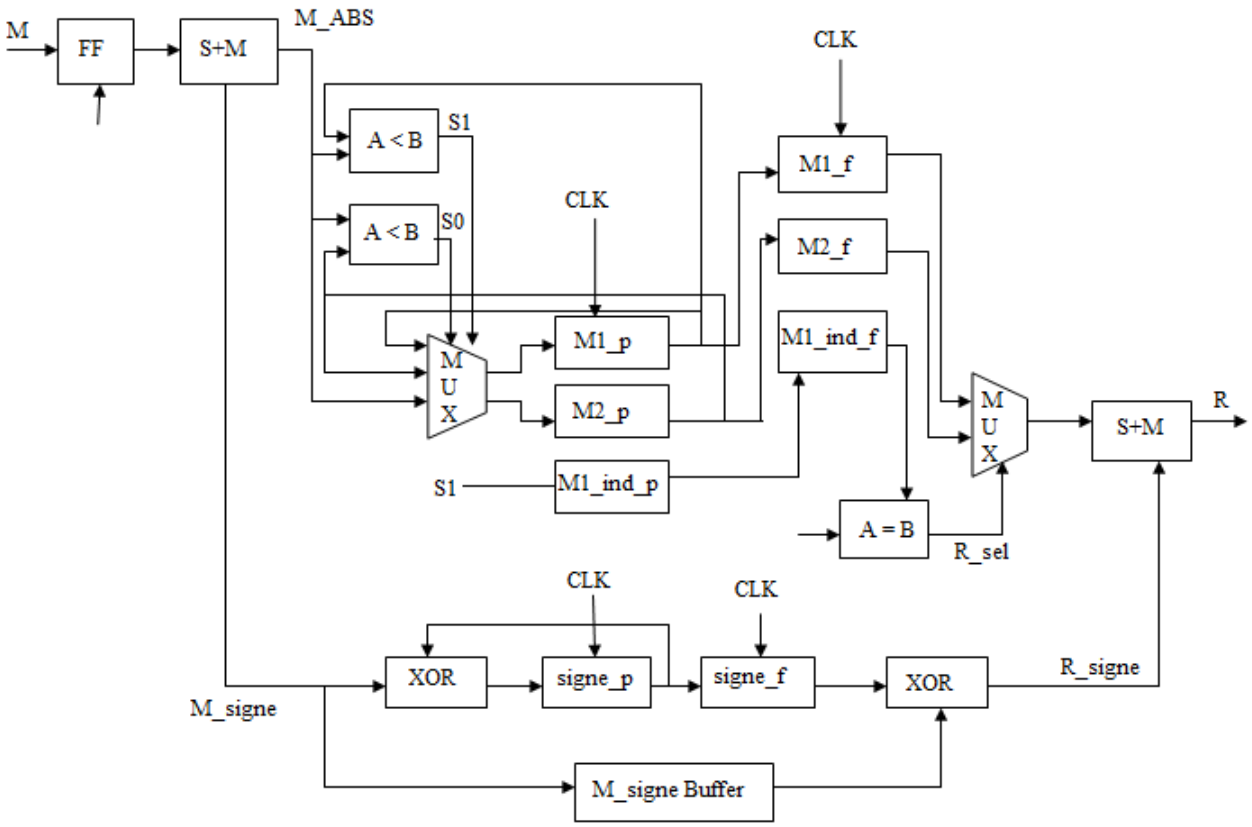

FIGURE 5.7 – Noeud de contrôle

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 série sont traités, une opération de XOR permet de déterminer le signe des valeurs de sortie, ces dernières sont elles aussi acheminées en série

5.3.2 Decaleur

Le décaleur de la figure 5.8se compose de 3 étages ayant 7 multiplexeurs chacun. Les étages peuvent produire des décalages de 1, 2 et 4 respectivement. Un étage n peut produire un décalage de $2n-1$ ou carrément ne pas produire de décalages selon la valeur du bit Shift(n) généré par le bloc de contrôle à partir d'un ensemble de 3 registre (12bits chacun) mis en parallèle pour former un vecteur de 3bits, ces bits sont en fait le vecteur Shift. les 4 premiers (3bits) du premier décaleur sont tous égale à 0 car selon l'ordonnancement choisi pour cette architecture le traitement des processeurs noeuds de données est un *traitement verticale* sachant de la matrice cyclique H est constituée de 3 matrices identités dans le premier bloc de colonne plus une matrice identité qui est contenu dans le deuxième bloc de colonne tandis que les cinq premier bits du deuxième décaleur sont tous des zéros selon le ordonnancement. ce qui donne les vecteur de permutation donnés dans le tableau **??**

TABLE 5.2 – Ordre d'apparition des permutaions en fonction des itérations

Itération	Permutations du premier décaleur			Permutations du deuxième décaleur		
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	1	0	0	0	0	0
6	0	1	0	0	1	1
7	0	0	0	1	0	1
8	0	1	0	1	1	0
9	0	0	1	0	0	0
10	0	0	0	1	0	1
11	0	0	1	1	1	0
12	0	1	1	1	0	0

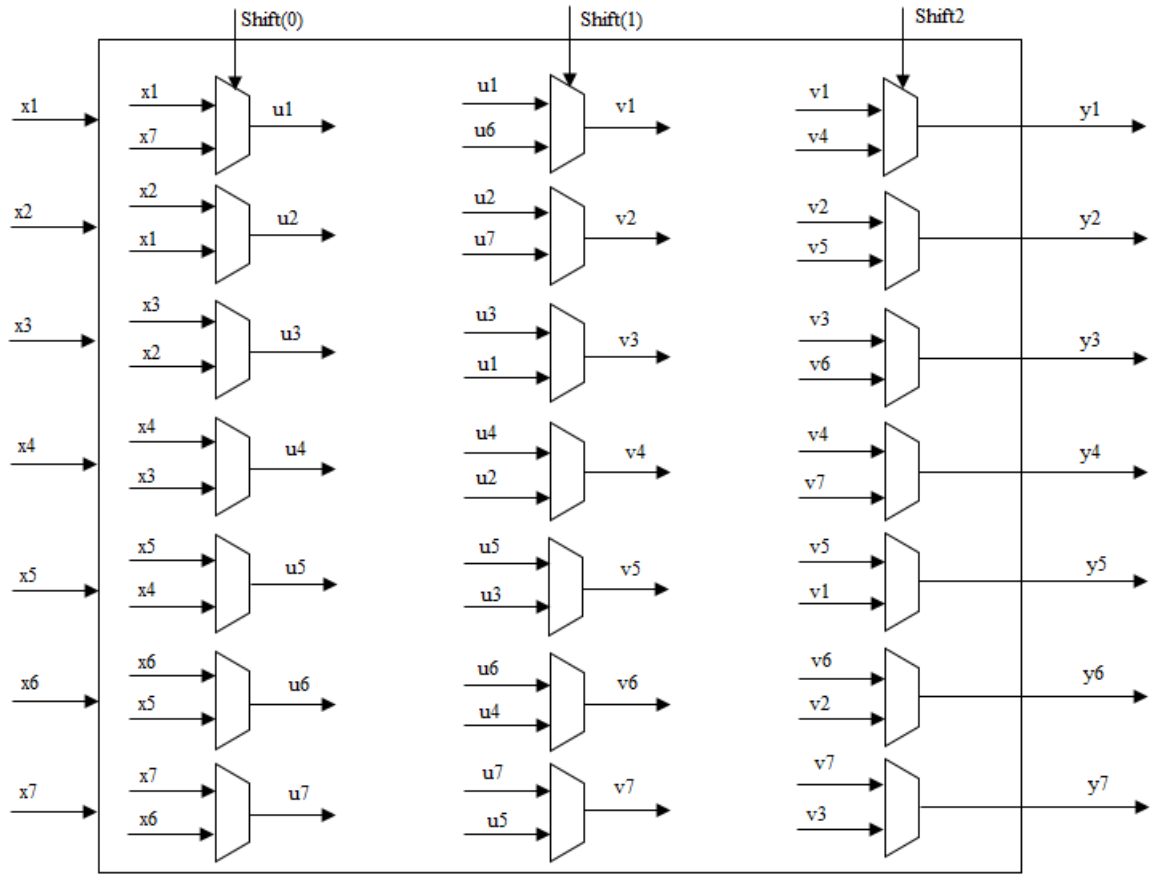

FIGURE 5.8 – Decaleur

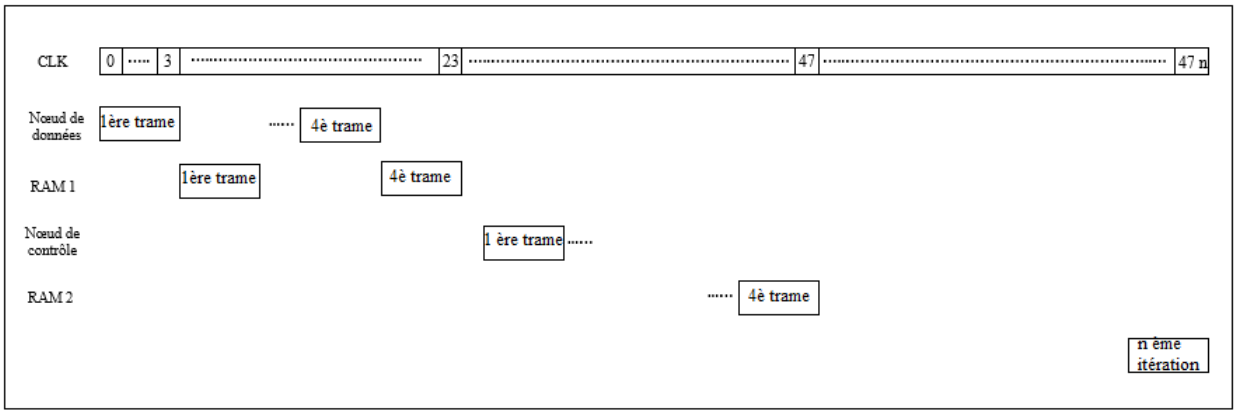

FIGURE 5.9 – Pipeline

5.3.3 Pipeline

La figure 5.9 montre l'ordre d'activation des blocs fonctionnels de l'architecture du décodeur, et le délais de propagation des messages pour accomplir n-itérations.

Calcul du débit :

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

- 4 cycles d'horloges pour convertir les messages binaires en LLR
- 3 cycles d'horloges sont nécessaire pour les processeur des noeuds de donnée pour fournir la première valeur de sortie
- 3 cycles d'horloges sont nécessaire pour écrire dans la RAM les messages en sorties de VNs
- ces trois opérations précédentes devant être multiplié fois 4 pour continuer vers les bloc suivants : avec à chaque fois un retard d'un cycle d'horloge due à la lecture du 2ème Bloc de RAMs.
- le bloc des noeud de données nécessitent 4 cycles d'horloges pour produire la première sortie.
- 4 cycles sont nécessaire pour écrire dans le 2ème bloc de RAMs
- Les deux opérations précédentes doivent elles aussi être multiplié par 3 pour compléter une seul itération avec 3 cycles de retards dues à la lecture du 1er bloc de RAM.
- 4 cycle d'horloges sont nécessaire à la fin du décodage pour produire en sortie le message corrigé en plus des 4 cycles qui sont écoulé au début du processus pour stocker les LLRs " a priori " dans les noeuds de données.

Chaque itération nécessite 48 cycles d'horloge. Dans le cas ou 4 itération sont imposés au décodeur (suffisamment large pour ce code), le nombre de cycle d'horloges à consommer est égale à 192. le calcul théorique du débit est donné ci-dessous : Supposons qu'on moyenne ça prend n nombre d'itérations pour chaque trame. chaque trame est codé sur $28 * 11$ bits i.e 308 bits. la fréquence de l'horloge est de $27MHz$. comme chaque itération nécessite 68 cycle d'horloge i.e $48 * n$, nous obtenons 308 bits en sortie, en d'autres termes,

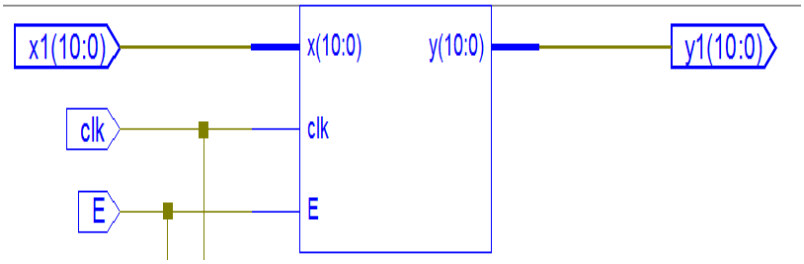

FIGURE 5.10 – Entrée/Sorties du bloc noeud de contrôle

308 bits chaque $68 * 37 * n$ nano secondes ; Cependant, pour une seconde nous obtenons $308/(48 * 37 * n)$, i.e, $173.423/nMbps$. Par exemple si $n = 4$ itérations, le débit donc serait égale à $43.355Mbps$.

Pour le cas du code $(259 * 148)$ représenté par la matrice (5.2) nous obtenons un débit égale à 400 Mbps. Tout de mème le plus grand avantage de cette architecture et la fléxibilité et la faible compléxité.

5.3.4 Bloc du décodeur ainsi généré sous xilinx ISE

Les bloc généré par Xilinx ISE sont donnée dans cette section :

A processeur de noeuds de contrôle

Fig 5.10 montre le schéma du bloc noeud de contrôle ainsi généré par Xilin ISE. Il prend en entrée une seule donnée, une horloge et un signal de sélection et donne en sortie une seule donnée. le temps de son exécution est de 8 cycles d'horloge. La figure 5.11 montre son bloc schématique dans un niveau plus bas.

.

B processeur de noeuds de données

Figure 5.12 montre le schéma du bloc noeud de données ainsi généré par Xilinx ISE. On peut remarquer aisément que le bloc traite les donnée en série en ayant une seule entrée et une seul sortie de données. Figure 5.13 montre son bloc schématique dans un niveau plus bas.

.

C bloc décaleur

Fig 5.14 montre le schéma du bloc décaleur, on voit les 7 entrée (sortie) et le vecteur de contrôle Shift.

D bloc de contrôle

Figure 5.15 montre le schéma du bloc de contrôle généré par Xilinx ISE

.

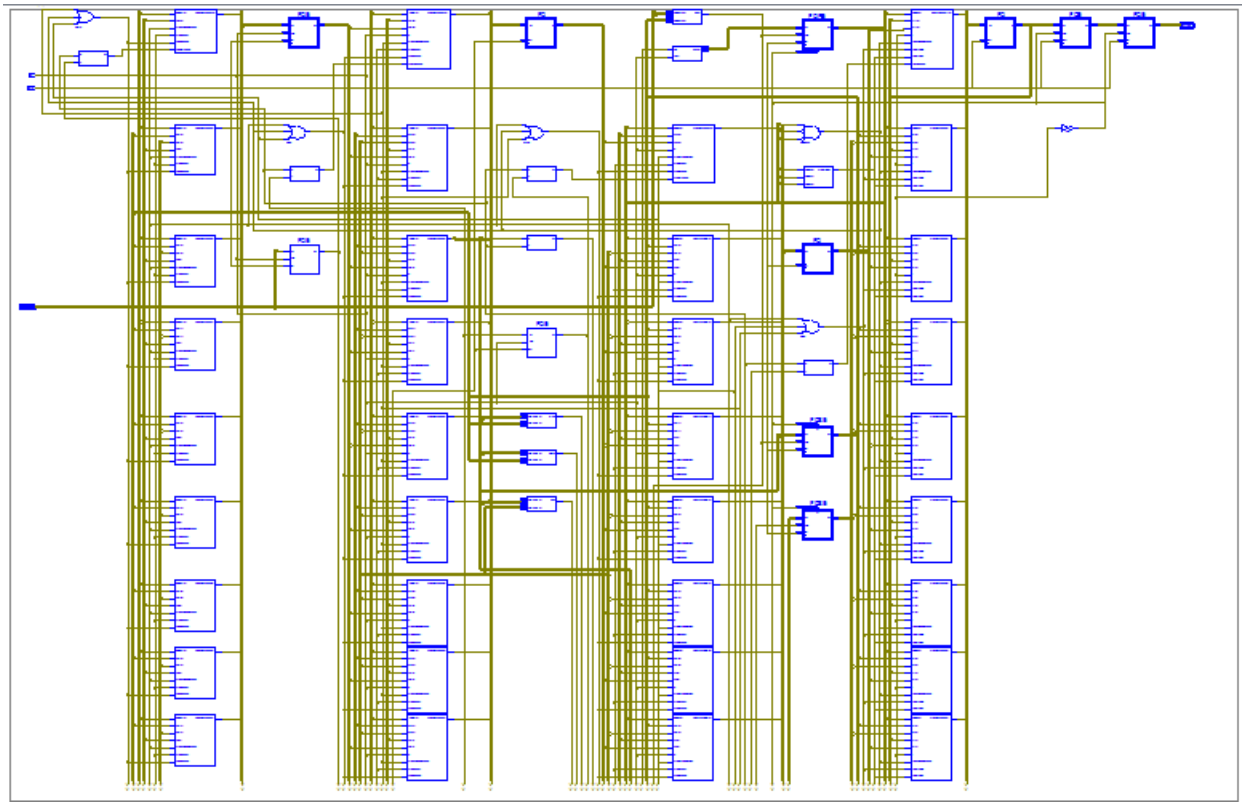

FIGURE 5.11 – Schématique du bloc noeud de contrôle généré par Xilinx ISE

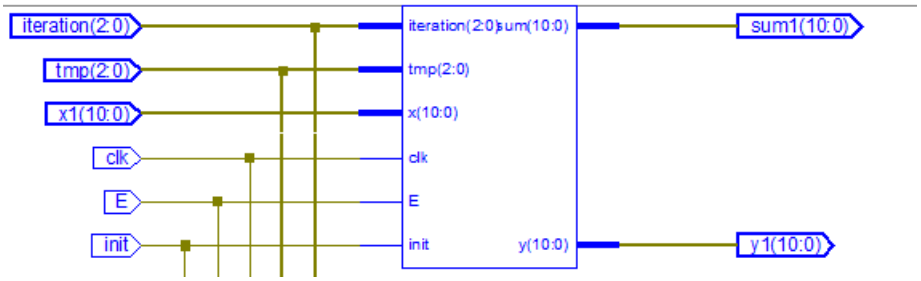

FIGURE 5.12 – Entrée/Sorties du bloc noeud de données

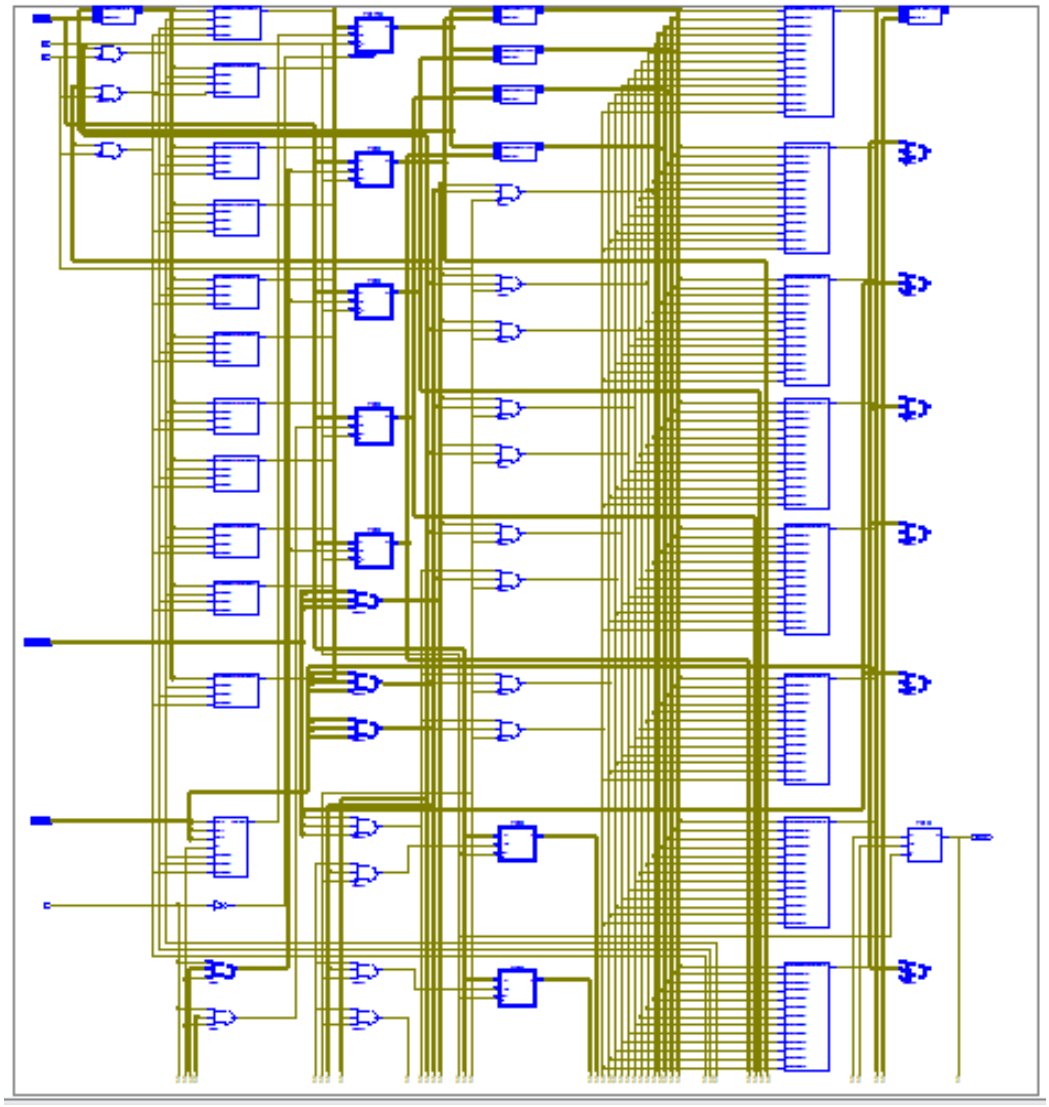

FIGURE 5.13 – Schématique du bloc noeud de données généré par Xilinx ISE

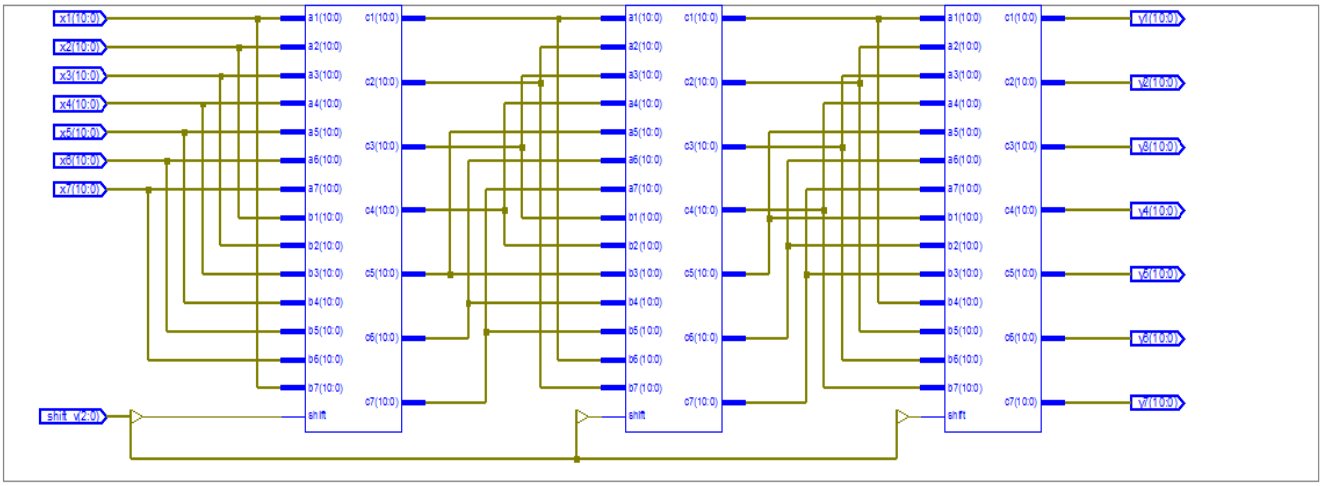

FIGURE 5.14 – Schématique du bloc décaleurs généré par Xilinx ISE

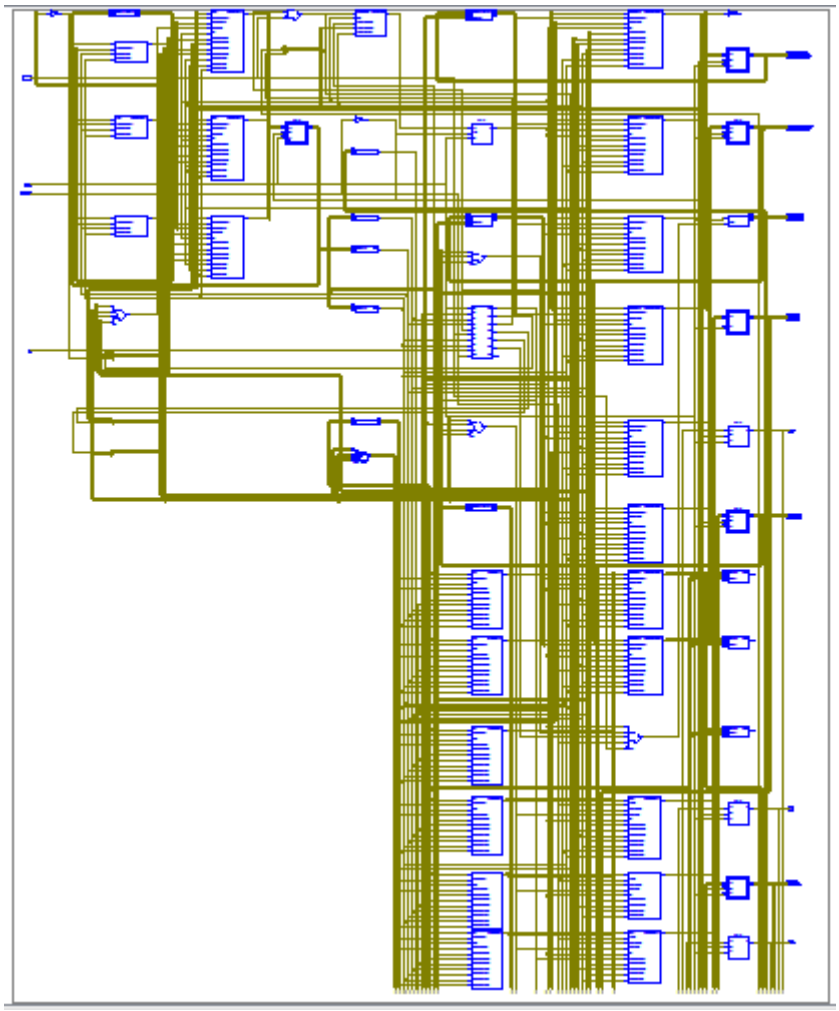

FIGURE 5.15 – Schématique du bloc de contrôle généré par Xilinx ISE

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	674	1920	35%
Number of Slice Flip Flops	623	3840	16%
Number of 4 input LUTs	1268	3840	33%

FIGURE 5.16 – Rapport de synthèse

La figure 5.16 montre le rapport de synthèse de cette architecture

.

Conclusion générale

Ce projet de fin d'études 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.

Dans un deuxième 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 Shanon, ce qui justifie par la suite notre choix pour ce type de codes correcteurs d'erreur.

Pour la bonne compréhension de la partie pratique de notre travaille, nous sommes allé en détail dans la théorie des codes LDPC, où, nous avons étudié les différents algorithmes de décodage, à la fin et pour des raison de simplicité nous avons opté pour l'algorithme sous-optimal Min-Sum dont les performances sont très proches du puissant algorithme de décodage Log-domain.

Avant de passer à la partie consacré à l'implémentation, nous avons jugé nécessaire d'aborder les principaux concepts relatifs au choix de conception, cette dernière est une étape décisive afin d'atteindre les performances et objectifs attendus en l'occurrence, le haut débit, la flexibilité et la faible complexité du décodeur.

Concernant la partie pratique de notre travaille, il s'agissait au départ d'implémenter une architecture d'un décodeur LDPC basé sur l'algorithme Min-sum. Le code que nous avons pris est celui décrit par la construction de R.Gallager définit par une longueur de code moyenne. Les résultats d'implémentations ont montré de très bonnes performances en débit, mais des performances défavorable en flexibilité et en complexité de l'architecture. Pour répondre à ce problème et afin d'améliorer la fléxibilité du décodeur tout en réduisant sa complexité, nous avons adopté le code cyclique du standard de communication CCSDS. Ce code nous a permis de proposer une architecture flexible et de faire un compromis entre le haut débit et la complexité. Les résultats et les détails d'implementations sont documenté dans ce rapport.

A la fin de ce projet nous pouvons conclure, que les codes LDPC donnent une bonne performance de correction d'erreur en se référant des résultats obtenu dans le dernier chapitre.

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

- Faire une étude comparative détaillée entre le codeur/décodeur LDPC et le codeur/-décodeur convolutif en terme de complexité d'implémentation.
- Refaire l'implémentation de l'architecture Min-Sum du code LDPC quasi-cyclique

pour un ordonnancement Vertical.

- 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 task force.* 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, New York* (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 turbocodes.* Lavoisier, 2005.

[13] D. Le RUYET. *Theorie de l'information, Codage Source-Canal.* Ecole Supérieure de Conception et de Production industrielles, France. Jan 2007.

[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 to LTE and WiMAX.* deuxième edition. Wiley, G Bretagne, 2008.

[16] J. B. DORÉ. "Optimisation de codes LDPC et leur architectures de décodage et mise en oeuvre sur FPGA". Thèse de doct. INSA de Rennes, Octobre 2007.

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

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

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

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

[25] M. Kanemasu. “Golay codes”. In : *MIT Undergraduate Journal of Mathematics* (2000).

[26] P. Elias. “Error-free coding”. In : *IEEE Transactions on Information Theory* 4 (sept. 1954), p. 29–37.

[27] A. Viterbi. “Error bounds for convolutional codes and asymptotically optimum decoding algorithm”. In : *IEEE Transactions on Information Theory* (avr. 1967).

[28] A. Viterbi et O. K. Omura. “Principales of Digital Communications and Coding”. In : *McGraw Hill* (1978).

[29] C. Berrou, A. Glavieux et P. Thitimajshima. “Near shannon limit error-correcting coding and decoding”. In : *IEEE International Conference on Communications* 2 (mai 1993).

[30] J. Hagenouer, E. Offer et L. Papke. “Iterative decoding of binary blocs and convolutional codes”. In : *IEEE Transactions on Information Theory* 42 (mar. 1996), p. 429–455.

[31] B. Vasic. “Structured iteratively decodable codes based on Steiner systems and their application in magnetic recording”. In : sous la dir. de TX San Antonio. IEEE Globecom Conf., San Antonio, TX, November 2001, p. 2954–2960.

[32] M. Luby, M. Mitzemmacher, A. Shokrollahi et D. Spielman. “Improved Low Density Parity Check codes using irregular graphs and belief propagation”. In : *IEEE Transactions on Information Theory* (avr. 1998).

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

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

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

[36] Y Mao et A. H. Banihashemi. “A heuristic search for good low-density parity-check codes at short block lengths.” In : sous la dir. d’IEEE International Conference on Communications. 34th European, 2001.

[37] Zhang, Juntan et M. Fossorier. “Shuffled belief propagation decoding.” In : *Signals, Systems and Computers*. Sous la dir. de Conference Record of the Thirty-Sixth Asilomar Conference on. 34th European, 2002.

[38] E. Yeo, B. Nikoli´c et V. Anantharam. “High Throughput Low- Density Parity-Check Decoder Architectures.” In : *EEE Globecom, San Antonio.* (Nov. 2001).

[39] CCSDS Standard. “Low Density Parity Check for use in the Near-Earth and Deep Space Applications”. In : *Signals, Systems and Computers*. Sous la dir. d’Orange Book. 34th European, 2007.

[40] F. Demongel, N. Fau et N. Drabik. “A Generic Architecture of CCDS Low Density Parity Check Decoder for Near-Earth Applications”. In : (2009).

[41] TEMMER Elias. “Etude des codes LDPC et application dans un système MIMO”. Thèse de doct. Ecole Nationale Polytechnique d’Alger, 2009.

[42] Alles. M. “Implementation Aspects of Advanced Channel Decoding”. Thèse de doct. University of Kaiserslautern, 2010.

[43] Blanksby.A, Howlan et C.J. “A 690-mW 1-Gb/s, Rate-1/2 low-density parity-check code decoder”. In : *IEEE J.Solide-State Circ.37(3)* 42 (2002), p. 404–412.

[44] 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.

[45] T. MOHSENIN. “Algorithms and Architectures for Ecient Low Density Parity Check (LDPC) Decoder Hardware”. Thèse de doct. UNIVERSITY OF CALIFORNIA, ELECTRICAL et COMPUTER ENGINEERING, 2010.

[46] B.Kurkoski. “Introduction to low density parity check codes,” in : (2007).

[47] R. Neal. “"Faster encoding for low density parity check codes using sparse matrix methods"”. In : *University of Toronto Canada* (avr. 1999).

[48] D. Hayes. *FPGA implementation of a Flexible LDPC decoder*. 2008. isbn : 3018978.

[49] Kiran Gunnam et Gwan Choi. “A LOW POWER ARCHITECTURE FOR MIN-SUM DECODING OF LDPC CODES”. In : *Dept. of Electrical and Computer Engineering, Texas A&M University, College Station, TX-77840* ().

[50] Kienle Frank. *Architectures for baseband signal processing*. Springer, 2014. isbn : 978-1-4614-8029-7.

[51] N. Wiberg. “Coding and decoding in general graphs”. In : Department of Electrical Engineering, Linkoping, Sweden, 2009.

[52] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet et N. Wehn. “A 477 mW NoC-based digital baseband for MIMO 4G SDR”. In : *In Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2010 IEEE International, February 2010, p. 278–279.

[53] T. Limberg, M. Winter, M. Bimberg et P. Robelly. “A fully programmable 40 gops sdr single chip baseband for lte/wimax terminals”. In : *In Solid-State Circuits Conference, 2008*. Sous la dir. d’ESSCIRC 2008. 34th European, september 2008, p. 466–469.

[54] IEEE 802.16e. “air interface for xed and mobile broadband wireless access systems”. In : *ieee p802.16e/d12 draft* (oct. 2005).