RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'électronique

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

Implémentation sur FPGA d'un Turbo Décodeur SOVA pour le Wimax

ADNANE Ayoub

Sous la direction de
M. Mohamed Oussaid TAGHI

Présenté et soutenu publiquement le 01/07/2019

Composition du Jury :

Président	M .Hicham BOUSBIA-SALAH	Dr	ENP
Promoteurs	M .Mohamed Oussaid TAGHI	Mr	ENP
Examinateur	M .Boualem BOUSSEKSOU	Mr	ENP

ENP 2019

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'électronique

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

Implémentation sur FPGA d'un Turbo Décodeur SOVA pour le Wimax

ADNANE Ayoub

Sous la direction de
M. Mohamed Oussaid TAGHI

Présenté et soutenu publiquement le 01/07/2019

Composition du Jury :

Président	M .Hicham BOUSBIA-SALAH	Dr	ENP
Promoteurs	M .Mohamed Oussaid TAGHI	Mr	ENP
Examinateur	M .Boualem BOUSSEKSOU	Mr	ENP

ENP 2019

Dédicace

Je dédie ce modeste travail
À mes chers parents
À ma sœur et mes frères
À tous mes amis
Aussi bien à tous ceux qui m'ont aidé, je cite particulièrement Feriel

Remerciement

Tout d'abord, je remercie le bon DIEU, le tout puissant, pour m'avoir donné la santé, le courage, la patience, la volonté et la force nécessaires, pour venir à bout de toutes les difficultés que j'ai dû croiser tout le long de mon chemin d'études.

Je remercie mes parents, qui m'ont soutenu tout au long de mes études.

Je tiens à exprimer mes vifs remerciements à M. TAGHI Mohamed qui m'a bien aidé dans la réalisation de mon PFE, ainsi que pour ses remarques et ses encouragements qui m'étaient de grande valeur.

Je remercie également les membres du jury pour avoir pris la peine d'évaluer ce modeste travail.

Enfin, j'adresse aussi mes remerciements à tous ceux qui ont contribué à ma formation de près ou de loin, depuis les premières lettres d'alphabet.

ملخص

تركز هذه الأطروحة على تنفيذ وحدة فك ترميز تربو تكرارية تستند إلى خوارزمية Viterbi ذات الإنتاج اللين (SOVA) في سياق الاتصالات اللاسلكية المتروبولية (WiMAX). تعد خوارزمية SOVA واحدة من عديد التقنيات لفك تشفير الكود التلافيفي المستخدمة في كثير من الأحيان في تشفير القناة.

وحدة فك الترميز المستخدمة هي وحدة فك ترميز تكرارية مع تقدير كمي مرن، ويتم تحسينها بتقريب الاحتمال الخلفي (APP). مع هذا النهج، يمكننا التوفيق بين الأداء في تصحيح الخطأ، الكمون، وتشغيل بسرعة عالية. يهدف النموذج الأولي إلى التحقق من صحة نتائج المحاكاة، فضلاً عن دقة التصحيح التي يمكن تحقيقها على بطاقة Xilinx FPGA Nexys-2 لأنواع مختلفة من الأخطاء.

الكلمات المفتاحية: تصحيح الأخطاء ، SOVA، WiMAX، التشفير ، فك التشفير ، توربو ، HDL.

Abstract

This thesis deals with the implementation of an iterative Turbo decoder based on the Viterbi Soft Output Algorithm (SOVA) for metropolitan wireless communications (WiMAX). The SOVA algorithm is one of the many convolutional code decoding techniques often used in channel coding. The decoder used is an iterative decoder with soft quantification, it is improved by the approximation of the a posteriori probability(APP). Thanks to this approach, error correction performance, latency, and high throughput can be reconciled. The prototype aims to validate the simulation results, as well as the correction accuracy that can be achieved on the Nexys-2 Xilinx FPGA board for different types of errors.

Keywords : error correction, SOVA, WiMAX, Encoder, Decoder, Turbo.

Résumé

Ce mémoire porte sur l'implémentation d'un Turbo décodeur itératif basé sur l'algorithme de Viterbi a sortie douce (SOVA) dans le cadre des communications sans fils métropolitains (WiMAX). L'algorithme SOVA est une des nombreuses techniques de décodage des codes convolutifs utilisés très souvent dans le codage de canal. Le décodeur utilisé est un décodeur itératif à quantification souple, il est amélioré par l'approximation de la probabilité a posteriori (APP). Grâce à cette approche, on arrive à concilier les performances en correction d'erreurs, la latence, et le haut débit de fonctionnement. Le prototype vise à valider les résultats de simulation, ainsi que la précision de correction que l'on peut atteindre sur la carte Xilinx FPGA Nexys-2 et ceci pour différents types d'erreurs.

Mots clés : la correction des erreurs, SOVA, WiMAX, Codeur, Décodeur, Turbo.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ARQ	Demande de répétition automatique (Automatic Repeat Request).
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BMC	Branch Metric Calculator
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CRSC	Circular Recursive Systematic Convolution code
CTC	convolutional turbo code
DM	Decisions Memory
DTM	Deltas Memory)
DU	Decision Unit
DVB-S2	2nd Generation Digital Video Broadcast Spatial
DVB-T	Digital Video Broadcast Terrestrial
FIFO	First in first out
FPGA	Field programmable gate array
GSM	Groupe Signaux Multidimensionnels
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
LIFO	Last In First Out
LLR	Log Likelihood Ratio
LIFO	Institute of Electrical and Electronics Engineers
NRNSC	Non Recursive Non Systematic Code
NSC	Non Systematic Code
OFDMA	Orthogonal Frequency Division Multiple Access
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RSC	Recursive Systematic Code
SISO	Soft input soft output
SOVA	Soft Output Viterbi Algorithem
VHDL	Very high speed integrated circuits hardware description language
WIMAX	Worldwide Interoperability for Microwave Access

Introduction générale

Les communications sans fils connaissent un grand essor depuis deux décennies, plusieurs standards et procédés techniques ont été développés pour assurer la fluidité, la rapidité et la sécurité de la transmission de données multimédias. Dans cette épopée vers les hauts débits, le bloc de codage canal, responsable de la lutte contre les erreurs de réception dans la chaine de communication, a fait l'objet de plusieurs solutions techniques visant à garantir la robustesse face aux erreurs de transmission.

Les codes classiques ne permettent pas d'atteindre la limite de Shannon, d'autres systèmes de codages basés sur les codes convolutifs ont été développés afin d'atteindre les objectifs en terme de rapidité de transfert exigés par les nouveaux standards de communications sans fils. Cette nouvelle classe de codes convolutionnels appelés codes turbo, formés d'une concaténation en parallèle de codes convolutionnels récursifs et systématiques, permet d'approcher la limite de Shannon dans un canal gaussien. Cependant, leur comportement dans le canal radio mobile reste difficilement prévisible, du moins sous la forme initialement proposée.

La structure d'encodage turbo repose sur la concaténation en parallèle de deux codes convolutionnels récursifs systématiques circulaires (CRSC) identiques, reliés par un entrelaceur. Grâce à l'entrelacement, les turbo-codes paraissent aléatoires au canal, ce qui constitue une caractéristique dont le décodage peut tirer bénéfice. Le design de l'entrelaceur affecte les performances de ces codes dans le sens qu'il modifie leurs propriétés de distance.

Le décodage des turbo codes a fait l'objet d'intenses recherches qui ont proposé plusieurs algorithmes et procédés afin de réaliser des objectifs de performances en haut débit.

Les résultats de la recherche ont montré que le décodage itératif pouvait permettre d'atteindre les fréquences exigées dans les récents standards de communication pour que

les architectures et les éléments constitutifs des décodeurs soient conçus dans cette optique. Les performances de ces codes font que les turbo codes sont aujourd'hui adoptés dans plusieurs standard de communication sans fils, tels que le WiMAXWLAN802.16e [1].

L'objectif principal de ce projet est d'implémenter sur support reconfigurable un turbo décodeur itératif basé sur l'algorithme de Viterbi a sortie douce SOVA pour les communications sans fils métropolitains (WiMAX)

Organisation du document :

Pour réaliser l'objectif de notre travail, ce mémoire est organisé comme suit :

Le **premier chapitre**, décrit brièvement les concepts généraux liés à la théorie de l'information et au codage canal, une description détaillée de la chaîne de communication, en indiquant le rôle de chaque élément, suivi d'une définition du standard de réseau sans fil métropolitain (WiMAX/ IEEE 802.16) et finalement une présentation des codes convolutifs en focalisant sur leurs différentes représentations.

Dans le **second chapitre**, une large présentation du turbo-codage utilisé dans la norme IEEE 802.16 est donnée, en suite nous allons introduire les codes convolutifs circulaires CRSC que nous avons utilisés. Finalement nous décrivons d'une façon détaillée le turbo-code convolutif CTC retenu pour l'implémentation.

Le **chapitre troisième** est divisé en deux parties, dans un premier temps, une profonde description de l'algorithme de décodage a base du SOVA incluant les notations et outils mathématiques indispensables à la compréhension en précisant la valeur rajouté par ces outils à l'algorithme de Viterbi moderne. Tandis que la deuxième partie est consacrée pour l'application de l'algorithme SOVA dans le décodage itératif.

Le **quatrième chapitre** décrit en détails notre travail où nous illustrons l'architecture proposée et les modifications apportées.

Dans Le **dernier chapitre**, une présentation du support d'implémentation (la carte FPGA Nexys-2) est donnée, puis, expose les architecture des différents blocs issus de la synthèse. Nous terminons par une discussion du rapport de synthèse généré par ISE-Xilinx.

Une **conclusion** couronne ce document en énumérant les perspectives pour la suite de ce travail.

Chapitre 1

Généralités sur le codage canal

1.1 Introduction

Ce premier chapitre présente les principes et les concepts fondamentaux utiles pour la compréhension des turbo codes et le turbo décodage. La chaîne de communication est illustrée, le rôle et l'intérêt de chacun de ses blocs sont brièvement expliqués. Les rappels nécessaires sur la théorie codage, et les codes convolutifs sont également donnés à ce niveau. Le standard Wimax/IEEE 802.16, sa définition, et ses caractéristiques sont détaillés en fin de chapitre.

1.2 La chaine de communication numérique

Les systèmes de communication se composent de trois unités principales : l'unité d'émission, le support de transmission et l'unité de réception correspondante. Les signaux de communication transportés à travers le système portent les informations provenant de la source où elles sont générées. La transmission de l'information à travers un support non idéal provoque l'atténuation et la déformation du signal d'information en raison de la perte dans le canal et de la limitation de la bande passante.

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

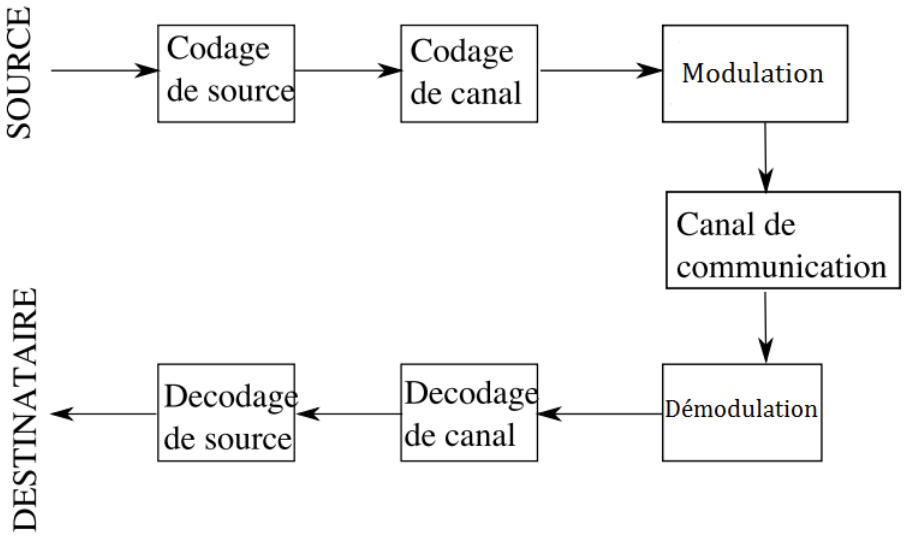

FIGURE 1.1 – Chaine de transmission numérique

1.2.1 Le codage de source

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

1.2.2 Codage canal

Le codage de canal sert à éliminer la redondance indésirable dans la séquence transmise afin de détecter et corriger les erreurs et les perturbations dus à la transmission, Cette opération se fait par l'ajout d'une redondance a l'information utile du message à transmettre. La redondance et l'information utile sont liées par une loi donnée. La sortie du codeur de canal est un mot de code à partir d'un code de canal, A la réception, le décodeur de canal exploite la redondance produite par le codeur dans le but de détecter,

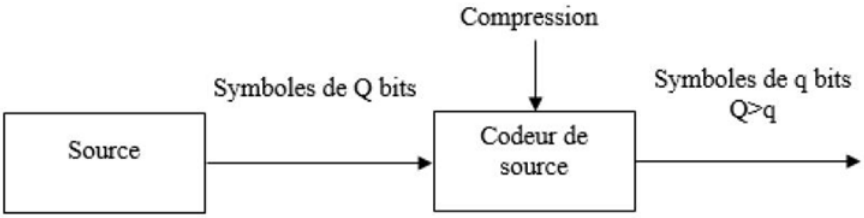

FIGURE 1.2 – Schéma simplifié d'un codeur de source

puis de corriger si c'est possible les erreurs introduites lors de la transmission[3].

1.2.3 La modulation

La modulation peut être considérée comme une conversion numérique analogique, préparant le flux de bits au canal de transmission qui est analogique. Initialement, le flux de bit est dans une représentation en bande de base, c'est-à-dire que ça fréquence est basse et que le changement de signaux se fait à une fréquence comparable à celle des symboles numériques représentés.

Même s'il est possible de transmettre le signal en bande de base, il est généralement translaté à une fréquence plus grande. La raison à cela est la possibilité d'utiliser différentes régions du spectre pour déférentes transmissions, ainsi que le fait que les hautes fréquences ont des longueurs d'ondes plus petites et nécessitent donc de plus petites antennes.
Plusieurs améliorations peuvent être apportées à la modulation :

- Modulation M-aire : plutôt que de coder chaque bit indépendamment, on les regroupe par bloc de $N(M = 2^N)$ A chaque symbole ainsi formé on fait correspondre une forme d'onde de la constellation. L'utilisation de constellation faisant correspondre à un symbole plusieurs bits permet d'améliorer l'efficacité spectrale (bits/sec/hertz). Les deux techniques les plus connues s'appellent M-QAM et M-PSK. La modulation M-aire peut être combinée au codage de canal pour donner des performances très intéressantes[4].

- Modulation multi-porteuses, l'OFDM (Onhogona1 Frequency Division Multiplexing) :

consiste à répartir l'information transmettre sur un grand nombre de porteuses chacune modulée à bas débit. Cela revient à faire une transforme de Fourier inverse discrète sur une série de signaux QAM ou PSK. C'est cette transformée qui est ensuite émise sur le canal.

1.2.4 Le canal de communication

Le canal de communication est le support physique permettant d'acheminer un message entre une source et un ou plusieurs destinataires. Au sens des communications numériques, et comme représenté dans la Figure 1.1, inclut le milieu de transmission (support physique entre l'émetteur et le récepteur : câble, fibre, espace libre,...), le bruit (bruit de fond, bruit impulsif, parasite, évanouissement, distorsion, défaut, panne), et les interférences (provenant des autres utilisateurs du milieu de transmission, de brouilleurs intentionnels ou non). Il existe plusieurs type de canaux, mais en théorie d'information, les canaux les plus utilises sont appelés canaux discrets.

Le canal de communication, caractérisé par sa réponse en fréquence, se comporte comme un filtre avec un ensemble de défauts qui le rendent non idéal. On assiste ainsi à :

- Une distorsion du signal suivant les fréquences. L'utilisation de la technique OFDM qui est robuste aux distorsions limite ce problème, et on rétablit le signal transmis par égalisation. Plusieurs modèles sont disponibles pour l'étude de ce phénomène, comme le canal de Rayleigh, le canal de Rice et le canal de Rummler[5].

- La limitation en bande passante du canal, impose un effet de filtre passe bas. Ceci entraîne des interférences inter symboles (liS), que l'on corrige par égalisation.

- Le canal étant un milieu sujet au bruit, ce dernier s'ajoute au signal. Dans sa forme la plus simple, on considérera un bruit blanc gaussien additif.

1.2.4.1 Le canal discret

Un canal discret est un système stochastique acceptant en entrée des suites de symboles définies sur un alphabet de sortie Y, relies par une loi de transition $P_{y/x}$ i.e. une matrice stochastique $M_{y/x}$:

$$M(y/x) = \begin{bmatrix} P_{y_1/x_1} & \cdots & P_{y_j/x_1} \\ \vdots & \ddots & \vdots \\ P_{y_1/x_k} & \cdots & P_{y_j/x_k} \end{bmatrix} \tag{1.1}$$

Différents modèles de canaux discrets existent dans la littérature. Selon le système de communication étudié, on choisit l'approche qui rend le plus fidèlement compte des phénomènes à analyser et du niveau de détail que l'on veut atteindre. Nous présentons, dans ce qui suit, les principaux modèles utilisés dans l'étude du codage de canal.

Le canal binaire symétrique BSC

Le canal binaire symétrique (BSC), présenté par le schéma de droite de la Figure 1.3, est le canal le plus simple qu'on puisse imaginer. Ce canal est caractérisé par des alphabets d'entrée et de sortie binaires, une probabilité d'erreur p et une matrice de transition $M_{x/y}$ donnée par la relation 1.2.

$$M_{Y/X(BSC)} = \begin{bmatrix} 1-P & 1 \\ 1 & 1-P \end{bmatrix} \tag{1.2}$$

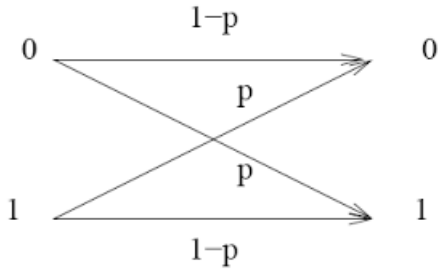

FIGURE 1.3 – Le canal binaire symétrique

Le canal à entrées binaires et bruit blanc additif

Ce canal est très important analytiquement dans l'étude des techniques de codage de canal et spécialement dans le cas du décodage à entrées souples. Les hypothèses de ce modèle sont :

- La source est binaire et idéale,

- Une modulation BPSK,
- le canal de transmission affecte les signaux par l'ajout d'un bruit indépendant du signal, stationnaire, gaussien et blanc avec une variance σ^2 et une densité spectrale bilatérale de $\frac{N_0}{2}$
- Le bruit introduit dans ce canal est modélisé par un signal aléatoire n(t) de moyenne μ et de variance σ^2, dont la distribution de probabilité suit la loi Gaussienne :

$$f_N(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(n-\mu)^2}{2\sigma^2} \tag{1.3}$$

  Tel que :

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

- Le démodulateur avec un filtre adapté est optimal,
- L'échantillonnage est supposé effectué aux instants adéquats.

On représente graphiquement ce canal par le diagramme de la figure 1.4. Les symboles de l'entrée sont binaires, mais les symboles de sortie prennent des valeurs dans l'espace des réels. On exprime la sortie suivant l'équation (1.4).

$$y(t) = as(t) + n(t) \tag{1.4}$$

Avec :

$y(t)$: Le signal reçu.
$s(t)$: Le signal transmis.
$s(t)$: Le bruit aléatoire additif.
a : Le facteur atténuation

1.2.5 Le canal radio mobile

La connaissance et la compréhension des caractéristiques du support de communication est indispensable pour aborder des travaux dans ce domaine. Les canaux radio mobiles sont considérés en particulier comme étant des canaux souffrant de nombreuses imperfections comme le mult-itrajet, l'effet Doppler, l'atténuation par parcours (Path Loss) et l'effet de masque(Shadowing). Ces facteurs perturbateurs peuvent affecter les informations transmises. Le signal reçu sera donc la somme de répliques atténuées, réfléchies, réfractées

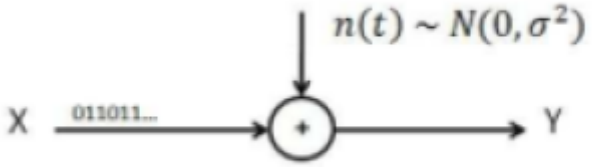

FIGURE 1.4 – Canal à entrée binaire perturbée par l'addition d'un bruit blanc gaussien

et diffractées du signal transmis. Donc l'optimisation de notre système de communication en prenant en compte ces imperfections devient vraiment primordiale [6].

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

1.2.5.1 Trajet multiples (Multipath)

La Propagation multi-trajets apparait comme conséquence de réflexion, dispersions, Atténuation de l'énergie du signal et diffractions par différents obstacles des ondes électromagnétiques émises. Cela a pour conséquence des retards à la réception, des changements de phases et des atténuations différentes [6]. Ces phénomènes sont causes par les obstacles naturels qui s'opposent à la propagation du signal radio comme les montagnes, les bâtiments, ... etc. Ces réflexions obligent le signal à suivre des chemins différents pour arriver au récepteur, ce qui donne naissance au phénomène de trajets multiples. La propagation multi-trajets apparait lorsqu'un signal émis par un émetteur prend plusieurs chemins pour arriver aux récepteurs. Le récepteur reçoit des versions retardées du signal émis et les voit comme des échos avec différents temps d'arrivé et d'amplitudes.

1.2.5.2 Effet Doppler

Quand la source et le récepteur se déplacent l'une par rapport à l'autre, la fréquence du signal reçue au récepteur n'est pas identique à celle de la source, la distance entre l'émetteur et le récepteur varie au cours du temps, on obtient donc un décalage fréquentiel [7]. L'effet Doppler représente ce décalage de fréquence. Cette différence entre la fréquence émise et reçue appelée fréquence Doppler peut s'écrire sous la forme :

$$f_d = \frac{v f_c \cos(\alpha)}{c} \tag{1.5}$$

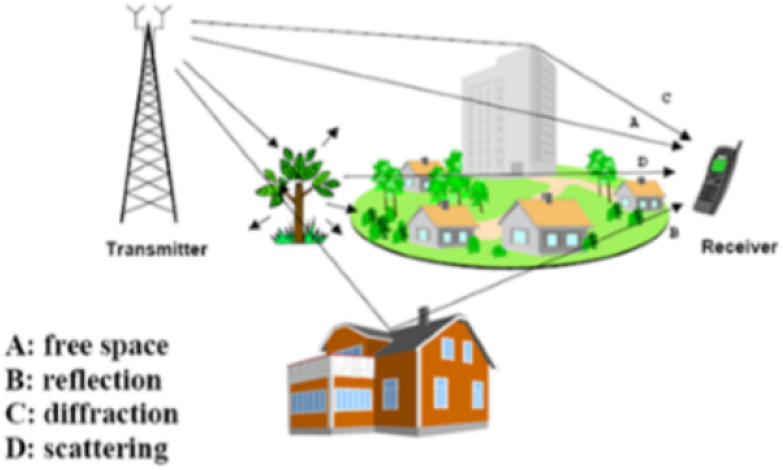

FIGURE 1.5 – Propagation par trajets multiples

tel que

f_d : Fréquence Doppler.

v : La vitesse de déplacement du récepteur.

c : La vitesse de propagation de l'onde électromagnétique dans le vide. récepteur.

α : est l'angle entre v (vitesse de déplacement) et k (direction de propagation du champ).

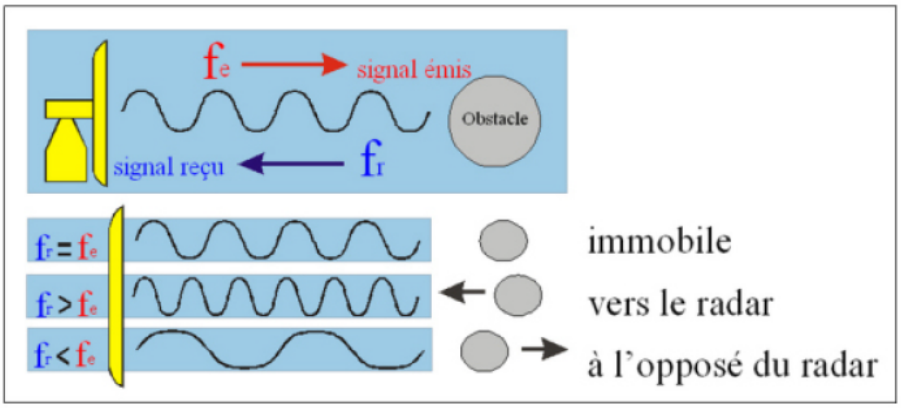

FIGURE 1.6 – Effet doppler

1.2.5.3 Effet de masquage (shadowing)

L'effet de masquage est un phénomène aléatoire plus local (sur quelques centaines de longueur d'onde), cause par l'obstruction des ondes qui se propagent, par de grands obstacles, tels que des collines, des édifices, des murs, des arbres ...etc., ce qui cause

une atténuation, plus ou moins importante, de la force du signal. Sa variation due à l'effet de masque est appelée évanouissement lent (slowfading) et peut être décrite par une distribution log-normale [6]. Les variations de la puissance reçue dues aux pertes par parcours et à l'effet de masque peuvent être neutralisées d'une manière efficace par le contrôle de puissance.

1.2.5.4 Pertes par parcours (pathloss)

Les pertes par parcours représentent l'atténuation que subit la puissance moyenne du signal transmis le long de la distance entre l'émetteur et le récepteur. En espace libre la puissance moyenne du signal est inversement proportionnelle au carré de la distance r^2. Ce pendant dans un canal radio mobile ou, en générale, il n'a pas de visibilité (no line of sight), la puissance moyenne est inversement proportionnelle à L (tel que $r^3 < L < r^5$) [6]. La figure1.7 résume tous les types d'évanouissement.

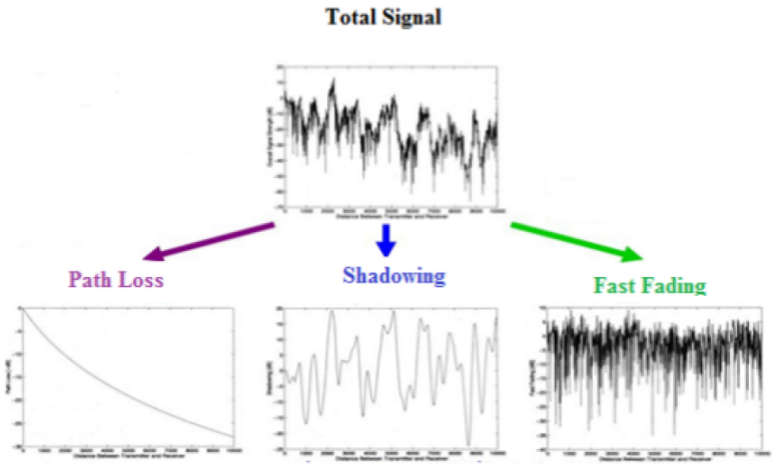

FIGURE 1.7 – Schéma récapitulatif des différents types d'évanouissement

1.2.6 Théorie de la capacité d'un canal

On définit la capacité d'un canal (exprimée en bit/s) comme le maximum de l'information mutuelle moyenne $I(X;Y)$, elle représente donc la plus grande quantité d'information pouvant transiter entre l'émetteur et le récepteur[8] , comme le montre la figure 1.8.

$$C = max_p I(X;Y) \tag{1.6}$$

On désigne par $H(X)$ l'entropie de la source qui représente la surprise moyenne ou la quantité d'information délivrée par une source d'information. Par contre,$H(X;Y)$ repré-

sente l'information requise pour supprimer l'ambigüité sur l'entrée.

$$H(x) = \sum_{i=1}^{n} P_i \log_2(P_i) \tag{1.7}$$

P_i : La probabilité d'apparition des lettres de l'alphabet de la source.

La capacité C s'exprime en Shannon par symbole ou bit par symbole. Il est également possible de l'exprimer en Shannon par seconde, on parle alors de capacité par unité de temps[9].Pour la distinguer de la capacité par symbole, on trouve généralement dans la littérature la notation suivante :

$$C_s = Cd_s \tag{1.8}$$

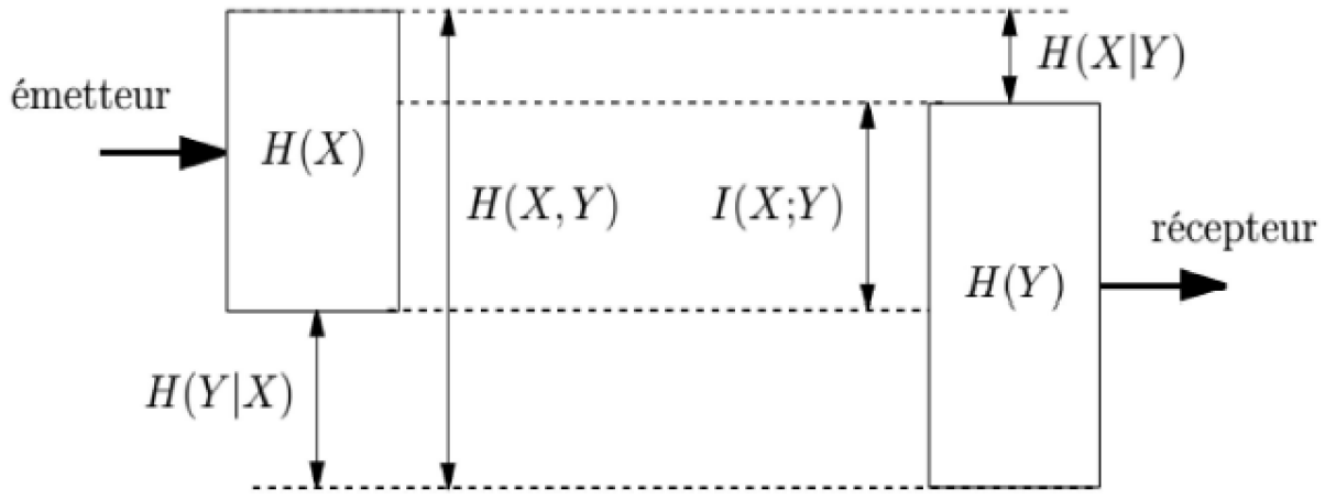

FIGURE 1.8 – Schéma représentatif de l'information mutuelle

Dans le cas d'une transmission sur deux voies en quadrature sur un canal gaussien, la capacité maximale est donnée par :

$$C = W \log_2 \left(1 + \frac{S}{N}\right) (Bit/s) \tag{1.9}$$

C : Capacité du canal en bits par seconde (bits/s),

W : Largeur de bande du signal (Hz),

S : Puissance moyenne reçue (Watt),

N : Puissance moyenne du bruit reçu (Watt) dans la bande W,

1.3 Le WIMAX

WIMAX est l'abréviation pour World Interoperability for Microwave Access .Il s'agit d'un standard de réseau sans fil métropolitain créé par les sociétés Intel et Alvarion en 2002 et ratifié par l'IEEE (Institute of Electrical and Electronics Engineer) sous le nom IEEE- 802.16.WIMAX est le label commercial délivré par le WIMAX Forum aux équipements conformes à la norme IEEE 802.16, afin de garantir l'interopérabilité entre deux standards de réseaux sans fils à savoir, HiperMAN proposé en Europe par l'ETSI (European Telecommunications Standards Institute) et 802.16 proposé par l'IEEE.
WIMAX a été étudié pour des connexions sans fils à haut-débit sur des zones de couverture de plusieurs kilomètres pour des usages en situation fixe ou en mobilité. WIMAX a le potentiel d'impacter toutes les infrastructures des télécommunications. Dans la téléphonie fixe, WIMAX Fixe (IEEE 802.16-2004 ou IEEE 802.16d) par sa configuration peut remplacer le réseau de collecte de l'opérateur téléphonique, le réseau de transport de la télévision par câble coaxial et fournir aussi les services d'un FAI (Fournisseur d'accès Internet). Dans sa variante WIMAX Mobile (IEEE 802.16e), WIMAX a le potentiel de remplacer le réseau de cellules (BTS, BSC).

WIMAX fixe, appelé aussi IEEE 802.16-2004 est prévu pour un usage fixe avec une antenne montée sur un toit, à la manière d'une antenne de télévision. Le WIMAX fixe opère dans les bandes de fréquence 2-11 GHz. La bande 2,5 GHz à 3,5 GHz, pour lesquelles une licence d'exploitation est nécessaire, 5.8 GHz est une bande libre [10].

Caractéristiques du WiMAX

Extensibilité

L'OFDMA est extensible sur lequel l'IEEE802.16e est basé fournit une bande passante extensible. Cette bande passante extensible permet une prise en charge dynamique de l'itinérance des utilisateurs sur différents réseaux. Ces réseaux peuvent avoir des allocations de bande passante différentes.

Qualité de service QoS

La couche MAC du WiMAX devrai supporter une variété d'applications avec différentes exigences de QoS telles que les applications basées sur le meilleur effort, les applications en temps réel et en temps non réel, les applications basées sur le débit binaire constant (CBR) et variable (VBR).

Mobilité

Le WiMAX peut prendre en charge de nombreux utilisateurs dans une zone de couverture allant jusqu'à 50 km. Afin de prendre en charge les applications mobiles, la station de base doivent introduire plusieurs fonctions de support de la mobilité dans le système WiMAX existant. Des mécanismes d'économie d'énergie devraient être utilisés. En outre, l'estimation plus fréquente des canaux et le contrôle de la puissance sont spécifiés aux fins de la mobilité.

Sécurité

Le WiMAX prend en charge les techniques de sécurité avancées et puissantes, telles que l'Advanced Encryption Standard (AES). Il précise également les procédures de sécurité utilisées pour authentifier et maintenir les clés de chiffrement privées. Ces clés de chiffrement privées sont utilisées pour chiffrer le trafic vers les voisins du premier saut ou vers la station de base[11].

Spécifications pour Le WiMAX Mobile

La technologie WiMAX (réseau WMAN) est basée sur deux principaux standards IEEE 802 .16d (WiMAX fixe) et IEEE 802.16e (WiMAX mobile). Le WiMAX mobile est caractérisé par :

- Son codage et modulation adaptative AMC (Adaptive modulation and coding).
- La demande de répétition automatique hybride H-ARQ (Hybrid- Automatic Repeat Request) et retour rapide CQICH (Fast Channel Feedback).
- Le support de la QPSK, 16QAM et 64QAM est indispensable dans le sens descendant alors que dans le sens ascendant, la 64QAM est optionnelle.
- L'utilisation des codes Convolutif (CC) et les Turbo Codes Convolutif (CTC), avec un taux de codage variable.

Ces dernières Spécifications ont été introduites dans la version mobile du Wimax pour améliorer la performance et la couverture. Le tableau 1.1 résume les différents codages et modulations qui peuvent être utilisés en Wimax Mobile. Notons que les codes et modulations optionnels en sens ascendant sont notés en italique.

Codes/Caractéristiques	Taux de codage	Modulation
CC	1/2, 2/3, 3/4, 5/6.	QPSK, 16QAM, 64QAM
CTC	1/2, 2/3, 3/4, 5/6.	QPSK, 16QAM, 64QAM

TABLE 1.1 – Codes et modulations supportés en WiMAX mobile.

1.4 Les codes convolutifs

Les codes convolutionnels ont été introduits en 1955 par Elias [12] comme alternatives aux codes en blocs. De ce fait, ils peuvent être considérés comme un cas particulier de ces derniers.
Ce type de codes ne cesse de s'imposer année après année, et constituent actuellement la famille la plus utilisée dans les systèmes de télécommunications fixes et mobiles. Ils s'appliquent, à la différence des codes en blocs, sur une séquence infinie de symboles pour générer une séquence infinie de symboles codés.

Les codes convolutionnels sont utilisés par exemple dans l'UMTS, et dans les systèmes cellulaires numériques utilisant le GSM. Ils sont aussi utilisés dans les communications par satellite, dans les réseaux sans fil locaux (LAN) et nombre d'autres applications. La principale raison de cette popularité est l'existence d'algorithmes de décodage efficaces. En particulier le très répandu l'algorithme de Viterbi à Sortie Souple qui est un décodeur qui utilise en générale le décodage ML. Il peut être implémenté avec un niveau de complexité matérielle raisonnable.

1.4.1 Définitions et notation

Paramètres du code

Le codeur de canal permet de générer un mot de code b de n bits à partir d'un mot d'information u de k bits. Ce code engendre donc m bits de redondance, avec $m = n - k$, appelés bits de parité,
Dans ce qui suit nous considèrerons les variables suivantes :

- Mot information à coder $u = (u_0, u_1, \ldots, u_{(k-1)})$ de longueur k ;
- Mot code émis $b = (b_0, b_1, \ldots, b_{(n-1)})$de longueur n ;
- Mot reçu à décoder $r = (r_0, r, \ldots, r_{(n-1)})$ de longueur n ;
- Mot décodé $\hat{u} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{(k-1)})$ de longueur k ;

Le mot décodé û peut être identique à u ou différent, selon que les erreurs soient cor-

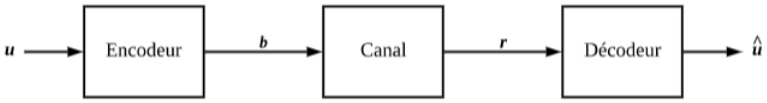

FIGURE 1.9 – État de l'information dans la chaîne de transmission

rigées ou pas.

Un code est dit systématique si les symboles de u apparaissent explicitement dans b.

Rendement du codage

En partant du principe que tous les symboles u_i de l'information peuvent prendre q valeurs, le nombre de mots informations possibles est donné par :

$$M = q^k \tag{1.10}$$

Sachant que le mot code de longueur n est plus long que le mot d'information de longueur k, le taux auquel l'information transmise à travers le canal est réduite est appelé le taux de codage du code :

$$R = \frac{\log_q(M)}{n} = \frac{k}{n} \tag{1.11}$$

$k = 1$, et $n = 3$; ce qui donne un taux de codage de $\frac{k}{n} = \frac{1}{3} \approx 0.3333$.

Poids et distance de Hamming

A chaque mot code $b = (b_0, b_1, \ldots, b_{(n-1)})$,peut être associée un poids $wt(b)$, défini comme le nombre de composantes non nulles $b_i \neq 0$: C'est le poids de Hamming [13].

$$wt(b) = |i : b_i \neq 0; 0 \leq i < n| \tag{1.12}$$

La distance entre deux mots b et $\hat{b}$ est donnée par la Distance de Hamming[13] :

$$dist(b,\hat{b}) = |i : b_i \neq \hat{b}_i; 0 \leq i < n|$$

La distance de Hamming traduit le nombre de composantes différentes entre b et $\hat{b}$, et mesure ainsi la proximité entre b et $\hat{b}$. Pour un code consistant en M mots codes $b_0, b_1, ..., b_M$, la distance de Hamming minimale est donnée par :

$$d = dist_{min} = \min_{\forall b \neq b'} dist(b; b') \tag{1.13}$$

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

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

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

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

1.4.2 Structure des codes convolutionnels

Il est aisé de considérer un code convolutionnel comme un code bloc linéaire décrit par une matrice génératrice binaire G de dimension $L * n$. Cette matrice peut être vue comme L matrices de dimension$k * n$ les éléments $g_0, g_1, \ldots, g_{(L-1)}$
Avec :

$$g_i = \begin{bmatrix} g_{i11} & \cdots & g_{i1n} \\ \vdots & \ddots & \vdots \\ g_{ik1} & \cdots & g_{ikn} \end{bmatrix} \tag{1.16}$$

On remarque que pour chaque k bit en entrée, on génère n bits en sortie d'où le rendement de ce code que l'on appelle aussi taux de codage $R = \frac{k}{n}$, L appelé la longueur de contrainte du code convolutionnel, et on définit la mémoire m qui est égale à $L + 1$.
On parle de convolution, puisque dans l'écriture de la séquence de bits b générée pour une suite de bits en entrée u, on a :

$$b = G.u \tag{1.17}$$

Et en développant le produit matriciel, pour chaque n bit de sortie on obtient un produit de convolution [5]. Prenons l'exemple $L = 3$, R=1/2 et $G = [111; 101]$. Le schéma de l'encodeur sera alors :

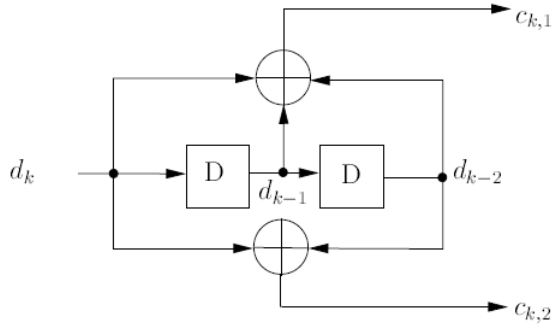

FIGURE 1.10 – Schéma d'un codeur convolutif pour $L = 2$,$K = 1$ et R=1/2

Les sorties binaires dépendent non seulement des k entrées binaires, mais aussi de l'état des registres à décalages. Il existe donc 2^k transitions possibles à partir d'un état du code à un instant donné. La complexité du codeur ne dépend ainsi que de la longueur de contrainte L et de k [5].

Présentations polynomiale

Considérons un code binaire classique (non-systématique et non-récursif). La connaissance de g_{jk} suffit alors à décrire le code. Les coefficients g_{jk} définissent donc n polynômes générateurs $g_k(D)$ en algèbre de D (Delay)[25] :

$$g_k(D) = \sum_{j=1}^{n} g_{jk} D^j \tag{1.18}$$

Prenons le cas du codeur défini en figure (1.11) Les sorties r_{1i} et $r_2 i$ s'expriment en fonction des données d successives de la manière suivante :

$$r_{1i} = d_i + d_{i-2} + d_{i-3} \tag{1.19}$$

Ce qui peut aussi s'écrire, via la transformée en D :

$$r_1(D) = g_1(D) d(D) \tag{1.20}$$

Avec $g_1(D) = 1 + D^2 + D^3$ premier polynôme générateur du code et $d(D)$ transformée en D du message à coder. De même, le second polynôme générateur est $g_2(D) = 1 + D + D^3$. Ces polynômes générateurs peuvent aussi se résumer par la suite de leurs coefficients, respectivement (1011) et (1101), généralement notée en représentation octale, respectivement $(13)_{octal}$ et $(15)_{octal}$.

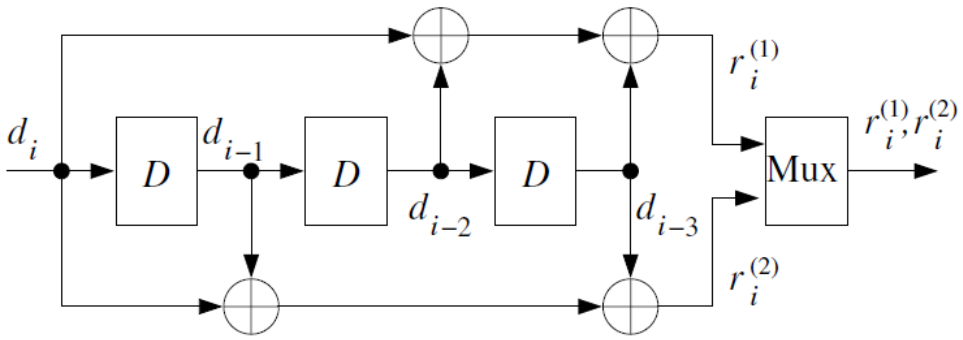

FIGURE 1.11 – Exemple de codeur de code convolutif non-systématique

Définir les polynômes générateurs d'un code systématique récursif est moins évident. Considérons l'exemple de la figure (1.12). Le premier polynôme générateur est trivial puisque le code est systématique. Pour identifier le second, il faut noter que :

$$s_i^{(0)} = d_i + s_i^{(1)} + s_i^{(3)} = d_i + s_{i-1}^{(0)} + s_{i-3}^{(0)} \tag{1.21}$$

et que :

$$r_i = s_i + s_i^{(2)} + s_i^{(3)} = s_i + s_{i-2}^{(0)} + s_{i-3}^{(0)} \tag{1.22}$$

ce qui est équivalent à :

$$d_i = s_i + s_{i-1}^{(0)} + s_{i-3}^{(0)} \tag{1.23}$$

$$r_i = s_i + s_{i-2}^{(0)} + s_{i-3}^{(0)} \tag{1.24}$$

Ce résultat peut être reformulé en introduisant la transformée en D :

$$d(D) = g_2(D)s(D) \tag{1.25}$$

$$r(D) = g_1(D)s(D) \tag{1.26}$$

où $g_1(D)$et $g_2(D)$sont les polynômes générateurs du code représenté en figure 1.11, ce qui conduit à :

$$s(D) = \frac{d(D)}{g_2(D)} \tag{1.27}$$

$$r(D) = \frac{g_1(D)}{g_2(D)}d(D) \tag{1.28}$$

Les codes générés par de tels codeurs peuvent être représentés graphiquement selon trois modèles : l'arbre, le treillis et la machine à états.

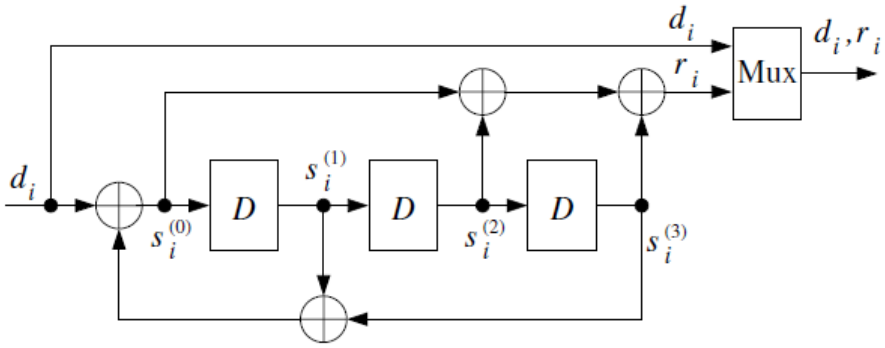

FIGURE 1.12 – Exemple de codeur de code convolutif systématique récursif

Arbre d'un code

La représentation en arbre permet de présenter toutes les séquences d'états possibles. La racine est associée à l'état de départ du codeur. On en dérive tous les états successifs possibles en fonction de l'entrée d_i du codeur. La branche reliant un état père à un état fils est étiquetée par la valeur des sorties du codeur lors de la transition associée. Ce principe est itéré pour chacun des états fils et ainsi de suite [25] . Le diagramme en arbre associé au codeur systématique de la figure1.13 est illustré en figure 1.12.Ce type de diagramme ne sera pas utilisé dans la suite,

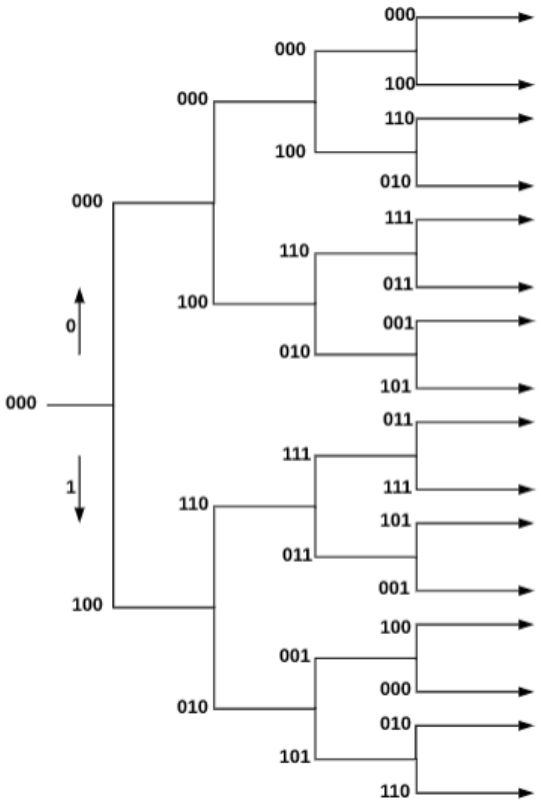

FIGURE 1.13 – Diagramme en arbre du code convolutif

Treillis d'un code

La représentation la plus courante d'un code convolutif est le diagramme en treillis. Il est d'une importance majeure aussi bien pour la définition des propriétés d'un code que pour son décodage[25] , À un instant i, l'état d'un codeur convolutif peut prendre 2^m valeurs. Chacune de ces valeurs possibles est représentée par un nœud. À chaque instant i est associée une colonne d'états-nœuds et en fonction de l'entrée d_i, le codeur transite d'un état s_i à s_{i+1} en délivrant les bits codés. Cette transition entre deux états est représentée

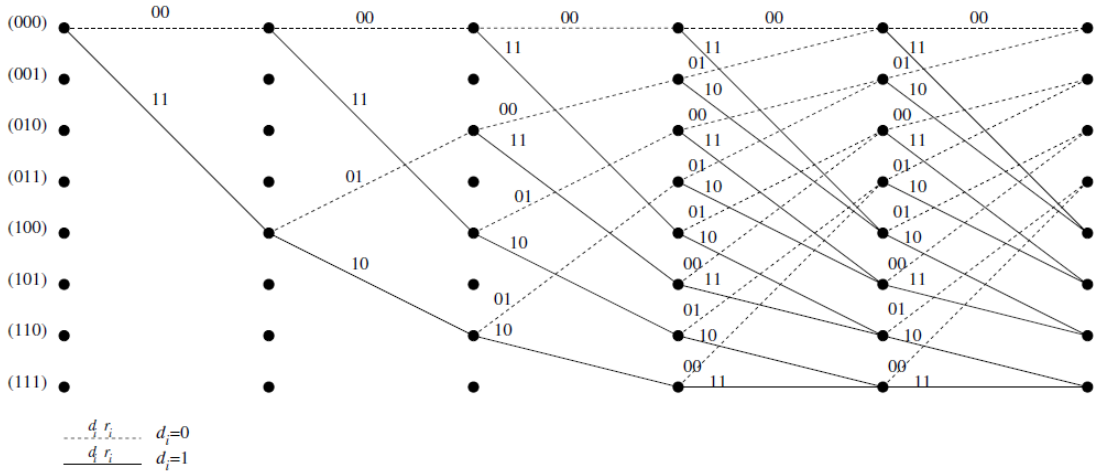

FIGURE 1.14 – Diagramme en treillis d'un code de polynômes générateurs $[1, 1+D+D^3]$.

par un arc entre les deux nœuds associés et étiqueté avec les sorties du codeur.
Dans le cas d'un code binaire, la transition sur une entrée à 0 (resp. 1) est représentée par un trait pointillé (resp. plein).
La succession des états s_i jusqu'à l'instant t est représentée par les différents chemins entre l'état de départ et les différents états possibles à l'instant t.
Illustrons ceci par l'exemple du codeur systématique de polynômes générateurs $[1, 1+D+D^3]$. En faisant l'hypothèse que l'état de départ s_0 est l'état (000) :

- si $d_1 = 0$ alors l'état suivant s_1, est aussi (000). La transition sur une entrée à 0 est représentée par un trait pointillé et étiquetée dans ce premier cas par 00, la valeur des sorties du codeur ;
- si $d_1 = 1$, alors l'état suivant s_1 est (100). La transition sur une entrée à 1 est représentée par un trait plein et étiquetée ici par 11.
- Il faut ensuite envisager les quatre transitions possibles : de $s_1 = (000)$ si$d_2 = 0$ ou $d_2 = 1$ et de $s_1 = (100)$ si $d_2 = 0$ ou $d_2) = 1$.

En itérant cette construction, on aboutit à la représentation, en figure1.14, de toutes les successions possibles d'états à partir de l'état de départ jusqu'à l'instant 5, sans la croissance illimitée du diagramme en arbre. Une section complète du treillis suffit à caractériser le code. La section de treillis du code précédent est ainsi représentée en figure 1.15.

Machine à états d'un code

Pour représenter les différentes transitions entre les états d'un codeur, il existe une dernière représentation qui est celle d'une machine à états. La convention pour définir

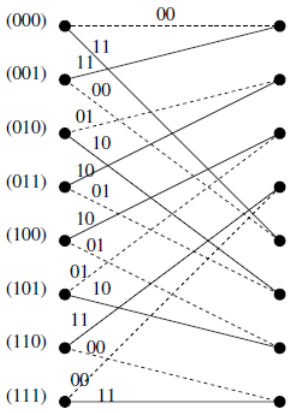

FIGURE 1.15 – Section en treillis de code de polynôme générateur $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$.

les arcs de transition est identique à celle utilisée dans la section précédente. Seuls 2^m nœuds sont représentés, indépendamment de l'instant i, ce qui revient à la représentation précédente sous forme de section de treillis [25]. Le codeur des figures 1.12 admet ainsi une représentation sous forme de machine à états illustrée en figures 1.16.

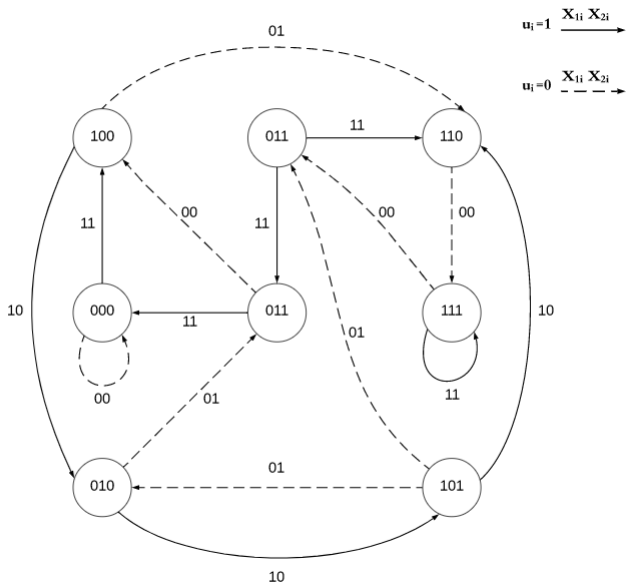

FIGURE 1.16 – Machine d'état pour un code de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$.

1.4.3 Codes convolutionnels systématiques et récursifs RSC

Le code est dit « systématique », lorsqu'un ensemble des n sorties reproduit fidèlement les k entrées. De même, un code convolutionnel est « récursif » lorsque les polynômes générateurs sont remplacés par les quotients de deux polynômes. Dans ce cas, une partie de la sortie est réinjectée dans les registres à décalage suivant les connexions fixées par les polynômes aux dénominateurs.
Il est possible de transformer un code convolutionnel non récursif, non systématique dont le rendement $R = k/(k+1)$ en un code systématique récursif [5].

Pour un codeur de rendement 1/2 par exemple qui est défini par deux polynômes générateurs $g_1(D)$ et $g_2(D)$ formant la matrice du codeur :

$$G = \begin{bmatrix} g_1(D) & g_2(D) \end{bmatrix} \tag{1.29}$$

On transformera ce code non récursif non systématique (NRNSC) en RSC en procédant à la modification suivante sur la matrice :

$$G = \begin{bmatrix} 1 & \dfrac{g_1(D)}{g_2(D)} \end{bmatrix} ou\ G = \begin{bmatrix} 1 & \dfrac{g_2(D)}{g_1(D)} \end{bmatrix} \tag{1.30}$$

1.5 Conclusion

Dans ce chapitre, nous avons introduit le schéma global d'une chaine de communication numérique, en expliquant brièvement chacune de ses parties. Ensuite, nous avons présenté quelques notions fondamentales sur le codage de canal. A la fin de ce chapitre, nous avons présenté les codes convolutifs en focalisant sur leurs différentes représentations. Ces derniers sont retenus par les concepteurs des turbo codes comme cœur de codage. Le prochain chapitre, porte sur les turbo codes convolutifs CTC .

Chapitre 2

Turbo Codage

2.1 Introduction

Dans la norme IEEE802.16e, le turbo-codage est défini comme un bloc optionnel utilisé pour le codage de canal. La norme définit deux types de turbocodes : Turbo-codage en bloc (BTC) et turbo-codage convolutionnel (CTC). Dans ce mémoire, seul le turbo-codage convolutif est implémenté. Il présente une amélioration des performances du système par rapport aux codes convolutionnels ordinaires. Le CTC a été largement utilisé dans de nombreuses normes de systèmes de communication sans fil à haut débit en raison de sa performance qui se rapproche de celle de la limite de Shannon. Il est utilisé dans 3GPP, DVB-RCS et WiMAX. Le turbo-codage a été introduit en 1993 par Berrou, Glavieux et Thitimajshima. Il est composé d'un ensemble de codeurs concaténés en séries ou en parallèles, chaque codeur code une version entrelacée des données originales.

Dans ce chapitre, nous traitons le turbo-codage utilisé dans la norme 802.16e. Ensuite, nous décrivant d'une façon détaillée le turbo-code convolutionnel CTC utilisé pour l'implémentation.

2.2 Structure des Turbo Codes

Les turbo-codes [14] constituent une famille de code correcteur d'erreurs construites, dans leur version originale, à partir d'une concaténation en parallèle de deux codes CCRS identiques, reliés par un entrelaceur.

Le codage en parallèle ajoute alors deux symboles redondants pour un symbole codé x et donc améliore la diversité du code. En effet, cette structure permet de distribuer l'énergie disponible à l'émission entre différents symboles redondants de telle sorte que le

décodeur correspondant puisse tirer le meilleur profit d'un effet de diversité.
La figure 2.1 montre le schéma fonctionnel du turbo-codeur utilisé par le WiMAX.

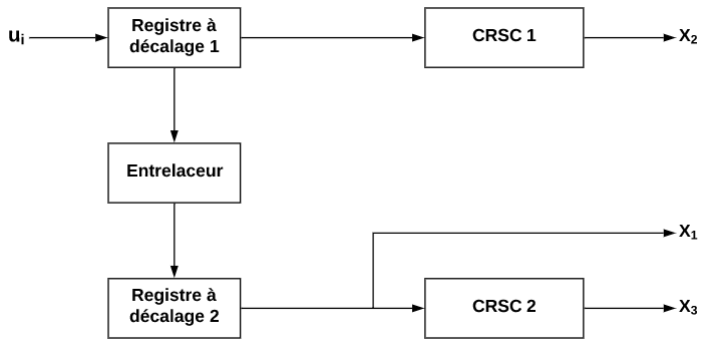

FIGURE 2.1 – Schéma fonctionnel du turbo-codeur utilisé par le WiMAX.

2.3 Entrelacement

L'entrelacement ou opération de génération de permutations consiste à changer selon l'application l'ordre des symboles du mot d'information ou du code, de manière déterministe afin de pouvoir les remettre dans l'ordre lors du décodage. La réorganisation des bits permet d'étaler les erreurs qui surviennent par paquets lors d'une transmission. L'utilisation par exemple d'un entrelacement avec un code convolutif diminue sa susceptibilité aux paquets d'erreurs. Les générateurs de permutations sont aussi utilisés dans le cas des turbo-codes afin de minimiser les séquences susceptible de générer des mots de codes à poids faible.

L'entrelacement périodique engendre des permutations selon une fonction périodique du temps.On peut le réaliser, soit par de la mémoire vive (entrelacement par blocs), technique utilisée dans tous les standards de téléphonie de deuxième génération ou par des registres à décalage (entrelacement convolutionnel). L'entrelacement pseudo-aléatoire consiste à écrire la séquence aux adresses successives d'une mémoire vive, puis d'effectuer la lecture suivant un ordre donné par un générateur pseudo-aléatoire. L'opération inverse consiste à écrire dans la mémoire suivant le même ordre pseudo-aléatoire et ensuite de lire la séquence suivant l'ordre croissant des adresses de la mémoire.

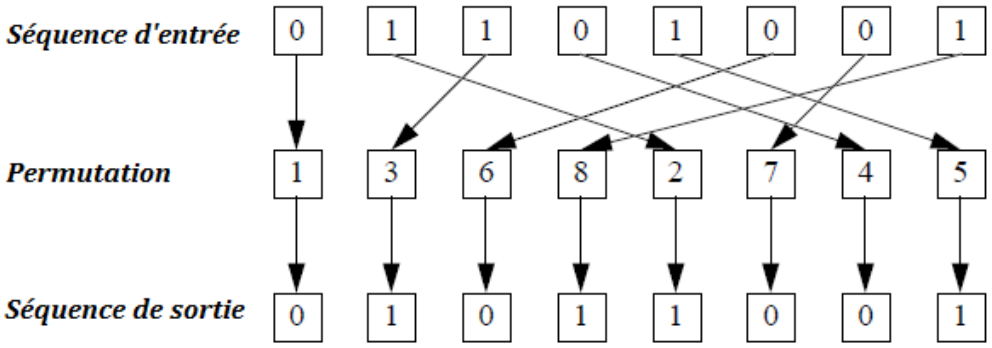

FIGURE 2.2 – Entrelacement pseudo-aléatoire avec $L = 8$.

2.3.1 Entrelaceur du turbo-code convolutionnel

L'entrelaceur CTC spécifié dans la norme IEEE802.16e se compose de deux étapes de permutation, l'une est une permutation au niveau de chaque symbole individuellement, et le second est au niveau de la séquence de tous les symboles. Les sous-sections suivantes illustrent les opérations d'entrelacement.

1. Changer de couple alternatif

   Dans cette étape, les entrées A, B sont envoyées dans leur ordre une fois, échangées pour la autre fois. Cette opération est répétée pour l'ensemble du bloc.
   La séquence d'entrée est : $U_0 = [(A_0, B_0), (A_1, B_1), (A_2, B_2), ..., (A_{N-1}, B_{N-1}$. Le résultat de cette étape est le suivant $U_1 = [(A_0, B_0), (B_1, A_1), (A_2, B_2), ...(B_{N-1}, A_{N-1})]$, où N est la taille du bloc d'entrée à entrelacer.
   L'opération ci-dessus est décrite comme suit :

   $$for\ i = 0\ to\ N-1$$
   $$if(imod_2 == 1)$$
   $$(A_i, B_i) \longrightarrow (B_i, A_i)$$

2. Calculer l'ordre entrelacé de la séquence U_1

   La séquence U_1 obtenue à l'étape précédente doit être mappée à une nouvelle séquence U_2. Le mappage est effectué par la fonction $P(j)$ définie de telle sorte que :

   $$U_2(j) = U_1(P(j)). \tag{2.1}$$

   L'opération est décrite comme suit :

   $$for\ j = 0\ to\ N-1$$
   $$switch\ j\ mod_4 :$$

$Case\ 0$:

$$P(j) = (P_0.j + 1)\ mod_N$$

$Case\ 1$:

$$P(j) = (P_0.j + 1 + N/2 + P_1)\ mod_N$$

$Case\ 2$:

$$P(j) = (P_0.j + 1 + P_2)\ mod_N$$

$Case\ 3$:

$$P(j) = (P_0.j + 1 + N/2 + P_3)\ mod_N$$

Où P_0, P_1, P_2 et P_3 sont des paramètres de codage spécifiques pour chaque taille de bloc, N, et sont fournis dans les normes. L'adresse j est entrelacée à l'adresse i. Le tableau 2.1 présente les combinaisons des paramètres par défaut à utiliser.

N	P_0	P_1	P_2	P_3
24	5	0	0	0
48	13	24	0	24
96	7	48	24	72
120	13	60	0	60
192	11	96	48	144
216	13	108	0	108
240	13	120	60	180
480	13	240	120	360
960	17	1200	600	1800

TABLE 2.1 – Paramètres de permutation du Turbo code pour le Wimax

La procédure ci-dessus calcule la séquence de bits entrelacés P(j) à partir de la séquence originale j. Dans le cas de 802.16e, le flux de bits d'entrée doit être lu par l'entrelaceur avec la séquence entrelacée P(j). Ensuite, la nouvelle séquence sera sortie linéairement.

2.4 Performances d'un turbo code

I 'approche du turbo-codage consiste à limiter le nombre de mots de code de poids faible. De plus, comme les séquences d'information de poids 2 sont les plus susceptibles de générer les mots de code de poids faible [16], elles sont les plus importantes dans la

conception des codes constituants. Plus leur poids augmente, plus leur importance dans la conception diminue. En effet, la combinaison d'un entrelacement avec un code de type CRSC permet de maximiser le poids de Hamming des mots de code en sortie. Les codes NRNSC ne sont pas utilisables à cause de leur propriété faible,en termes de distance (d_{min} faible).

La présence de l'entrelaceur permet de minimiser la probabilité que les deux codes constituants génèrent en même temps un mot de code de poids faible. En effet, grâce à l'entrelacement, les deux codes constituants ne reçoivent pas la séquence d'information dans le même ordre. Ainsi, si le premier codeur reçoit une séquence susceptible de produire un mot de code de poids faible, il est fort peu probable que cette même séquence après l'entrelacement induit aussi un mot de code de poids faible à la sortie. Néanmoins, les performances d'un turbo-code dépendent fortement du choix des codes constituants ainsi que du type et de la taille de l'entrelaceur utilisé.

2.5 Code convolutif circulaire CRSC

Une technique fut introduite au tournant des années 70 et 80 [17] pour terminer les treillis des codes convolutifs sans les effets de bord : le tail-biting. Elle consiste à rendre le treillis de décodage circulaire, c'est-à-dire à faire en sorte que l'état de départ et l'état final du codeur soient identiques. Cet état est alors appelé état de circulation. Cette technique de fermeture appliquée aux codes récursifs génère un code dit convolutif systématique récursif circulaire (CSRC)[25] . Le treillis d'un tel code est illustré par la figure 2.3 :

Pour expliquer la construction d'un tel code, il est nécessaire de revenir aux équations fondamentales qui régissent le fonctionnement d'un codeur. Il est possible d'établir une relation entre l'état$S_(i+1)$du codeur à un instant $i+1$, son état S_i et la donnée entrante d_i à un instant i par :

$$S_(i+1) = \mathbf{A}\ S_i + \mathbf{B}\ d_i \tag{2.2}$$

Où **A** est la matrice d'état et **B** celle d'entrée. Dans le cas du code systématique récursif précédemment cité de polynômes générateurs $[1, (1+D^2+D^3)/(1+D+D^3)]$, ces matrices sont définies comme :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \ et\ \ \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.3}$$

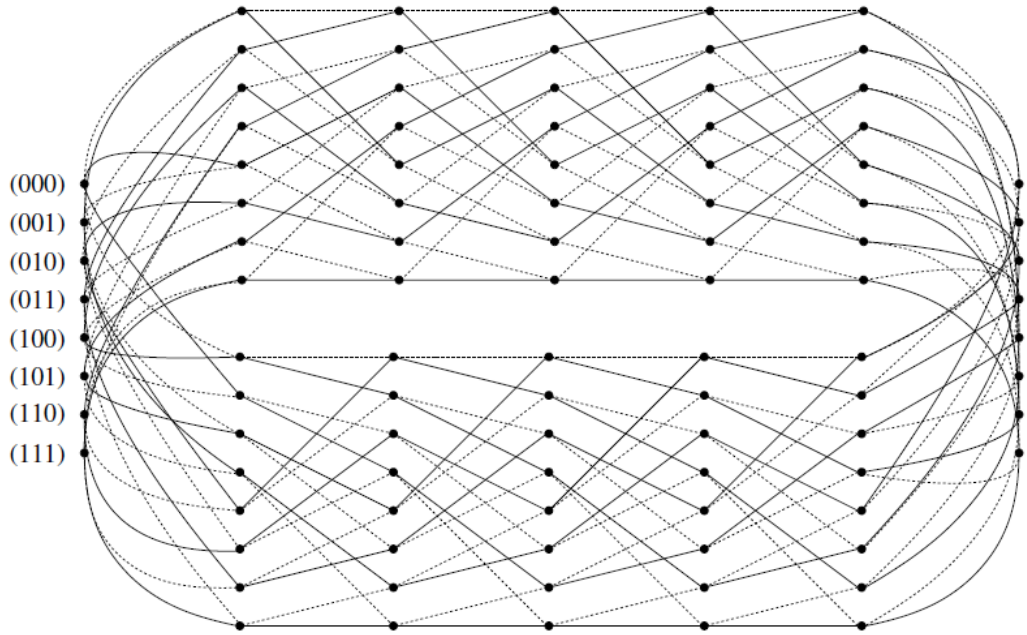

FIGURE 2.3 – Treillis d'un code CSRC à 8 états.

Si le codeur est initialisé à l'état $0(S_0 = 0)$, l'état final S_k^0 obtenu à la fin d'une trame de longueur k est :

$$S_k^0 = \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\ d_{k-j} \tag{2.4}$$

Lorsqu'il est initialisé dans un état quelconque S_c, l'état final S'_k s'exprime delà manière suivante :

$$S'_k = \mathbf{A}^k S_c + \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\ d_{k-j} \tag{2.5}$$

Pour que S'_k cet état soit égal à S_c l'état de départ et que celui-ci devienne donc l'état de circulation, il faut et il suffit que :

$$(\mathbf{I} - \mathbf{A}^k)S_c = \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\ d_{k-j} \tag{2.6}$$

Où $\mathbf{I}$ est la matrice identité de dimension $m \times m$ (m la taille de la mémoire). Ainsi, en introduisant l'état S_k^0 du codeur après initialisation à 0

$$S_c = (\mathbf{I} - \mathbf{A}^k)^{-1}\ S_c^0 \tag{2.7}$$

Ceci n'est possible qu'à condition que la matrice $(\mathbf{I} - \mathbf{A}^k)$ soit inversible : c'est la condition

d'existence de l'état de circulation.

En conclusion, s'il existe, l'état de circulation s'obtient en deux étapes. La première consiste à coder la trame de k bits en entrée après avoir initialisé le codeur à l'état zéro et à conserver l'état de terminaison. La seconde se résume à déduire l'état de circulation du précédent état de terminaison à l'aide d'une table (obtenue par l'inversion de $(\mathbf{I} - \mathbf{A}^k)$).

$S_k^0 \backslash\ k\ mod\ 7$	1	2	3	4	5	6
0	0	0	0	0	0	0
1	6	3	5	4	2	7
2	4	7	3	1	5	6
3	2	4	6	5	7	1
4	7	5	2	6	1	3
5	1	6	7	2	3	4
6	3	2	1	7	4	5
7	5	1	4	3	6	2

TABLE 2.2 – Table du code CSRC de polynômes générateurs [1, (1 + D2 + D3)/(1 + D+D3)] fournissant l'état de circulation.

2.6 Caractéristiques du turbo codeur convolutionnel (CTC) utilisé

Soit le Turbo codeur dont la structure est donnée dans la Figure 2.4. Le bit systématique est associé au codeur CRSC 2, le train de bits d'information d'entrée u est chargé dans le registre à décalage 1 pour former une trame de données. Cette trame de données est transmise à l'entrelaceur, sa sortie est introduite dans le registre à décalage 2. Les deux codeurs de composants codent ensuite leurs entrées respectives. Les trains de bits codés en sortie x_l, x_2 et x_3 sont multiplexés ensemble pour former un seul train de bits de transmission.

Dans notre cas, nous sommes appeler à opter pour un codeur convolutionnel systématique et récursif circulaire (CSRC) de polynômes générateurs $[1, (1+D^2+D^3)/(1+D+D^3)]$ de rendement $R = 1/2$ une mémoire de taille $m = 3$ (donc une longueur de contrainte $L = 4$ et un nombre d'états $n = 2^3 = 8$).
Le schéma du codeur CRSC utilisé est celui de la Figure 2.5.

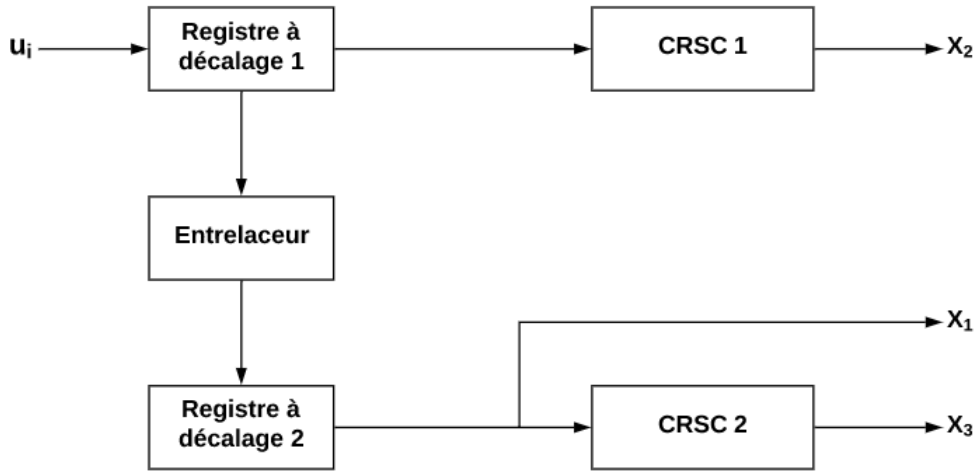

FIGURE 2.4 – Structure du turbo codeur convolutionnel (CTC) utilisé.

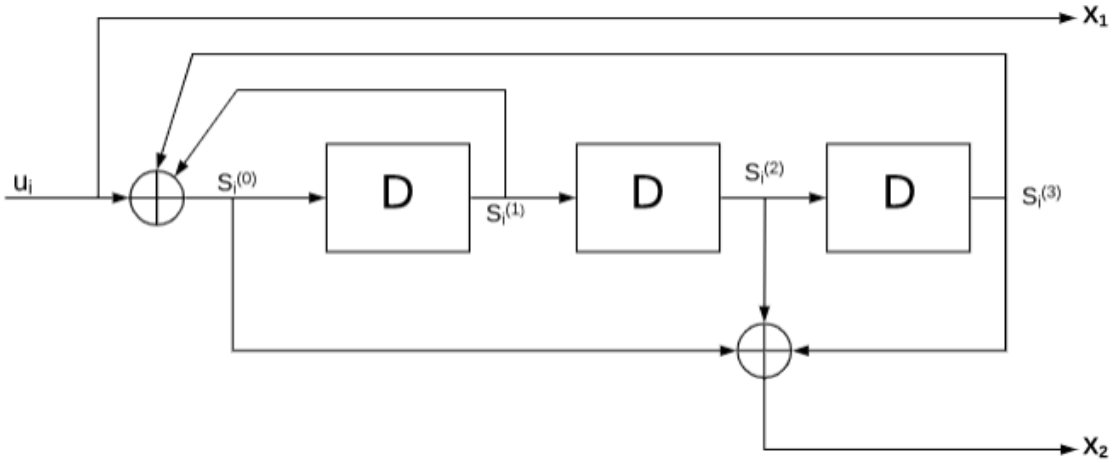

FIGURE 2.5 – Schéma du codeur CRSC utilisé.

Le diagramme d'état du codeur convolutionnel que nous avons choisi est exposé dans le paragraphe précédent et décrit dans la Figure 2.6.

- Chaque case représente un état du codeur, c'est-à-dire le contenu binaire des mémoires du registre à décalage.
- Chaque branche représente une transition entre deux états et est étiquetée de l'entrée et des deux sorties correspondantes.

Les bits codés doivent être mappés du domaine {0,1} au domaine {-1, + 1} pour une transmission BPSK. la Figure 2.7 montre ce diagramme d'état modifié.

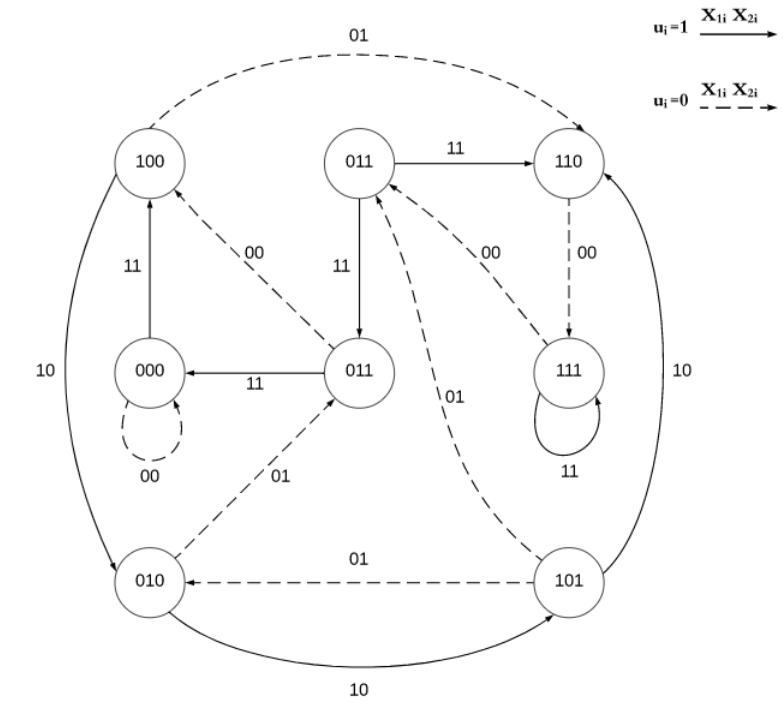

FIGURE 2.6 – Diagramme d'état du codeur convolutionnel CRSC utilisé.

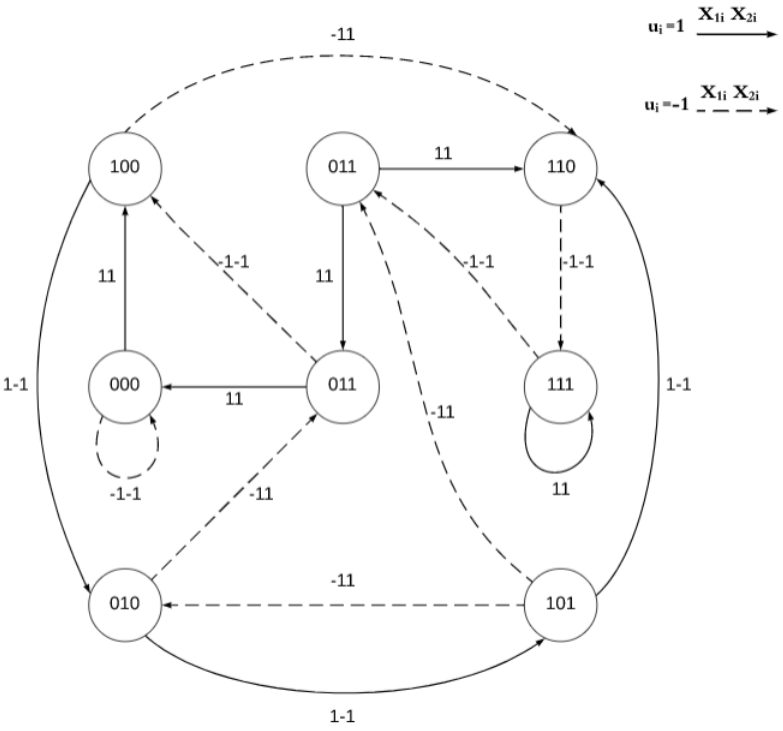

FIGURE 2.7 – Diagramme d'état de transmission du codeur convolutionnel CRSC utilisé.

Il existe néanmoins, une autre représentation possible des états du codeur convolutionnel qui est le diagramme en arbre Figure 2.8 :

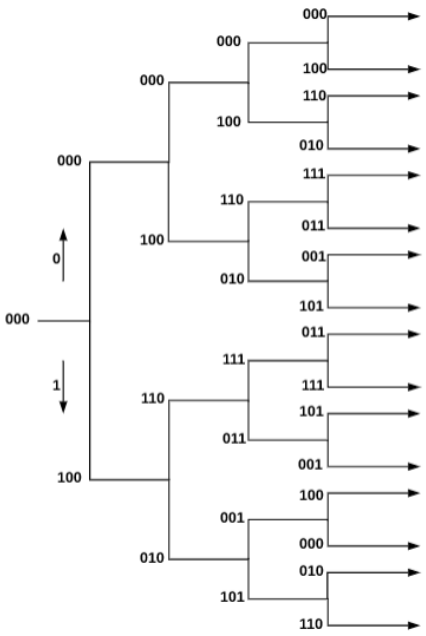

FIGURE 2.8 – Représentation en arbre du codeur convolutionnel CRSC utilisé.

Cependant, le codeur peut être représenté par un tableau qui contient le bit d'entrée, l'état courant du codeur, l'état suivant et le mot code de sortie. Table 2.3 .

Bit d'entrée	État courant	État suivant	X_1	X_2	Bit d'entrée	État courant	État suivant	X_1	X_2
0	000	000	0	0	1	000	100	1	1
0	001	100	0	0	1	001	000	1	1
0	010	001	0	1	1	010	101	1	0
0	011	101	0	1	1	011	001	1	1
0	100	110	0	1	1	100	010	1	0
0	101	010	0	1	1	101	110	1	0
0	110	111	0	0	1	110	011	1	1
0	111	011	0	0	1	111	111	1	1

TABLE 2.3 – Autre présentation du codeur convolutionnel utilisé.

Il est possible d'associer à cette table un treillis formé de nœuds représentants les différents états du codeur. Ces derniers sont reliés par des branches qui désignent les différentes transitions possibles d'un nœud à un autre. Ce treillis est illustré dans la figure 2.9.

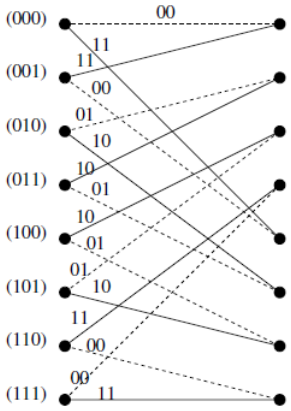

FIGURE 2.9 – Section en treillis du codeur convolutionnel CRSC utilisé.

2.7 Conclusion

Ce chapitre fait la lumière sur les Turbo codes. Il se termine par une illustration des codeurs récursifs et systématiques circulaire CRSC ainsi que leurs caractéristiques. Le chapitre suivant exposera le principe du turbo décodage itératif et l'algorithme retenu pour l'implémentation.

Chapitre 3

Décodage itératif des turbo codes

3.1 Introduction

Le processus de décodage Turbo est un processus itératif composé de deux décodeurs SISO (Soft In Soft Out), Chaque décodeur reçoit les valeurs douces de l'émetteur par le canal de communication. le décodage est effectué plusieurs fois et seules les informations extrinsèques (sous-résultats) sont transmises entre les itérations. Chaque décodeur calcule les Log Likely-hood Ratios (LLRs) qui sont la somme de deux termes : les LLRs intrinsèques provenant du canal de communication et les LLRs extrinsèques ajoutés par le décodeur lui-même. Après chaque demi-itération, les décodeurs échangent leurs informations extrinsèques. L'algorithme de décodage nécessite plusieurs itérations pour estimer les données transmises. Après le nombre prédéterminé d'itérations, généralement de 4 à 8 selon les demandes de TEB (taux d'erreur binaire) et de TOF (taux d'erreur de trame), une décision finale est prise en utilisant les informations extrinsèques des deux décodeurs SOVA et la trame systématique.

Dans ce chapitre, nous présentons le décodeur Turbo à base de l'algorithme SOVA (algorithme à sortie douce) pour le WIMAX.

3.2 Techniques de décodage

Le canal utilisé est le canal (AWGN) qui se caractérise par l'addition d'un bruit blanc gaussien au signal qui le traverse, et c'est en général le canal le plus utilisé pour les simulations parce qu'il est le modèle le plus rapproché aux canaux réels [18]. La modulation dans ce cas est la modulation BPSK. Le modulateur BPSK convertit l'information du domaine binaire {0,1}, au domaine -1,+1 où, le 'zéro' logique est converti en un niveau

-1 électrique et convertit le 'un' en un niveau représenté par +1 électrique. A l'entrée du canal, le signal codant l'information est un signal 'propre', il ne présente aucune anomalie. A la sortie du canal, le signal recueilli est altéré, il présente des 'ambiguïtés', ceci est dû au fait que le canal est bruité, il introduit une composante non uniforme sur le signal qui devient distordu. Cette distorsion est plus ou moins importante, si le canal est de très mauvaise qualité, un bit peut changer de valeur à la réception [19]

Le décodage à décision dure ou à décision douce (souple) désigne le type de quantification utilisée à la réception du signal codant les bits d'information,

3.2.1 Décodage par Décision Dure (Hard Decision Decoding)

Le décodage à décision dure utilise un seul bit pour quantifier l'information reçue du canal, pour ce type de décodage, un bloc est rajouté juste après le démodulateur comme étant le bloc de décision : si la valeur du signal d'information à l'entrée du démodulateur est supérieur à zéro, le bit d'information à la sortie du bloc de décision sera un ' un ' logique, alors que s'il est négatif, la décision serait un 'zéro' logique.

3.2.2 Décodage par Décision Douce (Soft Decision Decoding)

Pour le décodage à décision souple on utilise une quantification multi bits. Dans le cas idéal, on utilise le décodage à décisions douces avec une infinité de bits de quantification, ainsi, les valeurs reçues du canal sont directement utilisées dans le décodeur Dans le décodage à décision douce, on n'utilise pas de bloc qui fait la décision sur la valeur du bit avant le décodeur, on injecte la valeur recueillit du démodulateur directement dans le décodeur. De cette façon on intègre dans le décodage les perturbations causées par le canal de transmission, la décision sur la valeur du bit (1 ou 0) se fera après le décodage. La figure 3.1 récapitule les deux types de décision.

3.3 Algorithmes de décodage

Le décodage Viterbi représente une des plus populaires technique de correction d'erreurs directe dans les systèmes de communications, Cette technique de décodage a été proposée par A .Viterbi, elle est basé sur la théorie de maximum de vraisemblance. L'algorithme SOVA (Soft Output Viterbi Algorithme) est une Modification de l,algorithme Viterbi, dans laquelle on a introduire le concept « valeur douce », cette modification

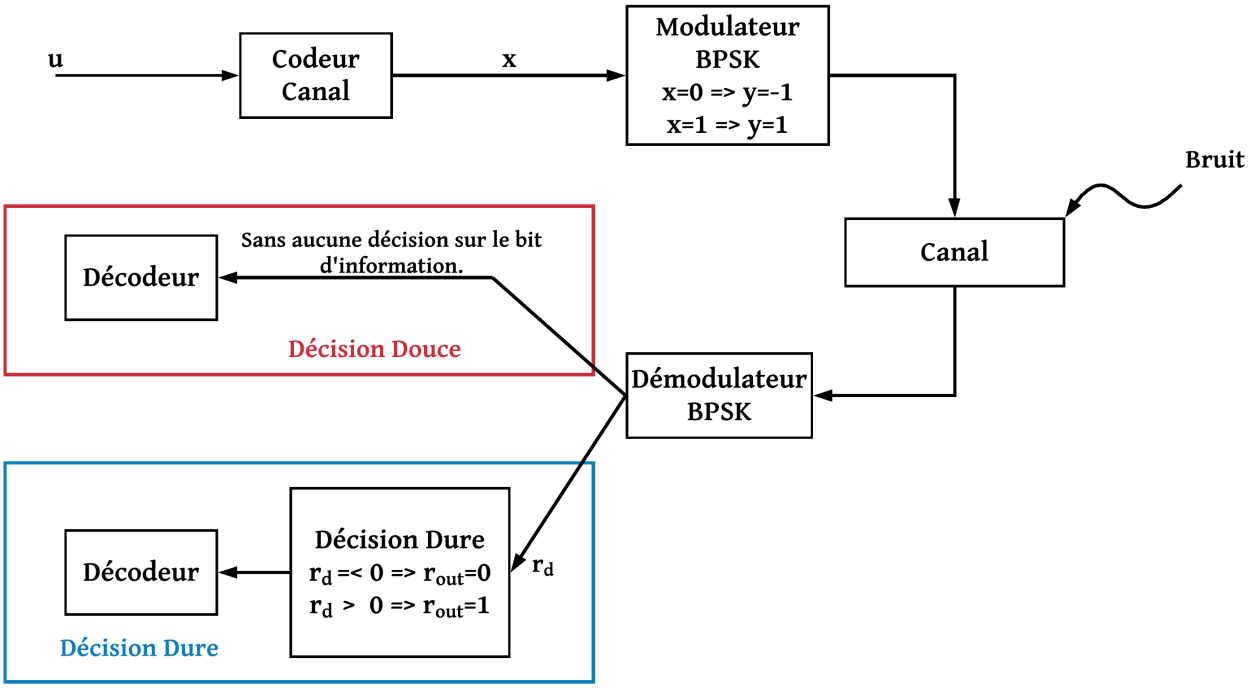

FIGURE 3.1 – Chaîne de transmission avec différents décisions

permet, entre autres, de quantifier les valeurs des bits reçus et de donner ainsi des informations plus précises sur son estimation. Ce qui améliore les performances du décodage et de la reconstitution de l'information initialement émise.

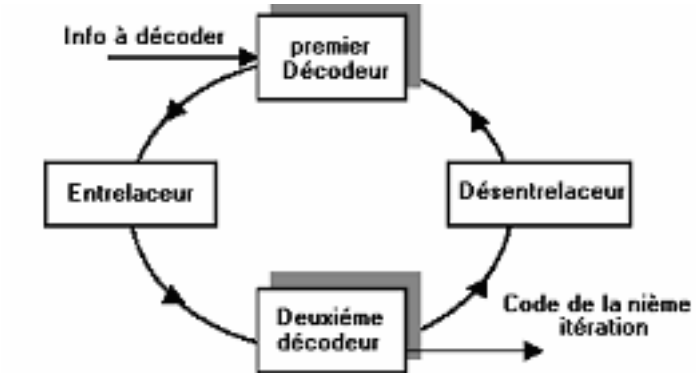

FIGURE 3.2 – Fonctionnement du turbo décodeur

Le décodeur SOVA (Soft Output Viterbi Algorithm) utilise un algorithme similaire à celui du décodeur Viterbi, les différences résident dans le fait que le décodeur SOVA a pour entrée et sortie des informations réelles, la sortie douce correspond en gros à la probabilité que chaque bit transmis est erroné. En effet cette sortie permet au décodeur

de prendre en compte cette probabilité pour les utiliser dans l'estimation de la séquence à corriger.

3.3.1 Algorithme de Viterbi

Pour décoder les codes convolutifs, Andrew Viterbi a proposé un schéma de décodage basé sur la théorie de maximum de vraisemblances [20] [21], Pour déterminer la séquence la plus vraisemblable, l'algorithme de Viterbi procède de la manière suivante : à chaque étape (transition dans le treillis), on calcule la distance de Hamming entre le bloc de k bits reçu et les différentes branches du treillis et on ajoute ensuite ces distances dites locales aux distances globales précédemment calculées pour déterminer les distances globales courantes. On compare ensuite les métriques (distances) globales des différentes séquences convergeant vers un même état pour en garder une seule et éliminer toutes les autres. La séquence retenue est celle correspondant à la distance de Hamming la plus petite. On l'appelle généralement la séquence survivante.

3.3.2 Faiblesse de l'algorithme de Viterbi vis à vis les TURBO-CODES

Les turbo-codes, par conception, utilisent une concaténation de deux codes convolutifs avec les différentes variantes de concaténation (série et parallèle) [19]. Le principe même des turbo codes repose sur le décodage itératif (voir figure 3.2), ainsi en utilisant une concaténation série au décodage, on injecte l'information recueillit à la sortie du premier décodeur à l'entrée du deuxième et ainsi on obéit au principe turbo d'où vient le nom du turbo code : Les turbo codes tirent leurs noms à partir des machines turbo qui utilisent leur sortie pour améliorer leur rendement en la réinjectant à l'entrée [22] Les faiblesses de l'utilisation de L'algorithme de Viterbi avec les turbo codes résident dans le fait qu'il produit la séquence d'information la plus vraisemblable pour les codes convolutifs. Cet algorithme génère une séquence d'estimation optimale pour un étage de code convolutif. Pour les codes convolutifs concaténé (multi étage), il existe deux inconvénients majeurs pour l'utilisation de décodeurs Viterbi convolutifs conventionnels qui sont : [23] [22]

- Le décodeur Viterbi interne produit des trames de bits erronés qui dégradent les performances du décodeur Viterbi externe.

- Le décodeur Viterbi interne produit des sorties à décision dures ce qui empêche le décodeur Viterbi externe de bénéficier des avantages de la décision douce.

Ces deux inconvénients peuvent être réduits et la performance des décodeurs concaténés peut être améliorée de manière significative si les décodeurs Viterbi sont capables de produire des valeurs de fiabilité (sortie douce). Les valeurs de fiabilité sont transmises aux décodeurs Viterbi comme information a priori pour améliorer la performance de décodage. Ce décodeur Viterbi modifié est appelé décodeur SOVA (Soft Output Viterbi Algorithme=algorithme Viterbi à sortie douce). [23] [22]

3.3.3 Algorithme SOVA pour les Turbo-Code

L'Algorithme SOVA est une modification de l'algorithme de Viterbi pour lui permettre d'être utilisé comme élément décodeur dans le turbo décodeur [22] [19]. Cette modification réside en deux points :

- Le calcul des métriques de branches est modifié pour permettre de prendre en compte les informations a priori.

- L'algorithme est modifié de façon qu'il produise une information de sortie douce sous la forme d'information a posteriori LLR (Log Likelyhood Ratio : taux de vraisemblance logarithmique) : $L(u_k|y)$ pour chaque bit décodé.

3.3.4 Algèbre de vraisemblance logarithmique

L'algèbre de log-vraisemblance utilisée pour le décodage SOVA des turbo-codes est basée sur une variable aléatoire binaire u dans GF(2) avec des éléments +1, -1, où +1 est l'élément logique 0 (élément "nul") et -1 est l'élément logique 1 sous l'addition modulo 2. La table de vérité présentée dans le tableau 3.1 montre le résultat de la somme de deux variables binaires aléatoires en utilisant le modèle décrit ci-dessus.

$u_1 \oplus u_2$	$u_2 = +1$	$u_2 = -1$
$u_1 = +1$	+1	-1
$u_1 = -1$	-1	+1

TABLE 3.1 – Résultat de l'addition de deux variables aléatoires binaires u_1 et u_2.

Le rapport logarithmique de vraisemblance L(u) pour une variable aléatoire binaire u est défini comme suit :

$$L(u) = \ln \frac{P(u = +1)}{P(u = -1)} \tag{3.1}$$

L(u) est souvent désigné comme la valeur "douce" ou la L-valeur de la variable aléatoire binaire u. Le signe de L(u) est la décision dure de u et la magnitude de L(u) est la fiabilité de cette décision.

3.3.5 Canal a sorties douces

A partir du modèle de système de la figure 3.3, le bit d'information u est mappé aux bits codés x. Les bits codés x sont transmis sur le canal et reçus comme y.

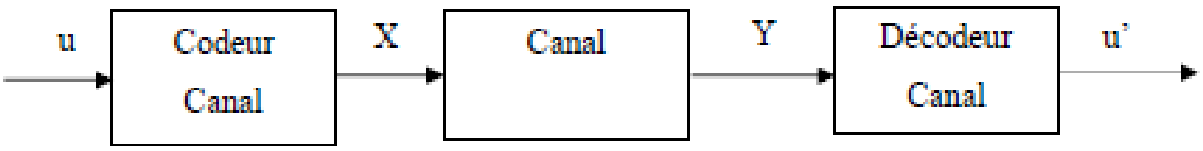

FIGURE 3.3 – Modèle de la chaine de transmission

A partir de ce modèle de système, le rapport log-vraisemblance de x conditionné sur y est calculé comme suit :

$$L(x/y) = \ln \frac{P(x=+1/y)}{P(x=-1/y)} \tag{3.2}$$

En utilisant le théorème de Bayes, ce rapport logarithmique de vraisemblance est équivalent à :

$$L(x/y) = \ln \left(\frac{P(y/x=+1)P(x=+1)}{P(y/x=-1)P(x=-1)} \right) \tag{3.3}$$

$$= \ln \frac{P(y/x=+1)}{P(y/x=-1))} + \ln \frac{P(x=+1)}{P(x=-1} \tag{3.4}$$

Le canal est supposé être à atténuation plate avec bruit gaussien. Le modèle du canal est représenté par la fonction de pondération gaussienne [24] :

$$f(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(z-\mu)^2}{2\sigma^2} \tag{3.5}$$

où μ est la moyenne et σ^2 est la variance du bruit, on peut montrer que :

$$\ln \frac{P(y/x=+1)}{P(y/x=-1)} = \ln \frac{\exp \frac{-E_b}{N_0}(y-a)^2}{\exp \frac{-E_b}{N_0}(y+a)^2} \tag{3.6}$$

$$= \ln \frac{\exp \frac{E_b}{N_0} 2ay}{\exp \frac{-E_b}{N_0} 2ay} \tag{3.7}$$

$$= 4\frac{E_b}{N_0}ay \tag{3.8}$$

Où E_b/N_0 est le taux de signal sur bruit (directement lié à la variance du bruit) et 'a' est l'atténuation du canal. Pour des canaux gaussiens non atténuants a = 1.

Le taux de vraisemblance logarithmique de x conditionné par y L(x|y) est équivalent à :

$$L(x/y) = L_c y + L(x) \tag{3.9}$$

Où L_c est définie comme étant la fiabilité du canal :

$$L_c = 4\frac{E_b}{N_0}a \tag{3.10}$$

Ainsi $L(x/y)$ est la valeur reçue y pondéré par L_c sommé avec le taux de vraisemblance logarithmique de $x, (L(x))$.

3.3.6 Composant décodeur SOVA pour un turbo-code

L'élément décodeur SOVA estime la séquence d'information en utilisant une des deux trames codées produites par le codeur turbo code et la trame systématique [19]. La figure 3.4 montre les entrées et les sorties de l'élément décodeur SOVA.

FIGURE 3.4 – Élément décodeur SOVA

L'élément décodeur SOVA traite les entrées L(U), $L_c y$ et $L_c y1$ où

- $L(u)$ est l'information a priori de la séquence d'information u.
- $L_c y_1$ est la séquence systématique pondéré aussi par L_c.
- $L_c y$ est la séquence pondérée reçue codé correspondant à cet élément décodeur.

Les séquences 'y' et 'y_1' sont reçues du canal, cependant la séquence L(u) est produite et obtenue à partir du précédent élément décodeur SOVA. S'il n'y a aucun élément décodeur SOVA précédent, alors il n'y a pas de valeurs a priori. Ainsi la séquence $L(u)$ est initialisée à zéro.

L'élément décodeur SOVA produit u' et $L(u')$ comme sorties où :

- u' est la séquence d'information estimée.
- $L(u')$ est l'information a posteriori de la séquence d'information estimée u.

L'élément décodeur SOVA fonctionne d'une manière similaire au décodeur Viterbi excepté que la séquence ML (Maximum Likelyhood) est calculée en utilisant une métrique différente. Cette métrique qui prend en compte les valeurs a priori est décrite ci-dessous.

L'algorithme de Viterbi fondamentale recherche la séquence d'état S(m) ou la séquence d'information $u(m)$ qui maximise la probabilité a posteriori $p(S(m)/y)$ [19] [22] [24]. Pour un treillis binaire (nombre d'entrées k=1), m peut être soit 1 ou 2 dénotant le chemin survivant et le chemin concurrent respectivement. En utilisant le théorème de Bayes, la probabilité a posteriori peut être exprimée comme suit [19] :

$$P(S^{(m)}/y) = P(y)/S^{(m)}\frac{P(S^{(m)}}{P(y)} \tag{3.11}$$

Partant de la relation 3.11, on détermine la métrique SOVA comme étant l'égalité suivante :

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c y_{t,1} + \sum_{j=2}^{N} x_{t,j}^{(m)} L_c y_{t,j} + u_t^{(m)} L(u_t) \tag{3.12}$$

m peut être 1 ou 2 pour désigner le chemin survivant et le chemin concurrent.respectivement. On remarque des relations 3.12 que la métrique SOVA comporte des valeurs de la métrique précédente, de la fiabilité du canal et de la fiabilité de la source (valeur a priori).

La figure 3.5 montre la fiabilité de la source utilisée dans le calcul de métrique SOVA. Cette figure montre le diagramme en treillis avec deux états Sa et Sb et une transition de l'instant t-1 à l'instant t. Les lignes continues indiquent que la transition va produire un bit d'information $u_t = +1$ alors que les lignes interrompues indiquent que la transition va produire un bit d'information $u_t = -1$. La fiabilité de source $L(u_t)$, qui peut prendre des valeurs positives ou négatives, provient de l'élément décodeur SOVA précédent. La valeur ajoutée est incorporée dans la métrique SOVA pour induire une décision plus sure sur le bit d'information estimé. Par exemple, si $L(u_t))$ est un grand nombre positif, alors il serait relativement plus difficile de changer la décision du bit estimé de +1 à –1 entre les étages de décodage [19]. Cependant si $L(u_t)$ est un petit nombre positif, alors il serait relativement plus facile de changer la décision du bit estimé de +1 à –1 entre les étages de décodage. Ainsi $L(u_t)$ est similaire à un buffer qui essai d'empêcher le décodeur de décider que le bit d'information est opposé à celui du décodeur précédent[19].
La figure 3.6 montre l'importance de la balance entre la fiabilité de canal et celle de la source pour la métrique SOVA.

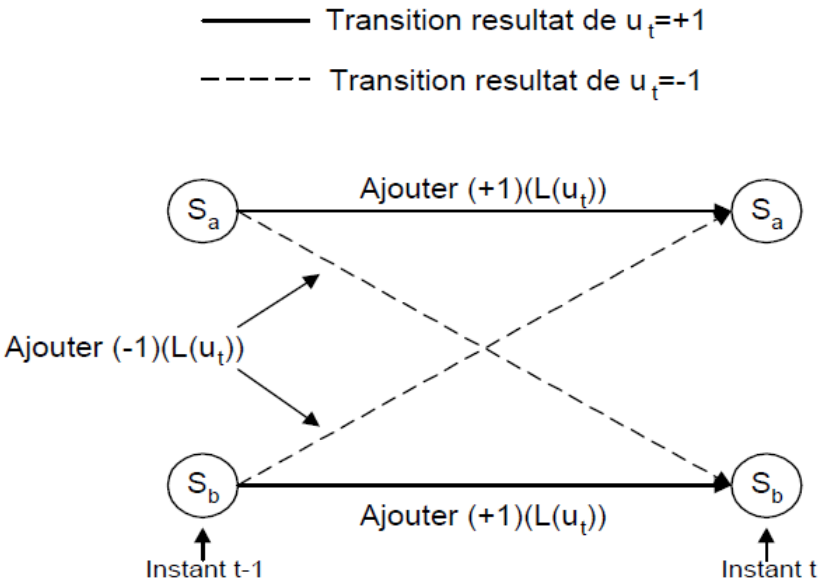

FIGURE 3.5 – Fiabilité de la source pour le calcul de la métrique SOVA

Ceci ne veut pas dire que les valeurs de fiabilité du canal et de la source doivent avoir la même amplitude mais que leurs valeurs relatives reflètent les conditions du canal et de la source. Par exemple, si le canal est de très bonne qualité, $L_c y$ va être supérieur à $|L(u)|$ et le décodage se basera essentiellement sur les valeurs reçues du canal. Cependant, si le canal est de très mauvaise qualité, le décodage se basera essentiellement sur l'information a priori $L(u)$. Si cette balance n'est pas réussie, des effets catastrophiques peuvent résulter et dégrader les performances du décodeur canal.

A chaque instant t, la valeur de fiabilité $\Delta_{j,t}$ (amplitude du rapport logarithmique/vraisemblance) attribuée à un nœud j (état)dans le treillis est déterminée à partir de :

$$\Delta_{j,t} = \frac{1}{2}|M_t^{(1)} - M_t^{(2)}| \tag{3.13}$$

A la fin de la séquence reçue, il en résulte deux matrices 3.14 et 3.15 qui correspondent aux valeurs de fiabilité et aux bits de décision à chaque instant et pour chaque nœud. Le vecteur 3.16 représente les métriques survivantes à cet instant (fin de la séquence). La position de maximum de ces métriques est nécessaire pour déduire l'état circulaire c (ceci a été expliqué précédemment dans le chapitre 2) qui correspond au point de départ pour parcourir le treillis dans le sens inverse (Trace Back) afin d'extraire la séquence u' estimée et mettre à jour la valeur de fiabilité :

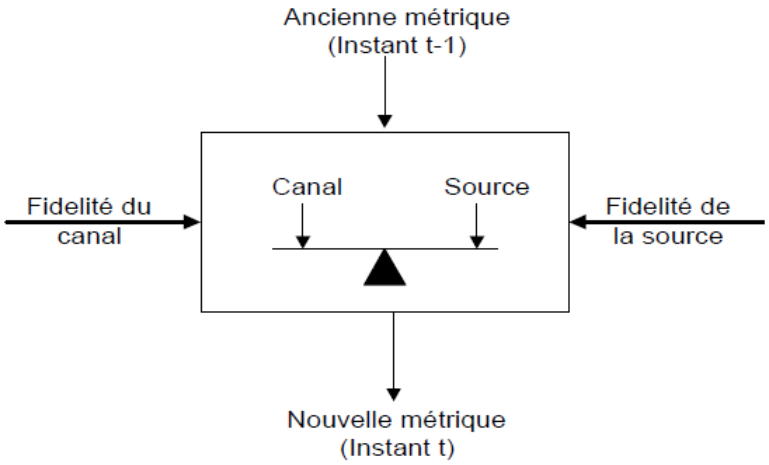

FIGURE 3.6 – Propriétés de poids de la métrique SOVA

$$\begin{bmatrix} \Delta_{0,t-N} & \Delta_{0,t-N+1} & \cdots & \Delta_{0,t} \\ \Delta_{1,t-N} & \Delta_{1,t-N+1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{7,t-N} & \cdots & \cdots & \Delta_{7,t} \end{bmatrix} \tag{3.14}$$

$$\begin{bmatrix} d_{0,t-N} & d_{0,t-N+1} & \cdots & d_{0,t} \\ d_{1,t-N} & s_{1,t-N+1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ d_{7,t-N} & \cdots & \cdots & d_{7,t} \end{bmatrix} \tag{3.15}$$

$$\begin{bmatrix} M_{0,t} \\ M_{1,t} \\ \vdots \\ M_{7,t} \end{bmatrix} \tag{3.16}$$

- Les bits de la séquence estimée u' sont les bits de chaque nœud du chemin survivant à chaque instant.
- Les valeurs de fiabilité le long du chemin survivant d'un nœud (état) particulier au temps t sont désignées par Δ_t^i (i=0,1,...,7). Au temps t, si le bit du chemin survivant est le même que le bit associé au chemin concurrent, alors la valeur de fiabilité de cette position j reste inchangée$\Delta_t^i = \Delta_{j,t}$, autrement dit c'est la valeur de fiabilité du chemin survivant est stockée. Cependant, si les bits de décision des

chemins survivant et concurrent diffèrent, cela signifie l'existence d'un bit erroné. La valeur de fiabilité à cette position d'erreur binaire doit ensuite être mise à jour en lui affectant la valeur de fiabilité de chemin survivant de l'instant précédent $\Delta_t^i = \Delta_t^{i-1}$.

Les mises à jour de fiabilité sont effectuées pour améliorer les valeurs "soft" ou L. Il est montré dans [20] que la valeur "soft" ou L d'une décision de bit est :

$$L(u') = u' \bullet \Delta \tag{3.17}$$

où $\bullet$ l'opérateur définit l'opération de multiplication élément par élément et Δ est la séquence de fiabilité mise à jour finale.

L'algorithme de Viterbi à sortie souple (ainsi que sa procédure de mise à jour de la fiabilité) peut être implémenté de la manière suivante :

1. • Initialiser le temps t = 0.
   • Initialiser $M_0^{(m)} = 0$ pour tous les états du diagramme de treillis.
2. • Passer vers l'itération suivante $t = t + 1$.
   • Calculer les métriques de branches pour chaque état du diagramme en treillis :

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c y_{t,1} \sum_{j=2}^{N} x_{t,j}^{(m)} L_c yt, j + u_t^{(m)} L_e(u_t) \tag{3.18}$$

   où :
   - m désigne la branche/transition de treillis binaire convergente vers un état ($m = 1, 2$).
   - $M_t^{(m)}$ est la métrique accumulée à l'instant t de la branche m.
   - $u_t^{(m)}$ est le bit systématique à l'instant t de la branche m.
   - $x_{t,j}^{(m)}$ est le j-ème bit de N bits à l'instant t de la branche m ($2 \leq j \leq N$).
   - $y_{t,j}^{(m)}$ est la valeur reçue du canal correspondant à $x_{t,j}^{(m)}$.
   - $L_c = \dfrac{E_b}{N_0}$ est la valeur de fiabilité du canal.
   - $L_e(u_t)$ est la valeur de fiabilité a-priori à l'instant t. Cette valeur provient du décodeur précédent. S'il n'y a pas de décodeur précédent, cette valeur est mise à zéro.
3. Comparer les deux métriques des transitions $M_t^{(1)}$ et $M_t^{(2)}$ qui convergent vers le même nœud j(état), le maximum est défini comme étant la métrique survivante $M_{j,t}$, selon cette comparaison le bit de décision $\hat{\mathrm{d}}_t$ sera obtenu à l'aide de la table de treillis (table111).

4. Calculer pour chaque état j le facteur de fiabilité $\Delta_{j,t}$ par la relation suivante :
$$\Delta_{j,t} = \frac{1}{2}|M_t^{(1)} - M_t^{(2)}| \tag{3.19}$$
5. Stocker les métriques survivantes ($M_{1,t}^{(1)}$,$M_{2,t}^{(1)}$,... ,$M_{8,t}^{(1)}$), leurs bits de décision ($d_{1,t}$,$d_{2,t}$,... ,$d_{8,t}$)et les facteurs de fiabilité pour chaque état et à chaque instant dans deux mémoires DTM (Deltas Memory) et DM (Decisions Memory) respectivement.
6. Passer à l'étape (2) et refaire les mêmes opérations précédentes jusqu'à la fin de la séquence reçue.

Une fois arrivée à la fin de la séquence reçue, le treillis sera parcouru dans le sens inverse (Trace Back) en réalisant les opérations suivantes :
RMQ :les bits de décision et les facteurs de fiabilité utilisés dans les opérations suivantes sont lu depuis des mémoires LIFO (DTM et DMM).

7 - Trouver la position de Max de la dernière séquence seulement afin de déduire l'état circulaire.
- Stocker le bit de décision comme étant le bit estimé u' et le facteur de fiabilité qui correspondent à l'état circulaire c ($\Delta_t^0 = \Delta_{c,t}$) .
- Trouver le chemin survivant et le chemin concurrent pour la prochaine itération à l'aide de bit de décision $d_{c,t}$ de l'état circulaire et la table de transition des branches (le bit de décision peut avoir la valeur 1 ou -1), l'état concurrent sera déduit à partir de l'autre valeur que le bit de décision a pris.

8 Passer à l'itération suivante.
- Stocker le bit de décision du chemin survivant dans un registre à décalage.
- Comparer le bit de décision du chemin survivant avec le bit de décision du chemin concurrent, si les deux bits sont égaux, le facteur de fiabilité Δ_t^1 qui correspond au chemin survivant à cet instant sera stocké, sinon le facteur de fiabilité ce de l'instant précédente Δ_t^0 qui sera stocké.

9 Revenir à l'étape (8) et suivre les mêmes démarches jusqu'à la mémoire sera vidée.

10 Les sorties du composant SOVA sont les bits estimés u' et la valeur douce "soft" $L(u) = \Delta \bullet u'$ où Δ est la valeur de fiabilité mise à jour finale.

3.3.7 Résultat de la comparaison entre l'algorithme SOVA et l'algorithme de Viterbi

Comme le montre la simulation de la figure 3.7 on voit bien que, le BER pour le décodeur turbo code avec l'algorithme de Viterbi est beaucoup plus important que celui

d'un décodeur utilisant l'algorithme SOVA, ceci revient à dire que la probabilité d'erreur binaire est moindre et ainsi décodeur SOVA corrige mieux les erreurs introduites par le canal de transmission. Cette simulation a été effectuée en utilisant des trames de longueurs égales à 400 bits, le codeur convolutif utilisé est un codeur récursif systématique de longueur de contrainte égale à K=3. Ici on n'a effectué qu'une seule itération pour pouvoir distinguer la différence entre le décodeur turbo code utilisant l'algorithme de Viterbi et celui utilisant l'algorithme SOVA.

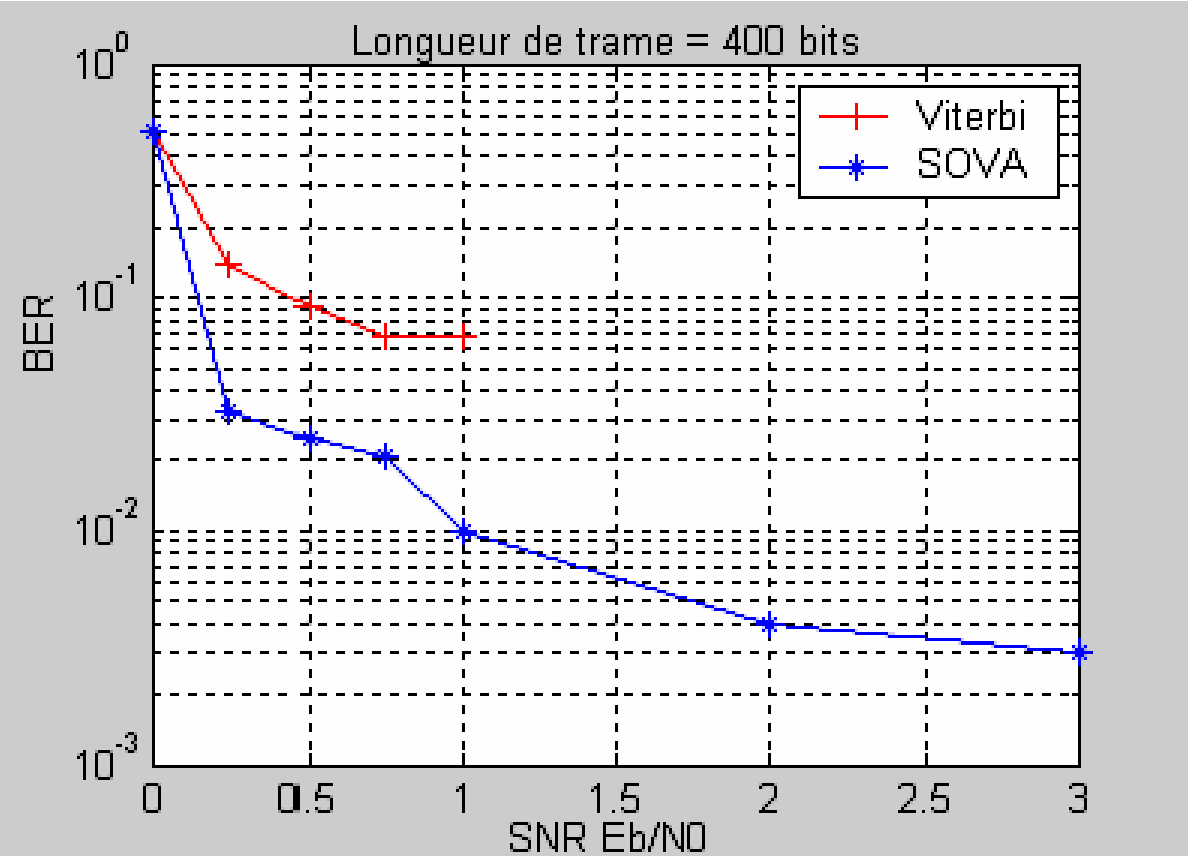

FIGURE 3.7 – Détermination du BER pour les deux algorithmes de décodage SOVA et viterbi, N=400bits.

3.4 Décodage itératif de Turbo codes à base de Sova

Le turbo décodeur itératif est composé de deux décodeurs de composants SOVA concaténés. La figure 3.8 montre la structure du turbo décodeur.

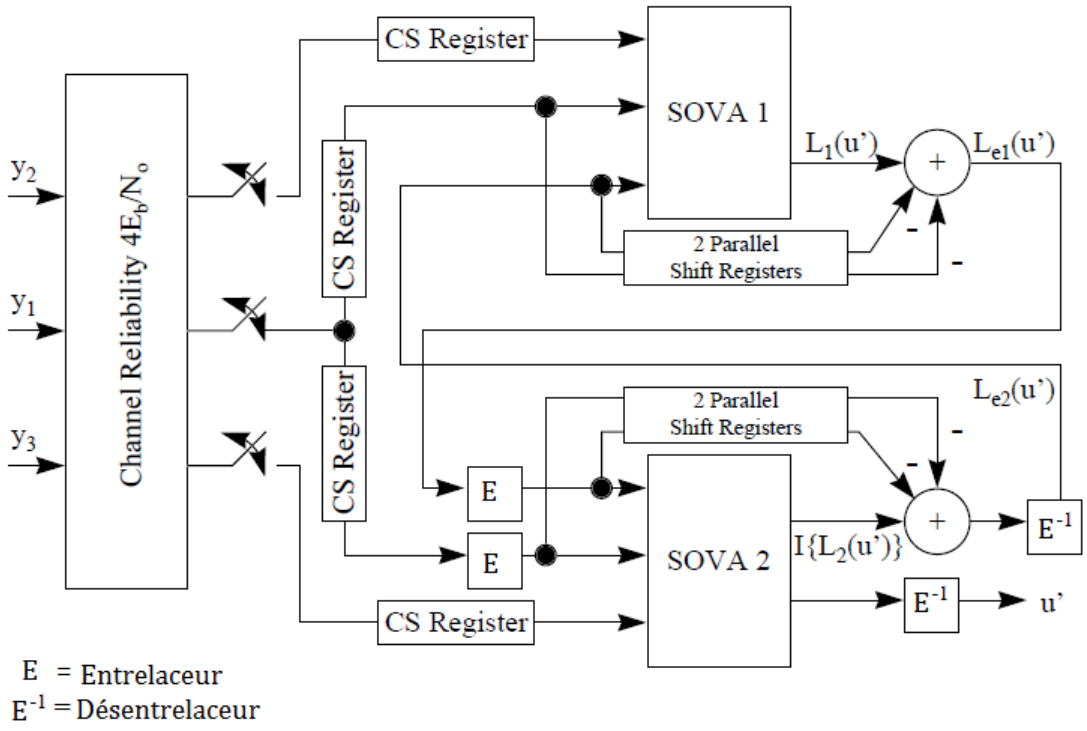

FIGURE 3.8 – Décodeur itératif à base de Sova

Le turbo décodeur traite les bits reçus du canal sous la forme d'une trame. Comme le montre la figure 3.8, les bits reçus du canal sont démultiplexés dans un flux systématique y1 et deux flux de contrôle de parité y_2 et y_3. Ces bits sont pondérés par la valeur de fiabilité du canal et chargés sur les registres à décalage circulaire CS. Les registres représentés sur la figure sont utilisés comme tampons pour stocker les séquences le temps qu'elles soient nécessaires pour le décodage. Les interrupteurs sont placés en position ouverte pour éviter que les bits de la trame suivante ne soient traités jusqu'à ce que la trame actuelle ait été traitée.

Le composant SOVA produit la valeur douce $L(u'_t)$ pour le bit u'_t estimé (à l'instant t). La valeur douce $L(u'_t)$ peut être décomposée en trois termes distincts.

$$L(u'_t) = L(u_t) + L_c y_{t,1} + L_e(u'_t) \tag{3.20}$$

où

- $L(u_t)$: la valeur a-priori produite par le décodeur SOVA précédent.
- $L_c y_{t,1}$: la valeur systématique reçue canal pondérée.
- $L_e(u'_t)$: la valeur extrinsèque produite par le décodeur SOVA actuel. L'information transmise entre les décodeurs SOVA est la valeur extrinsèque :

$$L_e(u'_t) = L(u'_t) - L(u_t) - L_c y_{t,1} \tag{3.21}$$

La figure 3.8 montre que le turbo décodeur itératif est une concaténation série en boucle fermée de deux décodeurs SOVA. Dans le décodage en boucle fermée présenté dans le schéma, chacun des décodeurs SOVA estime la séquence d'informations en utilisant un flux de contrôle de parité pondéré différent. Le turbo décodeur utilise en outre le décodage itératif pour fournir une fiabilité et des estimations a priori plus fiables à partir des deux différents flux de contrôle de parité pondérée, dans l'espoir d'obtenir de meilleures performances de décodage.

L'algorithme itératif de décodage du turbo-code pour la n-ème itération est le suivant :

1. Les séquences d'entrées du décodeur SOVA1 sont $4\frac{E_b}{N_0}y_1$ (systématique), $4\frac{E_b}{N_0}y_2$ (contrôle de parité) et $L_{e2}(u')$. La séquence de sortie de ce décodeur est la valeur douce "soft" $L_1(u')$. Pour la première itération, la séquence $L_{e2}(u')$ est nulle ($L_{e2}(u') = 0$) parce qu'il n'y a pas de valeur initiale a-priori (pas de valeur extrinsèque de SOVA 2).
2. L'information extrinsèque de SOVA 1 est obtenue par :

$$L_{e1}(u') = L_1(u') - L_{e2}(u') - L_c y_1 \tag{3.22}$$

   où $L_c = 4\frac{E_b}{N_0}$.
3. Les séquences $4\frac{E_b}{N_0}y_1$ et $L_{e1}(u')$ sont entrelacées et représentées par $I\left\{4\frac{E_b}{N_0}y_1\right\}$ et $I\{L_{e1}(u')\}$.
4. Le décodeur SOVA 2 reçoit les séquences $I\left\{4\frac{E_b}{N_0}y_1\right\}$ (systématique), $I\left\{4\frac{E_b}{N_0}y_3\right\}$(contrôle de parité déjà entrelacé par le turbo-codeur) et $I\{L_{e1}(u')\}$(information a-priori) et produit les séquences de sortie $I\{L_2(u')\}$ et $I\{u'\}$.
5. L'information extrinsèque de SOVA 2 est obtenue par :

$$I\{L_{e2}(u')\} = I\{L_2(u')\} - I\{L_{e1}(u')\} - I\{L_c y_1\} \tag{3.23}$$

6. Les séquences $I\{L_2(u')\}$ et $I\{u'\}$ sont dés-entrelacées et désignées par $L_2(u')$ et u' . $L_2(u')$ est renvoyé à SOVA 1 en tant que l'information a priori pour la prochaine itération et u' est la séquence estimée pour la n-ème itération.

Effet de nombre d'itération sur les performances des turbos codes

Le nombre d'itérations est un facteur clé dans la détermination des performances des turbos codes. Il permet à partir du processus de décodage itératif, l'amélioration continue de l'information à priori k L (u) des bits d'information. Il est clair d'après la figure 3.9 que l'augmentation du nombre d'itérations est suivie d'une amélioration considérable du taux d'erreurs binaire (BER) du turbo code considéré de l'ordre de 10^{-2} dans la première itération, pour atteindre l'ordre de 10^{-3} dans la cinquième itération pour un $E_b/N_0 =$ 2.0dB. Figure 3.9.

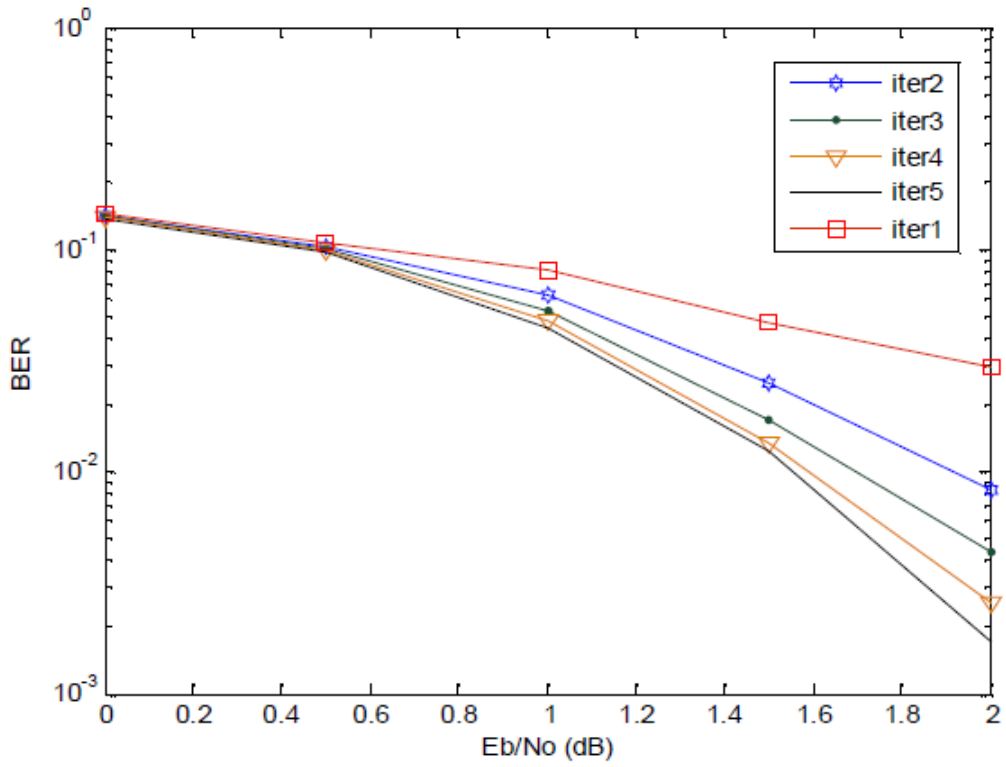

FIGURE 3.9 – Détermination du BER en fonction de $Eb/\ N0 = 2dB$ pour un certain nombre d'itérations en utilisant l'algorithme de décodage SOVA, N=1024 [26]

3.5 Conclusion

Dans ce chapitre, la notion de décodage itératif a été introduite, nous avons par la suite, décrit le procédé de décodage turbo à base de SOVA et développé les calculs nécessaires à l'aboutissement de l'opération de décodage. Finalement une architecture globale du Turbo-décodeur SOVA est proposée en fin de chapitre.
Au chapitre suivant, nous décrivons l'architecture du décodeur proposée avec une présentation détaillée de tous les composants qui constituent chaque unité.

Chapitre 4

Architecture retenue pour le décodeur SOVA

4.1 Introduction

Dans ce chapitre, nous décrivons l'architecture déduite de l'algorithme du décodeur SOVA(algorithme Viterbi à sortie douce) exposé dans le chapitre 3. nous passerons en détail les composants qui constituent chaque unité en définissant l'unité de contrôle qui synchronise et gère les autres unités du décodeur.

Dans notre travail, le décodeur est capable de décoder une séquence de bits codés avec un codeur convolutionnel systématique récursif et circulaire (CSRC) de rendement $R = 1/2$ une mémoire de taille $m = 3$ (donc une longueur de contrainte $L = 4$ et un nombre d'états $n = 2^3 = 8$) et de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)$.

4.2 Schéma proposé du décodeur

L'architecture de notre décodeur est constituée de deux blocs essentiels :une unité de calcul des métriques de chemin PMU et une unité trace back TBU. Ces modules sont représentés dans la Figure 4.1.
Le PMU se compose de 2^m unités élémentaires de calcul des métriques de chemin (PMeC) associé à chaque nœud, chaque PMeC comporte deux unités de calcul des métriques des branches BMC. Le module TBU (trace back) comprend deux blocs essentiels : un bloc appelé unité de décision, dont le rôle est d'extraire la séquence estimée ; l'autre bloc se défini comme étant l'unité de mise à jour de la valeur de fiabilité.

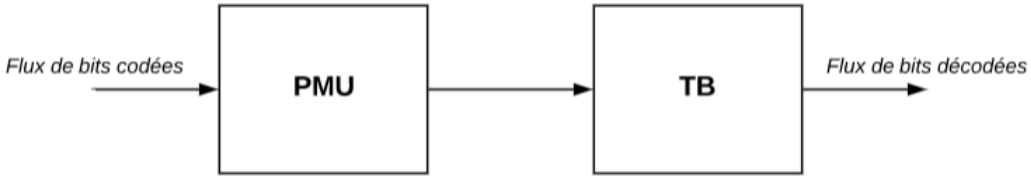

FIGURE 4.1 – Schéma bloc du décodeur.

La figure 4.2 illustre le schéma global de composant SOVA que nous avons proposé.

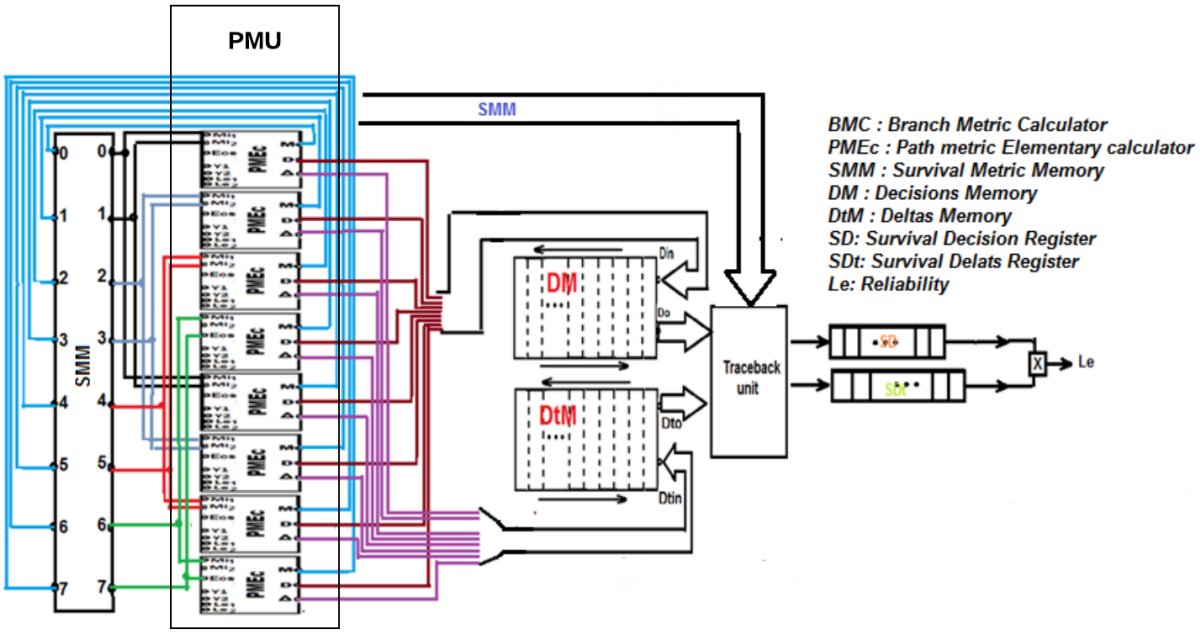

FIGURE 4.2 – Le schéma global du composant SOVA proposé.

Nous présentons dans ce qui suit l'architecture et le fonctionnement de chaque élément montré dans la figure 4.2

4.3 Unité de calcul des métriques de chemin (PMU)

L'unité de calcul des Métriques de chemin constituée principalement de huit blocs PMeC d'une façon que chaque PMeC représente un nœud dans le diagramme en treillis (voir la figure 4.2). Ce module stocke les métriques de branche calculées par les huit PMeC dans un registre à chargement parallèle et sortie parallèle (SMM) pour les réutilisées dans

l'itération suivante.

Chaque PMeC utilise deux métriques survivantes de l'itération précédente (stockées dans le registre SMM), ces dernières correspondent aux deux nœuds convergents vers le nœud associé à ce PMeC. La logique de convergence des nœuds diffère selon l'architecture du décodeur, dans notre cas, cette logique est illustrée dans la figure ci-dessous :

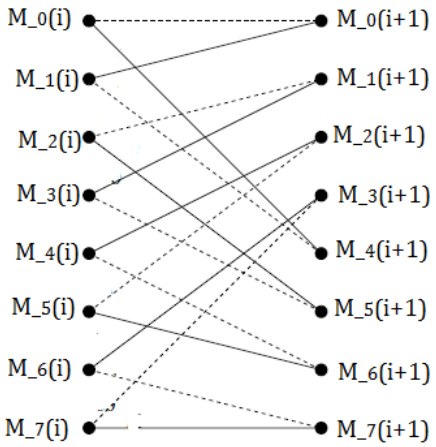

FIGURE 4.3 – Logique de convergence des nœuds dans le treillis.

Aussi cette unité délivre à chaque instant i, 2^m valeurs de fiabilité et 2^m bits de décision afin de stocker ces derniers dans deux mémoires DTM (Deltas Memory) et DM (Decisions Memory) respectivement.

4.3.1 Unité de calcul des métriques des branches BMC

L'unité BMC est une unité principale dans le schéma global, elle permet de calculer la métrique cumulée à chaque instant pour un état donné. Les entrées de cette unité sont les métriques de l'état i à l'instant t-1, les bits codés bruités sortant du canal et l'information a priori (extrinsèque) délivrée par le décodeur précédent.

Le calcul des métriques se fait par la relation qui suit :

Pour le premier composant SOVA :

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)}\ I^{-1}\{L_c\ y_{t,1}\} + x_{t,2}^{(m)}\ L_c\ y_{t,2} + u_t^{(m)}\ L_{e2}(u'_t) \qquad (4.1)$$

Pour le deuxième composant SOVA :

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)}\ L_c\ y_{t,1} + x_{t,2}^{(m)}\ L_c\ y_{t,3} + u_t^{(m)} I\{L_{e1}(u'_t)\} \tag{4.2}$$

Le schéma détaillé du bloc BMC est illustré dans la Figure 4.4.

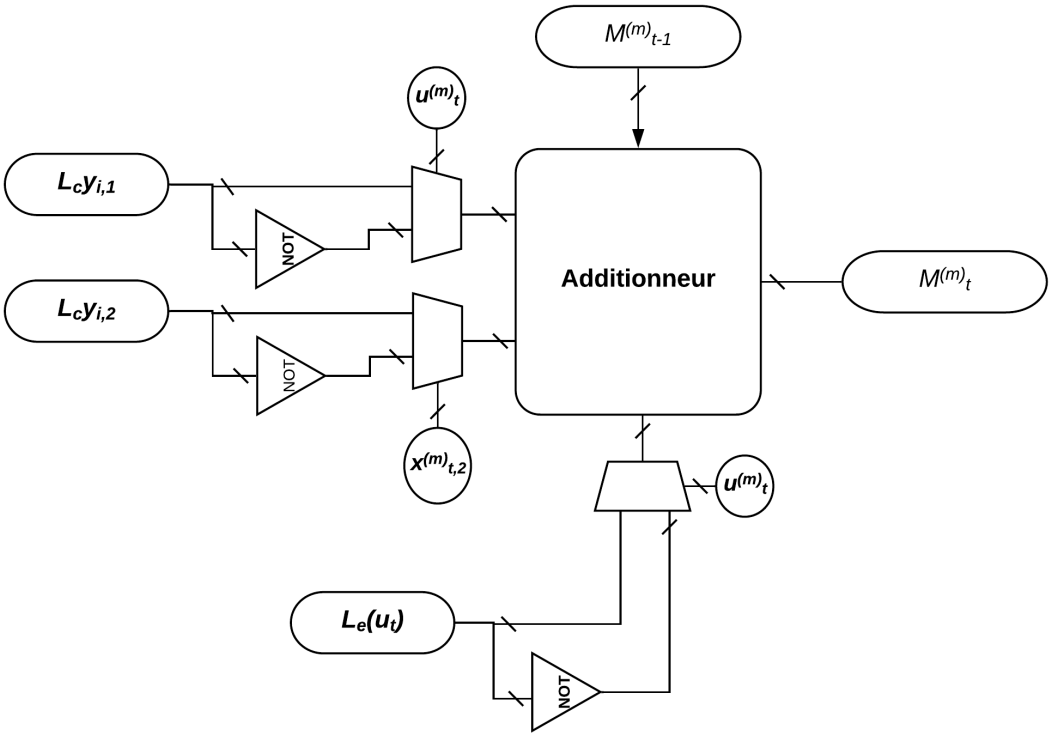

FIGURE 4.4 – Schéma détaillé du BMC.

4.3.2 Unité élémentaire de calcul des métriques de chemin (PMeC)

Cette unité est constituée de deux unités BMC qui servent à calculer les deux métriques de branche convergentes vers le même nœud. L'unité PMeC réalise pour un nœud donné j à chaque instant les trois opérations suivantes :

- Comparer les deux métriques de branche $M_t^{(1)}$ et$M_t^{(2)}$ qui convergent vers le même état en gardant le maximum entre eux comme étant la métrique de branche survivante SM_t.
- Déduire le bit de décision $d_{j,t}$ à partir de la table de transition présentée au dessous après avoir comparé $M_t^{(1)}$ et $M_t^{(2)}$.
- Calculer le facteur de fiabilité $\Delta_{j,t}$ par l'équation :

$$\Delta_{j,t} = \frac{1}{2}|M_t^{(1)} - M_t^{(2)}| \tag{4.3}$$

État d'arrivée	État de départ (T1)	État de départ (T2)	Bit de décision pour T1	Bit de décision pour T2
0	0	1	-1	1
1	2	3	-1	1
2	4	5	1	-1
3	6	7	1	-1
4	0	1	1	-1
5	2	3	1	-1
6	4	5	-1	1
7	6	7	-1	1

TABLE 4.1 – Les bits de décisions associés a chaque transition

La figure 4.5 représente l'architecture de PMeC que nous avons proposé pour réaliser les taches citées précédemment.

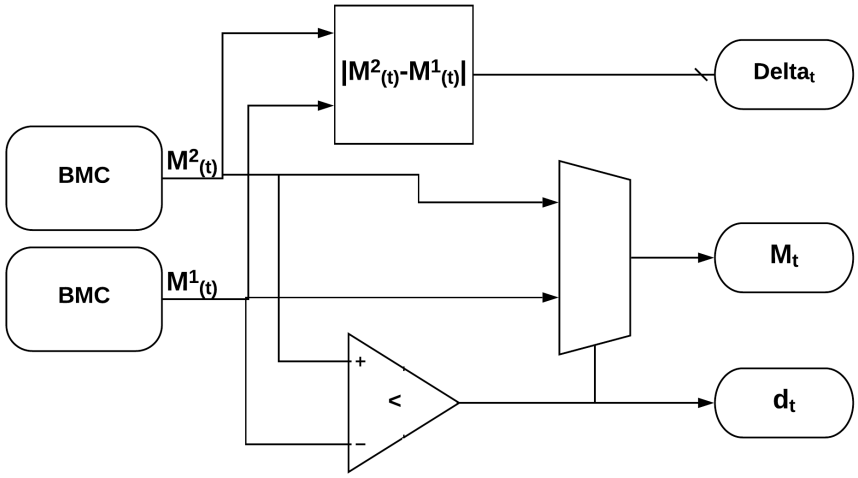

FIGURE 4.5 – Schéma détaillé du PMeC.

4.4 Unité de Trace back TBU

Après avoir fini tout le calcul des métriques cumulées $M_{j,t}$, les bits de décision $d_{j,t}$ et les valeurs de fiabilité $\Delta_{j,t}$, l'unité trace back est activée par un signal de contrôle pour commencer à tracer le chemin survivant ainsi que le chemin concurrent afin de donner les bits estimés u'_t et mettre à jour les valeurs de fiabilités.
Cette unité utilise les bits de décision et les valeurs de fiabilité générées par l'unité de

calcul des métriques pour décoder la séquence reçue et mettre à jour la valeur de fiabilité. Il y'a lieu de noter que cette unité travaille seulement après que le PMU parcourt tout le treillis.
L'unité Trace back comporte cinq blocs essentiels :

- Une unité de décision DU qui sert à donner les bits estimés.
- Une unité mise à jour de fiabilité.
- Bloc de recherche de l'état circulaire.
- Unité de recherche de l'état survivent.
- Une fonction de recherche du Maximum.

La figure 4.6 illustre l'architecture de l'unité de trace back que nous avons déjà expliqué précédemment.

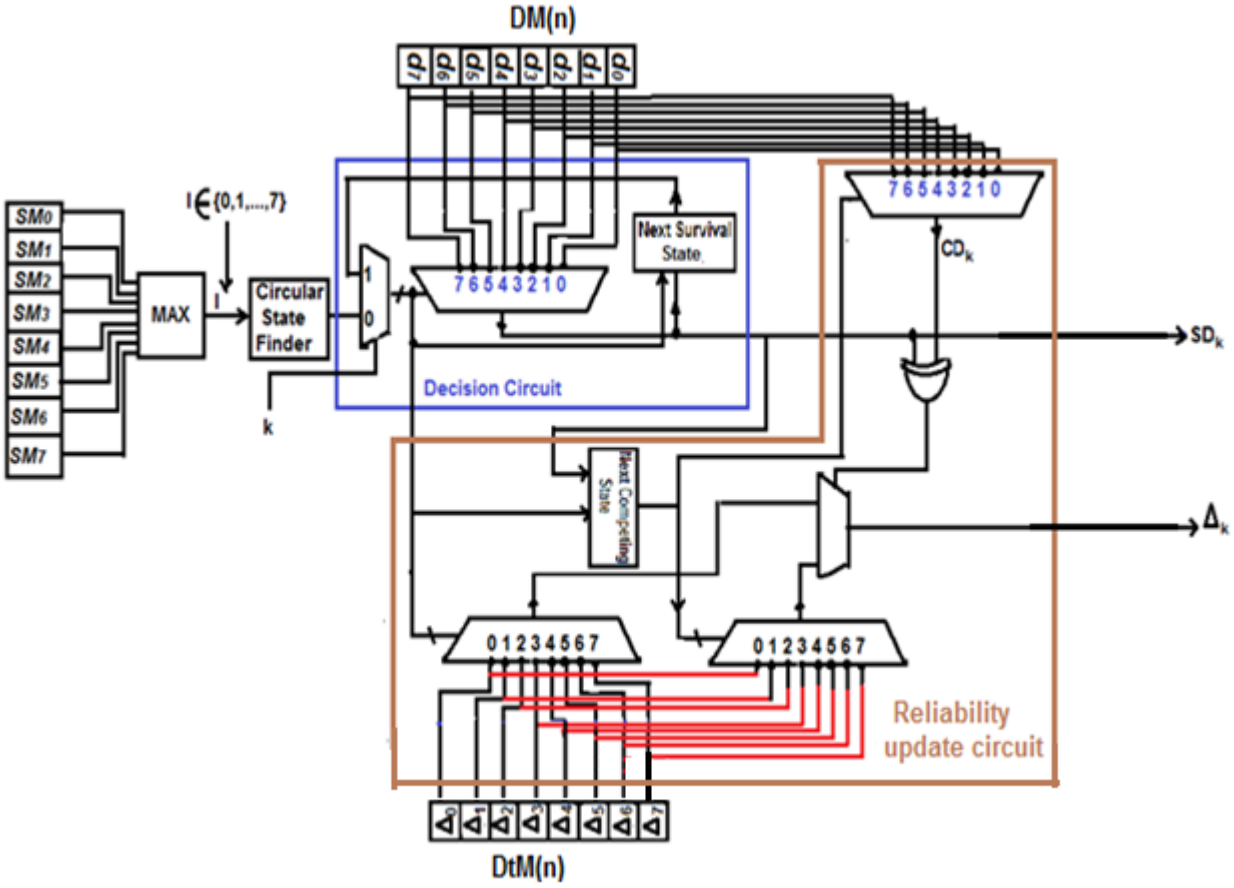

FIGURE 4.6 – Schéma détaillé du TBU.

4.4.1 Unité de decision DU

Cette approche consiste à remonter le treillis à partir de l'état circulaire en utilisant les bits de décisions délivrés par la PMU pour tracer le chemin le plus vraisemblable à la séquence reçue. Cette unité est la pièce maitresse du décodage. Il comporte l'unité de calcule de l'état survivent suivant et un multiplexeur pour sélectionner le bit de décision survivant à chaque itération (voir la figure 4.6).

L'état de départ du traçage correspond à l'état circulaire, l'état qui suit et avec lequel on doit adresser la mémoire du Trace Back (celle qui est en mode lecture) pour avoir le bit de décision survivant u' est déterminé à partir de l'état actuel et le bit de décision survivant à cet instant.

A chaque instant, on stocke le bit de sortie dans la pile LIFO, une fois le Trace Back est terminé, on doit décharger la pile pour avoir la séquence décodée u' dans l'ordre.

4.4.2 Unité de mise à jour des valeurs de fiabilité

Cette unité joue un rôle très important vu que la mise à jour des valeurs de fiabilité se fait à ce niveau.

Cette opération s'effectue en comparant le bit de décision du chemin survivant avec le bit de décision du chemin concurrent, si les deux bits sont égaux, le facteur de fiabilité qui correspond au chemin survivant à cet instant sera gardé, sinon, le facteur de fiabilité de l'instant précédente sera maintenu. A chaque instant on stocke la valeur de fiabilité choisie Δ_t^i dans la pile LIFO (DtM). Une fois le Trace Back est terminé, on doit décharger la pile pour avoir les valeurs de fiabilité dans l'ordre afin de calculer la valeur extrinsèque par la relation suivante :

$$L(u) = \Delta \bullet u' \tag{4.4}$$

4.4.3 Bloc de recherche de l'état circulaire

Ce bloc sert à trouver l'état circulaire à l'aide de la position de maximum de la dernière itération lors du parcours de treillis et la taille de la séquence à estimer, cela est résumé dans la table ci-dessous. Il est à noter que ce bloc fonctionne seulement au début de l'opération de Trace Back.

$S_k^0 \backslash\ k\ mod\ 7$	1	2	3	4	5	6
0	0	0	0	0	0	0
1	6	3	5	4	2	7
2	4	7	3	1	5	6
3	2	4	6	5	7	1
4	7	5	2	6	1	3
5	1	6	7	2	3	4
6	3	2	1	7	4	5
7	5	1	4	3	6	2

TABLE 4.2 – La table fournissant de l'état de circulation.

4.4.4 Unité de recherche de l'état survivent

Cette fonction est la pièce maitresse de l'opération de trace back, elle a pour but de trouver l'état survivant à chaque instant à l'aide de l'état survivant actuel et son bit de décision. La logique de l'opération d'adressage est illustrée dans la table ci-dessous :

État actuel	Bit de décision 1	L'état adressé 1	Bit de décision 2	L'état adressé 2
0	-1	0	1	1
1	-1	2	1	3
2	1	4	-1	5
3	1	6	-1	7
4	1	0	-1	1
5	1	2	-1	3
6	-1	4	1	5
7	-1	6	1	7

TABLE 4.3 – La table d'adressage pour le trace back.

4.4.5 Fonction de recherche du Maximum

Cette fonction a pour but de chercher la position de maximum dans le registre de stockage des métriques de chemin à la fin de la séquence reçue. Cette position est nécessaire pour désigner l'état circulaire afin d'initialiser le point de départ de Trace Back à cette position. Cette fonction est illustrée dans l'organigramme de la Figure 4.7.

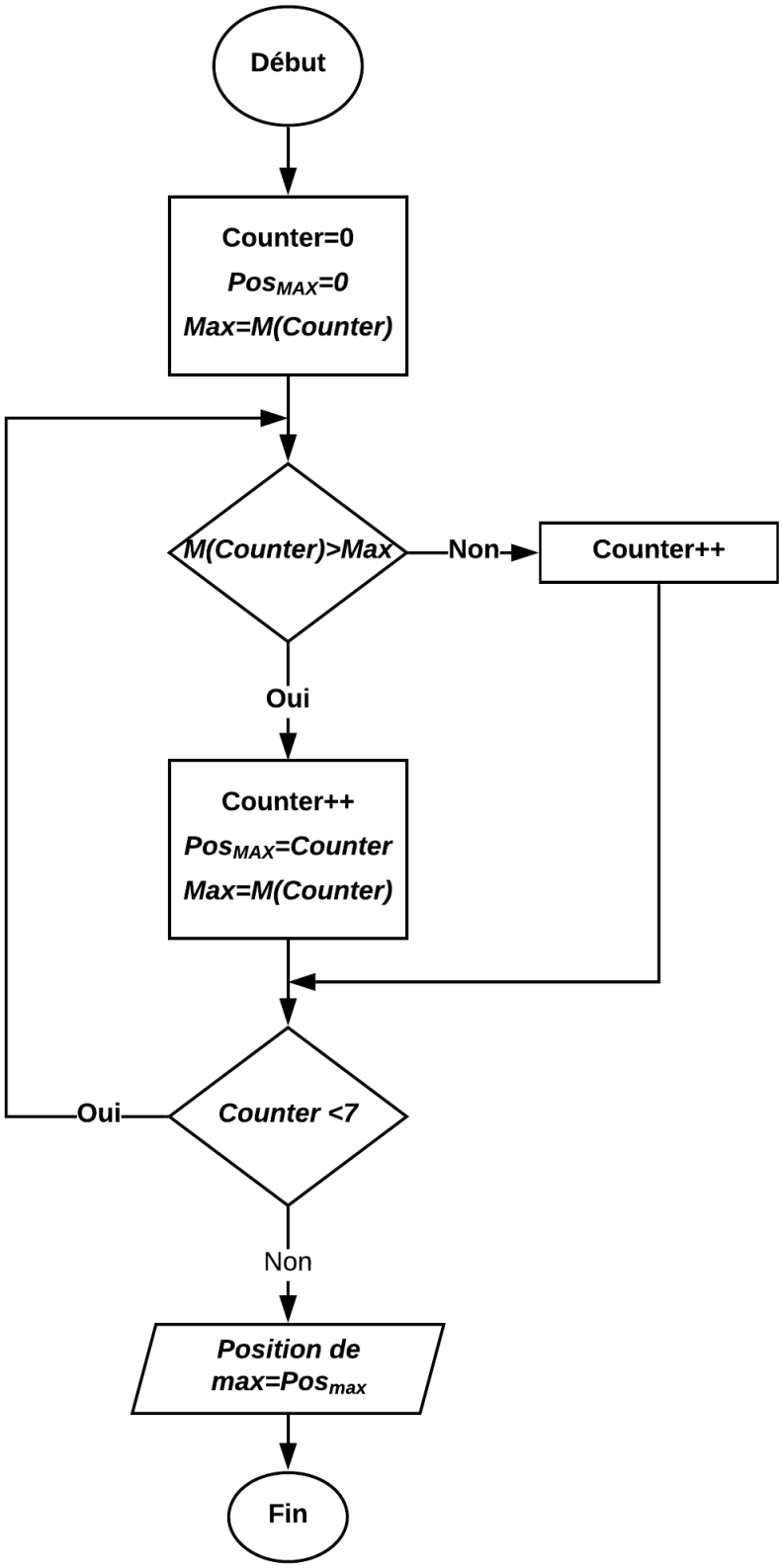

FIGURE 4.7 – Organigramme de la fonction de recherche du maximum.

4.5 Mémoire des métriques survivantes (SMM)

La mémoire des métriques survivantes (SMM) est une mémoire temporaire d'entrées et sorties parallèles, elle sert à garder les métriques cumulées le temps nécessaire pour les utiliser dans le prochain calcul.

4.6 Circuit de contrôle

Dans un système logique, l'unité de contrôle commande et contrôle le fonctionnement d'un système. Une unité de contrôle est un circuit logique séquentiel qui réalise un automate fini, plus précisément une machine d'état (de Moore ou de Mealy), qui génère des signaux de contrôle pour piloter les différents modules de traitement de la donnée.

La machine d'état du composant SOVA est illustrée dans la figure 4.8 , elle pilote tous les modules du décodeur SOVA, chaque étape de décodage est associée à un état dont leurs illustrations sont les suivantes :
PMU activée : la machine d'état passe à l'état 1 lorsqu'elle reçoit un signal go, cet état permet d'activer l'unité PMU pendent un nombre déterminé d'itérations N.
TB Validé : Cet état a lieu lorsque le PMU achève les N itérations, il permet d'activer l'unité de trace back pendent N itérations.
SD & CD : Cet état est réalisé à chaque itérations de l'opération de Trace back, il repose sur la comparaison de bit de décision du chemin survivant avec le bit de décision du chemin concurrent pour vérifier si la valeur de fiabilité nécessite une mise à jour.
Fiabilité update : dans le cas où les deux bit de décision respectivement du chemin survivant et du chemin concurrent sont différents, une mise à jour de la valeur de fiabilité est nécessaire, cette opération est présentée par l'état Fiabilité update.
SDt : cet état a lieu si les bits de décision sont égaux, autrement dit le valeur de fiabilité ne nécessite pas une modification.
RESULTAT : lorsque l'unité de Trace back accomplit le nombre N d'itérations, la machine passe à l'état RESULTAT qui donne les résultats de l'élément décodeur SOVA.

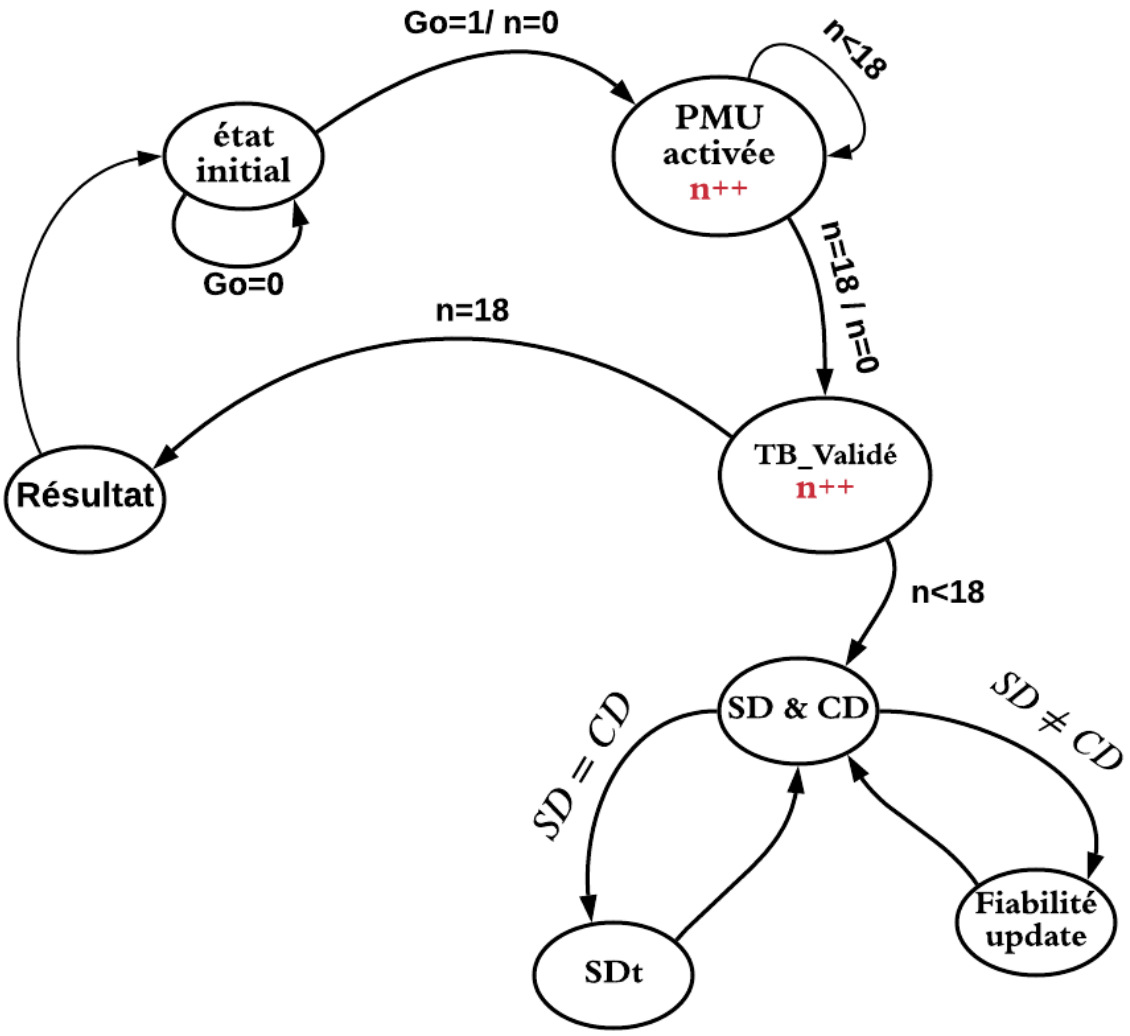

FIGURE 4.8 – Machine d'état du composant SOVA.

Chapitre 5

Implémentation

5.1 Introduction

Dans ce dernier chapitre, nous présentons une implémentation sur FPGA d'une architecture d'un turbo décodeur itératif basé sur l'algorithme de Viterbi à sortie douce SOVA, pour cela, nous présentons la carte de développement Nexys2 de Xilinx et ses performances. Ensuite, nous présentons l'architecture des différents blocs obtenus après la synthèse. Finalement, nous terminons par une discussion du rapport de synthèse généré par ISE-Xilinx.

5.2 Environnement de développement

5.2.1 Langage VHDL

Le VHDL (VHSIC Hardware Description Language) est un langage de description de matériel, il a été créé pour décrire des systèmes numérisés destinés à une Implantation dans des circuits logiques programmables (CPLD, FPGA) et circuits ASIC, il remplace la saisie de schéma traditionnelle.

5.2.2 Circuits FPGA

Field-Programmable Gate Array (un réseau de portes logiques programmables) est un circuit intégré conçu pour être reconfigurer après production. Sa configuration se fait grâce à un langage de description de matériel (HDL), ou bien un diagramme de circuit, cette dernière méthode est rarement utilisée de nos jours.

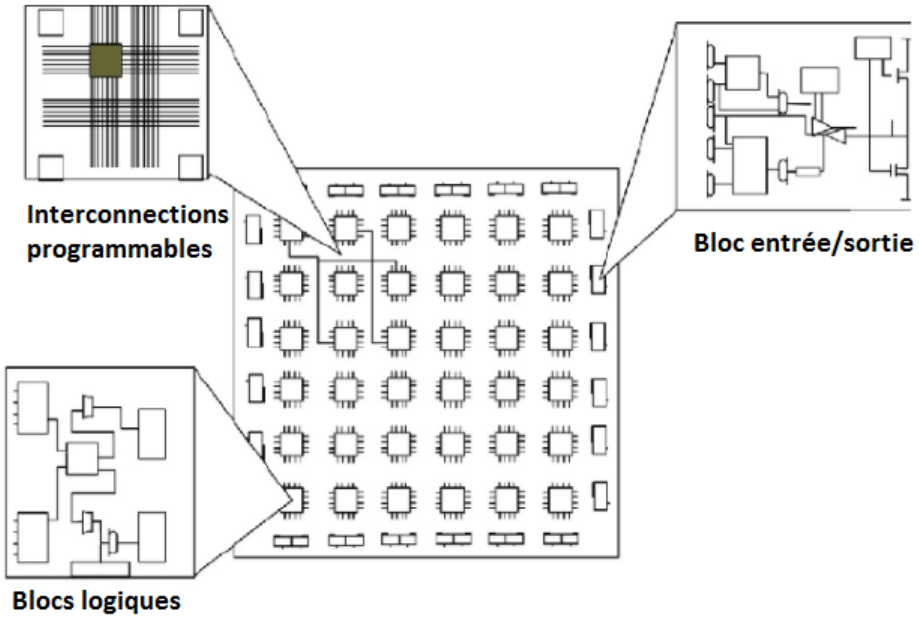

FIGURE 5.1 – Architecture interne d'un circuit FPGA .

5.2.3 Description de la carte FPGA- Nexys 2

Nous avons l'intention d'implémenter un turbo décodeur itératif basé sur l'algorithme de Viterbi a sortie douce SOVA sur la carte FPGA- Nexys -2, Pour cela il est important de connaitre les caractéristiques de la carte FPGA, qui va être le support de notre architecture.

Caractéristiques :

- FPGA Xilinx Spartan 3E à porte 500K.
- Configuration FPGA basée sur USB2 et transferts de données haute vitesse (à l'aide du logiciel gratuit Adept Suite Software).
- Alimenté par USB (des piles et / ou une prise murale peuvent également être utilisées).
- 16 Mo de mémoire PSDRAM Micron et 16 Mo de ROM Intel StrataFlash.
- Xilinx Platform Flash pour les configurations FPGA non volatiles.
- Alimentations à découpage efficaces (convient aux applications alimentées par batterie).
- Oscillateur 50 MHz plus socket pour le deuxième oscillateur.
- 60 entrées / sorties FPGA acheminées vers des connecteurs d'extension (un connecteur Hirose FX2 haute vitesse et quatre en-têtes à 6 broches).
- 8 DEL, afficheur à 7 chiffres à 4 chiffres, 4 boutons, 8 commutateurs à glissière.

- Expédié dans une mallette en plastique avec un câble USB.

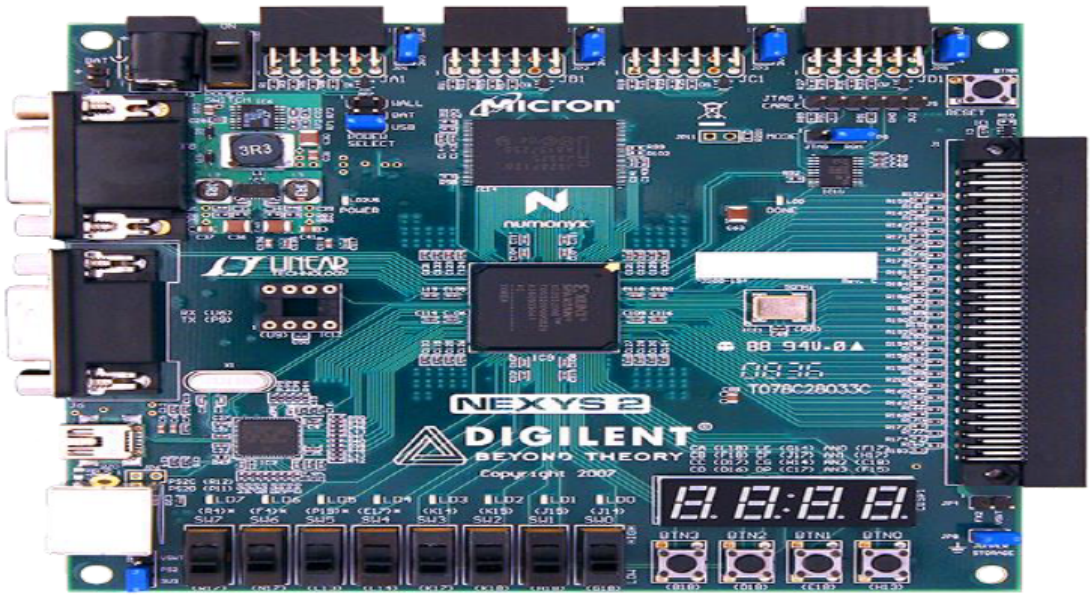

FIGURE 5.2 – La carte FPGA Nexys 2.

5.2.4 Outil de développement

L'ISE Design Suite comprend Xilinx Platform Studio (XPS), Kit de développement logiciel (SDK). ISE fournit les outils, les technologies et le flux de conception familier pour obtenir des résultats de conception optimum. Il s'agit notamment de la cadence d'horloge intelligente pour la réduction de puissance dynamique, la préservation du design pour la répétabilité du temps et une option de reconfiguration partielle pour une plus grande flexibilité, taille, puissance et réduction des coûts du système.

5.3 Les blocs générés dans Xilinx ISE

5.3.1 Unité de calcul des métriques des branches (BMC)

Ce bloc est élément principale dans le schéma global. Il permet de calculer la métrique de transition possible cumulée à chaque instant pour un état donné.La figure 5.3 illustre le schéma RTL du bloc de BMC tel qu'il est généré dans Xilinx ISE.

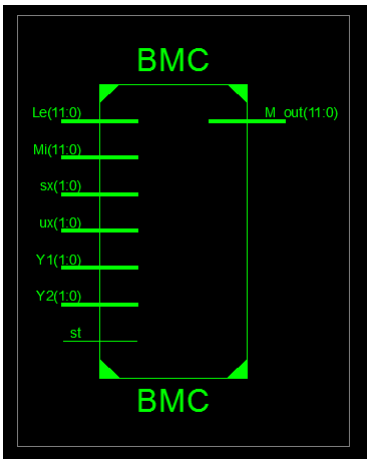

FIGURE 5.3 – Bloc de l'unité de calcul des métriques des branches (BMC) générée par Xilinx-ISE.

5.3.2 Unité élémentaire de calcul des métriques de chemin (PMeC)

Ce bloc sert à fournir pour un nœud donné dans le treilles la métrique de branche survivante, le bit de décision associé à la transition survivante et la valeur de fiabilité (cela a été déjà expliqué dans le chapitre 4. Le schéma RTL donné par la figure 5.4 montre le bloc du PMeC tel qu'il est généré dans Xilinx ISE après la synthèse.

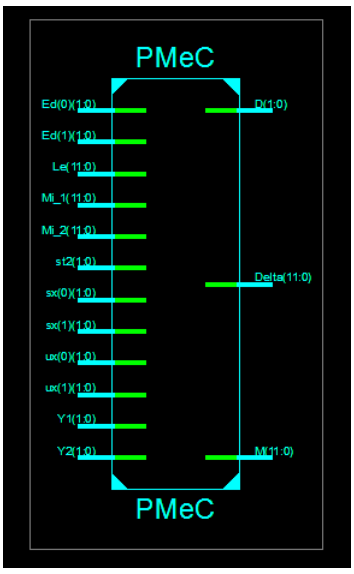

FIGURE 5.4 – Bloc de l'unité élémentaire de calcul des métriques de chemin (PMeC) générée par Xilinx-ISE .

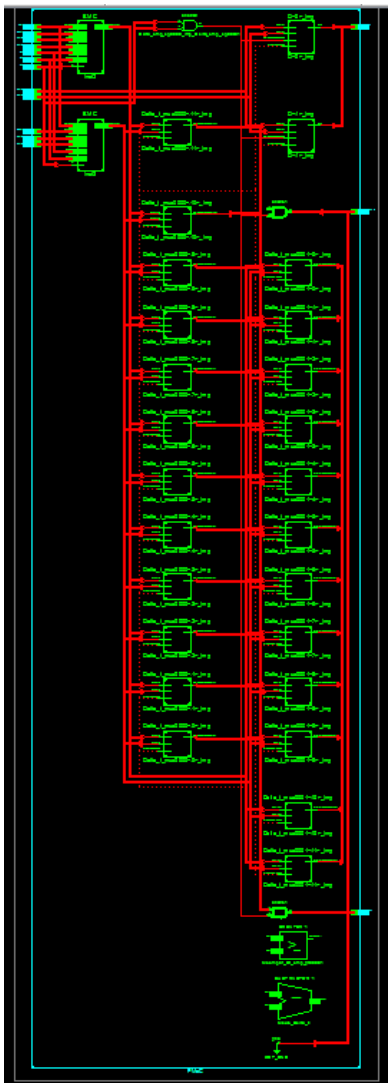

FIGURE 5.5 – Schématique du bloc de l'unité élémentaire de calcul des métriques de chemin (PMeC) générée par Xilinx-ISE.

5.3.3 Unité de calcul des métriques de chemin (PMU)

Ce bloc est l'élément responsable de fournir à chaque top d'horloge 8 bits de décision et 8 valeurs de fiabilité, Il reçoit les bits codés bruités sortants du canal, l'information a priori délivrée et les métriques accumulées de l'instant t-1.
Le bloc PMU est constitué principalement de huit blocs PMeC par lesquels le calcul parallèle est effectué.
Le schéma RTL donné par la figure 5.6 montre le bloc du PMU tel qu'il est généré dans Xilinx ISE.

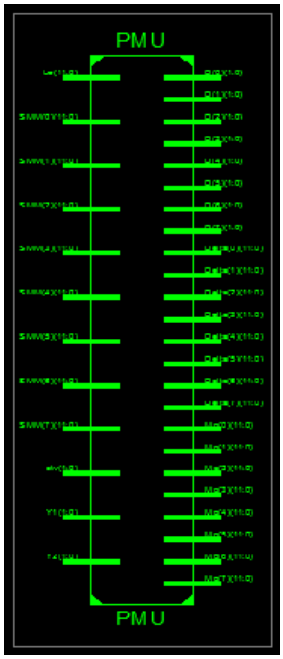

FIGURE 5.6 – Bloc de l'unité de calcul des métriques de chemin (PMU) générée par Xilinx-ISE.

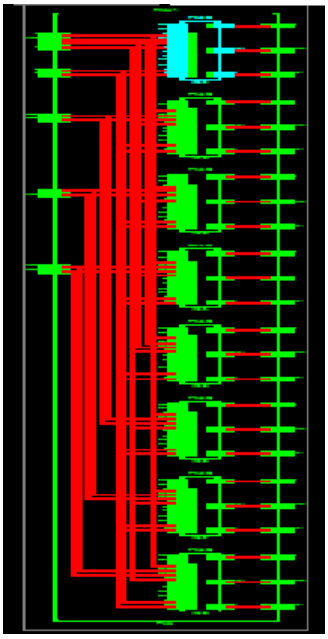

FIGURE 5.7 – Schématique du bloc de l'unité de calcul des métriques de chemin (PMU) générée par Xilinx-ISE .

5.3.4 Unité de Trace back TBU

Après avoir fini tout le calcul des métriques cumulées M, les bits de décision et les valeurs de fiabilité, ce bloc est activé par un signal de contrôle pour commencer à tracer le chemin survivant ainsi que le chemin concurrent afin de délivrer les bits estimés u et mettre à jour les valeurs de fiabilité. Le bloc TBU reçoit les bits de décision et les valeurs de fiabilité délivrées par le PMU qui ont été stockés dans deux mémoires DTM et DMM respectivement, par conséquent, cette unité travaille seulement après que le PMU parcourt tout le treilles.
Ce bloc comporte une unité de décision qui sert à donner les bits estimés et une unité mise à jour de fiabilité. La figure suivante montre le schéma RTL du bloc PMEc tel qu'il est généré dans Xilinx ISE après la synthèse.

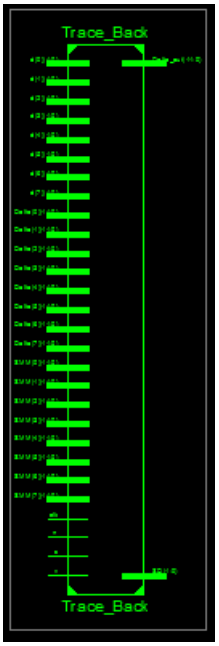

FIGURE 5.8 – Bloc de l'unité de Trace back TBU générée par Xilinx-ISE.

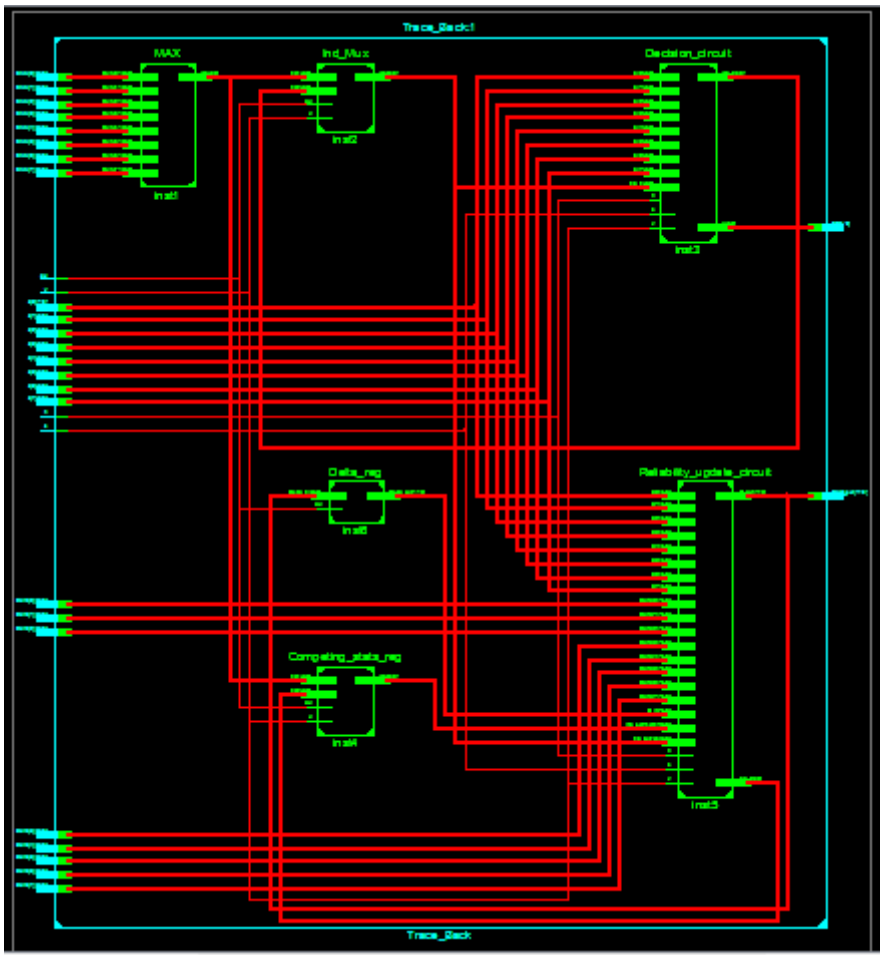

FIGURE 5.9 – Schématique du bloc de l'unité de Trace back TBU générée par Xilinx-ISE.

5.3.5 Bloc du décodeur itératif à base de SOVA

La synthèse du code vhdl qui décrit l'architecture du décodeur SOVA est présentée dans la figure 5.10. La figure suivante montre le schéma RTL du bloc de décodeur itératif à base de SOVA. Nous pouvons directement remarquer à l'œil nu l'immense complexité de l'architecture, même si la taille du code est réduite. Cette conception est celle qui garantie le meilleur débit possible.

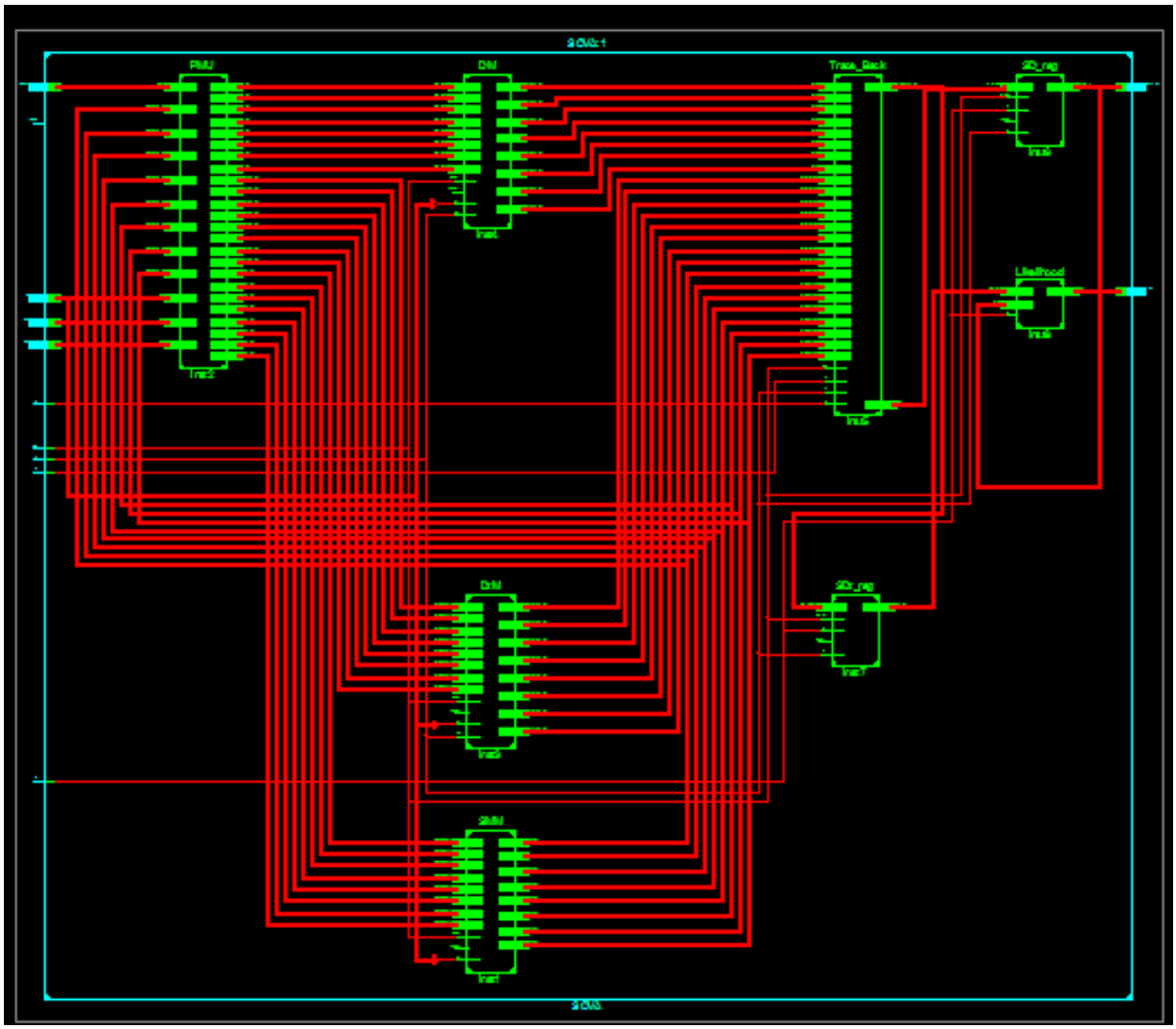

FIGURE 5.10 – Schématique du bloc du composant décodeur SOVA générée par Xilinx-ISE.

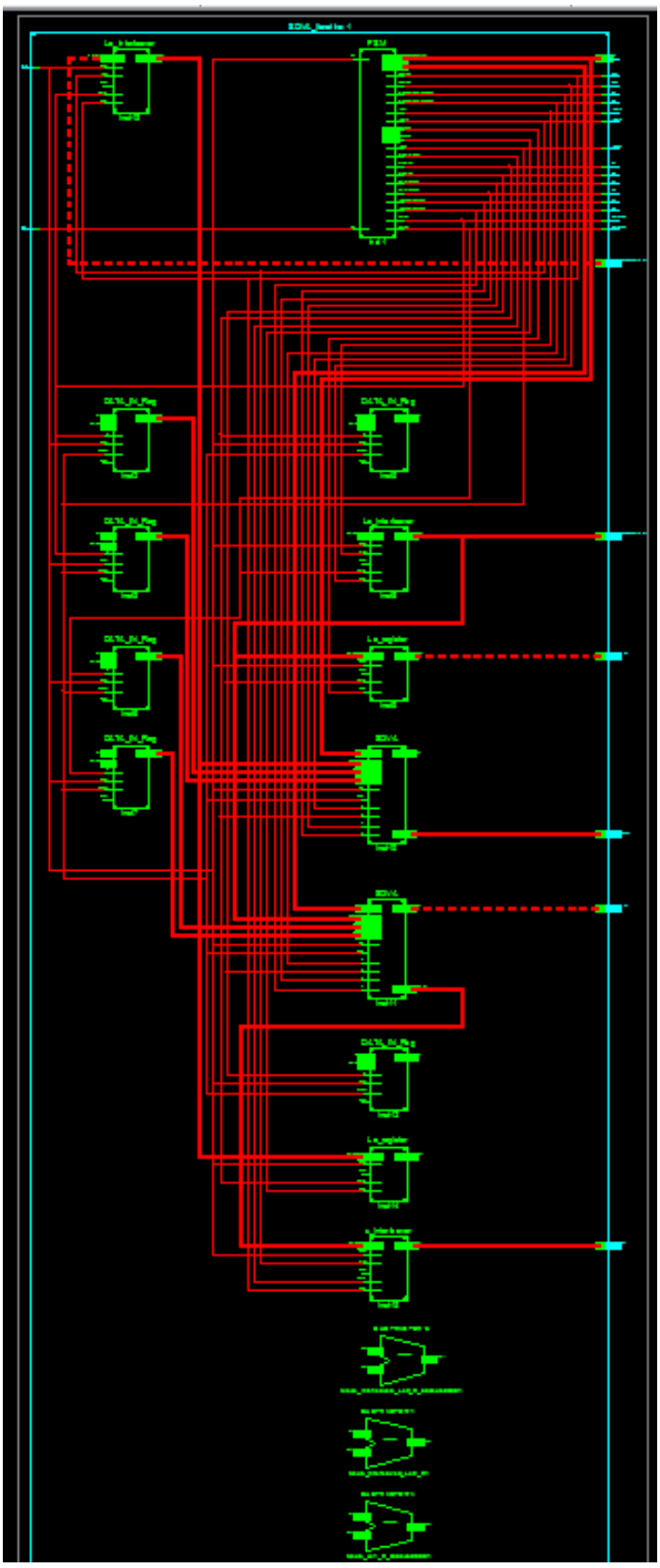

FIGURE 5.11 – Schématique du bloc du bloc de décodeur itératif à base de SOVA générée par Xilinx-ISE.

5.4 Simulation du décodeur

Dans la simulation, toutes les données seront représentées en complément à deux (CA2), 9 bits pour les valeurs extrinsèques et 2 bits pour les bits estimés après le décodage.
Considérons le message d'entrée

$$u = \{1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1\}$$

En choisissant l'entrelaceur (longueur L=19) « inverseur », la séquence d'entrée permutée devient :

$$I\{u\} = \{1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1\}$$

Le turbo codeur génère les sorties suivantes :

$$x_1 = \{0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1\}$$

$$x_2 = \{1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1\}$$

$$x_3 = \{1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1\}$$

Après la modulation, les séquences codées deviennent :

$$x_1 = \{-1, -1, 1, -1, 1, 1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, 1, 1\}$$

$$x_2 = \{1, 1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1\}$$

$$x_3 = \{1, 1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1\}$$

En prenant $L_c = 4$ ($E_b/N_0 = 1$)
Dans ce qui suit, nous allons tester les performances de notre décodeur avec différents types d'erreurs à la séquence reçue par le récepteur :

- Cas d'une erreur double concaténée :

$$y_1 = \{-1, -1, 1, 0, 0, 1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, 1, 1\}$$

$$y_2 = \{1, 1, 1, 0, 0, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1\}$$

$$y_3 = \{1, 1, 1, 0, 0, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1\}$$

- Cas d'une erreur double éparpillée :

$$y_1 = \{-1, -1, 1, -1, 0, 1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, 0, 1\}$$

$$y_2 = \{1, 1, 1, 1, 0, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, 0, 1\}$$

$$y_3 = \{1, 1, 1, 1, 0, 1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, 0, 1\}$$

Les ’0’ sont des erreurs introduites par le canal.

Les résultats de la simulation fonctionnelle sur ISE Simulator des deux cas traités sont représentés sur les figures 5.12, 5.13.

En analysant les résultats obtenus, les erreurs ont été détectées puis corrigées et les informations extrinsèques ont été calculées avec précision. Le turbo décodeur itératif converge vers le message correct après trois itérations pour le cas d’une erreur double concaténée et après deux itérations pour le cas d’une erreur double éparpillée.

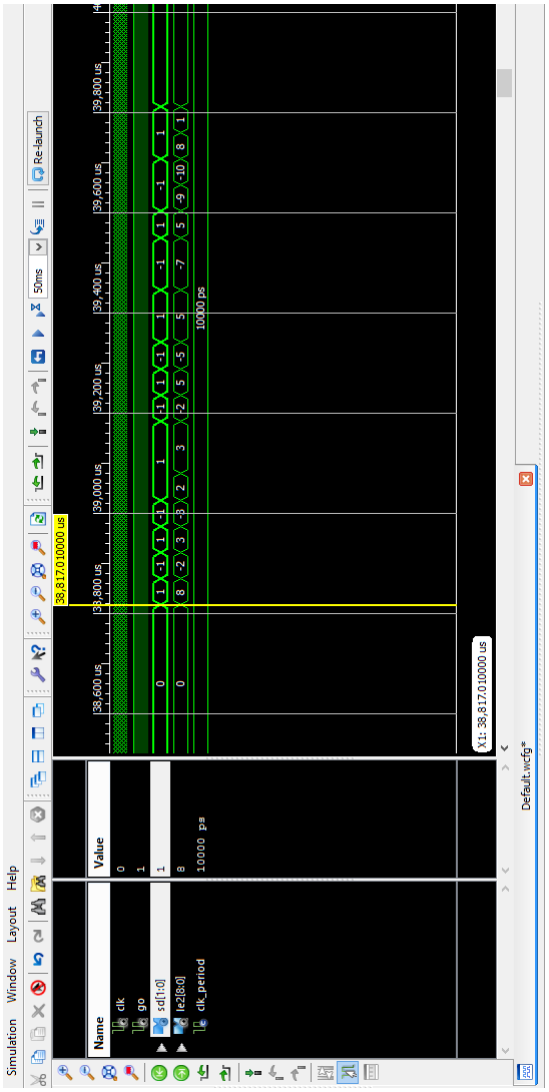

FIGURE 5.12 – Résultat de la simulation fonctionnelle du décodeur dans le cas d’une erreur double concaténée sur ISE Simulator.

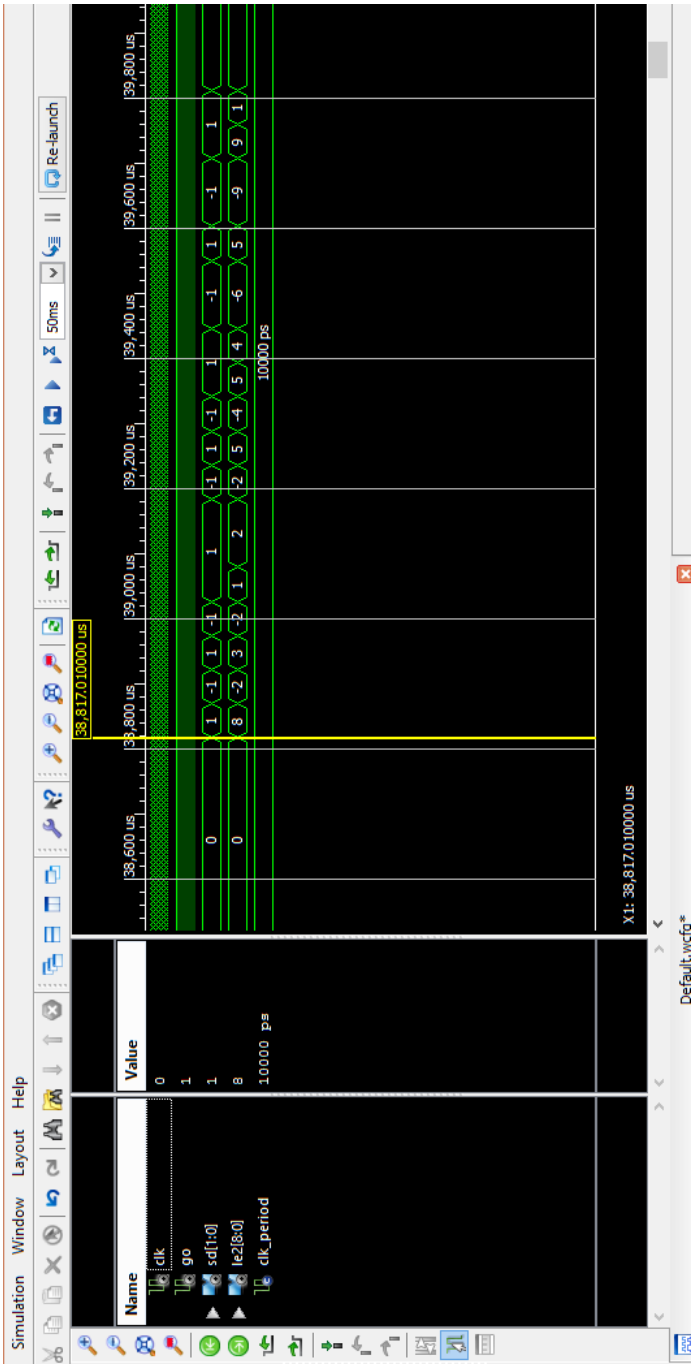

FIGURE 5.13 – Résultat de la simulation fonctionnelle du décodeur dans le cas d'une erreur double éparpillée sur ISE Simulator.

5.5 Test sur l'FPGA

Le fonctionnement du turbo-décodeur est régi par une machine d'état dont le rôle est d'établir le bon ordonnancement du processus de décodage itératif décrit dons le Chapitre 3.

La figure suivante correspond à la machine d'état de turbo décodeur itératif. Cette machine d'état a pour but de synchroniser les deux éléments SOVA qui constituent le décodeur itératif, ainsi assurer un nombre d'itération qui est égale à 7.

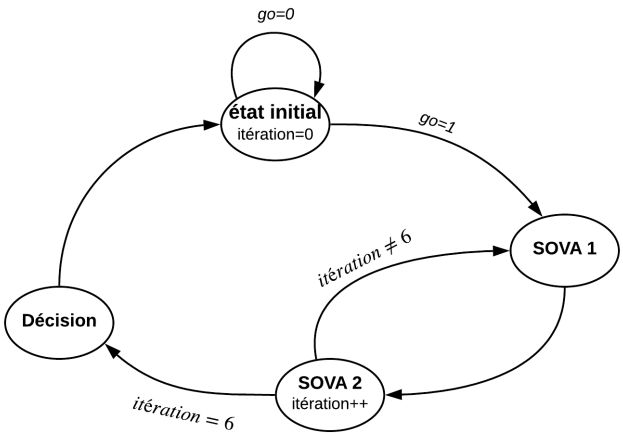

FIGURE 5.14 – Machine d'état du turbo décodeur itératif SOVA.

Pour pouvoir visualiser les résultats obtenus sur la carte FPGA, on utilise les quatre afficheurs 7-Segments de la carte FPGA, chaque 7-Segments affiche quatre bits du résultat en hexadécimal, les trois bits du résultat restants seront affichés sur les LEDs.

La figure 5.15 présente le résultat final du décodage sur la carte Nexys 2.
Le résultat affiché sur les 7-segments est 7593 :
Où :
$3 = (0011)_16$
$9 = (1001)_16$
$5 = (0101)_16$
$7 = (0111)_16$
Sur les LEDs le résultat en binaire est $(101)_2$
Le résultat obtenu correspond bien au résultat espéré qui est la séquence émise u.

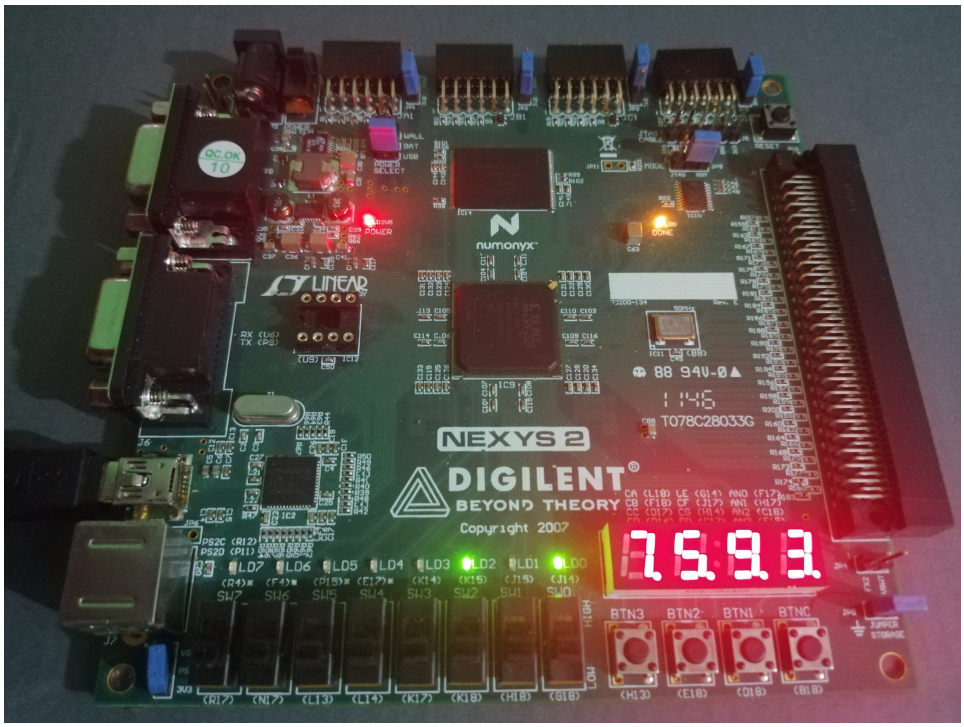

FIGURE 5.15 – Résultat du test obtenu sur la carte Nexys-2-FPGA.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	5,315	9,312	57%	
Number used as Flip Flops	5,296			
Number used as Latches	19			
Number of 4 input LUTs	8,208	9,312	88%	
Number of occupied Slices	4,584	4,656	98%	
Number of Slices containing only related logic	4,584	4,584	100%	
Number of Slices containing unrelated logic	0	4,584	0%	
Total Number of 4 input LUTs	8,267	9,312	88%	
Number used as logic	8,172			
Number used as a route-thru	59			
Number used as Shift registers	36			
Number of bonded IOBs	17	232	7%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	4.06			

FIGURE 5.16 – Rapport de synthèse.

Conclusion générale

Au cours de ce projet dédié à l'implémentation sur support reconfigurable d'un turbo décodeur SOVA, nous avons, étudié de façon détaillée le bloc codage canal de la chaine de communication sans fils. Nous avons particulièrement mis en avant les avantages et l'efficacité des turbo-codage/décodage dans la lutte contre les erreurs de transmission.

L'algorithme Sova (Viterbi à décision souple) retenu comme cœur des décodeurs Siso du turbo-décodeur a fait l'objet d'une description mathématique poussée dont l'analyse a permis d'aboutir à la conception d'une architecture traduisant le plus fidèlement possible le fonctionnement du décodage SOVA.

Les simulations faites sur le modèle HDL de ce décodeur associé à un procédé de décodage itératif, décrit également en HDL, ont montré de bonnes performances de reconstitution des signaux soumis à divers types d'erreurs.

L'implémentation du Turbo-décodeur à base de SOVA sur le kit Nexys2 de XILINX a confirmé les résultats de la simulation. Bien que ce turbo-décodeur à base de Sova présente des performances moindres que ceux basés sur l'algorithme MAP ou ses variantes, sa complexité réduite par rapport à ces derniers en fait un procédé très compétitif.

En guise de perspectives, nous suggérons d'explorer la possibilité d'accélérer ce turbo-décodeur en adoptant un procédé adaptatif consistant à la suspension de la mise à jour des fiabilités à la première intersection du chemin survivant. Il est, également, possible d'étendre cette approche de turbo-décodage à base de SOVA aux codes non binaires.

Bibliographie

[1] Gersho,A. *Vector Quantization and Compression.* Kluwer Academic Publishers Norwell : USA, 1991.

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

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

[4] Robertson,P. *Coded Modulation Scheme Employing Turbo Codes*, Electronics Letters, vol. 31, pp. 557-67, Sept 1995.

[5] Proakis,J. *Digital Communications (3e éd).*McGraw-Hill,1995.

[6] Kaiser,S et al. *Multi-carrier and spread spectrum systems : From OFDM and MCCDMA to LTE and WiMAX.* deuxième édition.Wiley, G Bretagne,2008.

[7] Gersho,A et al. *Vector Quantization and Compression.* Kluwer Academic Publishers Norwell, MA, USA, 1991.

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

[9] Le Ruyet,D. *Théorie de l'information : Codage Source-Canal.* Ecole Supérieure de Conception et de Production industrielles, France.Jan 2007.

[10] http ://wapiti.telecom-ille1.eu/commun/ens peda/options/ST/RIO/pub/exposes/exposesrio2006-ttnfa2007/Manga-Vrielynck/introduction.htm

[11] Ahson,Syed et al. *WiMAX standards and security* CRC press,September 2005.

[12] Elias,P. *Coding for Noisy Channels.* IRE Conv. Rec,1955, 4 :37-46.

[13] Wiley,John et al, *Channel coding for telecommunications.*1999.

[14] Berrou,C et al*Near Shannon Limit Error-Correcting Coding and Decoding : Turbo Codes.* IEEE Proceedings of the Int. Conf. on Communications, Geneva, Switzerland, May 1993,Vol. 2, pp. 1064-1070.

[15] Berrou,C et al*Near Optimum Correcting Coding and Decoding : Turbo Codes.* IEEE Transactions on Communications, October 1996, Vol. 44, pp.1261-1271.

[16] Dolinar,S et al. *Weight distributions for turbo codes using random and non random permutations* JPL TDA Progress Report 42-122, Aout. 1995.

[17] Ma,H et al. *On tail biting convolutional codes.* IEEE Transactions on Communications, vol. COM-34, pp. 104-111, Feb. 1986.

[18] Ping,Li. *Turbo-SPC Codes.* Department of Electronic Engineering :Brazil : City University of Hong Kong presented at IEEE Globecom'99.

[19] Huang,Fu-hua. *Evaluation of Soft Output Decoding for Turbo Codes.* These doctorat : Virginia :Polytechnic Institute and State University. May 29, 1997.

[20] Macchi,C et al. *Téléinformatique : transport et traitement de l'information dans les réseaux et systèmes téléinformatique.* Edition DUNOD E.N.S.T Paris 1983.

[21] Battue,D. *Télécommunications : Principes, Infrastructures et services.* Edition DUNOD 2ème édition 2001.

[22] Antonio,Fet al. *Floating Gate Analog Implementation of the Additive Soft-Input Soft-Output Decoding Algorithm* supported in part by Fulbright-CONACYT and Texas Instruments. Manuscript received July 5, 2001.

[23] Eritek,Inc. *VisSim/Comm^{TM} Turbo Codes Module User's Guide* Version 4.5,2001.

[24] Syed,Amjad. *Performance analysis of turbo codes over AXGN and reyleigh channels using different interleavers.* Thesis submitted in partial.

[25] Berrou,C *Codes et Turbocodes* Springer-Verlag France 2007.

[26] Makdour,M .*Application du Turbo-Code au codage de canal.*56p. Mémoire de Magister :Électronique : Alger :Ecole Nationale Polytechnique :2008.