Université Sidi Mohamed Ben Abdellah
Faculté des Sciences et Techniques Fès
Département Génie Electrique

Mémoire de Projet de fin d'étude

Préparé par

MAHMOUDY Imane

Pour l'obtention du diplôme

Master Sciences et Techniques

Electronique, Signaux et Systèmes Automatisés

(E.S.S.A)

Intitulé

Implémentation d'un décodeur LDPC sur une plateforme à base de DSP

Encadré par :

Pr Mouhcine RAZI (LERSI)

Soutenu le Mardi 12 Juin 2018, devant le jury composé de :

Pr Mouhcine RAZI :	**Encadrant**
Pr Ali AHAITOUF :	**Examinateur**
Pr Najiba El Amrani El Idrissi :	**Examinatrice**

ANNEE UNIVERSITAIRE 2017- 2018

Remerciements

Je tiens à exprimer mes sincères remerciements à toutes les personnes ayant contribué de près ou de loin à la réalisation de ce travail de projet de fin d'étude.

Je tiens également à exprimer ma profonde gratitude à monsieur RAZI MOUHCINE professeur de Faculté des Sciences et Techniques pour le temps qu'il a consacré à mon encadrement, pour leur implication, soutien et conseils dans les différentes phases de ce travail je le remercie aussi pour m'avoir montré ce qu'est le monde de la recherche.

De plus, je tiens à remercier l'ensemble des membres du laboratoire de recherche Energies Renouvelables et Systèmes intelligents (ERSI) de la Faculté des Sciences et Techniques- Fès.

Finalement, je remercie mes parents pour m'avoir permis d'arriver jusque-là et pour m'avoir encouragé et supporté durant ce travail.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

ARQ Automatic Repeat Request

ASIC Application Specific Integrated Circuit

ASK Apmplitude Shift Keying

AWGN Additif White Gaussian Noise

BER Bit Error Rate

BP Belief Propagation

BPSK Binary Phase Shift Keying

CCCS Codes Convolutifs Concaténés en Série

CCPC Codes Convolutifs Parallèlement Concaténés

CN Check Node

DSP Digital Signal Processor

DVB-S2 Digital Video Broadcasting Satellite-Second Generation

FEC Forward Error Correction.

FPGA Field Programmable Gate Array

FSK Frequency Shift Keying

LDPC Low Density Check

LLR Log Likelihood Ratio

LLR-BP Log Likelihood Ratio Belief Propagation

LUT Look Up Table

MAC Multiply and Accumulate

MSA Min-Sum Algorithm

PER Packet-Error-Rate

PSK Phase Shift Keying

PWL PieceWise linear

QAM Quadrature Amplitude Modulation

SAOMSA Self Adjustable Offset Min-Sum Algorithm

SNR Signal to Noise Ratio

SoPC system on Programmable Chip

VLIW Very Long Instruction Word

VN Variable Node

WiMAX Worldwide Interoperability for Microwave Access

Introduction générale

Les communications numériques connaissent de nos jours une évolution rapide, poussée par les besoins incessamment grandissants en débit, en qualité de transmission et en miniaturisation des circuits. Cette évolution impose un rythme d'amélioration soutenu pour tous les éléments de la chaîne de communication numérique, dont les codes correcteurs d'erreurs. Pour réduire les erreurs causées par le canal de transmission, il existe deux façons différentes couramment utilisées.

- L'une est la retransmission appelée Automatic Repeat reQuest (ARQ). Son principe général est : après détection d'une erreur, le récepteur demande à l'émetteur de retransmettre une nouvelle fois le message. [1].
- Une autre façon de réduire les erreurs dans un système de communication est l'autocorrection Forward Error Correction (FEC). En utilisant cette technique, le récepteur est capable de détecter et de corriger les erreurs sans demander une retransmission, Cela nécessite des algorithmes beaucoup plus complexes.

En 1948, Shannon a démontré que lorsque le taux de transmission du système est inférieur à la capacité du canal de transmission, les erreurs causées par le bruit du canal peuvent être réduites à un niveau arbitrairement bas par l'utilisation d'un codage et d'un décodage approprié. À partir de ce moment-là, les chercheurs ont commencé à étudier différentes méthodes de construction des codes correcteurs d'erreur.

Le but de la théorie des codes correcteurs d'erreurs est de minimiser le plus possible les erreurs de décodage, en assurant en même temps de très grandes vitesses de transmission et de faibles coûts du codeur et du décodeur.

Les codes Low Density Check LDPC sont des puissants systèmes de codage FEC qui peut atteindre de bonnes performances d'erreur sous des rapports signal-bruit très faibles.

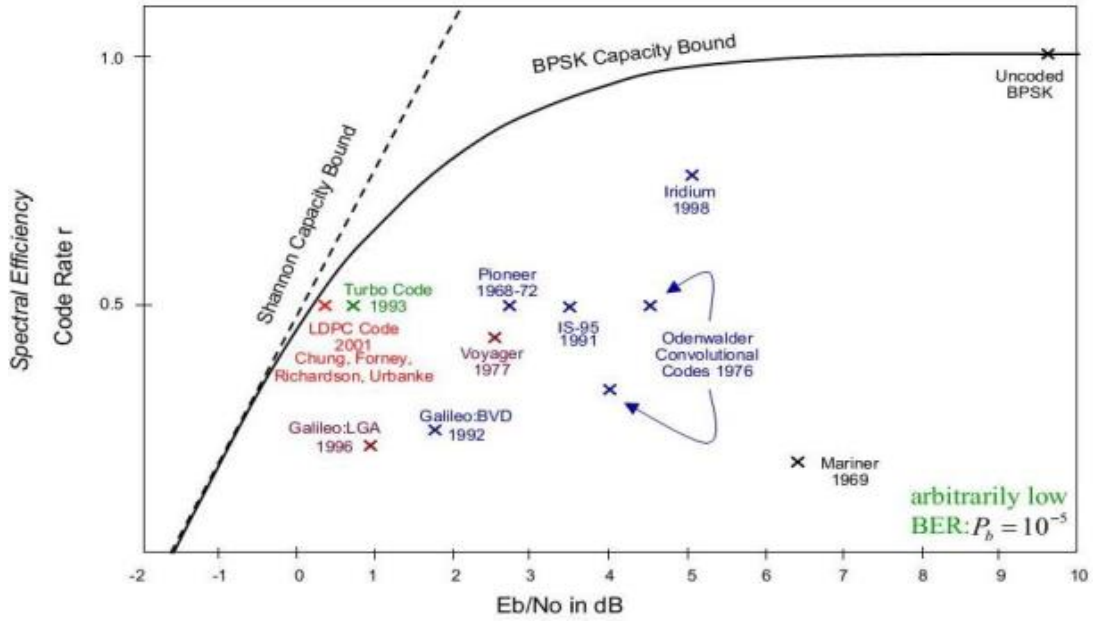

Figure 1 -Comparaison des performances du code FEC

La figure 1 compare les performances des différents codes correcteurs d'erreurs utilisés dans les systèmes de communication. [3] Il montre que les codes LDPC peuvent fonctionner près de la capacité de Shannon liée dans un environnement de rapports signal-bruit inférieur.

En plus de leur bonne performance, les codes LDPC ont une plus faible complexité dans le processus de décodage par rapport à d'autres codes FEC tels que les turbo-codes [3].

Les codes LDPC ont été proposé par Gallager en 1963 [4], Ils sont utilisés dans de nombreuses normes de communication à grande vitesse telles que la diffusion des vidéos numériques par voie satellitaire DVB-S2 (Digital Video Broadcasting Satellite-Second Generation), WiMAX (Worldwide Interoperability for Microwave Access), et les systèmes sans fil 4G.

L'invention des Turbo Codes et la redécouverte des codes LDPC ont donné un nouveau souffle à la recherche dans le domaine des codes correcteurs d'erreurs.

Ce mémoire s'articule sur quatre chapitres.

Le premier chapitre : a pour objectif de présenter l'organisme d'accueil, le laboratoire de recherche ERSI.

Le deuxième chapitre : aborde quelques concepts généraux concernant les codes LDPC, les principes de base du codage et décodage, ensuite nous montrons les algorithmes de décodages des codes LDPC et leurs approximations.

Le troisième chapitre : commence par une étude comparative des performances des algorithmes de décodage LDPC que nous avons mené, et se termine par le choix de l'algorithme le plus proche de l'algorithme LLR-BP et le plus simple au niveau de complexité de calcul et d'implémentions.

Le quatrième chapitre : est consacré à l'implémentation hardware des résultats obtenu dans le chapitre 3, d'abord nous avons donné une vue générale sur les DSPs, après l'architecture de la cible choisis. Ensuite nous avons décrits les digrammes de flux des codes C de la chaine de transmission numérique utilisant le décodage LDPC, et de décodeur utilisé.

Finalement, la conclusion générale de ce travail synthétise les différentes idées présentées dans ce rapport et quelques perspectives sont envisagées pour les travaux futurs.

Cahier des charges

Les techniques de décodage itératif pour les codes modernes dominent actuellement le choix pour la correction des erreurs dans un grand nombre d'applications. Les codes LDPC forment une classe de code en bloc qui se caractérisent par une matrice de contrôle creuse. Ils sont actuellement omniprésents dans le contexte des communications mobiles sans fil.

Les codes LDPC ont ainsi été adoptés dans plusieurs standards de transmission radio (téléphonie mobile 3G/3G+, TNT satellite 1ére et 2éme génération, réseaux métropolitain hertziens IEEE 802.16 WiMAX) et sont en passe de l'être dans d'autres (évolution mobile IEEE 802.11n des réseaux Wi-Fi par exemple). Les codes LDPC présentent de nombreuses fonctionnalités et caractéristiques qui lui permettent d'être adapté à une large variété d'applications. Mais ces caractéristiques demandent plus de puissance de calcul accompagnées d'une complexité algorithmique beaucoup plus élevée que les autres codes.

Ceci rend les mises en œuvre logicielles trop difficiles dès lors qu'un traitement temps-réel est requis pour certaines implémentations ayant des contraintes très sévères en termes de surface, de temps d'exécution ou de consommation d'énergie. Il est donc essentiel d'analyser les possibilités d'optimisation de l'implémentation des codes LDPC.

Vu le progrès que connait l'industrie des systèmes embarqués et particulièrement les SoPC (system on Programmable Chip) du point de vue capacité d'intégration, programmation parallèle et fréquence de fonctionnement, les architectures reconfigurables à base du SoPC constituent aujourd'hui une solution efficace et compétitive pour répondre aux contraintes temps-réel imposées par les applications d'aujourd'hui.

L'objectif du projet est d'implémenter un SoPC du décodeur des codes LDPC sur une plateforme à base de DSP pour les systèmes de communications sans fils.

Chapitre I

Organisme d'accueil

1. Introduction :

Ce chapitre sert à donner une représentation générale du laboratoire de recherche ERSI ainsi les Equipes qui le constitue et les thèmes de recherches**.**

Notre projet est fait partie des projets de l'équipe Microélectronique et Systèmes Embarqués (MSE). Ensuite nous présentons la problématique de recherche.

2. Identification du Laboratoire :

Intitulé du laboratoire : Energies Renouvelables et Systèmes intelligents (ERSI)

Directeur du laboratoire : Pr Ali AHAITOUF.

3. Equipes et thèmes de recherche :

ERSI constitue de trois équipes de recherche :

Equipe 1 : Microélectronique et Systèmes Embarqués (MSE) Responsable de l'équipe Pr Ali AHAITOUF.

Elle Possède quatre enseignants-chercheurs principaux se sont :

- AHAITOUF Ali.
- MANSOURI Anas.
- RAZI Mouhcine.
- JORIO Mohammed.
- MOUGHAMIR Khadija.

L'équipe prévus à traiter les thèmes de recherche suivants :

- Microélectronique, Electronique des composants
- CEM des circuits intégrés
- Systèmes embarqués, Description matériel
- Traitements et compression d'images et de vidéo
- Antennes et propagation
- Photovoltaïque à concentration

Equipe 2 : Système à Energies Renouvelables : Intégration & Gestion Intelligence (SERIGI) Responsable de l'équipe : Pr Abdallah MECHAQRANE

Les membres principaux de cette équipe sont :

- Mechaqrane Abdellah.
- ERRAHIMI Fatima.
- ES-SBAI Najia.

Les Thèmes de recherche prévus par cette équipe sont :

- Energie solaire et énergie éolienne, gisements et valorisation
- Traitement de signaux et modélisation des réseaux intelligents
- Algorithme d'optimisation
- Smart grid et développement des systèmes multi sources.

Equipe 3 : Réseaux, Télécommunications et Ingénierie Logicielle (RTL) Responsable de l'équipe : Pr BENNANI DOSSE Saad

Cette équipe rassemble les enseignants-chercheurs suivants :

- BENNANI DOSSE Saad.
- ELHAJ-BEN-ALI Safae.
- LAKHRISSI Younes.
- KENZI Adil.
- EZZOUHAIRI Abdellatif.

Les Thèmes de recherche :

- Théorie, conception et expérimentation des antennes micro-ondes pour des applications RFID
- Résolution des problèmes d'emplacement médian ordonnés en continu multi-installations.
- Services Web, Ingénierie dirigée par les Modèles, Adaptation
- Conception et mise en place de systèmes intelligents améliorant le concept ABC (Always Best Connected) pour la prochaine génération de véhicules connectés

- Calcul scientifique, Analyse numérique, Optimisation polynomiale et théorie de la localisation.
- Optimisation multi objectif et multicritères.

Chapitre II

Codes Correcteurs d'erreurs

1. Introduction :

Dans un premier temps, nous présentons les deux types des codes : turbo codes et LDPC (codage/décodage).

Après nous présentons les algorithmes de décodage des codes LDPC et le choix de l'algorithme à implémenter.

2. Chaine de transmission

Dans un système de transmission, le traitement numérique du signal peut être appliqué à plusieurs reprises.

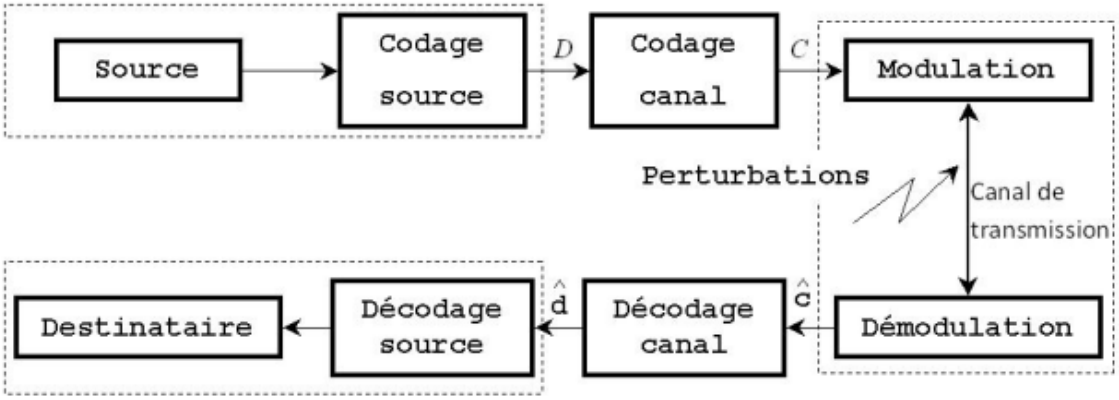

Figure 2.1 – Modélisation d'une chaîne de transmission numérique

2.1 Les caractéristiques de la chaine de transmission :

➢ **La source :** La source du message émet l'information sous la forme de symboles binaires

➢ **Le codage / décodage source :** Le codage source consiste à transformer le message de la source en une séquence d'information « $D(x)$ » de façon à :

1– minimiser la taille du message en éliminant les redondances naturelles de l'information source (algorithme de compression).

2– retrouver le message original à partir de la séquence de substitution « $D(x)$ » (algorithme réversible).

Les symboles émis par la source sont convertis à partir d'un alphabet de symboles (ordinairement des bits) afin que ceux-ci puissent être récupérés au cours de la réception sans modification à partir des données binaires (codage sans perte) ou alors avec une distorsion (codage avec perte). Le codeur source réduit la redondance contenue dans le message et minimise ainsi la quantité d'information utile à sa représentation.

Le décodage source réalise l'opération duale, le message d'information est décompressé afin de retrouver son équivalent à partir de la séquence de substitution « $D(x)$ » avant la transmission.

Il est à noter que les limites théoriques du codage source sont fixées par le premier théorème de Shannon. [5,6]

➢ **Le codage/décodage canal :** Le codage canal a pour rôle de protéger l'information émise contre les perturbations du canal de transmission susceptible de modifier son contenu.

Il s'agit donc de rajouter de la redondance de manière à détecter et éventuellement corriger les erreurs lors de la réception si la stratégie adoptée le permet. L'information $D(x)$ issue du codage source est transformée en séquence codée $C(x)$. Comme le décrit le théorème fondamental du codage canal, pour se rapprocher de la capacité du canal de transmission, il est nécessaire de coder l'information avant de la transmettre.

Au niveau du récepteur, le décodage canal consiste dans un premier temps à détecter la présence d'erreurs dans l'information et puis dans un deuxième temps de les corriger.

➢ **Le modulation/démodulation :** La modulation agit sur les paramètres d'un signal porteur afin de transmettre les données codées.

Dans le cas de la modulation numérique, le message codé est transformé à partir d'un alphabet dont l'entrée correspond à une partie du signal à transmettre. Le signal porteur est une sinusoïde dont on peut faire varier l'amplitude, la fréquence ou la phase indépendamment (ASK, FSK, PSK) ou simultanément amplitude et fréquence QAM, en fonction de l'information à émettre.

Le démodulateur joue le rôle dual du modulateur et transforme donc le signal reçu en un train binaire.

➢ **Le canal de transmission :** Il représente la liaison entre l'émetteur et le récepteur et peut être de différentes natures selon le type de grandeur qu'il permet de véhiculer. Le canal de transmission est caractérisé par sa capacité et sa bande passante. Il existe plusieurs modèles théoriques du canal de transmission en fonction des types d'erreurs les plus fréquents, nous nous limiterons au canal à Bruit Blanc Additif Gaussien (BBAG).

Le canal BBAG : il s'agit d'un canal à entrée binaire et sortie analogique. La sortie se représente par une variable aléatoire continue Y :

$$Y = \mathrm{x} + \mathrm{b} \tag{2.1}$$

Où x est le symbole binaire émis et b est une variable aléatoire gaussienne centrée de variance $\mathrm{б}^2$ correspondant au bruit du canal.

La variance en fonction du rapport signal à bruit est :

$$\mathrm{б}^2 = \frac{1}{2}\left(\frac{Eb}{N0}\right)^{-1} \tag{2.2}$$

Où Eb est l'énergie moyenne utilisée pour transmettre un symbole binaire et $N0$ la densité spectrale de puissance mono latérale du bruit additif.

3. Les codes correcteurs d'erreur :

Les codes correcteurs d'erreurs ont été utilisés pour la détection et la correction des erreurs induites par le canal la transmission. Ils sont regroupés suivant leurs caractéristiques et leurs propriétés.

La première grande caractéristique concerne leur propriété de linéarité. Une autre propriété concerne le caractère systématique d'un code.

Définition : Un code est dit systématique si le message à encoder d^K est contenu dans le mot encodé C^N .

Pour des codes correcteurs systématiques, le mot de code est donc divisé en K bits systématiques et M = N - K bits de redondance. La flexibilité des codes correcteurs est également un facteur de distinction. Les codes en blocs sont des codes de longueurs fixes. De ce fait, un code C se caractérise par un couple (N, K) prédéfini.

A l'inverse, les codes convolutifs sont des codes basés sur des longueurs non établies et donc potentiellement infinies. Dans le cadre des codes correcteurs d'erreurs avancés, les codes LDPC font partie de la famille des codes en blocs, tandis que les turbos codes convolutifs réutilisent le principe des codes convolutifs.

3.1 Familles des codes correcteurs d'erreurs avancés :

3.1.1 Les turbos codes :

Les turbo codes sont une classe de codes convolutifs concaténés. Cette concaténation est réalisée à l'aide d'un bloc mémoire, appelé entrelaceur, qui modifie l'ordre des données d'entrée d'un des codes composants. Selon la manière dont les codes convolutifs composants sont concaténés, deux structures principales sont possibles : Codes Convolutifs Parallèlement Concaténés (CCPC) et Codes Convolutifs Concaténés en Série (CCCS). Dans la figure 2.2, les deux structures sont représentées.

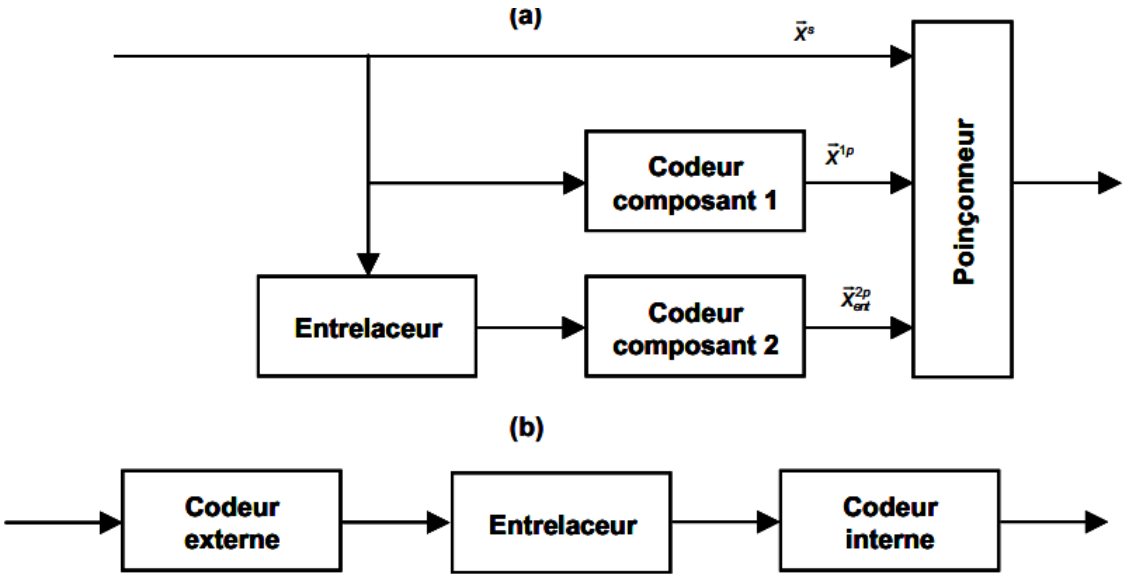

Figure 2.2 – Schéma d'un turbo-codeur : (a) à concaténation parallèle, (b) à concaténation série

En observant la figure 2.2, nous pouvons conclure que dans les codeurs CCPC, les deux codes composants (C1et C2) travaillent sur le même ensemble de données d'entrée. En revanche, dans une concaténation en série, la sortie du codeur externe (CE) devient l'entrée du codeur interne (CI).

L'intérêt de l'utilisation d'entrelaceur dans des codes concaténés est d'éclater les regroupements d'erreurs sortant d'un décodeur composant afin de présenter des erreurs isolées au décodeur suivant

Le bloc de poinçonnage est ajouté au turbo-codeur Pour diminuer la perte de bande passante due à l'utilisation des rendements faibles, Le rendement de codage défini comme

le rapport du nombre de bits d'information sur le nombre de bits transmis, Ce bloc supprime certains bits qui ne sont alors pas transmis

Cette famille de codes offre de hautes performances permettant de s'approcher très près de la limite théorique de Shannon. Ils marquent une révolution dans la communauté de la théorie de l'information. Depuis leur introduction, de très nombreux travaux de recherche relatifs aux turbos codes, et plus généralement aux codes concaténés, ont été effectués afin de comprendre et d'analyser leur comportement et d'améliorer leurs performances.

3.1.2 Les codes LDPC :

Les codes LDPC font partie de la classe des codes en blocs linéaires et qui s'approchent de la limite de Shannon. Il a été démontré que les codes LDPC longs avec un décodage itératif basé sur la propagation de croyance atteignent une performance d'erreur à une fraction de décibel de la limite du Shannon [7], [8], [9], [10].

Les codes LDPC sont en grande compétition avec les codes turbo dans les systèmes de communications numériques qui demandent une fiabilité élevée. Aussi, les codes LDPC ont quelques avantages par rapport aux codes turbo :

- Ils ne nécessitent pas d'entrelaceur pour réaliser une bonne performance d'erreur,
- Ces codes ont une complexité réduite par itération et offrent un degré important de parallélisme pour l'implémentation des décodeurs.
- Ils permettent également une flexibilité en termes de choix du rendement et de longueur du code.

Ces avantages ont motivé la communauté scientifique à poursuivre les travaux de recherche dans le domaine des codes LDPC.

4. Aperçu sur les codes LDPC :

4.1 Codes linéaires en bloc :

Dans un code C (N, K) en bloc les messages transmis sont groupés en bloc (mots code) de longueur N. Les symboles de l'information (symboles émis par la source) et les mots code sont des alphabets q *-aires*.

Si $q = 2$ on parle des codes en blocs binaires alors que si $q > 2$ on parle des codes en blocs non binaires.

Dans le cas d'un code en bloc binaire, un mot code composé de N symboles est considéré comme un vecteur dont les composantes appartiennent à l'alphabet binaire du corps fini $\{0, 1\}$.

Les mots de longueur N sont des éléments de $\{0,1\}^N$ qui sont écrits sous la forme de vecteurs lignes. Chacun des 2^K messages différents possibles correspond de manière unique à un des mots code.

Le code en bloc C (N, K) sera linéaire si les 2^K mots code possibles forment un sous espace vectoriel de dimension K de l'espace vectoriel $GF(2)^N$ ou $GF(2)$ est le corps de Galois dont les composantes sont des éléments binaires. Par conséquent, la somme de deux mots code et le produit d'un élément de GF (2) par un mot code sont aussi des mots code. On appelle distance de Hamming le nombre de symboles différents entre deux mots code. La distance minimale d'un code C (N, K), notée d_{min} est alors la distance de Hamming minimale entre deux mots code quelconques. Le nombre N-K, appelé le nombre de symboles de redondance, est le nombre de symboles de longueur K ajoutés à l'information pour générer le mot code.

Le code linéaire en bloc sera alors défini par trois paramètres,

- La longueur du mot N,
- La longueur de l'information K.
- La distance minimale d_{min},

Le code ainsi défini est capable de corriger $e = Ent\left(\frac{d_{min}-1}{2}\right)$ erreurs où Ent est la partie entière. Le rapport $R = \frac{K}{N}$ est appelé taux ou rendement du code. Si le nombre de bits de la redondance appliquée à une information croit, le rendement diminue. Plus il est proche de zéro, meilleure est la protection contre les erreurs de transmission. Cependant, le temps de transmission est important car il faut transmettre N bits dans une durée NT_b alors que nous n'avons besoin à la fin, de décodage, que des K bits d'informations qui nécessitent seulement une durée de KT_b (T_b : temps que va durer un bit, son unité est la seconde).

4.1.1 Matrice génératrice G :

Puisque le code C (N, K) est un sous espace vectoriel de l'espace vectoriel $GF(2)^N$, il existe K mots $g_0, g_1, \ldots, g_{k-1}$ linéairement indépendants considérés comme la base du sous espace vectoriel. Chaque mot $X \in C$ est une combinaison linéaire des mots

$g_0, g_1, \dots, g_{k-1}$ Ainsi le mot code *X* est défini par :

$$\boldsymbol{X = u_0 \bullet g_0 \oplus u_1 \bullet g_1 \oplus \dots \oplus u_{k-1} \bullet g_{k-1}} \quad \textbf{(2.3)}$$

Où le symbole '•' représente l'opération de multiplication d'un élément de *GF* (2) par un mot défini dans le sous espace vectoriel *C* (*N*, *K*), le symbole '⊕' désigne l'opération d'addition modulo 2 définie dans le sous espace vectoriel *C* (*N*, *K*) et les éléments $(u_0 \dots u_{k-1})$ représentent le vecteur d'information de longueur *K*. Les vecteurs $(g_0 \dots g_{k-1})$ de la base du sous espace vectoriel *C* (*N*, *K*) sont disposés comme des lignes d'une matrice, appelée la matrice génératrice *G* du code, définie par :

$$G = \begin{bmatrix} g0 \\ g1 \\ \vdots \\ gk-1 \end{bmatrix} = \begin{bmatrix} g0,0 & g0,1 & \dots & g0,N-1 \\ g1,0 & g1,1 & \dots & g1,N-1 \\ \vdots & \vdots & \ddots & \vdots \\ gk-1,0 & gk-1,1 & \dots & gk-1,N-1 \end{bmatrix} \quad (2.4)$$

Chaque mot code *X* peut être généré par la multiplication du vecteur $U = (u_0, u_1, \dots u_{k-1})$ par la matrice génératrice *G*. Le mot code sera alors défini par :

$$X = U \circ G = (u_0, u_1, \dots u_{k-1}) \circ \begin{bmatrix} g0 \\ g1 \\ \vdots \\ gk-1 \end{bmatrix} = u_0 \bullet g_0 \oplus u_1 \bullet g_1 \oplus \dots \oplus u_{k-1} \bullet g_{k-1} \quad (2.5)$$

Où le symbole '∘' désigne le produit interne entre deux matrices définies dans le sous espace vectoriel *C* (*N*, *K*).

4.1.2 Forme systématique d'un code linéaire en bloc

La structure du mot code en forme systématique est indiquée sur la figure (2.3).

N-K bits de redondance	K bits d'information

Figure 2.3 – Mot code en forme systématique

Les *N* - *K* bits de redondance et les *K* bits d'informations sont placés respectivement au début et à la fin du mot code. Dans ce cas la Matrice génératrice *G* du code linéaire en bloc *C* (*N*, *K*) est définie comme suit :

$$G=\begin{bmatrix} g0 \\ g1 \\ \vdots \\ gk-1 \end{bmatrix}=\begin{bmatrix} p0,0 & p0,1 & \dots & p0,N-k-1 & 1 & 0 & 0 & \dots & 0 \\ p1,0 & p1,1 & \dots & p1,N-k-1 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ pk-1,0 & pk-1,1 & \dots & pk-1,N-k-1 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad (2.6)$$

Sous matrice $P_{K\times(N-K)}$ sous matrice I_K

La matrice génératrice *G* peut s'écrire aussi sous la forme suivante :

$$G = [P_{K\times(N-K)}\ I_K] \quad (2.7)$$

Où $P_{K\times(N-K)}$ est une sous matrice de parité de dimension $K \times (N-K)$ et I_K est une sous matrice identité de dimension $K \times K$ [11, 12].

Dans le cas d'un code linéaire en bloc les relations qui lient les bits d'information *U*, les bits du mot code $X = (x_0, \dots, x_{N-1})$ et ceux de contrôle de parité sont comme suit :

$$x_{N-K+m} = u_m \qquad m = 0,1, \dots, K-1 \quad (2.8)$$

$$x_n = u_0 \bullet p_{0,n} \oplus u_1 \bullet p_{1,n} \oplus \dots \oplus u_{K-1} \bullet p_{K-1,n} \quad 0 \le n \le \mathrm{N-K-1} \quad (2.9)$$

Ainsi le mot code peut s'écrire sous sa forme systématique comme indiqué dans l'équation (2.10) :

$$X = (x_0, x_1 \dots, x_{N-1}) = (x_0, x_1 \dots, x_{N-K-1}, u_0, u_1 \dots, u_{K-1}) \quad (2.10)$$

4.1.3 Matrice de contrôle de parité H

La forme systématique de la matrice de contrôle de parité *H*, du code binaire *C* (*N*, *K*) généré à partir de la matrice *G*, est définie par :

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-K-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & p_{0,0} & p_{1,0} & \dots & p_{K-1,0} \\ 0 & 1 & 0 & \dots & 0 & p_{1,0} & p_{1,1} & \dots & p_{K-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & p_{0,N-K-1} & p_{1,N-K-1} & \dots & p_{K-1,N-K-1} \end{bmatrix} \quad (2.11)$$

Sous matrice I_{N-K} sous matrice $P^T_{(N-K)\times K}$

Où $P^T_{(N-K)\times K}$ est la sous matrice transposée de $P_{K\times(N-K)}$sa dimension est (*N* - *K*) × *K* et I_{N-K}est la sous matrice identité [11, 12].

Le produit interne entre chacun des lignes g_m et h_n des matrices *G* et *H* est égale à zéro. Par conséquent nous aurons

$$g_m \circ h_n = p_{m,n} \oplus p_{m,n} = 0 \quad (2.12)$$

Sous une forme matricielle l'équation (2.12) devient

$$G \circ H^T = 0 \tag{2.13}$$

4.1.4 Equations de contrôle de parité :

Les équations de contrôle de parité ou contraintes de parité peuvent être décrites soit à partir de la matrice génératrice G ou à partir de la matrice de contrôle H.

Nous considérons le mot X de l'équation (2.10), qui utilisé dans l'équation (2.14) nous permet de déduire les N – K équations de contrôle de parité de l'équation (2.15). Tous les mots code X ∈ C devraient vérifier l'équation (2.14), sinon le mot X n'est pas un mot code valide.

Les équations de contrôle de parité restent un moyen sûr pour détecter la présence des erreurs dans un message transmis. Elles sont définies par

$$X \circ H^T = U \circ G \circ H^T = 0 \tag{2.14}$$

Pour chaque ligne n de la matrice de contrôle H on peut écrire

$$x_n = u_0 \bullet p_{0,0} \oplus u_1 \bullet p_{1,0} \oplus \ldots \oplus u_{K-1} \bullet p_{n,K-1} \quad 0 \leq n \leq \mathrm{N} - \mathrm{K} - 1 \tag{2.15}$$

4.1.5 Détection des erreurs par syndrome :

Dans le cas d'un code binaire toutes les composantes des mots code $X = (x_0, x_1 \ldots, x_{N-1})$ appartiennent au champ de Galois GF (2). Ces mots code sont censés être transmis via un canal de transmission vulnérable aux bruits, par conséquent les messages reçus peuvent être entachés par un certain nombre d'erreurs de transmission.

Les messages reçus peuvent alors être différents de ceux transmis. $Y = (y_0, y_1 \ldots, y_{N-1})$ et $E = (e_0, e_1 \ldots, e_{N-1})$ dénotent respectivement les messages reçus et les patterns d'erreur, les composantes de ces deux vecteurs appartiennent au champ de Galois GF(2).

L'équation (2.16) met en évidence la relation qui lie les vecteurs X, Y et E dans le cas d'un modèle de canal AWGN.

$$E = Y \oplus X \tag{2.16}$$

Une erreur est détectée sur la position ou une composante du vecteur E est non nul. Le mécanisme de détection des erreurs par la technique du syndrome peut être résumé par l'expression suivante :

$$S_r = Y \circ H^T = (X \oplus E) \circ H^T = E \circ H^T = (S_0, S_1 \dots, S_{N-K-1}) \quad (2.17)$$

Le vecteur S_r s'appelle le vecteur du syndrome, il est le résultat du produit interne entre le message reçu *Y* et le transposé de la matrice *H*. A partir de l'équation (2.17) on remarque que le syndrome ne dépend que du vecteur du pattern d'erreurs *E*.

Si le vecteur du syndrome est nul le message reçu est un mot code, sinon le message reçu est entaché par une ou plusieurs erreurs de transmission.

4.1.6 Description des Codes LDPC :

Le code LDPC C (N, K) est un code linéaire en bloc caractérisé par une matrice de contrôle de parité creuse. Dans le cas des codes binaires, le nombre de 1 dans la matrice est faible par rapport au nombre de 0. Tous les mots code X ∈ C de longueur N devrait vérifier les N – K équations de contrôle de parité de l'équation (2.15).

Quand le nombre de 1 par ligne et le nombre de 1 par colonne est constant, le code LDPC est régulier.

Par conséquent chaque bit du mot code X participe à un même nombre d'équations de contrôle de parité. De même, chacune des équations de contrôle de parité utilise le même nombre de bits. A contrario, les codes LDPC irréguliers [13] sont des codes définis par des matrices de contrôle de parité ou le nombre de 1 par ligne et/ou par colonne n'est pas constant. Les codes irréguliers s'avèrent parfois plus performants en termes de BER que les codes réguliers [13]. D'autres détails sur les codes LDPC sont explicités dans les références [14, 15, 12].

4.1.6.1 Graphe de Tanner :

Le graphe de Tanner [16] est une représentation graphique des codes LDPC. Ce graphe contient deux types de nœuds, les nœuds de variable VNs (Variable Node) et les nœuds de contrôles CNs (Check Node). Les deux types de nœuds sont reliés par des branches. Les VNs et les CNs représentent respectivement les colonnes (les bits du mot code) et les lignes (contraintes de parité) de la matrice H.

La figure (2.4) représente la matrice de contrôle de parité (a) et le graphe de Tanner correspondant (b). Le nœud C_m est relié au nœud V_n par une branche si et seulement si la

composante de la matrice H, $H_{m,n} = 1$. Par convention, les VNs et les CNs sont respectivement représentés par des cercles et des carrés. Le nombre de branches connectées à un nœud s'appelle le degré du nœud. Dans le cas de la figure (2.4-a), la matrice de contrôle H est régulière et les nombres de 1 par colonne et par ligne sont respectivement $d_v = 3$ et $d_c = 6$ où dv et dc sont respectivement les degrés des VNs et CNs. Par la suite, 3 est le degré (nombre de branches connectées à un nœud) de chacun des nœuds de variables $(v_1, \dots, v_{10})$ et celui de chacun des nœuds de contrôle $(c_1, \dots, c_{10})$est 6.

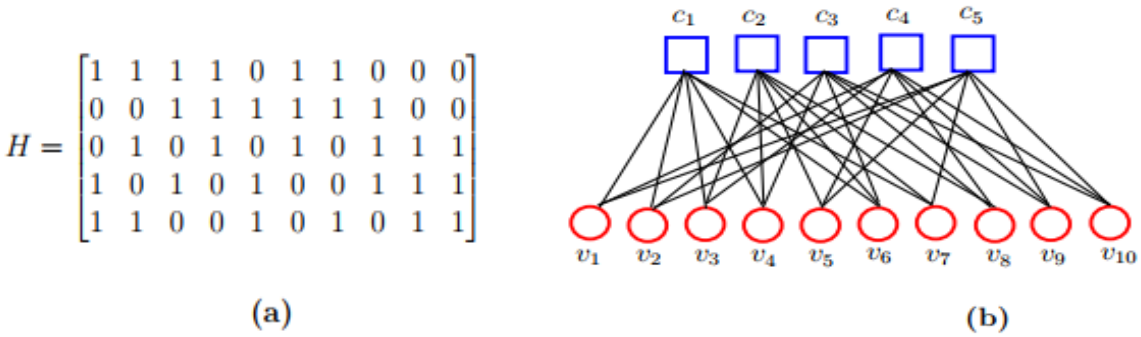

Figure 2.4 – (a) : Matrice de contrôle de parité *H* **(b)** : Graphe de Tanner correspondant

4.1.6.2 Choix de la matrice de contrôle de parité :

Le choix de la matrice de contrôle de parité *H* a un impact majeur sur les performances et la complexité du codeur/décodeur LDPC. Les matrices de faible densié sont toujours souhaitables. La construction de la matrice *H* peut se faire d'une manière aléatoire ou structurée. Les matrices aléatoires sont généralement utilisées sur les processeurs décrits en software par contre les matrices structurées sont souhaitables pour l'implémentation du codeur/décodeur sur les circuits programmables de type FPGAs ou ASICs ou DSPs.

La structure dans une matrice de contrôle de parité assure une flexibilité quant à la conception des unités de fonctionnements effectuant le traitement requis par les nœuds VNs et CNs. Ce type de construction requiert aussi moins de ressources hardware, en termes de mémoire pour le stockage de la matrice de contrôle de parité. C'est le cas par exemple des matrices utilisées dans le standard de la norme DVB-S2, pour laquelle la longueur du mot code (nombre de colonne de la matrice) est fixée à 64800 bits pour la trame normale et 16200 bits pour la trame courte [17]. Quelques constructions des matrices aléatoires sont disponibles online sur la base de données de Mackay [18].

La matrice H peut se mettre sous sa forme systématique $newH = I_{N-K}\, P^T$ par la méthode d'élimination de Gauss-Jordan qui consiste à effectuer un ensemble d'opérations telles que : la permutation des lignes, le remplacement d'une ligne par le résultat de l'addition modulo 2 de deux ou plusieurs lignes. Parfois quelques permutations des colonnes sont aussi effectuées.

4.1.6.3 Codage LDPC

Connaissant une matrice *H* de même forme que celle illustrée par l'expression (2.11), on peut en déduire la matrice génératrice *G* en utilisant l'équation (2.13). Celle-ci a la même forme (forme systématique) que celle présentée par (2.7). L'opération de codage est réalisée par la multiplication de l'information $U = (u_0, u_1, \dots, u_{k-1})$,de longueur *K*, par la matrice *G*. Le mot code construit est alors $X = U \circ G$.

Même si la matrice de contrôle de parité est de faible densité, la matrice *G* ne l'est pas nécessairement, par conséquent le produit interne entre le vecteur *U* et la matrice *G* requiert un nombre important de multiplications et d'additions modulo 2. La complexité du processus de codage est $O(N^2)$, soit exactement $\frac{N^2R(1-R)}{2}$,opérations, où *R* est le rendement du code et *N* la longueur du mot code [19].

La complexité de codage peut être réduite par un pré-traitement des matrices de contrôle de parité. Une technique efficace d'encodage a été développée par Richardson et al. [20]. Avant l'opération de codage, cette technique consiste à un réarrangement de la matrice de contrôle, qui transformée en une matrice triangulaire inférieur, réduisait la complexité quadratique en une complexité linéaire *O*(*N*).

D'autres approches consistent à imposer une certaine structure dans le graphe de Tanner afin de concevoir un codeur simple et de faible complexité. Les codes Repeat Accumulate (RA) en sont un exemple concret [21].

Exemple :

Dans cet exemple on considère la matrice de contrôle de parité de la figure (2.4-a). La nouvelle matrice *H* en forme systématique déduite par la méthode d'élimination de Gauss-Jordan est :

$$H = [I_5\ P_{5\times5}^T] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

La matrice génératrice *G* est déduite à partir de la matrice *H* en utilisant la relation $G \circ H^T = 0$, ainsi *G* sera :

$$G = [P_{5x5} I_5\ ] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour le vecteur d'information $U = (1, 0, 0, 1, 1)$. Le mot code est $X = U \circ G = (1, 1, 1, 0, 0, 1, 0, 0, 1, 1)$. Les 5 premiers bits $(1, 0, 0, 1, 1)$ sont les bits de redondance et les 5 bits qui restent sont les bits de l'information $U = (1, 0, 0, 1, 1)$.Il est donc évident que Les 5 bits de l'information apparaissent clairement à la fin du mot code.

4.1.6.4 Décodage LDPC :

Le décodage est le moyen qui permet de reconstruire le mot *X* transmis à partir du message *Y* (*Y* étant la sortie du canal de transmission). L'opération de décodage utilise la matrice de contrôle de parité *H*. La condition $\hat{X} \circ H^T = 0$ ($\hat{X}$ étant le message binaire estimé par le décodeur à partir du message reçu *Y)* définie les contraintes de parité que doit vérifier le mot estimé $\hat{X}$ pour qu'il puisse s'agir d'un mot code. En considérant la matrice *H* de la figure (I.4-a) et $\hat{X} = (\hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5, \hat{x}_6, \hat{x}_7, \hat{x}_8, \hat{x}_9)$, les 5 contraintes de parité seront alors :

$$\hat{x}_0 \oplus \hat{x}_1 \oplus \hat{x}_2 \oplus \hat{x}_3 \oplus \hat{x}_5 \oplus \hat{x}_6 = 0$$
$$\hat{x}_2 \oplus \hat{x}_3 \oplus \hat{x}_4 \oplus \hat{x}_5 \oplus \hat{x}_6 \oplus \hat{x}_7 = 0$$
$$\hat{x}_1 \oplus \hat{x}_3 \oplus \hat{x}_5 \oplus \hat{x}_7 \oplus \hat{x}_8 \oplus \hat{x}_9 = 0$$
$$\hat{x}_0 \oplus \hat{x}_2 \oplus \hat{x}_4 \oplus \hat{x}_7 \oplus \hat{x}_8 \oplus \hat{x}_9 = 0$$
$$\hat{x}_0 \oplus \hat{x}_1 \oplus \hat{x}_4 \oplus \hat{x}_6 \oplus \hat{x}_8 \oplus \hat{x}_9 = 0$$

Si le mot binaire estimé $\hat{X}$vérifie $\hat{X} \circ H^T = 0$ alors celui-ci est un vrai mot code, sinon le message binaire estimé par le décodeur contient encore des erreurs de transmission.

5. Algorithmes de décodage :

L'avantage des codes LDPC par rapport aux autres codes est de présenter un décodage moins complexe, sa complexité varie seulement linéairement avec la longueur du code. Cet aspect important est principalement dû à la faible densité de la matrice de parité [22].

Les algorithmes de décodage du code LDPC sont des algorithmes de décodages itératifs puisque le message passe entre les variables nœuds et les nœuds de parité itérativement dans un cycle jusqu'à ce qu'un résultat soit atteint ou le processus s'arrête.

Plusieurs algorithmes sont proposés : l'algorithme LLR-BP basé sur le calcul du logarithme du rapport de vraisemblance (log-Likelihood-Ratio), L'algorithme MSA (Min-Sum-Algorithm), l'algorithme SAOMSA (Self-Adjustable Offset Min-Sum-Algorithm).

5.1 Algorithme LLR-BP basé sur la règle "tanh-rule" :

Les étapes de l'algorithme LLR-BP peuvent être explicitées comme suit :

Etape 0 (Initialisation) : Pour toutes les positions (m, n) pour lesquelles $H_{m,n} = 1$, tous les LLRs partant des nœuds V_n vers les nœuds adjacents C_m, seront initialisés comme suit

$$Z_n = L_c = \frac{2y_n}{\sigma^2} \tag{2.18}$$

Etape 1 (Mise à jour des nœuds CNs) : Pour chaque m, et pour chaque $n \in N(m)$ tel que $H_{m,n} = 1$, calculer

$$L_m = \prod_{n' \in N(m) \setminus n} sign(Z_{n'}) 2\, tanh^{-1} \prod_{n' \in N(m) \setminus n} \tanh\left(\frac{|Z_{n'}|}{2}\right) \tag{2.19}$$

Etape 2 (Mise à jour des nœuds VNs) : Pour chaque n, et pour chaque $m \in M(n)$ tel que $H_{m,n} = 1$, calculer

$$Z_n = L_c + \sum_{m' \in M(n) \setminus m} L_{m'} \tag{2.20}$$

$$Z = L_c + \sum_{m' \in M(n) \setminus m} L_m \tag{2.21}$$

Etape 3 (Décision ferme) :

1. Créer une estimation $\hat{X} = [\hat{x}_n]$ telle que $\hat{x}_n = 0$ si $Z \geq 0$ et $\hat{x}_n = 1$ si $Z < 0$
2. L'algorithme de décodage s'arrête si l'un des deux critères suivants est valide :

- Le syndrome $\hat{X} \circ H^T = 0$ et dans ce cas, le message estimé $\hat{X}$ est un mot code valide.
- Le nombre maximal d'itérations est atteint et dans ce cas, La procédure de décodage échoue.

Si aucun des deux critères d'arrêt n'est satisfait, l'algorithme reprend à partir de l'étape 1. Malgré cette importante simplification qu'offre l'algorithme LLR-BP basé sur la règle "tanh-rule", le calcul des fonctions non-linéaires $tanh(x)$ et $tanh^{-1}(x)$ présentes dans l'étape de la mise à jour des nœuds CNs reste un chalenge à relever.

5.2 L'algorithme LLR-BP basé sur l'approche de Gallager :

Cet algorithme procède avec les mêmes étapes de calcul que son homologue basé sur la règle "*tanh-rule*", sauf que l'étape de la mise à jour des nœuds CNs est remplacée cette fois par l'équation suivante :

$$L_m = \prod_{n' \in N(m) \backslash n} sign(Z_{n'}) \, \phi\left(\sum_{n' \in N(m) \backslash n} \phi(|Z_{n'}|) \right) \qquad (2.22)$$

Où $\phi(x)$ est une fonction involutive représentée sur la figure (2.5).

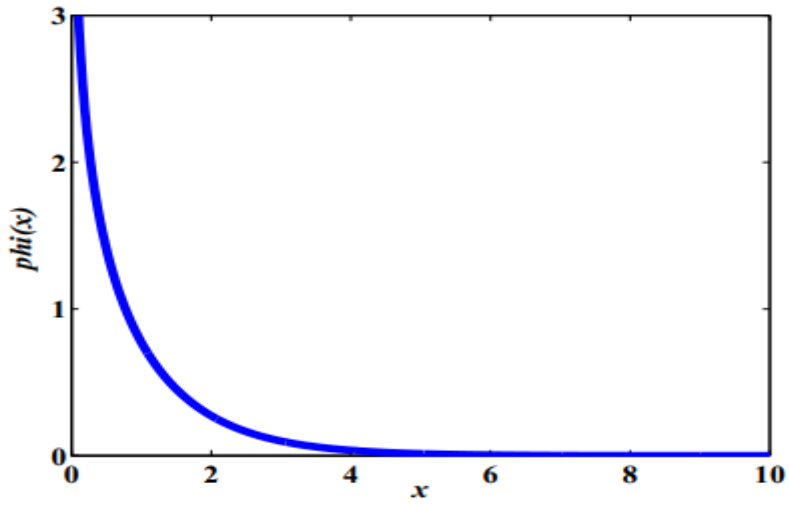

Figure 2.5 – Représentation graphique de la fonction involutive $\phi(x) = -\log(\tanh(\frac{x}{2}))$

Cet algorithme est simple, car La majorité des opérations traitées sont principalement des opérations d'additions. Malgré tout cela, l'implémentation hardware de la fonction non-linéaire $\phi(x)$ constitue un défi à relever.

5.3 Approximation de l'algorithme LLR-BP dans la littérature :

5.3.1 L'algorithme "Min-Sum Algorithm" MSA :

La simplification de l'algorithme MSA (Min-Sum Algorithm) [23, 24], repose sur le fait que le module du message partant d'un nœud C_m vers un nœud V_n est fortement dépendant du plus petit des modules des messages venant, au nœud C_m, des nœuds VNs adjacents excepté celui venant du nœud V_n. Cet algorithme pocéde de la même manière que l'algorithme LLR-BP sauf que la mise à jour des nœuds CNs est effectuée par l'équation (2.23) à la place de l'équation (2.22).

$$L_m = \prod_{n' \in N(m)\backslash n} sign(Z_{n'}) \min_{n' \in N(m)\backslash n} (|Z_{n'}|) \quad (2.23)$$

Malgré, les avantages que présente cet algorithme par rapport aux approximations de l'algorithme LLR-BP, il introduit des dégradations importantes des performances de décodage en termes de BER.

Ces pertes sont essentiellement dues à la surestimation autorisée au niveau du calcul du module $|L_m|$. Ces surestimations deviennent flagrantes surtout, lorsque les modules $|Z_{n'}|$ des messages parvenant au nœud C_m, sont proches car $\sum_{n' \in N(m)\backslash n} \phi(|Z_{n'}|)$devient très supérieur à $\phi\left(\min_{n' \in N(m)\backslash n} (|Z_{n'}|)\right)$ et par la suite $\min_{n' \in N(m)\backslash n} (|Z_{n'}|)$ devient très supérieur à $\phi\left(\sum_{n' \in N(m)\backslash n} \phi(|Z_{n'}|)\right)$ ainsi le module $|L_m|$ calculé sera très grand à ce qu'il devrait être normalement lorsqu'on décode avec l'algorithme LLR-BP.

La figure (2.6) présente la courbe de la fonction non-linéaire $\phi(x)$ et deux exemples de zones encerclées en rouge ou une grande surestimation est certaine pour un module $|L_m|$ issu d'un nœud C_m de degré égale à 6.

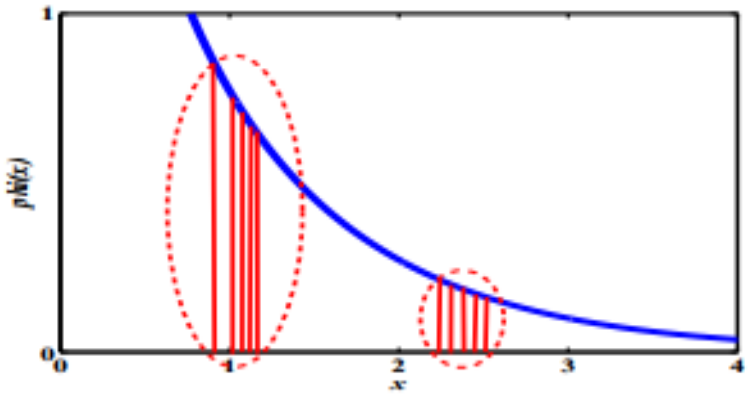

Figure 2.6 – Représentation graphique de la fonction involutive $\phi(x) = -\log(\tanh(\frac{x}{2}))$ avec deux exemples de zones ou une grande surestimation est certaine

5.3.2 L'algorithme "Self-Adjustable Offset Min-Sum Algorithm" SAOMSA :

Dans leur travail Ji et al. [25], proposent l'algorithme SAOMSA pour ajuster le facteur de correction durant le processus de décodage itératif des codes LDPC.

Ce facteur dépend principalement de la différence du second et du premier minimum des modules des messages parvenant au nœud C_m,.

L'approximation porte sur l'équation de la mise à jour des nœuds CNs de l'algorithme LLR-BP basé sur la règle de la tangente hyperbolique, en se servant du Jacobien logarithmique. L'équation de la mise à jour des nœuds de contrôle devient :

$$L_m = \left(\prod_{n' \in N(m)\backslash n} sign(Z_{n'})\right) max\left(\min_{n' \in N(m)\backslash n}(|Z_{n'}|) - \gamma' f(|Z_{min2}| - |Z_{min1}|, 0)\right) \quad (2.24)$$

Avec $f(x) = \log(1 + e^{-|x|})$

Pour réduire les dégradations des performances du décodage en termes de BER dues à la surestimation des modules $|L_m|$, le terme de correction $f(|Z_{min2}| - |Z_{min1}|)$ est multiplié par un facteur de normalisation optimal γ'.

L'algorithme SAOMSA [25] présente une solution qui permet de simplifier l'étape de mise à jour des nœuds de contrôle CNs.

Cette simplification ne dégrade pas trop les performances de décodage [25] comme dans le cas de l'algorithme MSA [23, 24] mais au prix d'une légère augmentation de la complexité de calcul et de l'implémentation matériel.

Ceci est en partie dû à la présence de la fonction non-linéaire $f(x)$ dans le calcul du facteur de correction. Cette fonction $f(x)$ est implémentée soit par les tables de correspondance LUTs [26] ou alors remplacée par des fonctions linéaires par morceaux PWL (PieceWise linear) [26, 27]. Dans le cas des LUTs, $f(x)$ est remplacée par une table de correspondance, la LUT, contenant un nombre de valeur précalculées de $f(x)$ .Si ce nombre est faible, les effets de quantification seront très importants ce qui dégrade fortement les performances de décodage en termes de BER.

Ces performances ne peuvent être améliorées que par une augmentation du nombre de valeurs stockées dans la LUT, ce qui augmente la latence du décodage et les ressources mémoires requises. C'est pour cette raison que certains auteurs proposent de remplacer la fonction $f(x)$ par des PWLs.

Nous présentons les approximations de $f(x)$ par 2, et 5 fonctions linéaires par morceaux, PWLs.

Une première solution a été proposée par [25]. Elle consiste à remplacer la fonction $f(x)$ par une table de correspondance, LUT, qui contient 8 valeurs précalculées de $f(x)$. En traitement avec une virgule fixe en format Qm.5 bits, les valeurs qui devraient être stockées dans la LUT seront celles indiquées par la table (2.1). Ces valeurs sont ensuite reportées sur la figure (2.7).

On voit que chaque constante représente plusieurs valeurs de $f(x)$. Cette solution est simple à mettre en œuvre en Hardware. Malgré cela, elle souffre des effets de quantification qui dégradent les performances de décodage.

x	$f(x)$ [26]	$f(x)$ quantifiée à $f = 5$ bits
$[0, 0.196[$	0.65	0.65625
$[0.196, 0.433[$	0.55	0.5625
$[0.433, 0.71[$	0.45	0.4375
$[0.71, 1.05[$	0.35	0.34375
$[1.05, 1.508[$	0.25	0.25
$[1.508, 2.252[$	0.15	0.15625
$[2.252, 4.5[$	0.05	0.0625
$[4.5, \infty[$	0.0	0

Tableau 2.1 –Approximation [25] de la fonction non-linéaire $f(x) = \log(1 + e^{-|x|})$ par LUT quantifiée à f = 5 bits.

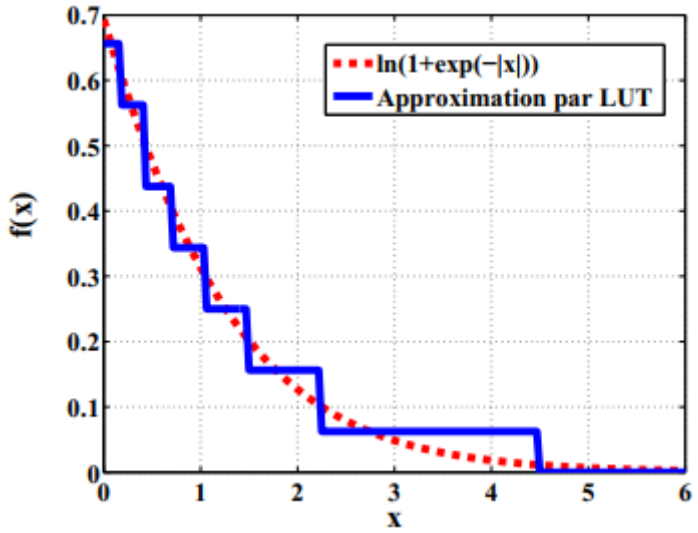

Figure 2.7 – Représentation graphique de la fonction non-linéaire $f(x) = \log(1 + e^{-|x|})$ et son approximation avec LUT quantifiée avec f = 5 bits.

Une deuxième solution [27, 25] qui a pour objectif de remplacer la fonction non linéaire f(x) par deux fonctions linéaires $-\frac{1}{4}x + \frac{5}{8}$, et 0 groupées dans la fonction donnée par :

$$\max\left(-\frac{1}{4}x + \frac{5}{8}, 0\right) = \max(-2^{-2}x + 0.625, 0) \qquad (2.25)$$

La figure (2.8) montre que si x ≤ 2.5 ; la fonction $f(x)$ est remplacée par $-\frac{1}{4}x + \frac{5}{8}$ tandis qu'une fonction nulle la remplace ailleurs. Cette approximation minimise énormément les ressources Hardware requises [25].

Toutefois, le bruit de quantification engendré dégrade les performances de décodages du fait que dans certains intervalles, les deux fonctions linéaires ne sont pas tangentes à la fonction $f(x)$.

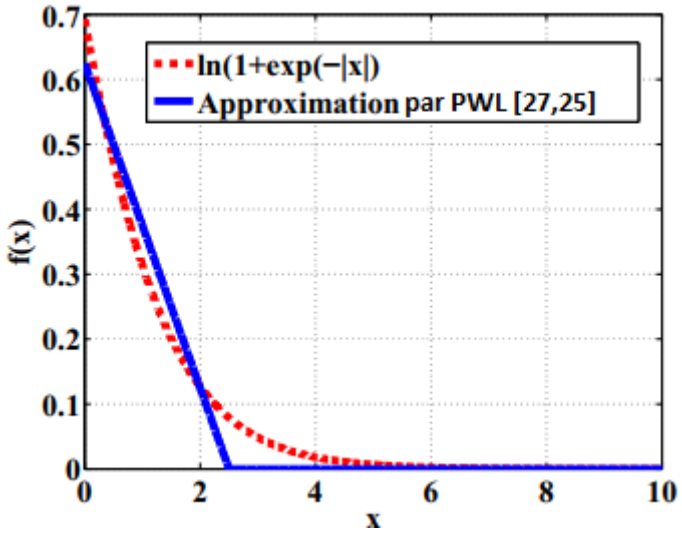

Figure 2.8 – Représentation graphique de la fonction non-linéaire $f(x) = \log(1 + e^{-|x|})$ et son approximation par deux fonctions linéaires [27,25].

Une autre approximation de l'algorithme SAOMSA qui sert à remplacer la fonction non linéaire $f(x)$ par cinq fonctions linéaires par morceaux PWL proposé par [28].

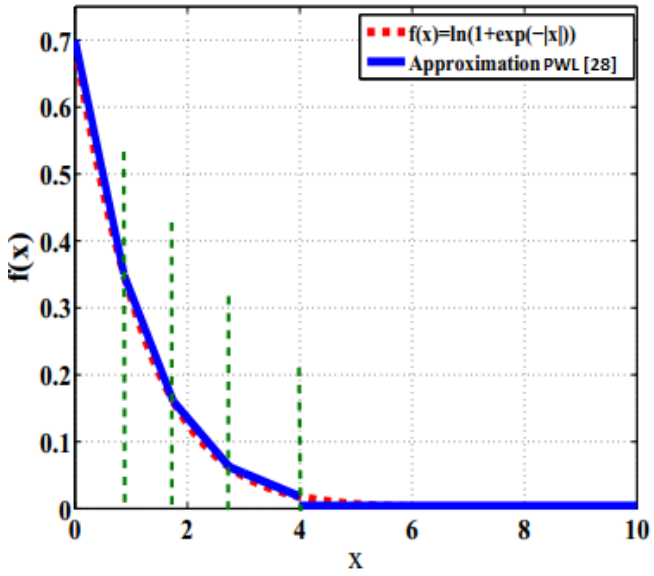

Figure 2.9 – Représentation graphique de la fonction non-linéaire $f(x) = \log(1 + e^{-|x|})$ et son approximation proposée sous forme d'une PWL composée de 5 fonctions linéaires. [28]

La figure (2.9) présente à la fois l'allure de la fonction $f(x)$ et celles des 5 fonctions linéaires $f_i(x) = a_i x + b_i$,choisies pour la remplacer. Les a_i et b_i ont été obtenus à partir de la régression linéaire des valeurs de la fonction $f(x)$dans l'intervalle correspondant.

Par exemple, pour une fonction $f_i(x)$, si $x1$ et $x2$ sont les limites de son domaine de définition, il suffit simplement de résoudre le système suivant :

$$a_i x1 + b_i = \log\left(1 + e^{-|x1|}\right)$$
$$a_i x2 + b_i = \log\left(1 + e^{-|x2|}\right) \quad (2.26)$$

Les a_i et b_i ainsi obtenus pour chaque fonction sont regroupés dans la table (2.2). A partir de x = 4, vu la valeur très faible de $f(x)$, nous avons considéré que la fonction $f(x)$ est nulle pour le reste des valeurs de x.

Intervalle	a_i	b_i	$f(x)$
$[0, 0.875]$	-0.4018	0.7	$-0.4018x + 0.7$
$[0.875, 1.75]$	-0.2151	0.5366	$-0.2151x + 0.5366$
$[1.75, 2.75]$	-0.0982	0.3321	$-0.0982x + 0.3321$
$[2.75, 4]$	-0.0351	0.1585	$-0.0351x + 0.1585$
$[4, +\infty[$	0	0	0

Tableau 2.2 – Les 5 fonctions linéaires proposées pour l'approximation de la fonction non-linéaire $f(x) = \log\left(1 + e^{-|x|}\right)$ [28].

Malgré que cette approximation requière plus d'opérations, elle fournit une meilleure approximation de la fonction *f*(*x*).

6. Conclusion :

Dans ce chapitre nous avons présenté quelques approches de l'algorithme de décodage LLR-BP. La simplification porte essentiellement sur l'équation de la mise à jour des nœuds de contrôle CNs.

L'approximation offerte par l'algorithme MSA est la moins coûteuse vu qu'elle ne requiert que $2 \times (d_c - 1)$ comparaisons pour déterminer le premier et le second minimum des modules des messages parvenant au nœud de contrôle c_m .Mais il introduit des dégradations des performances de décodage en termes de BER

L'algorithme SAOMSA est une autre approximation qui permet d'améliorer les performances d'une part. Mais d'autre part il provoque une augmentation de la complexité de calcul et de l'implémentation matériel dû à la présence de la fonction non-linéaire $f(x)$.

Pour simplifier les calculs d'autres approximations ont été présenté qui sert à remplacer la fonction non-linéaire $f(x)$ par des fonctions linéaires.

Chapitre III

Etude comparative des algorithmes de décodage des codes LDPC

1. Introduction

Une architecture de décodage générique et flexible implantée sur une cible doit vérifier une forte mutualisation des ressources matérielles.

Cette section vise à sélectionner l'algorithme de décodage qui peut être intégrés sur une plateforme à base de DSP. Pour cela, les performances des codes LDPC sont comparés en utilisant des paramètres équivalents en fonction des algorithmes de décodage présentés dans la section 5 du chapitre 2,

Après on modélise la chaine de transmission numérique sous Simulink.

2. Modélisation de la chaîne de transmission numérique avec et sans codage :

La modélisation des systèmes de communication numérique est un moyen efficace et rapide pour mettre en lumière les performances en termes de BER et les principales difficultés de conception de ces derniers.

Afin d'obtenir un système fonctionnant rapidement, nous avons utilisé les blocs codeur et décodeur LDPC de Simulink Communications Toolbox.

Ces blocs permettent de définir les matrices G et H ainsi que le nombre d'itérations de décodage.

La figure 3.1 montre les résultats de la conception d'une chaine de transmission sans codage de canal en utilisant une modulation BPSK et un canal à bruit blanc additif gaussien (AWGN).

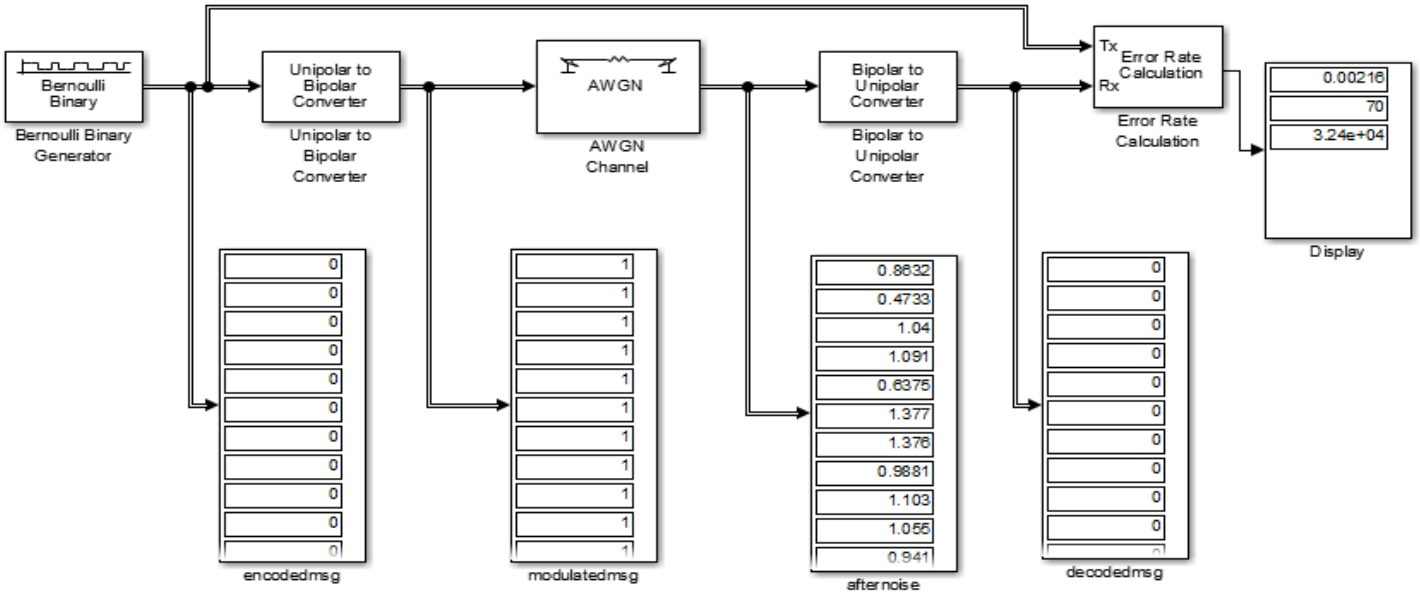

Figure 3.1 – Modélisation d'une chaine de transmission numérique sans codage de canal

La figure 3.2 montre les résultats de la modélisation de la chaine de transmission précédente en ajoutant le codage canal.

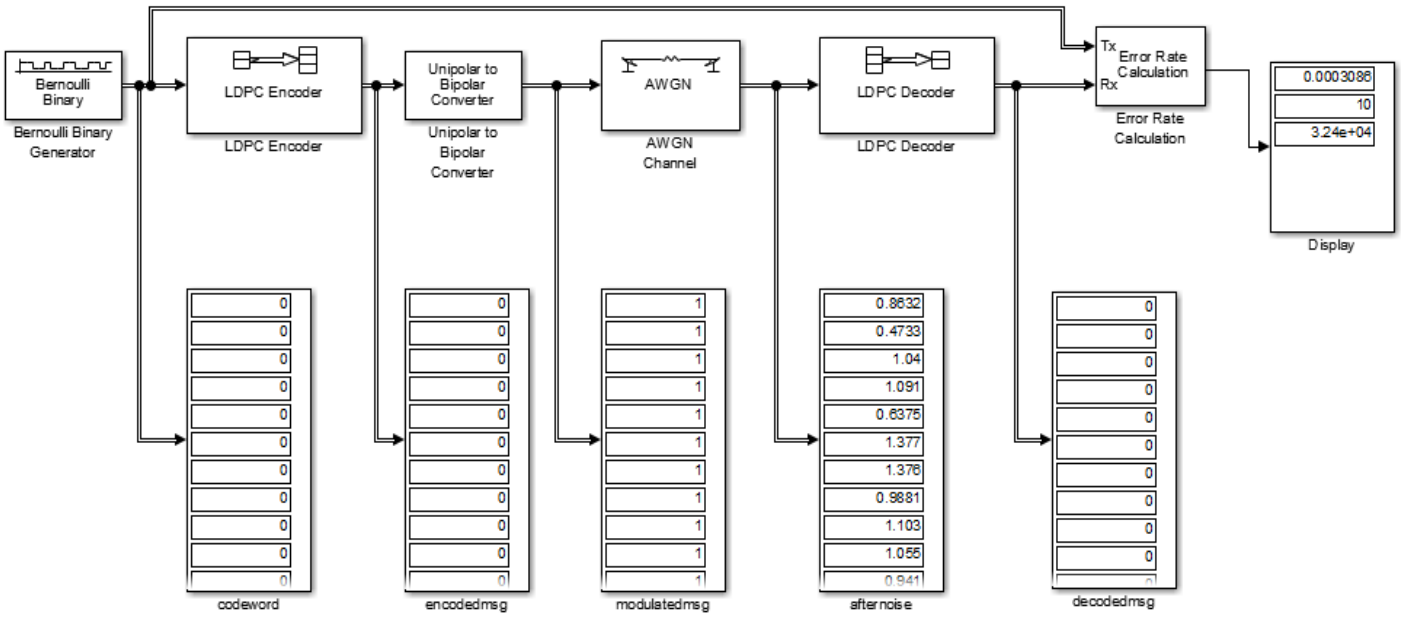

Figure 3.2 – Modélisation d'une chaine de transmission numérique avec codage de canal

On constate que le fait d'introduire les bits de redondance et d'utiliser des systèmes avec codage et décodage de canal améliore les performances en termes de BER.

❖ Modélisation de la chaine de transmission numérique avec codage du canal en utilisant les blocks à fonctions :

À fin de vérifier et valider la fonctionnalité du système de communication nous avons utilisé :

- Fonction pour lire la matrice de contrôle de parité H (5,10) de rendement R=1/2, qui reçoit comme entrée le nombre de ligne et le nombre de colonne de la matrice.
- Fonction pour le calcul de la densité spectrale de puissance d'après la relation suivante :

N0=Es/EsN0 tel que : Es=1 ; EbN0=10^(SNR/10) ; EsN0 = EbN0*R*log2(M).

Et le calcul de la variance tel que var =N0/2. En donnant le rendement R=1/2 M=2 (modulation BPSK) et SNR =1 en dB.

- Fonction « bpsk_modulation » pour moduler les bits du signal transmis en une modulation BPSK c'est-à-dire les symboles peut donc prendre les valeurs {1,-1} (puisque on transmet seulement des 0 on obtient en sortie des 1).
- Par la suite ces symboles sont transmis via le canal de transmission à bruit blanc additif gaussien AWGN « gen_awgn »
- Décodage : le signal reçu du canal est démodulé et décodé par le bloc de fonction « DECODER » pour estimer le message initial. Pour la méthode de décodage nous avons utilisé l'algorithme de décodage LLR-BP basé sur l'approche de Gallager. Il reçoit comme entré la matrice H et le nombre max d'itérations. L'algorithme de décodage s'arrête quand le nombre d'itérations atteint ou que le syndrome est nul.

Et il reçoit comme sortie le message estimé et le nombre d'itération de décodage.

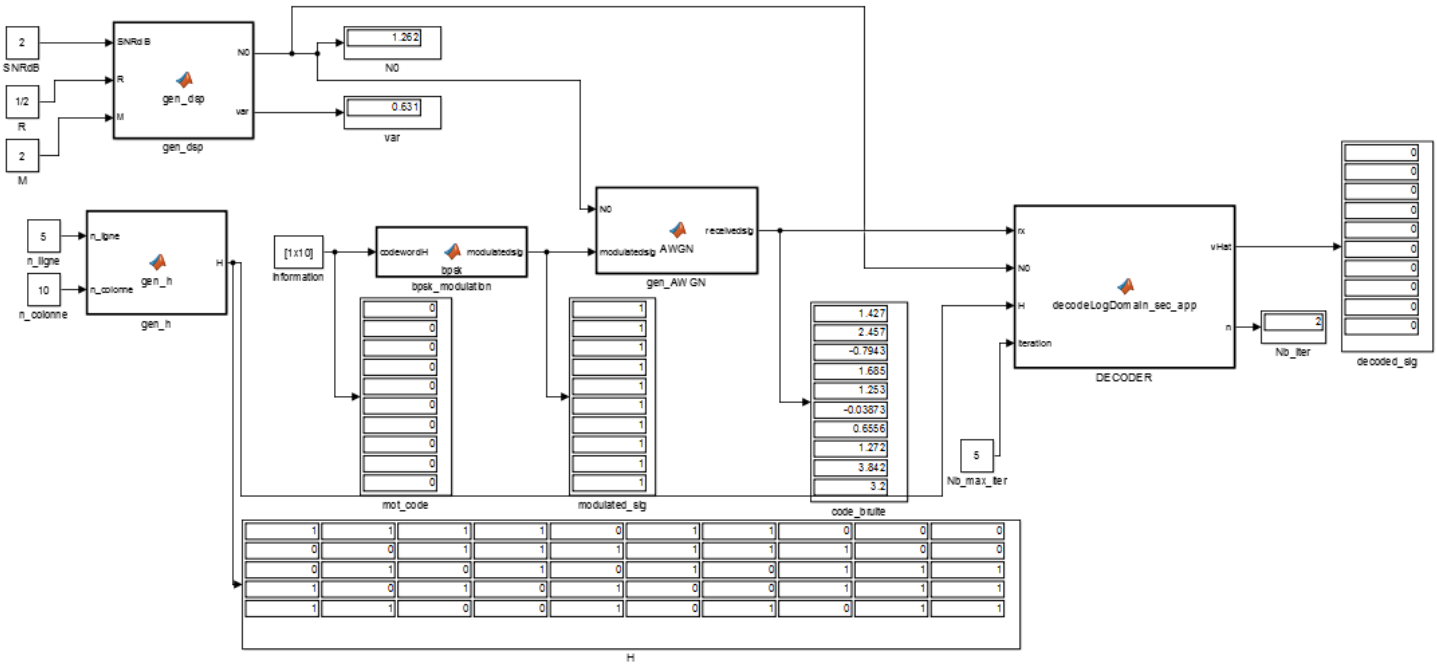

Figure 3.3 – Modélisation d'une chaine de transmission numérique avec codage de canal utilisant les blocs à fonctions.

Pour savoir la qualité de décodeur il faut envoyer plusieurs mot code c'est pour cela nous avons lancé le schéma de Simulink 2000 fois à partir d'un script Matlab voire ANNEXE.

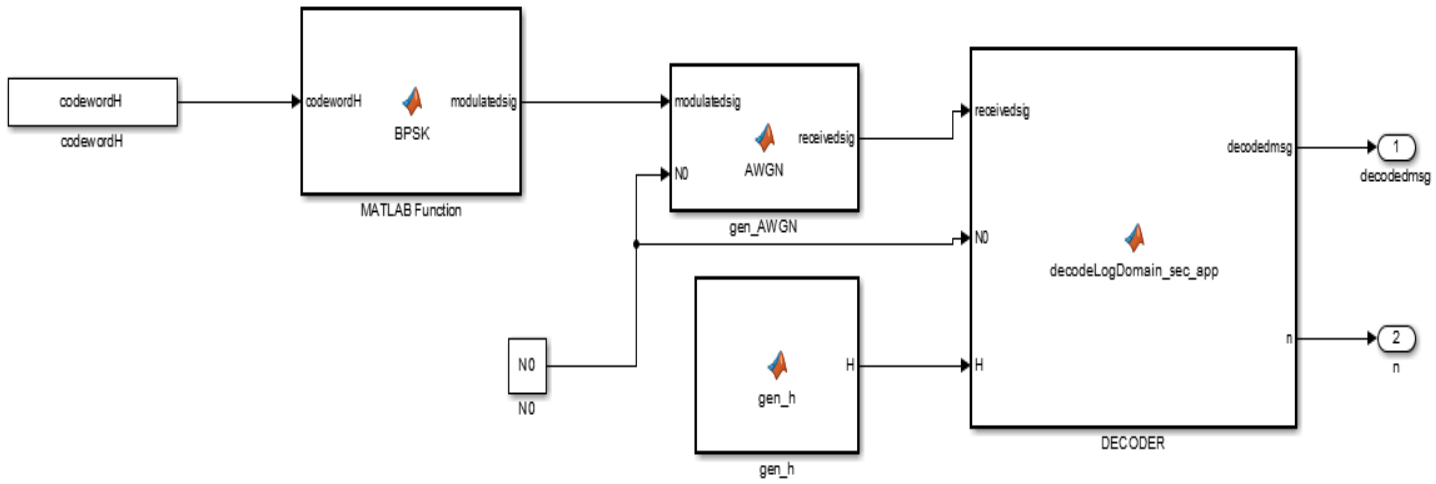

Figure 3.4 – Modélisation d'une chaine de transmission numérique avec codage de canal

3. Comparaison des performances de système au niveau des algorithmes de décodage :

Pour les trois types d'algorithmes de décodage nous avons choisi la matrice (576,288), utilisé dans le standard de communication sans fil haut débit WiMAX.

Les bits du mot code sont modulés avec une modulation bipolaire BPSK et envoyés sur un canal à bruit additif blanc et gaussien (AWGN). Les séquences reçues à la réception ont la même longueur que les mot-codes envoyés, mais les symboles reçus ont une valeur réelle quelconque en raison du canal AWGN.

À la réception, le signal reçu est démodulé et ensuite décodé pour estimer le message initial. Nous avons utilisé les algorithmes de décodage LLR-BP (logarithme du rapport de vraisemblance (Log-Likelihood-Ratio), MSA (Min-Sum Algorithm), SAOMSA (Self Adjustable Offset Min-Sum Algorithm) et l'approximation de l'algorithme SAOMSA proposé par [28]. L'algorithme de décodage s'arrête quand le nombre d'itérations prédéfini est atteint ou quand le syndrome est nul.

La construction et le décodage des codes ont été effectués par l'environnement MATLAB. Nous avons calculé la probabilité d'erreur par bit (BER) pour chaque type de code et nous avons tracé les graphiques de BER en fonction de rapport signal à bruit (SNR) pour analyser leurs performances et les comparer.

La figure (3.5) montre les résultats de comparaison des performances de trois versions d'algorithmes de décodage.

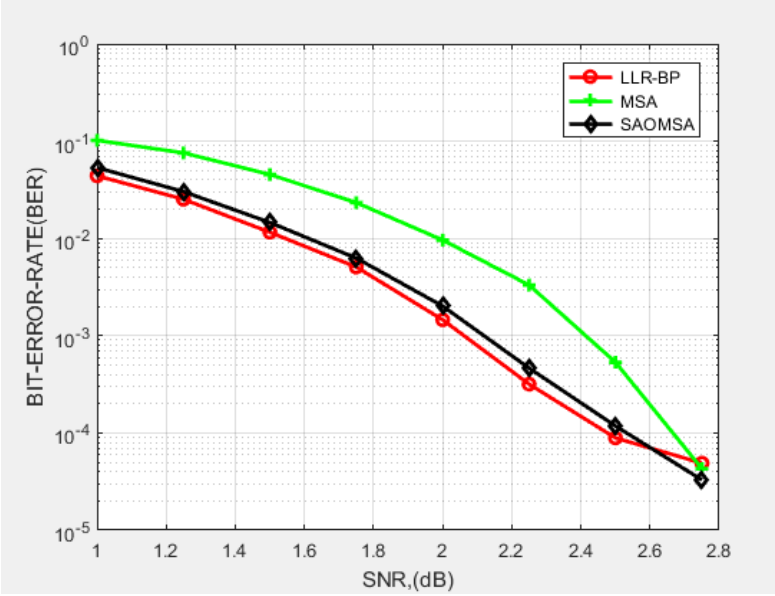

Figure 3.5 – Performance du code LDPC (576,288) pour différents algorithmes le nombre d'itérations est fixée à 50

On remarque que l'algorithme MSA introduit d'importantes dégradation des performances. Concernant l'algorithme SAOMSA, on remarque qu'il est proche de l'algorithme LLR-BP.

La figure (3.6) montre une comparaison des performances de décodage de l'algorithme SAOMSA tel qu'il était initialement conçu par la fonction non-linéaire $f(x)$ [25] et le même algorithme utilisant l'approximation de $f(x)$ par cinq fonctions linéaires par morceaux, PWLs [28].

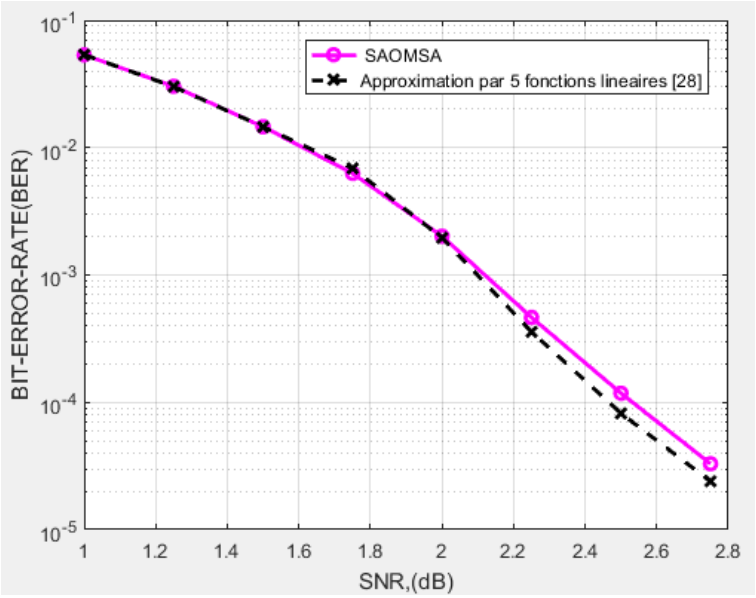

Figure 3.6 – Performance du code LDPC (576,288) pour l'algorithme SAOMSA et son approximation par 5 fonctions linéaires, PWLs.

Il en résulte que l'approximation par cinq fonctions linéaires réalise des performances de décodage comparables à celles de l'algorithme SAOMSA.

4. Comparaison des performances de systèmes en fonction de nombre d'itération :

Pour savoir le nombre maximal d'itération optimal nous avons fait une comparaison des performances pour différents itérations. (50 ,25,16)

Nous avons pris le code LDPC ((576,288) R=1/2) on considère les trois algorithmes de décodage LLR-BP, MSA, et SAOMSA.

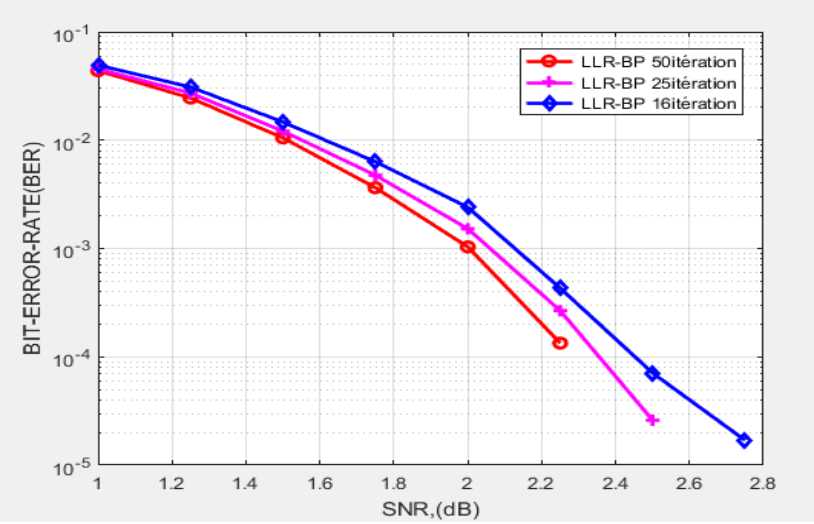

Figure 3.7 – Performance du code LDPC (576,288) algorithme de décodage LLR-BP pour différentes itérations

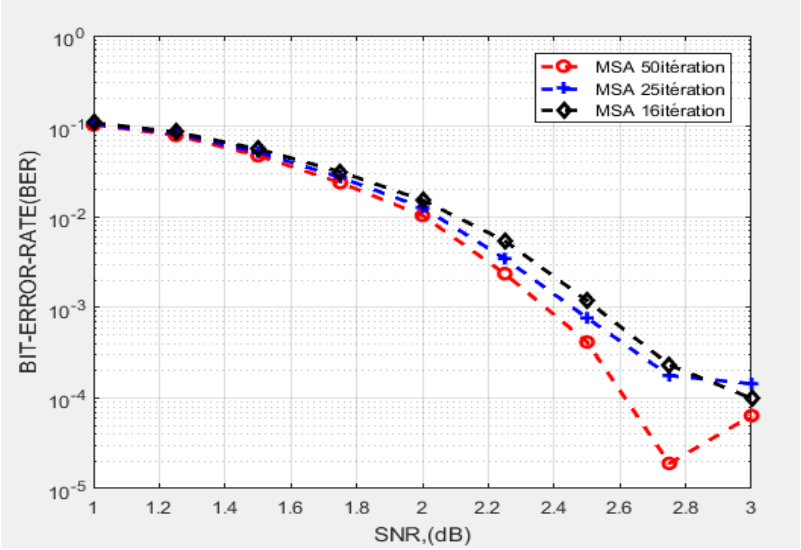

Figure 3.8 – Performance du code LDPC (576,288) algorithme de décodage MSA pour différentes itérations

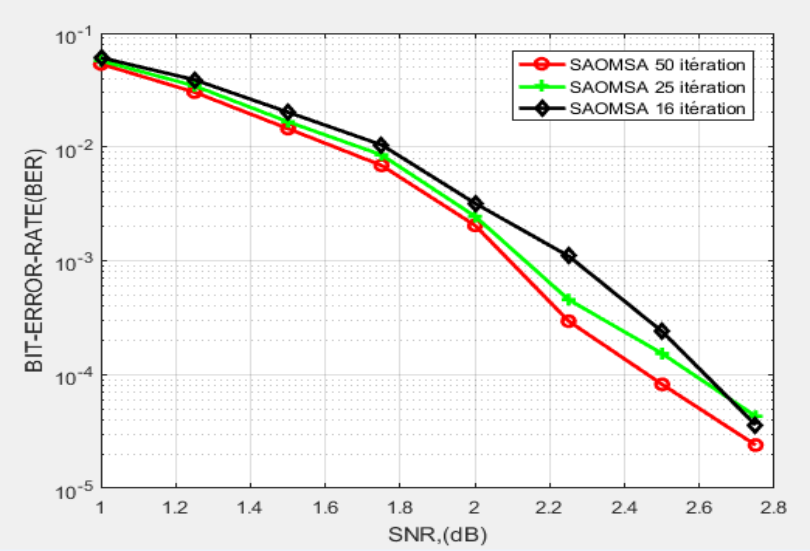

Figure 3.9 – Performance du code LDPC (576,288) algorithme de décodage SAOMSA pour différentes itérations

D'après les figures (3.7, 3.8, 3.9) on peut constater que lorsque on augmente le nombre maximal d'itération on trouve une amélioration des performances en termes de BER.

Mais le temps de décodage et la consommation augmente.

5. Comparaison des performances de systèmes au niveau de la taille du mot code :

La figure 3.10 compare les performances des deux codes LDPC (576,288) et (2304,1152), de la norme WiMAX, et de rendement 1/2, les algorithmes de décodage utilisé sont LLR-BP et MSA sur un canal AWGN, et une modulation BPSK.

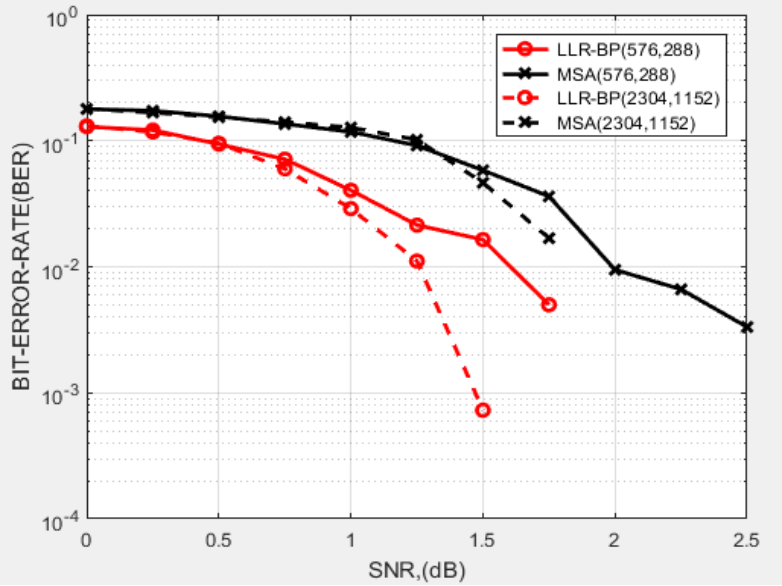

Figure 3.10 – Performance des code LDPC (576,288) et (2304,1152), algorithme de décodage MSA, 50 itération

Nous pouvons constater que pour garantir de meilleures performances il faut utiliser un code de taille grande. En revanche l'augmentation du mot code provoque une augmentation des complexités de calcul et d'implémentation.

6. Conclusion :

Dans ce chapitre, nous avons présenté les résultats comparatifs des performances en termes de BER des codes LDPC.

Nous avons conclu que l'algorithme LLR-BP basée sur l'approche de Gallager a une meilleure performance que les autres algorithmes, l'inconvénient majeur de cet algorithme est la difficulté d'implémenter la fonction non linéaire $\phi(x)$ présente dans l'équation de la mise à jour des nœuds de contrôle CNs, et que l'algorithme MSA ne présente pas des Complexités de calcul mais peut introduire des dégradations importantes des performances en termes de BER.

Donc d'après l'étude comparative on peut dire que l'algorithme le plus performant et qui représente moins de calcul c'est l'approximation de l'algorithme SAOMSA par des cinq fonctions non linéaires [28].

Chapitre IV

Implémentation du décodeur LDPC sur DSP

1. Introduction :

Ce chapitre présente quelques généralités sur les DSPs commençant par citer les avantages du traitement numérique de signal à base de DSP et les critères de son choix, puis décrivant l'architecture de la cible choisis TMS320C6713 et les différents blocs qui les constituent.

Après nous présentons les résultats de l'implémentation d'un décodeur LDPC sur la plateforme DSK TMS320C6713.

2. Généralités sur les DSP :

2.1 Introduction :

Un DSP est un processeur dont l'architecture est optimisée pour effectuer des calculs complexes en un coup d'horloge par la méthode VLIW (Very Long Instruction Word) mais aussi pour accéder très facilement à un grand nombre d'entrées-sorties (numériques ou analogiques). La fonction principale utilisée dans le DSP est la fonction MAC (Multiply and Accumulate),

Les DSP sont utilisés dans la plupart des applications du traitement numérique du signal en temps réel. On les trouve dans les modems (modem RTC, modem ADSL), les téléphones mobiles, les appareils multimédia (lecteur MP3), les récepteurs GPS...

Ils se caractérise par une grande bande passante pour les données liée à l'utilisation de bus indépendants pour les données et les instructions (Architecture Harvard).

Différence entre l'architecture de Von Neuman et Harvard :

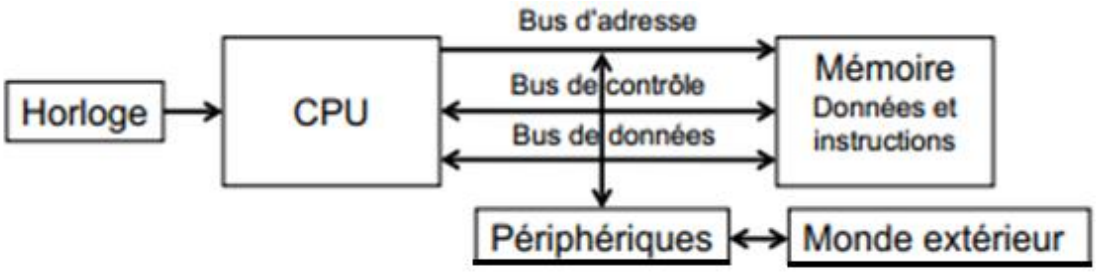

Figure 4.1 – Architecture Von Neuman pour les microprocesseurs

Architecture de HARVARD permet de séparer les mémoires données et programmes cela amène une meilleure utilisation de CPU (chargement du programme et des données en parallèle).

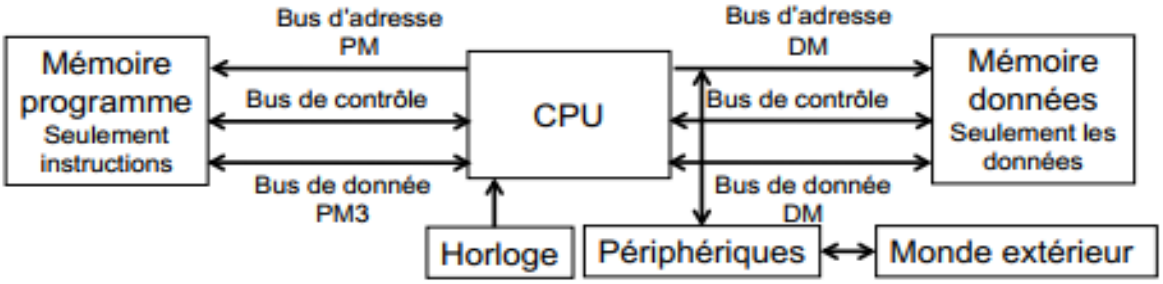

Figure 4.2 – Architecture HARVARD pour les DSPs

La souplesse du développement d'applications à base de DSP est un avantage important en termes de temps, de facilité, de fiabilité, et donc de coût. Comme le suggère la Figure 4.3 créer une application DSP, c'est mener de front deux études distinctes.

La partie matérielle :

Elle inclue la mise en œuvre du DSP lui-même, mais aussi la création d'une chaîne d'acquisition et/ou de restitution du signal (parfois des signaux) à traiter. Les moyens de transformation du signal analogique vers le domaine numérique s'appuient eux aussi sur des circuits spécialisés (AIC23 Codec : CAN, CNA). L'objectif est de rendre l'application finale homogène.

La partie logicielle :

Elle s'appuie sur des outils classiques adaptés aux spécificités des DSP.

L'approche est différente de celle utilisée pour la partie matérielle, car il est toujours possible de recommencer autant de fois que nécessaire pour arriver au résultat.

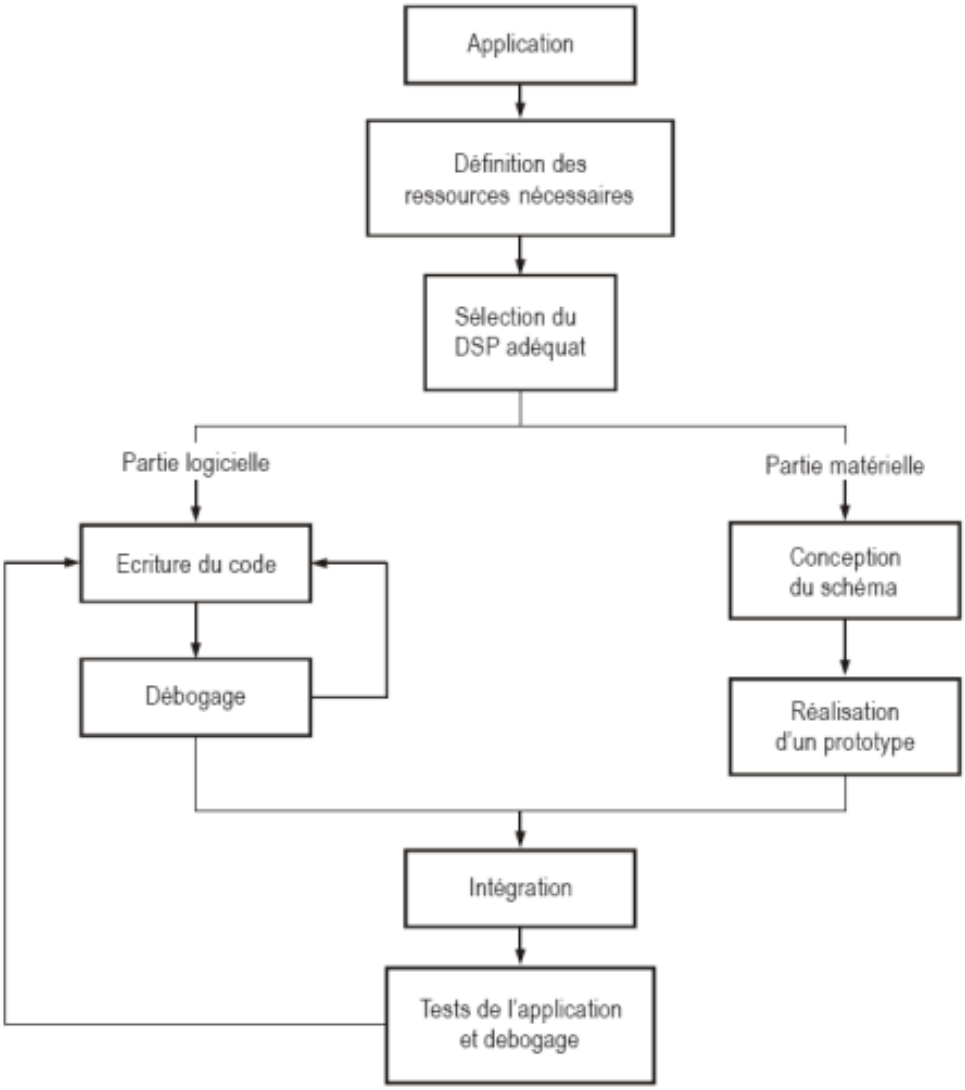

Figure 4.3 - exemple d'un processus typique d'une application à base de DSP

2.2 Avantages du TNS à base de DSP :

Tous les systèmes à base de DSP bénéficient des avantages suivants :

➢ **Souplesse de la programmation :** un DSP est avant tout un processeur exécutant un programme de traitement du signal. Ceci signifie que le système bénéficie donc d'une grande souplesse de développement.

De plus, les fonctions de traitement numérique peuvent évoluer en fonction des mises à jour des programmes, et cela pendant toute la durée de vie du produit incluant le système. La modification d'un filtre numérique ne nécessite pas un changement matériel.

➢ **Des possibilités propres au système de traitement numérique du signal :** certaines fonctions de traitement du signal sont difficiles à implanter en analogique, voire irréalisables (exemple : un filtre à réponse en phase linéaire).

➢ **Stabilité :** en analogique, les composants sont toujours plus ou moins soumis à des variations de leurs caractéristiques en fonction de la température, de la tension d'alimentation, du vieillissement, etc. Une étude sérieuse doit tenir compte de ces phénomènes, ce qui complique et augmente le temps de développement. Ces inconvénients n'existent pas en numérique.

➢ **Répétitivité, reproductibilité :** les valeurs des composants analogiques sont définies avec une marge de précision plus ou moins grande. Dans ces conditions, aucun montage analogique n'est strictement reproductible à l'identique, il existe toujours des différences qu'il convient de maintenir dans des limites acceptables. Un programme réalisant un traitement numérique est par contre parfaitement reproductible

2.3 Les critères de choix d'un DSP :

La sélection d'un DSP se base avant tout sur la puissance de traitement nécessaire, elle dépend de la rapidité de l'exécution des instructions, et donc de l'horloge, et sur le résultat de benchmarks réalisant des fonctions représentatives des traitements à réaliser. Toutefois, la performance du DSP n'est pas le seul critère à prendre en compte, il faut également tenir compte des impératifs suivants :

- Le type de DSP à utiliser (virgule fixe ou flottante) en fonction du domaine d'application.
- Les ressources mémoires utilisés, car s'il faut par exemple exécuter très rapidement une FFT 1024 points, un DSP intégrant plus de 2048 mots de mémoire vive statique peut être nécessaire.
- La nécessité éventuelle d'exécuter un système temps réel, qui s'avérera plus facile à implanter sur certains DSP.
- Le coût du DSP, son rapport « performance/prix » en fonction du volume de production envisagé.
- La pérennité du produit, c'est-à-dire l'évolution prévue par le fabricant (roadmap).

D'autres éléments non négligeables interviennent dans le choix d'un DSP, il s'agit des moyens disponibles pour mener le développement en un temps donné, comme :

- La qualité de la documentation (de préférence claire et abondante).
- La disponibilité de notes d'applications, d'un support technique.
- La qualité du système de développement utilisé.
- La possibilité d'utiliser un langage de haut niveau (Langage C).
- La présence de librairies (du constructeur ou de tierces parties).
- La possibilité de réaliser facilement des prototypes et à faible coût.

Le choix n'est pas toujours simple et certains critères peuvent être contradictoires, certaines règles de choix se dégagent quand même. Ainsi pour des applications destinées à faire un fort volume de production, le critère déterminant est sans conteste le prix du DSP. Pour des applications à faible volume de production, le prix du DSP importe peu, le critère est alors la facilité de développement.

Dans tous les cas, la présence d'un bon support technique est un facteur à ne pas négliger, car un DSP est quand même plus complexe à mettre en œuvre qu'un microprocesseur classique

2.4 Le processeur TMS320C6713

La famille TMS320 est divisée en trois plates-formes qui sont les TMS320C2000, TMS320C5000 et TMS320C6000. Cette dernière englobe deux types de processeurs, les TMS320C62xx, TMS320C64xx et TMS320C67xx à arithmétique flottante.

Le TMS320C6713 est considéré comme le membre le plus performant de la catégorie C67xx, son architecture VLIW lui permet le traitement par paquets de huit instructions en parallèle par huit unités fonctionnelles.

Figure 4.4 - Différent famille de Texas Instruments.

2.4.1 DSK TMS320C6713 :

DSK TMS320C6713 constitue de :

- DSP C6713 de type virgule flottante, de fréquence 225 MHz.
- Codec stéréo (CAN et CNA) AIC23, idéal pour les applications audios, avec des fréquences d'échantillonnage allant de 8 à 96 kHz.
- Mémoire :

  -Une mémoire SDRAM de 16 Mo.

  -Une mémoire Flash non volatile de 512 Ko (256 Ko configurations par défaut)
- GPIO (general purpose I/O)

  -4 LEDS

  -4 switches DIP (dual In Line Package)
- Connecteurs d'extension pour l'utilisation de cartes filles.
- Émulation JTAG via un émulateur JTAG intégré avec hôte USB interface ou émulateur externe
- Alimentation à tension unique (+ 5V)

Les deux figures présentent respectivement le diagramme à bloc fonctionnelles de DSK C6713 et son Layout physique.

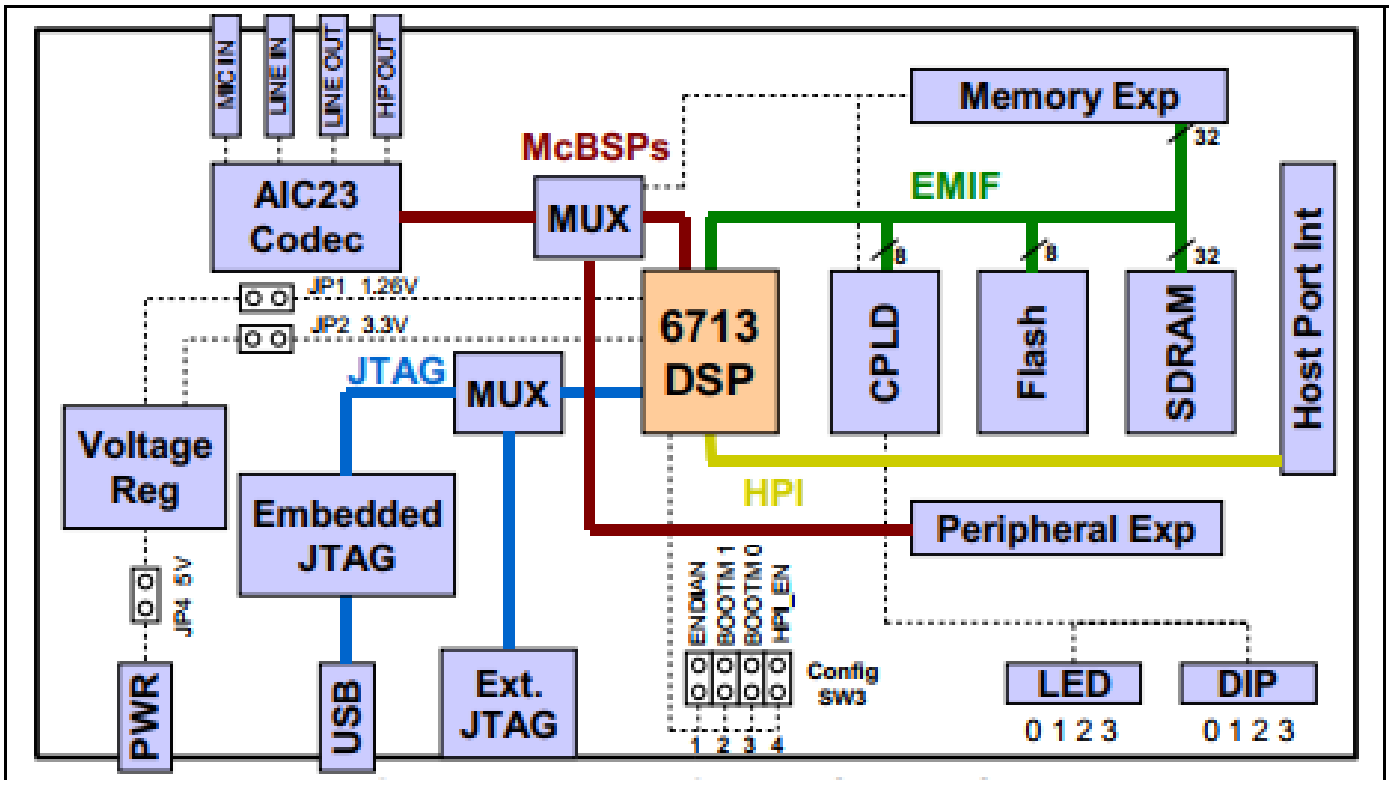

Figure 4.5 - Diagramme à bloc fonctionnels de DSK C6713

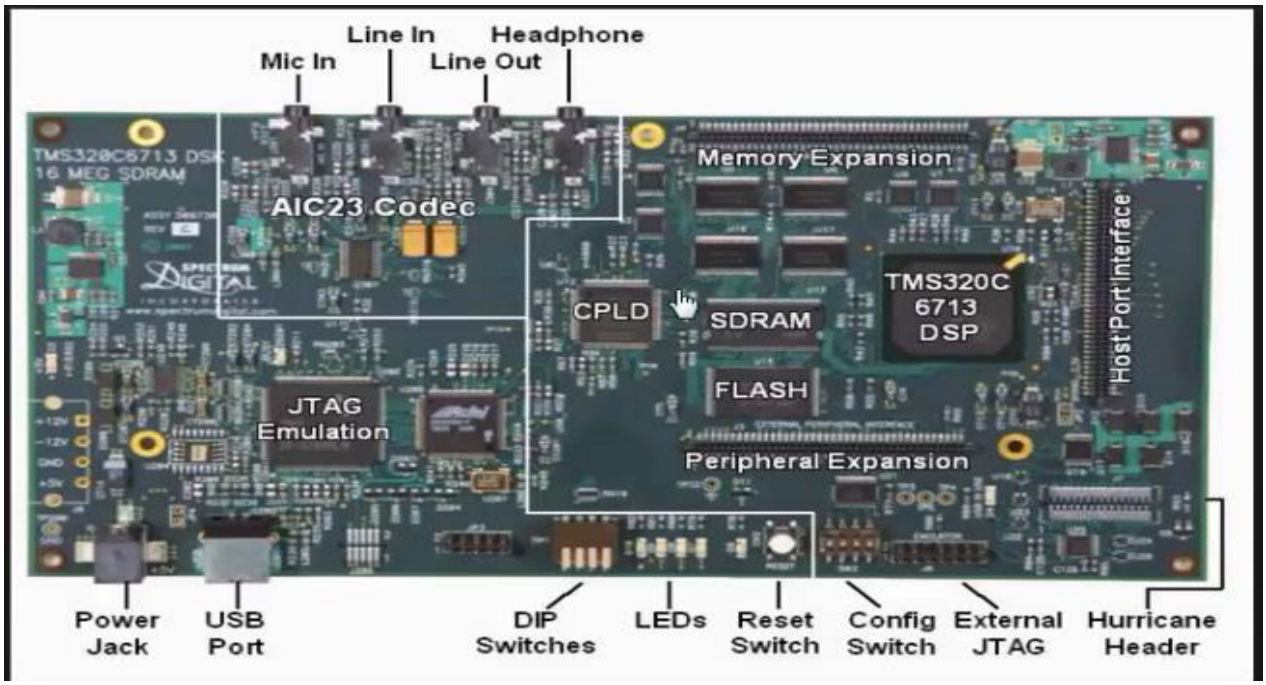

Figure 4.6 - LAYOUT physique de DSK C6713

2.4.2 Architecture interne de TMS320C6713 :

Le TMS320C6713 est constitué de trois parties principales [29] [30] [31]:

- L'unité centrale de traitement CPU
- Les périphériques
- La mémoire.

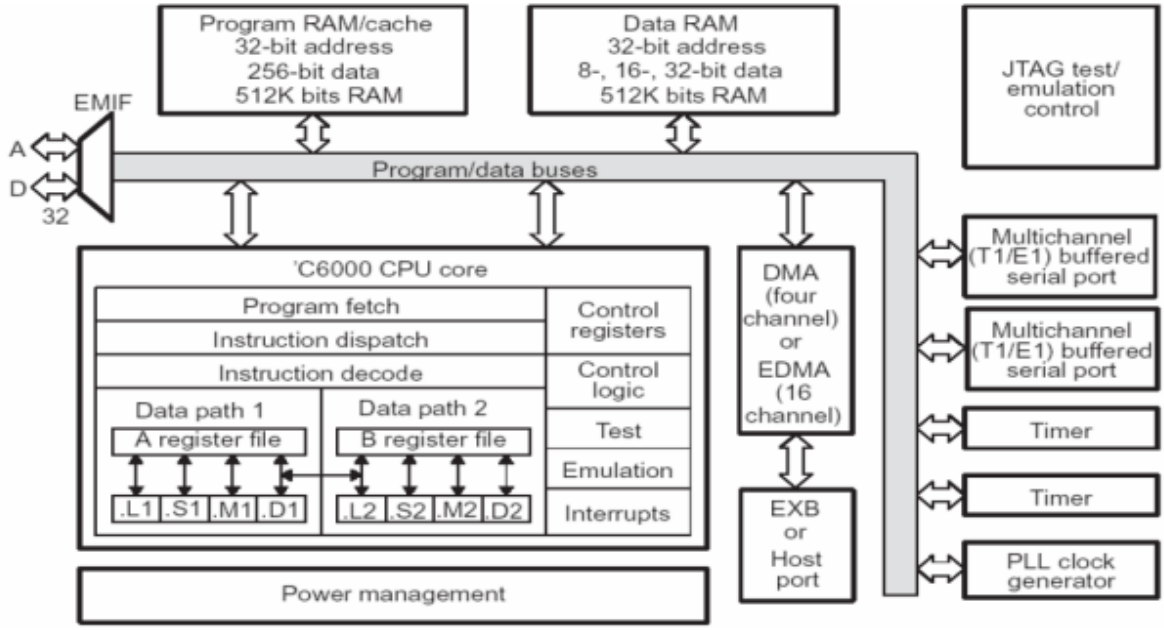

Figure 4.7 - Architecture interne du DSP.

2.4.2.1 Unité centrale de traitement (CPU)

Le CPU est constitué d'une unité de contrôle de programme, de deux unités fonctionnelles, de deux blocs de 16 registres de 32 bits, de contrôleurs d'interruptions et d'autres éléments.

A. Unité de contrôle de programme Elle est constituée des éléments suivants :

- **Program fetch :** unité de chargement de programme.
- **Instruction dispatch :** unité de répartition des instructions.
- **Instruction decode :** unité de décodage des instructions.

B. Unités fonctionnelles : Le CPU contient huit unités : fonctionnelles divisées en deux parties 1 et 2.

- **Unités .Ml et .M2 :** ces unités sont dédiées à la multiplication.

- **Unités .L1 et .L2 :** ces unités sont dédiées à l'arithmétique et la logique.
- **Unités .Dl et .D2 :** ces unités sont dédiées au chargement, la sauvegarde et calcul d'adresse.
- **Unités .S1 et .S2:** ces unités sont dédiées pour le décalage de bit, l'arithmétique, la logique et le branchement.

C. Registres : Le CPU contient 32 registres de 32 bits divisés en deux blocs égaux :

Registre fichier A (AO-A15) et registre fichier B (BO-B15), leurs fonctions sont réparties comme suit :

- **Les registres A1-A2 et BO-B1-B2 :** ils sont utilisés comme registres conditionnels.
- **Les registres A4-A7 et B4-B7 :** ils sont utilisés pour adressage circulaire.
- **Les registres AO-A9, BO-B2 et B4-B9 :** ils sont utilisés comme registres temporaires.
- **Les registres A10-A15 et B10-B15 :** ils sont utilisés pour la sauvegarde et la restitution de données d'un sous-programme.

2.4.2.2 Les périphériques du TMS320C6713

Le TMS320C6713 a plusieurs périphériques qui sont :

Le contrôleur DMA : Il permet sans l'aide du CPU de transférer des données entre les espaces mémoire (interne, externe et des périphériques). Il a quatre canaux programmables et un autre canal auxiliaire.

Le contrôleur EDMA : Il permet le transfert des données entre les espaces mémoire comme le DMA. Il a 16 canaux programmables.

Les compteurs : Le DSP possède deux compteurs qui peuvent être synchronisés par une source interne ou externe et ils sont utilisés comme générateurs de pulsations, compteurs d'événements externes, interrupteurs du CPU après l'exécution de tâches et déclencheur du DMA/EDMA.

Les interruptions : l'ensemble des périphériques contient jusqu'à 32 sources d'interruptions.

2.4.2.3 La structure de la mémoire :

Le TMS320C6713 basé sur l'architecture de Harvard modifiée utilise une mémoire externe et une mémoire interne.

Le DSP C6713 s'interface avec les périphériques externes via un bus de communication 32 bit EMIF (External Memory interFace).

La SDRAM flash CPLD et les connecteurs d'extension sont tous connectés au bus

EMIF a 4 région adressable distinctes appelé CHIP ENABLE SPACES(CE0-CE3) :

CE0 est occupé par SDRAM. Flash et CPLD partagent CE1. CE2 et CE3 est réservé pour les connecteurs d'extension.

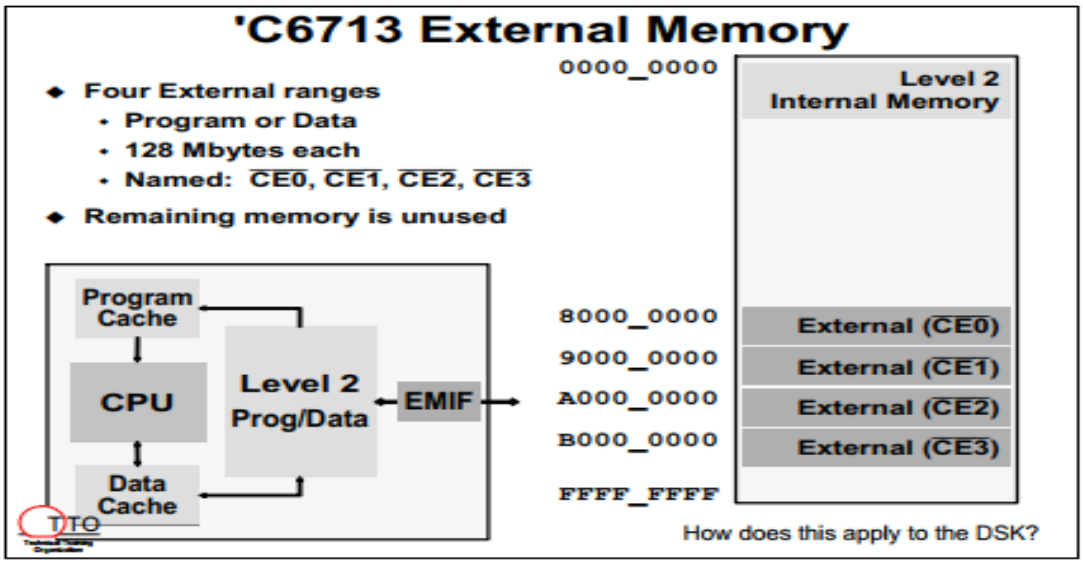

Figure 4.8 - mémoire externe de C6713

La mémoire interne a une taille de 260 KB qui est décomposée en deux niveaux :

- Le niveau (L1) est constitué de deux mémoires caches de 4 KB chacune, (L1P) qui est utilisée pour les programmes et (L1D) qui est utilisée pour les données.
- Le Niveau (L2) est composé de 256 KB de mémoire partagée entre mémoire des données et mémoire de programmes.

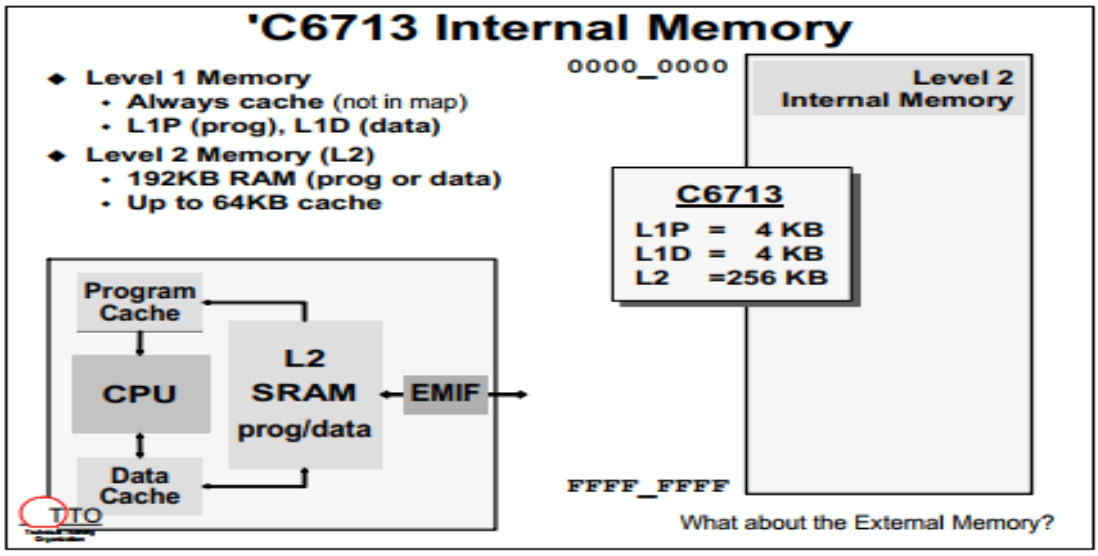

Figure 4.9 - mémoire interne de C6713

❖ Organisation de la mémoire du C6713 DSK

L'organisation de la mémoire de la carte C6713 DSK est décrite au niveau de la figure

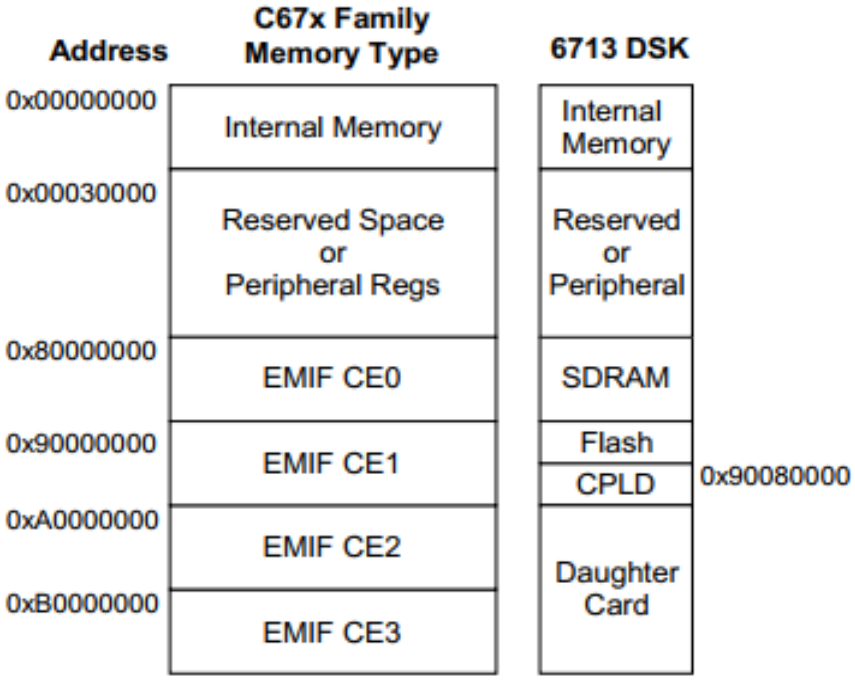

Figure 4.10 - Cartographie de la mémoire du C6713 DSK

2.4.3 AIC23 Codec

DSP s'interface aux signaux audios analogique à travers un codec AIC23, Il permet d'acquérir les signaux analogiques reçus sur l'entrée de la carte (line in) ou (Mic in) afin de les convertir en données numériques utilisables par le DSP.

Lorsque le DSP termine le traitement, il utilise le codec pour convertir les données numériques en signaux analogiques sur les sorties (line out) et (Head phone).

Le codec communique à l'aide de deux canaux série, l'un pour contrôler les registres de configuration interne du codec et l'autre pour envoyer et recevoir des échantillons audios numériques.

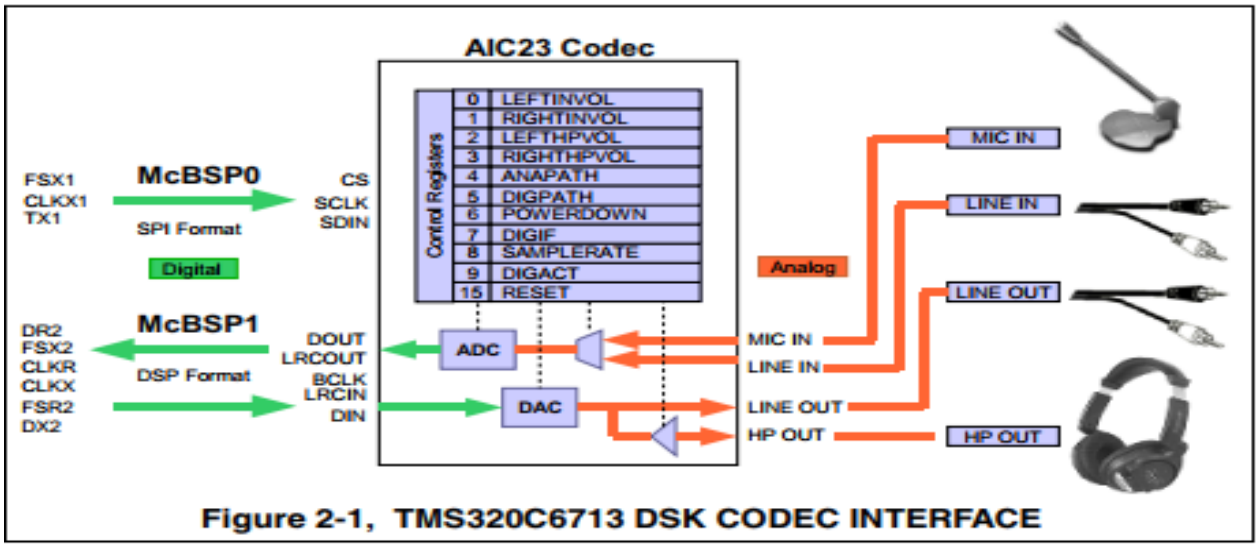

Figure 4.11 - CODEC TLVAIC23

2.4.4 CPLD

Configuration de la carte en software via les 4 registres implémentés dans CPLD :

Sur le DSK 6713 les registres sont principalement utilisés pour accéder aux LEDs, Les DIP switches et contrôler l'interface de la carte fille. Les registres sont mappés dans EMIF CE1 à l'adresse 0x90080000. Ils apparaissent sous la forme de registres à 8 bits avec un interface de mémoire asynchrone simple.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USER_REG	USR_SW3 R	USR_SW2 R	USR_SW1 R	USR_SW0 R	USR_LED3 R/W 0(Off)	USR_LED2 R/W 0(Off)	USR_LED1 R/W 0(Off)	USR_LED0 R/W 0(Off)
DC_REG	DC_DET R	0	DC_STAT1 R	DC_STAT0 R	DC_RST R 0(No reset)	0	DC_CNTL1 R/W 0(low)	DC_CNTL0 R/W 0(low)
VERSION	CPLD_VER[3.0] R				0	BOARD VERSION[2.0] R		
MISC	SCR_5 R/W 0	SCR_4 R/W 0	SCR_3 R/W 0	SCR_2 R/W 0	SCR_1 R/W 0	FLASH_PAGE R/W 0 (Flash A19=0)	McBSP1 ON/OFF Board R/W 0 (Onboard)	McBSP0 ON/OFF Board R/W 0 (Onboard)

Figure 4.12 : Les registres de CPLD

3. Implémentation du décodeur LDPC sur DSP C6713 :

3.1 Diagramme de flux du décodeur LDPC utilisant l'algorithme BP en code C :

Le digramme de la figure 4.13 décrit le déroulement du programme C de décodeur :

Le décodeur commence par l'initialisation des LLRs partant des nœuds V_n vers les nœuds C_m après il mis à jour tous les nœuds de contrôle suivis de la mise à jour des nœuds de variable.

L'algorithme de décodage s'arrête si l'un des deux critères suivants est valide :

Le syndrome est nul, dans ce cas, le message estimé $\hat{X}$ est un mot code valide. Ou le nombre maximal d'itérations est atteint et dans ce cas, la procédure de décodage échoue.

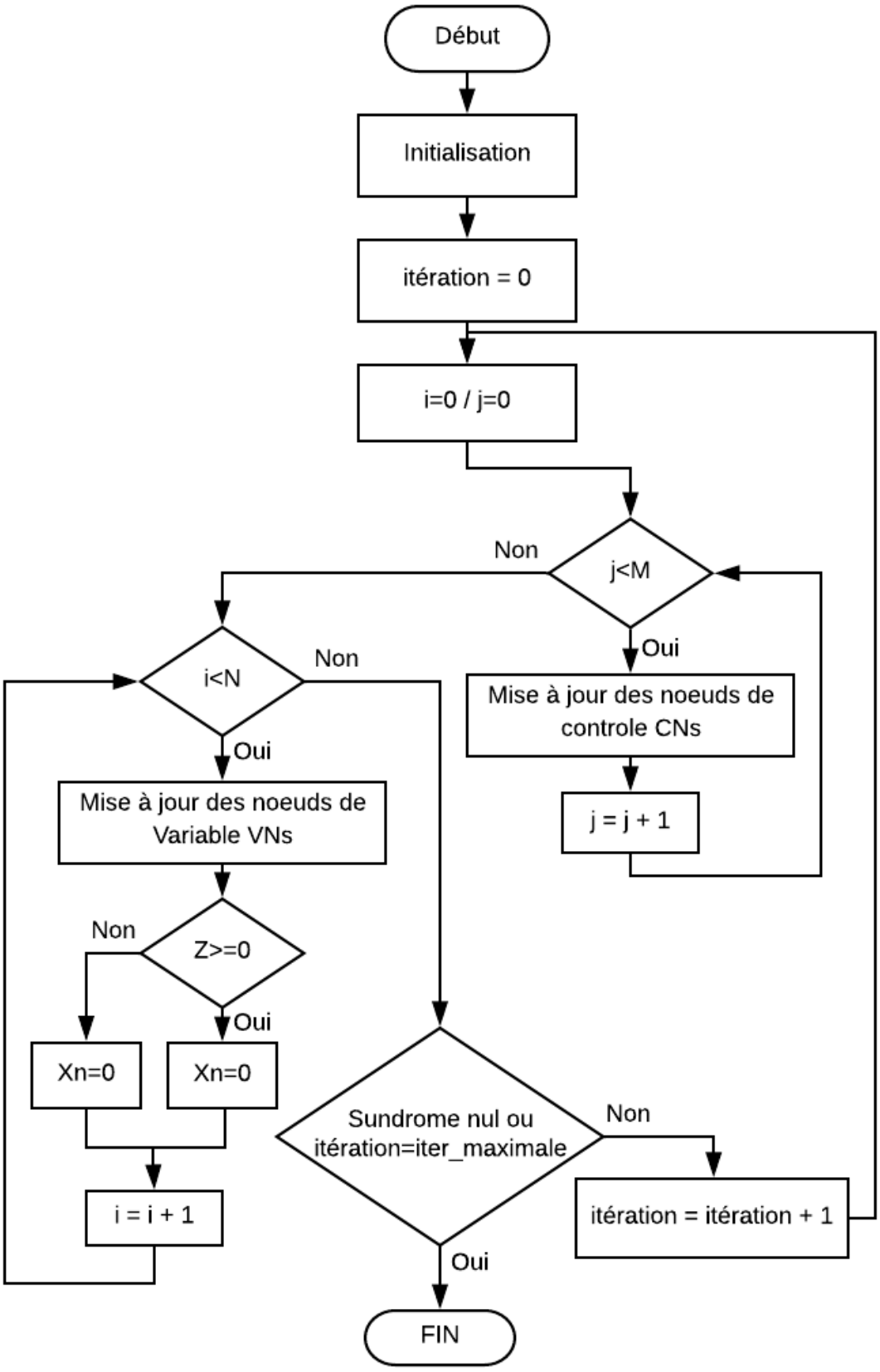

Figure 4.13 - Diagramme de flux décrivant l'algorithme BP du codage LDPC

3.2 Diagramme de flux de la chaine de transmission utilisant le décodage LDPC par l'algorithme BP :

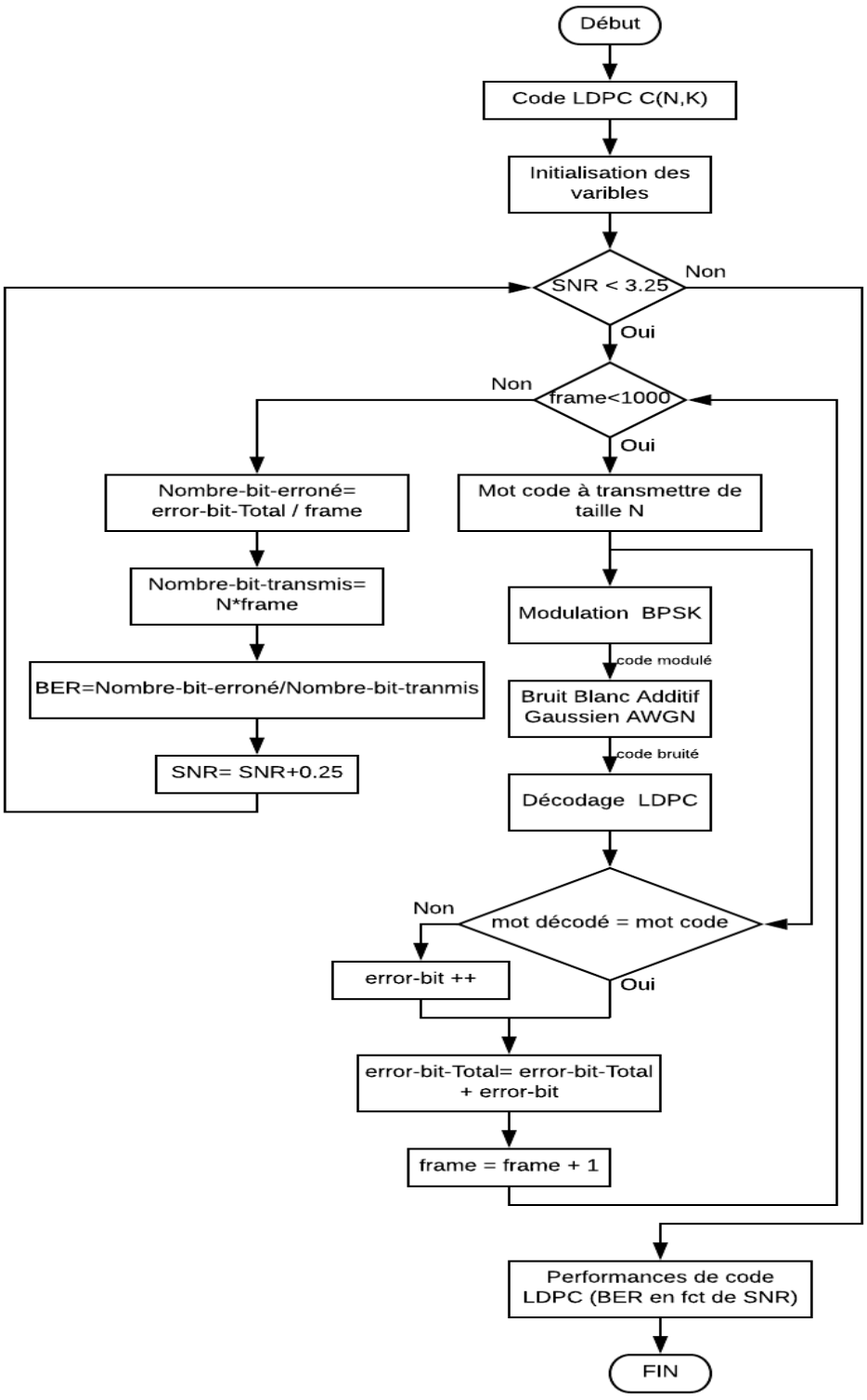

Figure 4.14- Diagramme de flux décrivant le programme principal du décodage LDPC

3.3 Comparaison entre le code C et le code MATLAB :

Pour mettre en œuvre une simulation du système de transmission avec codage canal (encodeur décodeur LDPC) nous avons utilisé le code MATLAB et le code C qui va être transmis vers DSP.

Une comparaison des codes C et MATLAB du décodeur LDPC est souhaitable pour valider notre codes C. Nous avons utilisé l'algorithme de décodage LLR-BP avec R=1/2, N=576, et le nombre maximal d'itération est fixé à 50., en utilisant le même bruit.

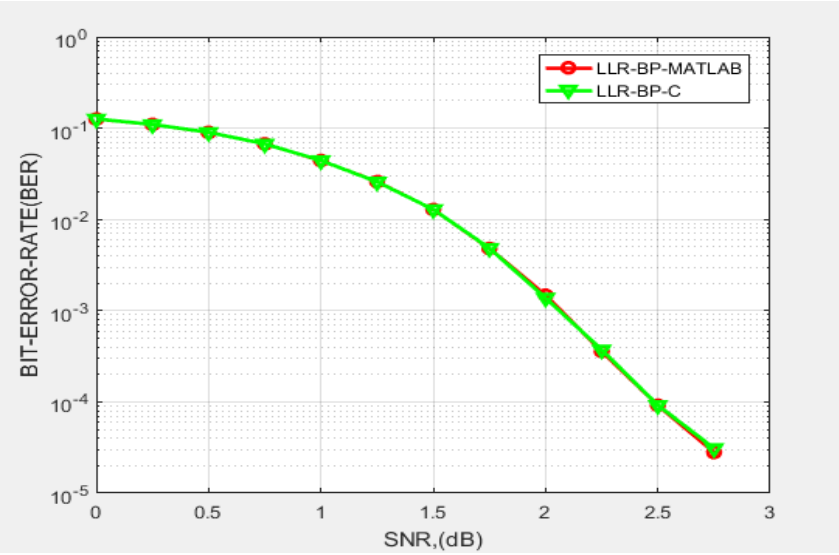

Figure 4.15 – Performance en termes de BER du système LDPC en utilisant le code C et code MATLAB

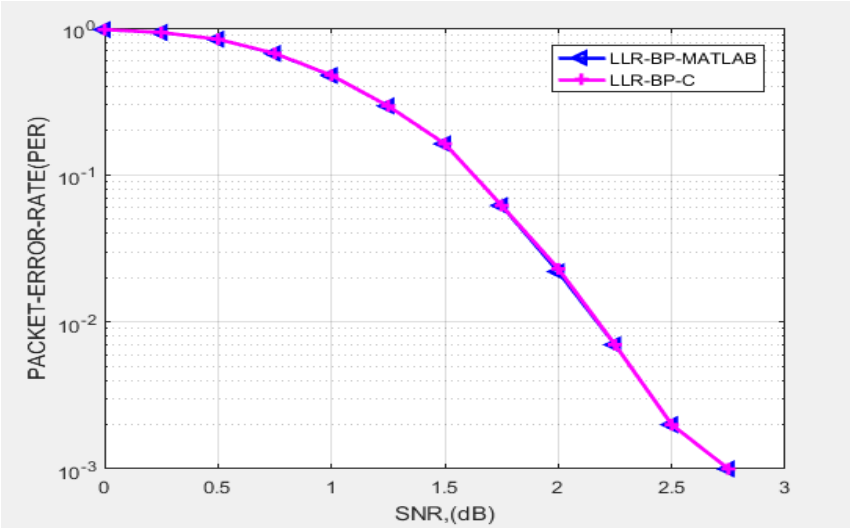

Figure 4.16 –Performance en termes de PER du système LDPC en utilisant le code C et code MATLAB

D'après les figure (4.15) et (4.16) on remarque que les deux code LLR-BP-C et LLR-BP-MATLAB présente les mêmes performances en termes de BER et PER (Packet Error Rate) pour différentes valeurs de SNRs.

La figure 4.17 présente la courbe des valeurs moyens du nombre d'itération pour différents SNRs pour les codes C et Matlab du décodeur LDPC.

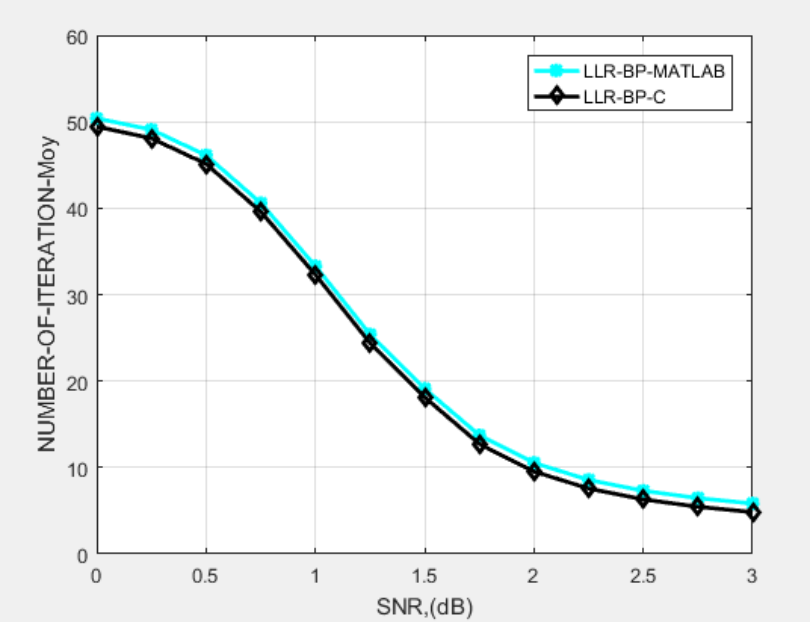

Figure 4.17 – Nombre d'itération en fonction des SNRs pour les codes C et Matlab du décodeur LDPC

D'après la figure on peut remarquer que le code C utilise moins d'itération, cette diminution provoque une diminution au niveau du temps d'exécution.

3.4 Implémentation du décodeur LDPC sur DSP :

Dans cette partie, nous présentons une implémentation d'un décodeur LDPC sur un processeur de signal numérique (DSP). Le DSP utilisé est un processeur de virgule flottante de Texas Instruments TMS320C6713 décrit dans la section 2.4. Le système a été développé et simulé à l'aide du logiciel Code Composer Studio version 4 (CCSv4).

3.4.1 Le fichier de support utilisé :

Le fichier de commande (C6713dsk.cmd) est répertorié dans la figure 4.18, il est indispensable pour le linker afin de définir l'emplacement des variables du code en mémoire.

```
/*C6713dsk.cmd  Linker command file*/

MEMORY
{
  IVECS:    org=0h,          len=0x220
  IRAM:     org=0x00000220, len=0x0002FDE0 /*internal memory*/
  SDRAM:    org=0x80000000, len=0x01000000 /*external memory*/
  FLASH:    org=0x90000000, len=0x00020000 /*flash memory*/
}

SECTIONS
{
  .EXT_RAM :> SDRAM
  .vectors :> IVECS /*in vector file*/
  .text    :> IRAM  /*contient le code à exécuter */
  .bss     :> IRAM  /*variables globales*/
  .cinit   :> IRAM  /*variables initiales */
  .stack   :> IRAM  /* variables locales */
  .sysmem  :> IRAM  /* allocation dynamique */
  .const   :> IRAM
  .switch  :> IRAM
  .far     :> IRAM
  .cio     :> IRAM /*Standard C I/O*/
  .csldata :> IRAM
}
```

Figure 4.18- Fichier de commande C6713dsk.cmd

Il montre que les sections telles que .text et .Stack et .bss résident dans la mémoire interne du DSP C6713 (IRAM), cette dernière offre un accès plus rapide que la mémoire externe.

3.4.2 Résultats de simulation d'implémentation sur DSP :

La figure 4.19 et la figure 4.20 montre respectivement les performances en termes de BER et PER des deux algorithmes de décodage LLR-BP basé sur l'approche de Gallager et MSA.

Pour simplifier l'implémentation nous avons pris le code LDPC (10,5) de rendement 1/2, et le nombre maximal d'itération est fixé à 16.

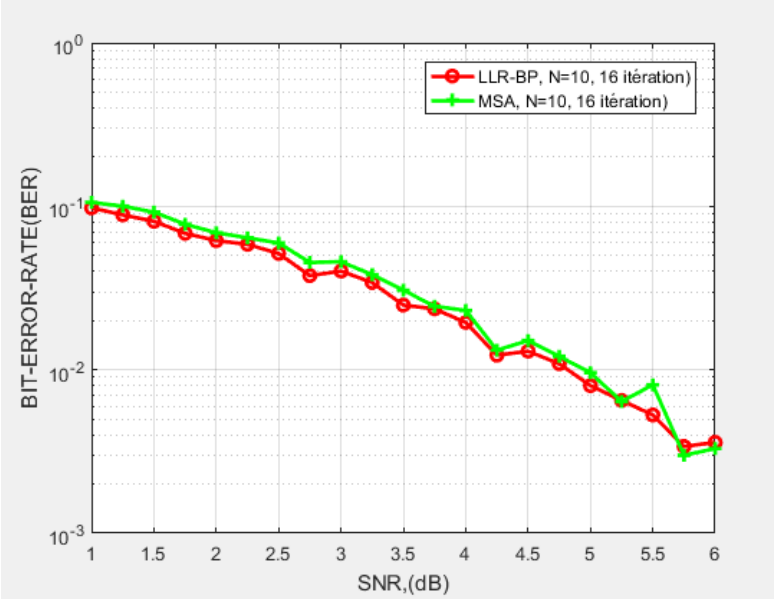

Figure 4.19 – Performance BER du code LDPC (10,5) implémenté sur DSP pour différents algorithmes

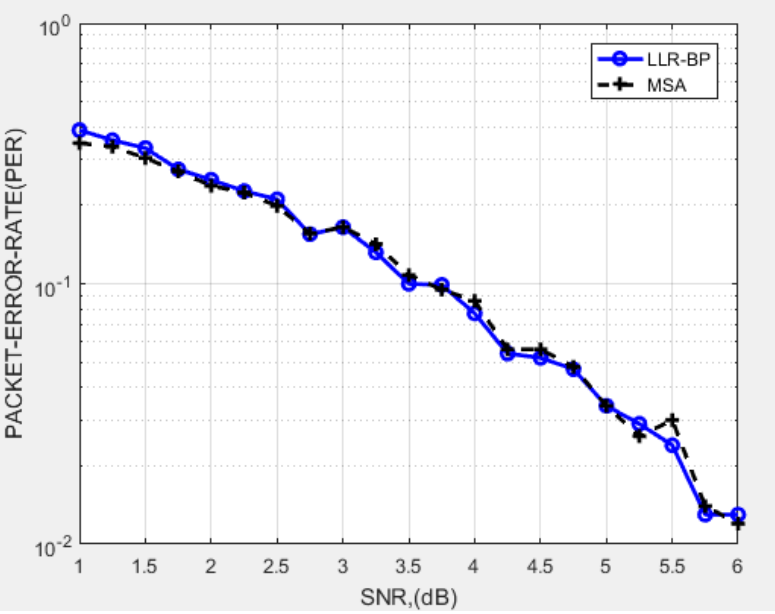

Figure 4.20 – Performance PER du code LDPC (10,5) implémenté sur DSP pour différents algorithmes

Il est clair que l'algorithme LLR-BP est meilleur que l'algorithmes MSA en termes de performances BER et PER, ce dernier présente des dégradations.

La figure 4.21 présente la courbe des valeurs moyens du nombre d'itération pour différents SNRs des deux algorithmes de décodage LLR-BP basé sur l'approche de Gallager et MSA.

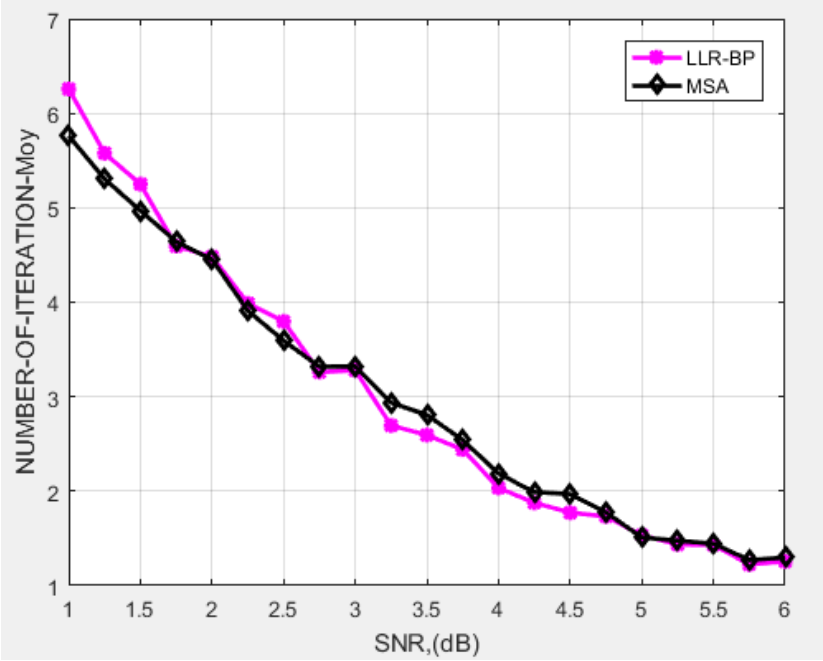

Figure 4.21 – Nombre d'itération des algorithmes en fonction des SNRs

On constate que l'algorithme MSA utilise moins d'itération ce qui donne un temps de décodage inferieur.

Pour valider notre code C nous avons calculé le temps d'exécution des deux algorithmes en utilisant le profilage d'horloge qui sert à compter le nombre de cycle d'horloge des instructions entre deux points d'arrêt.

Le profilage d'horloge affiche directement le nombre de cycles utilisé pour le décodage, dans notre cas la fréquence de C6713 est 255Mhz c'est-à-dire : Temps d'exécution = cycles*1/225 µs. Les résultats des calculs sont répertoriés dans la table 4.1

Différentes valeurs des SNRs	Temps d'exécution pris par l'algorithme LLR-BP en (ms) :	Temps d'exécution pris par l'algorithme MSA en (ms) :
0	58.85	1.63
0.25	4.53	0.26
0.5	8.46	0.47
0.75	59.2	1.69
1	4.55	0.2611
1.25	4.53	0.2610
1.5	59.1	1.71344
1.75	4.5490	0.99705
2	59.1429	0.991305
2.25	16.2304	1.311411
2.5	59.13755	1.309721
2.75	4.588466	0.260976
3	12.3304	0.46718

Tableau 4.1 : Mesure de temps d'exécution des algorithmes LLR-BP et MSA

Il en résulte que l'algorithme MSA a pris un temps de décodage inferieur que LLR-BP cela dues à la simplicité des calculs.

4. Conclusion :

Dans ce chapitre, nous avons présenté des notions de base sur les DSPs. Nous avons cité les avantages d'une chaine de traitement numérique du signal à base de DSP.

Nous avons donné quelques critères du choix de DSP le plus adapté. Nous nous sommes plus particulièrement penchés sur le processeur TMS320C6713 en raison de leur performance.

Finalement nous avons présenté les résultats de l'implémentation d'un décodeur LDPC C (10,5) sur le processeur TMS320C6713 des deux algorithmes LLR-BP et MSA.

Conclusion générale et perspectives

L'objectif de ce projet était d'explorer l'utilisation des DSP dans le processus de codage et de décodage des codes LDPC.

En effet le projet décrit l'algorithme LLR-BP basé sur le calcul des logarithmes des rapports de vraisemblances et ses approximations faites pour diminuer la complexité des calculs et améliorer les performances de décodage itératif.

Il illustre également des études comparatives des algorithmes de décodage des codes LDPC pour choisir l'algorithme le plus proche de l'algorithme LLR-BP et qui présente moins de complexité des calcul et d'implémentations.

Comme perspectives on vise à implémenter sur DSP l'algorithme SAOMSA qui sert à remplacer la fonction non linéaire $f(x)$ par cinq fonctions linéaires par morceaux pour cibler les technologies WiMAX, la diffusion des vidéos numériques par voie satellitaire DVB-S2, et les systèmes sans fil 4G.

Enfin Pour clore, et en termes de recherche, il serait très important de continuer les recherches sur ce type de décodage itératif pour améliorer les algorithmes utilisés afin de faciliter l'implémentation sur les plateformes embarquées.

Références bibliographiques

[1] B. Sklar, Digital Communications: Fundamentals and Applications, Prentice Hall PTR, 2001.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," IEEE ICC '93, Geneva, pp. 1064 1070, 1993.S.-Y. Chung, G. F. Jr., T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," IEEE Communications Letters, vol. 47, pp. 58 – 60, Feb. 2001

[3] M. Valenti, "Iterative Solutions Coded Modulation Library," 3 October 2005. [Online]. Available: http://www.cs.wvu.edu/~mvalenti/documents/CmlTheory.pdf. [Accessed May 2013].

[4] R.G Gallager, Low density parity check codes, Ph.D. thesis, MA: MIT Press, Cambridge, 1963.

[5] J. Oswald, Théorie de l'Information ou Analyse Diacritique des systèmes, Ed. Masson, 1986.

[6] A. Poli et Li. Huguet, Codes Correcteurs, Théorie et Applications, Ed. Masson,1989.

[7] T. Richardson, M. Shokrollahi et R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes", IEEE Trans. Inform. Theory, vol. 47(2), pp. 619-637, 2001

[8] M. C. Davey et D. J. MacKay, "Low Density Parity Check Codes over GF(q)", IEEE Commun. Lett, vol. 2(6), pp. 165-167, 1998.

[9] S. Lin, Y. Kou et M. Fossorier, "Low Density Parity Check Codes Construction Based on Finite Geometries", in Proc.GLOBECOM 2000, 2000. San Francisco, Calif., Novembre 27-Decembre 1.

[10] Y. Kou, S. Lin et M. Fossorier, "Construction of Low Density Parity Check CodesA Geometric Approach", in Proc. 2nd Intl. Symp. Turbo Codes and Related Topics,Brest, France, Septembre 2000.

[11] S.B. Wicker, Error control systems for digital communication and storage, Prentice Hall, New Jersey, 1995

[12] P.G. Moreira, J.C. Farrell, Essentials of error coding, John Willy & sons,Ltd, England, 2006

[13] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, Analysis of low density codes and improved designs using irregular graphs, Proceedings of the thirtieth annual ACM symposium on Theory of computing STOC 98 (New York, USA), 1998, pp. 249–258

[14] W.E. Ryan, An introduction to ldpc codes, August 2003, **http://tuk88.free.fr/LDPC/ldpcchap.pdf**

[15] A. Shorollahi, Ldpc codes : An introduction, April 2003, **http://www.telecom.tuc.gr/~alex/papers/amin.pdf**

[16] T.M. Tanner, A recursive approach to low complexity codes, IEEE Transactions on information theory **IT-27** (1981), no. 5, 533–547.

[17] Digital Video Broadcasting DVB, Second Generation Framing structure, Channel Coding and Modulation systems for Broadcasting, Interactive Services News Gathering and other broadband satellite applications, jun 2004

[18] D.J.C. MacKay, Online database of low-density parity-check codes **http://www.inference.org.uk/mackay/CodesFiles.html**

[19] G.A. Malema, Low-density-parity-check codes: Construction and implementation, Ph.D. thesis, The University of Adelaide, Australia, November 2007

[20] R.L. Richardson, T.G. Urbanke, Efficient encoding of low-density parity-check codes, IEEE Transactions on information theory **47** (2001), no. 2, 638–656

[21] H. Divsalar, D. Jin and R.J. McEliece, Coding theorems for 'turbo-like' codes, Proceedings of the 36th Conference on Communication, Control, and Computing (Allerton, USA), 1998, pp. 201–210.

[22] livre de R. G. Gallager," Low-density parity-check codes". En 1963.

[23] M.P.C. Fossorier, M. Mihaljevic, and H. Imai, Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, IEEE Transactions on information theory **47** (1999), no. 5, 673–680.

[24] J. Chen, A. Dholakia, H. Imai, E. Eleftheriou, M.P.C. Fossorier, and X.Y. Hu, Reduced-Complexity Decoding of LDPC Codes, IEEE Transactions on information theory **53** (2005), no. 8, 1288–1298.

[25] W. Ji, M. Hamaminato, H. Nakyama, and S. Goto, Self-Adjustable Offset MinSum Algorithm For ISDB-S2 LDPC Decoder, IEICE Electronis Express **7** (2010), no. 17, 1283–1289

[26] X.Y. Hu, E. Eleftheriou, D.M. Arnold, and A. Dholakia, Efficient implementations of the sum-product algorithm for decoding ldpc codes, IEEE GIOBCOM'01, DATE '02, vol. 2, 2002, pp. 1036–1036E.

[27] M.M. Mansour and N.R. Shanbhag, High-throughput ldpc decoders, IEEE Transactions on very large scaleintegration systems **11** (2003), 976–996.

[28] A.AIT MADI. A New PWL Approximation for the "Self-Adjustable Offset Min-Sum" Decoding with a Highly Reduced-Complexity", International Journal of Computer Applications (IJCA) **61** (2013), no. 19, 1–6.

[29] **http://c6000.spectrumdigital.com/dsk6713/revc/ (architecture interne de DSP)**

[30] **http://b.l.free.fr/Page13.html**

[31] DSK6713_TechRef de TEXAS INSTRUMANT (2006), Datasheet du C6713 DSK.

Annexes

Script Matlab qui permet de tracer les figures à partir des données des fichiers textes généré par la simulation sur DSP

```
%*****************lire les fichiers txt **********************************
data = importdata('resultBER.txt');
data2 = importdata('resultBER2.txt');
%************************Lire les données du fichier TXT*****************
SNRdB=data(:,1);
PER1=data(:,7);
PER2=data2(:,7);
%************************tracer PER en fct de SNR ************************
semilogy(SNRdB, PER1,'bo-','LineWidth',2);
hold on;
grid on;
semilogy(SNRdB, PER2,'k+--','LineWidth',2);
xlabel('SNR,(dB)')
ylabel('PACKET-ERROR-RATE(PER)')
legend('LLR-BP','MSA')
```

Script Matlab qui permet de lancer le modèle Simulink de la chaine de transmission en 2000 fois

```
clear all;
clc;
R=1/2;  %code rate
Mo=2;%BPSK modulation
file_c_matrix='matrix_cr12_f0.txt';
n_ligne= 288;
n_colonne= 576;
H=matrix_read(n_ligne,n_colonne,file_c_matrix);
H=sparse(H);
numerrmin =20000;
frame= 0;  %200;
SNRdB=1:0.25:3;
BER_bit1=zeros(size(SNRdB));
num_err_moy=zeros(size(SNRdB));
iter_moy=zeros(size(SNRdB));
M_bits=size(H,1);
N_bits=size(H,2);
k_bits=N_bits-M_bits;
codewordH=zeros(1,N_bits);
fileID = fopen('myfile.txt','w+');
format long;
```

```
for  k=1:length(SNRdB)
    fprintf('SNRdB: %d\n', SNRdB(k));
    frame = 0;  numerr = 0; rat_err1=0; p=0;
    EbN0_lin=10^(SNRdB(k)/10);
EsN0_lin=EbN0_lin*R*log2(Mo); Es= 1 ; N0=Es/EsN0_lin ;sigma=sqrt(N0/2);
while (numerr < numerrmin)
    frame = frame + 1;
    modulatedsig=1-2*codewordH;
    bruit= randn(size(modulatedsig));
    receivedsig =modulatedsig+sigma.*bruit;
    %*********lancer le modéle simulink**********%
simout= sim('decodeer.slx','SimulationMode','normal');
     Outputs=simout.get('yout') ;
     decodedmsg=Outputs.signals(1).values;
     n=Outputs.signals(2).values;
     p=p+n;
     %*******comparer le mot code et le mot décodé********%
    [Num1,Rat1] = biterr(codewordH,decodedmsg);
     numerr=numerr+Num1;
     rat_err1=rat_err1+Rat1;
     %******arret si le nombre de mot transmis est 2000*******%
    if  frame>2000
        break ;
     end
end
%****** calcul du BER , itération moyenne, numero des erreurs moyennes****%
   BER_bit1(k)=rat_err1/frame;
   num_err_moy(k)=numerr/frame;
   iter_moy(k)=p/frame;
   %**** ecrire les reultats dans un fichier texte *****%
fprintf(fileID,'SNRdB=%f, num_err_moy =%f, iter_moy=%f, BER_bit1=%f\n',SNRdB(k),num_err_moy(k),iter_moy(k),BER_bit1(k));
end %%%%%%%forrrr while
```

```
  semilogy(SNRdB, BER_bit1,'m-','LineWidth',2);
  hold on;
  grid on;
  title(' BER en fct de SNR(dB)')
  xlabel('SNR(dB)')
  ylabel('BER')
```