الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Etude

Présenté par :

Melle. Ibtissem DOB
&
Melle. Farah FODIL-CHERIF

Pour l'obtention du diplôme Master en Électronique option : **Système de Vision et Robotique**

Thème

Implémentation d'un Décodeur LDPC sur FPGA

Proposé par : Mr. M.MAAMOUN & Mr. M. AÏDJA

Année universitaire : 2012 / 2013

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Etude

Présenté par :

Melle. Ibtissem DOB
&
Melle. Farah FODIL-CHERIF

Pour l'obtention du diplôme Master en Électronique option : **Système de Vision et Robotique**

Thème

Implémentation d'un Décodeur LDPC sur FPGA

Proposé par : Mr. M.MAAMOUN & Mr. M. AÏDJA

Année universitaire : 2012 / 2013

Remerciements

*N*ous tenons à remercier tout d'abord Dieu tout puissant qui nous a donné le courage et la patience jusqu'au bout de nos études, et qui nous a permis de réaliser notre travail.

*U*n grand merci aux membres du jury, qui ont accepté d'évaluer notre travail de fin d'études.

*C'*est avec un immense plaisir d'exprimer nos remerciements les plus sincères à Mr. M. MAAMOUN qui nous a donné l'opportunité de découvrir le domaine de la communication numérique et de décodage LDPC à travers un projet dans son intégralité, qui, en tant que promoteur de mémoire, s'est toujours montré à l'écoute et est très disponible tout au long de la réalisation de ce mémoire, ainsi que pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nous remercions également Mr. M. Aïdja notre Co-Promoteur, pour son collaboration et son soutien continu.

*N*os chaleureux remerciements s'adressent également à tous nos professeurs du master II SVR, pour leur générosité, leur suivi continu et la grande patience dont ils ont su faire preuve malgré leurs charges académiques et professionnelles.

*N*ous tenons à remercier vivement toute l'équipe pédagogique du département d'électronique de l'université SAAD DAHLEB de BLIDA.

*N*ous aimerions aussi présenter un grand merci à tous les étudiants du master II SVR pour le partage des connaissances et la collaboration qu'ils ont présenté durant toute cette année universitaire.

Dédicace

Ibtissem

Je dédie ce mémoire à mes chers parents, ma source de courage, de soutien et de bonheur, merci pour tous ce que vous avez fait pour moi.

A mes sœurs, mon frère pour leur soutien moral, compréhension et aide continu.

Un merci aussi à mon binôme, mes chères amis (broox mina rochi chou tchektchouka) qui m'ont supporté ces derniers temps, à tous mes amis et à toutes les personnes qui m'on prodigué des encouragements et se sont données la peine de me soutenir durant cette année.

Enfin à tous ceux qui de prés ou de loin ont rendu ce travail possible.

Fréha

الملخص:

الموضوع الرئيسي لهذه المذكرة هو " تنفيذ جهاز فك الشيفرات LDPC على FPGA " . إخترنا نموذجا عالي الأداء لجهاز فك الشيفرات LDPC العشوائي و الذي يرتكز على نهج جديد اقترح مؤخرا من قبل مأمون و الآخرون تحت عنوان « ***Controlled start-up stochastic decoding of LDPC codes*** » .

عملنا ينحصر في دراسة النمودج المختار ، إنشاء وصفVHDL مناسب يتولد تلقائيا من Matlab و أخيرا تنفيذ نموذج فك الشيفرات على FPGA

الكلمات المفتاحية:

جهاز فك الشيفرات LDPC العشوائي ، LDPC، تشفير القناة، فك التشفير التكراري، التنفيذ على FPGA .

Résumé :

Le sujet principal de ce mémoire est « Implémentation d'un décodeur LDPC sur FPGA ». Nous avons opté pour un modèle de décodeur LDPC stochastique très performant qui repose sur une nouvelle approche proposé récemment par MAMOUNE et all, intitulé « Décodage LDPC Stochastique à démarrage contrôlé ». Notre travail consiste à étudier le modèle choisi, générer une description VHDL appropriée automatiquement à partir de Matlab et enfin implémenter notre décodeur sur FPGA.

Mots clés :

Décodeur LDPC stochastique, Contrôle de parité à faible densité, Codage de canal, Décodage itératif, Implémentation FPGA

Abstract :

The main subject of this document is « Implementation of LDPC decoder on FPGA ». We opted for a high-performance stochastic LDPC decoder, based on a new approach, proposed recently by MAMOUNE and all., entitled « Controlled start-up stochastic decoding of LDPC codes ».
Our work consists of studying the chosen model, generate an appropriate VHDL description automatically from Matlab and finally implement our decoder on FPGA.

Key words:

Stochastic LDPC decoder, Low density parity-check, channel coding, iterative decoding, implementation on FPGA

Abréviations :

LDPC	Low Density Parity Check		
FPGA	field-programmable gate array		
VHDL	VHSIC Hardware Description Language		
TEB	Taux d'Erreur par Bit		
SSI	Small Scale Integration		
MSI	Medium Scale Integration		
LSI	Large Scale Integration		
VLSI	Very Large Scale Integration		
ASIC	application-specific integrated circuit		
DSP	digital signal processor		
VHSIC	Very High Speed Integrated Circuit		
IEEE	Institute of Electical and Electronics Engineers		
SP	Sum-Product		

Liste des figures :

Table des matières

Introduction générale

Présentation

Le travail présenté dans ce manuscrit a été mené à Blida(Algérie), durant l'année universitaire 2012 / 2013, sous la surveillance et le suivi de Mr M. Maamoun à l'université SAAD Dahleb de Blida, Faculté de Technologie, Département d'électronique.

Contexte

En 1993, dans une conférence internationale IEEE de télécommunications, C. Berrou et A. Glavieux présentaient un nouveau schéma de décodage des codes de canal: les turbo-codes, et leur algorithme de décodage turbo associé. Turbo-codes ont rendu possible d'obtenir environ quelques dizaines de dB loin de la limite de Shannon, pour un taux d'erreur binaire de 10^{-5}.

A côté de l'impact majeur que les turbo-codes ont eu sur les systèmes de télécommunication, ils ont également fait rendre compte les chercheurs que d'autres codes performants existent. Par conséquent, les codes de contrôle de parité à faible densité (LDPC) inventés dans les années soixante par Robert Gallagher, ont été ressuscité dans le milieu des années Quatre Vingt Dix par David MacKay après avoir été oublié pendant plus de Trente ans.

Dès lors, un dense travail s'est focalisé sur ce domaine qui s'avérait très rentable. Suite à ça, de nombreux travaux ont été développés visant à atteindre le mieux la limite de Shannon. Le challenge des chercheurs était donc de concevoir un décodeur des codes LDPC qui soit le plus robuste possible face au bruit du canal (taux d'erreur binaire très faible) d'une part et qui corrigera les erreurs éventuelle lors de la

transmission d'autre part afin de restituer de l'autre bout du système de communication l'information avec la grande fidélité possible.

Problématique

La littérature a connu de nombreux travaux de décodage des codes LDPC, qui ne cesse d'être développés de jour en jour, mais le problème qui se pose avec cet essor est la complexité de l'architecture en termes de matériel. Cette fois-ci le défi est de concevoir un décodeur LDPC très performant et qui présente l'architecture matérielle la plus simple possible.

Face à ses contraintes les décodeurs LDPC stochastiques apparaissent. L'approche stochastique permet de transformer des opérations arithmétiques complexes sur les probabilités comme la multiplication ou la division par des opérations logiques simples utilisant des portes logiques élémentaires, des bascules JK ou des multiplexeurs.

La première implémentation des codes LDPC stochastique a été décrite par Warren Gross en 2005. [56]

Une approche de décodage plus efficace a été employée pour des codes LDPC par Sharifi et al. en 2006. Puis, en 2007, l'implémentation sur FPGA d'un décodeur LDPC stochastique (1056,528) a été mise en place.

Ces pour ce type de décodage que nous avons opté dans notre travail. Pour cela nous avons choisi un modèle très performant des LDPC stochastique très récemment proposé par Mamoune et all. appelé « Controlled Start-up Stochastic Decoding for LDPC code ».

Objectif et travaux réalisés

L'objectif du présent mémoire est l'étude de l'implémentation d'un décodeur LDPC stochastique sur FPGA, nous avons donc réaliser une description optimale du

décodeur LDPC stochastique, assurant un transfert nœud de variable ↔ nœud de parité très fiable et fidèle, et ce à partir de n'importe quelle forme de la matrice de parité H. Cette dernière sera adaptée selon le modèle de Mackay dit « Alist matrix » supporté par notre programme. Nous avons décrit une technique automatique pour exporter une description VHDL interprétable par ISE Xilinx et System Generator à partir de Matlab, cette technique fournie une description exacte de l'architecture et nous permet d'éviter les erreurs, la perte du temps et d'effort engendré par l'écriture directe de la description VHDL ligne par ligne sur ISE Xilinx.

Afin de faciliter la manipulation pour l'utilisateur nous avons réalisé une interface graphique qui permet de faire le traitement cité ci-dessous automatiquement sans avoir recours à la manipulation du programme lui-même.

Après cela et pour simuler la fiabilité de notre travail nous avons utilisé un système d'évaluation des décodeurs LDPC stochastique réalisé sur Simulink de Matlab, et que nous avons adapté selon les contraintes qu'imposent nos données.

Organisation du manuscrit

Ce manuscrit se décompose en quatre chapitres. **Le chapitre I**, en outre comprend quatre parties décrivant les notions de base nécessaires à la compréhension de notre mémoire. La première partie comporte le principe de la communication numérique et les notions principale associées. La deuxième partie propose une description générale du décodage de l'information. Puisque l'objectif de notre mémoire est l'étude de l'implémentation d'un décodeur LDPC sur FPGA, la troisième partie entamera la description des circuits FPGA, leurs avantages et justifiera le choix d'un tel circuit pour l'implémentation.
Comme tout circuit programmable, les FPGA ont besoin d'une manipulation software qui permette leur mise en marche, pour cela la quatrième partie va contenir la présentation du langage de description VHDL.

Le chapitre II sera consacré à la présentation des codes LDPC, il contient alors la description de ces codes, leur encodage et leur décodage ainsi qu'un petit état de l'art sur les principaux travaux élaboré dans ce domaine.

Le chapitre III présente la description théorique du modèle de décodeur employé, et il va parler plus en détail sur les décodeurs itératifs et stochastiques, ensuite il explique le principe du décodeur faisant objet de notre étude proposé par Maamoun et all.

Le chapitre IV, montrera le déroulement du travail et parlera plus précisément de l'aspect pratique du présent projet et les tâches accomplis dans ce cadre. Ce chapitre va d'abord débuter par la présentation des outils de travail utilisés puis entamera par la suite les travaux réalisés. Pour finir, des interprétations et commentaires vont être mis en place.
Enfin nous terminons notre étude par une conclusion générale.

Chapitre 1Systèmes de communication et circuits à logique programmable

Le présent chapitre –intitulé Généralités- décrit les notions de bases et les outils nécessaires à la compréhension de notre travail.

Dans un premier temps nous allons illustrer la description des systèmes de communication, en expliquant brièvement chacun de ses étages, ensuite nous allons développer la partie de décodage qui est notre centre d'intérêt.
Puisque le but de notre projet de fin d'étude est l'étude de l'implémentation des codes LDPC sur FPGA, la troisième section sera consacrée à la présentation des circuits programmables FPGA, et enfin nous terminerons ce chapitre par une présentation du langage de description VHDL que nous avons utilisé.

1.1 Système de communication

Nous vivons dans l'ère des télécommunications et de l'information. Lors des deux dernières décennies, les communications numériques ont beaucoup évolué. De nos jours, l'information est dans la plupart des cas véhiculée sous forme numérique, que ce soit sur support filaire (fibres optiques par exemple), ou en radio, réseaux cellulaires ou réseaux locaux sans fil ou bien des systèmes de stockage de l'information. Cette évolution a été déclenchée et entretenue par une forte demande de transmission et de traitement fiable, rapide et efficient de l'information de tous types (traitement de la voix, des données ou des images…etc.). Et ce phénomène est présent dans tous les domaines (militaire, gouvernemental, commercial, etc.).

Un système de communication numérique peut se décomposer selon le schéma présenté sur la figure suivante : [1] [2] [W1]

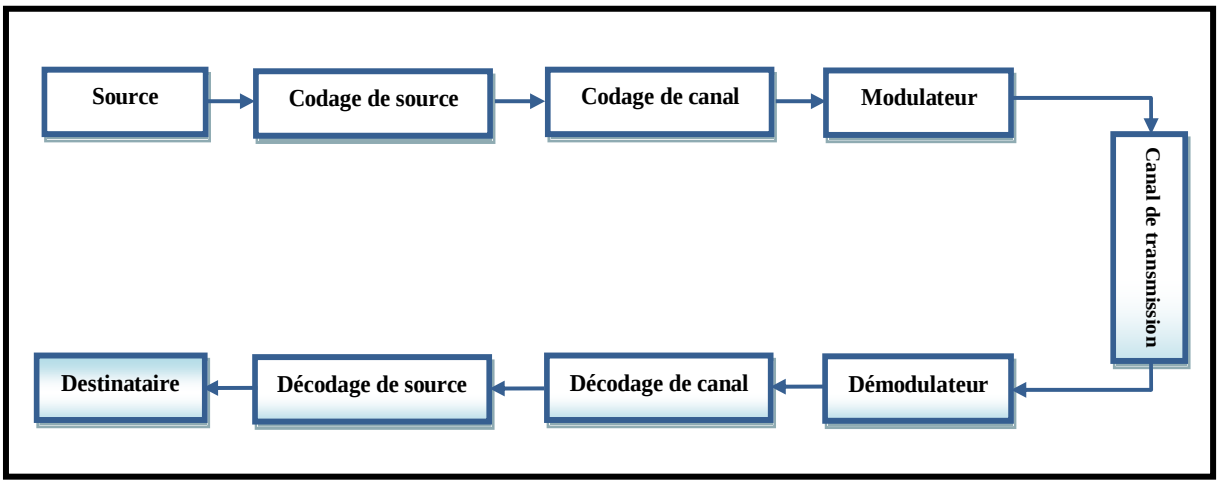


Fig. 1.1-Modèle de communication numérique

La figure (1.1) montre un modèle de système de communication numérique. Un message numérique provient de la source (cela aurait pu être obtenu à partir d'un signal analogique via un convertisseur analogique-numérique). Ces signaux numériques sont ensuite passés à travers un codeur de source. Le codeur de source élimine la redondance du système (fonctionne de la même manière que la compression des fichiers sur un ordinateur). Après codage de source, le signal est ensuite passé à travers le codeur de canal qui ajoute une redondance contrôlée au signal en vue de la protéger contre le bruit présent sur le canal de transmission. Le codage de canal n'est possible que si le débit de la source est inférieur à la capacité du canal de transmission (La probabilité d'erreur 'Pe' tend dans ce cas vers 0 d'après les travaux de Shannon), le signal est ensuite modulé et transmis sur le canal, La modulation a pour rôle d'adapter le spectre du signal au canal (milieu physique) sur lequel il sera émis. Le processus inverse se produit dans le récepteur. [1] [2]

Les trois caractéristiques principales permettant de comparer entre les différentes techniques de transmission sont les suivantes :

- La probabilité d'erreur 'Pe' par bit transmis permet d'évaluer la qualité d'un système de transmission. Elle est fonction de la technique de transmission utilisée, mais aussi du canal sur lequel le signal est transmis. En pratique, elle est estimée par le Taux d'Erreur par Bit 'TEB'.
- L'occupation spectrale du signal émis doit être connue pour utiliser efficacement la

Bande passante du canal de transmission. On est contraint d'utiliser de plus en plus des modulations à grande efficacité spectrale.

- La complexité du récepteur est le troisième aspect important d'un système de transmission. [1] [2]

Dans le cadre du présent projet de fin d'étude, nous nous sommes intéressées au décodage que nous allons entamer dans la section ci-après.

1.2 Décodage

Le décodage est l'opération inverse du codage, c'est le processus de transformation d'information d'un format à un autre. **[W2]**

En se basant sur le schéma simplifié du système de communication numérique de la figure (1.2), nous pouvons définir le principe général du décodage. [3]

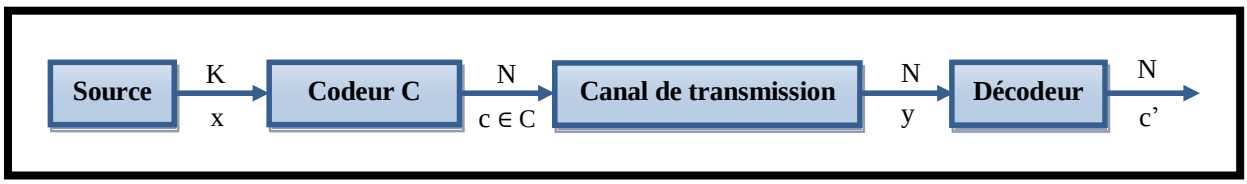


Fig. 1.2-Schéma simplifié pour le codage/décodage de canal

La source délivre la séquence d'information principale en vecteurs x de taille. Le codeur délivre le mot codé (codeword) c de taille N, qui est la version codé du mot x. le taux du code est défini par le rapport R=K/N. Le mot codé c est envoyé à travers le canal de transmission et le vecteur y est le mot reçu : version déformée de c.

Le but de décodage est de trouver le mot codé le plus probable d'être celui envoyé sur le canal de transmission, en se basant sur le canal de sortie et la connaissance du code.

De nombreux paramètres interviennent dans la performance de décodage. [3]

- L'algorithme de décodage

- les équations spécifiquement utilisées pour le calcul des données extrinsèques et leurs implantations,
- l'ordonnancement des équations de décodage
- la représentation des données (virgule flottante, virgule fixe) et les erreurs d'arrondis éventuelles qu'elle entraîne
- le degré de parallélisation du décodeur

Dans le domaine des systèmes de communication sur canal bruyant, une intense activité de recherche s'est développée depuis une cinquantaine d'années. Ces recherches se sont en particulier concentrées sur les techniques permettant de protéger les informations transmises vis-à-vis du bruit du canal et de corriger les erreurs de transmission. Deux grandes familles de codes correcteurs d'erreurs sont étudiées massivement depuis environ dix ans : les turbo-codes et les codes LDPC (Low Density Parity Check). [3]

La pièce maitresse de notre travail est le décodage LDPC, pour cela nous allons consacrer le chapitre suivant pour décrire ce type de décodage.

1.3 Circuits programmables

1.3.1 Introduction

Il existe une loi empirique, appelée loi de Moore, qui dit que la densité d'intégration dans les circuits intégrés numériques à base de silicium double tous les 18 à 24 mois. Cette loi s'est révélée remarquablement exacte jusqu'à ce jour. Durant les années 60, au début de l'ère des circuits intégrés numériques, les fonctions logiques telles que les portes, les registres, les compteurs et les ALU, étaient disponibles en circuit TTL. On parlait de composants SSI (Small Scale Integration) ou MSI (Medium Scale Integration) pour un tel niveau d'intégration. [4]

Dans les années 70, le nombre de transistors intégrés sur une puce de silicium augmentait régulièrement. Les fabricants mettaient sur le marché des composants LSI (Large Scale Integration) de plus en plus spécialisés. Par exemple, le circuit

74LS275 contenait 3multiplieurs de type Wallace. Ce genre de circuit n'était pas utilisable dans la majorité des applications. Cette spécialisation des boîtiers segmentait donc le marché des circuits intégrés et il devenait difficile de fabriquer des grandes séries. De plus, les coûts de fabrication et de conception augmentaient avec le nombre de transistors. Pour toutes ces raisons, les catalogues de composants logiques standards (série 74xx) se sont limités au niveau LSI. Pour tirer avantage des nouvelles structures VLSI (Very Large Scale Integration), les fabricants développèrent trois nouvelles familles : [4] [W3][W4]

- Les microprocesseurs et les mémoires RAM et ROM : les microprocesseurs et les circuits mémoires sont attrayants pour les fabricants. Composants de base pour les systèmes informatiques, ils sont produits en très grandes séries.

- Les circuits programmables sur site : n'importe quelle fonction logique, combinatoire ou séquentielle, avec un nombre fixe d'entrées et de sorties, peut être implantée dans ces circuits. A partir de cette simple idée, plusieurs variantes d'architecture ont été développées (PAL, EPLD, FPGA,...).

- Les ASIC programmés chez le fondeur : le circuit est conçu d'un point de vue logiciel par l'utilisateur, puis il est réalisé par le fondeur.

A l'heure actuelle, la majorité des circuits numériques est issue de ces trois familles.
Cependant, le catalogue standard (famille 74xx) est toujours utilisé.
Plus simplement, on peut distinguer deux catégories de circuits intégrés : les circuits standards et les circuits spécifiques à une application : [4] [W3] [W4]

- Les circuits standards se justifient pour de grandes quantités : microprocesseurs, contrôleurs, mémoires, ...

- Les circuits spécifiques sont destinés à réaliser une ou un ensemble de fonctions dans un système bien particulier.

La figure suivante représente une classification des circuits intégrés numériques :

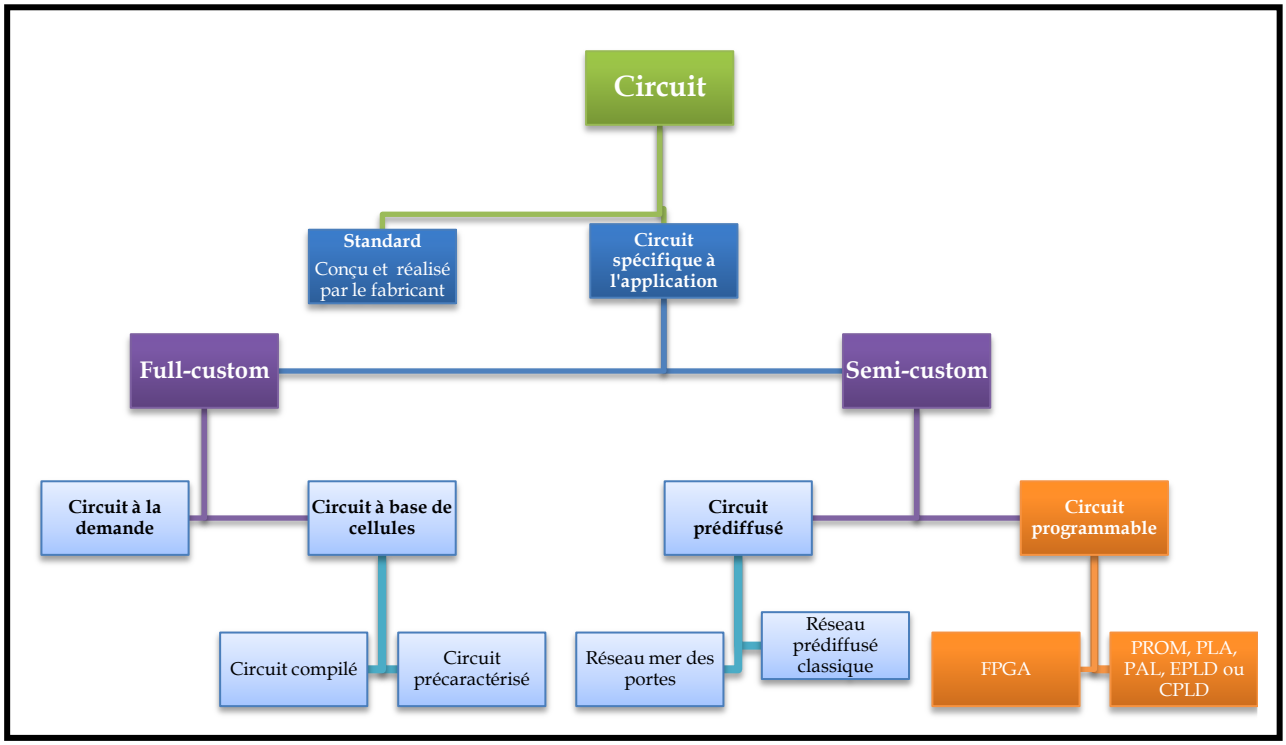


Fig. 1.3. Classification des circuits intégrés numériques

1.3.2 Description des FPGAs [2]

Les FPGAs (Field Programmable Gate Array, circuits pré-diffusés programmables en français ou réseau de portes programmable par l'utilisateur), apparus en 1985 sous l'initiative de l'entreprise Xilinx, ce sont des composants électroniques entièrement reconfigurables pouvant être reprogrammés afin d'accélérer notablement certaines phases de calculs. Ou tout simplement, c'est un ensemble de blocs logiques élémentaires que l'utilisateur peut interconnecter pour réaliser les fonctions logiques de son choix.

L'architecture générale d'un FPGA est la suivante : [5] [6]

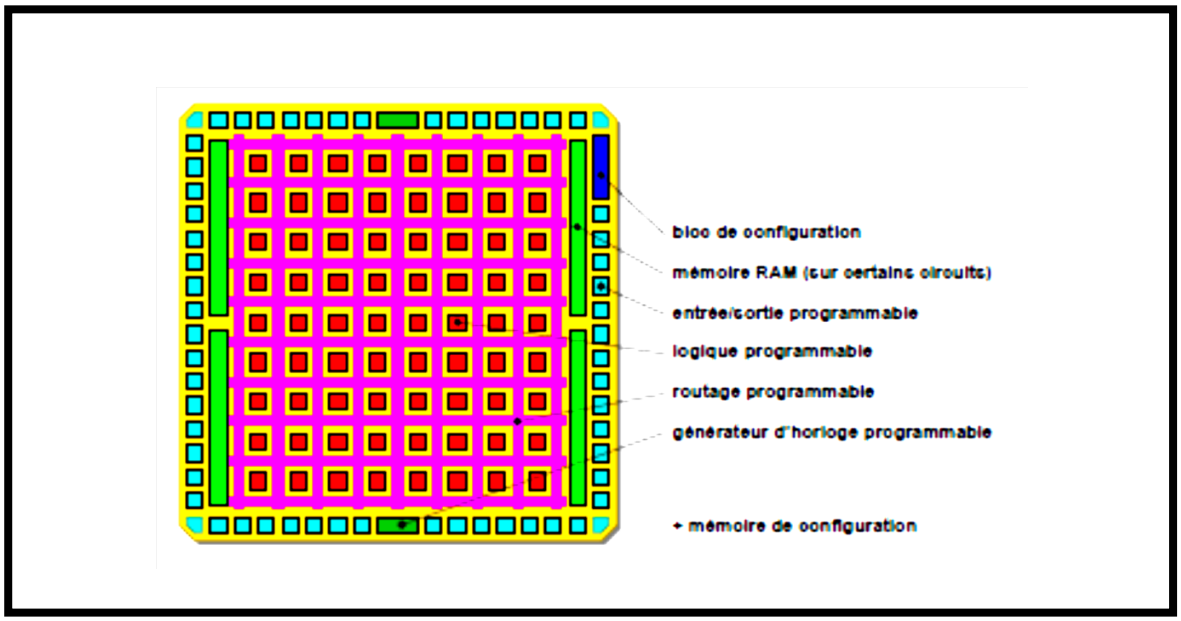


Fig. 1.4-Architecture générale d'un FPGA

- Les entrées/sorties peuvent être de diverses natures: signaux utilisateurs, signaux d'horloge (l'horloge définit la cadence des calculs), signaux de test, etc.
- Les blocs logiques programmables possèdent deux types principaux de composantes: les portes logiques configurables (pour faire des fonctions logiques assez simples à plusieurs entrées) et les bascules servant de mémoire élémentaire de l bit.
- Le routage programmable permet la connexion des blocs logiques entre eux.
- Le générateur d'horloge permet la distribution des horloges (plusieurs divisions de la fréquence d'horloge sont possibles) partout où elles sont nécessaires.

1.3.3 Processus de conception

Le processus de conception de systèmes avec les FPGAs passe par l'une des deux méthodes suivantes : [7] [8] [W5]

- **Une conception schématique :** cette approche est la plus laborieuse, la plus difficile pour les concepteurs. Elle consiste à dessiner au niveau logique

(portes, connexions) le circuit que l'on veut mettre en œuvre. Cette approche pose des problèmes lorsqu'il faut modifier ou réutiliser une conception existante; on note aussi des difficultés de migration d'une conception d'un constructeur à un autre.

- **Une conception abstraite :** elle consiste à utiliser des langages matériels HDL (Hardware Description Language) qui sont des langages de programmation de haut niveau permettant de décrire des circuits logiques. Les plus connus de ces HDL sont VHDL et Verilog. Dans ces langages comportementaux, le concepteur décrit, dans des fichiers textuels, les circuits en spécifiant leurs fonctions ou leurs comportements. Par la suite, le code de spécification subit plusieurs traitements (synthèse, simulation, compilation, placement/routage) afin de générer le circuit logique réel qui sera mis en œuvre dans le FPGA. Dans le cas de notre projet de recherche, nous utilisons le langage psC pour la conception de nos circuits. Grâce au compilateur associé à l'environnement de développement du langage, le code est traduit en VHDL; enfin, à l'aide d'outils de la compagnie Altera, le code VHDL passe par des phases de synthèse, compilation, placement et routage avant qu'un fichier binaire soit produit pour pouvoir être exécuté sur FPGA.

1.3.4 Avantage des circuits FPGAs

Les six principaux atouts de la technologie FPGA sont : [9] [W3] [W4]

1. Performances
2. Souplesse de programmation
3. Temps de mise sur le marché
4. Coût
5. Fiabilité
6. Maintenance à long terme

1. **Performances** - Comme ils tirent parti du parallélisme[1] matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application.

2. **Souplesse de programmation** qui permet l'emploi conjoint d'outils de schématique aussi bien que l'exploitation d'un langage de haut niveau tel VHDL. Ce qui permet de multiplier les essais, d'optimiser de diverses manières l'architecture développée, de vérifier à divers niveaux de simulation la fonctionnalité de cette architecture.

3. **Temps de mise sur le marché** - Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, vous pouvez tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé3. Par la suite, vous pourrez apporter les éventuelles modifications nécessaires à votre FPGA, en quelques heures au lieu de quelques semaines. Le matériel « sur étagère » actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cœurs de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux.

4. **Coût** - Les coûts d'ingénierie non récurrents (NRE) des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. La plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les

[1] Le **parallélisme** consiste à implémenter des architectures d'électronique numérique permettant de traiter des informations de manière simultanée

besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la re-conception d'un ASIC. Pour cela, il semble que de plus en plus fréquemment les concepteurs de circuits ASIC préfèrent passer par l'étape intermédiaire d'un FPGA ce qui est moins risqué économiquement, puis une fois que le modèle FPGA est au point, il est alors relativement aisé de le retranscrire dans une architecture de type pré-diffusé ou pré-caractérisés. Ce que tous les fondeurs de silicium savent effectivement faire pour en faire un circuit réellement personnalisé et confidentiel

5. **Fiabilité** - Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche.

6. **Reconfiguration dynamique & Maintenance à long terme**, les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la re-conception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, vous pouvez y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit. ILs permettent donc une reconfiguration partielle ou totale d'un circuit ce qui permet

d'une part, une meilleure exploitation du composant, une réduction de surface de silicium employé et donc du coût, et d'autre part, une évolutivité assurant la possibilité de couvrir à terme des besoins nouveaux sans nécessairement repenser l'architecture dans sa totalité. L'un des points forts de cette reconfiguration dynamique est effectivement de permettre de reconfigurer en temps réel en quelques microsecondes tout ou partie du circuit, c'est à dire de permettre de modifier la fonctionnalité d'un circuit en temps quasi réel. On dispose donc quasiment de la souplesse d'un système informatique qui peut exploiter successivement des programmes différents, mais avec la différence fondamentale qu'ici il ne s'agit pas de logiciel mais de configuration matérielle, ce qui est infiniment plus puissant.

Le progrès de ces technologies permet aujourd'hui de faire des composants toujours plus rapides et à plus haute intégration, ouvrant ainsi la voie au développement d'applications importantes (traitement du signal, codage/décodage, cryptage/décryptage, reconnaissance des formes, etc.). [9]

Notons enfin que ces circuits n'ont pas vocation à concurrencer les super calculateurs, mais plutôt à offrir une alternative en fonction de critères comme l'encombrement, les performances et le prix, et sont de ce fait bien adaptés à des applications de qualité dans le domaine des systèmes ambulatoires[2] ou nomades[3]. [W4]

1.4 Langages de description « VHDL »

1.4.1 Définition

[10] [11] [12]

Le VHDL est un langage de description de matériel qui est utilisé pour la spécification (description du fonctionnement), la simulation et la preuve formelle d'équivalence de circuits. Ensuite il a aussi été utilisé pour la synthèse automatique.

[2] Le terme ambulatoire est utilisé pour caractériser un acte comme par exemple une intervention chirurgicale voir une anesthésie se faisant chez un patient qui ne nécessite pas une hospitalisation au décours de l'intervention.
[3] Se dit d'un système qui peut être transporté facilement, tout en restant à tout moment disponible et fonctionnel

L'abréviation VHDL signifie VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (langage de description de matériel pour circuits à très haute vitesse d'intégration).

1.4.2 Historique

[11] [13] [12] [14]

En 1981, le Département de la Défense (DoD) des Etats-Unis d' Amérique a initié puis dirigé le projet "Very High Speed Integrated Circuit" (VHSIC). Ce projet avait pour but de formaliser la description des circuits intégrés développés pour le DoD dans un langage commun. L'intérêt premier était de définir, au travers du langage, une spécification complète et non ambiguë du circuit à développer indépendante de la technologie employée et des outils de CAO[4].

Il a ainsi mandaté des sociétés pour établir un langage. Parmi les langages proposés, le DOD a retenu le langage VHDL qui fut ensuite normalisé par IEEE. Le langage ADA est très proche, car celui-ci a servi de base pour l'établissement du langage VHDL.

La standardisation du VHDL s'effectuera jusqu'en 1987, époque à laquelle elle sera normalisée par l'IEEE (Institute of Electical and Electronics Engineers). Cette première normalisation a comme objectif:

- La spécification par la description de circuits et de systèmes.
- La simulation afin de vérifier la fonctionnalité du système.
- La conception afin de tester une fonctionnalité identique mais décrite avec des solutions d'implémentations de différents niveaux d'abstraction.

En 1993, une nouvelle normalisation par l'IEEE du VHDL a permis d'étendre le domaine d'utilisation du VHDL vers:

- La synthèse automatique de circuit à partir des descriptions.
- La vérification des contraintes temporelles.

[4] Conception assistée par ordinateur

• La preuve formelle d'équivalence de circuits.

En 1999, IEEE a édité la norme 1076.6 qui définit le sous-ensemble synthétisable. Cette norme définit l'ensemble des syntaxes autorisées en synthèse.

Nous pouvons affirmer que depuis l'année 2000 l'ensemble synthétisable du langage VHDL est de mieux en mieux défini. La majorité des outils de synthèse sont compatibles avec cette norme. Un véritable standard est donc établi pour la synthèse automatique avec le langage VHDL. [11]

La vivacité des groupes de travail constitués de concepteurs en électronique et de développeurs de logiciels de CAO a assuré le succès de ce langage. Aujourd'hui, il est proposé par tous les vendeurs de composants logiques programmables pour décrire la logique. Il a remplacé la multitude de langages propriétaires HDL (citons MINC, ABEL, PALASM), et ainsi facilité la réutilisation et l'échange entre concepteurs de descriptions logiques éprouvées. [14]

1.4.3 Pourquoi le langage VHDL ?

L'électronicien a toujours utilisé des outils de description pour représenter des structures logiques ou analogiques. Le schéma structurel que l'on utilise depuis si longtemps et si souvent n'est en fait qu'un outil de description graphique. Aujourd'hui, l'électronique numérique est de plus en plus présente et tend bien souvent à remplacer les structures analogiques utilisées jusqu'à présent **[16]**. Ainsi, l'ampleur des fonctions numériques à réaliser nous impose l'utilisation d'un autre outil de description. Il est en effet plus aisé de décrire un compteur ou un additionneur 64 bits en utilisant l'outil de description VHDL plutôt qu'un schéma [14] [15] [17].

Le deuxième point fort du VHDL est d'être "un langage de description de haut niveau". D'autres types de langage de description, comme l'ABEL par exemple, ne possèdent pas cette appellation. En fait, un langage est dit de haut niveau lorsqu'il fait la plus possible abstraction de l'objet auquel ou pour lequel il est écrit. Dans le

cas du langage VHDL, il n'est jamais fait référence au composant ou à la structure pour lesquels on l'utilise. Ainsi, il apparaît deux notions très importantes : [18] [19] [20] [21]

- **Portabilité des descriptions VHDL**, c'est-à-dire, possibilité de cibler une description VHDL dans le composant ou la structure que l'on souhaite en utilisant l'outil que l'on veut (en supposant, bien sûr, que la description en question puisse s'intégrer dans le composant choisi et que l'outil utilisé possède une entrée VHDL) ;
- **Conception de haut niveau**, c'est-à-dire qui ne suit plus la démarche descendante habituelle (du cahier des charges jusqu'à la réalisation et le calcul des structures finales) mais qui se "limite" à une description comportementale directement issue des spécifications techniques du produit que l'on souhaite obtenir.

Le VHDL est un langage normalisé, cela lui assure une pérennité. Il est indépendant d'un fournisseur d'outils. Il est devenu un standard reconnu par tous les vendeurs d'outils EDA. Cela permet aux industriels d'investir sur un outil qui n'est pas qu'une mode éphémère, c'est un produit commercialement inévitable. Techniquement, il est incontournable car c'est un langage puissant, moderne et qui permet une excellente lisibilité, une haute modularité et une meilleure productivité des descriptions. Il permet de mettre en œuvre les nouvelles méthodes de conception. [11] [24] [25]

En effet, ce langage s'adresse à des concepteurs de systèmes électroniques, qui n'ont pas forcément de grandes connaissances en langages de programmation.

Pour développer une application de logique programmable, il faudra suivre la démarche ci-dessous : [11] [26] [27]

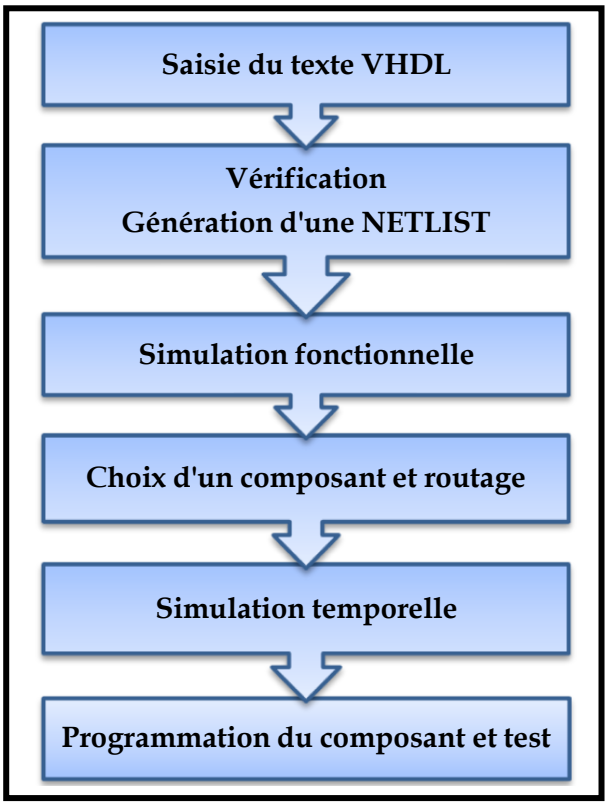


Fig. 1.5-Etapes de développement d'une application de logique programmable

1.4.4 Les limites actuelles :

La norme qui définit la syntaxe et les possibilités offertes par le langage de description VHDL est très ouverte. Il est donc possible de créer une description VHDL de systèmes numériques non réalisable, tout au moins, dans l'état actuel des choses. Il est par exemple possible de spécifier les temps de propagations et de transitions des signaux d'une fonction logique, c'est-à-dire créer une description VHDL du système que l'on souhaite obtenir en imposant des temps précis de propagation et de transition. Or les outils actuels de synthèses logiques sont incapables de réaliser une fonction avec de telles contraintes. Seuls des modèles théoriques de simulations peuvent être créés en utilisant toutes les possibilités du langage. [11] [14] [15]

Après avoir présenté cette notions générales on peut maintenant se mieux callé par rapport à notre travail dans les chapitres suivants.

Chapitre 2 Les codes LDPC

Avant de passer au décodage LDPC, dans ce Chapitre nous allons entamer les codes LDPC, leur principe et leur encodage. Comme pour tout code, un décodeur doit être associé, nous présenterons donc quelques algorithmes et techniques de décodage des codes LDPC sous forme de comparaison élaboré dans [34]. Enfin nous exposons un petit état de l'art des travaux réalisé dans le cadre de décodage des codes LDPC.

2.1 Historique

Les codes LDPC (Low Density Parity Check) ont été inventés par **Robert E. Gallager** en 1962. Ce sont une classe de codes à blocs linéaire qui s'approche de la limite de la capacité du canal de Shannon. Ces codes sont basés sur des matrices de contrôle de parité pseudo aléatoires de faible densité. [2][28]
Du fait de leur complexité d'encodage, de décodage et des moyens matériels de l'époque notamment lors de l'utilisation de long code, ces codes n'ont pas suscité suffisamment d'intérêt au sein de la communauté de la théorie du codage. [1][2][28][29]

Cet oubli durera jusqu'à l'introduction des turbo-codes et leur algorithme de décodage itératif associé en 1993 par C.BERROU et all. La performance remarquable qu'a dévoilé les turbo-codes a beaucoup attiré l'attention vers les techniques itératives. [1][28][29]
Ainsi, en 1995, D.J.C MacKay et R.M Neal, ont redécouvert les codes LDPC. Ils ont montré tout d'abord que l'algorithme de décodage développé par Gallager, peut également se décrire comme l'algorithme de propagation de croyance (Belief Propagation (BP)) de Pearl. Par la suite Luby, a introduit les codes LDPC irréguliers caractérisés par une matrice de contrôle de parité pour laquelle la distribution des nombres d'éléments non nuls par ligne et/ou colonne n'est pas uniforme. [1][29]

La relative simplicité de l'analyse théorique des codes (du moins par rapport à la complexité d'analyse des Turbo-codes) a motivé la communauté scientifique pour suivre les travaux dans le domaine des codes LDPC. Cette technologie de codage est

restée relativement ouverte en termes de propriété intellectuelle, du fait que l'invention des codes LDPC remonte à 1962, ce qui a motivé les industriels à explorer cette voie. Ainsi en 2004, un code LDPC a été pour la première fois normalisé dans un contexte de diffusion par satellite : DVB-S2. Plus récemment, les codes LDPC ont été introduits dans les standards IEEE 802.16e (Wimax mobile) et IEEE 802.11n (Wifi). Ces différents facteurs, ainsi que les nouveaux problèmes posés, ont contribué à accroître la popularité des codes LDPC dans le domaine industriel et scientifique. [1]

2.2 Description des codes LDPC

2.2.1 Codes à matrice creuse

Les codes LDPC, pour "Low Density Parity Check" sont des codes correcteurs d'erreurs. Ils sont appelés ainsi car ils ont la particularité d'avoir une matrice de parité possédant une faible densité. Dans le cas binaire, une matrice de parité d'un code LDPC possède une majorité d'entrées nulles et une minorité d'entrées égales à '1'. Grâce à cette faible densité de la matrice de parité, il est possible d'utiliser une classe particulière d'algorithmes de décodage reposant sur le principe de propagation de croyance ("Belief propagation"). Ces algorithmes ont l'avantage de posséder une complexité linéaire en la longueur du code (ces algorithmes seront détaillés dans le chapitre suivant). Ainsi les codes LDPC peuvent protéger efficacement des messages de grande longueur, ce qui peut se révéler essentiel en pratique. [30] [32]

Tout Comme pour les autres codes linéaires, la capacité de correction des codes LDPC dépend du poids des mots de codes et en particulier de la distance minimale. [32]

2.2.2 Représentation des codes LDPC

Il existe deux représentations usuelles des codes LDPC : le graphe biparti et la matrice de parité. Ces deux représentations sont totalement équivalentes, et l'on

choisira l'une ou l'autre en fonction du niveau de la simplification qu'elle apporte pour la résolution du problème donné. [32]

a Matrice de parité

Les codes LDPC, inventés par Gallager, sont des codes linéaires définis par une matrice de parité creuse (de faible densité) H (M×N) sur F2. Lorsqu'il a introduit les codes LDPC, Gallager a spécifié que cette matrice devait avoir un faible nombre d'éléments non-nuls (Le nombre de "1" dans cette matrice est faible devant le nombre de "0") d'où intervient la notion de faible densité. [28]

Aussi appelée matrice d'adjacence, la matrice de parité définit **M** contraintes de parité sur **N** symboles du mot de code, le nombre de bit d'information est donc donné par: [29]

$$K = N - M \quad \text{..... (2.1)}$$

Un mot de code "**c**" satisfait: [30] [1]

$$Hc^T = S = 0 \quad \text{..... (2.2)}$$

Où S : Syndrome checking

La figure (2.1) montre une matrice de parité H de taille (4×8).

Dans une telle matrice, chaque colonne correspond à un nœud dit de variable (variable node) et chaque ligne à un nœud de Parité (Check node). L'élément (i ; j) de la matrice est non nul si et seulement s'il existe une arête reliant le nœud d'indice j des nœuds de variable au nœud d'indice i des nœuds de parité.

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

Fig. 2.1. Matrice de parité (4×8), 4 nœuds de parité, 8 nœuds de variable, 4 bits de redondance et de rendement= ½

Le rendement d'un code LDPC est donnée par : [33]

$$R = \frac{M}{N} \qquad \ldots\ldots (2.3)$$

Comme pour tout code linéaire, la matrice de parité définit totalement le code. Plus précisément la matrice de parité définit des relations entre les symboles des mots du code, sous la forme d'un système linéaire. La matrice étant creuse, chaque relation va concerner un petit nombre de symboles.
Rappelons que pour un code linéaire donné, il peut exister plusieurs matrices de parité. [32]

Dans le cas d'un code binaire, si on considère ce système équation par équation on obtient la propriété suivante : pour un mot de code donné, pour chaque ligne de la matrice de parité, la somme des symboles (source et parités) présents dans celle-ci est nulle. [32] [2]

Au delà de la faible densité de sa matrice de parité, les performances des codes LDPC obtenus seront largement tributaires du nombre d'entrées non nulles par ligne et par colonne. On appelle cette quantité, le degré d'une ligne (resp. colonne). Quand cette quantité est constante, on appelle le code ainsi défini un code LDPC régulier. La figure (2.2) montre le système linéaire correspondant de la matrice H (4×8) présentée auparavant, où les $\{ xi \}$, $0 \leq i \leq 8$ sont les variables représentant les symboles. [32] [2]

$$\begin{aligned} x_0 \oplus x_4 &= 0 \\ x_1 \oplus x_4 \oplus x_5 &= 0 \\ x_2 \oplus x_5 \oplus x_6 &= 0 \\ x_3 \oplus x_6 \oplus x_7 &= 0 \end{aligned}$$

Fig. 2.2. Système linéaire défini par la matrice H

b Graphe de Tanner

La deuxième représentation des codes LDPC est un graphe biparti (voir figure II.3), aussi appelé graphe de Tanner. [32]

Les graphes de Tanner sont des moyens picturaux de représentation de la matrice de contrôle de parité des codes en bloc. [2]

Un graphe est dit biparti s'il existe deux ensembles de nœuds (sommets) **U** et **V** et un ensemble d'arêtes, tels que chaque arête relie un nœud de **U** à un nœud de **V**. La figure suivante représente un exemple de graphe de Tanner. [32] [2]

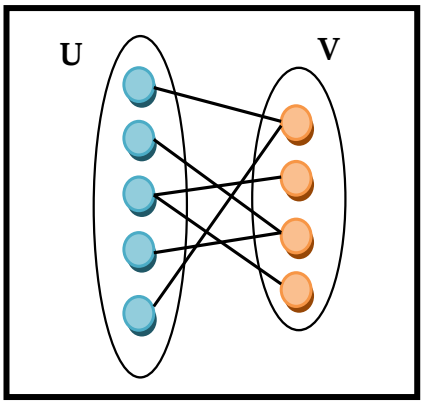


Fig. 2.3. Graphe bipartie

Nous nous sommes intéressées par ces graphes, car ils peuvent représenter les matrices de contrôle de parité **H** du code LDPC. Les lignes de cette matrice sont représentées par des nœuds de parité, alors que les colonnes sont représentées par des nœuds de variable. S'il ya un 1 dans une position donnée (i, j), où i est l'indice de ligne et j est l'indice de colonne (cela veut dire que le chek node **i** est connecté au variable node **j**), une arête est utilisé pour montrer cette connexion sur le graphe de Tanner. La figure (2.4) montre le graphe de Tanner correspondant à la matrice de parité présentée dans la figure (2.1).

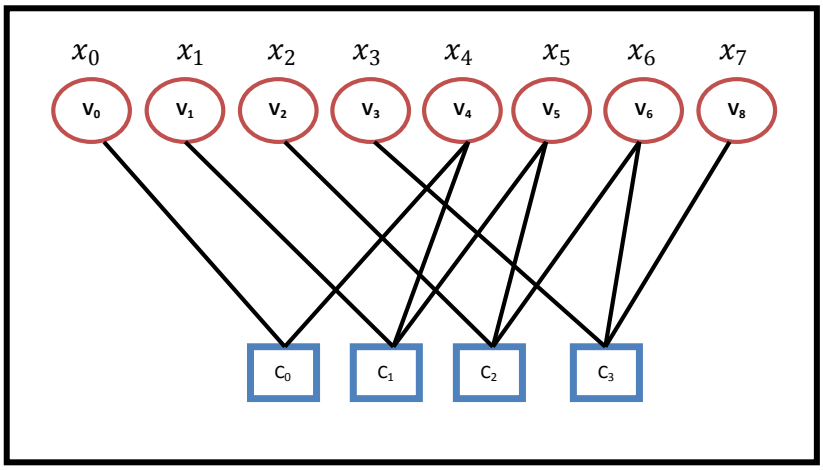


Fig. 2.4- Graphe de Tanner d'un code LDPC

Un code LDPC peut être défini à partir d'un graphe biparti dont l'ensemble des nœuds sous forme de cercle, noté V_i (nœuds variable), représente les symboles du mot de code, et l'ensemble des nœuds sous forme de carré, noté C_i (nœuds de contrainte), représente les contraintes. Une suite de symboles constitue alors un mot de code valide si et seulement si pour chaque nœud de contrainte, la somme des symboles correspondants aux nœuds variables adjacents est nulle.

La représentation d'un code LDPC sous la forme d'un graphe de Tanner est particulièrement utile lorsque l'on veut se représenter le mécanisme de passage de messages ("message passing") qui intervient dans les décodeurs itératifs. De plus, cette représentation permet également de faire appel aux puissants outils de la théorie des graphes pour étudier et concevoir les codes LDPC. [32]

2.2.3 Code LDPC régulier et irrégulier

Une famille de codes LDPC peut se décrire en termes de taux de connexions des équations de parité. Quand le nombre de "1" par ligne et le nombre de "1" par colonne est constant, on parle de code LDPC régulier. Par conséquent chaque bit du mot de code participe à un même nombre d'équations de parité. De même, chacune des équations de parité utilise le même nombre de bits. [1]

Une classe plus générale des codes LDPC a ensuite été introduite par Luby et all, il s'agit des codes LDPC irréguliers. [34]

Ce sont des codes définis par des matrices de contrôle de parité où le nombre de "1" par ligne et/ou par colonne n'est pas constant. Pour décrire ces codes, il est d'usage de spécifier l'irrégularité d'un code à travers deux polynômes λ (x) et ρ(x) : [1]

$$\lambda(x) = \sum_{i\geq 1} \lambda_i \; x^{i-1} \quad \text{..... (2.4)}$$

$$\rho(x) = \sum_{i\geq 2} \rho_i \; x^{i-1} \quad \text{..... (2.5)}$$

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

On peut relier le profil d'irrégularité du code au rendement de codage de la façon suivante : [1]

$$R \geq 1 - \frac{\sum_{i\geq 1} \frac{\lambda_i}{i}}{\sum_{i\geq 2} \frac{\rho_i}{i}} \quad \text{..... (2.6)}$$

Il y a égalité quand la matrice de contrôle de parité est de rang plein.

On peut aussi caractériser l'irrégularité d'un code à travers deux autres polynômes, $\lambda'(x)$ et $\rho'(x)$ qui caractérise la proportion de nœuds de même degré (plutôt que la proportion de branches) de la manière suivante : [1]

$$\lambda'^{(x)} = \sum_{i\geq 1} \lambda'_i \; x^{i-1} \quad \text{.....(2.7)}$$

$$\rho'(x) = \sum_{i\geq 2} \rho'_i \; x^{i-1} \quad \text{.....(2.8)}$$

Les coefficients λ'_i et ρ'_i de ces polynômes représentent la proportion des nœuds de données et de contrôles de degré i. Les deux représentations polynomiales sont reliées par :

$$\lambda'^{(x)} = \frac{1}{i} \frac{\lambda_i}{\sum_{i\geq 1}\frac{\lambda_i}{i}} \ldots\ldots (2.9)$$

$$\rho'^{(x)} = \frac{1}{i} \frac{\rho_i}{\sum_{i\geq 2}\frac{\rho_i}{i}} \ldots\ldots (2.10)$$

Le code représenté par la figure (2.4) a une distribution des degrés égale à :

$$\lambda'^{(x)} = \frac{5}{8} + \frac{3}{8}\, x$$

$$\rho'^{(x)} = \frac{1}{4}x + \frac{3}{4}\, x^2$$

qui peut aussi être exprimée de la façon suivante :

$$\lambda^{(x)} = \frac{5}{11} + \frac{6}{11}\, x$$

$$\rho^{(x)} = \frac{2}{11}x + \frac{9}{11}\, x^2$$

2.3 Encodage des codes LDPC

Le problème d'encodage des codes LDPC est de déterminer le mot de code "**c**" à partir du vecteur **b** contenant les bits d'information et de la matrice de contrôle de parité **H** sous la contraint définie par l'équation **(2.2)**. Si **G** est la matrice génératrice du code LDPC à encoder, nous avons : [34]

$$\boxed{c = G\, b} \quad \ldots\ldots (2.11)$$

Notons que si l'on introduit cette équation dans **(2.2)**, nous obtenons la relation suivante :

$$\boxed{H\, G = 0} \quad \ldots\ldots (2.12)$$

Pour effectuer l'encodage avec un code LDPC, on peut utiliser la matrice génératrice comme pour tout code linéaire : une multiplication du vecteur source par la matrice génératrice nous donne le vecteur correspondant au mot de code. [32]

Dans la majorité des cas, la matrice génératrice associée à un code dont la matrice de contrôle de parité de faible densité est dense (dans le cas de la manipulation de longs mots). La complexité d'encodage associé est alors importante. [1]
Cette approche présente cependant plusieurs inconvénients : [32]

- la matrice génératrice doit être connue. Si ce n'est pas le cas il faut la calculer par une élimination de Gauss sur la matrice de parité, mais c'est une opération coûteuse. Cette opération peut parfois être effectuée en avance et n'a donc pas d'impact sur la complexité de l'étape d'encodage. Cependant lorsque le code est généré à la volée, il est impossible d'effectuer ce pré-calcul, et la matrice génératrice devra être calculée avant de pouvoir faire l'encodage.

- la multiplication par la matrice génératrice (qui est dense à l'inverse de la matrice de parité) est coûteuse en nombre d'opérations (complexité en O((N-k)N)).

Pour réduire cette complexité, des approches que l'on peut classer dans deux grandes familles ont été développées. La première consiste à post-traiter la matrice de contrôle de parité de façon à introduire une forme facilement encodable. La seconde est basée sur la construction d'une matrice de contrôlé de parité contrainte, construite à l'origine pour faciliter l'encodage.
En ce qui concerne la première approche, les auteurs préconisent une transformation de la matrice de contrôle de parité par combinaisons linéaires de lignes et de colonnes en une autre matrice de contrôle de parité de forme semi-triangulaire.
La complexité d'encodage dépend alors d'un paramètre caractérisant l'écart entre la matrice semi-triangulaire et la matrice triangulaire. Une fois les bits intervenant dans la forme semi-triangulaire obtenue, les autres bits de redondance sont obtenus par substitution. [1]

La seconde méthode consiste en la définition d'une structure de code contrainte. Une première construction très largement répandue, consiste à construire une matrice de contrôle de parité définie par : [1]

$$\mathbf{H} = [\mathbf{H_s}\ \mathbf{H_p}] \text{ …. (II.13)}$$

Le mot de code **c** est alors divisé en un mot d'information b et un mot de redondance $\boldsymbol{p}$. La relation de parité s'écrit alors :

$$\mathbf{H}\ \mathbf{c^t} = \mathbf{0^t} \text{ …. (II.14)}$$

$$[\mathbf{H_s}\ \mathbf{H_p}]\begin{bmatrix}\mathbf{b^t}\\ \boldsymbol{p^t}\end{bmatrix} \text{ …. (II.15)}$$

$$\mathbf{H_p}\ \boldsymbol{p^t} = \mathbf{H_s}\ \boldsymbol{b^t} \text{ ….. (II.16)}$$

Pour simplifier les notations, nous appellerons vecteur $\boldsymbol{v}$ de projection le vecteur défini par : [1]

$$\boldsymbol{v^t} = \mathbf{H_s}\ \boldsymbol{b^t} \text{ ….. (II.17)}$$

Il en résulte donc que l'ensemble des bits de parité peut se déduire de la façon suivante : [1]

$$\boldsymbol{p^t} = \boldsymbol{H_p}^{-1}\ \boldsymbol{v^t} \text{ ….. (II.18)}$$

Cette relation montre que la première contrainte sur le code est l'existence de la matrice inverse $\boldsymbol{H_p}^{-1}$. La matrice $\boldsymbol{H_p}$ peut être de forme triangulaire permettant par simple substitution le calcul des bits de redondance. [1]

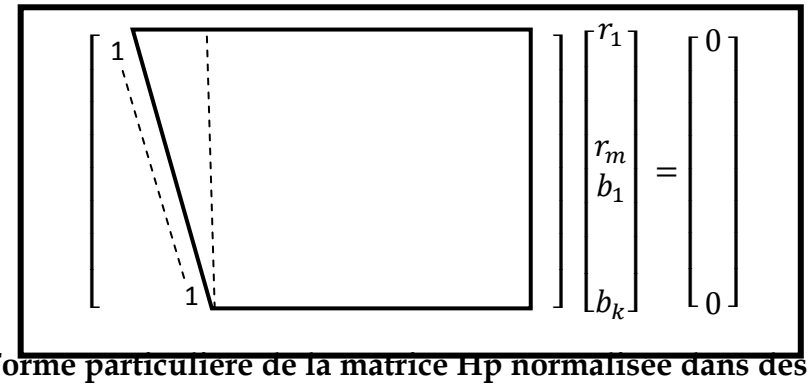


Fig. 2.5- Forme particulière de la matrice Hp normalisée dans des standards

Mackay et all ont proposé une approche qui consiste à forcer la matrice de vérification de parité à être systématique (soit directement encodable) c.à.d. que l'ensemble des codes est restreint par le fait que la matrice H soit de forme semi-triangulaire tel que décrit par la figure (2.6). Etant donnée que la matrice de parité est creuse, cette restriction garantie une complexité d'encodage linéaire [34]. Et ce modèle de matrice de parité que nous avons choisi pour notre étude.

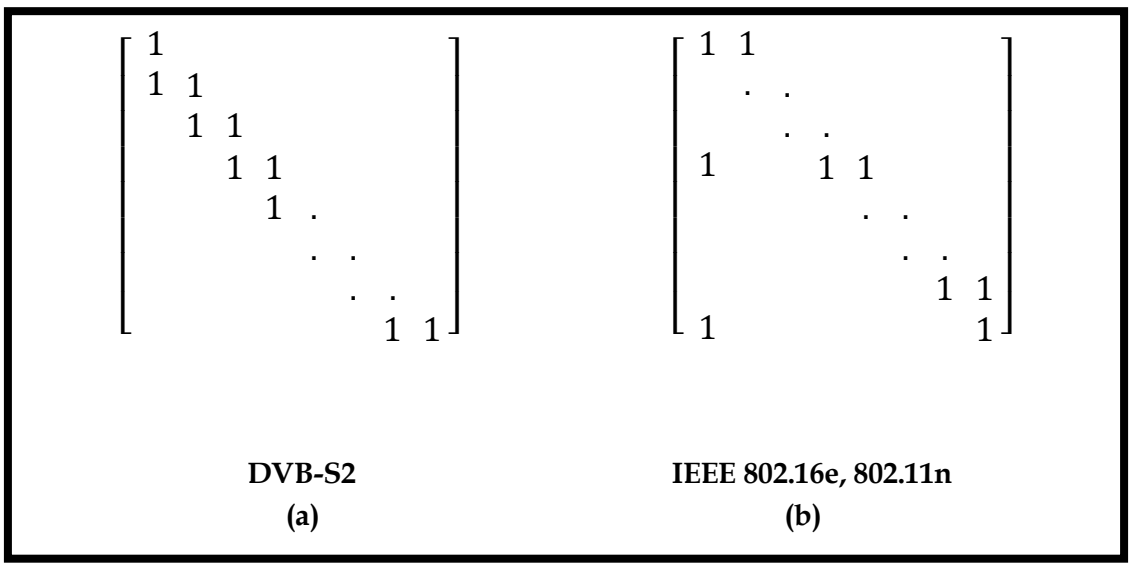


Fig. 2.6- Encodage des codes LDPC (Modèle de Mackay)

A chaque code est associé un décodeur. Pour le décodage des codes LDPC présenté dans le présent chapitre, plein d'algorithmes de décodage existent. La littérature a connu de nombreux travaux de décodage des codes LDPC, dans la section suivante

nous allons faire une présentation non exhaustive des techniques de décodage LDPC les plus utilisés, en se basant sur une comparaison qui a été faite dans [34]

2.4 Quelques algorithmes de décodage des codes LDPC

Plusieurs algorithmes de décodage des codes LDPC existent dans la littérature, nous pouvons citer : [30]

- Sum-Product (SP)
- Min-Sum (MS)
- Offset-MS (OMS)
- Normalized-MS (NMS)
- Self-Corrected-MS (SCMS)
- Bahl, Cocke, Jelinek and Raviv algorithm (BCJR)

Après une large étude élaboré dans cet ordre d'idée, nous pouvons montrer la comparaison entre les différent algorithmes cites ci-dessus.
L'algorithme self-corrected-MS (SCMS) présente un bon compromis entre la complexité et la performance. La Figure (II.7) montre la performance (taux d'erreur ou bit-error-rate BER) de chaque algorithme pour le décodage des codes LDPC dans le standard IEEE 802.11n à travers le canal AWGN avec une modulation QPSK; la taille des mots utilisée est 1944 avec un taux de codage 1/2 et 60 itérations maximum.

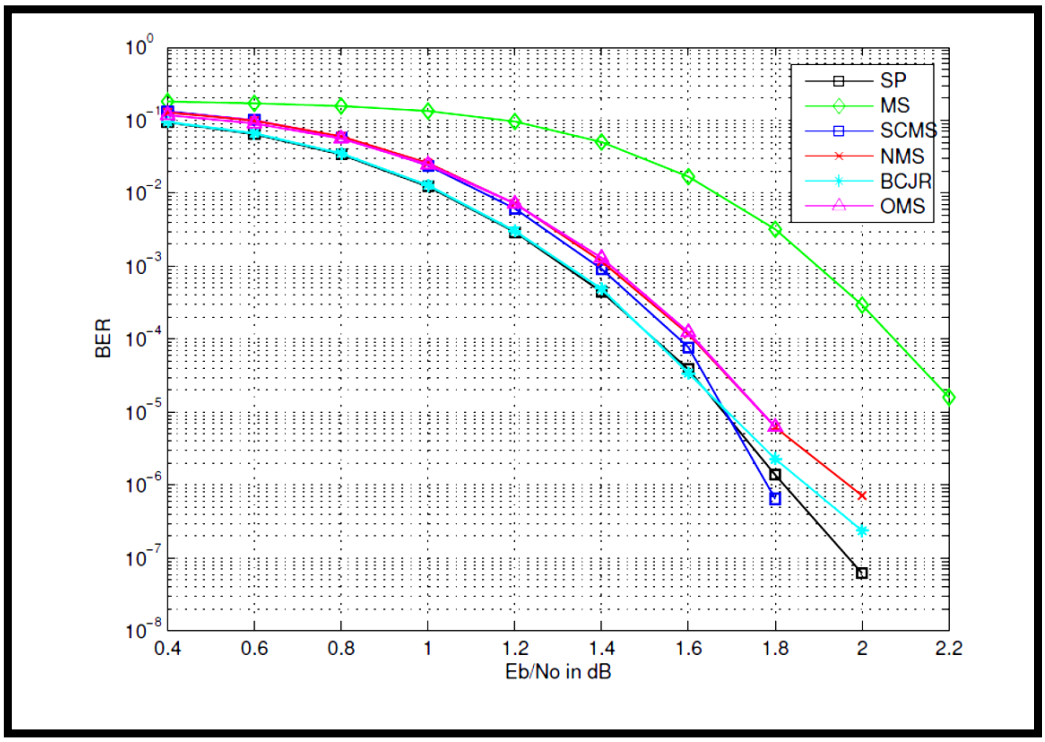


Fig. 2.7- Taux d'erreur des algorithmes

La Figure (2.8) montre les résultats d'une implémentation d'intégration à très grande échelle (VLSI) des différents algorithmes de faible complexité dans une technologie CMOS de 65nm. L'implémentation correspond à un processeur série avec une quantification de message sur 6-bits. Les résultats incluent la surface et la consommation moyenne d'´energie par itération.

L'algorithme avec la plus grande surface et consommation d'énergie est le BCJR. Cet algorithme présente des calculs de haute complexité.

D'autre part, l'algorithme MS présente la surface la plus faible en raison de calculs de faible complexité. Toutefois, l'algorithme SCMS présente une réduction de la consommation d'´energie en raison d'une réduction du nombre de messages échangés à chaque itération. Effectivement l'algorithme SCMS présente un compromis intéressant entre consommation d'énergie, surface et performance. Nous avons propos´e de geler le processeur SCMS de parité dès que deux, ou plus, de ses entrées sont effacées. Un gain de consommation moyenne de 10% est réalisé.

Nous examinons le taux de convergence des algorithmes. Ce taux est affecté par le nombre de messages par itération et leur précision. La Figure (2.9) a montre le taux

de convergence pour les algorithmes de la Figure (2.7). L'algorithme SCMS a une vitesse de convergence lente en raison de la réduction du nombre de messages, mais en général cet algorithme a la consommation d'énergie la plus faible. La Figure (2.9.b) montre les données de la Figure (2.8) (l'´energie) combinées avec les taux de convergence.

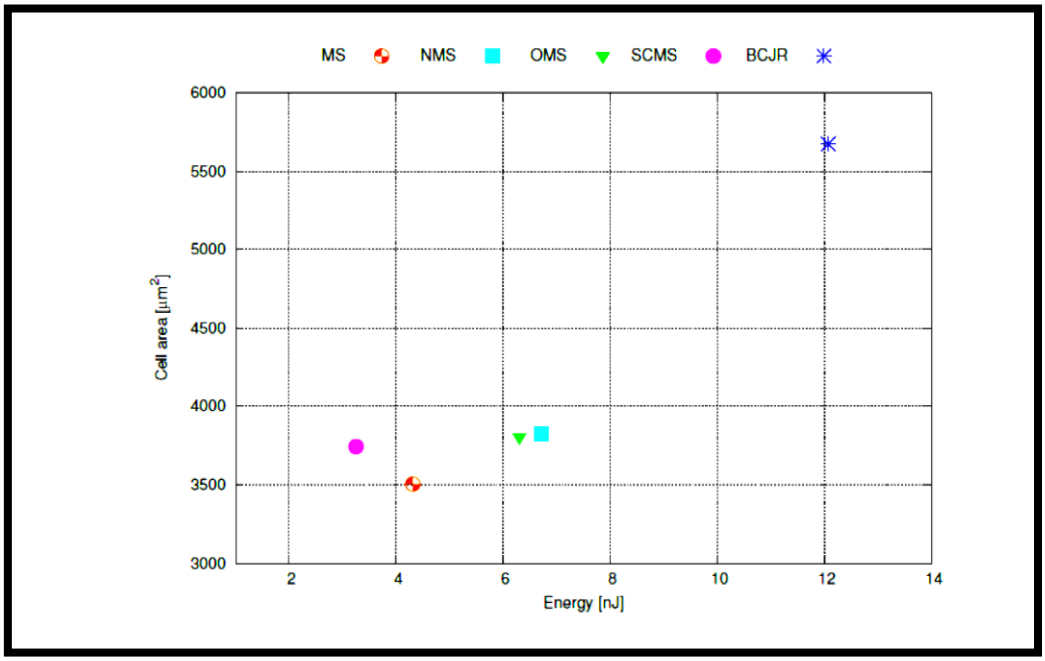
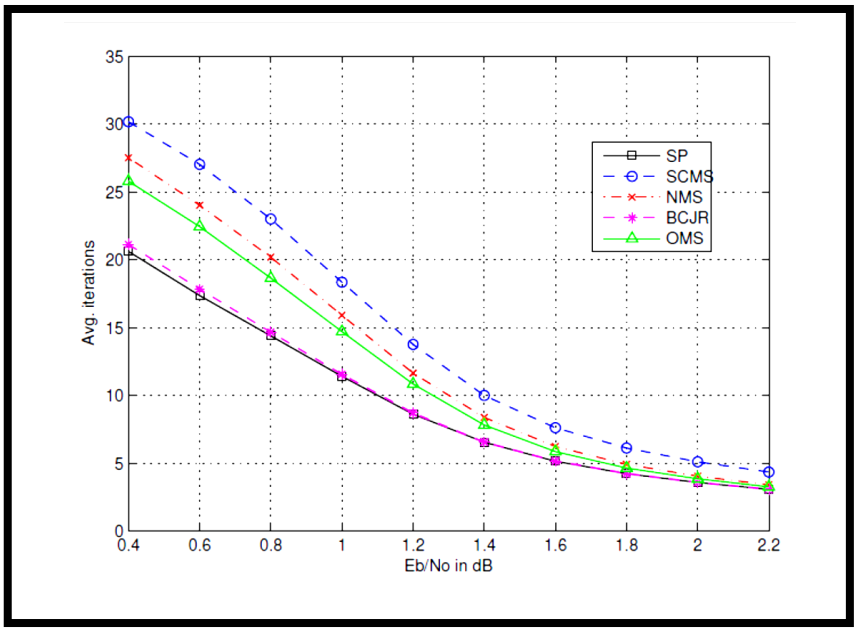
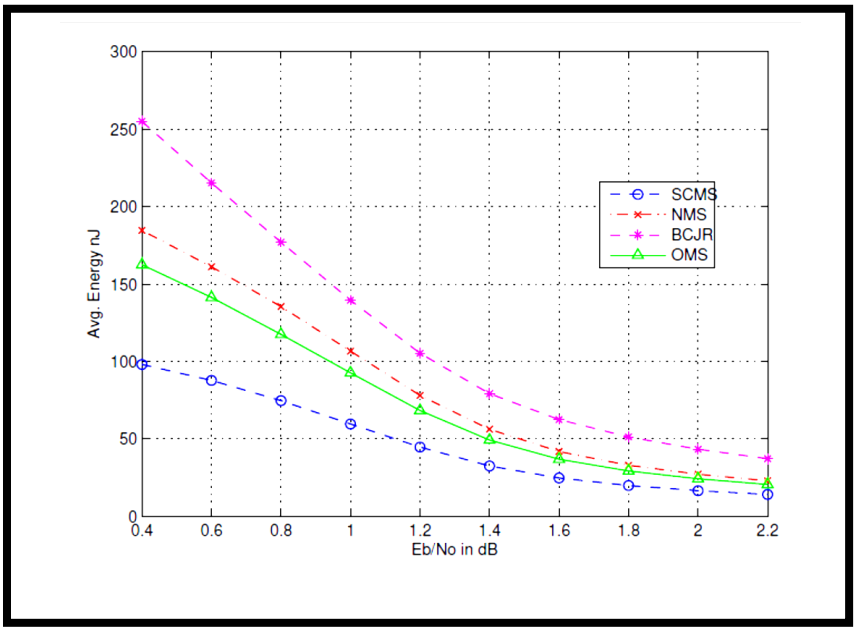


Figure 2. 8- Comparaison des algorithmes dans une implémentation VLSL



(a)- Nombre moyen d'itérations



(b)- Consommation d'énergie des algorithmes

Fig. 2.9- Taux de convergence et consommation d'énergie des algorithmes

2.5 Etat de l'art des techniques de décodage LDPC

- Dans [30], Erick Amador a propose l'algorithme de Self-Corrected-MS (SCMS) pour le décodage des codes LDPC dans le standard IEEE 802.11n à travers le canal AWGN avec une modulation QPSK. Et pour le calcul du syndrome il a proposé une méthode dite à la volée on-the-fly: chaque contrainte est évaluée après le traitement de chaque ligne. La méthode proposée peut avoir des résultats incorrects à la fin du processus de décodage. Pour cela il a proposé une méthode pour valider les décisions erronées en effectuant un calcul du syndrome lors du décodage du mot suivant.

- **Bilal SAMS** dans [33] s'est intéressé à un type des codes LDPC de la famille des codes LDPC non binaires dit « Cluster Non Binaire LDPC codes », et il a apporté des modifications à l'algorithme EMS (Extended Min Sum) basé sur la diversité des codes Cluster NB LDPC pour les appliquer sur ce type de code ainsi il a proposé une autre approche basée sur les listes des codes non-binaires.

- Dans [58], Chris Winstead et all ont proposé une approche de décodage stochastique basée sur la théorie du graphe factoriel (de Tanner) dite approche du graphe contraint. Il présente la relation entre l'information du code et les contraintes. Cette approche a pour but n'ont pas de produire des cycles indépendants mais de produire une approximation de faible complexité de l'algorithme Sum Product. La fonction de la contrainte définie lesquelles des combinaisons des variables sont permise.

- **Erbao LI** dans [60] a apporté une amélioration aux décodeurs FAIDs (Finit Alphabet Iterative Decoders) en proposant un algorithme basé sur la diversité des décodeurs pour les codes LDPC binaires, et qui a permis de corriger des événements d'erreurs que les décodeurs traditionnels (BP et Min Sum) ne parviennent pas à décoder. Et pour le décodage des codes non binaires il a proposé un nouvel algorithme appelé Trellis Extended Min Sum (T EMS) qui est basé sur la transformation du domaine des messages en un domaine dit delta.

- **Ludovic Danjean** et all. dans [61] a présenté une nouvelle classe d'algorithmes de décodage itératif appelés décodeurs multi-niveaux. Ces décodeurs parviennent à corriger des évènements d'erreurs de poids faible avec une complexité raisonnable sur le canal binaire symétrique (CBS). Les performances de ces nouveaux décodeurs sont très prometteuses en terme de taux d'erreur, car même avec une légère dégradation des performances dans

la zone de faible rapport signal sur bruit, ces décodeurs obtiennent de meilleurs performance que l'algorithme de propagation de croyances dans la zone de plancher d'erreur.

- **Saeed Sharifi Tehrani et all**, ont proposé dans [67] une approche de décodage LDPC complètement parallélisable, et pour remédier au problème de bloquant lors de la présence des cycles notamment pour les mots longs, il ont opté opté pour la méthodes des mémoire de front (Edge Memories Ems), ils ont utilisé aussi des mémoires internes pour améliorer les performances.

- **Ali Naderi et all,** dans [68] ont proposé une approche de décodage LDPC stochastique dite « Majority-Based Tracking Forecast Memories (MTFM), qui Comparée aux Ems et TFM, sont moins complexes en termes d'architecture matérielle. Cependant, comme l'approche EM, dans l'approche TFM les VNs utilisent une TFM par front sortant. Alors que pour l'approche MFTM. Au lieu d'assigner une mémoire pour chaque branche comme dans les Ems et FTM, chaque VN utilise uniquement une seule MTFM comme son unité de re-randomisation. D'où une réduction de complexité remarquable en conservant les mêmes performances que les TFMs

- **Shie Mannor et all,** ont proposé dans [69] une nouvelle approche dite « Delayed Stochastic LDPC Decoding », qui est basée sur les états des nœuds et les délais de traitement dans les nœuds de parité. En employant les états des nœuds, elle simplifie l'architecture des nœuds de variable et réduit la nécessité d'utilisation des mémoires dans le décodage LDPC stochastique.

2.6 Domaines d'application des décodeurs LDPC

Les techniques de décodage itératif pour les codes modernes dominent actuellement le choix pour la correction des erreurs dans une pléthore d'applications.

Les Turbo codes, présentés en 1993, ont déclenché une révolution dans le domaine du codage de canal parce qu'ils permettent de s'approcher de la limite de Shannon. Ensuite, les codes LDPC (low-density parity-check) ont été redécouverts.
Ces codes sont actuellement omniprésents dans le contexte des communications mobiles sans fil, entre autres domaines d'application. Par exemple, les Turbo codes sont utilisés dans le standard de téléphonie cellulaire de troisième génération 3GPP Universal Mobile Telecommunications System (UMTS) et son évolution Long Term Evolution (LTE). D'un autre côté, nous pouvons trouver les codes LDPC dans les standards de Wireless Local/Metropolitan Area Networks (LAN/MAN) (IEEE 802.11n et 802.16e) ainsi que la Wireless Personal Area Networks (PAN) (IEEE 802.15.3c). D'autres applications comprennent la transmission de vidéo par satellite de seconde génération Digital Video Broadcast (DVB-S2). [30] [32] [35]

Dans ce chapitre nous avons présenté la description des codes LDPC, leurs types et leur encodage, ensuite nous avons montré quelques Algorithme de décodage des codes LDPC vu dans la littérature. Dans le chapitre suivant nous verrons plus en détail la technique de décodage que nous avons choisi dans le cadre de notre projet de fin d'étude, en faisant appel à la description du décodage itératif et du calcul stochastique

Chapitre 3 Décodage LDPC

Ce chapitre comporte le modèle du décodeur que nous avons choisi, appelé « Décodeur stochastique à initialisation controllée » (Controlled Start-up Stochastic Decoder) proposé par Maamoun et all en 2013.

Il va débuter tout d'abord par la description de la technique de décodage itératif (Sum Product Algorithm comme exemple), suivie de la présentation du calcul stochastique, puis application du calcul stochastique sur les décodeurs LDPC.

3.1 Algorithme somme Produit (Sum Prodauct Algorithm : (SPA))

Dans cette section nous nous intéressons au décodage itératif des codes LDPC. Appliqué aux codes LDPC il permet d'atteindre une très bonne capacité de correction. Cette classe d'algorithmes de décodage a été introduite initialement par Gallager, revu ensuite par MacKay dans le cadre de la théorie des graphes. L'algorithme résultant est alors connu sous plusieurs noms tel que "Propagation de croyance" (Belief Propagation) ou bien encore "Algorithme Somme Produit" (Sum Poduct). Cet algorithme peut être vu comme un algorithme d'échange d'information entre les nœuds du graphe à travers les branches. Ces messages transitant de nœuds en nœuds portent une information probabiliste sur l'état des nœuds. [1] [30] [32] [34]

Le principe de la propagation de croyance est l'application directe de la règle de Bayes sur chaque bit d'une équation de parité. La vérification de parité permet de calculer une estimation de chaque bit. Ces estimations, formant des messages se propageant sur les branches du graphe, sont alors échangées itérativement afin de calculer une information a posteriori sur chaque bit. Dans le cas d'une propagation de croyance sur un graphe sans cycle, les messages échangés sont indépendants, ce qui conduit au calcul simple et exact des probabilités à posteriori : l'algorithme est dans ce cas optimal. Dans le cas des codes LDPC, le graphe factoriel présente des cycles. [30] [32] [34]

Dans ces conditions, l'hypothèse de messages indépendants n'est plus valide. Cependant, plus le graphe est creux (c'est à dire moins la matrice de contrôle de

parité est dense), plus l'approximation d'un graphe sans cycle devient valide. C'est donc sous cette hypothèse que l'algorithme de décodage est décrit. [1]

3.1.1 Algorithme de propagation de croyance

Chaque itération de propagation de croyance est composée de deux étapes : [1] [34]

- Une étape de mis à jour des messages lorsqu'ils passent par un nœud de variable appelée Data pass
- Une étape de mis à jour des messages lorsqu'ils passent par un nœud de contrôle appelée Check pass

Une fois l'ensemble des messages mis à jour, ceux-ci sont propagés des nœuds de contrôle vers les nœuds de données. Enfin, après un certain nombre d'itérations, l'information à posteriori associée à chaque nœud de données est mise à jour avant la prise de décision.

Afin de faciliter la lecture de ce manuscrit, les messages probabilistes seront exprimés en log-rapports de vraisemblance. Nous noterons $\mathbf{m}_{vc}$ les messages se propageant d'un nœud de données à un nœud de contrôle. De la même façon, la notation $\mathbf{m}_{cv}$ est utilisée pour désigner les messages issus d'un nœud de contrôle et transmis à un nœud de données. [1] [34]

La mise à jour des messages $\mathbf{m}_{vc}$ issus du nœud de données v à l'itération **i** est calculée de la façon suivante (figure 3.1):

$$m_{vc}^{i} = v_0 + \sum_{c' \in C_v/c} m_{c'v}^{i-1} \quad \text{..... (3.1)}$$

Où v_0 représente le log-rapport de vraisemblance (LLR) issu de l'observation y_v en sortie du canal :

$$v_0 = ln \frac{\Pr(y_v|v=0)}{\Pr(y_v|v=1)} \quad \text{..... (3.2)}$$

et où C_v représente l'ensemble des nœuds de contrôle connectés au nœud de données ***v***. A la première itération, les messages provenant des nœuds de contrôle sont nuls. [1] [34]

La deuxième étape de l'algorithme de propagation de croyance consiste à mettre à jour les messages en sortie d'un nœud de contrôle. Les messages $\mathbf{m}_{cv}$ sont calculés à l'itération i de la façon suivante (figure 3.2) : [1] [34]

$$sign(m_{cv}^{i}) = \prod_{v' \in V_c/v} \; sign(m_{v'_c}^{i}) \ldots\ldots (3.3)$$

$$\left|m_{cv}^{i}\right| = f\left(\sum_{v' \in V_c/v} \; f\left(\left|m_{v'_c}^{i}\right|\right)\right) \ldots\ldots (3.4)$$

Où Vc représente l'ensemble des nœuds de données connectés au nœud de contrôle *c*.

Une itération de l'algorithme de propagation de croyance est réalisée lorsque tous les messages se propageant le long des branches ont été calculés par les deux relations précédentes. Après chaque itération, une décision peut être prise sur l'information à posteriori A_v^i associée au nœud de données ***v*** : [1] [34]

$$A_v^i = v^0 + \sum_{c' \in C_v} m_{c'v}^{i} \ldots\ldots (3.5)$$

La décision sur la valeur binaire de chaque nœud de données est donc calculée en fonction du signe de l'information a posteriori. Le processus itératif est arrêté au bout d'un nombre maximum d'itérations. On peut également arrêter le processus itératif avant le nombre maximum d'itérations en calculant à chaque itération le syndrome. Si celui-ci est nul alors le décodage itératif a convergé vers un mot de code et le processus peut être arrêté. [1] [34]

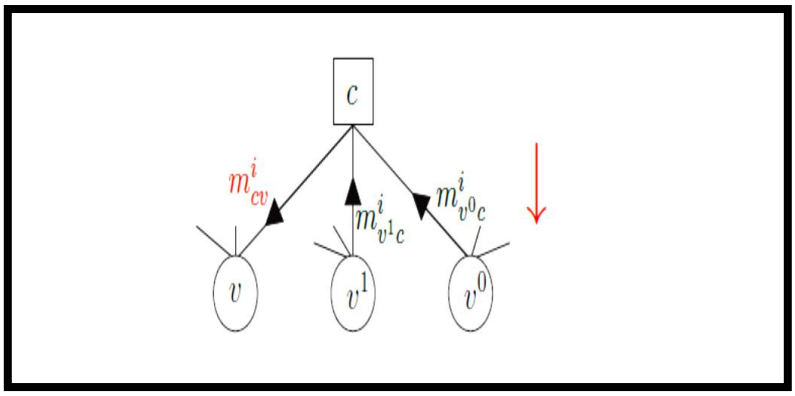


Fig.3.1- illustration de la mise à jour des messages se propageant d'un nœud de contrôle à un nœud de données m_{cv}

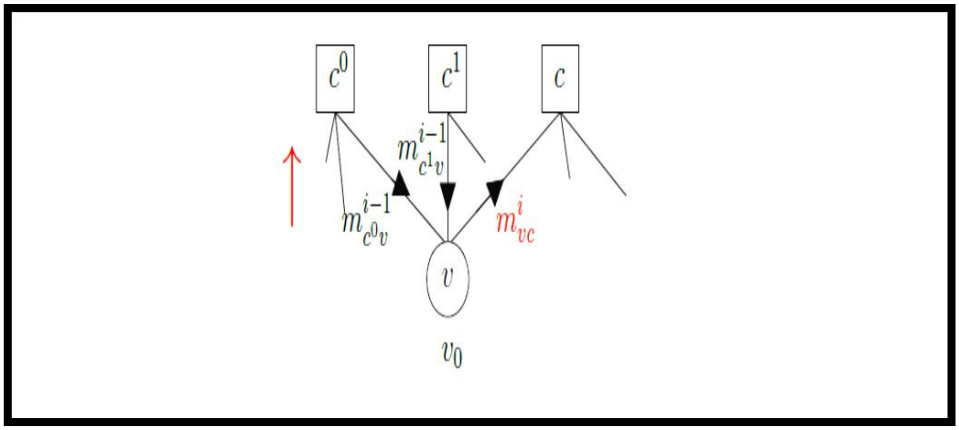


Fig.3.2- illustration de la mise à jour des messages se propageant d'un nœud de données à un nœud de contrôle m_{vc}

3.2 Décodage LDPC Stochastique

La technique de calcul stochastique est d'abord présentée dans les années 1960 par Gaines. Elle a été appliquée avec succès aux différentes applications comme l'implémentation des réseaux de neurones et des systèmes de contrôle de moteur. Elle a récemment été utilisée pour décoder des codes utilisant des graphes. Le décodage stochastique fût d'abord considéré pour des codes de petite taille : les codes (7,4) Hamming en 2003 par Gaudet et all. La première implémentation du décodage stochastique des codes (16,8) LDPC a été décrite par Warren Gross en 2005. Plus tard, un algorithme de décodage stochastique basé sur la représentation de treillis a été appliqué pour décoder des turbo-codes en bloc (256,121). Une approche de décodage plus efficace a été employée pour des codes LDPC par Sharifi et al. en

2006. Puis, en 2007, l'implémentation sur FPGA d'un décodeur LDPC(1056,528) produit un débit maximum de 1,66 Gbps. Il s'agit du premier décodeur stochastique atteignant un débit supérieur au Gbps. Récemment, en 2010, un débit notable de 61,3Gbps a été obtenu par une implémentation d'un décodeur LDPC(2048,1723) sur cible ASIC. En comparaison avec l'implémentation conventionnelle en virgule fixe, le décodage stochastique peut produire une performance optimale pour des codes LDPC de petite taille, et une performance quasiment optimale pour des codes LDPC similaires à celles des standards. Il est à noter que sa complexité d'architecture s'approche à celle de l'architecture la plus efficace d'aujourd'hui. **[56] [57] [58]**

Dans le calcul stochastique, les informations s'expriment par des séquences de bits - les séquences de Bernoulli. La probabilité d'occurrence de bits à "1" dans la séquence est égale à la probabilité que le message transmet. Ainsi, des opérations arithmétiques classiques comme multiplication, division et addition sont remplacées par des opérations logiques utilisant des portes logiques simples. Dès lors, le décodage stochastique se déroule par l'échange des informations binaires en série, au lieu de propager des messages probabilistes entre les nœuds de graphe. [59]
La suite de la section détaille le décodage stochastique des codes LPDC par l'introduction des concepts de base du traitement stochastique, ainsi que les blocs logiques utilisés pour les opérations arithmétiques. [56] [57] [58]

3.2.1 Principe du calcul stochastique

[56] [60] [61] [62]

Le calcul stochastique est une technique qui associe à chaque probabilité une séquence discrète aléatoire des valeurs numériques {ai} où i = 1...L avec L la longueur de la séquence. ai prend les valeurs binaires (0,1). Ces bits constituent une séquence de Bernoulli, dans laquelle la probabilité représentée est contenue dans la statistique du flux stochastique {ai}. Lors de sa génération, la probabilité du symbole "1" est équivalente à la probabilité que ce flux représente. Un comparateur peut être utilisé

afin de convertir une probabilité en flux stochastique comme illustré dans la figure (3.3).

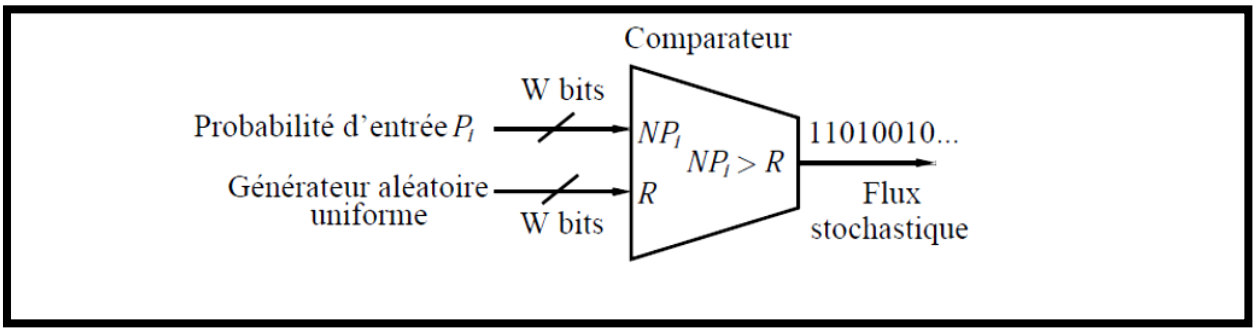


Fig. 3.3- Génération d'un flux stochastique représentant une probabilité P_i en utilisant un comparateur et un générateur pseudo-aléatoire à distribution uniforme.

Dans ce cas, la probabilité Pi est mise à l'échelle à W bits de largeur sur un bus de NPi. Le résultat est comparé avec un R de la même largeur W qui contient des valeurs aléatoires variant dans le temps selon une distribution uniforme. La sortie du comparateur est égale à "1" si NPi est supérieure à R (NPi > R). Elle est égale à "0" dans le cas contraire. La probabilité de production d'un bit "1" dans le flux généré est équivalente à NPi 2W et le nombre total de bit "1" présent dans le flux désigne la probabilité Pi. Par exemple, si une séquence binaire de longueur 20 bits contient 4 bits à "1", alors la probabilité correspondante est 0,2. De même, si 5 bits sont à "1" dans une séquence de longueur 10 bits, la séquence porte un message de probabilité 0,5. Cependant, de nombreuses représentations peuvent être considérées pour coder une même probabilité, comme l'exemple montré en figure **(3.4)**. [56] [62] [63] [64]

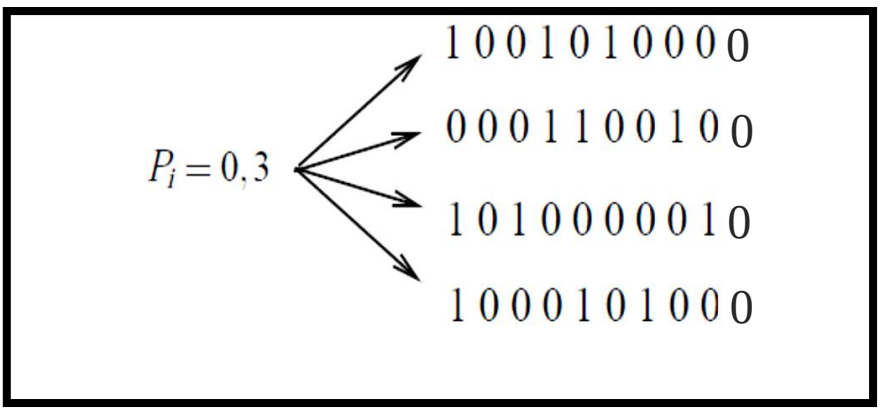


Fig. 3.4- Quelques exemple de représentation d'une probabilité P_i=0,3

Cela signifie que les positions d'occurrence du bit "1" dans la séquence ne sont pas importantes. En utilisant la représentation stochastique, la probabilité est convertie en une série de bits, c'est pourquoi, un de ses avantages est de ne nécessiter qu'une seule équipotentielle pour représenter le signal de communication entre les éléments de calcul. Dès lors, les opérations arithmétiques sur les probabilités sont remplacées par des opérations logiques plus simples sur des séries de bits. Cet avantage permet d'atteindre une fréquence d'horloge de fonctionnement élevée. De plus, lors d'une présence de bruit à un moment, un seul bit stochastique dans le flux est affecté au lieu de l'intégralité du message interprété. Alors, une représentation par des flux stochastiques est robuste lors de la présence de bruit binaire. [56] [60] [64]

Une des contraintes de l'approche stochastique est qu'en présence d'une probabilité faible, une large précision est nécessaire. Par exemple, seuls 10 bits sont vraisemblablement suffisants au domaine des virgules fixes pour exprimer une probabilité de 0,001. Par contre, au moins 1000 bits stochastiques sont nécessaires pour représenter cette probabilité. Une meilleure précision de calcul stochastique est obtenue en observant plus longtemps la séquence des bits. Par ailleurs, les traitements stochastiques reposent sur les changements dans la statistique des flux plutôt que sur la précision des séquences stochastiques. Ceci entraîne que la précision des calculs stochastiques ne peut pas être aussi importante. [56]

3.2.2 Décodage stochastique des codes LDPC

[56] [58] [61] [62]

L'approche stochastique permet de transformer des opérations arithmétiques complexes sur les probabilités comme la multiplication ou la division par des opérations logiques simples utilisant des portes logiques élémentaires, des bascules JK ou des multiplexeurs. En définissant que les flux des bits stochastiques {ai} et {bi}

représentent les probabilités d'entrée Pa = Pr(ai = 1) et Pb = Pr (bi = 1), le flux { ci } représente la probabilité de sortie de Pc = Pr(ci = 1) qui est le résultat des opérations stochastiques sur les flux des probabilités Pa et Pb. [56][65]

En profitant des atouts de l'approche stochastique - simplicité du calcul et de la structure matérielle correspondante - le décodage stochastique est applicable à des codes LDPC. Différentes implémentations sur une carte FPGA ont abouti à des débits très élevés : 706 Mbps; 1,66 Gbps. Le premier décodeur stochastique sur cible ASIC a atteint un débit notable : 61,3 Gbps. De plus, ce décodeur stochastique utilise une très faible complexité de l'architecture. Son efficacité architecturale est seulement plus petite de 4% par rapport à l'architecture la plus efficace en littérature. Ces résultats démontrent que le décodage stochastique est une des approches les plus efficaces pour atteindre une architecture de décodeur des codes correcteurs d'erreurs à haut débit et faible coût. [58][59][63]

Le décodage stochastique des codes LDPC débute par la conversion des vraisemblances des bits reçus en des séquences de Bernoulli. Chaque séquence de bits est ensuite propagée à travers le graphe bipartite [58]. Dès lors un seul fil est nécessaire pour présenter un arc entre un nœud de variable et un nœud de parité. Les figures **(3.5)** et **(3.6)** présente les architectures stochastiques des nœuds de variable et de parité respectivement. Cet avantage simplifie significativement la complexité des nœuds et réduit également le problème de congestion de routage. Ceci implique que durant chaque cycle de décodage, le procès de décodage stochastique est effectué par l'échange d'un bit entre un nœud de variable et un nœud de parité sur l'arc de graphe correspondant. Chaque cycle est défini comme un cycle de décodage (decoding cycle - DC en anglais) qui est différent d'une itération de décodage dans l'algorithme par propagation de croyance. Le processus de décodage stochastique des codes LDPC s'arrête après un nombre maximum de cycles de décodage, ou lorsque la matrice de contrôle H est vérifiée. **[56]** [60] [61] [62]

3.2.3 Approches de décodage LDPC stochastique

a Algorithme de décodage LDPC stochastique classique : 2003

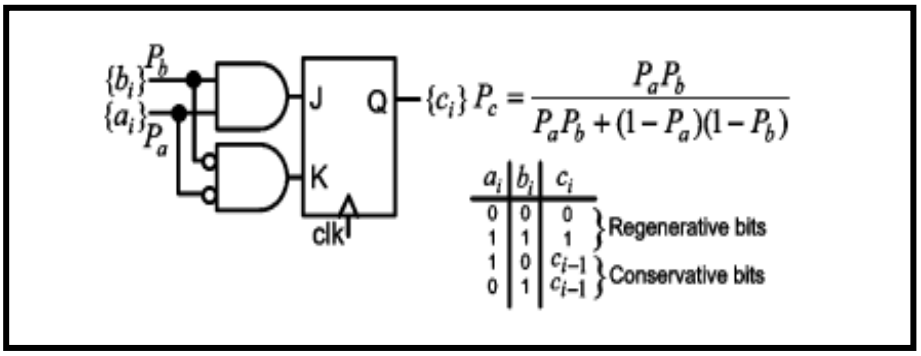


Fig. 3.5- Les nœuds de variable stochastiques du graphe de Tanner du code LDPC proposée par Gaudet et Rappely

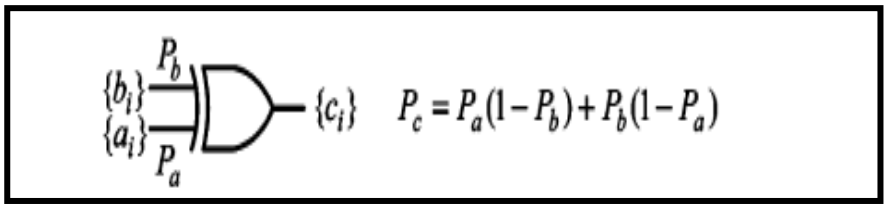


Fig. 3.6- Les nœuds de parité stochastiques du graphe de Tanner du code LDPC proposée par Gaudet et Rappely

Si les entrées du nœud de variable sont identiques, une des deux entrées sera verrouillée à la sortie. Quand les entrées soient différentes le nœud de variable exige une méthode très avancée pour générer le bit de sortie. Cet état est appelé l'état de l'influence ou état de désaccord. Une des méthodes stochastique avancée de génération du bit de sortie est (ASMBG). Après cette première évolution, beaucoup de travaux se sont focalisés sur ce contexte on se basant principalement sur le développement de nouvelle architecture des nœuds notamment les nœuds de variables, en vu d'accroitre les performances. La figure **(3.7)** montre une structure avancée du nœud de variable. [63] [64]

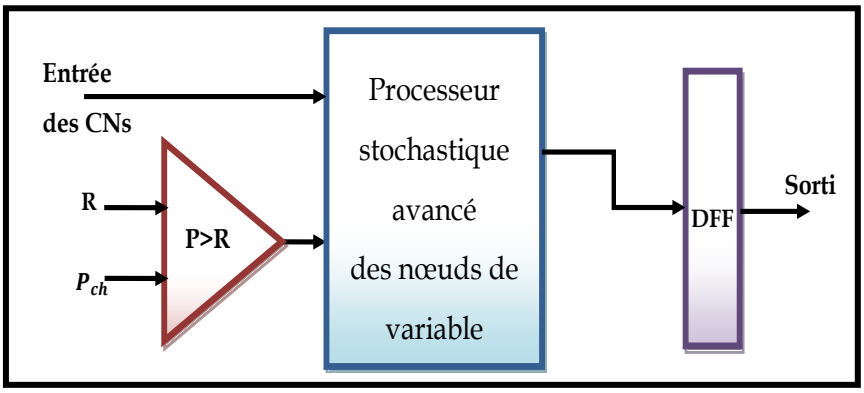


Fig.3.7- Structure principal du nœud de variable stochastique avancé [57]

b *Méthode des mémoires de front (Edge Memories : EM en Anglais) : (2007/2008)* [67]

Les EMs sont des mémoires assignées aux fronts du graphe factoriel. EMs sont utilisées pour surmonter la corrélation entre les séquences stochastiques et le problème de verrouillage.

Les nœuds de variable

EMs sont implémentées comme des registres de décalage (shift registers), avec un bit sélectionné. Les nœuds de variable ont 2 modes de fonctionnement, décris comme suit :

- **Mode d'initialisation partielle** : précède l'opération de décodage. Après le chargement des probabilités du canal, les nœuds de variable commencent à initialiser les EMs, suivant la probabilité reçue. L'initialisation améliore donc la convergence du décodeur stochastique.
- **Mode de décodage :** Après la phase d'initialisation partielle, l'opération de décodage commence. Chaque nœud de variable dans la figure (3.9) utilise le signal U pour mettre à jour EM et IM (internel memory).

Les nœuds de parité :

La construction des check nodes est basée sur le XOR des bits d'entrée reçu à partir des variables nodes. En plus chaque variable node produit un signal de sortie de

contrôle de parité qui détermine si le contrôle de parité correspondant est satisfaisant.

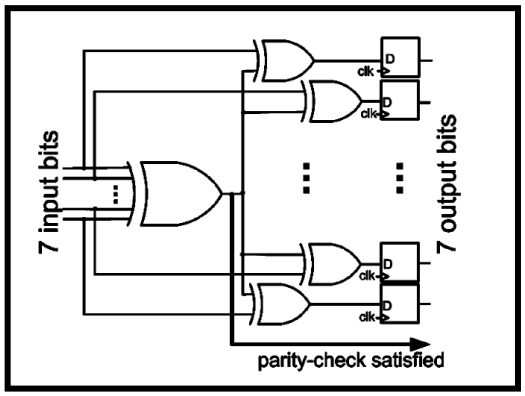
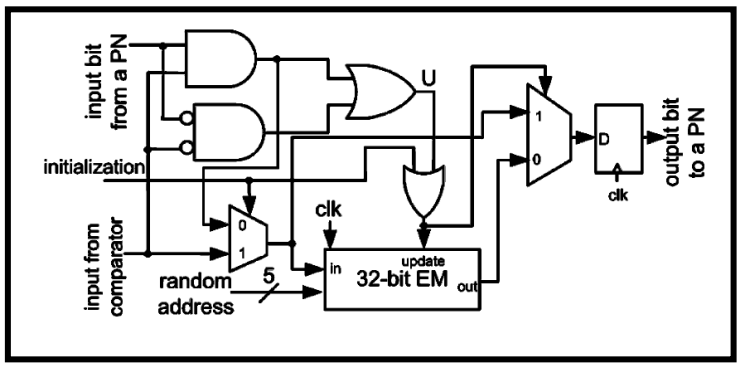
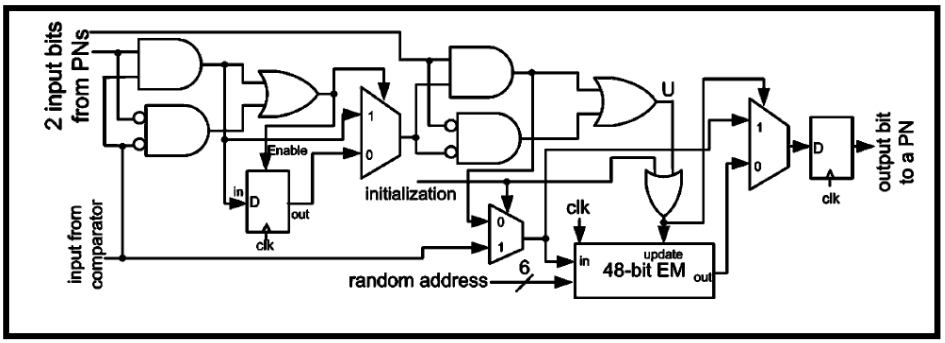


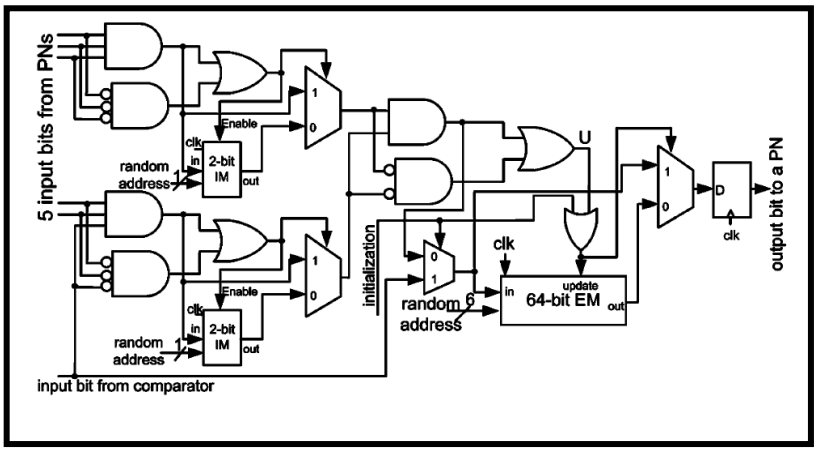
Fig.3.8- Architecture du nœud de parité stochastique avec un d_c=7



(a)



(b)



(c)

Fig.3.9- Architecture de (a) nœud de variable à d_v=2, (b) d_v=3, (c) d_v=6 basés sur EMs (dans chaque figure, seulement une sortie et son entrée correspondante sont montrées)

c *Méthode des mémoires de poursuite de prédiction (Tracking Forecast Memories : TFM) : (2009)*

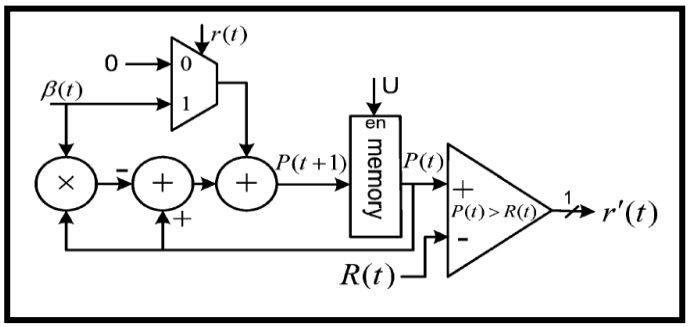


Fig.3.10- Architecture générale de TFM

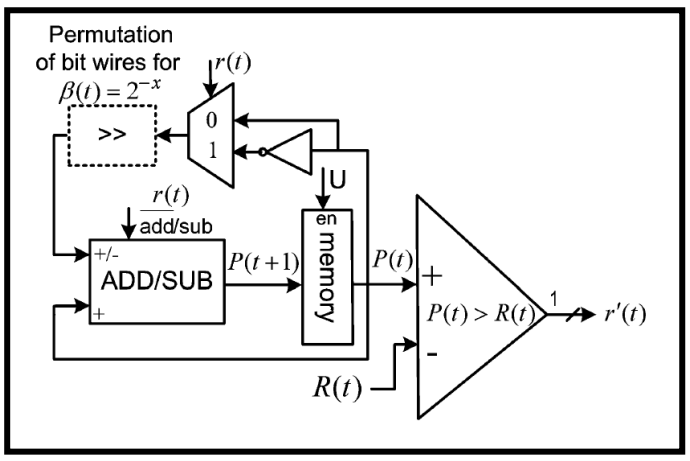


Fig.3.11- Architecture de TEM à complexité réduite

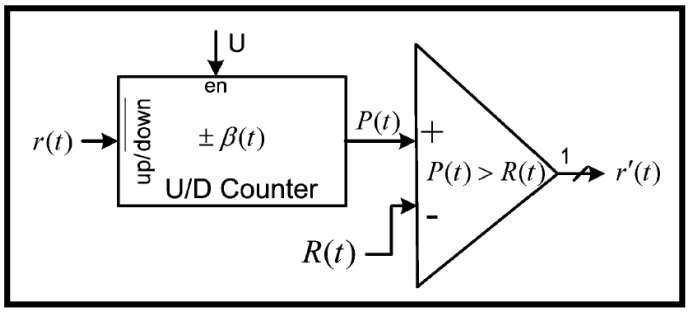


Fig.3.12- Architecture d'un compteur approximatif basé TFM

La TFM extrait la probabilité moyenne P(t) de la série stochastique basée sur la relaxation successive. La TFM est mise à jour selon l'état du nœud de variable.

Cette architecture utilise un multiplieur, 2 additionneurs, 1 comparateur et un registre.

Architecture à complexité réduite : (fig. 3.11)

Comparée à l'architecture générale de TFM, cette architecture n'utilise aucun multiplieur et un additionneur en moins.

D'après les simulations, on a constaté qu'en utilisant cette architecture ils ont abouti aux même performances.

d *Méthode TFM basée sur la Majorité (Majority-Based TFM : MTFM) : (2010)* [68]

Comparée aux EMs, TFMs sont moins complexes en termes d'architecture matérielle. Cependant, comme l'approche EM, dans l'approche TFM les VNs utilisent une TFM par front sortant. Cependant, le nombre des TFM dans décodeur stochastique égal au nombre de branches du graphe de Tanner associé. Même si la complexité des TFM est réduite par rapport à l'approche EM, l'architecture des TFM reste toujours complexe. Dans l'approche MTFM, au lieu d'assigner une mémoire pour chaque branche, chaque VN utilise uniquement une seule MTFM. D'où une réduction supplémentaire de la complexité.

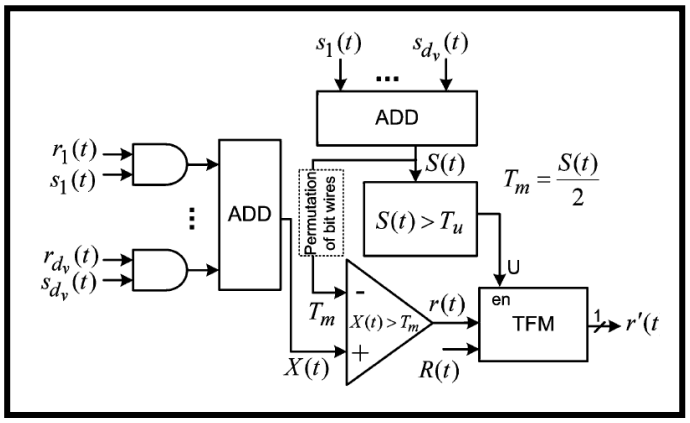


Fig.3.13- Architecture générale de MTFM

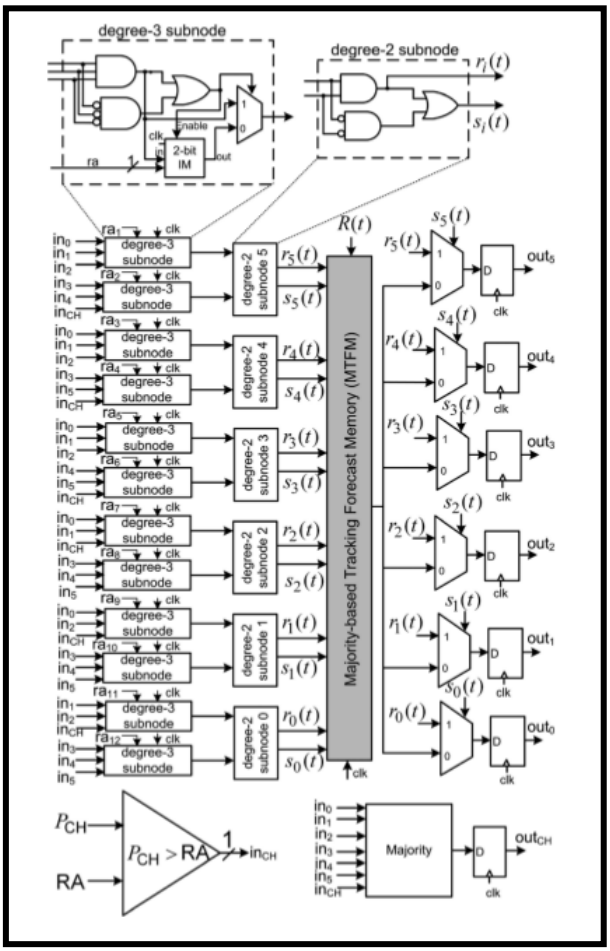


Fig.3.14- Structure du nœud de variable de degré 6 basé MTFM

e *Méthode de décodage stochastique retardé (Delayed Stochastic LDPC decoding: DS) : (2011)* [69]

Notons que les nœuds de variable des méthodes stochastiques basées sur les filtres, telles que TFM et EM, n'utilisent jamais ces derniers quand tous les bits reçus sont similaires. Pour cela Ali Naderi et all. ont Introduit dans [69] la technique DS.

Nœuds de parité (CNs):

La figure (3.15) montre structure du nœud de parité proposé. Comme on le constate, si aucun état des signaux égal à 1, CS = 0. Dans tel état, les sorties (b2, b1, b0) sont égaux aux entrées.

Si tous les signaux égaux à 0, les nœuds de variable sont en état régulier, et la probabilité sera calculée selon une formule données (pour plus de détail voir [69]).

Nœuds de variables (VNs) :

La figure (3.16) montre la structure du nœud de variable proposé. La valeur de la probabilité est comparée à un nombre aléatoire (distribution uniforme) à chaque itération. La sortie de la comparaison est un bit aléatoire (p_{ch}) avec la probabilité donnée. Afin d'augmenter la rapidité du décodage, le nœud de variable fonctionne comme suit :

- Si la majorité des états sont égaux à 0 : chaque front envoie son état actuel. Si cet état est un état décidé il envoie le bit régénératif trouvé, autrement un bit aléatoire.
- Si la majorité des états sont égaux à 1 : il change tous les états dans le VN en un état artificiel, et envoie aléatoirement les bits à partir de la mémoire interne à travers chaque front.

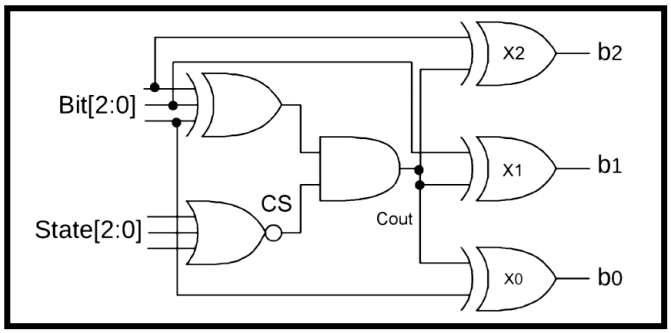
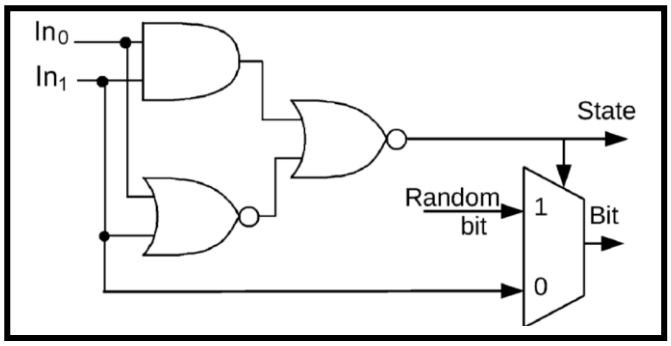
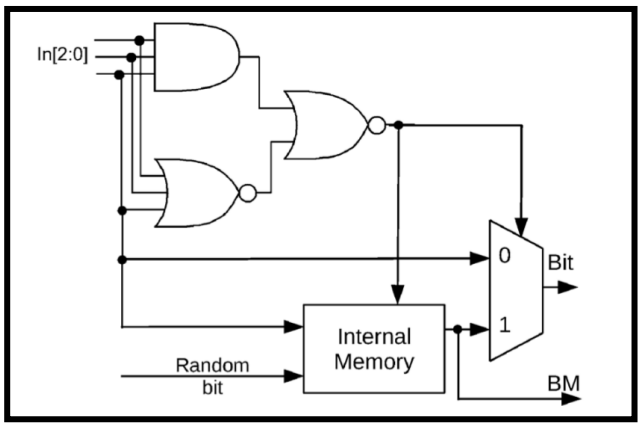


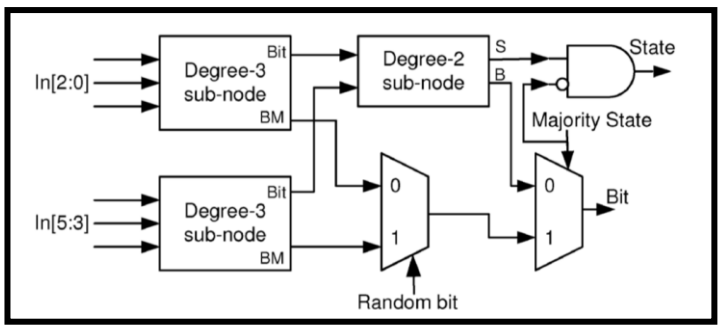
Fig.3.15- Structure du nœud de parité de degré 3 basé DS



(a)



(b)



(c)

Fig.3.16- Structure du sub-nœud de variable basé DS (a) de degré 2, (b) de degré 3, de degré 6

3.3 Décodeur stochastique à initialisation contrôlée (CSS)

Dans l'algorithme Sum Produt (SPA) ou autre processus itératif, tous les nœuds de variable sont initialisés par la probabilité reçu P_{ch}. SPA atteint la décision par la comparaison des valeurs des probabilités finales de la décision dure (hard decision) à la fin du processus itératif. [57][60] [61][65]

Dans tous les décodeurs LDPC stochastique publiés, les nœuds de variable sont initialisés par un zéro ou un état aléatoire. Le décodeur stochastique s'arrête lorsqu'il atteint le code correct ou le nombre maximum d'itérations. [57][62][66]

La séquence de Bernouli est utilisée comme entrée du nœud de variable.

La probabilité du nœud de variable est calculée comme suit :

$$P_c(N) = \begin{cases} bit\ aléatoire & si\ N = 1 \\ P_a\ ou\ P_b & si\ N \neq 1\ et\ P_a = P_b \\ bit\ avec\ ASMBG & sinon \end{cases} \quad \ldots\ldots (3.8)$$

Dans le décodage CSS, l'algorithme précédent est modifié pour améliorer la convergence et réduire le cycle de décodage par une initialisation simple et habile des nœuds de variable. Tous les nœuds de variable du CSS sont initialisés par une probabilité codée sur un bit. L'algorithme suivant illustre le calcul de la probabilité initiale P_c.

Algorithme d'initialisation du nœud de variable

Si $\boldsymbol{P_{ch} \geq 0,5}$ ***alors***
  $\boldsymbol{P_c(1) = 1}$
Sinon
  $\boldsymbol{P_c(1) = 0}$
Fin si

La probabilité du nœud de variable deviendra calculable comme suit :

$$P_c(N) = \begin{cases} Pst & si\ N = 1 \\ P_a\ ou\ P_b & si\ N \neq 1\ et\ P_a = P_b \\ bit\ avec\ ASMBG & sinon \end{cases} \quad \ldots\ldots (3.9)$$

$$\text{Avec: } Pst = \begin{cases} 1 & si\ P_{ch} \geq 0{,}5 \\ 0 & sinon \end{cases} \quad \ldots\ldots (3.10)$$

A partir de (3.9) et (3.10), l'implémentation du décodeur CSS n'implique pas une complexité d'implémentation matérielle, que se soit pour les ASIC que pour les FPGAs. Seulement un multiplexeur (2X1) est rajouté. En plus, cette technique calcule la probabilité reçue sans avoir recours à des cycles supplémentaires. La figure (3.17) montre la structure du nœud de variable du décodeur CSS.

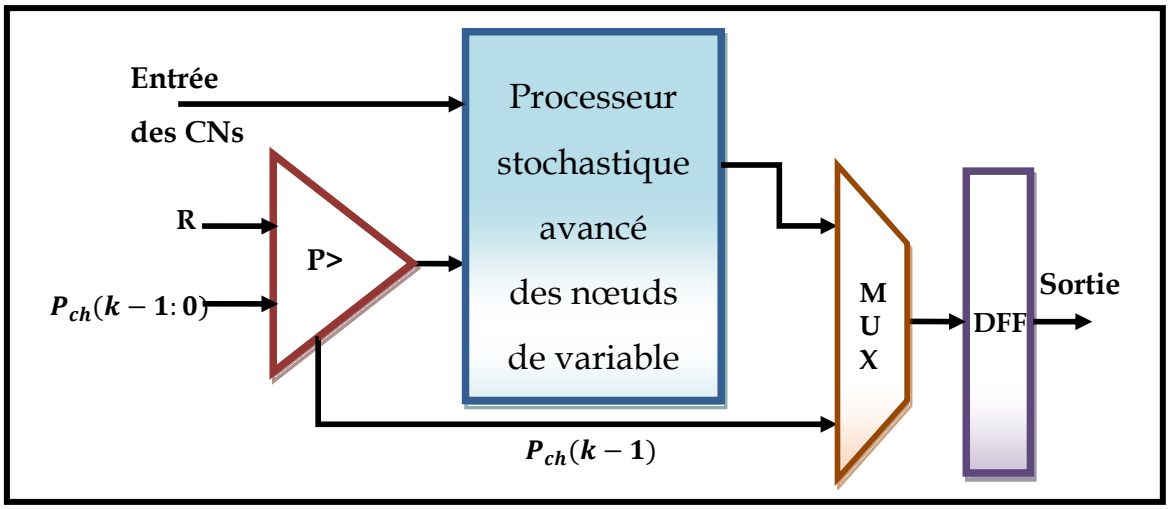


Fig.3.17- Structure du nœud de variable du décodeur CSS

Le multiplexeur ajouté est placé entre le processeur du nœud de variable et la sortie de la bascule D (D Flip Flop). Le multiplexeur sélectionne l'un des deux signaux

d'entrée, la sortie du nœud du processeur du nœud de variable ou le bit de poids fort de P_{ch} .

Le cycle de décodage (DC) correspond à une itération du décodeur CSS. Durant N itérations du processus de décodage (de 1 à N) le multiplexeur transfère le signal $P_{ch}(k-1)$ vers la sortie DFF pendant la 1ére itération uniquement. Du deuxième au Nième itération, le multiplexeur transfère la sortie du processeur du nœud de variable vers la sortie du DFF. Avec cette approche, un nombre important des nœuds de variable commence par un bit correct tout en évitant le bit stochastique aléatoire. Dans le cas d'un code (2048, 1729), 98% des nœuds de variable commence par un bit correct si E_b/E_0= 5 dB.

Après avoir présenté une étude du décodeur LDPC Stochastique CSS « Controlled Start-up Stochastique Decoding for LDPC codes » pour lequel nous avons opté.

Dans le chapitre qui suit, nous allons exposer la procédure de travail et nous discutons les résultats obtenus.

Chapitre 4 Implémentation et résultats

Après avoir entamé les chapitres précédents qui nous ont servi de prélude pour notre étude pratique, il est temps alors de décrire pratiquement notre travail.

Ce chapitre va tout d'abord débuter par une simple présentation de l'environnement de travail et des logiciels utilisés, la procédure de travail que nous avons suivi pour réaliser notre travail.

Nous exposons la structure du décodeur CSS que nous avons choisi, l'interface réalisée pour l'exportation de la description VHDL, ses élément et son mode d'emploi.

Puis nous exhibons la présentation, l'interprétation et la comparaison des résultats de simulation de l'algorithme de décodage étudié avec d'autres algorithme de décodage LDPC les plus fréquemment utilisés.

4.1 Environnement de travail

Ce travail a été réalisé à l'université de Blida sous le suivi et la surveillance de Mr. M. MAAMOUN.

Nous avons utilisé deux ordinateurs, le 1er équipé d'un processeur intel core i3 2350M, GPU INVIDIA GeForce GT 540M 2GB DDR3 et d'une mémoire vive RAM 4 GB , le 2ème doté du processeur pentium R Dual-Core CPU T4200 @ 2.00 GHz et d'une mémoire vive RAM de 4 GB, sous le système d'exploitation linux Mint 14.

4.2 Langages adoptés pour la programmation et la description

Nous avons choisi comme langage de programmation « Matlab » Version 7.11.0 (R2010 b).

Notre choix est justifié par la souplesse et la facilité d'utilisation et de mise en œuvre des scripts avec le langage Matlab. En plus, Matlab (matrix laboratory) est une bibliothèque très riche et simple à utiliser, elle nous a permis d'utiliser ces propres

fonctions, citant à titre d'exemple : Les fonction d'exportation et d'importation de plusieurs format de fichiers.

Matlab est un langage très performant, il est utilisé pour :

- Le calcul scientifique
- L'acquisition des données
- Le développement des algorithmes
- La modélisation et simulation
- L'analyse, l'exploration et visualisation des données et des résultats

L'outil Guide (GUI) nous a bien facilité la tâche de la réalisation d'une interface graphique contenant les scripts de traitement, afin de fournir à l'utilisateur un outil simple à manipuler et rendre le travail plus professionnel. Elle est utilisé pour l'exportation des fichiers (*.vhd) vers ISE Xilinx ou System Generator, contenant la description VHDL du décodeur LDPC stochastique étudié dans notre mémoire, ainsi que pour l'adaptation de tout type de matrices H de décodage selon un modèle normalisé de Mackay & Neal, et ce, pour l'automatisation du processus global.

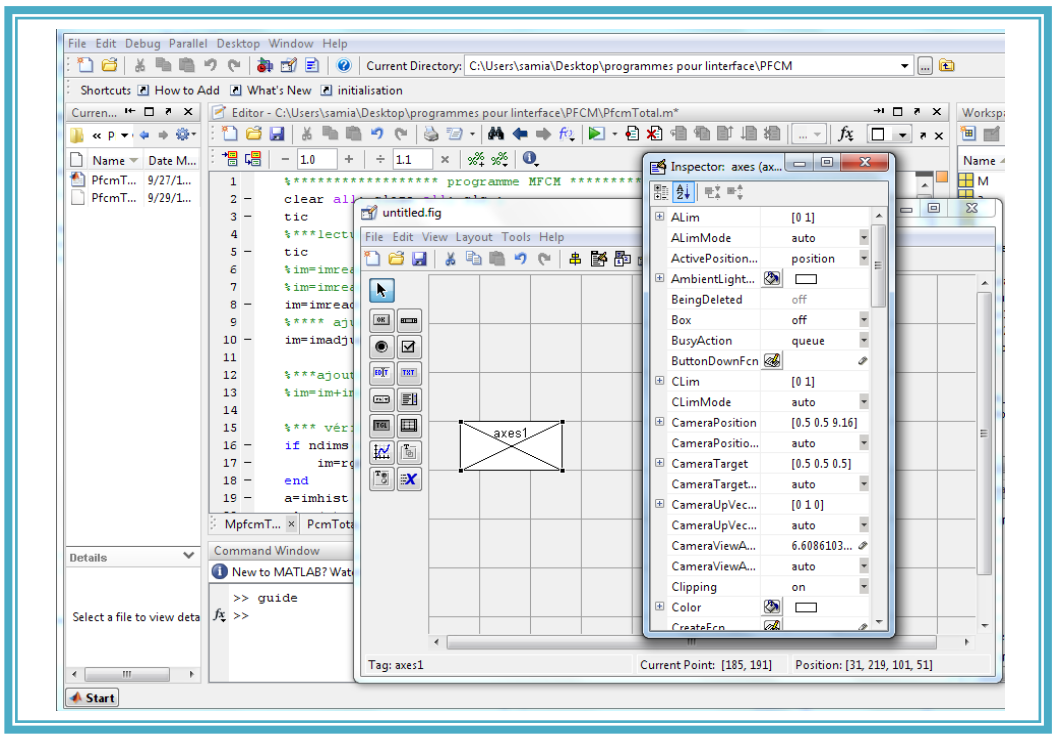


Fig. 4.1 - Présentation du guide de Matlab

Nous avons choisi le langage VHDL comme langage de description du décodeur à implémenter sur FPGA, vu les grands avantages qu'il offre et que nous avons décrit dans le chapitre 1.

4.3 Procédure de travail

Notre but principal est d'étudier l'implémenter sur FPGA, d'un nouveau modèle -très performant- de décodeur LDPC stochastique, publié très récemment dans [57].

Et afin d'aboutir à un résultat optimal, nous avons procédé comme le montre l'organigramme de la figure (4.2) :

En premier lieu une étude des différents travaux réalisés dans le domaine de décodage LDPC a été mise en place, à travers un état de l'art des principaux travaux de décodage LDPC. Parmi ces travaux, notre choix s'est dirigé vers le décodage LDPC Stochastique proposé dans [57] appelé « Décodeur stochastique à initialisation contrôlée » (Controlled Start-up Stochastique Decoder).

Après avoir étudié le principe de ce décodage, nous avons élaboré un script sous Matlab permettant de faire l'exportation d'une description VHDL du décodeur en question, plus précisément des interconnexions entre les nœuds de variable et les nœuds de parité inclus tous leurs composants associés vers ISE Xilinx ou System Generator, ensuite nous avons élaboré un test très performant en se servant une plateforme réalisée par Simulink de Matlab afin d'évaluer les performances du modèle choisi. Qui avant l'utiliser, nous l'avons adapté selon les contraintes qu'impose notre système.
Une dernière étape est l'implémentation sur FPGA du modèle Choisi.

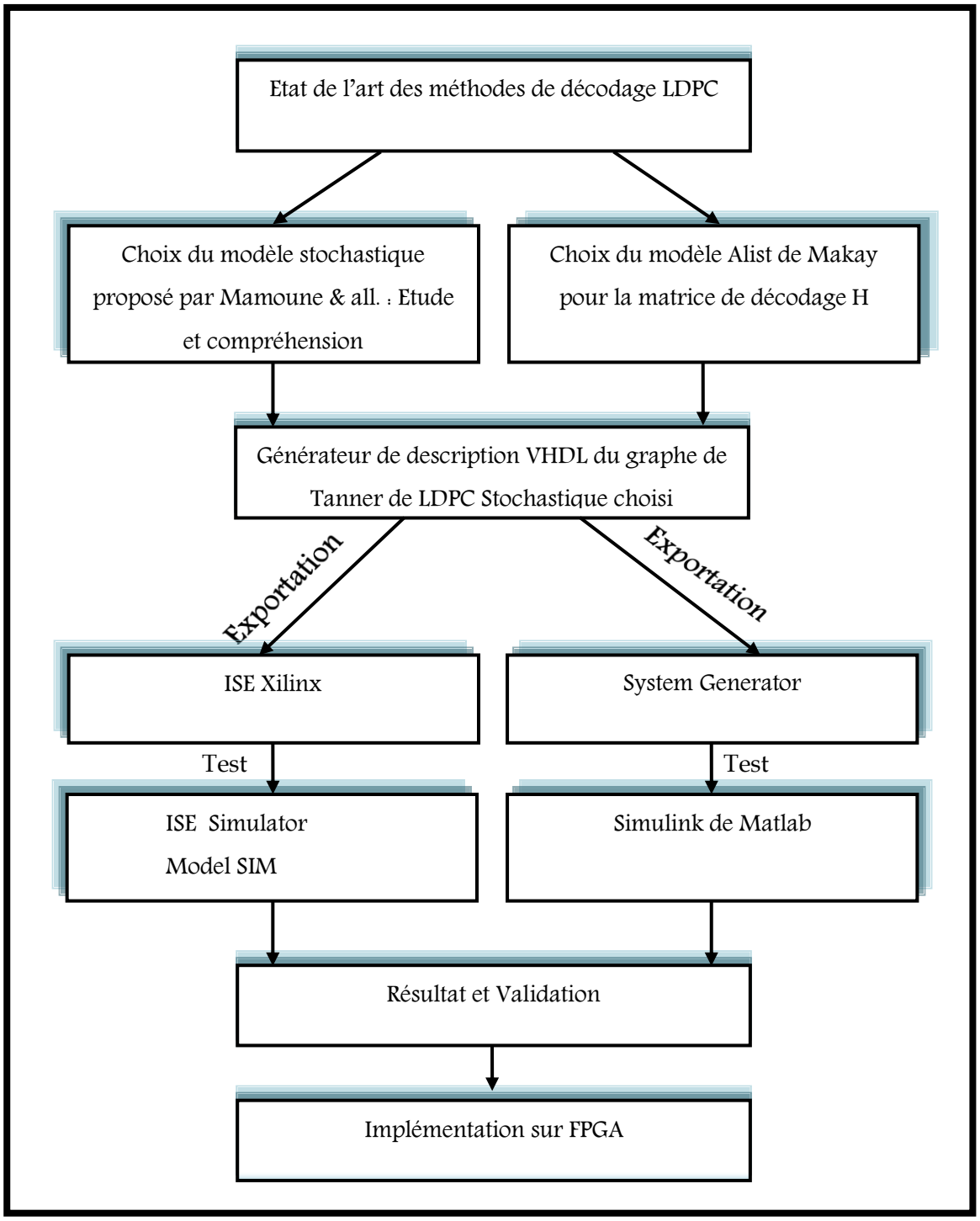


Fig. 4.2 - Procédure de travail

4.4 Structure générale du décodeur CSS

La structure générale du décodeur étudié dans notre mémoire est montrée sur la figure (4.3). Cette structure est composée de deux blocs principaux :

- Bloc de la structure principale
- Bloc de l'interface I/O

Le 1er bloc décrit la structure globale du décodeur LDPC CSS notamment les connexions nœuds de variable ↔ nœuds de parité et des composants associés, il reçoit les probabilités du canal P_{chs} au niveau des nœuds de variable d'une façon parallèle. Ces message seront échangés entre nœuds de variable et nœuds de parité itérativement jusqu'à obtention d'une information correcte ou jusqu'à atteindre le nombre maximum d'itérations fixé au préalable. En calculant à chaque fois le syndrome.

Le 2ème bloc présente l'interface d'entrée/sortie qui est basée sur la conversion série parallèle de l'information provenant du canal de transmission, qui est les probabilités du canal. Elle est utilisée lorsque le nombre d'entrées / sorties à manipuler est supérieur à celui supporté par le circuit FPGA dont on dispose.

Cette interface I/O sert à recevoir les bits d'information en série, puis les transmettre paquet par paquet d'une façon parallèle vers le bloc principal, qui seront concaténés puis stockés, une fois tous les paquets de la première série transmis, les paquets de la deuxième seront transmis.

Le décodeur de l'interface I/O a pour but de placer les paquets transmis selon l'ordre voulu dans l'élément de stockage. Cette interface a été simulée sur Simulink de matlab pour manipuler des matrice de contrôle de parité de taille élevée : H(1024, 512), H(1723,2048) …etc.

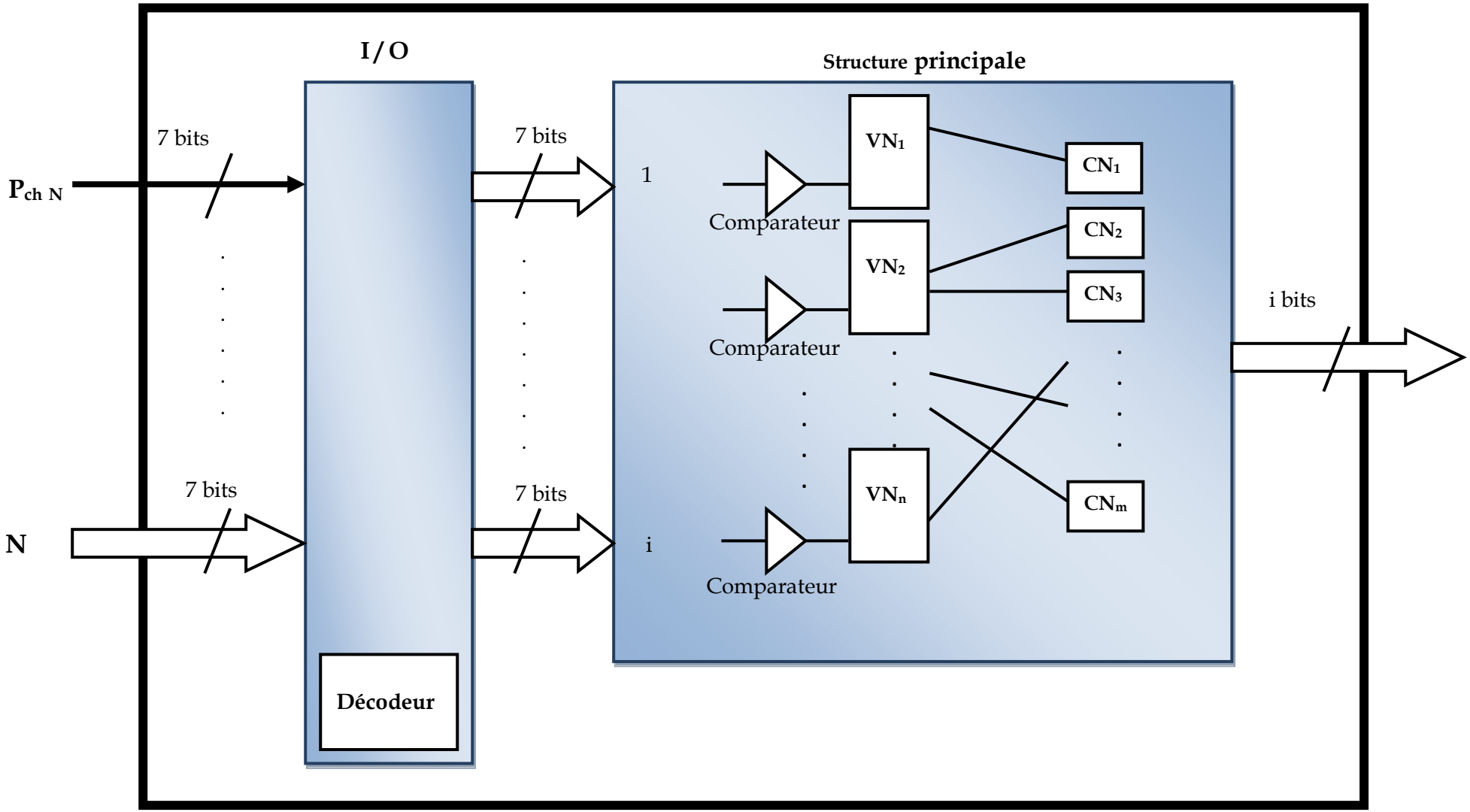


Fig. 4.3- Synoptique générale du décodeur LDPC CSS

4.5 Synthétiseur VHDL pour Décodeur CSS

4.5.1 Interface graphique du synthétiseur

Nous avons réalisé une interface graphique en utilisant le Guide de Matlab, cette interface a pour objectif de générer une description VHDL interprétable par ISE Xilinx et System Generator, et ce à partir de la matrice de parité H. Cette interface a comme avantage d'adapter pas mal de types de matrices de parité H selon le modèle supporter par notre software - qui est le modèle de Mackay dit Alist Matrix -, puis exporter le VHDL vers ISE Xilinx ou System Generator. Donc c'est un système automatique qui permet :

- La lecture de la matrice de parité H
- La convertir en un format proposé par Mackay (Alist Matrix) supporté par notre software
- Enfin, exporter la description VHDL du décodeur plus précisément l'architecture d'interconnexion entre les Nœuds de variable et les Nœuds de parité mutuellement.

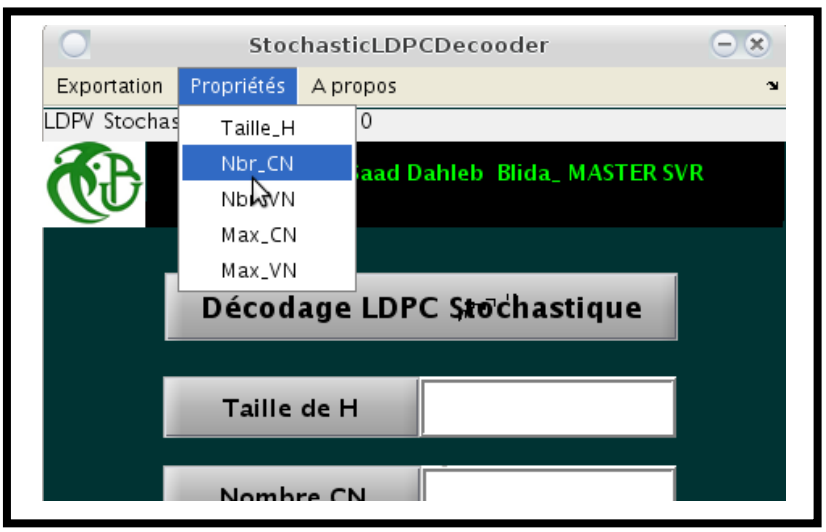
Il était donc nécessaire de fournir à l'utilisateur un outil permettant de manipuler automatiquement tout format de matrice de parité sans avoir recours à la modification du programme selon le format de la matrice à manipuler, à travers une petite interface graphique très facile à utiliser. La figure (a_4.4**)** présente l'interface graphique réalisée.

Cette interface contient Sept composants :

- Un composant dit Décodage LDPC Stochastique : qui permet de faire la description VHDL demandée
- Un composant de calcul de la taille de la matrice H
- Un composant calcul du nombre de nœuds de variable
- Un composant calcul du nombre de nœuds de parité

- Un composant calcul du nombre du degré maximal des nœuds de variable
- Un composant calcul du nombre du degré maximal des nœuds de parité
- Une barre de menus

(a)



(b)

Fig. 4.4 - Présentation de l'interface d'exportation de la description VHDL du décodeur CSS

4.5.2 Principe de fonctionnement de l'interface Graphique du synthétiseur

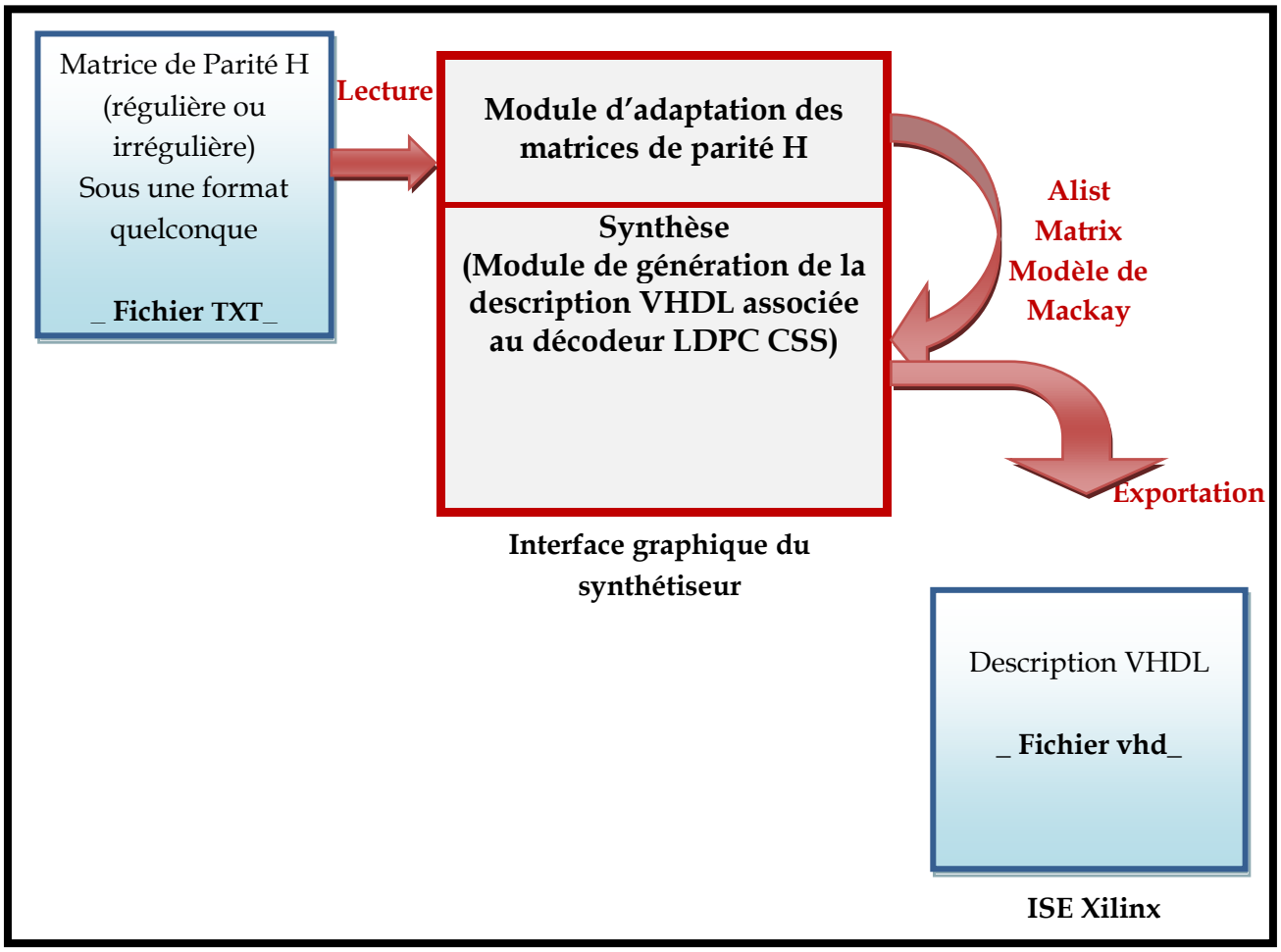


Fig. 4.5 – Principe de fonctionnement de l'interface du synthétiseur

L'interface graphique est composée de deux modules principaux, comme le montre la figure (4.5):

- **Un module d'adaptation du format de la matrice de parité :** qui permet de lire une matrice de parité du code LDPC à partir d'un fichier text (*.txt), qui peut être sous plusieurs formats. Son rôle est de modifier la format de cette matrice afin d'avoir un format de matrice H supportée par notre software, le format adapté est le modèle Alist Matrix proposé par Mackay
- **Un module de création de la description VHDL du Décodeur CSS étudié :** une fois la matrice traitée, elle sera injectée dans ce module, où la description

VHDL appropriée sera établie, puis exportée sous forme d'un fichier VHDL (*.vhd) vers ISE Xilinx ou System Generator.

4.5.3 Structure des fichiers source et destination

Dans ce qui suit, nous allons présenter pratiquement les formats des fichiers source et destination à manipuler et le fonctionnement pratique de notre synthétiseur.

- Alist Matrix de Mackay : Format de la matrice de parité que nous avons choisi est le suivant :

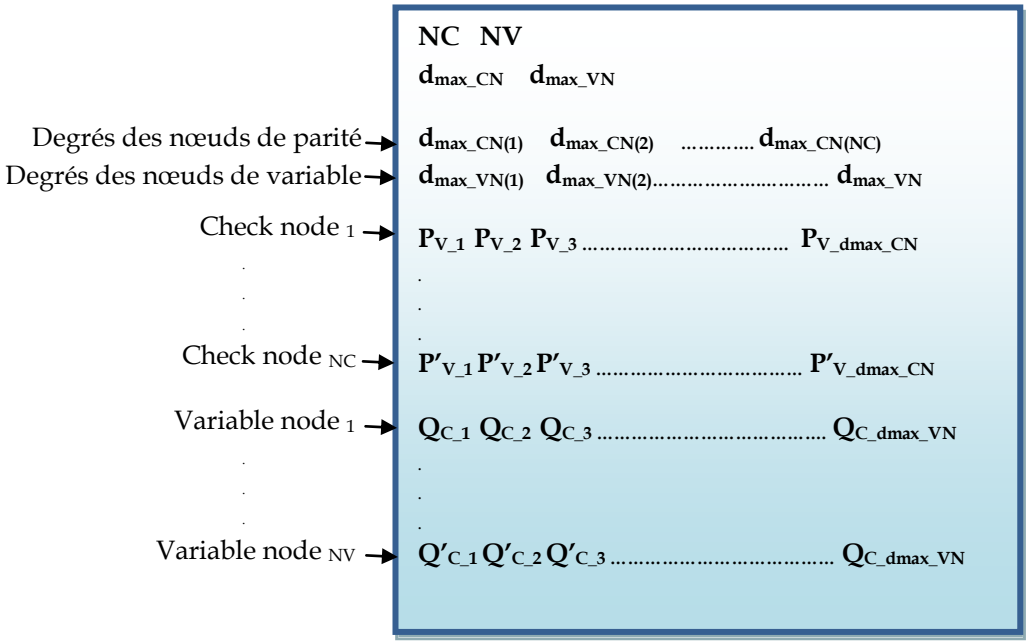


Fig. 4.6- Forme d'une matrice Alist régulière dmax_CN (dmax_VN) est le même degré de tous les nœuds de parité (variable)

Avec :

- **NC, NV :** nombre des nœuds de parité et des nœuds de variables respectivement
- **dmax_CN, dmax_VN :** degré maximum des nœuds de parité et des nœuds de variables respectivement
- **P_{V_i} :** position du nœud de variable connecté au nœud de parité correspondant à cette ligne, avec i = [1 : dmax_CN]

- Q_{C_j} : position du nœud de parité connecté au nœud de variable correspondant, avec j = [1 : dmax_VN]

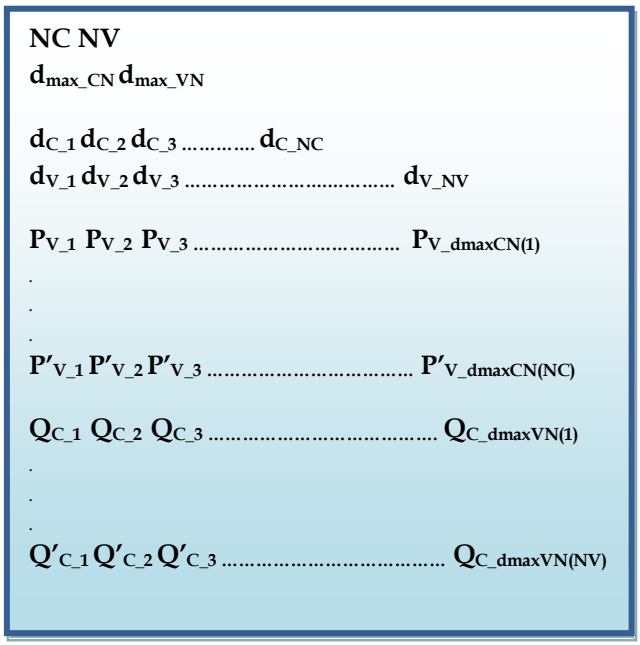
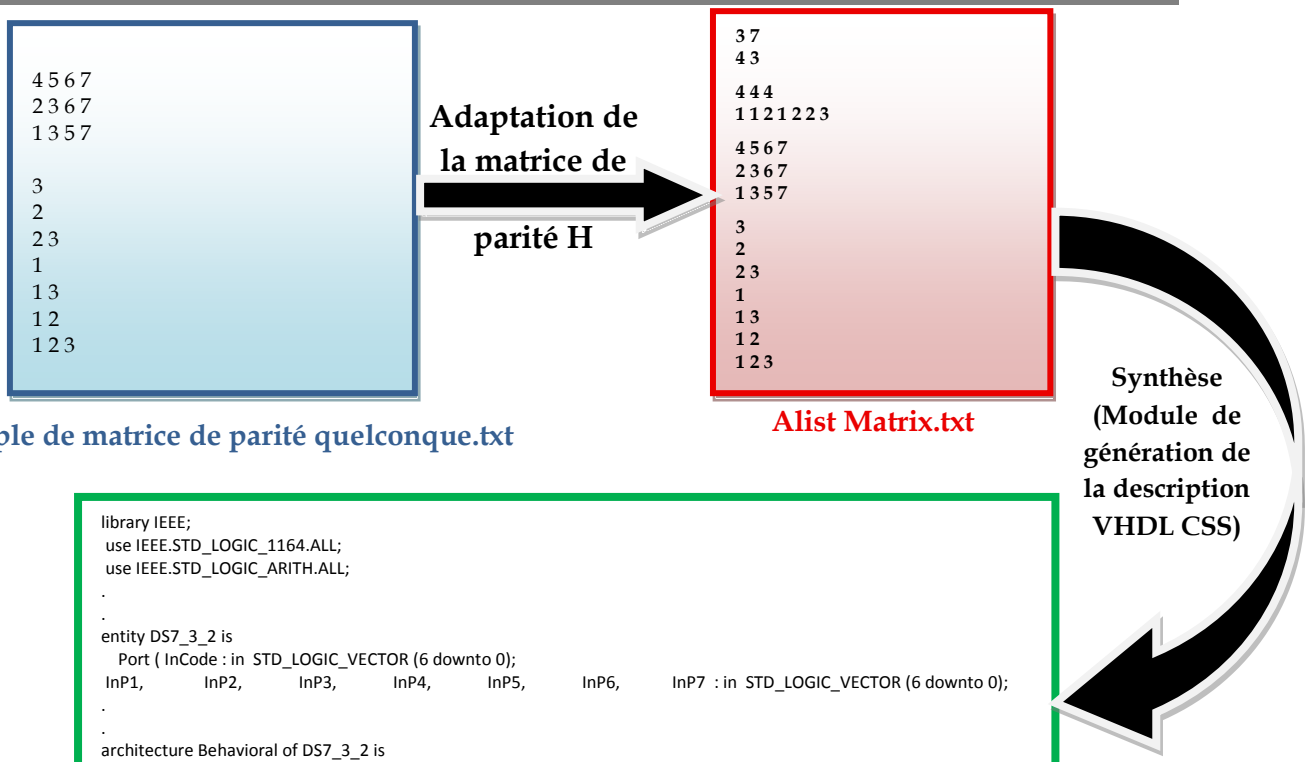


Fig. 4.7- Forme d'une matrice Alist irrégulière les degrés des nœuds de parité (variable) n'est pad identique pour tous les nœuds de parité (variable)

Avec:

- d_{max_CN}=Max(d_{max_CN}(i)), i=[1 : NC]
- d_{max_VN}=Max(d_{max_VN}(i)), i=[1 : NV]
- d_{C_1}, d_{C_2}, d_{C_3}, ... d_{C_NC} : degrés des nœuds de parité 1,2,3 …..NC respectivement
- d_{V_1}, d_{V_2}, d_{V_3}, ... D_{V_NV} : degrés des nœuds de variable 1,2,3 …..NV respectivement

Pour mieux illustrer le principe de fonctionnement présenté par la figure (4.5), nous allons présenter le traitement d'une matrice de parité de petite taille de Hamming H(3,7) :



Exemple de matrice de parité quelconque.txt

Alist Matrix.txt

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
.
.
entity DS7_3_2 is
   Port ( InCode : in  STD_LOGIC_VECTOR (6 downto 0);
 InP1,          InP2,          InP3,          InP4,          InP5,          InP6,          InP7  : in  STD_LOGIC_VECTOR (6 downto 0);
.
.
architecture Behavioral of DS7_3_2 is

signal RGE_out  : std_logic_VECTOR (15 downto 0); -- entre le GENERATEUR et Comparateurs
-- Signal pour CE_out
 signal CE_signal  : std_logic;
 .
.
  -- V Nodes

V1 :          DS_VN_005
  port map(                                                State1=>VN1_CN2,
                                                VNin1=>CN2_VN1,
               In_Code=>InCode(0),
               LD_Code=>LdCode,
                                                VNout=>VNout1,
                                                CLK=>C,
              CE=>CtrDec_CEall,
                                                A=>RGE_out(1),
                                                Pch=>CMP_VN1);


V2 :          DS_VN_005
  port map(                                                State1=>VN2_CN2,
                                                VNin1=>CN2_VN2,
.
.
  -- C Nodes

C1 :        DS_CN1
              port map (   CNout=>CN1_NLDPC,
                                                CNin1=>VNout1,
                                                CNout1=>CN1_VN1,
                                                State1=>VN1_CN1);
.
.
C2 :        DS_CN1
              port map (   CNout=>CN2_NLDPC,
 .                                              CNin1=>VNout1,
 .                                              CNout1=>CN2_VN1,
                                                State1=>VN1_CN2);
end Behavioral;
```

Description VHDL.vhd

Fig. 4.8- Présentation du traitement d'une Matrice de Parité H(3,7)

4.6 Test et validation

Une fois la description VHDL est réalisée, elle doit être testée pour évaluer les performances du décodeur à implémenter. Pour ce fait, nous avons utilisé une plateforme de test et d'évaluation réalisée sous Simulink de Matlab, que nous avons d'abord adapté aux contraintes posées par nos données.

Ce système se base principalement sur la comparaison entre l'information originale reçu à partir du canal de transmission et l'information reçu après décodage par le décodeur CSS.

4.7 Implémentation et résultats

Afin d'évaluer la performance de l'algorithme proposé, un code LDPC (48,24) est implémenté sur Xilinx Virtex-6 VLX 240T FPGA, avec et sans décodage CSS.

L'approche de décodage stochastique différé (DS : Delayed Stochastique) est utilisée comme un processeur du nœud de variable stochastique avancé. En comparaison avec TFM et MTFM, l'algorithme DS réduit la complexité matérielle pour les codes LDPC de longue et moyenne taille avec une perte de performance très réduite.

La figure (4.9) présente les résultats d'implémentation du BER (taux d'erreur de bits) du décodeur stochastique (48,24) étudié. Un maximum de cycle de décodage de 4000 est choisi.

La technique CSS fourni un BER 100 fois meilleure que les techniques courantes, à E_b/N_0=4,5 dB. Il est clair que l'approche DS courante ne peut pas délivrer de bonne performance avec un code LDPC court contrairement à l'approche proposé dans [57].

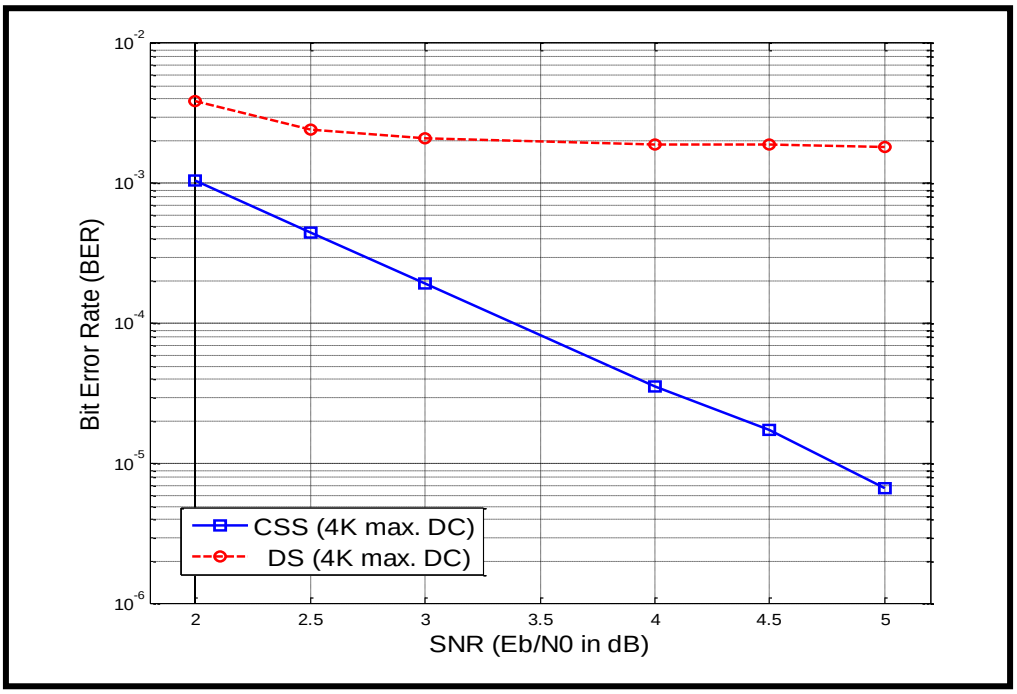


Fig. 4.9- Comparaison de la performance entre le décodeur CSS (48,24) proposé et le décodeur DS standard

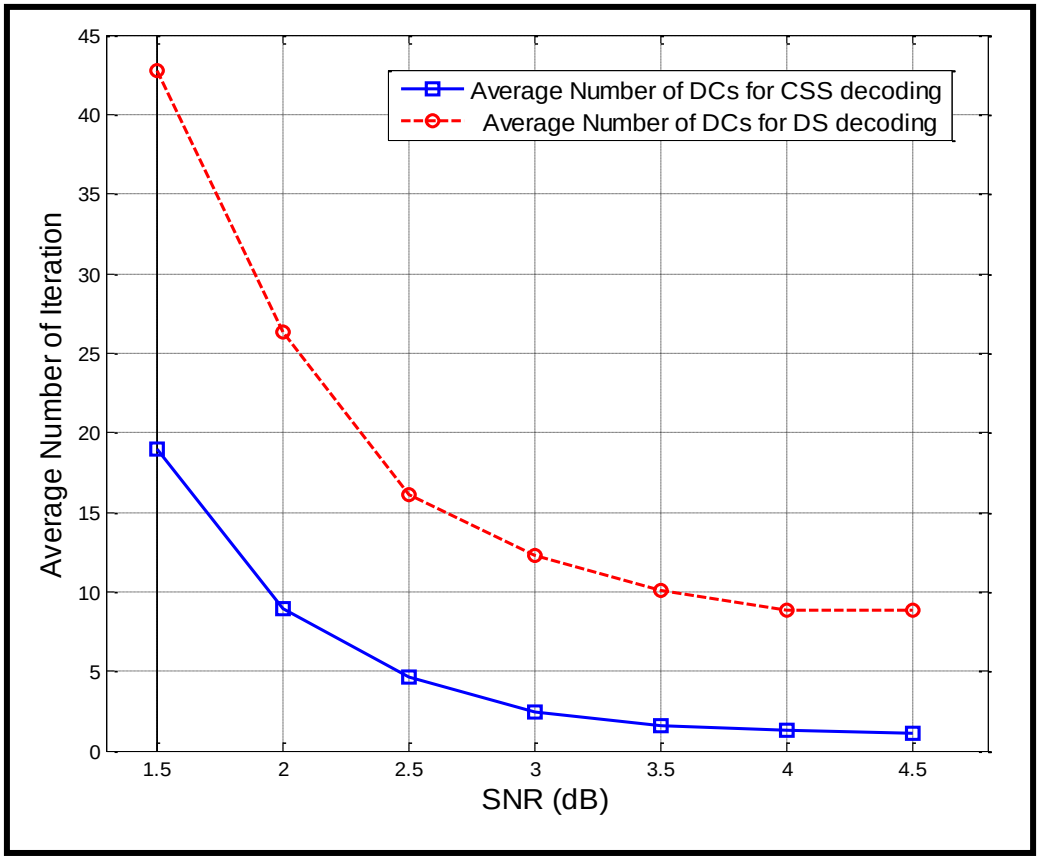


Fig. 4.10- Nombre d'itérations du décodage DS et du décodage CSS proposé dans [57]

En plus de l'amélioration de convergence, l'algorithme CSS réduit la latence du processus de décodage. Il améliore la vitesse de Sept fois plus rapide que les méthodes courantes à E_b/N_0=4,5 dB.

Le nombre moyen des itérations du décodeur diminue en augmentant le SNR. Il est autour de 1,06 cycles de décodage à E_b/N_0=5.0 dB.

Dans l'ordre de montrer l'amélioration de l'approche proposée, les résultats du décodeur LDPC (48,24) sont comparés à l'algorithme Offset Min Sum (OMSA), et à l'algorithme Sum Product (SPA) pour un code LDPC (2048,1723). Les résultats d'implémentation du décodeur stochastique prouvent que le décodage stochastique proposé peut dépasser les algorithmes Offset Min Sum (OMSA) et Sum Product (SPA).

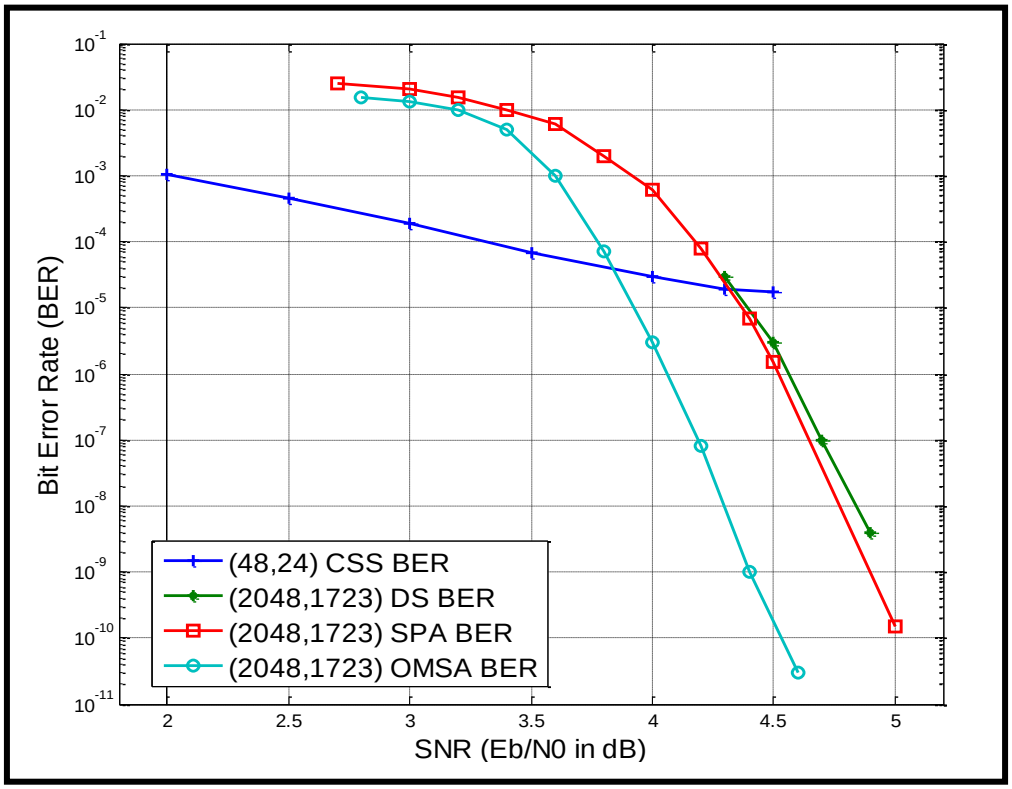


Fig. 4.11- Comparaison de la performance du décodage CSS (48,24), DS (2048,1723), SPA (2048,1723) et OMSA (2048,1723)

La figure (4.11) indique que le décodeur CSS (48,24), fourni un BER meilleure que l'algorithme Offset Min Sum (OMSA) (2048,1723), au dessous de E_b/N_0=3,9 dB. Et un BER meilleure que l'algorithme Sum Product (SPA) au dessous de E_b/N_0=4.3 dB.

Conclusion

Nous avons étudié une approche de décodage LDPC stochastique complètement parallélisable, qui peut devancer l'algorithme Sum Product (SPA) et l'algorithme Min Sum (MSA) à virgule fixe. Cet algorithme est appelé « Controlled Start-up Stochastic Decoding ». L'amélioration est obtenue en introduisant un processus d'initialisation simple et astucieux des nœuds de variable. Ainsi la performance est largement améliorée avec une réduction de cycle de décodage.

Conclusion générale

Dans ce mémoire nous avons présenté une nouvelle approche de décodage LDPC stochastique appelée « Décodage LDPC stochastique à initialisation contrôlée » (Controlled Start-up stochastique LDPC : CSS), proposé en 2013 par Maamoun et all. dans [57].
Cette approche est basée sur l'algorithme DS « Delayed Srochastic decoding » proposé par Ali Naderi et all. dans [69].
Avec une initialisation simple et astucieuse des Nœuds de variable, la complexité de l'architecture matérielle du décodeur est considérablement réduite, en plus la vitesse du traitement est meilleure par rapport au DS classique, TFM et MTFM, mais avec une légère perte de performance qui est très négligeable.

Ce décodeur présente donc, un bon compromis performance / simplicité.

Notre objectif était l'étude d'implémentation du décodeur CSS sur FPGA, dans ce cadre nous avons élaboré une plateforme permettant de générer la description VHDL du CSS automatiquement par Matlab, cette description est interprétable par ISE Xilinx et System Generator.

Cette technique nous a permis d'éviter les erreurs éventuelles dues à l'écriture ligne par ligne de la description VHDL du décodeur ainsi que de gagner du temps en générant automatiquement la description VHDL appropriée. Notons que pour des matrices de grande taille par exemple (1723,2048**)** nous aurons au minimum 2048 connexion entre VNs et CNs si nous considérons que tous les VNs sont de degré 1, s'ajoutent les autres signaux nécessaires du décodeur, nous aurons à manipuler donc des milliers de lignes de description VHDL à écrire, ce qui ne s'avère pas du tout évident.

Bibliographie :

[1]: Jean-Baptiste Doré, «Optimisation conjointe de codes LDPC et de leurs architectures de décodage et mise en oeuvre sur FPGA», Renne France, Thèse de doctorat, 2007

[2]: Dan Dechene & Kevin Peets, « SIMULATED PERFORMANCE OF LOW-DENSITY PARITY-CHECK CODES », LAKEHEAD UNIVERSITY CANADA, Article, 2006

[3]: François Verdier, «Conception d'architectures embarquées: des décodeurs LDPC aux systèmes sur puce reconfigurables», Université de CergyPontoise(Pays), Thèse de doctorat, 2006

[4]: C.ALEXANDRE, « Circuits numériques _Partie 1 », Polycopié de cour Electronique, Conservatoire nationale de l'art et métiers, version du 19 Janvier 2004

[5] : TISSERAND, Arnaud; (Décembre 2003) "Introduction aux circuits FPGA". INRIA LIP Arénaire, Séminaire MIM.

[6] : N. JULIEN, «Circuits logiques
programmables», Université de Bretagne Sud,Livre, 1999

[7]: Jason Kwok-San Lee, Benjamin Lee, Jeremy Thorpe, Kenneth Andrews, Sam Dolinar, Jon Hamkinst, « Design and Implementation of a Structured LDPC Decoder on FPGA », California Institute of Technology USA

[8]: Keren PARNEL & Nick MEHTA, «Xilinx; "Programmable Logic Design Quick Start Handbook», Fourth Edition ISE 5.1i, Juin 2003

[9]: DUTRIEUX. L & DEMIGNY. D.; «Logique programmable», Paris, Editions (1997)

[10]: Philippe LECARDONNEL & Philippe LETENNEUR, « Introduction à la Synthèse logique V.H.D.L. », Lycée Julliot de la Morandière – GRANVILLE, Manuel, 2001

[11]: Etienne Messerli , « Manuel VHDL synthèse et simulation Version partielle septembre 2007 (Chapitres : 1, 2, 3, 4, 6, 7, 9, 10, 15) », Haute école d'Ingénierie et de Gestion du Canton de Vaud Suisse, Manuel Version 6-a, 2007

[12] : Lycée Felix Le Dantec – Lannion, « COURS : le langage VHDL combinatoire», Renne France, 2011/2012

[13] : N.Nolhier, « VHDL Support de cours», Université Paul Sabatier 1997

[14] : Hervé Le Provost, « Initiation au langage VHDL », Livre, Université de Marne-La-Vallée France, 2008

[15] : T. BLOTIN, « Le language de description VHDL », Lycée Paul-Eluard 93206 SAINT-DENIS

[16]:« Cours VHDL FPGA», TRS 511

[17]: Clement Farabet, Cyril Poulet, Jefferson Y. Han & Yann LeCun, « CNP: AN FPGA-BASED PROCESSOR FOR CONVOLUTIONAL NETWORKS », New York University USA

[18]: C.ALEXANDRE, « Circuits logiques programmables », cours Electronique B9, Conservatoire National des Arts et Métiers, 2005

[19]: Emilie HERAULT, « FPGA et VHDL », Université Savoie

[20]: Patrice NOUEL, « Conception de circuits et langage VHDL modélisation et synthèse », ENSEIRC Bordeau

[21]:Sébastien SNAIDERO, «Modélisation multidisciplinaire VHDL-AMS de systèmes complexes : vers le Prototypage Virtuel», Université Louis Pasteur Strasbourg I, Thèse de Doctorat, 2004

[22]: Clément Farabet,Cyril Poulet and Yann LeCun, «An FPGA-Based Stream Processor for Embedded Real-Time Vision with Convolutional Networks», New York University

[23]: Ernest Jamro, «Parameterised automated generation of convolers implemented in FPGAs», University of mining and metallurgy Poland, Thèse de Doctorat, 2001

[24]: Xilinx tutoriel_xcn12023

[25]: J.M, Language VHDL, cour, université Paris Sud

[26]: Tomas Martinek & Lukas Sekanina, « An Evolvable Image Filter : Experimental Evaluation of Complete Hardware Implémentation in FPGA, University de Bozetechova, république chèque, Article

[27]: ALAIN HORÉ, «TRAITEMENT DES IMAGES BIDIMENSIONNELLES A L'AIDE DES FPGAs», Université de Québec, Mémoire de la maîtrise en ingénierie, 2005

[28] : IRINA ADJUDEANU, «Codes correcteurs d'erreurs LDPC structurés», UNIVERSITÉ LAVAL QUÉBEC, Mémoire de maitre en science, 2010

[29]: Frédéric GUILLOUD_«Generic Architecture for LDPC Codes Decoding» _TÉLÉCOM PARIS_Thèse Doctorat_2004

[30]: Erick Amador, « Aspects of Energy Efficient LDPC Decoders », thèse de doctorat, Telecom Paris Tech, 2011]

[31]: David Hayes, «FPGA implementation of a Flexible LDPC decoder», University of Newcastle, Australia, A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Telecommunication Engineering, 2008

[32]: Mathieu Cunche, «Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC», UNIVERSITE DE GRENOBLE France, Thèse de doctorat, 2010

[33]: Bilal SHAMS, «Les codes LDPC non binaire de la nouvelle génération», Université de Cergy –Pontoise, Thèse de doctorat, 2010

[34]: Valérian MANNONI, «Optimisation des codes LDPC pour les communications multi-porteuses», Université de Reims Champagne-Ardenne, Thèse de doctorat, 2004

[35]:Shiva Kumar PLANJERY, «Iterative Decoding Beyond Belief Propagation for Low-Density Parity-Check Codes», Université Cergy-Pontoise, Thèse de doctorat, 2012

[36]: Erick Amador, «Aspects of Energy Efficient LDPC Decoders», Université TELECOM ParisTech , 2011,

[37]: Gabi Sarkis, «Stochastic Decoding of LDPC Codes over GF(q)», Université de McGill Montréal Québec, 2009,

[38]: Mohammad M. Mansour & Naresh R. Shanbhag, «High-Throughput LDPC Decoders», IEEE, 2003

[39]: Mohammad M. Mansour & Naresh R. Shanbhag, «Low-Power VLSI Decoder Architectures for LDPC Codes», University of Illinois at Urbana-Champaign,année

[40]: François Verdier,«Architectures de décodeurs LDPC et
Systèmes sur puce reconfigurables», Université de CergyPontoise, Journée «IA embarquée », 2007

[41]: François VERDIER, David DECLERCQ, Jean-Marc PHILIPPE,«Parallélisation et implantation FPGA d'un décodeur LDPC», Laboratoire ETIS Cergy-Pontoise Cedex,

[42]: F. GUILLOUD,E. BOUTILLON, «Architecture générique de décodage de code LDPC», Université de Bretagne Sud, Séminaire

[43]:T. Mohsenin and B. M. Baas, «Split-Row: A Reduced Complexity, High Throughput
LDPC Decoder Architecture»,

[44]: Hemesh Yasotharan, Anthony Chan Carusone, «A Flexible Hardware Encoder for stematic
Low-Density Parity-Check Codes», University of Toronto Canada, article

[45]: Alex Balatsoukas-Stimming, « Decoding of LDPC codes using the Sum-Product algorithm for the AWGN channel with BPSK modulation », Technical University of Crete, Article, 2009

[46]: Bernhard M.J. Leiner, « LDPC Codes – a brief Tutorial », 2005

[47]: Matthew Weiner,Borivoje Nikolić & Zhengya Zhang, « LDPC Decoder Architecture for High-Data Rate Personal-Area Networks », University of California & University of Michigan USA, Article IEEE, 2011

[48]: Frédéric Guilloud, Emmannuel Boutillon, Jacky Tousch & Jean-Luc Danger, « Generic description and synthesis of LDPC decoder », I EEE Transactions on Communications Vol. 55, n°11 (2007) pp 2084 – 2091, 2007

[49]: Guido Masera, Federico Quaglio & Fabrizio Vacca, « Implementation of a Flexible LDPC Decoder », IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS

[50]: Ahmad Darabiha, Anthony Chan Carusone & Frank R. Kschischang, « Power Reduction Techniques for LDPC Decoders », IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 43, NO. 8, 2008

[51]: C. Studer-N. Preyss, C. Roth & A. Burg, « Configurable High-Throughput Decoder Architecture for Quasi-Cyclic LDPC Codes », Zurich- Switzerland

[52]: Chen, Yongmei Dai & Zhiyuan Yan, « Partly Parallel Overlapped Sum-Product Decoder Architectures for Quasi-Cyclic LDPC Codes », Lehigh University USA

[53]: Frédéric GUILLOUD, Emmanuel Boutillon & Jean-Luck DANGER, « Decodage des codes LDPC par l'algorithme λ-Min, Université de Bretagne Sud & télécom Paris,

[54]: Kiran Gunnam, « LDPC Decoding: VLSI Architectures and Implementations », Santa Clara, CA, Flash Memory Summit, 2012

[55]: Paul H. Siegel, «An Introduction to Low-Density Parity-Check Codes » University of California, San Diego, Présentation PPT, 2007

[56]: Quang Trung DONG, «Le principe du calcul stochastique appliqué au décodage des turbocodes : conception, implémentation et prototypage sur circuit FPGA», Université Télécom Bretagne, Thèse de doctorat, 2011

[57]: M.Mamoune, R.Bradaï et all, «Controlled Start-up Stochastic Decoding of LDPC Codes», Université de Blida, Article, 2013

[58]: Warren J. Gross & Vincent C. Gaudet et all**,** «Stochastic Implementation of LDPC Decoders», Université McGill et université de Alberta Canada , Article, 2005

[59]: Chris Winstead**,** Anthony Rapley et all, «Stochastic Iterative Decoders», Université Utah State USA & Université de Alberta Canada, Article

[60]: Erbao LI, « Décodeurs Haute Performance et Faible Complexité pour les codes LDPC Binaires et Non-Binaires », Université de Cergy-Pontoise, thèse de doctorat, 2012

[61] : Ludovic Danjeany, David Declercq et all. , « Décodeurs multi-niveaux pour le décodage quasi-optimal des codes LDPC », Article, MajecSTIC 2010 Bordeaux, France, 2010

[62]: Anthony C. Rapley, Chris Winstead, Vincent C. Gaudet, and Christian Schlegel «Stochastic Iterative Decoding on Factor Graphs » Edmonton, Alberta, CANADA.

[63] : Saeed Sharifi Tehrani, Warren J. and Shie Mannor, « Stochastic Decoding of LDPC Codes» *Member, IEEE,* Article, 10, OCTOBER 2006

[64]: Saeed Sharifi Tehrani, Shie Mannor and Warren J. Gross « an area-efficient FPGA-based architure for fully-parallel stochastic LDPC decoding », McGill University, article 2007.

[65]: Lara Dolecek, Zhengya Zhang and all « Analysis of Absorbing Sets and Fully Absorbing Sets of Array-Based LDPC Codes », *Member, IEEE*, article 1, JANUARY 2010.

[66]: Saeed Sharifi Tehrani, Shie Mannor and all « Fully Parallel Stochastic LDPC Decoders », Member, IEEE, article 11, NOVEMBER 2008.

[67]: Saeed Sharifi Tehrani, Shie Mannor and Warren J. Gross, « A fully-parallel stochastic LDPC decoders », McGill University, article, 2008.

[68]: Saeed Sharifi Tehrani,Ali Naderi & all. « Majority-Based Tracking Forecast Memories for Stochastic LDPC Decoding», McGill University, article, 2011.

[69]: Ali Naderi, Shie Mannor & all. « Delayed stochastic Decoding of LDPC codes », McGill University, article, 2011.

Webographie:

[W1]: http://s1.e-monsite.com/2009/09/11/2745497chaine-pdf.pdf

[W2]: http://en.wikipedia.org/wiki/Decoding

[W3]: http://www.ni.com/white-paper/8043/fr

[W4]: http://michel.hubin.pagesperso-orange.fr/physique/microp/chap_mp5.htm

[W5]: http://www.xilinx.com/univ/beginnersbookjune2003ver2.pdf> consulté le 28 août 2004