RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Département d'Électronique

Laboratoire des Dispositifs de Communication et de Conversion Photovoltaïque

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'INGÉNIEUR d'ÉTAT en Électronique

IMPLEMÉNTATION SUR CIRCUIT RECONFIGURABLE D'UN TURBO-DECODEUR SOVA POUR LES COMMUNICATIONS NUMÉRIQUES

Thouria MEGHNOUDJ & Hamza TADRIST

Sous la direction de

Mr Mohamed Oussaid TAGHI

Présenté et soutenue publiquement le (21/06/2018) devant le jury composé de :

Président	Mr Boualam BOUSSEKSOU (ENP)
Examinateur	Mr Llies SAADAOUI (ENP)
Promoteur	Mr TAGHI Mohamed Oussaid (ENP)

ENP 2018

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Département d'Électronique

Laboratoire des Dispositifs de Communication et de Conversion Photovoltaïque

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'INGÉNIEUR d'ÉTAT en Électronique

IMPLEMÉNTATION SUR CIRCUIT RECONFIGURABLE D'UN TURBO-DECODEUR SOVA POUR LES COMMUNICATIONS NUMÉRIQUES

Thouria MEGHNOUDJ & Hamza TADRIST

Sous la direction de

Mr Mohamed Oussaid TAGHI

Présenté et soutenue publiquement le (21/06/2018) devant le jury composé de :

Président	Mr Boualam BOUSSEKSOU (ENP)
Examinateur	Mr Lleis SAADAOUI (ENP)
Promoteur	Mr Mohamed Oussaid TAGHI (ENP)

ENP 2018

Dédicaces

Du profond de mon cœur, je dédie ce modeste travail aux personnes qui sont chers à mon cœur

A mes chers parents

Que nulle dédicace ne puisse exprimer mon amour éternel et infini, respect et considération pour leurs patiences et sacrifice.

A ma chère famille

Sans tous un chacun je ne serais à cette place aujourd'hui, pour votre soutien et amour.

A mes amis

À toutes les personnes qui, par leurs actions, gestes, paroles ou écoutes nous ont soutenus tout au long de cette thèse.

Nous citerons tout particulièrement : Younes, Sofiane, Oussama, Hind, Salim, Mehdi, Mohamed, Nawele, Hamza.

Thouria MEGHNOUDJ

Dédicace

Â *mes parents.*

Â *ma famille.*

Â *mes amis.*

Â *l'ensemble de mes enseignants depuis première années primaire.*

Hamza Tadrist

Remerciements

Nous tenons à remercier vivement notre directeur de mémoire Mr TAGHI Mohamed Oussaid, professeur à l'École Nationale Polytechnique, pour nous avoir dirigé tout le long de ce travail, ainsi que pour toute l'aide qu'il nous a apporté.

Nous tenons également à remercier les membres du jury : Mr. BOUSSEKSOU et Mr. SAADAOUI, professeurs à l'École Nationale Polytechnique, pour l'intérêt qu'ils ont porté à nos travaux.

Enfin, nous tenons à présenter nos sincères remerciements à l'ensemble des enseignants du département d'Électronique de l'ENP qui ont fait de leur mieux pour nous instruire.

ملخص

في هذا العمل قمنا بدراسة إحدى أهم التقنيات الترميز في معالجة و تصحيح الأخطاء الناتجة عن التشويش في مجال الاتصالات اللاسلكية.

لإرسال المعلومات في مجال الاتصالات نحتاج إلى مرمز و إلى مفك الترميز. و لهذا الغرض قمنا باستخدام منهاج "توربو" الذي يتمحور على تقنيتين أساسيتين هما: التقنية الأولى MAP (خوارزمية الحد الأقصى) التي تتميز بالدقة و لكنها معقدة. أما التقنية الثانية فهي SOVA التي تم اقتراحها من طرف Hagenauer التي تمتاز باستخدامها الواسع في مجال الإتصال و ذلك راجع لتعقيدها المنخفض. وقد قمنا ببرمجتها باستخدام MATLAB لتشفير و فك التشفير وبعدها قمنا باقتراح دارة خاصة لها و ذلك باستخدام لغة vhdl.

الكلمات الرئيسية : مرمز , تصحيح الأخطاء , الاتصالات اللاسلكية , مفك الترميز ،MATLAB،MAP،SOVA،VHDL

Abstract

In the present work, we studied one of the most important techniques in the treatment of errors in the field of wireless communications. In order to send information in the field of communications, a coder and a decoder are necessary. Therefore, our studies focus on the Turbo coding which is based on two techniques : MAP and SOVA. The former is known for its precision and complexity. The latter, proposed by Hagenauer, is well known for its reduced complexity.

We propose an implementation of a material architecture of the turbo-code system by use of the VHDL language along with simulations of the encoder and decoder.

Key Words : treatment of errors, MAP, SOVA, coder, decoder, VHDL, trubocode.

Résumé

Dans ce travail, nous avons étudié l'une des techniques les plus importantes dans le traitement et la correction des erreurs pour les communications sans fils.

Pour envoyer des informations dans le domaine des communications nous avons besoin d’un encodeur et d’un décodeur et à cette fin nous avons utilisé le concept Turbo, qui se décline en deux techniques: MAP et SOVA. La première est caractérisée par sa précision et sa complexité. Tandis que la deuxième, proposée par Hagenauer, est très répandue dans le domaine de la communication à cause de sa complexité réduite.

Nous proposons l'implémentation sur FPGA d'une architecture matérielle du système « turbocodes » en utilisant le langage de description matérielle VHDL accompagnée des simulation de l'encodeur et du décodeur.

Mots clés : la correction des erreurs, communication sans fils, encodeur, décodeur, turbo, MAP, SOVA,VHDL

Table des matières

Liste des tableaux

Chapitre 3

Chapitre 4

Chapitre 5

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Liste des signes

C : Capacité d’un canal

CC : Classic Code (code classique)

d_{min} :La distance de Hamming minimale

Eb : L’énergie transmise dans un bit d’information.

E_c : L'énergie par symbole

M : Nombre de mémoire ou registre

N : La taille de l‘entrelaceur

$N0$: La densité spectrale du bruit.

PCCC :Parallèle Concaténâtes Convolutive code (Concaténation parallèle de codes convolutionels)

R : Le taux de codage

R_2 : Le rendements

P_w :La probabilités d'erreur par mot

P_b :La probabilités d'erreur par bit

ψ (t) :Signal de durée T

$\propto$: Entrelacements

σ^2 **:** Variance

n' :Est un bruit blanc gaussien

B : Band passant

H : L’entropie

I : L’information mutuelle

F_c : La fréquence de porteuse

Liste des abréviations

APP : Probabilité de justes a posteriori

ARQ : Demande de répétition automatique (Automatic Repeat Request).

BER : Taux d'erreurs sur les bits (Bit error ratio).

BPSK: Modulation par changement de phase Bipolaire (Bipolar phase-shift keying*).*

FER : Correction d'erreur directe (Forward Error Correction).

FPGA : Réseaux logiques programmables (Field Programmable Gate Arrays) .

LLR : Log-Likelihood Ratio

LRV: logarithme rapport de vraisemblance

MAP : Algorithme de Maximum a Posteriori.

NRZ : None Return a Zero

NSC : Non systématique convolutive code.

ROM: Read only Memory

RAM : Random Acess Memory

RSC : Récursive systématique convolutive code.

PDF: Fonction de densité de probabilité.

QPSK: Modulation par changement de phase Quadripolaire (Quadrupole phase-shift keying*).*

SRAM :static RAM

SISO: Entrée Souple Sortie Souple (Soft Input Soft Output).

SNR :Rapport Signale sur bruit .

SOVA : Sortie souple de l'algorithme de viterbi (Soft Output Viterbi Algorithem).

VHDL : Langage de description de matériel (Very high Hardware description language).

Introduction général

Dans les systèmes de communication numérique, on cherche à transmettre l'information provenant d'une source vers un récepteur à travers un canal de transmission. Le terme numérique est relatif à la séquence des symboles qu'il utilise pour représenter l'information. L'alphabet utilisé est binaire. Ce genre de transmission des données est très intéressant dans le sens où il permet l'utilisation de nombreuses techniques de manipulation de l'information [1]. Parmi celles-ci, nous pouvons penser à la compression de l'information, au cryptage de cette dernière et aussi aux codes correcteurs d'erreurs. Ce sont ces derniers qui font l'objet de beaucoup de recherches actuellement. Les perturbations intervenant sur le canal de transmission induisent des erreurs de transmission que le codage de canal s'efforce de combattre. L'objectif est alors d'assurer un taux d'erreur minimal. Le codage de canal est basé sur l'insertion parmi les éléments d'information d'éléments supplémentaires (la redondance) qui suivent une loi connue.

La théorie de l'information fondée par Claude E. Shannon en 1918, fournit un ensemble de développements théoriques qui permettent l'évaluation de la capacité des canaux de communication [3]. Celle-ci assure qu'on peut trouver une probabilité d'erreur aussi petite que l'on veut, si l'on emploie un code correcteur d'erreur approprié et si le taux de transmission du code est inférieur à la capacité du canal. En 1993, Berrou, Glavieux et Thitimajshima inventent les codes turbo [2] qui approchent les limites données par la théorie de l'information.

Organisation du mémoire :

L'objectif de notre travail consiste à étudier le turbo décodeur à base de l'algorithme de Viterbi à sortie Douce (SOVA) et à l'implémentation de son architecture sur Circuit reconfigurable (FPGA).

Ce mémoire est organisé en cinq chapitres. Dans le premier chapitre, nous commençons un bref rappel théorique concernant la chaine de transmission numérique. Nous enchainons, ensuite, avec des définitions sur le codage de canal et les codes correcteurs, puis nous les deux principaux procédés de codage canal, les codes en bloc et les codes convolutionnels.

Le second chapitre traite des turbo-codeurs qui résultent de la concaténation de deux codeurs convolutionnels identiques de type RSC associés à un dispositif d'entrelacement servant à permuter la séquence envoyée au second codeur. Nous avons, notamment, présenté les turbo codeurs issus de la concaténation série et celle parallèle. Le chapitre se termine par un exemple de codage turbo implémenté dans la suite de ce mémoire.

Au troisième chapitre, nous présentons les différents algorithmes de décodage des codes convolutionnels, Map, LogMap, et SOVA, utilisés pour le turbo décodage en association avec le principe du décodage itératif (chapitre 4).

Le quatrième chapitre décrit en détails le principe du décodage itératif, à base de SOVA, utilisé pour le turbo décodage.

Dans le cinquième chapitre nous décrivons l'implémentation du turbo-codeur et du turbo-décodeur étudiés sur Fpga, où nous commençons par présenter brièvement notre support d'implémentation, la carte FPGA (Nexys 2), puis nous exposons les simulations faites sous MATLAB et modelsim du turbo-codeur et du turbo-décodeur étudiés dans les chapitres précédents, nous donnons en fin de chapitre le résultat de l'implémentation sous Xilinx ISE, des différentes blocs du décodeur turbo ainsi que le décodeur turbo global à base de Sova.

Le présent mémoire est couronné par une conclusion générale.

Chapitre 1 : Codage du canal

1 .1. Introduction

La transmission numérique peut être affectée d'erreur liée aux imperfections du canal de transmission. Le codage canal vise à réduire l'effet des erreurs sur la qualité de la transmission.

1.2. Chaîne de transmission numérique

Le schéma de principe d'une chaîne de transmission numérique est représenté sur la figure 1.1, sans pour l'instant aucune justification de ses différents éléments. On peut distinguer : la source de message, le milieu de transmission et le destinataire qui sont des données du problème. Le codage et le décodage de source, le codage et le décodage de canal, l'émetteur et le récepteur représentent les degrés de liberté du concepteur pour réaliser le système de transmission. Nous allons maintenant décrire de façon succincte les différents éléments qui constituent une chaîne de transmission, en partant de la source de message vers le destinataire.

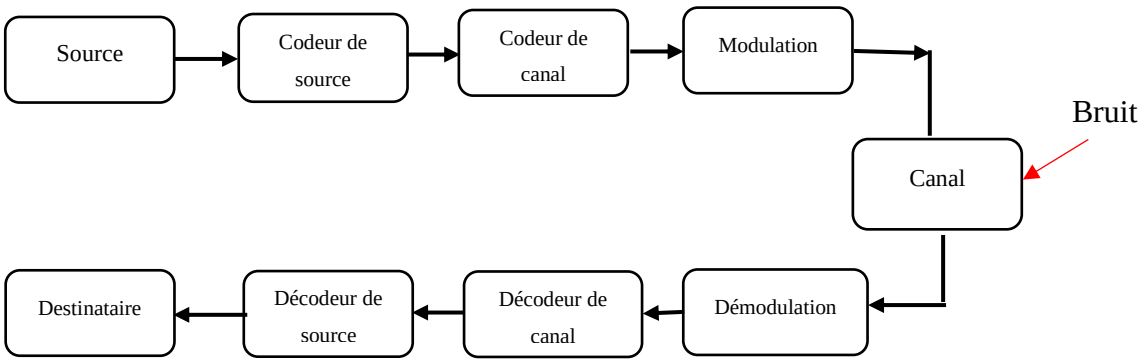

Figure 1. 1 : chaine de transmission numérique

1.2.1. Le canal

Le canal, au sens des communications numériques, et comme représenté à la Figure1.1, inclut le milieu de transmission (lien physique entre l'émetteur et le récepteur : câble, fibre, espace libre,…), le bruit (perturbation aléatoire issue du milieu, des équipements électroniques), et les interférences (provenant des autres utilisateurs du milieu de transmission, de brouilleurs intentionnels ou non). Par la suite, aussi bien dans la présentation des résultats de la théorie de l'information que dans cette introduction au codage de canal,

nous utiliserons un modèle de canal plus global, incluant une partie : de l'émetteur et du récepteur.

1.2.2. La capacité d'un canal

On définit la capacité d'un canal comme le maximum de l'information mutuelle moyenne I(X ; Y), elle représente donc la plus grande quantité d'information pouvant transiter entre l'émetteur et le récepteur [1], comme le montre la figure 1.2.

$$\boldsymbol{C = \max_{P(x)} I(X; Y)} \quad (1.1$$

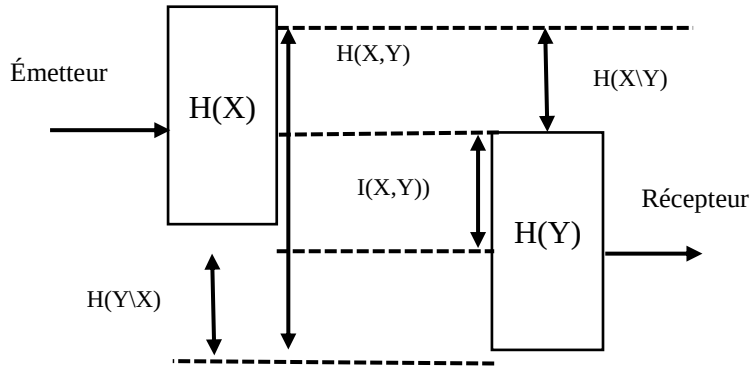

Figure 1. 2 : schéma représentatif de l'information mutuelle

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

$$\boldsymbol{I(X ; Y) = H(X) - H(X\backslash Y) = = H(Y) - H(Y\backslash X)} \quad (1.2$$

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, $H(X\backslash Y)$ représente l'information requise pour supprimer l'ambiguïté sur l'entrée.

$$\boldsymbol{H(x) = -\sum_i^N P_i \log_2 P_i} \quad (1.3$$

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 [2]. Pour la distinguer de la capacité par symbole, on trouve généralement dans la littérature la notation suivante :

$$\boldsymbol{C_s = C.d_s} \quad (1.4$$

Où ds, représente la probabilité d'apparition des lettres de l'alphabet de la source.

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

$$\mathbf{C_{AWGN}=1/2log_2}\left(\mathbf{1}+\frac{\boldsymbol{P}}{\boldsymbol{N}}\right) \qquad (1.5$$

Où $N = N_0B$ est la puissance du bruit. L'allure de la capacité C en fonction du rapport signal sur bruit *SNR* est illustrée dans la Figure 1.3.

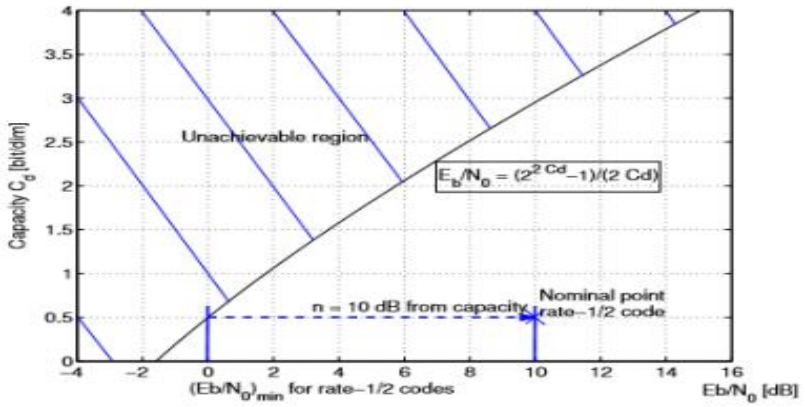

Figure 1. 3 : variation de la capacité du canal AWGN en fonction du SNR

1.2.3. Codeur de source

Le codage de source vise à la concision maximale du message, afin de minimiser les ressources nécessaires à la transmission (temps, puissance, bande passante, surface de stockage, etc.). Ce codage peut donc, pour diminuer le coût de la transmission, substituer un message aussi court que possible au message émis par la source, dans la mesure où cette substitution est réversible (i.e. que le message initial peut être exactement restitué). Le codage de source ne sera pas étudié. Indiquons simplement que ses limites théoriques sont fixées par la théorie de l'information (Premier théorème de Shannon).

1.2.4. Codeur de canal

Le codage de canal vise quant à lui à la protection du message contre les perturbations du canal de transmission. Si les perturbations engendrées induisent une qualité de restitution (souvent mesurée quantitativement par la probabilité d'erreur par bit sortant du codeur de source ou par message, trame, etc.) incompatible avec les spécifications fixées, le codage de canal se propose de transformer le message de manière à en augmenter la sûreté de transmission. Le « prix » qu'il en coûte est alors un accroissement de la taille du message.

Il y a donc antagonisme entre codage de source et codage de canal, l'objectif du premier étant de diminuer la redondance du message de source, celui du deuxième d'en ajouter dans un but de protection. Nous verrons que la théorie de l'information fixe les conditions pour lesquelles cet antagonisme n'est qu'apparent, et la division des tâches entre codeur de source et codeur de canal, pour l'instant arbitraire, justifiée.

1.2.5 Modulation

Le modulateur peut être considère comme un convertisseur 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 a une fréquence comparable à celle des symboles numériques représentés .Une représentation appropriée est le format NRZ (Non Return to Zero), qui représente les bits par des niveaux +V ou –V.

Même s'il est possible de transmettre ce signal, il est généralement translaté à une fréquence plus élevée. La raison à cela est la possibilité d'utiliser différentes régions du spectre pour différentes transmissions, ainsi que le fait que les hautes fréquences ont des longueurs d'ondes plus petites et nécessitent donc de plus petites antennes.

Un schéma de modulation très utilisé est le BPSK (Binary Phase Shift Keying).Dans ce schéma ,on multiplie le signal NRZ en bande de base par une porteuse dont la fréquence est un multiple le signal NRZ en bande de base par une porteuse dont la fréquence est un multiple du débit de bits. Il en résulte que le signal transmis durant l'intervalle de temps correspondant à un bit est soit le porteuse elle-même, soit son inverse.

Il est aussi possible d'utiliser une seconde porteuse déphasé de $\frac{\pi}{2}$ de la premier, la moduler et ajouter à la première. En d'autres termes, si le signal BPSK est $\pm$ cos ($2\pi f_c$t)

(f_c étant la fréquence de la porteuse et t le temps) le second signal est $\pm$ sin ($2\pi f_c$t) et

$s(t) = \sqrt{2}\cos(2\pi f_c t + i\frac{\pi}{4})$; i=-3,--1,+1,+3

Ceci est connu sous le om QPSK (QuadriPhase Shift Keying),est possède l'avantage par rapport à la modulation BPSK de paramètre du transmission de deux fois plus de bits durant le même temps et en utilisant la même bande passante.

Un diagramme de phase est décrit dans la figure .la distribution des valeurs des bits sur les phases implique que chaque porteuse est indépendamment modulée en alternant les bits du flux de données codés.

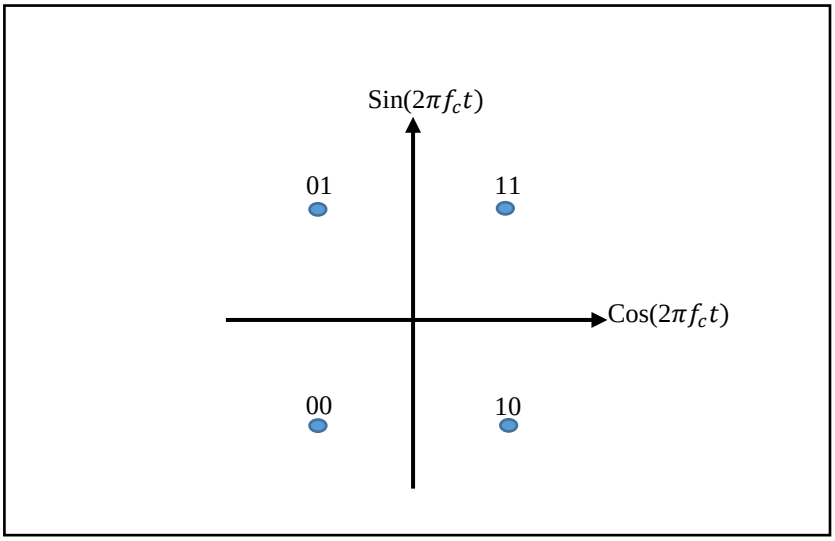

Figure 1. 4: Diagramme de la modulation QPSK utilisant le code de Gray.

1.2.6. Le décodeur de canal

Plusieurs stratégies différentes peuvent être utilisées par le décodeur de canal. La première est la détection d'erreurs. Le décodeur observe la séquence reçue (ferme ou souple) et détecte la présence éventuelle d'erreur. Cette détection peut servir à contrôler le taux d'erreur (Error Monitoring) ou à mettre en œuvre des techniques de retransmission (ARQ : Automatic Repeat Request) : le décodeur demande à l'émetteur de retransmettre la séquence dans laquelle une erreur a été détectée. Il est évident que ce type de procédé nécessite une voie de retour. Cette stratégie de

détection est surtout utilisée par les couches transport et supérieures du modèle OSI. Les techniques employées ne seront que mentionnées, dans la mesure où il ne s'agit qu'un cas particulier et simple de la seconde stratégie.

La deuxième est la correction d'erreurs (FEC : For Ward Error Correction). Elle nécessite des algorithmes beaucoup plus complexes que la simple détection, et plus de redondance dans la séquence émise. Toutefois, le milieu de transmission est utilisé de manière plus efficace.

1.3 Codage de canal et Codes correcteur d'erreur

Même si la théorie de l'information nous dit qu'il est possible de trouver un code de canal qui nous donne une probabilité d'erreur aussi petite que l'on veut pour un canal donné, il n'en demeure pas moins que d'un point de vue pratique cela ne soit pas si aisé.

Il est généralement difficile de concevoir un bon code de canal. Dans cette section nous parlerons des paramètres utilisés pour décrire un code de canal puis nous aborderons le sujet essentiel nous concernant : les codes convolutifs (ou convolutionnels).

1.4 Les classes de code correcteurs d'erreurs

Pour approcher au mieux la capacité C du canal si, conformément au théorème de Shannon, l'entropie de la source H (X) < C nous pouvons ajouter de la redondance dans le codage de la source afin de diminuer les erreurs de transmission. C'est le problème du codage de canal. A côté des premiers codes empiriques (bit de parité, répétition,…) deux grandes catégories de codes ont été développées et sont actuellement utilisées en faisant l'objet permanent de perfectionnements :

1. Les codes en blocs linéaires.
2. Les codes de convolution.

On dite que un code est linéaire si la fonction de codage est une application linéaire, sinon il est dit non-linéaire.

Lorsque les traitements sont effectués pour obtenir les propriétés de détection ou de correction ont lieu par bloc de N symboles, on dit qu'on a affaire à un code en bloc.

Lorsque les symboles créés par la source ne sont pas traités par des blocs, mais de manière continue, on dit qu'on a affaire à un code convolutif.

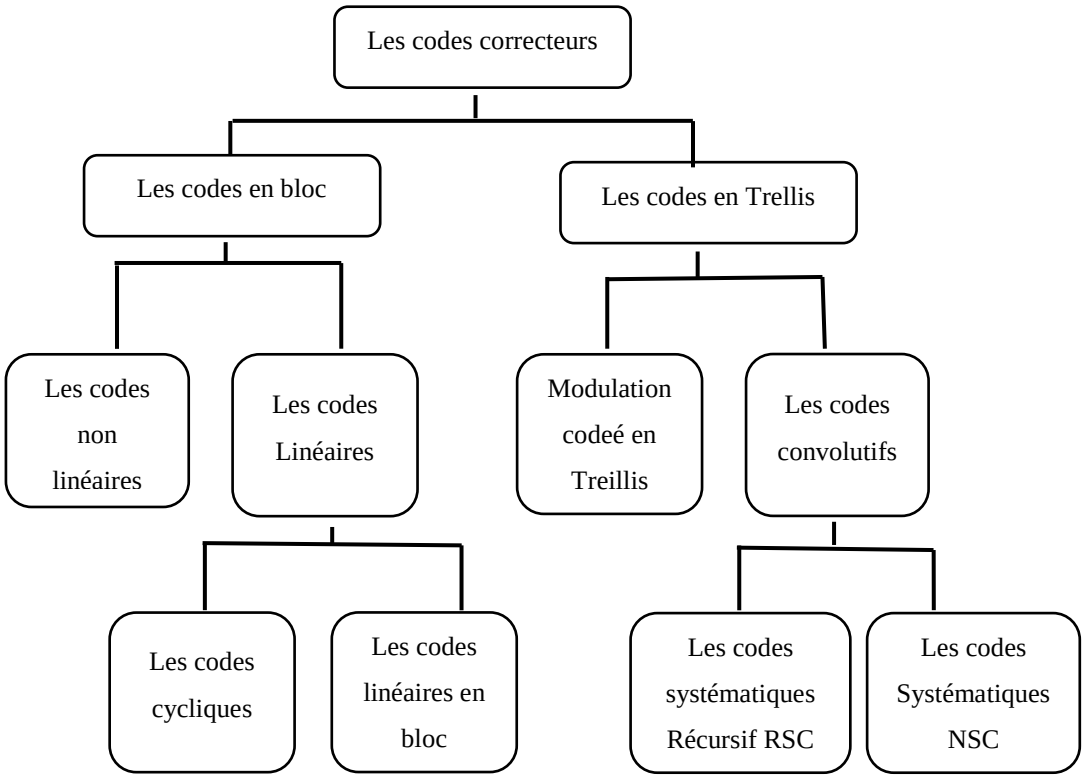

Figure 1. 5 : Classification codes correcteurs d'erreurs

1.5 Définitions et propriétés des codes de blocs linéaires

1.5.1 Code binaire

Le codage de canal utilise les symboles d'une source pour construire des objets appelés mots de codes. Pour une source discrète binaire nous utiliserons deux symboles. Pour un alphabet non binaire de q symboles (source q-ary) si $q = 2^b$, un code de longueur N peut se ramener à l'étude d'un code binaire de longueur $b.N$.

1.5.2 Codes en blocs

L'information de la source est mise en trames de longueur fixe que nous devons transmettre c'est le message. Le codage de canal prend ce message pour en faire un mot de code :

$$\text{message} \rightarrow \text{codage de canal} \rightarrow \text{mot de code}$$

Le message est constitué de k caractères soit 2^k messages possibles. Le mot de code utilisé sera lui aussi de longueur fixe de n caractères soit 2^n mots de code possibles. Avec $n > k$ il y aura donc $n - k$ caractères du mot de code qui sont redondants et serviront à traiter les erreurs éventuelles. Par ailleurs, $2^n > 2^k$ donc un certain nombre de mots de code ne correspondent pas à un message mais seulement à des erreurs de transmission. On parle ainsi d'un code de bloc (n, k) et le rendement du code défini par :

$$R_c = k/n$$

Un cas particulier des codes de blocs est celui des codes où le message apparaît explicitement sur ses k caractères c'est à dire "en clair". A côté de ces k caractères seront donc ajoutés $n - k$ caractères redondants. Nous obtenons alors un code dit code systématique et c'est ce sous ensemble que nous étudierons.

1.5.3 Poids d'un code

Le poids d'un mot de code est par définition le nombre de caractères non nuls que contient ce mot.

Exemple : 11001010 est de poids 4 10011000 est de poids 3 00000000 est de poids nul

A l'ensemble des mots d'un code on associe l'ensemble des poids qui est la distribution de poids du code. Un code est dit de poids fixe (ou poids constant) si tous les mots du code ont le même poids.

1.5.4 Distance de Hamming

La distance de Hamming d_{ij} entre deux mots de code est le nombre de bits dont ils diffèrent.

Exemple : 11001000 et 10011010 ont une distance de 3.

La distance minimale du code est le minimum de l'ensemble des distances entre codes :

$$d_{min} = \min \{d_y\} \qquad (1.6$$

1.6 Codes convolutifs

1.6.1 Introduction

Le principe des codes convolutifs, inventés par « Peter Elias » en 1954, est de considérer des séquences semi-infinies a_0, a_1, a_2….de symboles qui passent à travers une succession de registres à décalage dont le nombre d'étages m est appelé mémoire du code et 2^m le nombre d'états possibles.

Les codes convolutifs constituent une classe extrêmement flexible et efficace de codes correcteurs d'erreurs, Ce sont les codes les plus utilisés dans les communications fixes et mobiles.

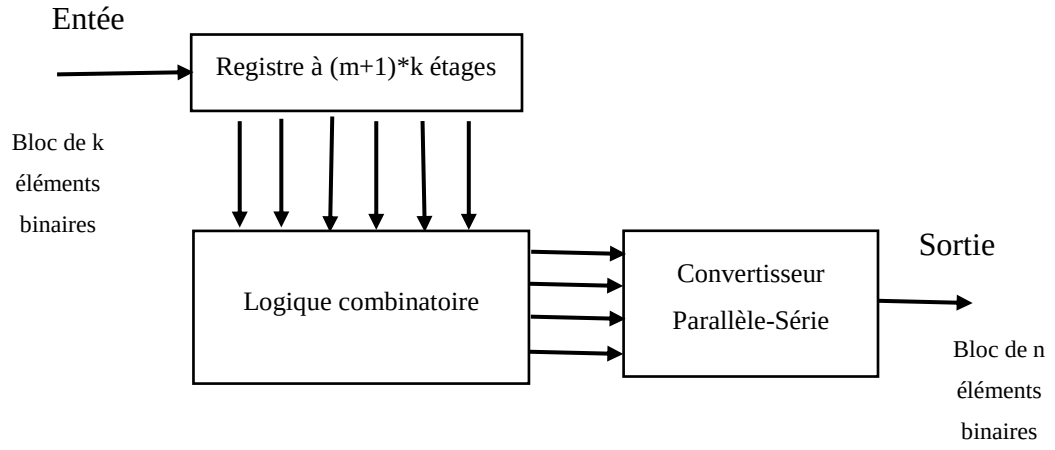

Figure 1. 6 : Schéma de principe d'un codeur convolutif de rendement R et de mémoire m.

1.6.2 Propriétés

Les propriétés des codes convolutifs sont :

- ✓ Le rendement du code est R= $\frac{K}{n}$
- ✓ La longueur de contrainte du code est : (m+1)*k.
- ✓ Linéarité : les mots de code associés à une combinaison linéaire de séquences d'entrée correspondent à la combinaison linéaire des mots de code de chacune des ces séquences.
- ✓ Stationnarité : Lorsqu'un message source, décalé dans le temps, est envoyé sur l'encodeur, on doit retrouver à la sortie, le mot de code correspondant décalé de la même manière dans le temps.
- ✓ Code convolutif systématique :

Mot code : $C = (X_1\ Y_1\ X_2\ Y_2 \ldots X_j\ Y_j \ldots)$

Avec $X_j = (X_j^1 X_j^2 \ldots X_j^k)$ Information

$Y_j = (Y_j^1 Y_j^2 \ldots Y_j^k)$ Contrôle

1.6.3 Principe du codage convolutif

- Les codes convolutifs forment une classe extrêmement souple et efficace de codes correcteurs d'erreur.
- Ce sont les codes les plus utilisés dans les communications fixes et mobiles.
- Les codes convolutifs ont les mêmes caractéristiques que les codes en bloc sauf qu'ils s'appliquent à des séquences infinies de symboles d'information et génèrent des séquences infinies de symboles de code.[5,6]

Pour illustrer bien le principe des codes convolutifs, voici un exemple présenté par la Figure « Exemple d'un code convolutif de rendement *R* = 1/2. », pour *K* = 1, *m* = 2 et *N* = 2. A t parvient au codeur à l'instant t, les bits de sortie X et Y sont calculés par les relations suivantes:

$$X = a_t + a_{t-1} + a_{t-2} \quad (1.7$$

$$Y = a_t + a_{t-2} \quad (1.8$$

Supposons que le codeur reçoive le message 1011, les registres étant initialement

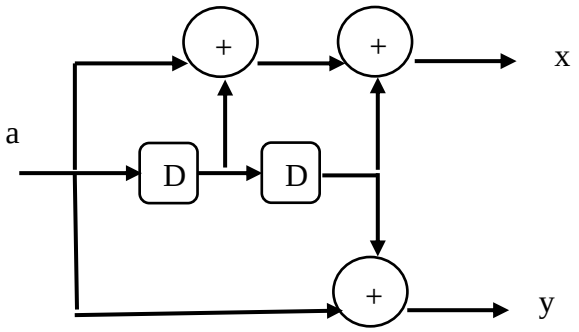

Figure 1. 7: Exemple d'un code convolutif de rendement *R* = 1/2

1.6.4 Représentations des codes convolutifs

Plusieurs représentations pourraient être trouvées pour représenter des codeurs de convolution. Les plus importants sont :

- Représentations numériques :
  - Transformée en D
  - Matrice de transfert
- Représentations graphiques :
  - Diagramme d'état
  - Arbre
  - Treillis

A. Représentations numériques :

i. Transformé en D

Une séquence de symboles est représentée par une série formelle en la variable D. Cette variable représente l'opérateur de retard unitaire :

$$\begin{cases} s(D) = s_0 + s_1.D + s_2.D^2 + ... + s_j.D^j + ... \\ x^i(D) = x_0^i + x_1^i.D + x_2^i.D^2 + ... + x_j^i.D^j + ... \end{cases}$$

La réponse impulsionnelle du $i^{ème}$ module, $h^i(D)$, est la séquence de sortie produite lorsque le message d'entrée est une suite commençant par le symbole '1' et se terminant par une suite de '0' de longueur infinie :

$$x^i(D) = h^i(D).s(D) \qquad (1.9$$

Exemple : k=1, m=2, n=2, R=1/2

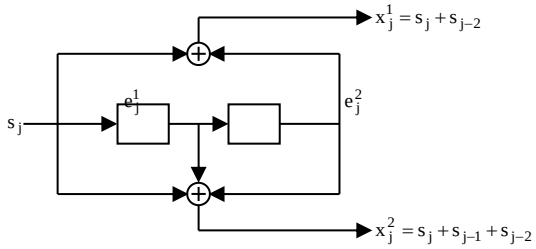

Figure 1. 8: Exemple d'un code convolutif de rendement $R = 1/2$

$e^1(D) = D.s(D)$

$e^2(D) = D.e^1(D) = D^2.s(D)$

$$\boxed{H(D) = \begin{pmatrix} 1+D^2 & 1+D+D^2 \end{pmatrix}}$$

$x^1(D) = s(D) + e^2(D) = (1 + D^2).s(D)$

$x^2(D) = s(D) + e^1(D) + e^2(D) = (1 + D + D^2).s(D)$

ii. Matrice de transfert

La matrice de transfert donne la relation entrée-sortie sous forme matricielle. On l'écrit pour chaque étage de sortie. Pour la $k^{ième}$ sortie :

La $i^{ème}$ ligne donne la relation entre x_j^k et s_j^i

La $(i + 1)^{ème}$ ligne donne la relation entre x_j^k et s_j^i

La $1^{ère}$ colonne correspond à l'instant j,

La $2^{ème}$ colonne correspond à l'instant $(j – 1)$ …

$$\begin{array}{c} \\ s^i \\ s^{i+1} \\ \dots \end{array} \begin{array}{c} \begin{array}{ccc} j & j-1 & \dots \end{array} \\ \begin{pmatrix} \dots.. & \dots.. & \dots.. \\ \dots.. & \dots.. & \dots.. \\ \dots.. & \dots.. & \dots.. \end{pmatrix} \end{array}$$

Exemple : k=2,m=3,n=3 et R=2/3

Matrices de transfert intermédiaires :

$$G_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$$

$$G_2 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

$$G_3 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

Matrices de transfert :

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

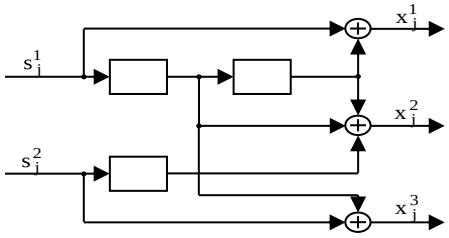

Figure 1. 9 : Exemple pour R=2/3

B. Représentations graphiques :

Chaque bloc de *n* éléments binaires en sortie du codeur dépend :

- Du bloc de *k* éléments binaires présents à son entrée ;
- Des *m* blocs de *k* éléments binaires contenus dans sa mémoire.

Ces *m.k* éléments binaires définissent l'état du codeur.

Les quatre états possibles du codeur sont : '00' ;'01' ; '10' et '11'

i. Digramme d'état

Seules deux (*q*) transitions sont possibles à partir de chacun des états.

Les étiquettes de chaque branche correspondent aux sorties du codeur.

Exemple :

La représentation graphiques du l'exemple 2

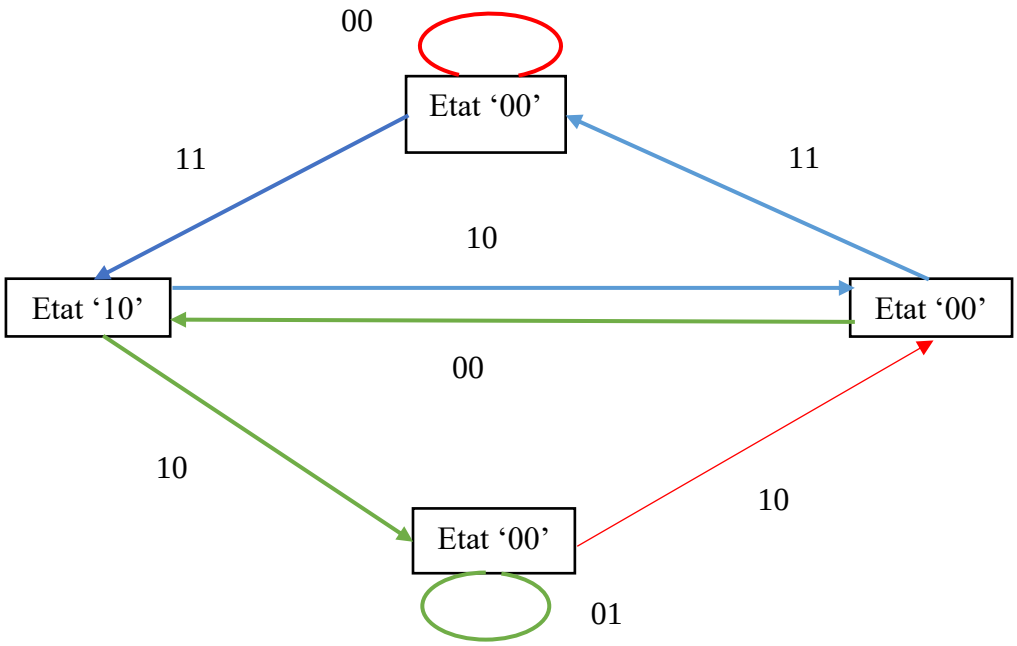

Figure 1. 10: La représentation graphique de digramme d'état

Remarque

Lorsque l'élément binaire d'entrée du codeur est égal à '0' (resp. '1'), le couple binaire en sortie du codeur est porté par la branche rouge (resp. verte).

ii. Arbre

C'est un développement du diagramme d'état en fonction du temps discrétisé.

Les conventions adoptées :

- Le temps s'écoule de la gauche vers la droite

- Lorsque l'élément binaire d'entrée du codeur est égal à '0' (resp. '1'), le couple binaire en sortie du codeur est porté par la branche supérieure (resp. inférieure).

Les branches se séparent en un point appelé nœud. Chaque nœud donne naissance à 2^k (q^k) branches.

Quelque soit l'état initial du codeur, après ($m + 1$) décalages à l'entrée du codeur, tous les états du codeur peuvent être atteints.

Exemple : si la séquence d'information est : 1001 le mot de code associé est 11011111

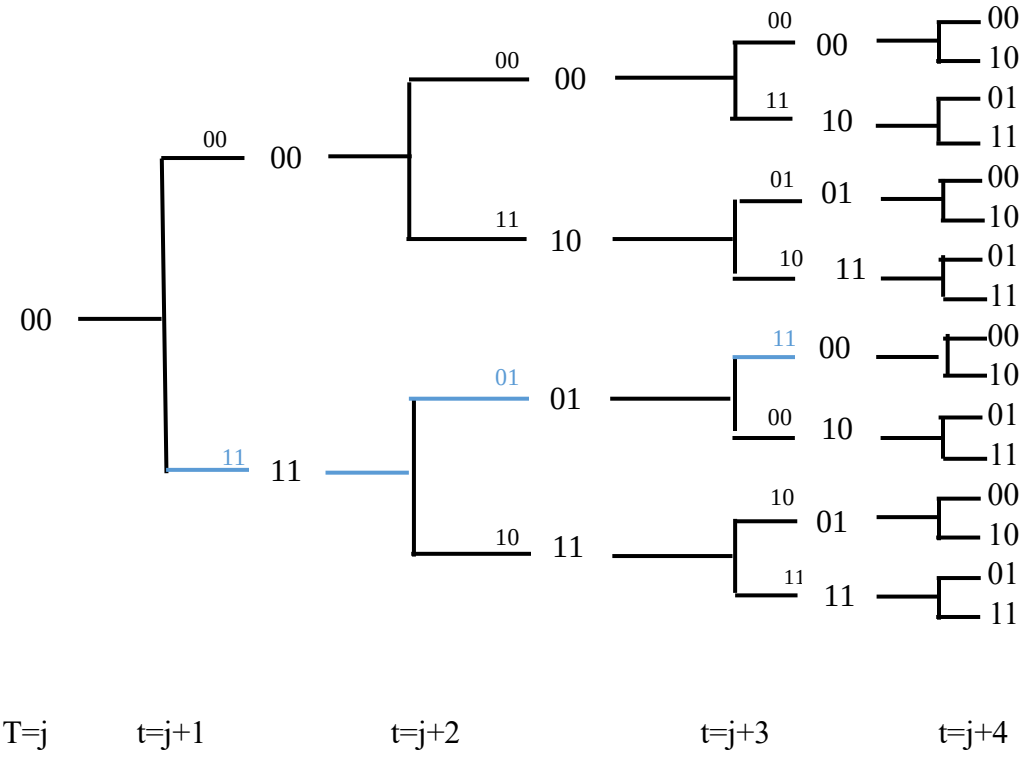

Figure 1. 11: La représentation graphique de l'arbre

iii. Treillis

- Les conventions adoptées :
  - Lorsque l'élément binaire d'entrée du codeur est égal à '0' (resp. '1'), le couple binaire en sortie du codeur est porté par la branche rouge (resp. verte).
- De chaque nœud partent 2^k (q^k) branches.
- En chaque nœud convergent 2^k (q^k) branches.

- Les étiquettes de chaque branche correspondent aux sorties du codeur.
  Exemple :
  La séquence d'information est 1001 le mot de code associé est 11011111

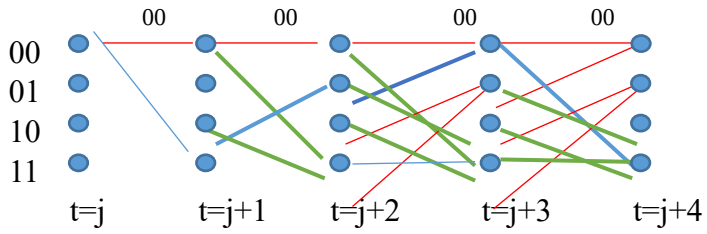

Figure 1. 12: La représentation graphique de treillis

1.6.5 Décodage convolutif

Dans les canaux de communication sans mémoire, les systèmes utilisant le codage convolutif sont parmi les plus intéressants tant du point de vue de leurs performances (s'approchant le plus des performances ultimes prévues par la théorie de Shannon) que du point de vue de leur réalisation et implantation matérielle.

Les deux principales techniques de décodage des codes convolutifs sont le décodage de Viterbi et le décodage séquentiel.

Chacune de ses techniques consiste à trouver un chemin particulier (le message transmis), dans un graphe orienté où on assigne aux branches des métriques ou valeurs de vraisemblance entre les données reçues et les données qui auraient pu être transmises.

L'objectif général du décodeur se résume donc à déterminer avec la plus grande fiabilité et le minimum d'efforts le chemin de métrique minimale. Ce chemin est la séquence décodée

1.7 Les codes NSC et RSC

On désigne par NSC les codes non-systématiques et par RSC les codes récursifs et systématiques. Un code convolutif est dit récursif si la séquence passant dans les registres à décalages est alimentée par le contenu de ses registres (Figure (b)).Si les K symboles

d'information à l'entrée du codeur se retrouvent explicitement dans le code, alors le code est dit systématique, sinon il est dit non-systématique [7] (Figure (a)). Les codes non-systématiques et non-récursifs présentent, pour des SNR élevés, des performances meilleures qu'un code systématique et non-récursive et l'inverse pour les SNR faibles [8], [9]. Pour cette raison, les codes NSC ont été principalement étudiés et utilisés jusqu'au début des années 1990. On rappelle que la puissance d'un code (capacité de correction d'erreurs) et la complexité du décodeur augmentent avec l'augmentation de la mémoire m de ce code. Quant aux codes récursifs systématiques (RSC), ils sont utilisés par les turbo-codes, car ils sont les seuls susceptibles d'atteindre la limite de Shannon

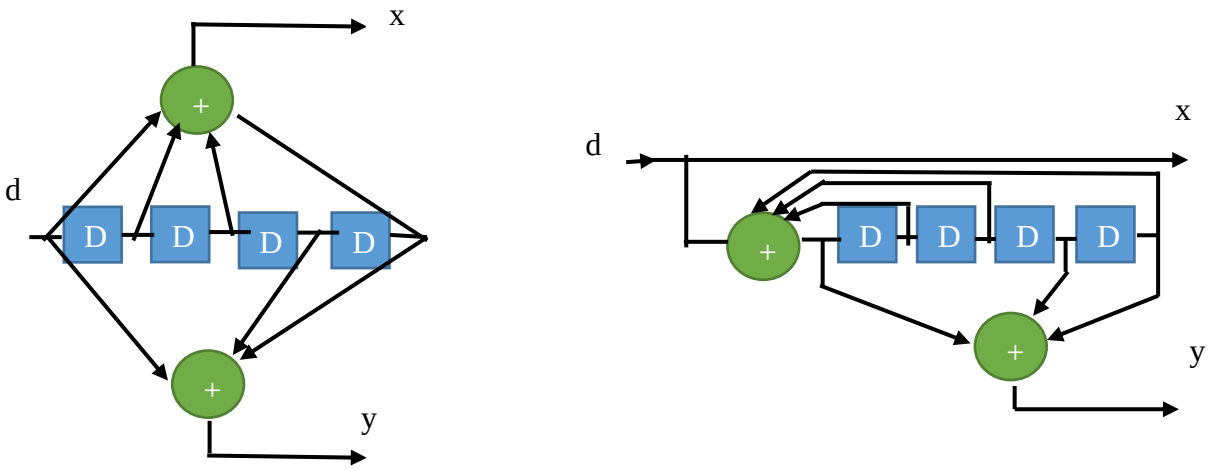

Figure 1. 13 : (a) Un code non-systématique (NSC) (b) Un code récursif systématique (RSC).

1.8 Conclusion

Dans ce chapitre, après avoir rappelé le rôle des différents éléments de la chaine de communication numérique, nous avons décrit les principaux procédés de codage canal, les codes en blocs, et les codes convolutifs. Ces derniers retenus par les concepteurs des turbocodes comme technique de codage côté émission, sont expliqués de façon plus détaillé. Les algorithmes de décodage des codes convolutionnels seront exposés dans le chapitre deuxième.

Chapitre 2 : Turbo code

2.1 Introduction

Les turbo codes ont été inventés en 1991, et présentés à la communauté scientifique en 1993, par une équipe de l'Ecole Nationale Supérieure des Télécommunications de Brest dirigée par Claude Berrou et Alain Glavieux. Les spécialistes des codes correcteurs d'erreurs ont tout d'abord accueilli cette invention avec beaucoup de scepticisme, du fait des extraordinaires performances annoncées. Cependant, d'autres équipes, dans le monde entier, sont parvenues peu après aux mêmes résultats, ce qui a contribué au développement des turbos codes. Ils ont été adoptés par toutes les agences spatiales mondiales, et sont utilisés dans la transmission des données du nouveau standard de téléphonie mobile qui va succéder au LTE, 3GPP…

2.2 Structure des Turbo Codes

Les concaténations des plus célèbres sont la concaténation parallèle et la concaténation série de deux codes convolutifs. Il est possible de fabriquer des turbo code hybrides en combinant les concaténations série et parallèle. [13]

2.2.1 Les Turbo Codes Parallèles

Un turbo code parallèle convolutif est construit par la concaténation parallèle de codes convolutifs séparés par des entrelacements $\propto_i$. Chaque code constituant traite une version entrelacée différente de la même séquence d'information.

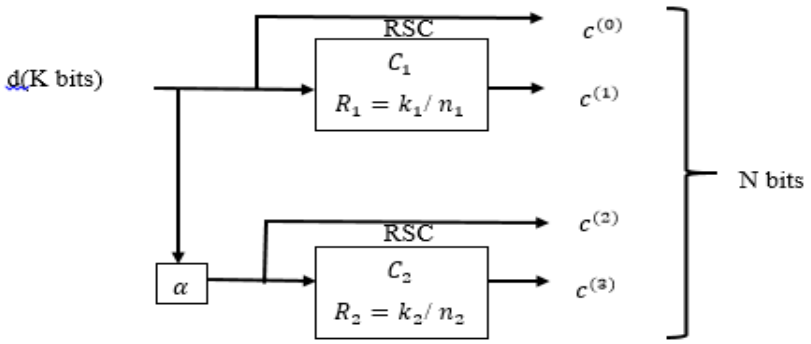

Figure 2. 1: Turbo Code Parallèle

La figure 2.1 représente un turbo code construit à partir de deux codes constituants C_1 et C_2. En pratique, on choisit les deux codes C_1 et C_2 similaires de type RSC. De plus, afin d'augmenter le rendement, il est possible de ne pas utiliser la systématique C [14].En négligent la fermeture de treillis, le rendement totale du turbo code parallèle est donné par les équitations suivant :

$$R_1 \neq R_2 \quad \rightarrow \quad R = \frac{K}{N} = \frac{R_1 . R_2}{R_1 + R_2 - R_1 . R_2} \qquad (2.1$$

$$R_1 = R_2 \quad \rightarrow \quad R = \frac{K}{N} = \frac{R_1}{2 - R_1} \qquad (2.2$$

K est le nombre de bit en entrée et N le nombre de bits codés en sortie. $R_1 = \frac{k_1}{n_1}$ et $R_2 = \frac{k_2}{n_2}$ sont les rendements respectifs de chaque code constituant C_1 et C_2.

En considérant la terminaison du treillis de chaque code constituant, n_1T_{t1} (T c'est la période de cycle) bits pour C_1 et n_2T_{t2} bits pour C_2 doivent être rajoutés, dans ce cas le rendement est :

$$R_1 \neq R_2 \rightarrow \quad R = \frac{K}{N + n_1T_{t1} + n_2T_{t2}} = \frac{R_1 . R_2}{R_1 + R_2 - R_1 . R_2 + \frac{n_1T_{t1} + n_2T_{t2}}{K}} \qquad (2.3$$

$$R_1 = R_2 \quad \rightarrow \quad R = \frac{K}{N + n_1T_{t1} + n_2T_{t2}} = \frac{R_1}{2 - R_1 + \frac{2\, n_1T_{t1}}{R_1\, K}} \qquad (2.4$$

2.2.2 Les Turbo Codes série

Un turbo code convolutif série est obtenu par la concaténation série de codes convolutifs séparés par des entrelacements ∝. Dans le cas d'un turbo code à deux niveaux (figure 2.2) le code en amont de l'entrelaceur est appelé code externe et celui en aval est le code interne. Pour un codage correct les deux codes constituants démarrent de l'état zéro $S_{(0,0)}$ et se terminent à l'état zéro$S_{(0,L)}$. En pratique, le code externe C_1 peut-être un code RSC ou NRNSC. Suant au code interne C_2 il doit toujours être RSC.

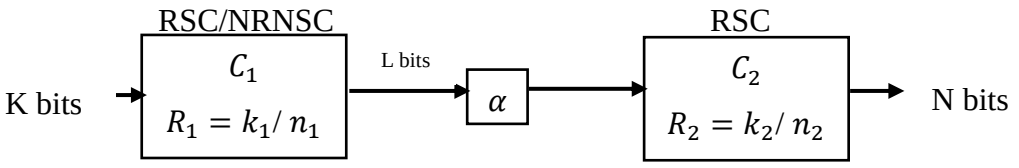

Figure 2. 2: Turbo Code série

En considérant que $R_1 = \frac{k_1}{n_1}$ et $R_2 = \frac{k_2}{n_2}$ sont les rendements respectifs du code externe C_1 et interne C_2, que K bits d'informations entrent dans le codeur et que N bits codés en sortent, le code C_1 génère $L = \frac{K}{R_1}$ bits et le code C_2 en génère $N = \frac{L}{R_2}$.

En négligeant la fermeture du treillis le rendement total du codeur série est :

$$R = \frac{K}{N} = R_1 . R_2 \qquad (2.5$$

T_{t1} cycles sont nécessaires pour terminer le treillis de C_1 et T_{t2} cycles pourC_2. Le nombre de bits additionnels en sortie est donc $n_1 T_{t1}$pour C_1 et $n_2 T_{t2}$pourC_2, dans ce cas le rendement est :

$$R = \frac{K}{J + n_1 T_{t1} + n_2 T_{t2}} = \frac{R_1 . R_2}{1 + (n_1 T_{t1} + n_2 T_{t2}) . \frac{(R_1 . R_2)}{K}} \qquad (2.6$$

Remarque :

Les turbo codes série peuvent atteindre des rendements plus élevés que les parallèles. Aussi, la distance minimale d'un turbo codeur série est plus élevée que celle du parallèle car la concaténation série permet d'exploiter les séquences d'information, qui présentent au mois la distance minimale $d_{min}(C_1)$, issues du premier codeur. Par la mise en œuvre est plus complexe du fait de la complexité de son treillis.

2.3 Performances d'un turbo code

Dans cette partie nous nous intéressons uniquement aux performances des turbo-codes parallèles. Alors que les codes traditionnel sont pour but réduire la distance de Hamming minimale d_{min} d'un code, I 'approche du turbo-codage est de 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 [15], 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 RSC 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 terme de distance (d_{min} faible), alors que l'aspect filtre à réponse impulsionnelle infinie d'un codeur RSC lui assure en général un grand poids de Hamming. La présence de I'entrelaceur permet de minimiser la probabilité que les deux codes constituants gérèrent en même temps un mot de code de poids faible. En effet, grâce à I'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 I'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 des entrelaceurs utilisés.

2.4 L'Entrelacement

Un entrelaceur α permet de modifier les indices (positions des bits en paquet d'information) des données par permutations, il représente un élément du groupe de permutation pour *N* éléments. Des groupes de permutation peuvent être décomposés en utilisant une seule méthode. Par exemple, α_{16} ={15,10,1,12,2,0,13,9,5,3,8,11,7,4,14,6}peut être écrit dans la décomposition de cycle de disjonction avec deux cycles :

α_{16} ={(0,15,6,13,4,2,1,10,8,5),(3,12,7,9) }

Ce qui signifie simplement que les symboles sont pris par ordre comme suit : 0 →15 →6 →13 →4→2 →1 →10 →8 →5 →0 →15…… dans un cycle de longueur 8, et

un deuxième cycle de longueur 4. Alternativement, l'entrelaceur peut être exprimé par sa fonction d'indice.

$$\alpha(i) = j, 0 \leq i, j < N \quad (2.7$$

Soit l'entrelaceur le plus commun, l'entrelaceur de bloc de taille $N = mn$. Ce type d'entrelaceur est basé sur l'écriture en ligne (entrée d'entrelaceur) et la lecture en colonne (sortie d'entrelaceur) d'une matrice d'information. L'entrelacement est accompli par la lecture de l'information par la colonne (sortie de l'entrelaceur). Cet entrelaceur a la fonction d'indice :

$$\alpha(i) = ni + \left\lfloor \frac{i}{m} \right\rfloor \bmod N, 0 \leq i, j < N \quad (2.8$$

La fonction $\left(\left\lfloor \frac{i}{m} \right\rfloor\right)$ est l'entrelaceur difficile à analyser, et un entrelaceur "linéarisé" est présenté dans la référence [2] avec la fonction d'indice :

$$\alpha(i) = ki + u \bmod N, 0 \leq i, j < N \quad (2.9$$

Avec k et u sont des nombres entiers fixes.

Les entrelaceurs en bloc peuvent réaliser de bonnes valeurs de distance libre, mais ils échouent à produire un spectre des distances très étroit. Les coefficients angulaires d'approximativement N semblent fonctionner mieux, ayant pour résultat la bonne dispersion des points indexés [25].

On dispose aussi pour améliorer la performance du turbo code dans l'intervalle entre les valeurs moyennes et grandes du rapport signal sur bruit des entrelaceurs semi aléatoires [26], ce qui évite explicitement d'étaler des indices proches à l'entrée aux indices proches à la sortie. Ils sont produits de façon analogue aux entrelaceurs aléatoires, choisissant chaque incrément $p(i)$ aléatoirement et testant si $|\alpha(i) - \alpha(l)|^3 \geq T$ pour tout $i - S < l < i$, c'est-à-dire en contrôlant tous les choix précédents. Si un choix d'indice ne remplit pas cette condition, il est rejeté et un nouvel indice est choisi. Le processus continue jusqu'à ce que tous les N indices aient été choisis. Évidemment, le temps de recherche pour un entrelaceur approprié augmente avec S et T, et l'algorithme peut même ne pas se terminer avec succès. Cependant, selon la référence [26], des valeurs de S, $T < \sqrt{N/2}$ finissent l'indexage dans un temps raisonnable.

Pour les entrelaceurs déterministes qui ont les propriétés statistiques des entrelaceurs

aléatoires, Takeshita et Costello [25] ont conçu ce qu'ils ont appelé les entrelaceurs quadratiques, qui sont définis par la fonction quadratique d'indice suivante :

$$\alpha(i) = \frac{ki(i+1)}{2} \; mod \; N \qquad (2.\ 10$$

Avec k est une constante impaire [24].

2.5 Exemple de turbo codeur

Soit le Turbo codeur dont la structure est donnée dans la Figure 2.7.

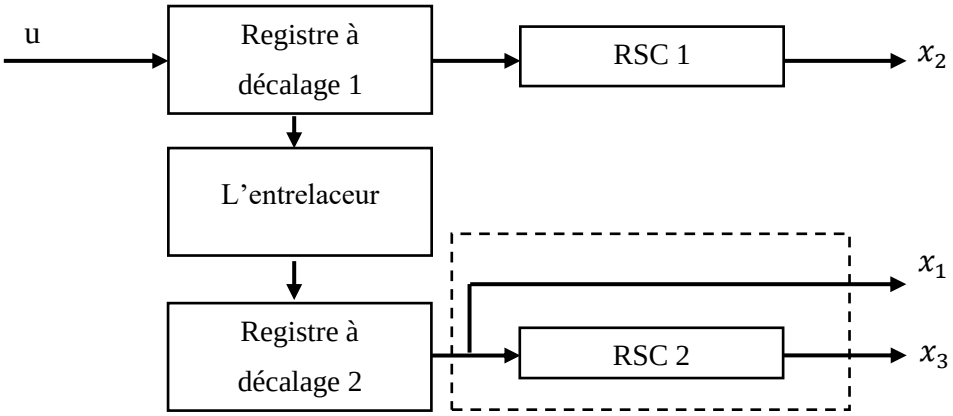

Figure 2. 3: Structure d'encodeur de code turbo

Le bit systématique est associé au codeur RSC 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 et 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.

La figure 2.8 montre le codeur RSC utilisé dans l'exemple.

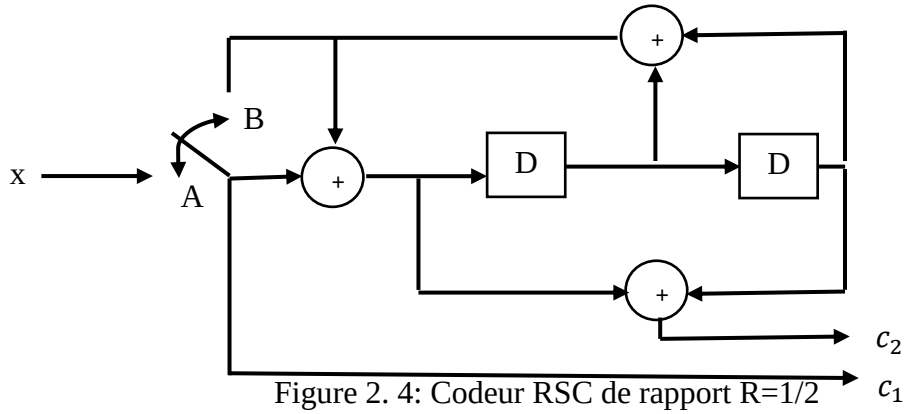

Figure 2. 4: Codeur RSC de rapport R=1/2

Le commutateur est mis en position A pour coder la séquence d'entrée et est activé en position B pour terminer le treillis. La figure 2.5 montre le diagramme d'état du codeur de composant RSC.

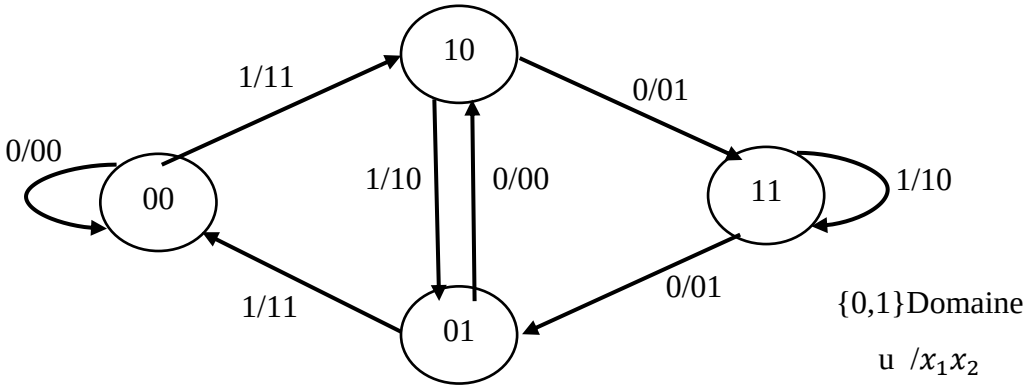

Figure 2. 5: machine d'état codeur RSC utilisé

Les bits codés doivent être mappés du domaine {0,1} au domaine {-1, + 1} pour une transmission Bpsk. la Figure 2.16 montre ce diagramme d'état modifié.

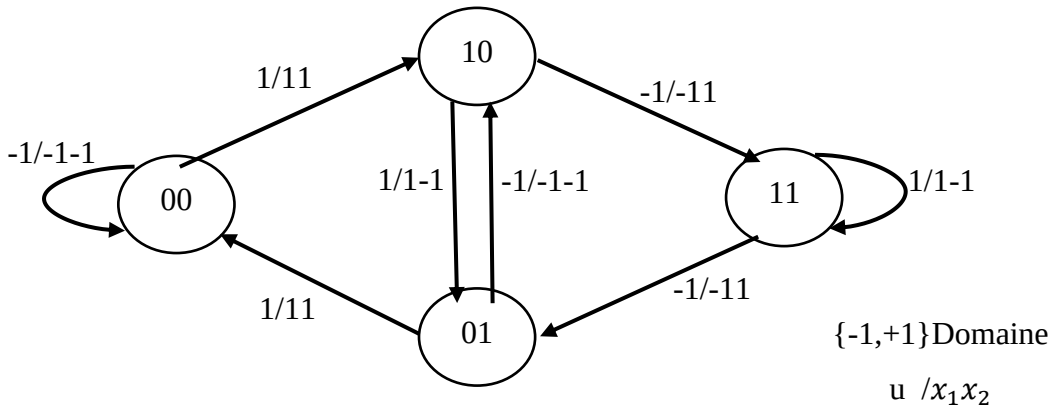

Figure 2. 6 : Diagramme d'état de transmission du codeur de composant RSC pour l'Exemple.

2.6 Conclusion

Dans ce chapitre, Nous nous sommes, intéressés au codage convolutif et en particulier aux codeurs récursifs et systématiques utilisés pour la conception des codes turbo. Nous avons expliqué le principe du turbo codage, décrit les constituants des turbo codeurs, et présenté les différentes structures de turbo codage. Le chapitre suivant exposera le principe du décodage turbo et les algorithmes utilisés.

Chapitre 3 : Décodage itératif des turbo codes

3.1 Introduction

L'opération de décodage consiste à retrouver une séquence d'état s d'un processus de Markov en présence de bruit. Les deux principales solutions sur treillis qui permettent la résolution de ce type de problème sont connues sous les noms d'algorithme de SOVA à bas de Viterbi et d'algorithme MAP. Le premier algorithme détermine la séquence d'états la plus probable à partir de la séquence reçue y selon l'équation (3.1), tandis que le deuxième estime chaque état individuellement à partir de la séquence reçue y d'après (3.2)

$$\hat{s} = \arg\max\{P(s\backslash y)\} \quad (3.1$$

$$\hat{s_i} = \arg\max\{P(s_i\backslash y)\} \quad (3.2$$

Une fois que la séquence d'états est évaluée, il devient aisé de déterminer le message où le mot de code à lui associer. En communication numérique, l'algorithme de Viterbi minimise la probabilité d'erreur par mot de code « FER» tandis que MAP minimise la probabilité d'erreur par bit (ou symbole) « BER».

3.2 Techniques de décodage

3.2.1 Décodage par Décision Douce (Soft Decision Decoding)

Dans un canal AWGN, une détection optimale est principalement une détection basée sur la minimisation de la distance Euclidienne entre le signal reçu et le signal transmis. En d'autres termes, une fois la sortie du canal reçue, soit passée par les filtres appariés (Matched Filters), on doit choisir le signal message ayant le minimum de distance au sens de Hamming avec le signal reçu. Avec l'utilisation d'une modulation (Phase Shift Key), pour la transmission des données codées, le mot code $C_l = (c_{i1}, c_{i2,\ldots}\ c_{in})$ est modélisé, en une séquence :

$$s_i(t) = \sum_{k=1}^{n} \psi_{ik}(t - (k-1)T) \quad (3.3$$

$$\text{avec:} \quad \psi_{ik}(t) \begin{cases} \psi(t) & c_{ik} = 1 \\ -\psi(t) & c_{ik} = 0 \end{cases} \quad (3.4$$

$\psi(t)$ est un signal de durée T est d'énergie E, de valeur nulle en dehors de l'intervalle [0, T]. La distance Ecludienne entre deux signaux choisis arbitrairement [18] :

$$(d_{ij}^{E})^2 = \sum_{1\le k\le n}(\pm 2\sqrt{E})^2 = 4d_{ij}^{H}E \qquad (3.5$$

Ce qui donne une simple relation entre la distance Ecludienne et la distance de Hamming, pour une modulation BPSK. Une modulation optimale consiste alors à faire passer le signal reçu *r(t)*, à travers une série de filtres appariés, en vue d'obtenir à la fin un vecteur r. Un décodage par décision soft, (Soft Decision Decoding), consiste alors à trouver le point le plus proche, au sens Ecludien (Minimum Ecludean Distance) du vecteur r. Cette méthode de décodage nécessite un traitement en nombre réel des données.

3.2.2 Décodage par Décision Dure (Hard Decision Decoding)

Une technique plus simple et plus souvent utilisée par l'attribution de valeurs binaires *(*Hard Binary Decision) aux composantes du vecteur reçues r, et trouver le code, le plus proche de r, au sens de Hamming. Il existe trois étapes élémentaires dans le processus de décodage par décision douce *(*Hard Decision Decoding*)*.

Premièrement, le signal brut reçu *r(t)* passe par une série de filtres appariés, l'échantillonneur bloqueur, pour obtenir le vecteur r.

Deuxièmement, on compare les éléments du vecteur r, avec des niveaux seuils, et quantifier chaque élément par un des deux niveaux, ce qui définit le nouveau vecteur y.

Finalement, le décodage est réalisé par la recherche du mot code le plus proche du vecteur y, au sens de Hamming.

Il est prouvé que la différence entre les performances d'un décodage par décision douce et un décodage par décision dure est autour de 2dB [19] pour un canal AWGN, par conséquent, la probabilité d'erreur d'une décision dure, dont la puissance est supérieure à 2dB, est comparable à celle d'une décision douce. D'autre part, il est encore prouvé qu'au lieu de quantifier tous les éléments du vecteur r, selon deux niveaux distincts, une quantification par huit niveaux (trois bits par élément) réduit l'écart de performances entre les deux à 0.1dB. Cette quantification à plusieurs niveaux, qui constitue un

Compromis entre le modèle de décodage par décision douce (Précision Infinie) et le décodage par décision dure.

3.3 Principe du Décodage Itératif

Le processus de décodage itératif consiste à un échange mutuel d'information dite extrinsèque (à postériori) entre deux décodeurs concaténés comme le montre la figure du turbo-décodeur ci-dessous . Le module de décodage associé à chaque code est constitué par un algorithme de type SISO (décodeur à entrées/Sorties souples). Le but du décodeur entier est d'estimer le rapport de vraisemblance de chaque bit d'information sachant le signal reçu à la sortie des filtres adaptés. Le processus se déroule à plusieurs reprises (itérations) jusqu'à ce que les vraisemblances mesurées aux sorties des 2 modules SISO convergent.

Figure 3. 1: Décodage SISO de rendement ½

Le décodeur SISO traite des informations à priori et fournit à sa sortie des informations à posteriori. Les bits souples de décision sont généralement représentés parle logarithme du rapport de vraisemblance « LRV » « LLR » donné par l'équation (3.6).

$$\Lambda_i = \ln\left(\frac{\Pr\{d_i = 1|y\}}{\Pr\{d_i = 0|y\}}\right) \quad (3.6$$

En considérant une modulation BPSK la sortie modulée est exprimée suivant l'équation (3.7).

$$y' = a\sqrt{E_c.(2c-1)} + n' \quad (3.7$$

Où **a** représenta une amplitude. Dans le cas d'un canal AWGN, a est une constante, sinon on est en présence d'un canal à évanouissement de type Rayleigh ou Rice selon la distribution de a. Le terme $\sqrt{E_c.(2c-1)}$ représente le symbole modulé, avec E_c l'énergie par symbole

modulé et c ∈ {0,1}, n' est un bruit blanc gaussien de moyenne nulle et de variance $\sigma^2 = N_0/2$. L'équation (3.6) peut être exprimée de manière simplifiée par (3.8).

$$Y = a\,(2c-1) + n \qquad (3.8$$

Dans ce cas, la variance s'exprime par $\sigma^2 = N_0/2E_0$ et le LRV par (3.9)

$$\Lambda_i = \frac{4\alpha_i^{(s)} E_c}{N_0} y_i^{(s)} + z_i + l_i \qquad (3.9$$

Où $y_i^{(s)}$ représente les bits systématiques reçus et z_i I 'information a priori qui provient du deuxième décodeur. l_i est appelé information intrinsèque et représente à chaque itération une information dérivée des LRV des deux décodeurs et de l'entrée systématique.

$$l_i = \Lambda_i - \frac{4\alpha_i^{(s)} E_c}{N_0} y_i^{(s)} - z_i \qquad (3.10$$

3.4 Turbo-Décodage

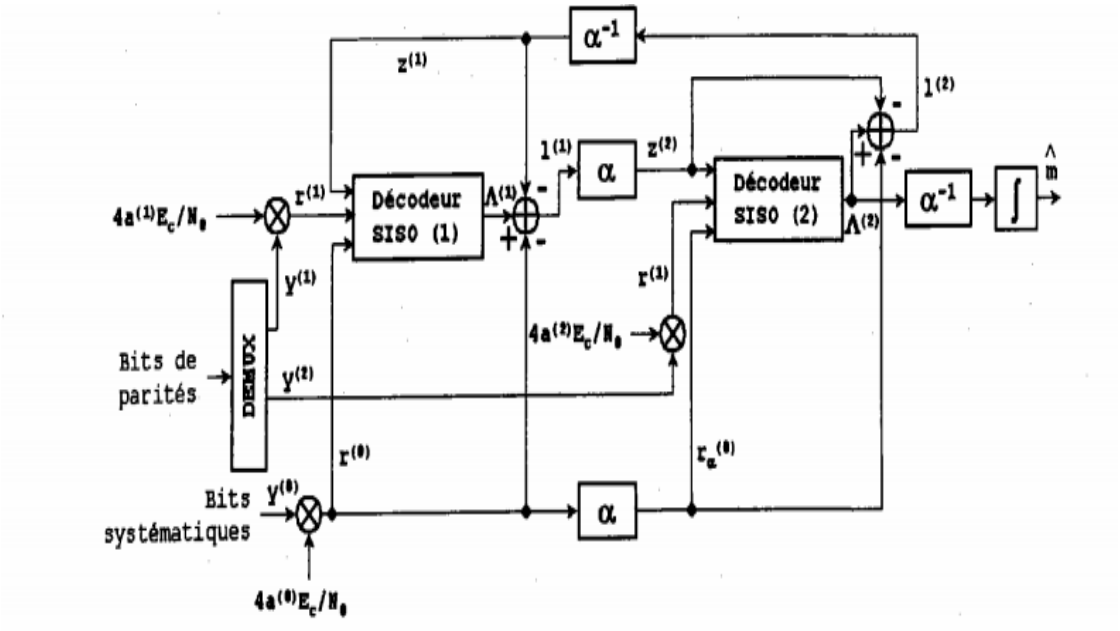

Figure 3. 2: Exemple d'un turbo décodeur

Le turbo décodeur fonctionne de la manière suivant :

1. Le premier décodeur SISO reçoit :
   - $r^{(0)}$: les bits systématiques reçus $y^{(0)}$ multipliés $\frac{4\propto_i^{(0)} E_c}{N_0}$;
   - $r^{(1)}$: les bits de parités $y^{(1)}$ multipliés par $\frac{4\propto_i^{(1)} E_c}{N_0}$
   - $z^{(1)}$: l'information p priori dérivée du deuxième décodeur. Elle représente l'information intrinsèque $l^{(1)}$ après désentrelacèrent $l_{\propto-1}^{(2)}$. $l^{(2)}$ est calculé par la soustraction de l'information intrinsèque entrelacée $l_{\propto}^{(1)}$ et $r_{\propto}^{(0)}$du premier décodeur à la sortie LRV du $\Lambda^{(2)}$deuxième décodeur.

2. Le deuxième décodeur SISO reçoit :
   - $r_{\alpha}^{(0)}$: les bits systématiques reçus $y^{(0)}$ multipliés pas $\frac{4\propto_i^{(0)} E_c}{N_0}$ puis entrelacés.
   - $r^{(2)}$: les bits de parités $y^{(2)}$ multipliés pas $\frac{4\propto_i^{(2)} E_c}{N_0}$.
   - $z^{(2)}$: l'information a priori dérivée du premier décodeur. Elle représente l'information intrinsèque $l^{(1)}$ après entrelacement $l_{\propto}^{(l)}$. $l^{(1)}$ est calculé par soustraction de l'information intrinsèque désentrelacée $l_{\propto^{-1}}^{(l)}$ et $r^{(0)}$ du deuxième décodeur à la sortie LRV $\Lambda^{(1)}$ du premier décodeur.

Après un certain nombre d'itérations qui déterminent la précision du décodage, Le message estimé $\widehat{d_\iota}$ est obtenu par seuillage de la valeur LRV $\Lambda^{(2)}$ après désentrelacement $\Lambda_{\propto^{-1}}^{(2)}$:

$$\widehat{d_\iota} = \begin{cases} 1\ si & \Lambda_{\propto^{-1}}^{(2)} \geq 0, \\ 0\ si & \Lambda_{\propto^{-1}}^{(2)} < 0 \end{cases} \qquad (3.\ 11$$

3.5 Algorithmes de décodage SISO

Il y'a deux familles d'algorithme de décodage itératif, la première est à la base de l'algorithme MAP et la seconde c'est l'algorithme SOVA.

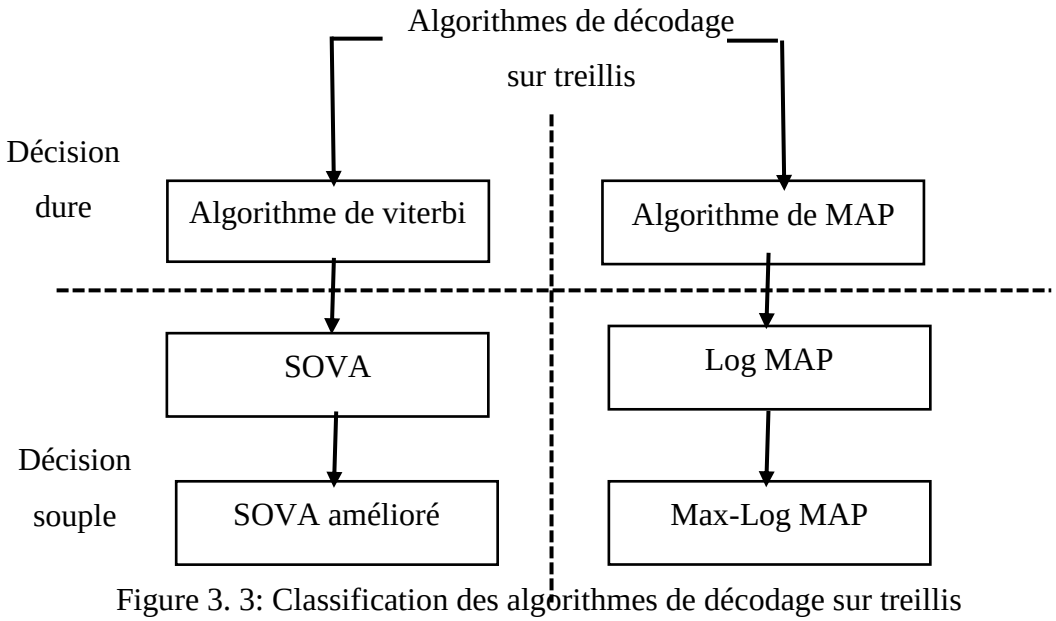

Figure 3. 3: Classification des algorithmes de décodage sur treillis

Les algorithmes de décodage SISO sont dérivés des deux principaux algorithmes de décodage sur treillis, Viterbi et MAP. L'algorithme de Viterbi à sortie souple SOVA fournit en sortie une valeur de confiance des bits estimés. L'algorithme SOVA amélioré quant à lui, applique un facteur de correction à la valeur de confiance fournie, ce qui permet d'augmenter les performances de l'approche SOVA. L'algorithme MAP nécessite des calculs intensifs d'où la complexité de sa mise œuvre. Les algorithmes qui en dérivent, Log Map et Max-Log Map [20], s'exécutent dans le domaine logarithmique d'où une complexité réduite par rapport au MAP originel. Le Log Map correspond à deux algorithmes de Viterbi « généraux » dont l'un par court le treillis en sens direct et l'autre en sens contraire. Le LRV est calculé en combinant les valeurs des métriques correspondantes dans les deux sens. Le Max-Log Map s'exécute de la même, manière que le Log Map mais en simplifiant la formule de calcul des métriques.

Du point de vue complexité, l'algorithme SOVA est le moins complexe à mettre en œuvre, en outre il requiert moins de mémoire que les algorithmes SISO dérivées du MAP. Les complexités du log MAP est double de celle du SOVA. Log MAP requiert l'utilisation d'une

table de correspondance pour les facteurs de corrections. L'algorithme Max-Log MAP présente une complexité intermédiaire. En effet, il est la version simplifiée du Log-MAP. Il ne prend pas en compte de facteur de correction et ne nécessite pas de table de correspondance. Du point de vue de la performance de décodage dans le cas d'un canal AWGN, l'algorithme Log MAP apporte une amélioration du gain de 0.5 dB par rapport aux Max-Log MAP, qui surclasse à son tour l'algorithme SOVA de 0.5dB. Quant au SOVA amélioré, il présente un gain de 0.3 à 0.5 dB par rapport à l'algorithme SOVA.

3.5.1 Algorithme de Viterbi

3.5.1.1 Introduction

Le décodage du code convolutif est obtenu par la recherche d'un chemin code dans la structure en treillis vérifiant la distance minimale au sens de Hamming avec la séquence reçue.

3.5.1.2 Les étapes d'algorithme de viterbi

Dans un chaque instant, deux branches appartenant à deux chemins différents, convergent vers chaque noeud. De ces deux chemins, l'un est plus vraisemblable, c'est-à-dire se trouve à une distance plus petite de la séquence reçue, que l'autre chemin.

Les distances étant additives, il est possible de ne conserver en chaque nœud que le chemin le plus vraisemblable, appelé survivant. Si deux chemins sont aussi vraisemblables, un seul chemin est arbitrairement conservé [21].

Exemple :

- Supposons que la séquence à l'entrée du codeur soit '1 0 0 1'.
- Si le codeur est dans l'état '00' à l'instant initial, la séquence correspondante en sortie du codeur est '11 01 11 11'.
- Considérons un canal binaire symétrique introduisant une erreur en position 4. La séquence reçue à l'entrée du décodeur est '11 00 11 11'.

Voici le déroulement de l'algorithme de Viterbi :

1) A l'instant t = 0 :

   Deux branches partent de l'état '00'. Elles sont respectivement à la distance 2 et 0 du premier couple binaire reçu. Reportons ces deux distances, appelées métriques de branche sur le treillis. (Voir figure 2.11)

2) A l'instant t = 1 :

Évaluons la distance entre le deuxième couple binaire reçu et les quatre branches qui partent des états '00' et '10', puis reportons ces quatre métriques sur le treillis.

En sommant les métriques de branches appartenant à un même chemin, nous obtenons les métriques cumulées.

Nous avons désormais quatre chemins qui permettent d'accéder, en t = 2, aux quatre états possibles du codeur. (Voir figure 3.4)

3) A l'instant t = 2 :

Il existe désormais deux chemins qui convergent vers chaque nœud du treillis. On va donc :

Calculer les métriques de branche.

Calculer les métriques cumulées pour chaque chemin atteignant en t = 3, un nœud donné du treillis.

En chaque nœud, ne retenir que le survivant. (Voir figure 3.4 et figure 3.5)

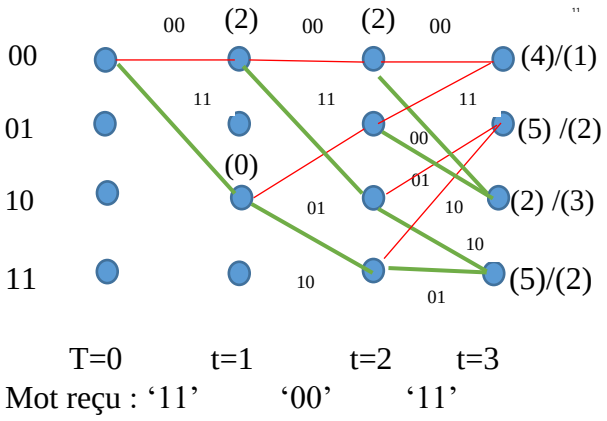

Figure 3. 4: exemple algorithme de Viterbi a t=3

4) A l'instant t = 3 :

On procède de la même façon

Finalement, le chemin le plus vraisemblable est celui qui arrive en '10'.

En remontant le treillis de la droite vers la gauche, on voit que la séquence la plus vraisemblable est celle qui part de '00' à t = 0 et qui arrive à '10' à t = 4. Elle correspond au code vraisemblablement émis : '11 01 11 11'.

Ce code correspond à une séquence sur l'entrée du codeur égale à '1001'. L'erreur en position 4 est donc corrigée. (Voir figure 3.5)

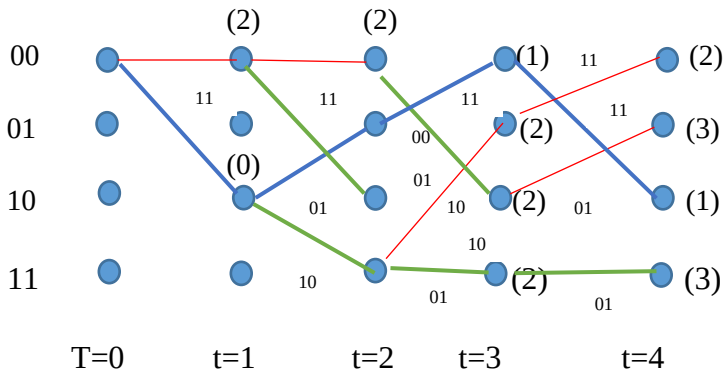

Figure 3. 5: exemple algorithme de Viterbi a t=4

3.5.1.3 Conditions d'application de l'algorithme de Viterbi

L'algorithme de Viterbi fonctionne dans les conditions suivantes :

- Données binaires
- BPSK modulé
- Transmis sur une chaîne AWGN

3.5.2 Algorithme de Viterbi à Sortie Souple (SOVA)

L'algorithme de Viterbi à sortie souple c'exécute en deux phase. Ma première consiste à exécuter l'algorithme de Viterbi standard avec quelques modifications suivies d'une étape de calcul étape de calcul des valeurs de confiance associées aux bits estimés. Les modifications apportées à la première étape sont :

1. Crées un tableau $\Delta(j,i)$ avec j∈ $\{0,\dots\dots,2^m-1\}$ et i∈ $\{0,\dots\dots,K\}$. Ce tableau contient différence entre les métriques du chemin survivant et de son compétiteur.

2. Crée un tableau C(j,i) avec j∈ $\{0,\dots\dots,2^m-1\}$ et i∈ $\{0,\dots\dots,K\}$. Ce tableau contient séquence d'état (s_0^*,…, s_i^*) du chemin compétiteur.

6. Mettre à jour le tableau $\Delta(j,i)$ selon :

$$\Delta(j,i) = max_{s_{i-1}=s'_j\in\Omega}\{\Gamma\,(j',i-1)+\lambda(s_i\to s_{i+1})\}$$

$$-\,min_{s_{i-1}=s'_j\in\Omega}\{\Gamma\,(j',i-1)+\lambda(s_i\to s_{i+1})\} \qquad (3.\ 12$$

7. Sauvegarder la séquence d'états du chemin compétiteur suivant :

$$C(j,i) = (S(j'', i-1), s_i) \quad (3.13$$

j'' correspond à l'index de l'état $S_{j''}$, qui est le paramètre de la minimisation de l'équation 3.12

La deuxième étape consiste, pour chaque bit estimé, à calculer les valeurs de confiance associées.

1. Créer un vecteur de confiance $\rho = (\rho_0, \dots \dots, \rho_{K-1})$, avec $\rho_i = \infty \ \forall i$.
2. Effectuer i=K
3. Effectuer $S_j = \widehat{s_i}$, $\hat{s} = (\widehat{s_0}, \dots, \widehat{s_K})$ étant la séquence d'états calculée par I 'algorithme de Viterbi lors de la première phase.
4. Soit $\widehat{s^*}$=C(j,i)=$\widehat{s_0^*}, \dots, \widehat{s_K^*}$) la séquence d'états associée au chemin compétiteur sélectionnée au nœud $S_{j,i}$.
5. Soit $\hat{d}$=($\widehat{d_0}, \dots, \widehat{d_i}$) le message estimé par I 'algorithme de Viterbi lors de la première phase et $\check{d} = (\widecheck{d_0}, .., \widecheck{d_i})$ le message associé au chemin compétiteur sélectionné au nœud $S_{j,i}$.
6. Effectuer k=1.
7. Si $\widecheck{d_{i-k}} \neq \widehat{d_{i-k}}$, alors $\rho_{i-k} = \min\{\rho_{i-k}, \Delta(j,i)\}$
8. Effectuer k=k+1.
9. Si $\widehat{s_{i-k}^*} = \widehat{s_{i-k}}$ aller à l'état 10, sino revenir à l'étape 7.
10. Effecteur i=i-1.
11. Si i>0 revenir à l'étape 3, sinon aller à l'étape 12.
12. Calculer les LRV suivant l'équation (4.29).

$$\Lambda_i = (2\widehat{d_i}\text{-}1)\rho_i \text{ , i}\in \{0, \dots .., K\} \quad (3.14$$

Remarque :

Dans le cas de codes de rendement 1/N, deux branches sont connectées à chaque nœud du treillis.

3.5.3 Algorithme SOVA amélioré

Il a été démontré dans [22] que les sorties estimées Λ_i fournies par l'algorithme SOVA sont biaisées. En effet l'algorithme SOVA doit fournir un LRV de la forme :

$$L_i = \frac{P(d_i = 1 \backslash \Lambda_i)}{P(d_i = 0 \backslash \Lambda_i)} = \frac{2d_\Lambda}{\sigma_\Lambda^2}\Lambda_i \quad (3.15$$

Le SOVA amélioré corrige les estimations biaisées du SOVA en les multipliant par un facteur de correction F_c.

$$F_c = \frac{2d_\Lambda}{\sigma_\Lambda^2} \quad (3.16$$

Dans ce cas la moyenne et la variance de la sortie SOVA doivent être estimées au préalable ce qui a pour effet d'augmenter la complexité du circuit. Néanmoins, la correction des sorties SOVA apporte une amélioration de 0.3 à 0.5 dB par rapport au SOVA sans correction.

3.5.4 Algorithme MAP

L'algorithme MAP a été proposé pour la première fois en 1974 par Bahl et al. [23]. Deux versions de cet algorithme existent, la version appelée type-I parcourt le treillis dans les sens direct et inverse, elle est orientée vers le traitement en bloc des données. Quant à la version type-II, elle parcourt le treillis uniquement en sens direct et est orientée vers le traitement continu des données. Elle est plus complexe à mettre en œuvre et nécessite plus de mémoires que la type-I.

Comme les turbo-codes sont assimilés à des codes blocs, L'algorithme MAP de type-I est utilisé de préférence [24]. Dans le processus de décodage, l'algorithme MAP calcule les probabilités de justesse a posteriori (APP) des bits du message reçu. P($d_i = 1\backslash y$) et P($d_i = 0\backslash y$) sont ensuite mises sous la forme de rapports de vraisemblance logarithmique (LRV). En utilisant la définition de la probabilité conditionnelle (équ.3.17) et en appliquant les propriétés du processus de Markov pour partitionner le numérateur on obtient l'équation (3.18) :

$$\mathrm{P}(s_i \to s_{i+1}\backslash y) = \frac{\mathrm{P}(s_i \to s_{i+1}, y)}{P(y)} \quad (3.17$$

$$\mathrm{P}(s_i \to s_{i+1}, y) = \propto (s_i)\gamma(s_i \to s_{i+1})\beta(s_{i+1}) \quad (3.18$$

Avec :

$$\propto (s_i) = P(s_i, (y_0, .., y_{i+1})) \quad (3.19$$

$$\gamma(s_i \to s_{i+1}, y) = \mathrm{P}(s_{i+1}, y_i\backslash s_i) \quad (3.20$$

$$\beta(s_{i+1}) = P((y_{i+1}, \dots, y_{k-1})\backslash s_{i+1}) \quad (3.21$$

Avec : $(s_i \to s_{i+1})$ la transition d'état.

3.5.5 Algorithme log MAP

Dans la pratique l'algorithme MAP est difficile à mettre en œuvre car il nécessite un grand nombre de calculs pour chaque bit estimé (6 x 2^m multiplications + 6 x 2^madditions). De plus, il est sensible aux erreurs d'arrondis lors qu'on utilise des valeurs numériques de précision finie. Ces deux contraintes pratiques sont résolues en exécutant L'algorithme MAP directement dans le domaine logarithmique, au lieu d'appliquer le logarithme au rapport de vraisemblance à la dernière étape. Dans le domaine logarithmique, toutes les multiplications se transforment en addition.

3.5.6 Algorithme Max-Log MAP

Comme pour Log MAP, l'algorithme Max-Log MAP s'exécute dans le domaine logarithmique Seulement, par conséquent, I 'algorithme Max-Log MAP effectue le calcul des $\propto_l (s_i)$ et $\beta_l(s_i)$ en utilisant l'algorithme de Viterbi. L'algorithme Max-LogMap s'exécute de la même façon que le Log Map mais en prenant les valeurs maximum.

3.6 Comparaison des algorithmes de décodage SISO

Du point de vue complexité, l'algorithme SOVA est le moins coûteux et nécessite moins de mémoire. Il peut être implanté en associant à un décodeur de Viterbi standard un co-processeur pour calculer les valeurs de confiance [25]. Dans le cas du SOVA amélioré, chaque sortie est multipliée par un facteur de correction dont la valeur dépend de la moyenne et de la variance des sorties. Chaque facteur doit être calculé au préalable ce qui a pour effet d'augmenter la complexité du circuit. L'algorithme Max-Log MAP réduit considérablement la complexité du MAP qui nécessite un grand nombre de multiplications et l'utilisation de fonctions non-linéaires [26]. L'exécution de I 'algorithme dans le domaine logarithmique permet de réduire toutes les opérations à des additions et des calculs de « max » [26]. Le Max-LogMAP requiert cependant beaucoup de mémoire pour stocker les valeurs des métriques cumulées. Le Log MAP applique un terme de correction à I'algorithme du Max-Log MAP, ce qui lui permet de compenser l'erreur due à l'approximation

A cet effet, une table de correspondance doit être utilisée pour stocker les facteurs de correction [26].

En pratique, pour réduire la complexité du circuit, I 'algorithme SOVA peut s'exécuter uniquement sur des blocs partiels d'information. La taille du bloc partiel traité (fenêtre) doit être choisie suffisamment grande pour pouvoir supposer que les chemins survivants sont assez fiables.

A chaque fin traitement, le chemin qui a la plus faible métrique cumulée est choisi. En cas d'égalité dans la métrique de deux survivants, on peut utiliser un algorithme permettant de les départager ou en prendre un au hasard. L'algorithme SMAP (slidingMAP) permet de réduire les besoins en mémoire de I 'algorithme MAP en s'exécutant sur des blocs partiels de données [27] Cependant comme pour le SOVA, I 'algorithme doit fournir un point de départ fiable pour le traitement des blocs partiels suivants. Cette technique existe aussi pour le Log MAP et le Max-Log MAP (SLog-MAPet SMax Log MAP). L'algorithme Log MAP apporte un gain de 0.5 dB par rapport au Max-Log MAP et le Max-Log MAP un gain de 0.5dB par rapport au SOVA [24,25].

Le tableau 3.1donneun bref comparatif des principaux algorithmes de décodage SISO vue précédemment :

	Log MAP	Max-log MAP	SOVA	SOVA amélioré
Performance	-0.3 dB[a] +0.5 dB[c]	+0.5 dB[b]	Le moins performant	+0.3 à +0.5 dB[b]
complexité	2*SOVA	Log-Map< ML Map Ml Map<SOVA	Le moins complexe	SOVA <

tableau 3. 1 : Comparaison des algorithmes de décodage SISO

L'ensemble des algorithmes SISO passés en revue sont basés sur l'algorittrme de Viterbi. Le SOVA et le SOVA amélioré utilisent des versions modifiées de Viterbi. L'algorithrne Max-LogMAP utilise deux algorithmes de Viterbi, l'un parcourant le treillis en sens direct pour le calcul de $\propto_i(s_i)$ et le deuxième parcourant de treillis en sens inverse pour le calcul des $\beta_i(s_i)$.

Le Log Map enfin, utilise deux algorithmes de Viterbi généraux et s'exécute de la même façon que le Max-LogMap.

3.7 Conclusion

Ce chapitre est consacré au turbo décodage, proprement dit décodage itératif avec les algorithmes à entrées-sorties douces SISO, à savoir l'algorithme de Viterbi à sortie douce SOVA et l'algorithme de maximum a posteriori MAP. L'algorithme MAP semble meilleur, mais il est extrêmement complexe à cause du nombre de multiplications éffectuées. L'algorithme Log-MAP réduit la complexité de l'algorithme MAP par sa transformation en domaine logarithmique (réduit l'effet des multiplications) en gardant les mêmes performances. L'algorithme Max-Log-MAP simplifie la complexité de MAP, mais ceci dégrade les performances de 0.35 dB [16] par rapport à l'algorithme MAP. Le dernier algorithme SOVA réduit de façon plus nette la complexité par rapport aux autres algorithmes, mais il présente une dégradation en performances de 0.6 dB [16] par rapport à l'algorithme MAP.

[A] : Dégradation par rapport au MAP

[B] : Dégradation par rapport au SOVA

[C] : Amélioration par rapport au Max-Log MAP

Chapitre 4 : Turbo décodeur SOVA

4.1 Introduction

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 entrer 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 et il peut ainsi les utiliser dans l'estimation de la séquence à corriger.

4.2 Algorithme SOVA pour turbo décodge

L'algorithme SOVA est une modification de l'algorithme de Viterbi pour lui permettre d'être utilisé comme élément décodeur dans le turbo décodeur [28,29]. 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. Un examen de prés de l'algèbre de vraisemblance logarithmique est requis avant d'essayer d'utiliser la métrique modifiée de Viterbi. La figure 4.1 montre le modèle du système qu'on va utiliser pour décrire l'algorithme.

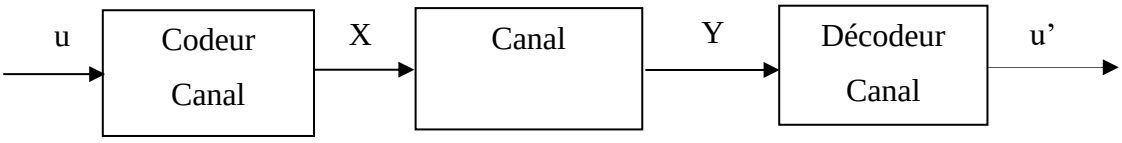

Figure 4. 1 : Modèle de la chaine de transmission

Le décodage par décisions douces nécessite la définition d'une algèbre qui nous permettra de s'adapter et facilitera le calcul des différentes métriques, c'est l'algèbre de vraisemblance logarithmique, elle est utilisée pour le décodage SOVA des turbo codes et est basée sur une variable binaire aléatoire u dans le corps de Galois CG(2) avec les éléments {+1,-1}, où -1 représente le zéro '0' logique (élément neutre) et +1 représente l'élément logique '1'

sous l'addition modulo 2 ($\oplus$) [29]. La table de vérité présentée dans le tableau 4.1 montre le résultat de la somme de deux variables binaires aléatoires en utilisant le modèle décrit ci-dessus.

U1\U2	U2=+1	U2=-1
U1=-1	+1	-1
U1=+1	-1	+1

Tableau 4. 1 : Résultat de l'addition de deux variables aléatoires binaires U1 et U2.

Pour une variable binaire aléatoire u, le taux de vraisemblance logarithmique (LLR) : L(u) est définis comme étant :

$$L(u) = \ln\left(\frac{P(u=+1)}{P(u=-1)}\right) \quad (4.1$$

L(u) est souvent appelé valeur douce (Soft Value) ou LLR de la variable binaire aléatoire u. Le signe de L(u) représente la décision dure de u, l'amplitude de L(u) est la fiabilité de cette décision. L(u) donne une forme de fiabilité à u. La valeur de LLR pour un chemin du treillis est la somme des LLR des différents bits et est donnée par l'égalité (5.2) et (5.3).

$$\sum_{\substack{[+]\\ j=1}}^{j} L(u_j) = L\left(\sum_{\substack{\oplus\\ j=1}}^{j} u_j\right) \quad (4.2$$

$$\approx \left(\prod_{j=1}^{j} sign(L(u_j))\right)_{j\,=\,1..t}^{Min}\{L(uj)\} \quad (4.3$$

4.2.1 Canal à sorties douces

À partir du modèle du système de la figure 4.1, le bit d'information u est traduit en un bit codé x. Les bits codés x sont transmis à travers le canal et reçue comme étant y. à partir de ce modèle le taux de vraisemblance logarithmique de x conditionné par y est calculé comme suit :

$$L(x|y) = ln\left(\frac{P(x=+1|y)}{P(x=-1|y)}\right) \quad (4.4$$

En utilisant le théorème de bayes, ce taux est équivalent à :

$$L(x|y) = ln\left(\frac{P(y|x=+1)}{P(y|x=-1)}\left(\frac{P(x=+1)}{P(x=-1)}\right)\right) \quad (4.5$$

$$= ln\frac{P(y|x=+1)}{P(y|x=-1)} + ln\left(\frac{P(x=+1)}{P(x=-1)}\right) \quad (4.6$$

Le canal est supposé être à atténuation plate avec bruit gaussien. Le modèle du canal est représenté par la fonction de pondération gaussienne [30]:

$$f(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(zim)^2}{2\sigma^2}} \quad (4.7$$

Où m est la moyenne es σ^2 est la variance du bruit, on peut montrer que [30]:

$$L(x|y) = ln\left(\frac{P(y|x=+1)}{P(y|x=-1)}\right) = ln\frac{e^{-\frac{E_b}{N_0}(y-a)^2}}{e^{-\frac{E_b}{N_0}(y+a)^2}} \quad (4.8$$

$$= ln\frac{e^{\frac{E_b}{N_0}2ya}}{e^{\frac{-E_b}{N_0}2ya}} \quad (4.9$$

$$= 4\frac{E_b}{N_0}\, ay \quad (4.10$$

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 :

$$L(x|y) = L_c y + L(x) \quad (4.11$$

Où L_c est définie comme étant la fiabilité du canal :

$$L_c = 4\frac{E_b}{N_0}\, a \quad (4.12$$

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

4.2.2 Calcul de Métriques

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 [28]. La figure 4.2 montre les entrées et les sorties de l'élément décodeur SOVA.

Figure 4. 2 : Élément décodeur SOVA

L'élément décodeur SOVA traite les entrées L(u),

L_cy et L_cy 1 où

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

Les séquences 'y' et 'y1' 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 la séquence des taux de vraisemblances logarithmiques.

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) [28,29,30].

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 [28]:

$$P\left(S^{(m)}\middle|y\right) = P(y|S^{(m)})\frac{P(S^{(m)})}{P(y)} \quad (4.\ 13$$

Partant de la relation (4.13), on détermine la métrique SOVA comme étant l'égalité suivante :

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c\, y_{t,1} + \sum_{j=2}^{N} X_{t,j}^{(m)}\ L_c Y_{t,j} + u_t^{(m)} L(u_t) \quad (4.\ 14$$

On remarque des relations (4.14) 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 5.3 montre la fiabilité de la source utilisée dans le calcul de métrique SOVA.

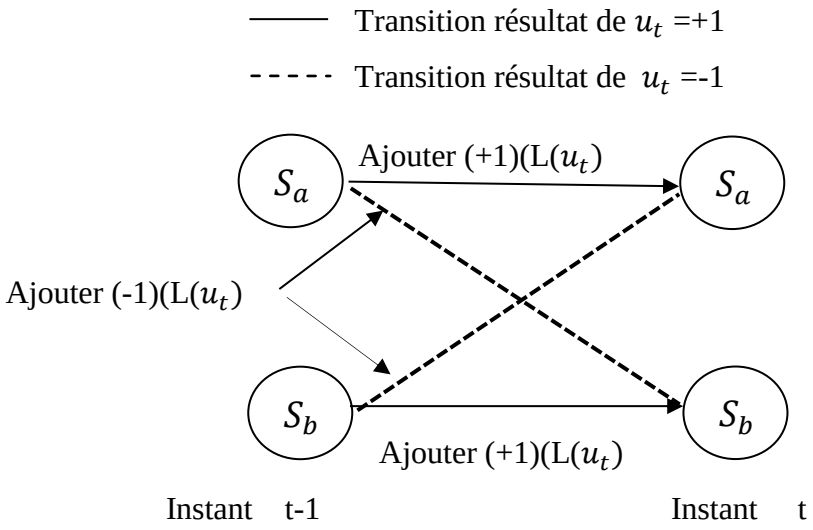

Figure 4. 3 : Fiabilité de la source pour le calcul de la métrique SOVA

La figure 4. 3 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(ut), 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.

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 [28].

La figure 4.4 montre l'importance de la balance entre la fiabilité de canal et celle de la source pour la métrique SOVA.

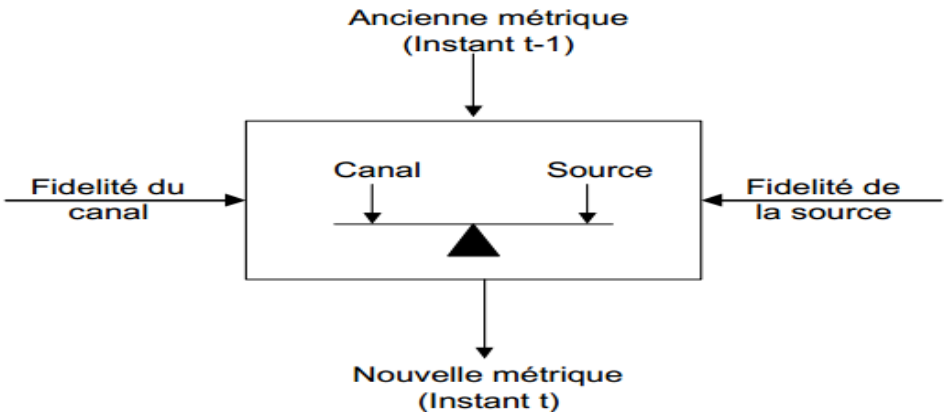

Figure 4. 4: propriétés de poids de la métrique SOVA

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_cy 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 l'instant t, la valeur de fiabilité (grandeur du rapport log-vraisemblance) affectée à un nœud dans le treillis est déterminée à partir de :

$$\Delta_t^0 = \frac{1}{2}\left|M_t^{(1)} - M_t^{(2)}\right| \qquad (4.15$$

Où :

Δ(t) désigne la valeur de fiabilité par rapport au temps.

Le décodeur SOVA peut être implémenté de différentes manières. La mise en œuvre directe du décodeur SOVA peut devenir intensive en termes de calculs pour les codes K de grande longueur de contrainte et les tailles de trame longues en raison de la nécessité de mettre à jour tous les chemins de survivant. Parce que la procédure de mise à jour est significative uniquement pour le chemin ML, une implémentation du décodeur SOVA qui exécute uniquement la procédure de mise à jour pour le chemin ML est représentée sur la Figure 4.5.

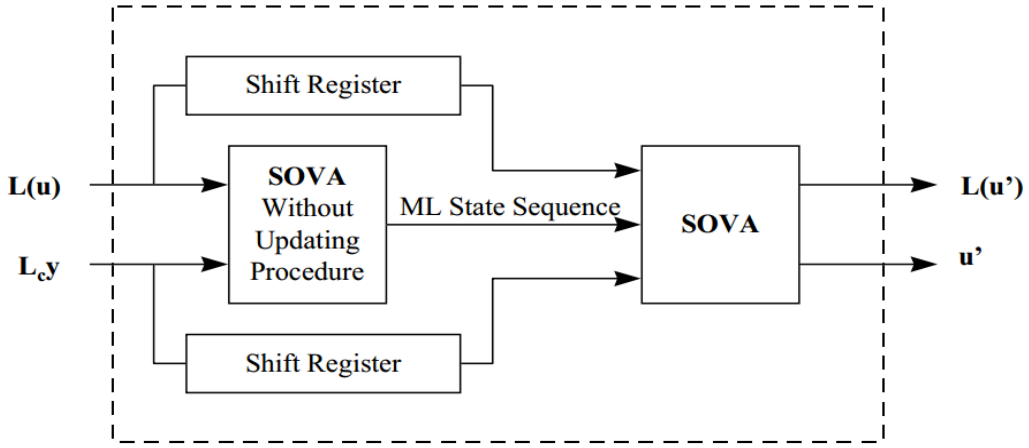

Figure 4. 5: implementation décodeur SOVA.

Le décodeur SOVA a comme entre L (u) et L_cy (les valeurs a priori et les valeurs reçues pondérées respectivement) et comme sorties u 'et L (u')(les décisions de bits estimées et leurs valeurs "soft" ou L associées, respectivement.) Cette implémentation du décodeur SOVA est composée de deux décodeurs SOVA distincts. Le premier décodeur SOVA calcule les métriques pour le chemin ML uniquement et ne calcule pas (supprime) les valeurs de fiabilité.

Les registres à décalage sont utilisés pour mettre les entrées en tampon pendant que le premier décodeur SOVA traite le trajet ML. Le second décodeur SOVA (avec la connaissance du chemin ML) recalcule le chemin ML et calcule et met également à jour les valeurs de fiabilité. Comme on peut le voir, cette méthode de mise en œuvre réduit la complexité du processus de mise à jour. Au lieu de suivre et de mettre à jour 2 m de chemins survivants, seul le chemin ML doit être traité.

4.3 Turbo Décodeur

Le décodeur itératif de turbo code est composé de deux décodeurs SOVA concaténés Parallèle. La Figure 4. 6 montres la structure du décodeur de turbo code.

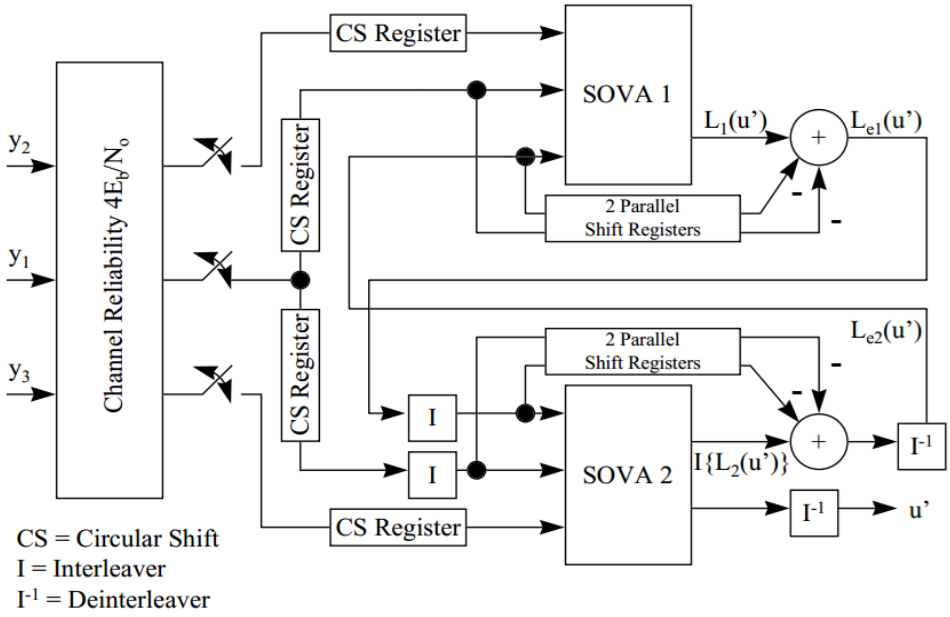

Figure 4. 6 : décodeur iterative SOVA de turbo code .

Le décodeur de turbo code traite les bits de canal reçus sur une base de trame. Comme le montre la Figure 4.6, les bits de canal reçus sont démultiplexes dans le flux systématique y1 et deux flux de contrôle de parité y2 et y3 à partir des codeurs de composant 1 et 2 respectivement.

Ces bits sont pondérés par la valeur de fiabilité du canal et chargés sur les registres CS. Les registres représentés sur la figure 4.6 sont utilisés comme tampons pour stocker les séquences jusqu'à ce qu'elles soient nécessaires. Les commutateurs sont placés en position ouverte pour empêcher le traitement des bits de la trame suivante jusqu'à ce que la trame actuelle ait été traitée.

Le décodeur de composant SOVA produit le "soft" ou L-valeur L (u_t) pour le bit estimé (pour le temps t). Le "soft" ou L-valeur L (u_t) peut être décomposé en trois termes distincts.

$$L(u_t') = L(u_t) + L_c y_{t,1} + L_e(u_t) \quad (4.16$$

L(u_t) : est la valeur a priori et est produite par le décodeur SOVA précédent.
L_cy : est la valeur de canal systématique pondérée reçue.
$L_e(u_t)$: Est la valeur extrinsèque produite par le présent décodeur de composant SOVA. L'information qui est transmise entre les décodeurs de composant SOVA est la valeur extrinsèque.

$$L_e(u_t) = L(u_t') - L_c y_{t,1} - L_e(u_t) \quad (4.17$$

4.4Conclusion

Dans ce chapitre, nous avons décrit le procédé de décodage turbo à base de SOVA et développé les calculs nécessaires à l'aboutissement de l'opération de décodage. Une architecture globale du Turbo-décodeur SOVA est proposée en fin de chapitre.

Chapitre 5 : Implémentation

5.1 Introduction

A la constante évolution des applications vers des systèmes miniaturisés à hautes performances, les systèmes reconfigurables apportent une réelle réponse s'appuyant sur le développement d'architecteurs.

Dans ce chapitre nous avons implémenté un turbo codeur et décodeur itératif à base de SOVA, en utilisant la carte développement Nexys2 de la famille SPARTAN-3E et l'outil de développement Modelsim : qui permettre la simulation et Xilinx ISE 14.7 : qui permet la synthèse et l'implémentation du circuit suivant les étapes de flot de conception de Xilinx [31] (ISE design flow).

5.2 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 schémas traditionnelles. L'outil de développement ISE (Integrated Software Environnement) de Xilinx – qu'on a utilisé pendant notre projet – permet les deux types de saisie.

5.3 Circuits FPGA

Field-Programmable Gate Array (un réseau de portes logiques programmables) est un circuit intégré conçu pour être reconfiguré après production.

Sa configuration se fait grâce à un langage de description de matériel (HDL), ou bien un circuit, cette dernière méthode est rarement utilisée de nos jours [32].

5.4 Fonctionnement et composition des FPGAs de Xilinx

Pour mémoriser leur configuration, ces circuits programmables utilisent différentes méthodes. Comme la technologie "fusible", "anti-fusible" ou SRAM (static RAM). Les Xilinx utilisent la technologie SRAM. Cela permet non seulement de reprogrammer le système mais également la réalisation de configurations reprogrammables dynamiquement : c'est-à- dire qu'il est possible de changer la fonctionnalité du FPGA (Field Programmable Gate Array), ou d'une partie du FPGA. En cours de traitement. Dans le cas de la technologie SRAM, le programme est envoyé au circuit soit via une ROM(Read only Memory), soit via un système dit "intelligent" tel qu'un microprocesseur. Un circuit FPGA est composé de quatre parties (figure 5.1)

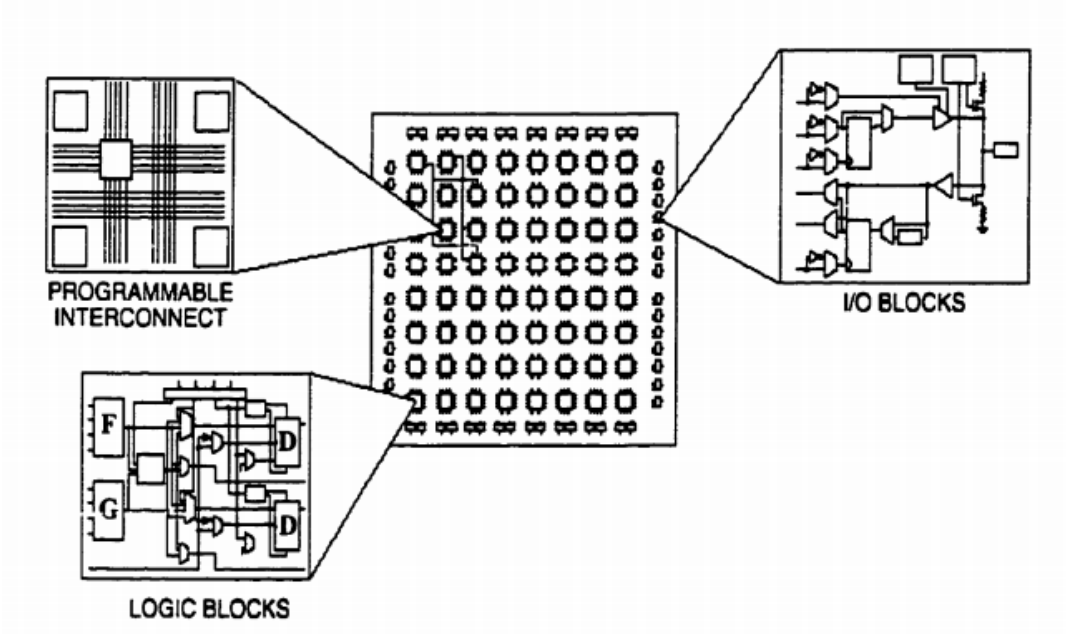

Figure 5. 1: Architecture interne d'un circuit FPGA

- Des blocs d'entrées/ sorties configurables (IOB, input/output blok)
- Des blocs de logiques configurables (CLB, Combinatorial logic Block)
- Des matrices d'interconnexions programmables (Switch Matrix)
- De lignes de connexion de différentes longueurs (long line, short line connection…)

Les CLB des Xilinx peuvent non seulement être utilisés pour des fonctions combinatoires et synchrones mais également comme des modules de RAM. Chaque CLB contient 3 look-up table (LUT : Look-Up Table), nommées F,G et H, et 2 bascules D ainsi que

quelques autres circuits (multiplexeur ..). Chaque bascule D possède un reset, un set et un enable indépendants. Les CLB sont arrangés suivant un tableau à deux dimensions. Chaque LUT est une mémoire RAM(Random access memory) de un bit de large et de 16 bits de long. Elle est donc capable de simuler n'importe quelle fonction logique à quatre entrées et une sortie. De plus les CLB possèdent des circuits spécifiques pour permettre l'implémentation efficace de fonction arithmétique comme le fast carry logic pour l'addition.

Chaque IOB gère une des pattes du composant. Chacune peut être configurée en entrée, en sortie ou en entrée-sortie. Elle peut ou non utiliser une bascule D et peut être configurable ou non en haute impédance.

L'interconnexion des CLB utilise des canaux, aussi bien verticaux qu'horizontaux, formés dematrices de connexion, switch matrix, et de lignes de différentes longueurs. Les canaux relientaussi les CLBs aux IOBs. Les lignes de connexion se divisent en trois groupes : les lignes courtes, short line. qui vont jusqu'à la matrice de connexion la plus proche, les lignes longues, long lines. qui s'étendent sur la longueur de 2 CLBs et les très longues lignes, very long line, qui vont d'un bout à l'autre du Xilinx. Des bascules trois états, inclus entre les CLBs, permettent de relier ensemble plusieurs lignes de maniéré dynamique. Un point ennuyeux à noter pour Xilinx est que pour aller d'un CLB à un autre, un signal passe à travers un certain nombre de matrice de connexion et de lignes de connexion. Le temps d'interconnexion va donc dépendre du nombre de matrices de connexion traversées. C'est-à-dire de comment le signal a été routé par le logiciel de CAO (Conception Assisté par Ordinateur).

Xilinx a introduit les premiers FPGA vers 1985 avec sa famille XC2000. Depuis, trois autresfamilles Xilinx ont été introduites : les XC3000,XC4000 et les XC5000. Actuellement lesXC4000 sont les plus utilises. Le plus gros Xilinx actuellement sur le marché, le Xc4025, possède 1024CLB, 256 IOB et est équivalent à 25 000 portes logiques élémentaires.

5.5 Caractéristiques de la carte Nexys 2

On notera quelques-unes[33] :

- Xilinx XC3S500E Spartan-3E FPGA.
- 500K Porte Xilinx Spartan 3E FPGA
- Configuration FPGA basée sur USB2 et transferts de données à haute vitesse (en utilisant le logiciel
- gratuit Adept Suite)
- Alimentation USB (les piles et / ou les prises murales peuvent également être utilisées)

- 16 Mo de microns PSDRAM et 16 Mo de mémoire Intel StrataFlash
- Xilinx Platform Flash pour les configurations FPGA non volatiles
- Alimentations à découpage efficaces (bon pour les applications alimentées par batterie)
- Oscillateur 50MHz plus socket pour second oscillateur

- 60 E / S FPGA acheminées vers des connecteurs d'extension (un connecteur haute vitesse Hirose FX2 et quatre en-têtes à 6 broches)

- 8 LED, affichage à 7 chiffres à 4 segments, 4 boutons poussoirs, 8 interrupteurs à glissière.
- PS/2 port pour sourie ou clavier et une interface JTAG.

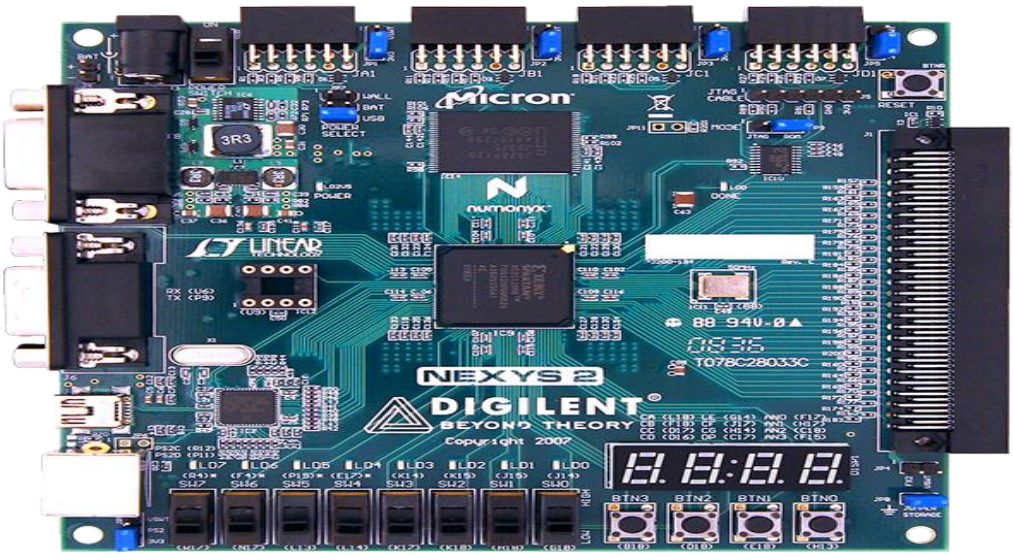

Figure 5. 2: Carte Nexys 2

5.6 Simulation et implémentation

Les simulations qui on a fait dans cette partie est fait par MATLAB et Xilinx ISE avec un exemple.

5.6.1 Implémentation et simulation d'un codeur Récursif systématique

Le codeur RSC utilise dans l'implémentation apparaitre dans la figure 5.3 suivant :

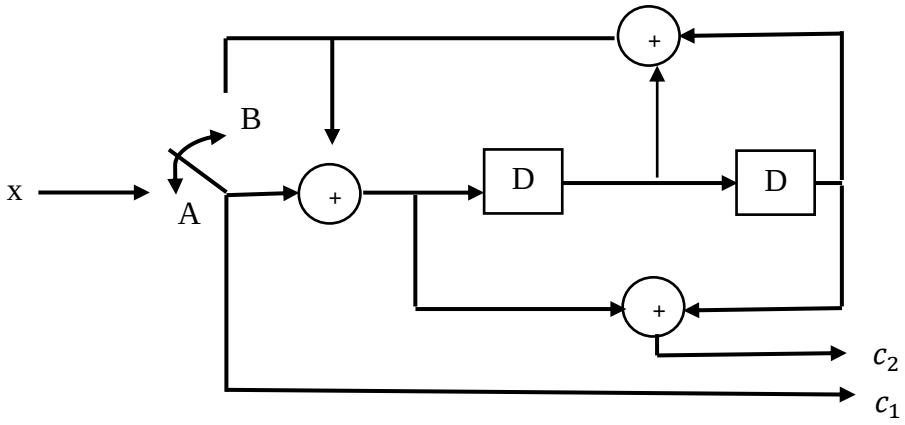

Figure 5. 3: Codeur RSC de rapport R=1/2

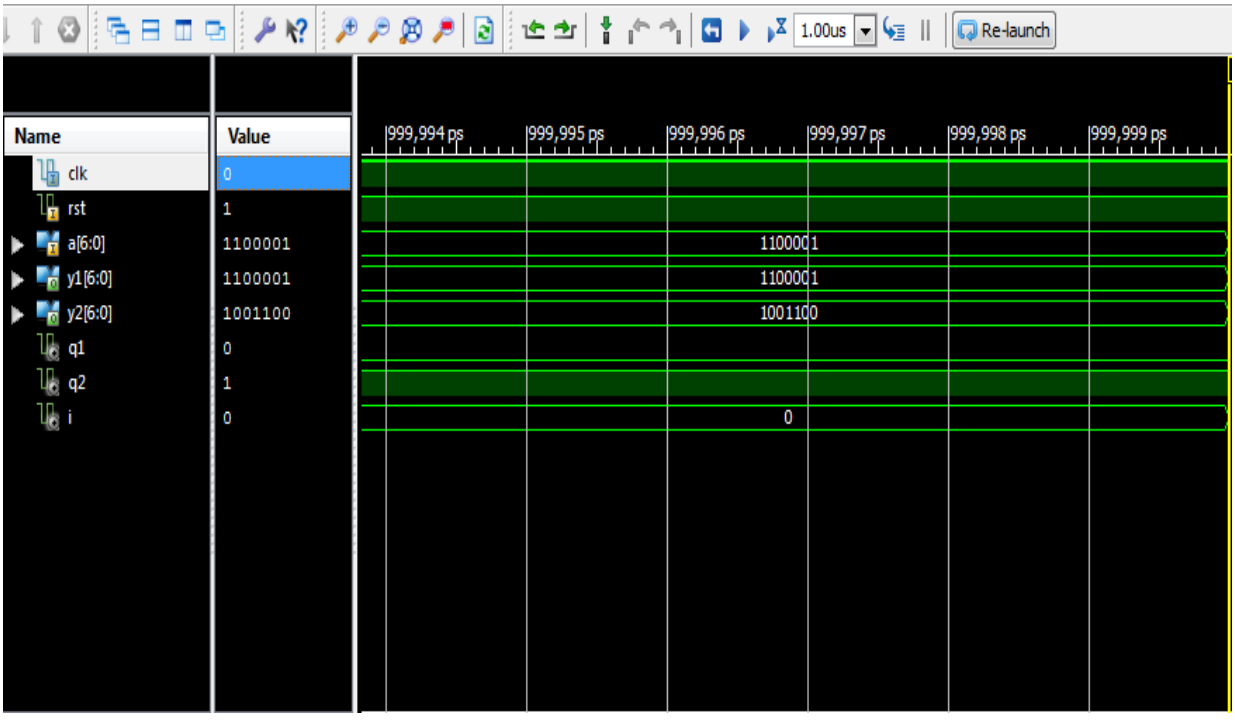

Figure 5. 4: Simulation d'un RSC codeur

Avec :

- a : présente l'entré a codé : {1,1,0,0,0,0,1}
- Y1 : représente la sortie systématique : {1,1,0,0,0,0,1}.
- Y2 : représente la sortie codée avec le codeur RSC{1,0,0,1,1,0,0}.

5.6.2 Simulation du turbo codeur (avec un rapport R= 1/3)

On prend un exemple d'une séquence d'entrée u={1,0,0,1,1}.telle que Y_1 est la sortie systématique entrelacé, Y_2est la sortie du premier codeur (RSC1) et Y_3 est la sortie du deuxième codeur (RSC2). Les résultats de la simulation du turbo codeur sont représentés dans la Figure 5.6. Et les résultats de la synthèse sont représentés dans Figure 5.7.

Le codeur turbo codeur utilisé dans l'implémentation apparait dans la figure 5.5 suivant :

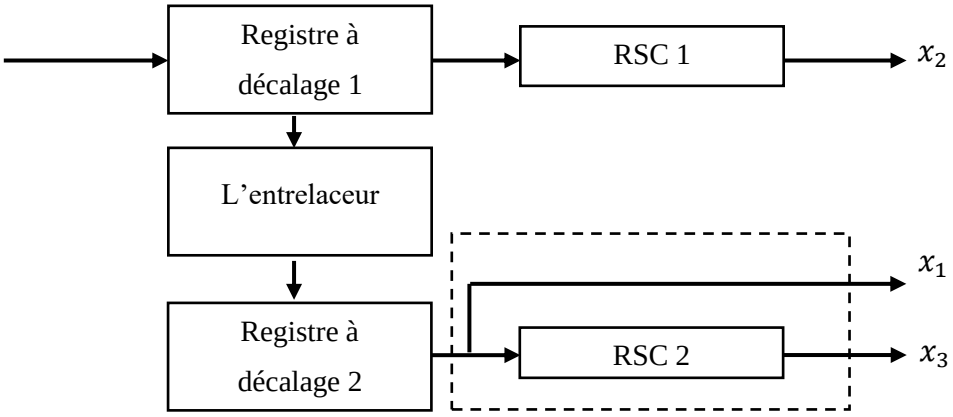

Figure 5. 5 : Structure d'encodeur de code turbo

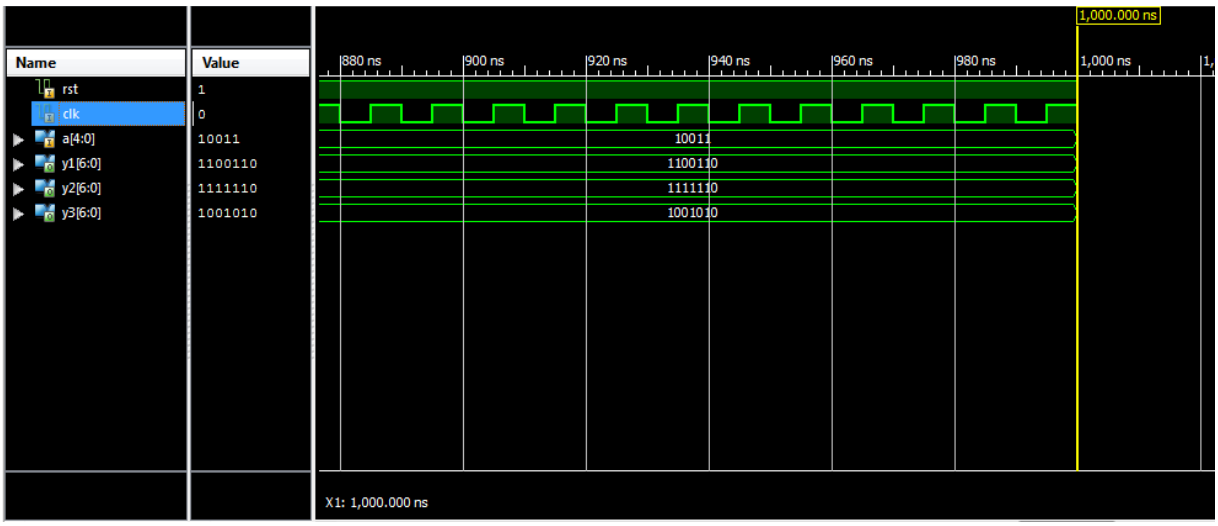

Figure 5.6 : Simulation du turbo codeur

On a obtenir des sorties :

Y_1={1,1,0,0,1,1,0},Y_2={1,1,1,1,1,1,0} et Y_3={1,0,0,1,0,1,0}.

avec l'ajout des {10} après le codage (tail biting).

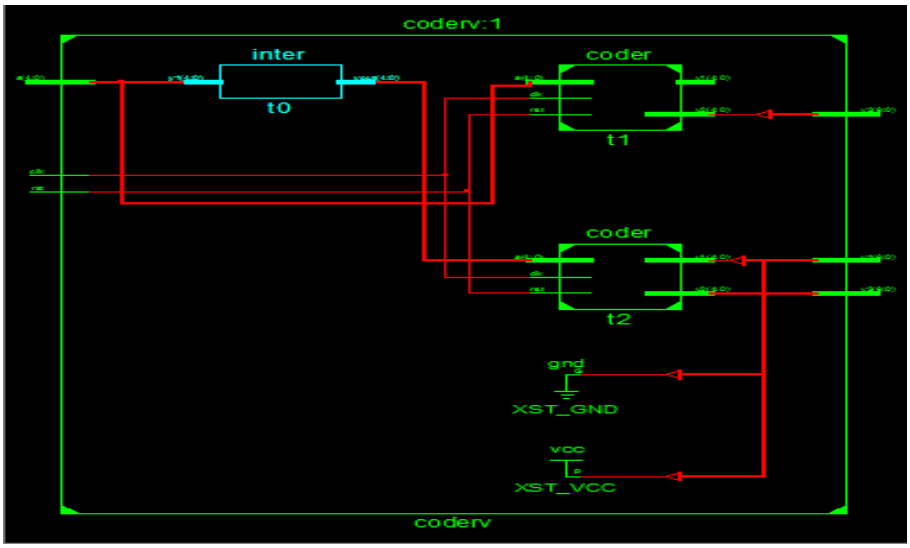

Figure 5. 7 : Le schéma RTL du turbo codeur

5.6.3 Architecture du Bloc SOVA

5.6.3.1 Déroulement des taches exécutées par le bloc SOVA

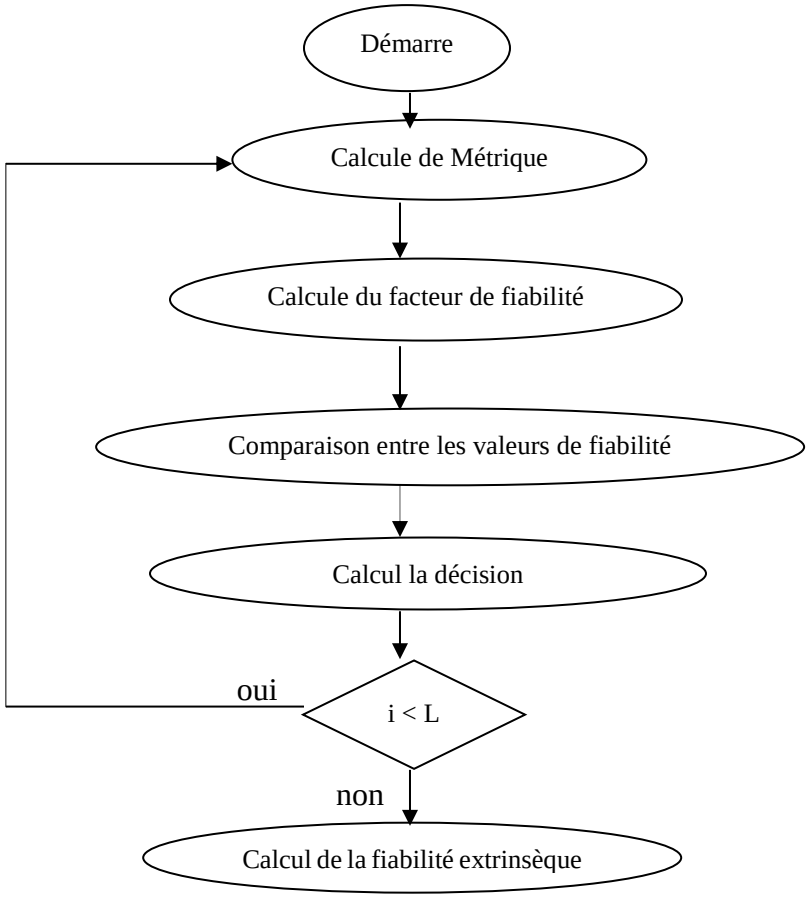

Figure 5. 8 : Déroulement des taches du turbo-code

5.6.3.2 Structure de turbo décodeur

1-SOVA décodeur

Constitué de :

- 1Bloc PPME : constitué de 4 :
  - **Le bloc PMEc** (path metric calculator) calcule les métriques arrive à chaque état.la figure 5.9 nous illustre un RTL schématique de PPME.

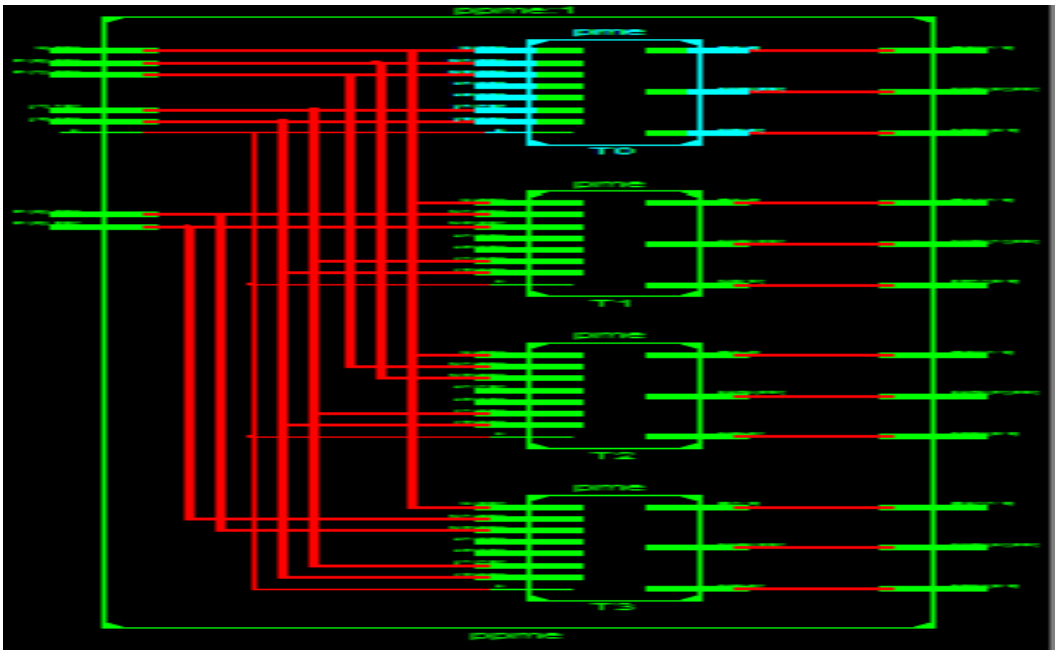

Figure 5. 9 : RTL schématique de PPME

- **Bloc BMC** fait les calculs dès métriques de la branche

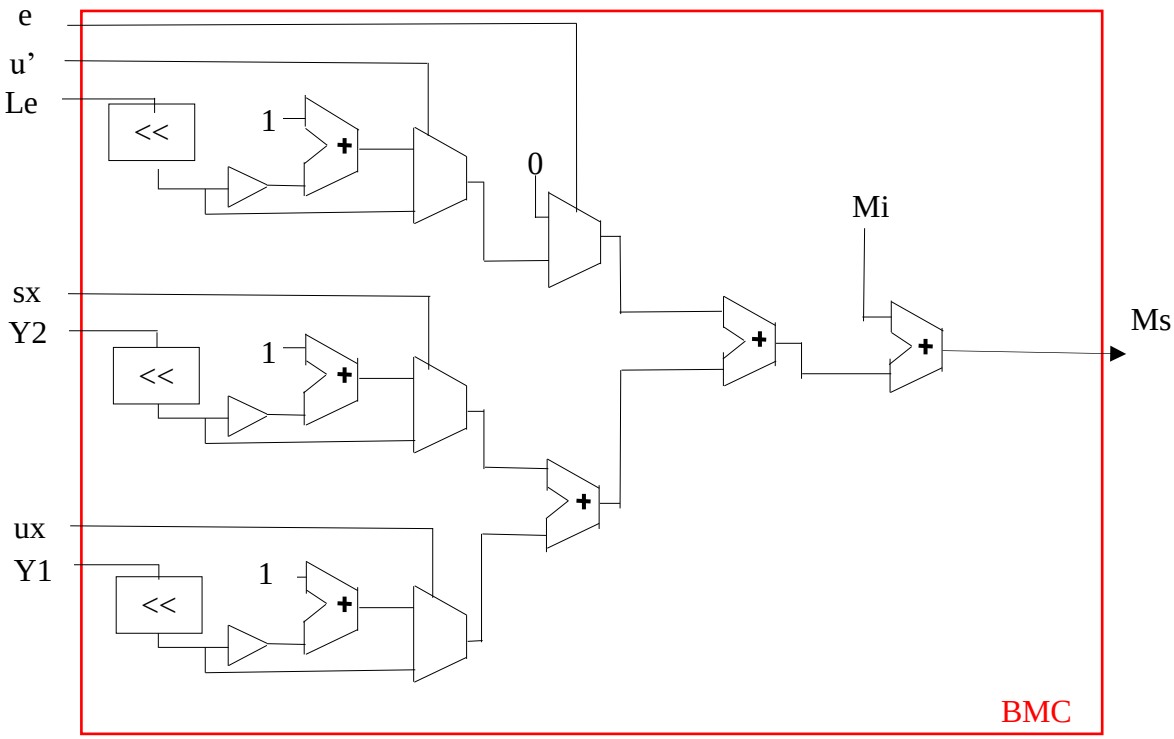

Figure 5. 10 : schema synoptique bloc BMC (Branch Metric Calculator)

- **Bloc PMEc :** est constitué de 2 blocs de BMC (branche métrique calculateur) qui calculent les métriques des deux chemins arrivant a un nœud qui représente un état. Ce nous donne la décision dure et compare entre les deux métriques arrivants et choisi ce la métrique pour le nœud. La figure 5.11 nous donne un schéma illustrateur :

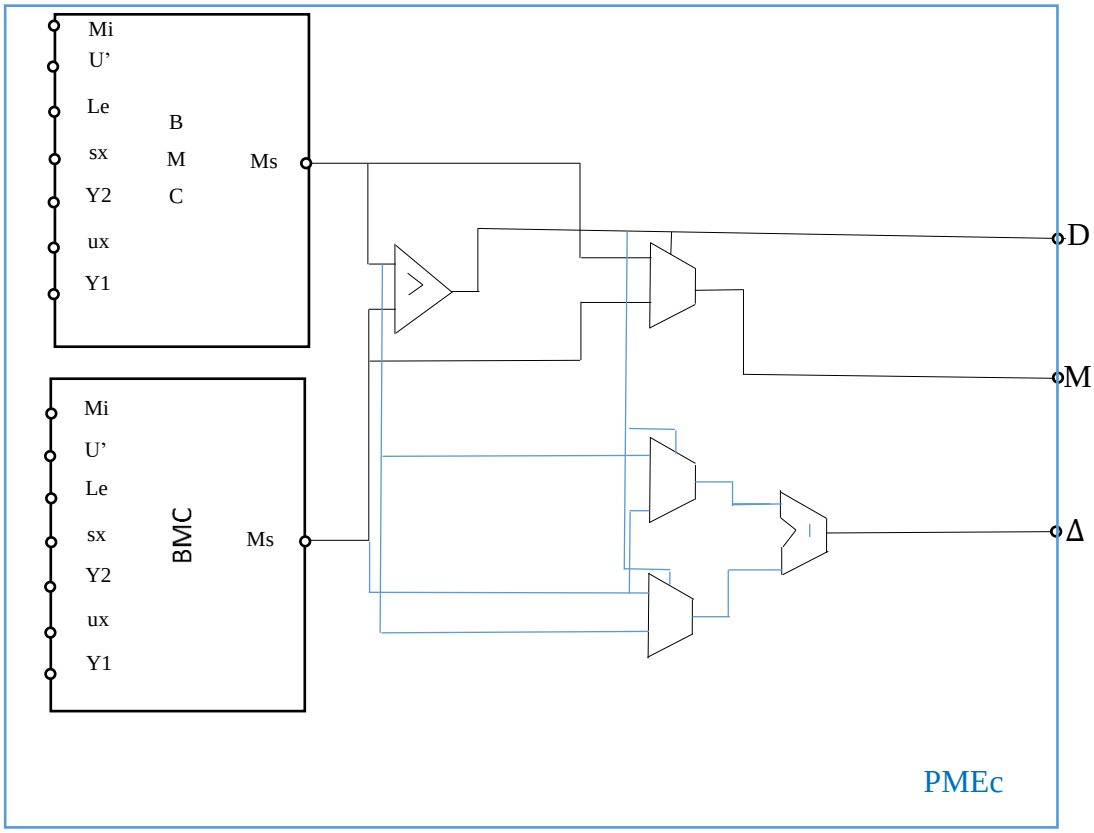

Figure 5. 11: schéma synoptique du bloc PMEc

- Un bloc compare entre les facteurs de fiabilité et sélectionne le maximum dans chaque étage temporel. Ce qui permet de pondères une des décisions.
- Un bloc calcule la fiabilité extrinsèque.

5.6.3.3 Entrelaceur

Dans notre cas, On a utilisé un entrelaceur de type inverseur de longueur 5bit.

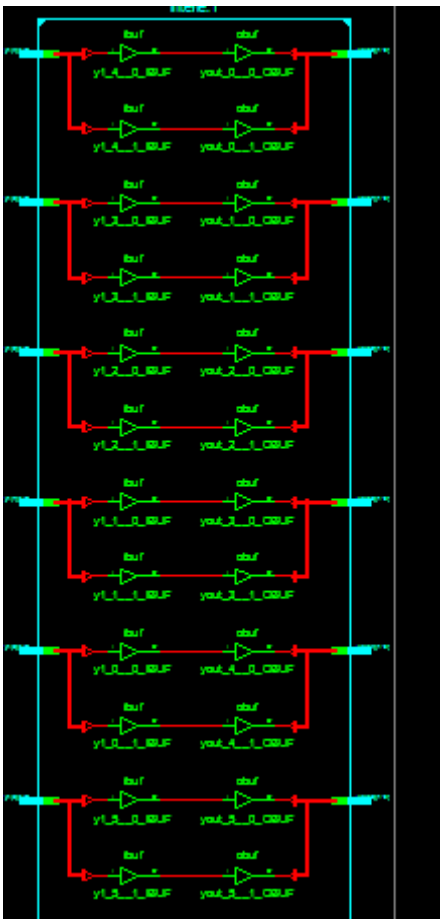

Figure 5. 12 : Technologie schématique de l'entrelaceur

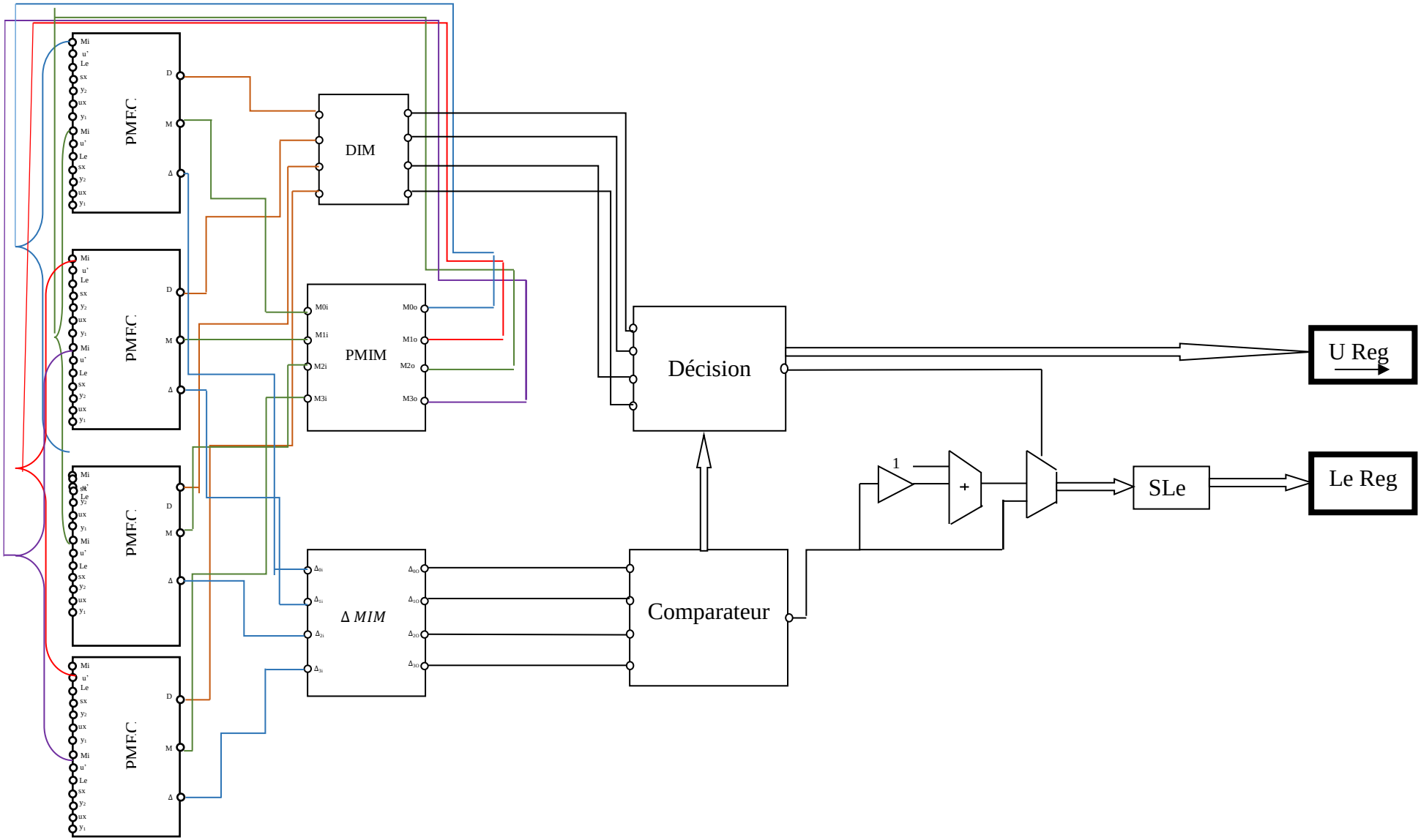

Figure 5. 13 :Schéma synoptique du bloc SOVA

5.7 Implémentation

Le fonctionnement du turbo-décodeur est régi par une machine d'état ci-après dont le rôle est d'établir le bon ordonnancement du processus de décodage itératif décrit dons le Chapitre 4.

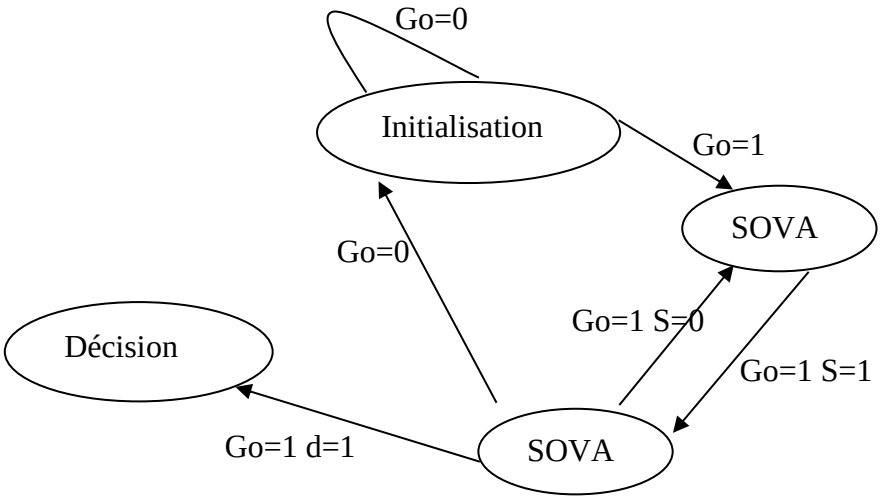

Figure 5. 14: Machine d'état du turbo décodeur itératif SOVA

5.8 Turbo-décodeur à base de SOVA

L'architecture du Turbo-Décodeur étudiée est donnée dans la figure 4.6 .

1-<u>Simulation</u>

Considérons le message d'entrée u={0,1,1,0,1} , En choisissant l'entrelaceur (longueur L=5) « inverse », la séquence d'entrée permutée devient : I{u}={1,0,1,1,0}.

turbo codeur génère les sorties suivantes :

x1={1,0,1,1,0,1,0},x2={0,1,0,0,0,1,0}et x3={1,1,0,0,1,1,0}

Après modulation, les séquences codées deviennent :

x1={+1,-1,+1,+1,-1,+1,-1},x2={-1,+1,-1,-1,-1,+1,-1} et x3={+1,+1,-1,-1,+1,+1,-1}.

Supposons qu'au niveau du récepteur, Les séquences reçues soient :

Y1={+1,-1,+1,+1,-1,+1,-1},Y2={0,+1,-1,-1,-1,+1,-1} et Y3={0,+1,-1,-1,+1,+1,-1}.

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

En prenant Eb/N0=1 et L_c=4 :

L_cy$_1$={4,-4,4,4,-4,4,-4},L_cy2={0,4,-4,-4,-4,4,-4} et L_cy3={0,4,-4,-4,4,4,-4}.

Le bloc décodeur SOVA1 aura pour entrée :

$I^{-1}L_c$cy$_1$,L_cy$_2$et L_e(u'), $I^{-1}L_c$y$_1$ l'entrée systématique entrelacée

Ces entrées sont donc:

$I^{-1}L_c$y$_1$={-4,4,4,-4,4,4,-4},L_cy2={0,4,-4,-4,-4,4,-4} et L_{e2}(u')={0,0,0,0,0,0,0}.

Les Métriques sont calculées par le bloc SOVA1 comme illustré dans la figure 5.15.

```
d2 =

     0     0    -4    12     4    44    12
     0     0   -12     4    -4    20    20
     0     0    -4    28     4    28    28
     0     0     4     4    12    20    20


d =

     4    -4    -4     4    12     4    52
     0    -4    20    -4    36    12    28
    -4    12    -4   -12    12    20    36
     0    -4     4    -4    20    12    28
```

Figure 5. 15: Calcul de Métrique par SOVA1 (simulation MATLAB).

d2 et d sont les métriques des deux chemine existants.

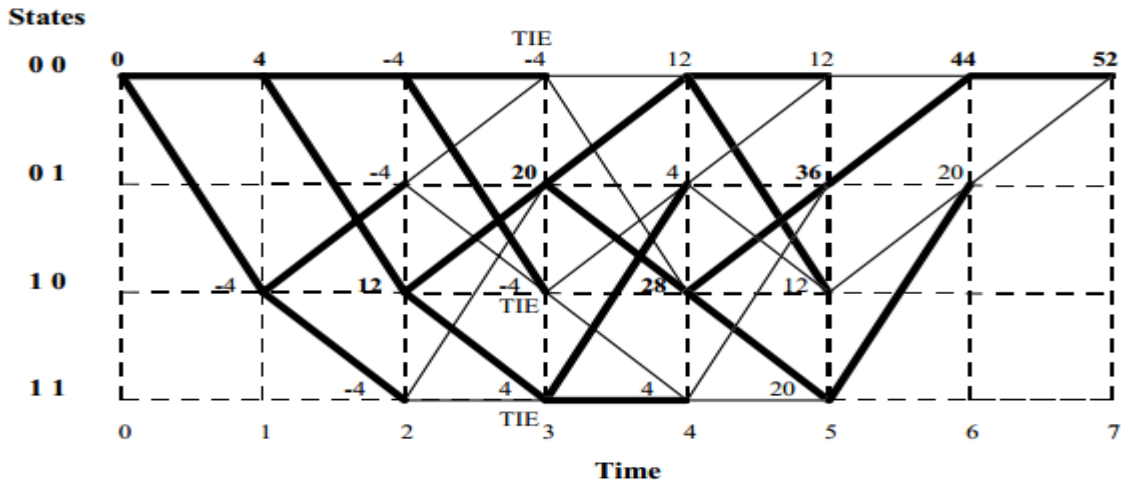

Figure 5. 16: Diagramme des treillis.

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7
Etat 00	0	4	-4	-4/-4	4/12	12/4	4/44	52/12
Etat 01			-4	20/-12	-4/4	36/-4	12/20	
Etat 10		-4	12	-4/-4	-12/28	12/4		
Etat 11			-4	4/4	-4/4	20/12		

Tableau 5. 1 : Calcule de Métrique par SOVA1

Le calcul Comparaison entre les valeurs de fiabilité donne :

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7
Etat 00	0	20	8	0	4	4	20	20
Etat 01		0	12	16	4	20	4	
Etat 10		12	24	0	20	8		
Etat 11		0	12	0	4	4		

Tableau 5. 2 :Le calcul de la fiabilité

Nous déduisons les décisions dures et les décisions souples dans la tableau 5.4

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7
Etat 00	0	-1	-1	1	1	-1	1	-1
Etat 01		-1	1	1	-1	1	-1	1
Etat 10		1	1	1	-1	1	-1	1
Etat 11		-1	-1	1	1	-1	1	-1

Tableau 5. 3 : Les décisions dures et les décisions souples.

Ce qui nous donne les résultats suivants : -1/00,1/10,1/01,-1/10,1/01,1/00,-1/00

Δ={20,24,16,20,20,20,20}

$L_1(u') = I^{-1} y1 \cdot \Delta$ (5. 1

Et la fiabilités extrinsèque $L_{e1}(u') = L_1(u') - I^{-1} L_c y1 - L_{e1}(u')$ (5. 2

= {-16,20,12,-16 ,16,16,-16}

La simulation fonctionnelle sur Modelsim confirme ces résultats :

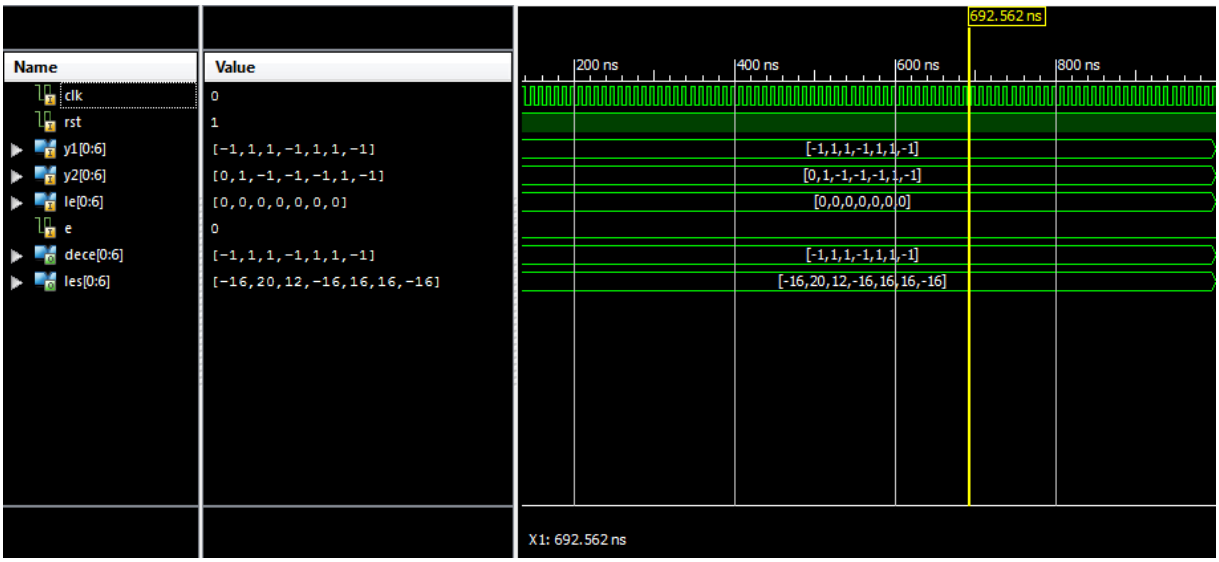

Figure 5. 17 : Simulation fonctionnel sur Modelsim du Bloc SOVA1.

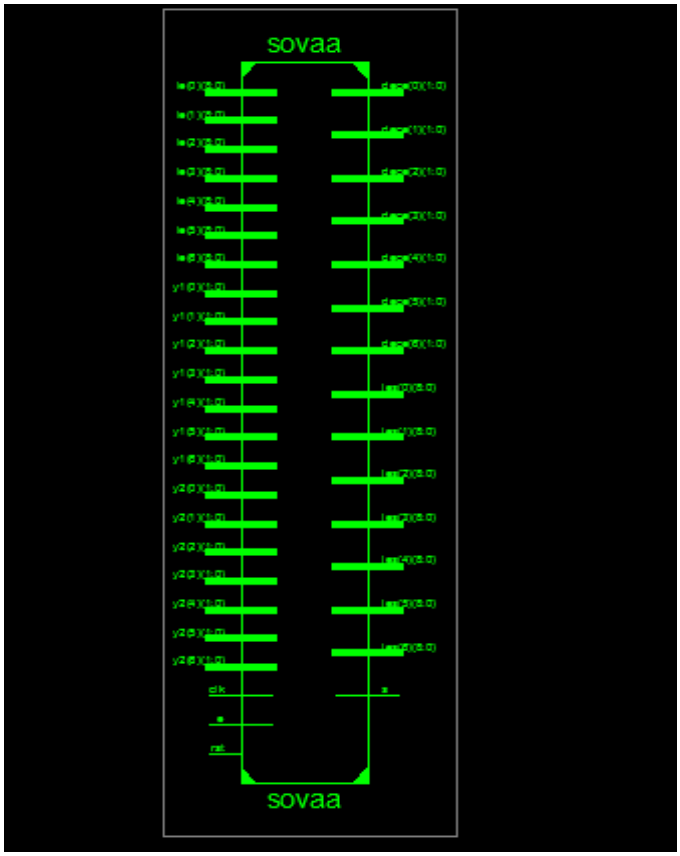

Figure 5. 18: RTL schématique du Bloc SOVA1

De même pour le bloc SOVA2 utilisé pour décode RSC2r, le bloc SOVA2 a comme entrées L_cy1,L_cy3etIL_{e1}(u').

Les entrés sont :

Lcy1={4,-4,4,4,-4,4,-4}, Lcy3={0,4,-4,-4,4,4,-4} et L_{e1} (u')={-16,20,12,-16 ,16,16,-16}

On a obtenu les résultats suivant :

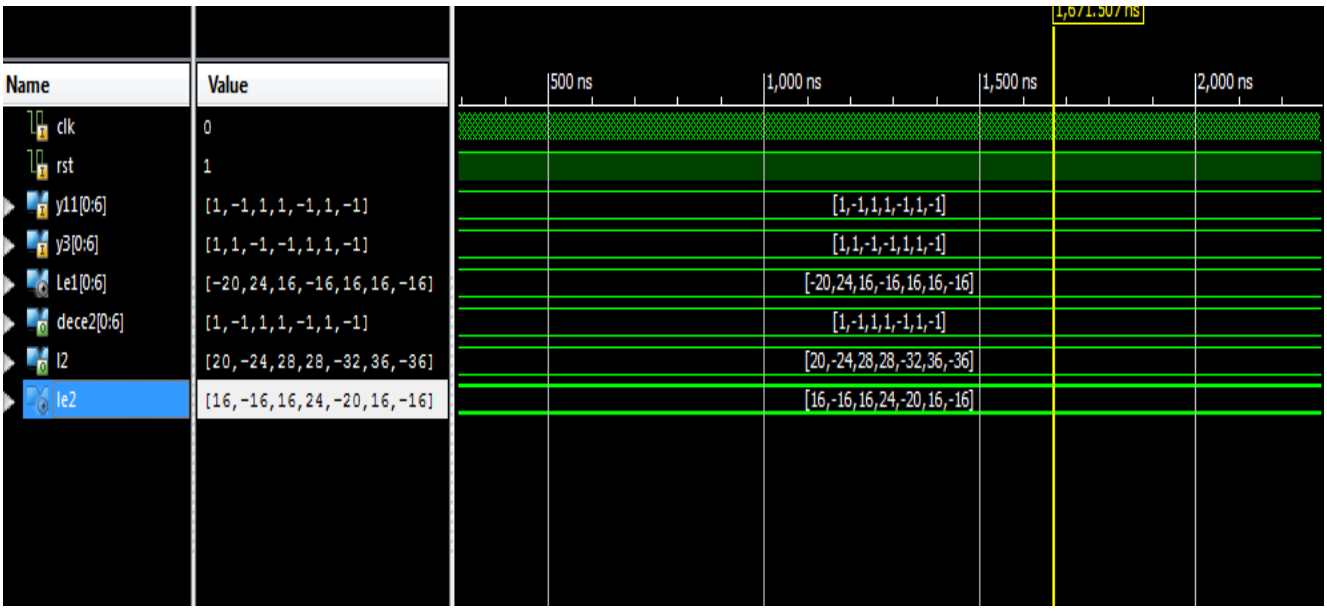

Figure 5. 19: Simulation fonctionnelle sur Modelsim du Bloc SOVA2

La séquence entrelacée I{u'}={+1,-1,+1,+1,-1,+1,-1} après dés-entrelacement, nous obtenons uu{-1,+1,+1,-1,+1,+1,-1}, après suppression des bits de terminaison, les 5 bit restants constitueront l'information utile, ce qui est confirmé par le résultat illustré dans Fig.5.20 .

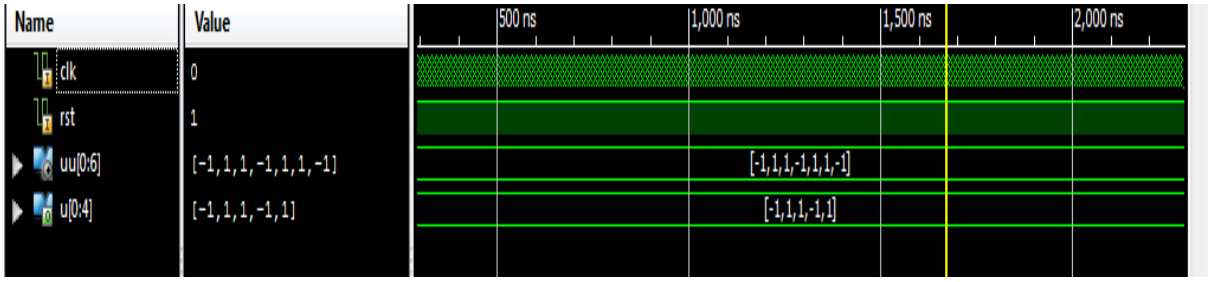

Figure 5. 20 : Résultat final.

Nous avons donc récupéré les résultats désirés avec une correction des erreurs. Après démodulation on obtient u={0,1,1,0,1}. Avec les résulta est donne dans la figure 5.21

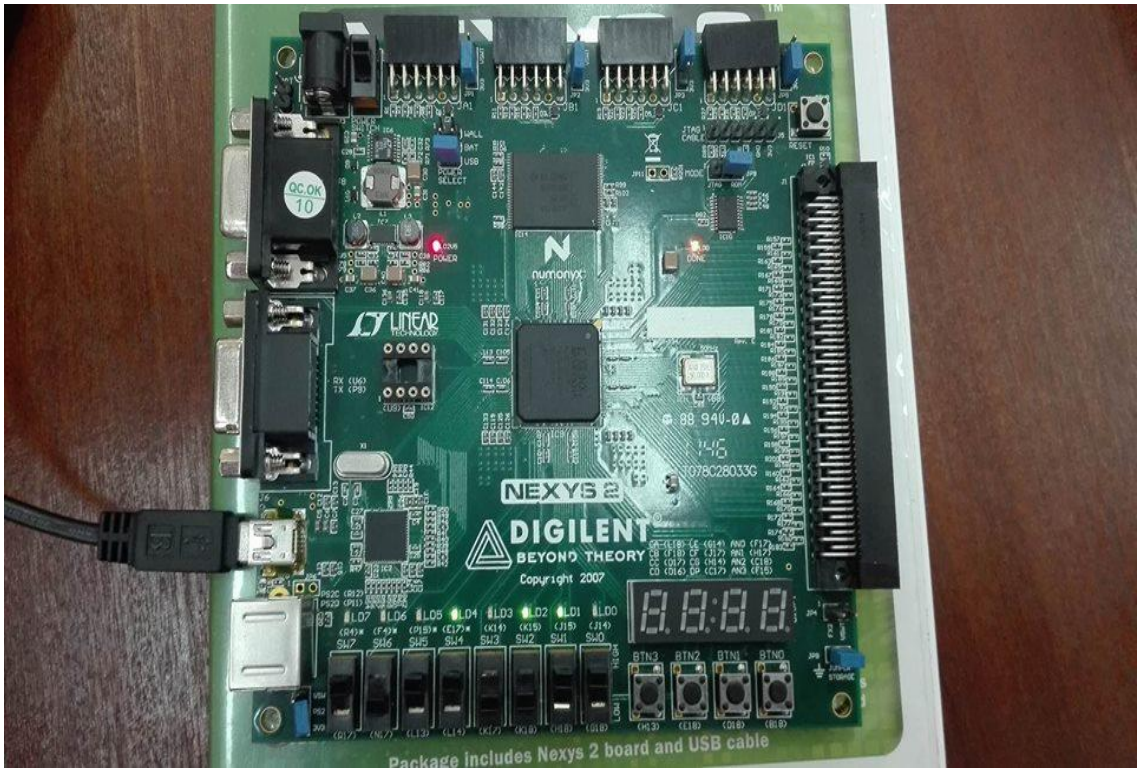

Figure 5. 21 : Résultat sur la carte

Ci-dessous le schéma RTL et le rapport de synthèse généré par ISE du Décodeur itératif à base de SOVA sont illustrés.

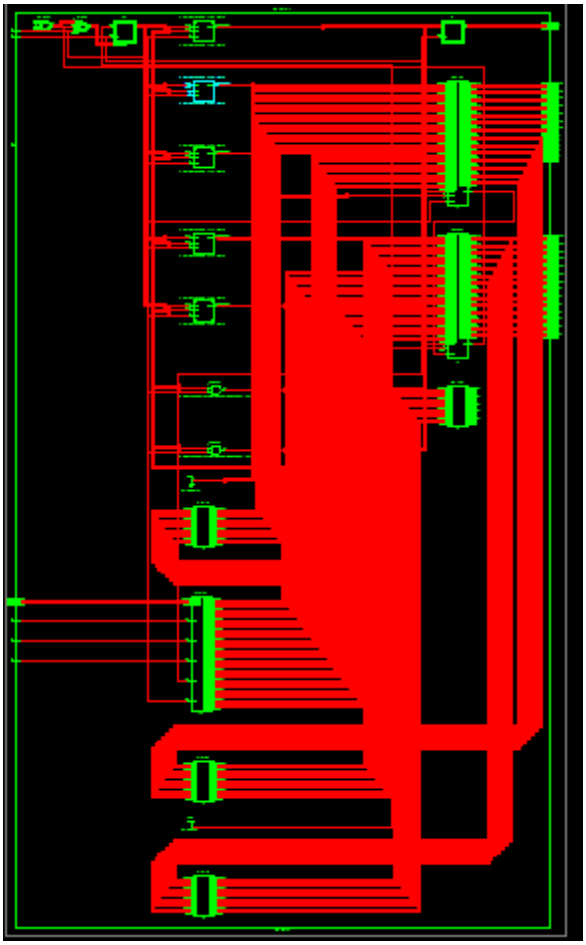

Figure 5. 22 : schéma RTL du Décodeur itératif à base de SOVA.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	406	9,312	4%	
Number used as Flip Flops	181			
Number used as Latches	225			
Number of 4 input LUTs	896	9,312	9%	
Number of occupied Slices	629	4,656	13%	
Number of Slices containing only related logic	629	629	100%	
Number of Slices containing unrelated logic	0	629	0%	
Total Number of 4 input LUTs	1,008	9,312	10%	
Number used as logic	896			
Number used as a route-thru	112			
Number of bonded IOBs	120	232	51%	
IOB Latches	82			
Number of BUFGMUXs	3	24	12%	
Average Fanout of Non-Clock Nets	2.96			

Figure 5. 23 : Rapport de synthèse du Décodeur itératif à base de SOVA

5.9 Conclusion

Ce chapitre décrit l'implémentation sous Xilinx Ise d'un turbo décodeur Sova sur la carte Nexys2 embarquant le FPGA spartan 3E. Nous avons, tout d'abord, effectué les simulations nécessaires pour la validation du fonctionnement des algorithmes de codage et décodage Turbo. L'étape de modélisation a permis de coder en Vhdl les différents éléments (bloc) architecturaux des circuits de codage décodage Turbo. La simulation fonctionnelle, à l'aide de Modelsim, a confirmé les résultats obtenus par simulation algorithmique.

La synthèse logique et l'implémentation sous Ise 14.7 s'est terminé par production di fichier binaire (bil file) représentant le circuit turbo-décodeur.

La projection du bit file sur le support reconfigurable a montré que le turbo-décodeur récupéré bien le message codé tout en détectant et en corrigeant les erreurs.

Conclusion Général

L'objectif de ce mémoire est l'implémentation d'un turbo décodeur SOVA sur circuit reconfigurable. Nous avons commencé par l'étude de la problématique de codage canal dans la chaine de communication où nous avons montré que les codes turbo constituent une parmi les quelques solutions qui permettent d'atteindre les objectifs de détection/correction des erreurs de transmission tout en garantissant des débits très élevés proche de la limite de Shannon.

Nous avons également passé en revue les différentes solutions techniques candidates pour figurer dans les circuits de codage/décodage Turbo. Le cahier de charge ayant imposé l'algorithme SOVA comme module de décodage de base du Turbo-décodeur, nous l'avons tout naturellement étudié sous toutes ses facettes, et nous l'avons utilisé en association avec le concept de décodage itératif pour réaliser notre Turbo décodeur. Les résultats de simulation exposés dans le chapitre 5 et l'implémentation réalisée sur la carte Nexys2 et décrite au même chapitre ont montré l'efficacité du décodeur Turbo et sa simplicité comparée à codeurs turbo utilisant d'autres procédés de décodage, Map par Exemple.

Ce travail est susceptible d'être amélioré et enrichi notamment

- En considérant de réaliser les codes turbo pour les différents standard de communication et faire une comparaison avec les Codecs actuellement utilisé.
- En appliquant le concept Turbo pour le décodage des signaux de transmission non binaires

Bibliographie

[1]. A.Dinu, J.Barral, «Codes correcteurs d'erreurs: implémentation des turbo codes ».

[2]. Berrou, Glavieux et Thitirnajshima, «Near shannon limit error correcting coding and decoding: turbo codes », IEEE transmission and communication, proceedings of patents, Geneva, Switzerland, p.1064-1070, May 1993.

[3]. K. Lajnef, «Etude des performances des codes turbo », mémoire de maitrise, école polytechnique de Montréal, Juin 2001.

[4] J. G. Proakis. «Digital Communication». 5 edition. McGraw Hill, 2008.

[5] D. Le Ruyet. «Theorie de l'information, Codage Source-Canal». École Supérieure de Conception et de Production industrielles, France. Janvier 2007.

[6] A. Viterbi. «Error bounds for convolutional codes and asymptotically optimum decoding algorithm». In : IEEE Transactions on Information Theory Avril. 1967.

[7] A. Viterbi et O. K. Omura. «Principales of Digital Communications and Coding». In : McGraw Hill 1978.

[8] A. Galvieux. «Codage de canal des bases théoriques aux turbocodes». Lavoisier, 2005.

[9] C. Berrou, A. Glavieux et P. Thitimajshima. «Near shannon limit error-correcting coding and decoding». In : IEEE International Conference on Communications 2 mai 1993.

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

[11] Adrian S. Barbuleseu, « Iterative Decoding to Turbo Codes and other concatenated codes» Ph.D Dissertation, Uni of south Australia,Feb 1996

[12] J. J. Boutros, « Les turbo codes parallèles et séries, décodage SISO et performance ML », rapport, Oct. 1998

[13] S. Dolinar and D. Divsalar, «Weight distributions for turbo codes using random and nonrandom permutations », JPL TDA Progress Report 42-122,Aug. 1995

[14] O. Y. Takeshita and D. J. Costello, Jr. «New deterministic interleaver designs for turbo codes» IEEE Trans. Inform. Theory, 1988.

[15] S. Dolinar and D. Divsalar, «Weight distributions for turbo codes using random and nonrandom permutations» JPL TDA Progress Report 42-122, Aout. 1995.

[16] B. Christian Schlegel and C. Lance Pérez, «trellis and turbo coding», Wiley-IEEE Press, 2005.

[17] S. Crosier, J. Lodge, P. Guinand, and A. Hunt, «Performance of turbo-codes with relative prime and golden interleaving strategies». Communication Research Centre, Station

[18] G. Proakis, «Digital Communications » Third Edition, MacGraw Hill, 1995.

[19] G. Proakis, M. Salehi, «Communication Systems Engineering, » Prentice-Hall, 1994

[20] P. Robertson, E. Villebrun, and P. Hoeher, « A comparison of optimal and sub -optimal decoding algorithms in the log domain », Proc Fcc. Seattle, USA ,p. 1009-1013, juin 1995.

[21] A. Migan, S. Argentieri « transmission de l'information : les codes convolutifs », Université Pierre & Marie curie, p 12-56 ,2011

[22] L. Papke, P. Robertson and E. Villebrun, « Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme», IEEE Int. Conf. on Commun. p. 102-106, 1996.

[23] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, « Optimal decoding of linear codes for minimizing symbol error rat», IEEE Trans. on Inf, Theory, vol. IT-20,p 284-287, March 1974.

[24] Matthew C. Valenti, «Iterative detection and decoding for wireless communication». manuscript de these PhD, Virginia Polytechnic Institute and State University, (Blacksburg,Virginia), July1999.Disponible sur http://scholar.lib.vt.edr/theses/available/etd-071399-133447.

[25] O. Joerssen, M. Vaupeland H. «Meyer, High-speed VLSI architectures for soft-output Viterbi decoding» Proc. Application Specific Array Processors, (Berkeley, CA), p.373-384,Aout. 1992

[26] J. Vogt, K. Koora, A. Finger and G. Fetweis, «Comparison of different turbo decoder realizations for IM12000 », Global Telecommunications Conf., p 2704-2708, l999.

[27] H. Dawid and H. Meyer, «Real-time algorithms and VLSI architectures for soft output MAP convolutional decoding», 6th IEEE Int. Symp. on Personal, In-door and Mobile Radio Communications, vol. l, p.193-197,SePt. 1995.

[28] Fu-hua Huang «Evaluation of Soft Output Decoding for Turbo Codes» Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University. May 29, 1997 Blacksburg, Virginia.

[29] Antonio F. Mondragón-Torres, R. Krishna Narayanan, Edgar Sánchez-Sinencio, «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.

[30] A. Syed Amjad « Performance analysis of turbo codes over awgn and Rayleigh channels using different interleavers » Thesis submitted in partial fulfillment of the requirements for the degree of master of science in electrical and electronics engineering astern Mediterranean university September, 2001.

[31] Xlinx. « Boards & Kits». Disponible sur : www.Xlinx.com. Consulté le 10 mai 2018.

[32] Architecture interne d'un circuit FPGA _fig1_Figure-1- Disponible sur : https://www.researchgate.net/figure/277474835

[33] DIGILENT. Disponible sur: https://reference.digilentinc.com/reference/programmable-logic/nexys-2/reference manual.