UNIVERSITÉ DE NEUCHÂTEL
INSTITUT DE MICROTECHNIQUE

Contributions au décodage des codes convolutifs utilisés dans les systèmes de communication mobile *UMTS*

THÈSE

Davide Manetti

PRÉSENTÉE À LA FACULTÉ DES SCIENCES
POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

Septembre 2005

Tesi dedicata ai miei nonni,
Renato e Ada, Ugo e Giovanna.

Université de Neuchâtel unine

IMPRIMATUR POUR LA THESE

Contributions au décodage des codes convolutifs utilisés dans les systèmes de communication mobile UMTS

Davide MANETTI

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel,
sur le rapport des membres du jury

Mme F. Vatta (Trieste I),
MM. F. Pellandini (directeur de thèse),
P.-A. Farine, M. Ansorge,
A. Heubi (AMI SemiconductorMarin)

autorise l'impression de la présente thèse.

Neuchâtel, le 26 septembre 2005

La doyenne:

M. Rahier

Faculté des Sciences
■ Rue Emile-Argand 11 ■ CP 2 ■ CH-2007 Neuchâtel
■ Téléphone : +41 32 718 21 00 ■ Fax : +41 32 718 21 03 ■ E-mail : secretariat.sciences@unine.ch ■ www.unine.ch

Résumé

La communication numérique mobile est en forte expansion. Ainsi, le codage de canal est devenu un élément indispensable puisqu'il permet d'assurer la protection de l'information transmise, en contrôlant les erreurs de transmission. Le codage convolutif est l'une des techniques principales de codage de canal.

L'étude présentée ici propose des méthodes améliorées pour réaliser le décodage des codes convolutifs. Ces méthodes sont avantageuses du point de vue de la réduction de la complexité de calcul, de la qualité du décodage et des possibilités offertes au niveau de la réalisation matérielle.
L'intérêt d'une telle étude est apparu suite aux innovations introduites par le nouveau standard *UMTS*. Du point de vue du codage de canal, ce standard emploie des codeurs convolutifs possédant des longueurs de contrainte supérieures aux codes des actuels standards de communication, afin d'améliorer la protection contre les erreurs de transmission. Par suite de cette modification, les réalisations software et hardware actuellement utilisées ne peuvent pas être employées pour le décodage des codes du nouveau standard *UMTS*. En plus, l'importante longueur de contrainte de ces codes accroît la complexité de calcul nécessaire au décodage classique de chaque symbole protégé. D'autre part, l'élargissement de la gamme des débits de transmission de l'*UMTS* implique une forte variabilité des ressources de calcul demandées par l'opération entière de décodage.

La première partie de ce rapport se concentre sur les méthodes destinées à une implantation software. En analysant à la fois les potentialités du codage de canal établi par le standard *UMTS* et l'état de l'art relatif aux principaux algorithmes de décodage convolutif, des méthodes alternatives à la solution de décodage classique (l'algorithme de Viterbi) sont proposées. Parmi les solutions étudiées, on trouve les méthodes itératives dénommées *List Viterbi Algorithm* et *List Decoding intégrant la validation CRC*. Ces deux méthodes exploitent notamment les informations fournies par l'enchaînement d'un code convolutif avec un code en bloc. Cet enchaînement est l'une des structures de protection définies par le standard *UMTS*. Par rapport à l'algorithme de Viterbi, ces méthodes itératives possèdent des propriétés intéressantes du point de vue de la qualité de protection de l'information et de la complexité de calcul. Afin d'évaluer et de comparer ces propriétés, chaque méthode a été

décrite au moyen d'une extension du langage *ANSI-C* en virgule fixe, prenant en considération les particularités opérationnelles des processeurs de traitement numérique du signal ('*DSP*').

La seconde partie de ce rapport décrit la réalisation matérielle d'une méthode de décodage, en s'appuyant sur l'algorithme de Viterbi. La conception et la synthèse des divers modules constituant l'algorithme sont systématiquement présentées.
Parmi les solutions proposées et analysées dans cette seconde partie, une méthode alternative a été étudiée, qui facilite l'initialisation de l'opération de prise de décision du bit d'information. Cette variante prévoit le décodage d'un chemin quelconque, pour autant qu'il y ait une distance suffisante entre le niveau de profondeur atteint par le chemin et celui du bit d'information. Une stratégie innovatrice pour l'exécution de l'opération de prise de décision est ensuite proposée. Cette stratégie prévoit une réalisation sous forme pipeline, ce qui augmente le débit de traitement du système. Enfin, l'analyse de l'opération de prise de décision présente les avantages d'une stratégie de prise de décision exploitant la redondance des opérations de décodage. La technologie *UMC25-0.25µm* est utilisée pour la synthèse logique des diverses solutions.

Ce rapport de thèse se conclut par la comparaison des avantages pratiques de l'utilisation des solutions software et hardware présentées au cours de ce travail. Les aspects considérés sont la surface de silicium nécessaire aux circuits, la rapidité de traitement et la dissipation d'énergie pour chaque symbole traité.

Définitions

Cette section définit les symboles, les traductions utilisées (glossaire), les fonctions mathématiques et les abréviations utilisées dans ce rapport de thèse.

Symboles

Sauf indication contraire lors de l'utilisation du symbole, la signification des symboles suivants est:

b:	nombre de bits formant le symbole d'information à l'entrée du codeur de canal.
B:	nombre de symboles d'information contenus dans le message à protéger.
C:	paramètre de la fonction de métrique utilisée dans la méthode de décodage séquentielle.
d_{free}:	distance libre d'un code convolutif.
d_j:	distance de Hamming existant entre une séquence de symboles j et la séquence contenant seulement des symboles '0'.
$d_{i,j}$:	distance de Hamming qui sépare les séquences de symboles i et j.
d_{min}:	distance minimale d'un code en bloc.
d_{mm}:	distance maximale entre les valeurs de métriques cumulées.
e:	nombre d'erreurs de transmission affectant la séquence de symboles reçus.
E_s :	énergie de transmission pour chaque symbole du message codé.
E_b:	énergie de transmission pour chaque bit d'information du message.
G_i :	fonction génératrice des codes convolutifs.
G_{kxn}:	matrice génératrice des codes en bloc.
H_{kxn}:	matrice de parité des codes en bloc.
Info Bit$_i$:	i-ème bit du message.
Info Bit$_{1x.i}$:	message formé par les bits d'information *Info Bit*$_0$, *Info Bit*$_1$, …, *Info Bit*$_i$.
k:	nombre de bits formant le bloc d'entrée traité par un code en bloc.

K:	longueur de contrainte d'un code convolutif.
L:	nombre des chemins candidats retenus à chaque itération de la méthode *List Decoding*
m:	m-ème état d'un processus de Markov.
M:	nombre d'états différents du processus de Markov.
n:	nombre de bits de sorties de l'opération de codage de canal.
np:	niveau de profondeur de la représentation graphique du codage convolutif (diagramme en arbre et en treillis).
N_0:	densité spectrale du bruit blanc gaussien.
P_e:	probabilité de croisement du model de canal *Binary symmetric channel*.
r_t:	t-ème observation particulière, formée par les n symboles $[y_{1,t} \ldots y_{n,t}]$ reçus et ayant été générés par le codage du t-ème symbole du message.
R:	séquence de symboles reçue par un canal discret.
R_c:	rendement du code (taux de codage [(*)]) $R_c = b/n$.
s_t :	état du processus de Markov au temps t (profondeur t dans l'arbre/treillis de recherche).
S:	séquence complète des états s_t du processus de Markov.
S_i:	séquence complète i.
t:	aspect temporel dans les représentations graphiques du codage convolutif.
$T(D,N,J)$:	fonction de transfert d'un code convolutif.
x_t:	sortie générée par le changement d'états du processus de Markov du temps $t-1$ au temps t.
$\boldsymbol{x}_{1 \times i}$:	vecteur-ligne représentant le message codé formé par les bits $x_0, x_1, \ldots, x_{i-1}$.
X:	séquence de sortie complète générée par le codeur.
$y_{i,t}$:	i-ème symbole reçu ayant été généré par le t-ème symbole du message.
w_i:	*poids* du message.

Glossaire

arbre de recherche	*Search Tree*
bits de terminaison	*Tail Bits*
canal de type sans mémoire	*Memoryless Channel*

(*) Dans certaines publications le rapport R_c=b/n est également appelé *taux de codage*, même simplement *taux*. Ce document utilise la nomenclature plus répandue "*rendement du code*".

capacité du canal	*Channel Capacity*
chemin survivant	*Survivor Path*
code concaténé de manière séquentielle	*Serially Concatenated Code*
code en bloc	*Bloc Code*
codeur extérieur	*Outer Encoder*
codeur intérieur	*Inner Encoder*
croissance minimale de la métrique	*Minimum Distance Growth*
décision ferme	*Hard Decision*
décision souple ou pondérée	*Soft Decision*
décodeur extérieur	*Outer Decoder*
décodeur intérieur	*Inner Decoder*
distance de Hamming	*Hamming Distance*
erreurs non détectées	*Undetected Errors*
étalement	*Spreading*
gain de codage	*Coding Gain*
longueur de contrainte	*Constraint Length*
maximum de vraisemblance	*Maximum Likelihood*
poids d'un message codé	*Weight of a code word*
processus de Markov	*Markov Process*
protection différenciée contre les erreurs	*Unequal Error Protection*
quantité d'information	*Information quantity*
recouvrement d'erreurs	*Error Concealment*
récursion arrière	*Backward Recursion*
récursion avant	*Forward Recursion*
rendement du code	*Code Rate*
structure en treillis	*Trellis*
théorème de la non-optimalité	*Theorem of non-optimality path*
tri	*Sorting*

Fonctions mathématiques

$A \propto B$	la valeur de 'A' est proportionnelle à celle de 'B'.
ln	fonction logarithme naturel.
log	fonction logarithmique, sans définition de la base.
max	fonction de maximisation.
min	fonction de minimisation.
Pr[A]	probabilité de l'événement 'A'.
Pr[A | B]	probabilité conditionnelle de l'événement 'A' étant donné l'événement 'B'.

Pr[A, B]	probabilité jointe des événements 'A' et 'B'.
$\lfloor A \rfloor$	arrondissement vers -∞.
$\lceil A \rceil$	arrondissement vers +∞.
$\arg\left\{\max_A F(A)\right\}$	élément de l'ensemble 'A' qui maximise la fonction 'F'.

Abréviations

Abréviations mathématiques

$BestMetr_t$	métrique cumulée du chemin le plus probable au niveau de profondeur *t*.
MaxContribBranche	contribution maximale de la métrique de branche.
MinMetrCroissance	augmentation minimale de la métrique cumulée d'un chemin incorrect, normalisée par le nombre de symboles reçus.
ValMaxMetrique	valeur maximale de la représentation numérique de la métrique cumulée.

Institutions de standardisation

3GPP	*3rd Generation Partnership Project* (projet de partenariat pour la *3G*).
ANSI	*American National Standards Institute* (organisme privé de normalisation américain).
ETSI	Institut européen des normes de télécommunication.
ISO	Organisation internationale de normalisation.
ITU	Union internationale des télécommunications.
ITU-R	Secteur des radiocommunications de l'union internationale des télécommunications.

Abréviations techniques

2G	Second Generation of Mobile Communication.
3G	Third Generation of Mobile Communication.
ACS	'Add-compare-select' operation in the Viterbi Algorithm.
ADC	Analog to Digital Converter.
AFC	Automatic Frequency Control.
AGC	Automatic Gain Control.

AMR	Adaptive Multi-Rate (Speech Coder).
AMR-NB	Narrow-Band (300-3400 Hz) Adaptive Multi-Rate (Speech Coder).
AMR-WB	Wide-Band (50-7000 Hz) Adaptive Multi-Rate (Speech Coder).
ARQ	Automatic Repeat Request.
ASIC	Application Specific Integrated Circuit.
AWGN	Additive White Gaussian Noise.
BER	Bit Error Rate.
CCTrCH	Coded Composite Transport Channel.
CMOS	Complementary Metal Oxide Semiconductor.
CRC	Cyclic Redundancy Code.
DAC	Digital to Analog Converter.
DCH	Dedicated Channel.
DLL	Delay Locked Loop.
DPE	Multipath Search ou Delay Profile Estimation.
DSP	Digital Signal Processor.
DTX	Discontinuous Transmission.
FDD	Frequency Division Duplex.
FEC	Forward Error Correction.
FER	Frame Error Rate.
FPGA	Field Programmable Gate Array.
GF	Galois Field.
GSM	Global System for Mobile Communications.
HDL	Hardware Description Language.
IF	Intermediate Frequency.
kbps	Kilo Bit Per Second.
LCC	Loosely Coupled Coprocessor.
LSB	Least Significant Bit.
MAC	Medium Access Control.
MIPS	Millions of Instructions per Second .
MMuACCs	Millions of 'Multiply-Accumulation' Operations per Second.
MOPS	Millions of Operations per Second.
MRC	Maximal Ratio Combination.
MSB	Most Significant Bit.
MuACC	'Multiply-Accumulation' Operation.
OCT	Octal (base).
PAM	Pulse Amplitude Modulation.
PhCH	Physical Channel.
RRC	Radio Resource Control.

RX	Receiver.
SF	Spreading Factor.
SIMD	Single-Instruction Multiple-Data.
SNR	Signal to Noise Ratio.
SoC	System on Chip.
RF	Radio Frequency.
TB	Trace back operation in the Viterbi Algorithm.
TCC	Tightly Coupled Coprocessor.
TFCI	Transport Format Combination Indicator.
TrCH	Transport Channel.
TTI	Transmission Time Interval.
TX	Transmitter.
UE	User Equipment.
UEP	Unequal Error Protection.
UMTS	Universal Mobile Telecommunications System.
UTRA	Universal Terrestrial Radio Access.
UTRAN	Universal Terrestrial Radio Access Network.
VHDL	Very High-Speed Integrated Circuit Hardware Description Language.
WCDMA	Wideband Code Division Multiple Access.
WCwMOPS	Extended version of the wMOPS measure.
WcwOP	Extended version of the wOP measure.
wMOPS	Millions of weighted Operations per Second (*ITU*/*ETSI*).
wOP	Weighted Operations (*ITU*/*ETSI*).

Table des matières

1 Introduction

Cette thèse étudie la réalisation software et hardware de méthodes pour le décodage de données protégées par un codage convolutif. Les codes convolutifs ainsi que la structure de protection traités sont ceux des standards de transmission numérique sans fil UMTS.

Ce chapitre présente les motivations qui ont mené à ce travail de thèse, ses objectifs ainsi que l'organisation de ce rapport. Finalement, ce chapitre présentera les principales contributions apportées au domaine du décodage convolutif.

1.1 Contexte

Au cours des dix dernières années, nous avons assisté à une augmentation continue de la demande pour des systèmes de transmission numériques fiables [Lust00]. L'explosion de l'échange d'informations, ainsi que les nouvelles possibilités offertes par le traitement numérique du signal, ont accentué cette tendance.
Parmi les critères de développement des récents standards de communication mobile, la qualité de la transmission est devenue un facteur déterminant. En effet, comme les progrès techniques et l'évolution des besoins du marché ont élargi la gamme des applications potentielles (Table 1-1), la qualité est devenue un élément indispensable des actuels systèmes de communication numérique.

La qualité d'une transmission numérique dépend principalement de la probabilité d'erreur des symboles transmis [Thit93]. Cette probabilité étant fonction du rapport *signal sur bruit*, une amélioration de la qualité de transmission peut être envisagée en augmentant la puissance d'émission et en diminuant le facteur de bruit du récepteur. Malheureusement, cette solution implique des coûts énergétiques et technologiques, ce qui en limite sensiblement l'emploi. Le contrôle des erreurs par codage est ainsi indispensable.

Communication	**Personnels**	**Bureautiques**
Voix Images Messages Accès aux réseaux d'informations (Internet) Modem …	Informations/Nouvelles Gestion de budget Clé électronique Passeport/*ID* …	Courrier électronique Agenda synchronisé Dictionnaire évolutif Fax Divers outils/programmes …
Négociation	**Monitorage médical**	**Sécurité**
Tele-banking Réservation de restaurant/hôtel/spectacles Achat/Vente Transfert d'argent Paiements …	Surveillance médicale Diagnostic à distance Consultation à distance	Surveillance à distance Alarme *GPS* Appel d'urgence/ demande d'aide …
Voyages	**Distraction**	**…**
Roaming Guide/Carte des villes Informations locales *GPS* Météo Traductions …	Jeux électroniques Photo/Vidéo/Musiques Radio/TV *Chat* …	…

Table 1-1: catégories de services sans fil et exemples correspondants, qui sont envisageables dans un proche futur [Gath02].

L'utilisation de techniques de traitement numérique du signal, et notamment le codage des informations à transmettre, permet la détection et/ou la correction d'éventuelles erreurs de transmission. Comme ces techniques permettent de contrôler les erreurs induites par le bruit du canal de transmission, elles sont nommées "*codages de canal*". Parmi les principales techniques existantes, le codage en bloc et le codage convolutif sont prédominantes. Le codage convolutif est fortement diffusé dans les systèmes de communication numérique sans fil.

La stratégie de base du codage consiste en l'insertion d'une quantité contrôlée de redondance dans la série d'informations à envoyer, au moyen d'un nouveau codage des informations (Figure 1-1). La procédure de génération de

redondance traite les informations, soit en blocs (codage en bloc), soit de manière continue (codage convolutif). L'addition d'une redondance confère au décodeur la capacité de détection et/ou de correction d'un nombre fini d'erreurs de transmission.

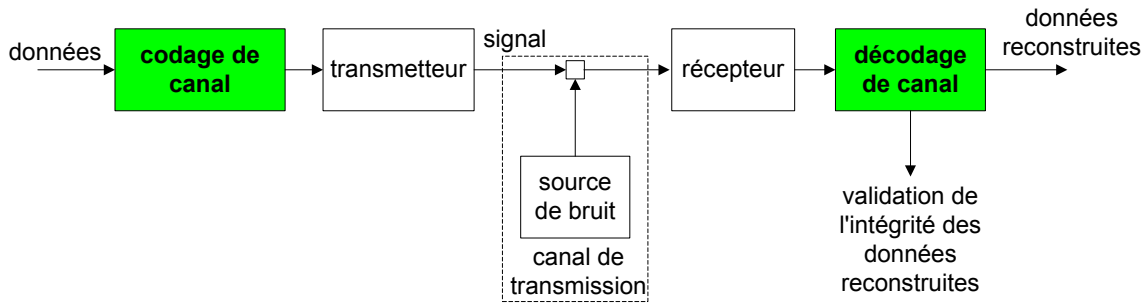

Figure 1-1: positionnement du codage de canal dans les systèmes de communication [Shan49].

Le nombre d'erreurs affectant la transmission des informations dépend du moyen de transmission. Le débit des erreurs et sa distribution temporelle diffèrent si le moyen de transport est une ligne téléphonique, une ligne numérique, un lien satellite ou un canal de communication sans fil.
La qualité de la transmission numérique par réseaux cellulaires sans fil est influencée par des facteurs techniques, tels que les types de modulation et d'antenne des systèmes de communication, ainsi que par des facteurs environnementaux, tels que la dimension et la géométrie de la cellule concernée, son environnement, la distance entre la station de base et l'utilisateur et la mobilité de ce dernier. Pour réduire les effets des erreurs de transmission, les standards *UMTS* prévoient un codage de canal dont le niveau de protection peut être adapté aux exigences des applications.

Il est dès lors évident que l'introduction de techniques de codage de canal provoque une augmentation de la complexité de calcul du traitement numérique du système de communication. L'importance de cette augmentation est fonction du niveau de protection envisagé par l'opération de codage: l'exigence d'une protection plus efficace contre les erreurs de transmission implique l'utilisation de méthodes de codage plus complexes, rendant ainsi les procédures de décodage onéreuses.

1.2 Objectifs de la thèse

L'une des motivations à l'origine de l'établissement des nouveaux standards de communication mobile de la troisième génération ('*3G*') *UMTS* est

l'amélioration de l'efficacité de protection contre les erreurs de transmission. Ces standards répondent à cette exigence par la mise à disposition de techniques de "*codage turbo*" ainsi que de codes convolutifs plus performants que ceux définis par les standards de la seconde génération ('*2G*') [ETSI909] [Ts25212]. L'augmentation du débit de transmission maximal et l'emploi de codes convolutifs plus complexes augmentent sensiblement la complexité de calcul des méthodes de décodage dites *classiques*, utilisées dans les systèmes de la *2G*. Ces méthodes classiques sont basées sur l'algorithme de Viterbi.

L'étude présentée dans le cadre de ce rapport de thèse évalue l'efficacité des implantations software et des réalisations matérielles des méthodes classiques de décodage, dans le nouveau contexte de codage des standards *UMTS*. Des méthodes améliorées ainsi que des solutions intéressantes sont proposées.
La structure de protection des standards *UMTS* permet l'implantation software de méthodes améliorées de décodage, méthodes qui exploitent itérativement et exhaustivement les informations supplémentaires du codage en bloc cyclique. Ces méthodes possèdent des caractéristiques intéressantes du point de vue de la complexité de calcul et de la qualité du décodage.
Le développement d'une réalisation matérielle d'une méthode de décodage convolutif est également analysé. L'utilisation de l'algorithme de Viterbi permet l'étude de solutions alternatives de réalisation. Les aspects considérés sont le débit de traitement et la dissipation d'énergie.
Ce rapport se conclut par une comparaison des potentialités des implantations software et des réalisations matérielles de l'algorithme de Viterbi. L'analyse critique examine le débit de traitement, la dissipation d'énergie et la surface de silicium des circuits. Cette comparaison permet d'identifier les points forts des implantations software et des réalisations matérielles de l'algorithme de Viterbi.

Les implantations software sont validées en utilisant la structure de codage du service de parole *AMR-NB* à 12.2kbps [Ts26071] [Ts26101]. L'analyse de la réalisation matérielle d'un décodeur convolutif est faite sur la base de quatre codages convolutifs possédant différentes caractéristiques de protection (standards *GSM* [ETSI909] et *UMTS* [Ts25212]).

1.3 Organisation du document

Ce rapport de thèse est organisé comme suit. Le Chapitre 2 introduit brièvement le lecteur dans le domaine du codage de canal, en passant de la contribution pionnière donnée par Shannon, aux principes du contrôle des erreurs par codage enchaîné.

La structure et les objectifs de la protection de canal établi par le standard *UMTS* [Ts25212] sont traités dans le troisième chapitre. Le cas de la protection des données du service de parole *AMR-NB* à 12.2 kbps y est illustré à titre d'exemple. Cette structure sera ultérieurement utilisée au Chapitre 6 pour la validation et l'analyse des performances des implantations software des méthodes de décodage les plus intéressantes.

Le quatrième chapitre traite la transmission de données par réseaux cellulaires sans fils. Les facteurs influançant la transmission sont présentés, ainsi que la modélisation du canal de transmission.

En s'appuyant sur les connaissances présentées du Chapitre 2 au Chapitre 4, le Chapitre 5 passe en revue les principes de décodage des codes convolutifs ainsi que les méthodes les plus représentatives de ce codage. Les méthodes présentées sont le "*Décodage séquentiel*", le "*List Decoding*", "*l'Algorithme de Viterbi*", le "*Symbol-by-symbol Maximum A Posteriori Algorithm*", le "*Max-Log-MAP*", le "*Log-MAP*" et le "*Bidirectional Soft Output Viterbi Algorithm*". Le Chapitre 5 se termine par une comparaison et une discussion des caractéristiques algorithmiques et de la réalisation de ces algorithmes.

L'implantation software de méthodes de décodage de codes convolutifs fait l'objet du sixième chapitre. Ce chapitre analyse à la fois les potentialités de la structure de protection établie par le standard *UMTS* ainsi que les caractéristiques des principes de décodage, afin de déterminer les performances envisageables pour une implantation software. Parmi les méthodes étudiées, on trouve non seulement l'algorithme de Viterbi, mais également des méthodes itératives de décodage. Ces méthodes itératives sont une nouvelle réalisation de la méthode *List Viterbi Algorithm*, ainsi que la nouvelle méthode *List Decoding intégrant la validation CRC*.

Le sujet du Chapitre 7 est la réalisation matérielle d'une méthode de décodage convolutif. La méthode choisie est l'algorithme de Viterbi. Les analyses et les solutions proposées se basent sur les résultats de la synthèse logique (outil *Synopsys*, technologie *UMC25-0.25µm* [UMC25]), en décomposant l'algorithme en éléments constituants. Parmi les solutions analysées, on trouve une stratégie novatrice pour l'exécution parallèle de plusieurs opérations de prise de décision, une méthode alternative d'initialisation de cette opération, ainsi qu'une stratégie exploitant la redondance des opérations de reconstruction.

Le Chapitre 8 discute les coûts des solutions software et hardware proposées. Les aspects considérés sont la surface de silicium, la rapidité de traitement et la dissipation d'énergie par symbole traité.

Ce rapport se termine par les conclusions finales de l'étude effectuée, établies dans le Chapitre 9.

1.4 Contributions

Cette section énumère les contributions principales apportées par cette étude des méthodes de décodage. Après une première partie introductive, ce rapport présente à la fois une analyse critique des méthodes de décodage convolutif existantes, la proposition de deux méthodes améliorées de décodage software, l'étude d'alternatives pour la réalisation matérielle de l'algorithme de Viterbi, ainsi qu'une analyse critique des résultats de l'étude.

Une première contribution est fournie en **Chapitre 5**. Elle consiste en la présentation globale des principes de fonctionnement et des propriétés des algorithmes de décodage convolutif, y inclus de l'état de l'art correspondant et de l'analyse critique des algorithmes les plus représentatifs.

Les contributions principales du **Chapitre 6** sont la proposition de deux méthodes améliorées de décodage software exploitant itérativement les informations supplémentaires du codage en bloc *CRC*: la méthode *List Viterbi Algorithm* ('*LVA*') et la méthode '*List Decoding intégrant la validation CRC*'. Les contributions du Chapitre 6 sont:

- Proposition d'une nouvelle réalisation de la méthode *List Viterbi Algorithm*. Cette méthode, qui envisage l'amélioration de la qualité de protection de l'algorithme de Viterbi, a été originalement proposée dans le contexte de codage des standards *GSM*. Dans ce chapitre, cette méthode est revalorisée en proposant une nouvelle réalisation conforme à la structure de codage *UMTS* ainsi qu'à une exécution en temps réel, montrant ainsi sa faisabilité et son efficacité de protection dans ce nouveau contexte de codage.
- Proposition d'une nouvelle méthode de décodage itérative: la méthode '*List Decoding intégrant la validation CRC*'. Une nouvelle réalisation plus efficace de l'algorithme *List Decoding* a permis la proposition de cette méthode de décodage itérative. A la différence de la méthode précédente, cette méthode envisage la réduction de la charge de calcul,

tout en gardant la même qualité de protection que l'algorithme de Viterbi.

- Réalisation et analyse critique de l'implantation software de l'algorithme de Viterbi conforme à la structure de codage *UMTS* ainsi qu'à une exécution en temps réel. Les aspects de l'implantation, ses potentialités et sa complexité de calcul sont présentés.

L'analyse critique des réalisations software présentées dans le Chapitre 6 utilise la structure de codage établie pour le service de parole *AMR-NB* à 12.2kbps, ainsi qu'une stratégie d'évaluation de la complexité de calcul qui a été développée spécifiquement pour ce domaine d'application.

Les contributions du **Chapitre 7** sont la proposition et l'analyse critique de solutions alternatives de réalisation matérielle de l'algorithme de Viterbi:

- Méthode alternative d'initialisation de l'opération de prise de décision du bit d'information. Cette variante prévoit le décodage d'un chemin quelconque, pour autant qu'il y ait une distance suffisante entre le niveau de profondeur atteint par le chemin et celui du bit d'information.
- Stratégie innovatrice pour l'exécution de l'opération de prise de décision. Cette stratégie prévoit une réalisation sous forme pipeline, ce qui augmente le débit de traitement du système.
- Stratégie de prise de décision exploitant la redondance des opérations impliquées dans le décodage du chemin. L'analyse de l'opération de prise de décision présente les avantages d'épargne d'énergie d'une telle stratégie de prise de décision.
- Explication des aspects liés à la réalisation matérielle d'une méthode générale de décodage basée sur l'algorithme de Viterbi.

Les analyses et les solutions proposées se basent sur les résultats obtenus par la synthèse du système de base en utilisant l'outil de synthèse logique "Design Compiler" de Synopsys (technologie UMC25-0.25µm).

La dernière contribution au domaine du décodage convolutif est l'analyse critique des réalisations logicielles et matérielles présentée au **Chapitre 8**. Ce chapitre compare les coûts et les potentialités des deux approches software et hardware, tels que le débit de traitement, la consommation d'énergie et la surface de silicium du circuit. Les potentialités sont estimées sur la base des informations disponibles sur le marché (printemps 2003) des *DSP* et sur la base de la technologie de synthèse *UMC25-0.25µm*.

Références

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grand Bretagne, 2002.

[Lust00] F. Lustenberger, *On the Design of Analog VLSI Iterative Decoders*, Dissertation *ETH* No. 13879, Serie in Signal and Information Processing: Volume 2, Hartung Gorre, Konstanz, Allemagne, novembre 2000.

[Shan49] C. E. Shannon, "Communication in the presence of noise", *Proceeding of the Institute of Radio Engineers* (*IRE*), Vol. 37, janvier 1949, pp. 10-21. Ré-edition: (*Reprinted in*): *Proceeding of the IEEE*, Vol. 86, No.2, février 1998, pp.447-457.

[Thit93] P. Thitimajshima, *Les codes Convolutifs Récursifs Systématiques et leur application à la concaténation parallèle*, Thèse de Doctorat en Electronique, Université de Bretagne Occidentale, France, 1993.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Ts26071] 3GPP, *AMR Speech Codec: General Description*, document 3GPP TS 26.071, version 4.0.0.

[Ts26101] 3GPP, *AMR Speech Codec Frame Structure*, document 3GPP TS 26.101, version 1.6.0.

[UMC25] Virtual Silicon Technology Inc., *Diplomat-25 Standard Cell Library: 0.25µm UMC Process*, rev. 2.1, décembre 1999.

2 Notions de base sur la protection de canal

Ce chapitre introduit les notions fondamentales relatives à la protection des données contre les erreurs de transmission.

La Section 2.1 résume l'article "pionnier" de Shannon qui fixe les bases de la théorie de l'information, en stipulant qu'une transmission avec un taux d'erreurs contrôlable est possible à travers un canal bruité.

La Section 2.2 décrit deux modes de contrôle des erreurs par codage, soit le Automatic Repeat Request (ARQ) et le Forward Error Correction (FEC). A partir de ces notions, la section suivante introduit les principes du codage en bloc linéaire et du codage convolutif. Les caractéristiques de ces codages ainsi que les capacités potentielles de correction d'erreurs sont ensuite discutées.

La Section 2.4 présente un rappel des principes du contrôle des erreurs par codage enchaîné. L'intérêt de l'enchaînement de plusieurs opérations de codage de canal –soit en série soit en parallèle- y est brièvement présenté.

2.1 Article de C. E. Shannon

La théorie de l'information est née en 1948 avec l'article de C. E. Shannon "A Mathematical Theory of Communication" [Draj02]. Cette théorie déterminait les limites de performance des systèmes de communication numérique, et anticipait notablement les besoins pratiques de ce type de communication.

En généralisant le schéma d'un système de communication (Figure 2-1), Shannon présente des notions, qui s'avèreront être à la base de la théorie de

l'information. Dans ce schéma, Shannon identifie cinq éléments principaux [Shan49]:

- La source d'informations: cet élément génère le message à transmettre, message qui appartient à un groupe prédéfini de messages possibles.
- Le transmetteur: cet élément est responsable de la préparation du message de manière à permettre sa transmission.
- Le canal: le canal de transmission entraîne des modifications du signal selon les caractéristiques physiques du media de communication. La modélisation du canal se base sur plusieurs paramètres et éléments, dont certains possèdent des propriétés non-prédictives.
- Le récepteur: la tâche de cet élément est la reconstruction et/ou l'estimation du message original à partir du signal reçu.
- Le destinataire: cet élément constitue l'entité à qui le message est adressé.

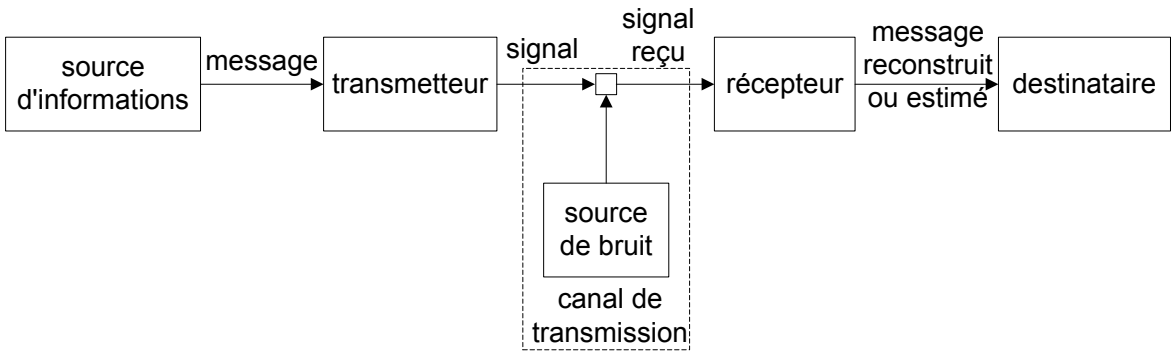

Figure 2-1: modèle générique des systèmes de communication [Shan49].

Parmi les nombreuses informations présentées dans l'article de Shannon, il convient de citer l'importance du seconde Théorème. Ce théoreme montre qu'une communication numérique fiable est possible via un canal de transmission bruité, en recourrant à un système d'encodage suffisamment complexe [Shan49]. Toutefois, Shannon n'indique pas la méthode pour atteindre ce type de communication.

2.2 Contrôle des erreurs par codage

Suite à la publication des articles de Shannon, une stratégie de codage basée sur la génération du signal numérique à transmettre en deux étapes s'est imposée (Figure 2-2). La première étape consiste principalement en l'élimination de la redondance de l'information dans le message ou du moins

en sa réduction. Cette étape est nommée *codage de source*. Dans une deuxième étape, le *codage de canal* insère une redondance contrôlée dans le message afin de permettre la gestion des erreurs de transmission par codage (*Error Control Coding*).

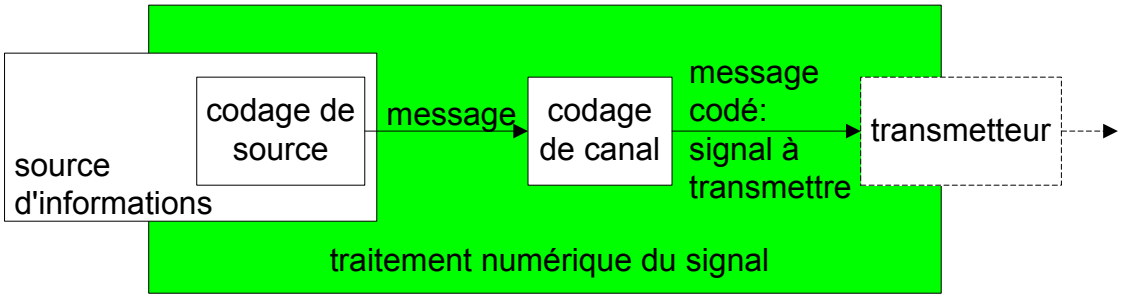

Figure 2-2: principe du contrôle des erreurs par codage.

Encore aujourd'hui, plus de 50 après la publication des articles de Shannon, le principe de dissociation des deux étapes de codage (codage de source et de canal) est toujours appliqué. Toutefois on favorise actuellement l'échange d'informations entre les divers éléments de la chaîne de transmission et de réception.

L'objectif du codage de canal est d'établir un système de contrôle des erreurs par un nouveau codage du message. Ceci se réalise en créant et en insérant une certaine redondance au message. Cette redondance permet au récepteur de détecter, voire de corriger les erreurs de transmission (Figure 2-3) [Karn95].
Naturellement, la procédure de génération de la redondance doit être adaptée aux caractéristiques du support de transmission et doit être connue par le système de décodage.

Les méthodes de contrôle des erreurs se regroupent principalement en deux modes d'utilisation: le *Automatic Repeat Request* (*ARQ*) et le *Forward Error Correction* (*FEC*).
L'objectif du mode *ARQ* est l'ajout d'une petite quantité de redondance au message, de manière à permettre la détection d'éventuelles erreurs de transmission. Dans le cas d'une détection d'erreurs, le décodeur demande la retransmission du message erroné.
Par contre, dans le cas du mode *FEC*, la redondance introduite permet de détecter et corriger au niveau du décodeur un nombre fini d'erreurs. La quantité de redondance nécessaire est naturellement plus grande pour le mode *FEC* que pour le mode *ARQ*.

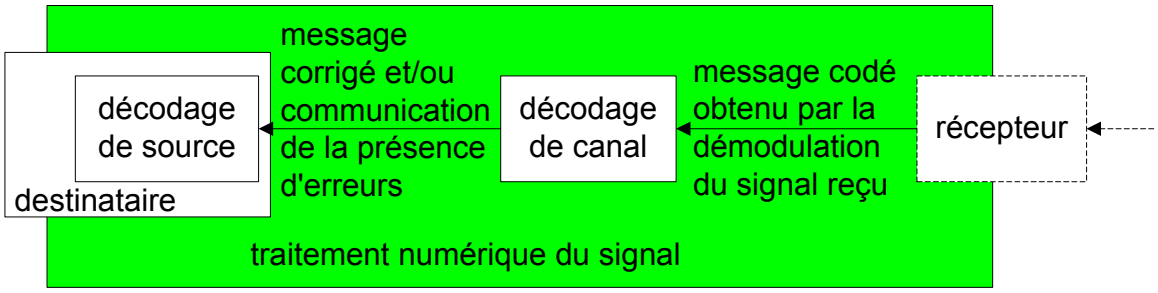

Figure 2-3: principe du décodage d'un message protégé par codage de canal.

Le principal désavantage du mode *FEC* est l'utilisation constante d'une plus large bande passante, même en l'absence d'erreurs. Cette méthode s'applique surtout pour des systèmes de communication où le retard de retransmission n'est pas acceptable. Généralement, pour des systèmes ayant des taux d'erreurs raisonnables, les coûts liés aux demandes et aux retransmissions des blocs de signal erronés (mode *ARQ*) sont normalement moins importants que ceux causés par l'usage d'une bande passante plus large (mode *FEC*).

On observe que pour des applications supportant un retard de transmission, un mode mixte (hybrid ARQ-FEC) permettant de bénéficier des avantages des deux approches est couramment utilisée. Grâce à la redondance fournie par le mode *FEC*, le système cherche d'éventuelles erreurs qui sont ensuite corrigées. Si le taux d'erreurs est supérieur à celui supportable par la méthode *FEC*, la méthode *ARQ* intervient en exigeant la retransmission du message [Noki00] [Pana01].

2.3 Types de codage de canal

2.3.1 Codes en bloc linéaires

Alors que Shannon a défini les limites d'une communication fiable, Hamming et Golay se sont intéressés au développement du premier système permettant un contrôle des erreurs, ce qui a donné naissance à la branche mathématique de la théorie du codage (*coding theory*) [Vale98].

Le mérite de la découverte du premier code permettant la correction des erreurs est attribué à Hamming. En 1946, alors qu'il travaillait dans les laboratoires Bell, il était frustré par la non-fiabilité des ordinateurs de ce temps-là; les ordinateurs équipés de systèmes de détection d'erreurs arrêtaient prématurément l'exécution des programmes en présence d'erreurs. Ainsi,

Hamming chercha un moyen pour coder les données d'entrées de manière à ce que les ordinateurs puissent non seulement détecter les erreurs mais également les corriger. Il proposa de regrouper les données en paquets de 4 bits afin de déterminer 3 bits supplémentaires par combinaison linéaire (formule (2.1), Table 2-1). Les ordinateurs disposaient ainsi d'informations suffisantes pour pouvoir identifier et corriger l'éventuelle erreur présente dans un tel bloc élargi à 7 bits (Figure 2-8) [Vale98].
En 1950, Hamming publia dans le *The Bell System Technical Journal* les résultats de son étude sur les codes corrigeant une erreur unique [Hamm50].

Message	Bits de parité	Message codé	Message	Bits de parité	Message codé
0000	**000**	0000**000**	1000	**101**	1000**101**
0001	**011**	0001**011**	1001	**110**	1001**110**
0010	**110**	0010**110**	1010	**011**	1010**011**
0011	**101**	0011**101**	1011	**000**	1011**000**
0100	**111**	0100**111**	1100	**010**	1100**010**
0101	**100**	0101**100**	1101	**001**	1101**001**
0110	**001**	0110**001**	1110	**100**	1110**100**
0111	**010**	0111**010**	1111	**111**	1111**111**

Table 2-1: exemple de code de Hamming (7,4). Pour chaque groupe de 4 bits (message), ce code génère 3 bits supplémentaires (bits de parité), formant ainsi un message codé de 7 bits. L'équation (2.1) définit la combinaison linéaire permettant la génération des messages codés.

Les premiers codes en bloc montraient des limites importantes: la correction d'une erreur unique demandait un nombre conséquent de bits supplémentaires. Cet aspect indésirable poussa Golay dans le perfectionnement de la génération de codes en bloc. Ce travail permit la définition de codes dont les capacités de correction étaient supérieures à celles des codes initiaux de Hamming [Vale98].

A partir de ce résultat prometteur, d'autres codes ont été ensuite développés et raffinés, élargissant la gamme des codes en bloc et le nombre de contributions au domaine de la théorie du codage. En particulier [Proa95] [Vale98] [Liew02]:

- Les codes Reed-Muller ('*RM*') qui, par rapport aux codes de Hamming et Golay, permettent d'obtenir une vitesse de décodage supérieure, un

choix plus vaste de tailles des blocs sortants (*code words*) et une amélioration des propriétés de correction.

- Les codes cycliques (*Cyclic Redundancy Codes*, '*CRC*') qui possèdent des propriétés mathématiques particulières, favorisant la réalisation matérielle de l'encodeur et du détecteur d'erreurs (Figure 2-9). Ainsi, ils sont principalement utilisés pour la détection d'erreurs.
- Le codes *Bose-Chaudhuri-Hocquenghem* (*'BCH'*), représentant une large classe des codes cycliques, qui ont été découverts simultanément par Hocquenghem et par le groupe Bose et Ray-Chaudhuri. La taille des messages codés vaut $n=q^m-1$, où m est une valeur entière et q la taille de l'alphabet du code. Les codes *BCH* permettent l'utilisation d'un alphabet non-binaire ($q>2$)[1]. L'utilisation des codes dans un champ de Galois (*Galois fields*) *GF(q)* plus étendu ($q>2$) permet d'améliorer la qualité de protection contre les concentrations temporelles d'erreurs (*Burst errors*). L'exemple le plus représentatif est la famille de codes *Reed Solomon*.

Principe des codes en bloc

Chaque famille de codes possède des caractéristiques et potentialités propres, bien qu'elles aient été développées en suivant le même principe: le regroupement des symboles d'entrée en blocs afin de leur ajouter une quantité contrôlée de redondance (Figure 2-4).

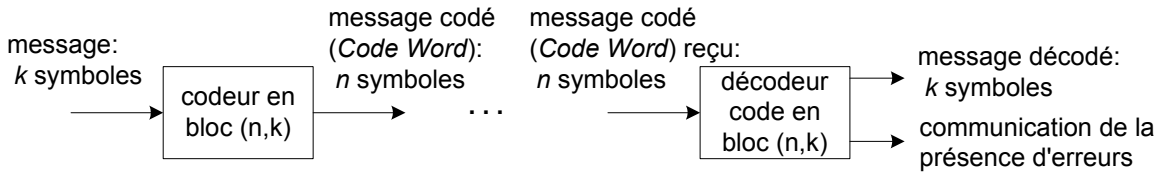

Figure 2-4: signaux d'entrée et de sortie du codage et du décodage des codes en blocs.

En considérant le cas d'un code binaire linéaire *(n,k)*, la stratégie commune des codes en blocs est la modification de la représentation numérique des 2^k messages possibles d'entrées. L'utilisation d'un espace de codage plus grand de celui utilisé pour représenter le message introduit la redondance nécessaire au codage de canal (Figure 2-5). Si la transformation d'un espace de codage à l'autre se passe de manière linéaire, le code est appelé *code linéaire* (*linear code*) [Lust00].

[1] Un codage qui utilise un alphabet contenant un nombre $q>2$ de symboles est dit non-binaire (*nonbinary coding*).

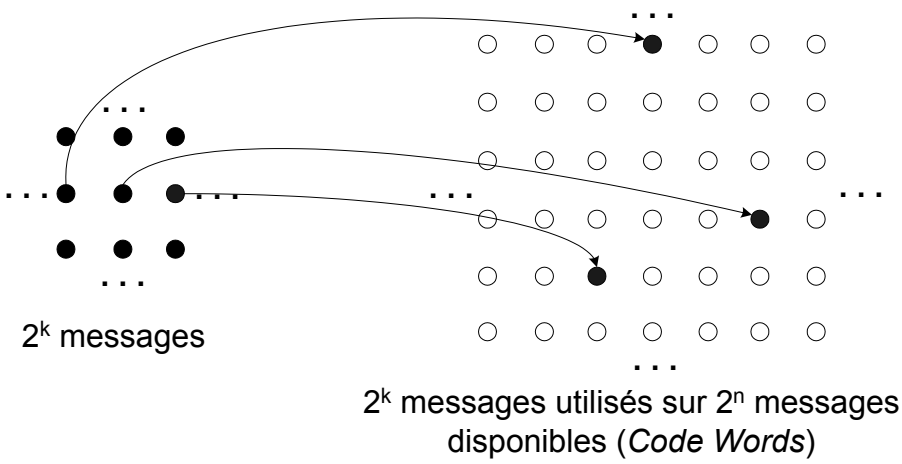

Figure 2-5: principe de l'insertion de redondance par une nouvelle représentation du message utilisant un plus grand espace de codage.

Un exemple est le code de *Hamming* (7,4) [Proa95], qui utilise 16 messages codés dans un espace de 128 messages possibles. Le message codé $\boldsymbol{x}_{1x7}$ est formé par la multiplication (modulo 2) des bits d'information du message ***Info Bits***$_{1x4}$ par une matrice dite "génératrice" qui décrit les combinaisons linéaires utilisées. On a donc:

$$\left[x_0\, x_1 \ldots x_6\right] = \left[\mathit{Info\ Bits}_0\ \mathit{Info\ Bits}_1 \ldots \mathit{Info\ Bits}_3\right] \cdot \begin{bmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{bmatrix} \quad (2.1)$$

où le vecteur-ligne ***Info Bits***$_{1x4}$ représente les *4* bits formant le message à protéger. En général, un message codé $\boldsymbol{x}_n$ obtenu en utilisant un code linéaire *(n,k)* est décrit par la relation

$$\boldsymbol{x}_{1\times n} = \boldsymbol{Info\ Bits}_{1\times k} \cdot \boldsymbol{G}_{k\times n}, \quad (2.2)$$

où le vecteur-ligne ***Info Bits***$_{1xk}$ représentent les *k* bits du message et $\boldsymbol{G}_{kxn}$ est la matrice génératrice.

Chaque code est normalement spécifié par le nombre de bits du message codé *n* et par le nombre *k* de bits du message original à protéger (appelés 'bits d'informations', *Information bits*). Si la nouvelle représentation est formée par la simple concaténation d'un nombre de bits supplémentaires, le code est appelé *systématique* et les bits supplémentaires sont dénommes *bits de parités*

(*Parity bits*). Le rapport entre les tailles de ces deux messages $R_c=k/n$ est appelé *rendement du code* (*Code Rate*).

Distance de Hamming et *poids* d'un message

Un important paramètre, caractérisant le code en bloc, est la distance minimale existant entre les divers messages codés (*Minimal distance of the code*) [Proa95], qui se mesure par la distance de Hamming (*Hamming Distance*).

La distance de Hamming d_{ij}, entre deux messages i et j, indique le nombre de cas où les symboles correspondants des deux messages sont différents. En utilisant cette notion, la distance minimale d_{min} du code est définie comme la distance d_{ij} la plus petite existant entre les messages générés par le code en bloc:

$$d_{\min}=\min_{i\neq j}\left\{d_{ij}\right\}. \tag{2.3}$$

Dans le cas de codes en bloc linéaires, la recherche de la distance minimale d_{min} peut être simplifiée. Si les messages sont générés par des fonctions linéaires, la différence entre deux messages est également un message [Proa95]. Soit w_i le *poids* (*weigth*) d'un message codé $\boldsymbol{i}$, identifiant le nombre de ses éléments non-nuls, la distance minimale d_{min} (2.3) peut être définie ainsi:

$$\begin{aligned} &d_{\min}=\min_{\boldsymbol{i},\boldsymbol{i}\neq\boldsymbol{i}_0}\left\{w_i\right\}, \\ &\boldsymbol{i_0} \text{ est le message codé 'zéro'} \\ &\quad \text{(message formé uniquement de zéros).} \end{aligned} \tag{2.4}$$

En utilisant la relation (2.4), la recherche de la distance minimale se réduit au calcul du *poids* de 2^k-1 messages au lieu de déterminer $2^{k-1}\cdot(2^k-1)$ distance de Hamming [Proa95].

Qualité de protection

En l'absence d'un codage de canal ou/et d'une redondance "suffisante" des informations contenues dans le message à coder (d_{min}=1), il n'est pas possible de détecter si le message reçu est corrompu (Figure 2-6).

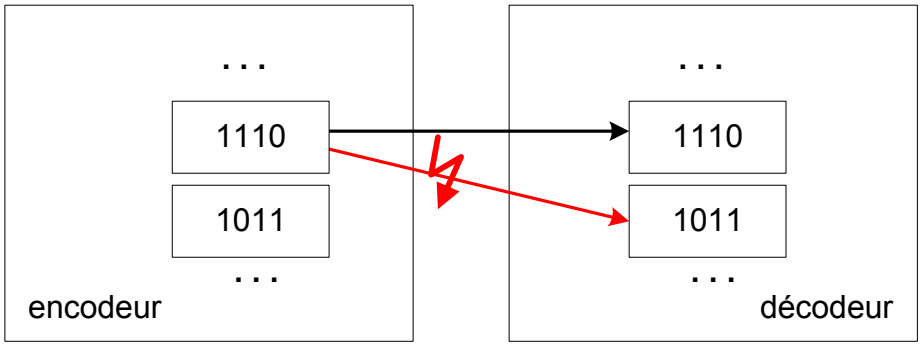

Figure 2-6: envoi d'un message sans protection. Le décodeur n'a aucun moyen de détecter la transmission erronée des données.

En présence d'un message codé (d_{min}>1), le décodeur est confronté à deux situations possibles (Figure 2-7):

- le message reçu appartient au sous-ensemble des messages utilisés lors de l'encodage.
- le message reçu appartient au sous-ensemble des messages qui ne sont pas utilisés par le codeur en bloc. Cette situation indique qu'on est en présence d'une transmission erronée du message.

Un message est défini comme *valide* s'il appartient au groupe de messages utilisés par le codage en bloc. La méthode de base pour vérifier la validité d'un message utilise une matrice $\boldsymbol{H}_{k\times n}$, appelée 'matrice de parité' (*parity-check matrix*). Cette matrice est dérivée de la matrice de génération $\boldsymbol{G}_{k\times n}$ (2.2), de manière à ce que la relation

$$\boldsymbol{H}_{k\times n}\cdot \boldsymbol{r}^{T}_{1\times n} = \boldsymbol{0}_{1\times k}$$

$\boldsymbol{r}_{1\times n}$ est le vecteur ligne représant les bits du message reçu (2.5)

$\boldsymbol{0}_{1\times k}$ est le vecteur ligne formé que par des zéros

soit satisfaite seulement en présence d'un message valide. Par rapport à l'exemple (2.1) précédemment illustré, sa matrice de parité est:

$$\boldsymbol{H}_{3\times 7} = \begin{bmatrix} 1\,1\,1\,0\,1\,0\,0 \\ 0\,1\,1\,1\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,1 \end{bmatrix} \tag{2.6}$$

[Proa95]. L'invalidité d'un message correspond ainsi nécessairement à une transmission erronée des données.

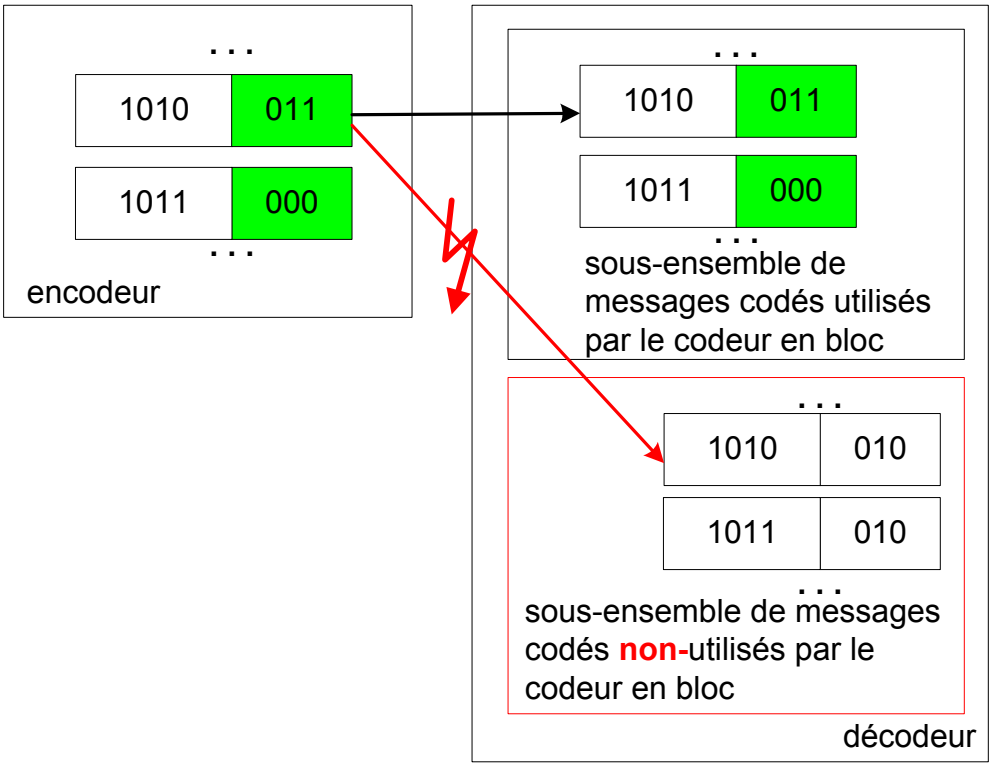

Figure 2-7: envoi d'un message protégé par un code de Hamming (7,4).

Si l'on considère à titre d'exemple un code en bloc avec distance minimale de 3 (Figure 2-8), un tel code garantit la détection de 2 erreurs de transmission qui pourraient affecter le message codé. Les potentialités de détection sont supérieures à ce seuil. Toutefois, un nombre d'erreurs supérieur à $(d_{min}-1)$ peut transformer le message codé par l'encodeur en un message valide qui fausse la méthode de détection (2.5). Cette situation potentielle est nommée situation de *fausse validation*.

Les performances de correction d'erreurs d'un code en bloc sont également déterminées par les distances de Hamming existant entre les messages utilisés pour le codage. Selon le principe de correction d'erreurs par détermination du message valide le plus proche (en termes de distance de Hamming), l'utilisation d'un code en bloc garantit la correction de $\lfloor (d_{\min}-1)/2 \rfloor$ erreurs. Le

code en bloc utilisé à titre d'exemple (Figure 2-8) garantit ainsi la détection de 2 erreurs et la correction d'une seule erreur de transmission.

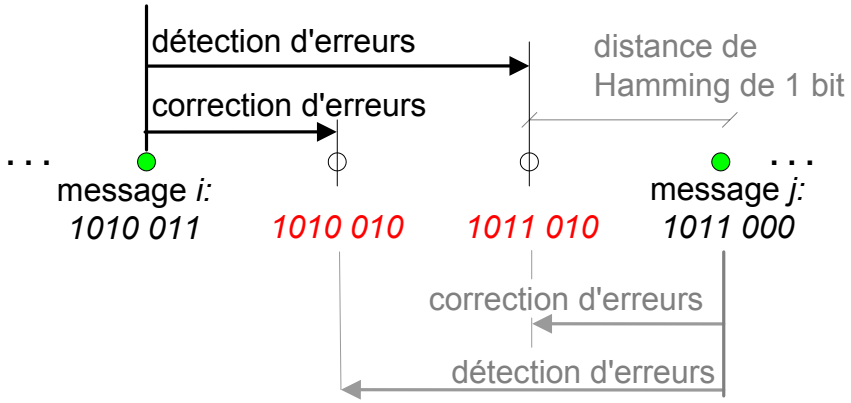

Figure 2-8: exemple graphique de la potentialité de détection et de correction d'erreurs par la détermination du message valide le plus proche.

Codes en bloc *CRC* utilisés par les standards *UMTS*

Les standards *UMTS* disposent de quatre codes *CRC* [Ts25212] pour le contrôle de l'absence d'erreurs dans les messages livrés au destinataire. Les notions de base permettant de comprendre les principes des codes en bloc *CRC* sont décrites ci-après.

Notions de base des codes en bloc *CRC*

Un code linéaire *(n,k)* est nommé code cyclique si tous les décalages cycliques (*cyclic shift*) d'un message codé sont également des messages codés [Lin82] [Proa95].

Dans le domaine du codage cyclique, on associe au message codé $V=[v_{n-1}\ldots\ v_2\ v_1\ v_0]$ le polynôme $v(X)$ de degré inférieur ou égal à $n-1$, tel que:

$$v(X) = v_{n-1}X^{n-1} + \ldots + v_2X^2 + v_1X + v_0, \quad \text{pour les codes binaires: } v_i \in GF(2) \text{ pour } i = 0,\ 1,\ \ldots,\ n-1. \tag{2.7}$$

La méthode de codage *CRC* se base sur la génération du message codé $v(X)$ par la multiplication polynomiale (modulo 2 pour les codes binaires) du message $u(X)$ avec le générateur polynomial $g(X)$, donné par:

$$
\begin{aligned}
v(X) &= u(X)\cdot g(X) = \\
&= (u_{k-1}X^{k-1} + \ldots + u_1 X + u_0)\cdot(g_{n-k}X^{n-k} + \ldots + g_1 X + g_0) = \\
&= (v_{n-1}X^{n-1} + \ldots + v_2 X^2 + v_1 X + v_0).
\end{aligned}
\qquad (2.8)
$$

Si le polynôme générateur *g(X)* de degré *n-k* est un facteur du polynôme X^n+1, ce polynôme génère un code en bloc cyclique *(n,k)* [Lin82] [Pres92] [Proa95] [Will99].

Etant donné le générateur polynomial *g(X)*, on peut demander que le message codé *v(X)* contienne directement le message *u(X)*, tout en respectant les caractéristiques des codes en bloc cycliques.
Une solution est la formation du message codé *v(X)* par la multiplication du message *u(X)* avec le polynôme X^{n-k}, en additionnant successivement le polynôme *b(X)* qui garanti la divisibilité du polynôme *v(X)* par *g(X)* (2.8):

$$
v(X) = X^{n-k}\cdot u(X) + b(X) = a\left(X\right)\cdot g\left(X\right) \qquad (2.9)
$$

La tâche de l'encodeur systématique est ainsi de déterminer le polynôme *b(X)*, qui correspond au reste de la division du terme $X^{n-k}\cdot u(X)$ par le générateur polynomial *g(X)*, tel que:

$$
b(X) = \text{reste}\left\{\frac{X^{n-k}\cdot u(X)}{g\left(X\right)}\right\} \qquad (2.10)
$$

Cette tâche est réalisable matériellement en utilisant un registre à décalage avec une boucle de contre-réaction utilisant des additions modulo 2 (Figure 2-9) [Proa95].

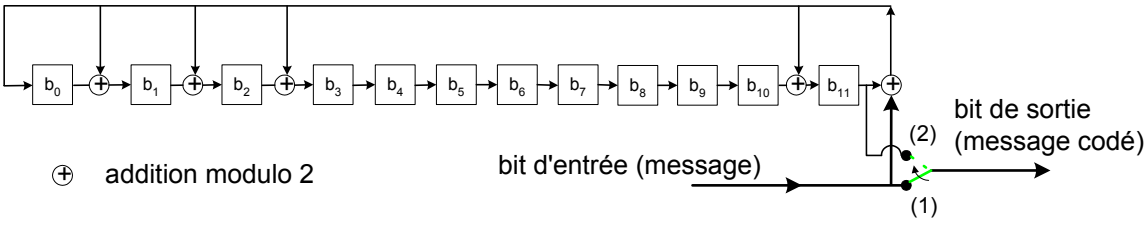

Figure 2-9: schéma à bloc de l'encodeur systématique *CRC* à 12 bits avec générateur polynomial $g(X)=X^{12}+X^{11}+X^3+X^2+X+1$. Après le traitement du dernier bit du message, le contenu des 12 bascules représente le polynôme *b(X)*.

Le standard *UMTS* met à disposition quatre codes cycliques systématiques (Table 2-2), mais il modifie l'ordre des bits de parité lors de leur concaténation [Ts25212]. L'ajout des bits de parité s'effectue dans l'ordre inverse, en commençant par le bit b_0 jusqu'au bit b_{n-k-1}.

Codeur	Longueur [bits]	Taille des messages (n, k)	Générateur polynomial g(X)
$g_{CRC24}(X)$	24	$(2^{23}-1, 2^{23}-25)$	$X^{24} + X^{23} + X^6 + X^5 + X + 1$
$g_{CRC16}(X)$	16	(32767, 32752)	$X^{16} + X^{12} + X^5 + 1$
$g_{CRC12}(X)$	12	(2047, 2035)	$X^{12} + X^{11} + X^3 + X^2 + X + 1$
$g_{CRC8}(X)$	8	(127, 119)	$X^8 + X^7 + X^4 + X^3 + X + 1$

Table 2-2: les générateurs polynomiaux utilisés dans le standard de codage de canal *UMTS* [Ts25212].

Décodage des codes *CRC*

Etant donné que les codes *CRC* sont des codes en blocs, ils sont capables de détecter et de corriger des erreurs. Toutefois, ils ne sont souvent utilisés que comme moyen de détection d'erreurs.

Dans le contexte des codes *CRC* systématiques, deux méthodes principales sont utilisées pour la détection d'erreurs. La première méthode consiste à diviser le message reçu par son générateur polynomial *g(X)*. La valeur du reste de cette division permet la détection et l'éventuelle correction d'erreurs. La réalisation matérielle de cette méthode utilise un registre à décalage avec une boucle de contre-réaction et des opérations d'additions modulo 2 [Proa95].
La seconde méthode calcule les bits de parité en utilisant la partie du message reçu qui représente les bits d'information. Elle compare ensuite ces bits de parité avec ceux reçus dans le message codé.

Situation particulière de codage

Si la taille k_m des messages à protéger ne s'adapte à aucun code cyclique, on peut envisager d'utiliser un code cyclique (n,k) traitant des messages plus larges ($k > k_m$) [Lin82]. En exploitant ce code cyclique (n,k), on traite seulement les messages dont les $L=(k- k_m)$ bits de poids fort (*MSB*) sont des zéros.
L'élimination de ces L bits génère un code linéaire $(n-L,k_m)$, qui n'est désormais plus un code cyclique: ce code est nommé *Shortened cyclic code*.

Toutefois, cette modification ne compromet pas l'utilisation des méthodes de codage et de détection des erreurs précédemment présentées.
Par rapport à la qualité de protection du code $(n\text{-}L,k_m)$, ce code garde les même qualités que celles du code cyclique original (n,k).

Codes en bloc: conclusions

Malgré le succès du codage en bloc, ce type de codage comporte des inconvénients importants [Liew02] [Vale98].
Le regroupement du message en bloc implique normalement une latence due à la nécessité de disposer de tous les symboles du message avant de procéder au codage ou au décodage[2]. Les codes travaillant sur des grands blocs peuvent ainsi entraîner une forte latence, ce qui rend le retard algorithmique du système important.
De plus, la procédure de décodage est affectée par un inconvénient supplémentaire: le décodeur a impérativement besoin de connaître la position du début et de la fin du message codé. La connaissance de la position correcte du premier symbole du message dans le flux de bits est ainsi indispensable au bon déroulement des procédures de contrôle et de correction d'erreurs.
Le dernier inconvénient des codes en bloc provient du fait que le décodage de canal a été initialement considéré uniquement comme un élément qui s'insère entre le récepteur et le destinataire (Figure 2-3). A l'origine, les procédures de décodage des codes en bloc ont ainsi été développées en envisageant le traitement de messages codés avec la même représentation numérique que le message provenant de la source d'information (situation du décodage ferme, *Hard Decoding*). Pourtant, dans le but d'atteindre les limites prédites par Shannon, l'utilisation d'une quantification moins rigide (situation du décodage souple ou pondéré, *Soft Decoding*) est devenue impérative: une meilleure qualité de correction ne peut être atteinte qu'en augmentant le flux de l'information mise à disposition du décodeur. L'inconvénient des codes en bloc est que les méthodes de correction développées s'adaptent mal à ces situations de décodage souple, en raison de leur incapacité de traiter le flux supplémentaire d'informations disponible.

[2] Certaines familles de codes en bloc systématiques possèdent des caractéristiques particulières du codage qui offrent la possibilité de réduire efficacement la latence du décodage. Un exemple est donné par la famille de codes *CRC*.

2.3.2 Codes convolutifs

Les principaux inconvénients du codage en bloc peuvent être évités par une approche différente de la problématique du codage: c'est le cas du codage convolutif (*Convolutional Coding*) [Vale98].
Présenté par Elias en 1955, ce type de codage ajoute systématiquement de la redondance au message codé au fur et à mesure que les symboles du message (chacun formés de *b* bits) sont livrés au codeur. Le message codé se forme ainsi itérativement en utilisant un registre à décalage. Ce registre est dimensionné pour accueillir les *K* symboles les plus récents du message et la génération du message codé utilise *n* fonctions linéaires algébriques. Ces fonctions sont appelées fonctions génératrices (*linear algebraic function generators*).
Un exemple de codeur convolutif est illustré à la Figure 2-10.

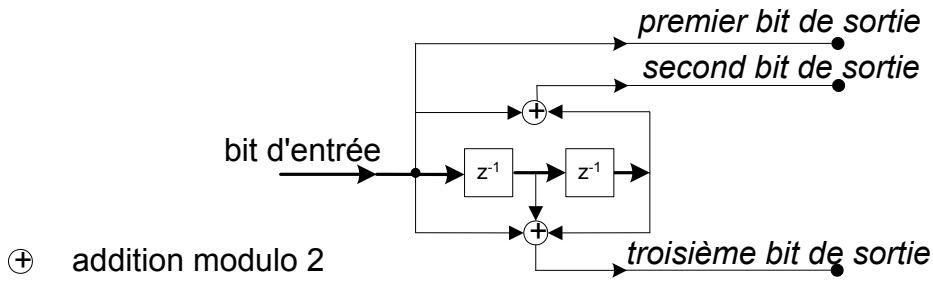

Figure 2-10: exemple de codeur convolutif avec une longueur de contrainte *K*=3. Ce codeur génère 3 bits de sortie (*n*=3) pour chaque bit d'entré (*b*=1).

A chaque instant *t*, le codeur de la Figure 2-10 produit trois bits de sortie en fonction des trois bits d'entrée les plus récents, selon les fonctions algébriques suivantes:

$$\begin{aligned} \text{premier bit}\,(t) &= \text{bit d'entrée}(t) \\ \text{second bit}\,(t) &= \text{bit d'entrée}(t) \oplus \text{bit d'entrée}(t-2) \\ \text{troisième bit}\,(t) &= \text{bit d'entrée}(t) \oplus \text{bit d'entrée}(t-1) \\ &\quad \oplus \text{bit d'entrée}(t-2). \end{aligned} \tag{2.11}$$

Le signal de sortie se forme itérativement en enchaînant les *n* bits de sortie du codeur convolutif (Figure 2-11).

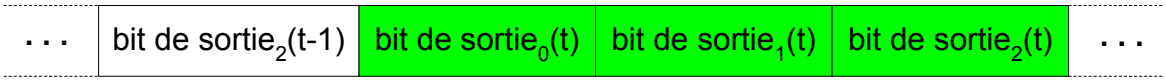

Figure 2-11: représentation graphique du signal de sortie du codeur convolutif de la Figure 2-10.

Le nombre de symboles K, qui contribuent à la génération des n bits de sortie, est nommé *longueur de contrainte* du code (*Constraint Length*). Similairement au cas des codes en bloc, le rendement du code R_c indique toujours le rapport entre les nombres de bits d'entrée et de sortie ($R_c=b/n$).

Les *fonctions génératrices* $\boldsymbol{G_i}$ décrivent les fonctions linéaires algébriques du code. En utilisant l'exemple de la Figure 2-10, chaque fonction algébrique (2.11) est reformulée ainsi:

$$\begin{aligned} \text{bit de sortie}_i(t) = {} & g_i(2)\cdot\text{bit d'entrée}(t) \\ & \oplus g_i(1)\cdot\text{bit d'entrée}(t-1) \\ & \oplus g_i(0)\cdot\text{bit d'entrée}(t-2), \\ g_i(.)\in[0,1],\quad \boldsymbol{G_i} \triangleq {} & [g_i(2)\ g_i(1)\ g_i(0)],\quad i=0,\ 1 \text{ et } 2. \end{aligned} \tag{2.12}$$

Selon cette représentation (2.12), les fonctions génératrices $\boldsymbol{G_0}$=[100], $\boldsymbol{G_1}$=[101] et $\boldsymbol{G_2}$=[111] décrivent exhaustivement les fonctions algébriques (2.11) utilisées par le codeur analysé. Dans le domaine du codage convolutif, il est usuel d'indiquer les valeurs de ces fonctions G_i en représentation octale (base 8).
Si un des bits de sortie transmet systématiquement le bit d'entrée sans aucune modification, le code est dit systématique. Bien que les codes systématiques soient moins performants par rapport aux potentialités de protection des codes non-systématiques, ils offrent l'avantage considérable d'être intrinsèquement non catastrophiques[3] [Thit93].

Les codes convolutifs ne se distinguent pas seulement sur la base de la longueur de contrainte, du rendement du code et des fonctions G_i, mais aussi selon la structure propre au registre à décalage. Dans le cas de l'exploitation d'une boucle de contre-réaction interne (*feedback*) le code est dit 'récursif' [Thit93]. Un tel code est illustré à la Figure 2-12.

[3] Les codes sont appelés catastrophiques, s'ils créent les conditions pour lesquelles un nombre fini d'erreurs de transmission peut engendrer un nombre infini d'erreurs dans le message décodé [Thit93].

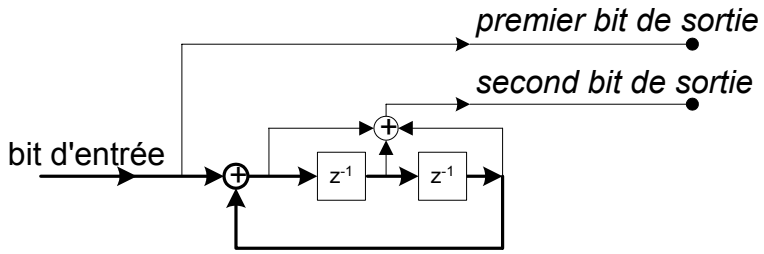

Figure 2-12: exemple de code convolutif récursif avec une longueur de contrainte K=3 (rendement R_c=1/2) [Thit93]. Le registre à décalage présente une boucle de contre-réaction permettant d'additionner (modulo 2) le signal de sortie du régistre à décalage au signal d'entrée.

Le standard *UMTS* [Ts25212] met à disposition deux codeurs convolutifs, possédant chacun une longueur de contrainte de 9. Leurs fonctions génératrices sont G_0=561oct et G_1=753oct pour le code ayant un rendement R_c=1/2 et G_0=557oct, G_1=663oct et G_2=711oct pour le code ayant un rendement de R_c=1/3 (Figure 2-13).

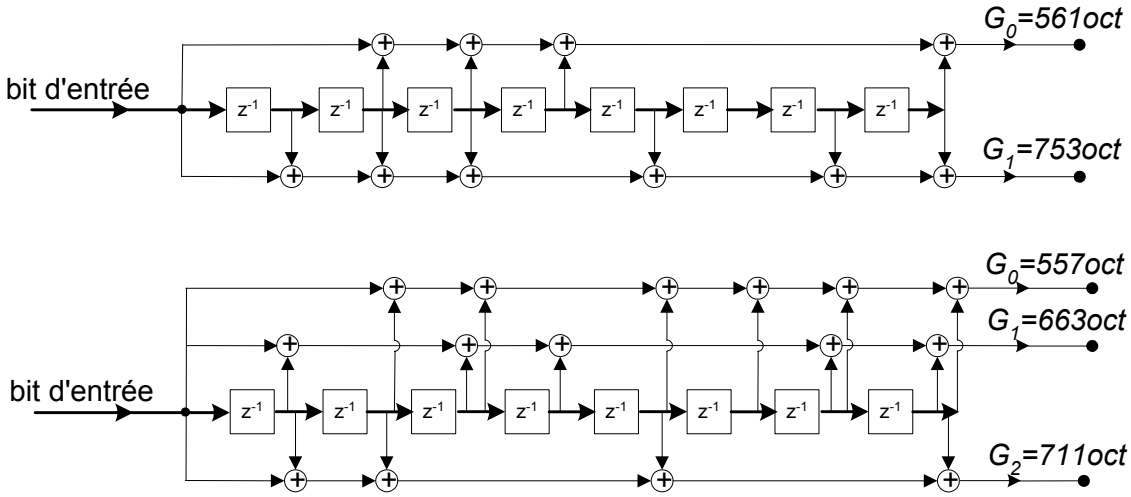

Figure 2-13: représentation graphique des deux codes convolutifs mis à disposition pour la protection des données par le standard *UMTS* [Ts25212].

Principe de protection: formation de la redondance

Analytiquement, le codeur convolutif protège le symbole d'entrée, en tenant compte des *K-1* derniers symboles d'entrée. Pratiquement, il s'agit d'une machine d'états, qui produit une sortie en fonction de l'entrée (nouveau

symbole) et de l'état de la mémoire (formée par les *K-1* anciens symboles d'entrée).
Du point de vue de la protection de canal, la redondance est générée principalement par l'augmentation de la taille du message et renforcée ensuite par la participation des derniers *K-1* symboles à la procédure de codage.
Dans le cas idéal du codage d'une suite infinie de symboles, chaque symbole d'entrée prend part *K* fois à la génération des *n* bits de sortie. L'influence de chaque symbole s'étale ainsi sur un nombre important de bits de sortie ($K \cdot n$ bits).
Il en résulte qu'en augmentant ces deux paramètres (la longueur de contrainte du code *K* et le nombre de bits de sortie *n*), on renforce la redondance du codage convolutif, ce qui permet de disposer d'un système de protection plus efficace contre les erreurs de transmission. Il faut aussi souligner que le changement du nombre de bits de sortie *n*, entraîne la modification du rendement du code R_c.

Représentation du déroulement du codage

Trois méthodes sont communément utilisées pour la représentation graphique du déroulement du codage convolutif: le *diagramme en arbre* (*Tree Diagram*, Figure 2-14), *le diagramme d'états* (*State Diagram*, Figure 2-15) et le *diagramme en treillis* (*Trellis Diagram*, Figure 2-16) [Proa95].

En utilisant à titre d'exemple le code de la Figure 2-10, la Figure 2-14 illustre le début de la procédure de codage par un diagramme en arbre.
Le critère à la base de cette représentation est l'illustration du comportement de la procédure de codage, à chaque instant *t*, en fonction des symboles potentiellement déjà codés. Toutefois, la répétition (persistante) de la même structure rend cette représentation rapidement trop compliquée en pratique.

L'approche du diagramme d'états illustré à la Figure 2-15 est due au codeur convolutif, qui se conduit comme une machine d'états. La valeur des bits de sortie est fonction du symbole d'entrée et de la mémoire du codeur, formée par *K-1* symboles. Ainsi, pour la suite de ce document, le mot 'état' indiquera les symboles formant la *mémoire* temporaire du codeur.

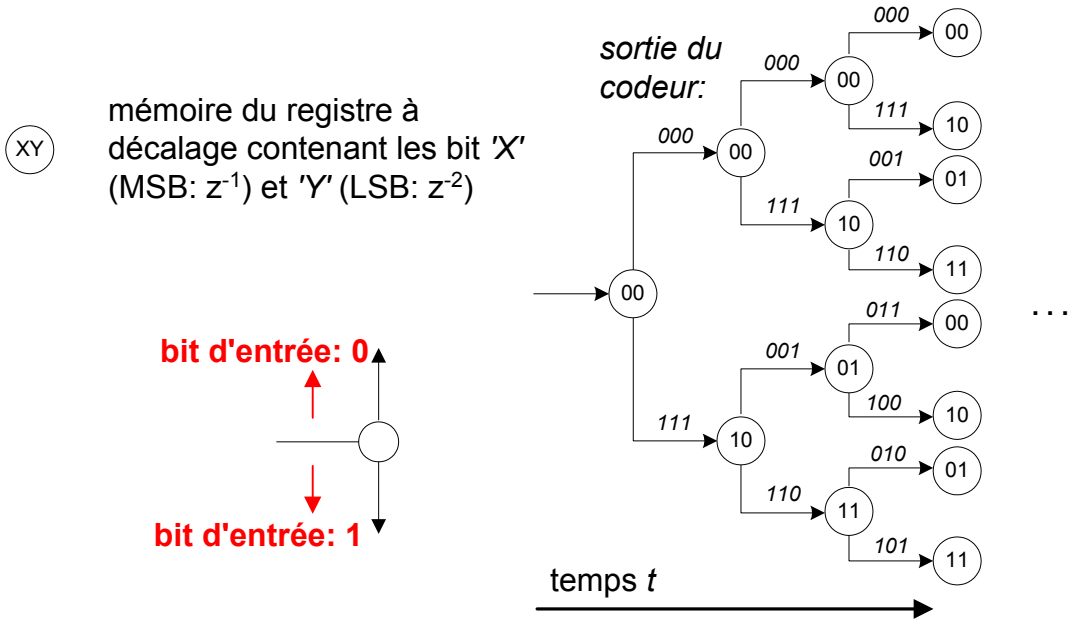

Figure 2-14: diagramme en arbre du codeur de la Figure 2-10, étant donné la remise à zéro de la mémoire du registre à décalage avant le début du codage.

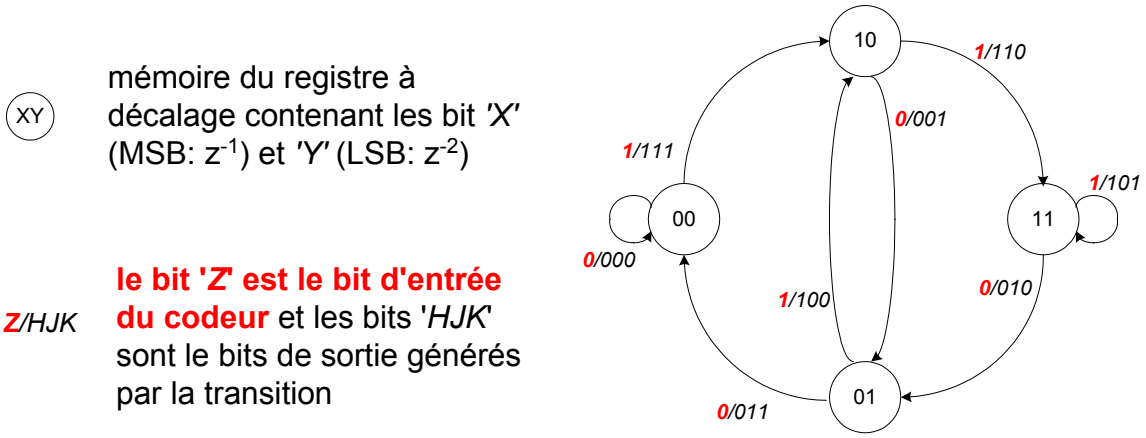

Figure 2-15: diagramme d'états du codeur de la Figure 2-10.

L'avantage de cette approche est la représentation compacte du déroulement du codage. Si l'aspect temporel du codage est aussi demandé, le diagramme en treillis est la méthode de représentation optimale (Figure 2-16).
Cette représentation modifie la méthode de diagramme d'états, en lui ajoutant la dimension du déroulement dans le temps.

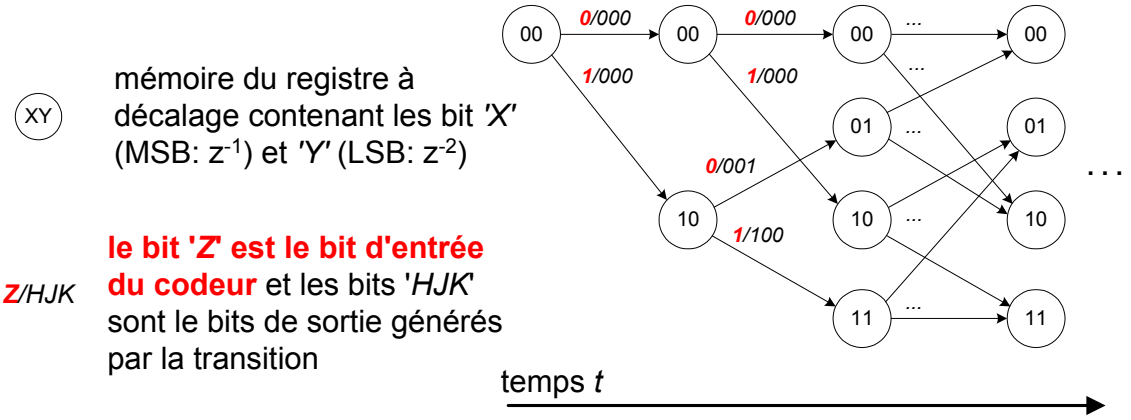

Figure 2-16: diagramme en treillis du codeur de la Figure 2-10, étant donné la remise à zéro de la mémoire du registre à décalage avant le début du codage.

Etat de la mémoire du codeur convolutif au début et à la fin de la procédure de codage

En raison de "l'effet mémoire" du codage convolutif, le caractère univoque du codage ne peut être obtenu au début de la procédure de codage que par l'attribution d'une valeur arbitraire à cette mémoire (*K-1* symboles). Dans la suite de ce document, le terme *condition de départ* se réfère à l'initialisation de la mémoire du codeur convolutif.

Lors du codage de messages contenant un nombre fini de symboles, l'effet mémoire du codage est à l'origine d'un inconvénient: si la procédure de génération des bits de sortie s'arrête immédiatement dès le codage du dernier symbole du message, on est confronté à une situation de protection inégale des symboles à transmettre. Contrairement aux symboles précédents, les derniers *K-1* symboles ne participent pas *K* fois à la procédure de codage. Par conséquent, l'information de ces symboles s'étale sur un nombre réduit de bits de sortie ($<K \cdot n$ bits), diminuant la qualité de protection de ces bits.

Diverses solutions ont été développées afin de remédier à une telle situation de protection inégale. Le système le plus populaire parmi les standards de communication propose l'enchaînement d'une série de *K-1* symboles (*Tail Bits*) à la fin du message à coder. La seule fonction de ces symboles (dénommés dans ce document *bits de terminaison*) est de s'assurer que tous les symboles du message contribuent *K* fois à la procédure de codage: étant donné que ces bits ne transmettent pas d'information, la valeur de ces bits de terminaison est insignifiante.

Du point de vue du décodage, la connaissance des valeurs de ces bits de terminaison permet une ultérieure amélioration des capacités de correction d'erreurs. Ces valeurs déterminent l'état final de la mémoire du codeur, ce qui représente une information importante pour l'opération de décodage. Dans la suite de ce document, on entendra par *conditions d'arrêt de la procédure de codage* les informations tirées du codage de ces bits de terminaison, la valeur de ces bits étant fixée à priori.
Le prix à payer pour une telle solution est la génération d'un message codé qui est allongé de $(K-1)\cdot n$ bits de sortie, en raison du codage supplémentaire de $K-1$ bits (bits de terminaison).

Une solution alternative à la situation de protection inégale est définie par le principe du *Tail-Biting Trellis Code* [Lust00]. Ce principe pose une contrainte supplémentaire à la procédure de codage: les états de la mémoire du codeur, au début et à la fin de la procédure de codage d'un bloc de symboles, doivent être égaux. Cette information supplémentaire est utilisée efficacement lors du décodage des derniers symboles du message, ce qui augmente la qualité de la protection. L'avantage de cette méthode est qu'elle ne nécessite ni l'insertion de bits supplémentaires au message, ni la définition des états de départ et d'arrivée de la mémoire du codeur convolutif.

Capacité de correction des codes convolutifs

La capacité de correction du codage convolutif peut être estimée de manière analogue à celle du codage en bloc. La capacité de correction des codes convolutifs est ainsi évaluée à partir de la distance minimale existant entre deux messages codés. Dans le contexte du codage convolutif, cette distance est appelée *distance libre* ('d_{free}').

La recherche de la distance libre bénéficie de la propriété de linéarité des fonctions génératrices, qui permet de se concentrer uniquement sur les distances entre les messages potentiels et le message *zéro*[4]. Malheureusement, la méthode de recherche (2.4) utilisée par les codes en blocs, ne s'adapte pas de manière optimale à ce contexte de codage, principalement à cause de la variabilité de la taille des messages.

[4] Le message "zéro" est le message codé contenant une séquence infinie de zéros, généré par le codage d'une série infinie de bits d'informations 'zéro'.

Fonction de transfert d'un code convolutif

Les valeurs des distances entre les messages codés peuvent être extraites du diagramme d'états du code analysé [Proa95].

Le code convolutif illustré à la Figure 2-10 est utilisé ici pour décrire la méthode permettant d'établir la qualité de protection du code convolutif et les propriétés de ses distances. Son diagramme d'états est illustré en Figure 2-15.

Etant donné qu'on envisage l'obtention des distances de *Hamming* entre les divers messages et le message zéro, on modifie le diagramme d'états original de la Figure 2-15 de la manière suivante (Figure 2-17) [Proa95]:

- On élimine la boucle de l'état '00'. Etant donné que le message 'zéro' est le message de référence, cette boucle n'apporte aucune contribution à la détermination des distances de *Hamming*.
- L'état '00' est scindé en deux. Ces deux états représentent l'état de départ et d'arrivée du nouveau diagramme d'états. L'objectif est de prendre en compte uniquement les messages divergeant du et ensuite convergeant vers le message zéro.
- On attribue aux états du diagramme les symboles X_A, X_B, X_C, X_D, X_E.
- Les transitions entre les états sont annotées de la manière suivante:
    - Le *poids*[5] des bits de sortie est indiqué par l'exposant de la lettre *D*.
    - Le poids du symbole d'entrée du codeur est désigné par l'exposant de la lettre *N*.
    - Chaque transition est indiquée par la lettre *J*. Cette notation permet de connaître le nombre total de transitions pendant lesquelles le message analysé diverge du message zéro.

En utilisant un tel diagramme d'états (Figure 2-17), qui contient donc cinq états, les équations des états peuvent être formulées ainsi:

$$\begin{aligned} X_B &= JND^3X_A + JNDX_C \\ X_C &= JDX_B + JDX_D \\ X_D &= JND^2X_B + JND^2X_D \\ X_E &= JD^2X_C \end{aligned} \tag{2.13}$$

[5] Le nombre de bits non-zéro.

La fonction de transfert T(D,N,J) du code est définie comme le rapport existant entre l'état final ('X_E') et celui de départ ('X_A'). En utilisant les quatre équations données par (2.13), on obtient la fonction de transfert suivante[6]:

$$T(D,N,J)=\frac{X_E}{X_A}=\frac{J^3ND^6}{1-JND^2(1+J)} \qquad (2.14)$$
$$=J^3ND^6+J^4N^2D^8+J^5N^2D^8+J^5N^3D^{10}+2\cdot J^6N^3D^{10}...$$

Le premier terme ("J^3ND^6") de la fonction de transfert de (2.14) indique qu'il existe un message codé avec une distance de *Hamming* de 6 par rapport au message zéro. Ce message codé diverge sur trois périodes avant de converger vers le message zéro ("J^3") et l'exposant de la lettre *N* indique que le message ne possède qu'un seul bit d'information '1'. Les indications fournies par les six premiers termes[7] de cette fonction de transfert sont décrites à la Table 2-3.

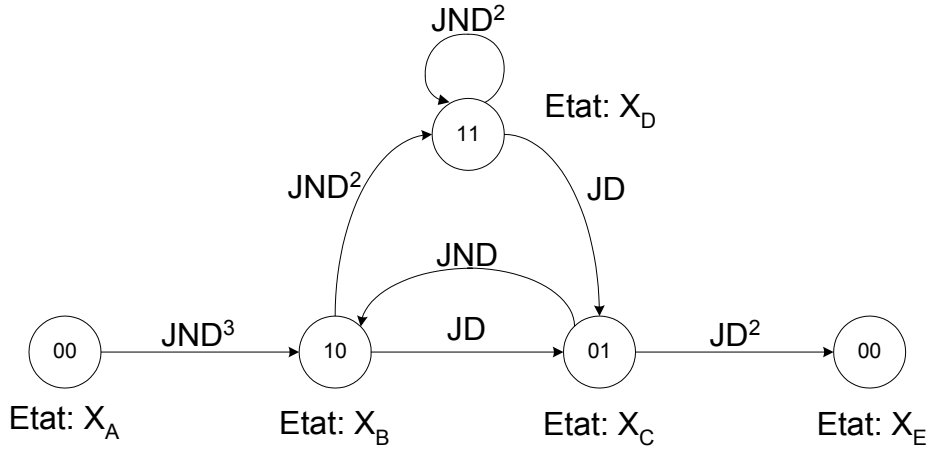

Figure 2-17: diagramme d'état prédisposé pour la détermination de la qualité de protection du code convolutif de la Figure 2-10. La lettre *J* indique la transition, l'exposant de la lettre *N* désigne le poids du symbole d'entrée et celui de la lettre *D* le poids des bits de sortie.

La distance minimale existant entre deux messages codés est indiquée par le plus petit exposant de la lettre '*D*' des termes de la fonction de transfert. Dans notre cas (2.14), la distance libre d_{free} correspond à 6 (Table 2-3).

---

[6] La fonction de transfert peut être directement obtenue en appliquant la règle de Mason (*Mason's rule*) au diagramme de la Figure 2-17.

[7] Les six termes comprenant les exposants de la lettre '*D*' les plus petits. C'est-à-dire, les six messages codés possédant les distances de *Hamming* les plus petites.

En considérant la distance minimale d_{free} et une correction d'erreurs par sélection du message codé le plus proche, le codage convolutif assure la correction de $\lfloor (d_{free}-1)/2 \rfloor$ erreurs de transmission affectant le message codé reçu. L'amélioration de la protection contre les erreurs peut être ainsi réalisée en augmentant la longueur de contrainte K du code convolutif[8] et le nombre n de bits de sortie[9]. Le paramètre modifié est typiquement le premier: cette modification améliore la qualité de protection contre les erreurs sans altérer le rendement du code R_c [Lust00].

Distance de Hamming du message ('*D*')	≤5	6	7	8	9	10	…
Nombre des messages	0	1	0	2	0	4	…
Période de divergence ('*J*')	-	3	-	4, 5	-	5, 6, 6, 7	…
Distance de Hamming du message concerné ('*N*')	-	1	-	2	-	3	…

Table 2-3: caractéristiques des messages possédant les distances de *Hamming* les plus petites, selon la fonction de transfert (2.14).

La Table 2-4 montre les valeurs de distance libre des deux codeurs convolutifs des standards *UMTS*. Ces codes appartiennent à la liste de codes convolutifs établie par Odenwalder (1970), Larsen (1973), Paaske (1974) et Daut (1982) [Proa95]. Etant donné une longueur de contrainte K et un rendement du code R_c, ces codes possèdent la distance libre d_{free} la plus élevée. Ces codes ont été identifiés à l'aide de méthodes de recherche par ordinateur.

Rendement du codeur	**Longueur de contrainte**	**Polynômes générateurs**	**Distance libre**
R_c=1/2	K=9	G_0=561oct, G_1=753oct	d_{free}=12
R_c=1/3	K=9	G_0=557oct, G_1=663oct, G_2=711oct	d_{free}=18

Table 2-4: caractéristiques des codeurs convolutifs proposés par le standard *UMTS* [Ts25212] pour la protection des données.

Les potentialités pratiques de correction de ces codes sont considérablement supérieures au seuil minimal de $\lfloor (d_{free}-1)/2 \rfloor$ erreurs: la qualité de la correction dépend de la méthode de décodage utilisée, du débit des erreurs de

[8] La longueur K augmente la période minimale de divergence des messages codés.

[9] Le nombre n de bits de sortie augmente la distance de Hamming entre les symboles codés.

transmission et de leur emplacement à l'intérieur du message codé. Il faut noter que les concentrations temporelles des erreurs de transmission (*burst errors*) baisse la qualité effective de la correction vers le seuil minimal.

Le terme "*qualité de protection*" se référera dans la suite du document à la capacité de correction de la méthode de décodage traitée.

2.3.3 Utilisation des deux types de codage

Le codage convolutif diffère principalement du codage en bloc par le traitement continu et immédiat des symboles d'information. Le codage convolutif offre de plus une meilleure protection pour la même complexité d'implantation [Karn95].

Du point de vue du décodage, le codage convolutif offre une plus vaste gamme de solutions capables de bénéficier des avantages apportés par une situation de décodage souple. Parmi ces solutions on trouve des méthodes qui se concentrent sur la livraison du message le plus probable, des méthodes estimant les symboles d'informations les plus probables, ainsi que des méthodes livrant des informations supplémentaires relatives à la procédure de décodage (cf. Chapitre 5). De par leur capacité à traiter l'information en continu, les méthodes de décodage convolutif sont principalement utilisés pour la correction d'erreurs de transmission, alors que la détection d'erreurs est essentiellement traitée par le codage en bloc.
Le codage convolutif présente une forte asymétrie entre la complexité de calcul de la procédure de codage et celle de décodage. Malheureusement, la qualité de correction du codage convolutif est très sensible aux concentrations des erreurs de transmission (*burst errors*).

2.4 Codes enchaînés

Si l'on désire créer un code encore plus performant, alors la concaténation de deux types de codes permet de bénéficier des avantages de chaque type de code [Proa95]. La complémentarité des codes en bloc et convolutifs suggère leur mise en cascade, de manière à mieux profiter des qualités des deux familles de codes (Figure 2-18).

Le *codeur extérieur* (*Outer Encoder*) chargé du premier codage du message, est communément un code en bloc. Un code convolutif exécute le second codage et fournit le message codé final ("*code intérieur*", *Inner Encoder*)

[Proa95]. Le décodage de cette double protection est normalement exécuté de manière séquentielle, soit décodant le message reçu, puis en vérifiant et/ou corrigeant le message obtenu.

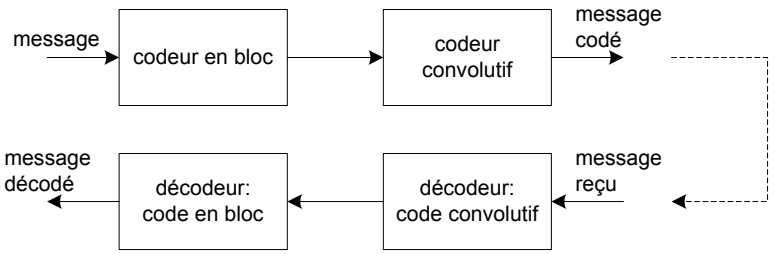

Figure 2-18: structure classique de la concaténation entre un code en bloc et un code convolutif.

Une approche plus récente consiste en l'exécution parallèle de plusieurs opérations de codage des même symboles du message (Figure 2-19). Après la transmission, les messages codés protégeant les même symboles d'information montrent des distorsions non corrélées entre eux: ce codage multiple et indépendant permet au décodeur d'utiliser (iterativement) les informations de décodage de ces messages pour améliorer la qualité de protection contre les erreurs. Le codage *Turbo* utilise cette approche de codage parallèle.

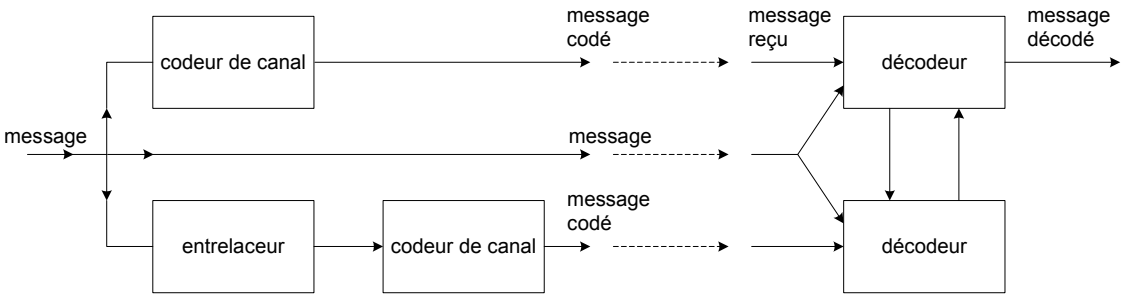

Figure 2-19: principe à la base du codage *Turbo* qui est basé sur l'utilisation parallèle de plusieurs codeurs convolutifs.

2.5 Conclusions

Ce chapitre a brièvement introduit les notions fondamentales de la protection des données contre les erreurs de transmission. A partir des contributions pionnières de Shannon, les deux modes principaux de contrôle des erreurs *FEC* et *ARQ* ont été présentés. Puis, ce chapitre a décrit les principes ainsi que

les propriétés du codage en bloc et du codage convolutif. Ces notions de base seront utilisées dans la suite du document.
En dernier lieu, le chapitre introduit brièvement le codage enchaîné. Le principe et les propriétés de structure particulière de codage ont fait l'objet d'une étude dans le cadre de cette thèse et seront discutés dans le Chapitre 6.

Références

[Draj02] D. Drajic, D. Bajic, "Communication System Performance: Achieving the Ultimate Information-Theoretic Limits?", *IEEE Communications Magazine*, Vol. 40, juin 2002, pp. 124-129.

[Hamm50] R. W. Hamming, "Error Detecting and Error Correcting Codes", *The Bell System Technical Journal*, Vol. 29, No. 2, 1950, pp. 147-160.

[Karn95] P. Karn, "TAPR Error Control Coding Seminar", présentation à *the 1995 Tucson Amateur Packet Radio symposium*, St. Louis, mars 1995, page Internet de P. Karn accédée en été 2002: http://www.ka9q.net/papers/

[Liew02] T. H. Liew, L. Hanzo, "Space-Time Codes and Concatenated Channel Codes for Wireless Communications, *Proceedings of the IEEE*, Vol. 90, No. 2, février 2002, pp. 187-219.

[Lin82] S. Lin, D. J. Costello Jr, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall Series in Computer Applications in Electrical Engineering, New Jersey, Etats-Unis d'Amérique, 1982, Chapitre 4.

[Lust00] F. Lustenberger, "On the Design of Analog VLSI Iterative Decoders", Dissertation *ETH* No. 13879, *Serie in Signal and Information Processing: Volume 2*, Hartung Gorre, Konstanz, Allemagne, novembre 2000.

[Noki00] Nokia, "Hybrid ARQ methods for FDD in Release 2000", *TSGR1#14(00)0869*, TSG-RAN Working Group1, meeting #14, Oulu, Finlande, 4-7 juillet 2000.

[Pana01] Panasonic, "Proposal of bit mapping for type-III HARQ", *TSGR1#18(01)0031*, TSG-RAN Working Group1, meeting #18, Boston, Etats-Unis d'Amérique, 15-18 janvier 2001.

[Pres92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, Cambridge University Press, 1992.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[Shan49] C. E. Shannon, "Communication in the presence of noise", *Proceeding of the Institute of Radio Engineers* (*IRE*), Vol. 37, janvier 1949, pp. 10-21. Ré-edition: (*Reprinted in*): *Proceeding of the IEEE*, Vol. 86, No.2, février 1998, pp. 447-457.

[Thit93] P. Thitimajshima, *Les codes Convolutifs Récursifs Systématiques et leur application à la concaténation parallèle*, Thèse de Doctorat en Electronique, Université de Bretagne Occidentale, France, 1993.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Vale98] M. C. Valenti, *Iterative Detection and Decoding for Wireless Communications*, A Preliminary Review of Initial Research and Proposal for Current and Future Work towards Doctor of Philosophy degree, Virginia Polytechnique Institute and State University, Blacksburg, Virginia, Etats-Unis d'Amérique, 1998.

[Will99] R. N. Williams, "A Painless Guide to CRC Error Detection Algorithm", version 3, page Internet accédée au printemps 1999: ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt

3 Structure de protection établie par les standards UMTS

Le chapitre décrit les standards de communication numérique UMTS. Leur structure de codage et leurs caractéristiques sont rappelées.

Les opérations de codage de canal et de multiplexage des standards UMTS sont passées en revue en les situant d'abord dans le contexte d'un modèle d'abstraction du système UMTS. Les opérations concernées sont le codage en bloc CRC, la concaténation et la segmentation du message, la protection des données par codage, les entrelacements, l'adaptation du débit de transmission, l'insertion des indications de transmission, le regroupement et le multiplexage des données protégées pour la transmission.

La stratégie basique pour l'adaptation de la qualité de protection, en fonction des exigences des diverses applications, est mise en évidence par l'illustration de la structure de codage du service de parole AMR-NB à 12.2 kbps. Cette structure a le mérite de présenter les trois plus intéressants contextes de codage convolutif.

3.1 Modèle hiérarchique du système UMTS à 3 niveaux d'abstraction

Selon un concept général, l'interface radio (*Radio Interface*) des standards *UMTS* est basée sur un modèle de protocole à trois niveaux d'abstraction, aussi appelées couches (*Layers*) d'abstraction [Gath02] [Ts25201]. Les tâches de chaque couche sont bien délimitées et les interactions inter-couches se produisent uniquement à l'aide d'unités spécifiques.

La première couche, dénommée *Physical Layer* ('*Layer 1*'), contient le codage de canal. Cette couche est responsable de la transmission des données, qui lui

parviennent de la seconde couche, la *Data Link Layer* ('*Layer 2'*), sous le contrôle de la troisième couche, la *Network Layer* ('*Layer 3'*, Figure 3-1).

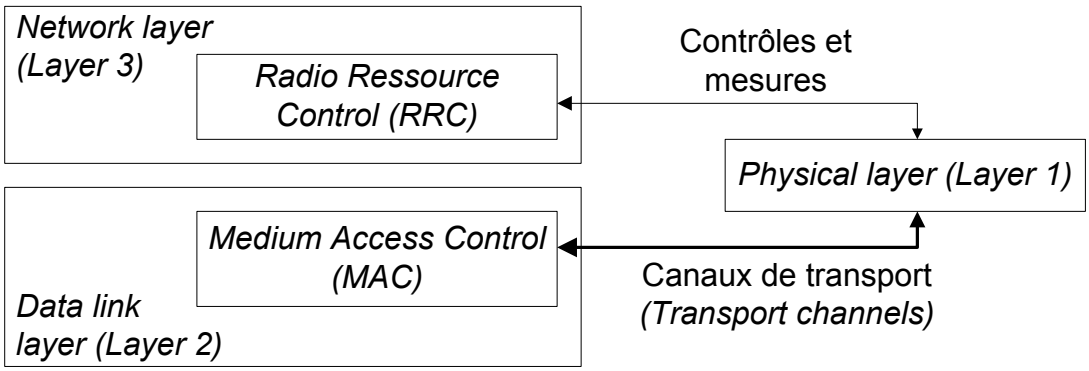

Figure 3-1: illustration graphique des itérations entre le *Physical Layer* et les autres éléments de l'interface radio prévue par l'*UMTS* [Ts25201].

Les données du message parviennent à la première couche par *canaux* séparés *de transport* (*Transport Channels*, '*TrCHs'*). Cette méthodologie permet l'application de niveaux de protection adaptés aux exigences de chaque canal de transport (*Unequal-Error-Protection*, '*UEP'*). La possibilité d'appliquer des niveaux de protection différenciés est conforme à l'optique de l'*UMTS*, qui vise à offrir un degré de souplesse à la procédure de codage. Cette souplesse permet ainsi aux applications utilisant une telle procédure de bénéficier d'une qualité de protection conforme à leurs exigences.

3.2 Structure de codage de canal et multiplexage

Les données parviennent à l'élément responsable du codage de canal par blocs (canaux de transport), à intervalles de temps réguliers. L'intervalle de temps de transmission (*Transmission time interval*, '*TTI*') est établi selon les caractéristiques et les exigences de l'application considérée. La gamme de choix d'intervalle vaut: 10 ms, 20 ms, 40 ms ou 80 ms.

Le déroulement des opérations relatives au codage de canal et de multiplexage des standards *UMTS* (selon le mode *Frequency Division Duplex 'FDD'*[10] de l'*Universal Terrestrial Radio Access 'UTRA'*) [Ts25212], peut être divisée en deux étapes principales:

[10] Le mode *Frequency Division Duplex* ('*FDD*') est une méthode de transmission en duplex, où les deux transmissions radio unidirectionnelles (*uplink* et *downlink*) utilisent deux fréquences radio différentes.

- La première étape traite de manière individuelle les canaux de transport selon les consignes particulières de l'application. Cette étape englobe le codage en bloc *CRC*, la concaténation et la segmentation du message, le codage de canal par code convolutif ou *Turbo*, le premier entrelacement, l'adaptation du débit de transmission et l'insertion des indications fixes de transmission *DTX*.
- La seconde partie d'opérations regroupe les données traitées par les divers canaux de transport, afin de les préparer à la transmission. Les opérations concernées sont celles du multiplexage, de l'insertion des indications flexibles de transmission *DTX*, de la segmentation par le nombre de canaux physiques et du second entrelacement.

Il faut noter qu'en fonction de la direction de la transmission de type *uplink*[11] ou *downlink*[12], il existe deux séquences d'exécution des opérations. Les différences entre ces deux séquences sont toutefois minimes: la séquence de la transmission *uplink* retarde l'exécution de la tâche d'adaptation du débit de transmission et elle ne nécessite pas l'insertion des indications de transmission *DTX* [Ts25212].

Les simulations, utilisées pour l'analyse et la vérification des performances des méthodes présentées dans ce document, se basent sur la structure de codage de la communication *uplink* (Figure 3-2).
Les opérations impliquées dans cette structure sont les suivantes [Ts25212]:

- Codage en bloc *CRC*. Si l'application requiert une détection d'erreurs après le décodage, les couches supérieures (Figure 3-1) peuvent établir les conditions d'utilisation d'un des quatre codes *CRC*. Cette éventualité permet aussi l'exploitation du mode de contrôle d'erreurs *ARQ*. Le nombre de bits de parités de ce code en bloc vaut 8, 12, 16 ou 24 [Ts25212].

[11] Le lien radio *uplink* est la transmission unidirectionnelle des signaux depuis l'équipement de l'utilisateur (*User equipement*) vers une station de base (*Base station*), depuis une station mobile (*Mobile station*) vers une station mobile de base (*Mobile base station*) ou depuis une station mobile de base à une station de base [Tr21905].

[12] Le lien radio *downlink* indique généralement une transmission unidirectionnelle du réseau vers l'équipement de l'utilisateur [Tr21905].

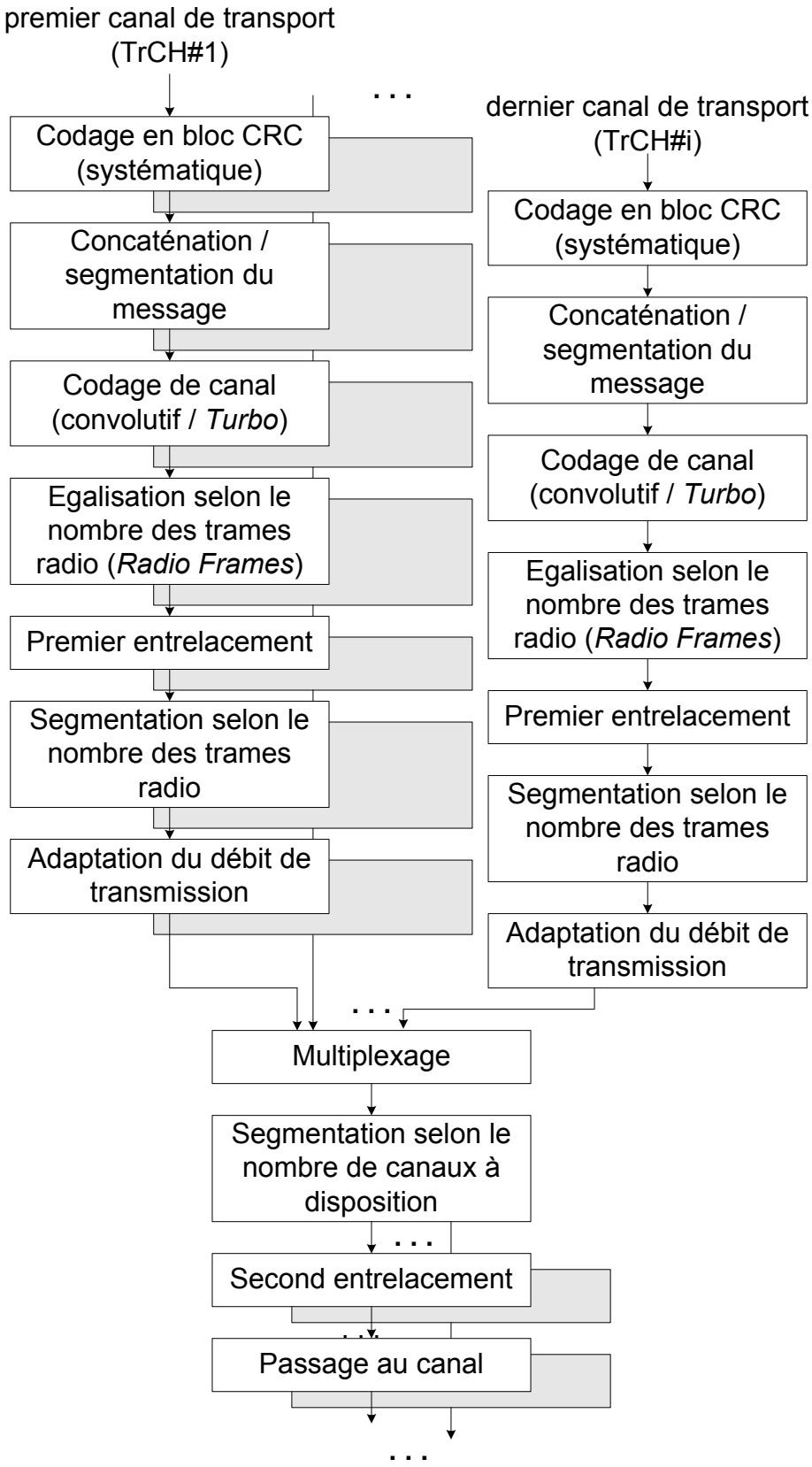

Figure 3-2: diagramme par blocs de la séquence d'opérations selon une transmission *uplink* en mode *FDD* [Ts25212].

- Concaténation et segmentation du message. Tous les blocs de données, qui appartiennent au même intervalle de transmission (*TTI*) et au même canal de transport, sont enchaînés de manière séquentielle. Si le nombre total de bits dépasse le nombre maximal établi par le standard *UMTS* [Ts25212], le bloc de données est segmenté. Ce nombre maximal varie en fonction du type de codage de canal utilisé (codage convolutif ou *Turbo*).
- Protection des donnés par codage. Les données de chaque canal de transport peuvent être protégées séparément, soit par un code convolutif (avec rendement du code R_c de 1/2 ou 1/3), soit par un code *Turbo*. Dans le cas de données segmentées en plusieurs blocs, les divers messages codés sont multiplexés de manière séquentielle après le codage.
- Adaptation du débit de transmission. Etant donné la variation possible du nombre de données d'entrée des divers canaux, il se peut que la taille totale des messages codés ne coïncide pas avec le nombre exact de bits transmissibles par période. Dans ce cas, selon les consignes des deux couches supérieures, certains bits des messages codés sont sélectionnés afin d'être soit répétés, soit éliminés (poinçonnés).
- Segmentation du message codé. Cette séparation se fait selon le nombre des trames radio (*Radio Frames*) disponibles pour la transmission des données de tous les canaux. Dans le système *UMTS*, une trame radio couvre la transmission des données pour un intervalle de 10 ms. Si l'intervalle de transmission *TTI* de l'application dépasse les 10 ms, les données de chaque canal sont groupées de manière à permettre une répartition successive en plusieurs trames radio.
- Egalisation. L'égalisation se fait en fonction du nombre des trames radio. Si le système de codage utilise plusieurs trames radio, cette opération assure une segmentation uniforme des données, en ajoutant, si nécessaire, des bits supplémentaires.
- Premier et second entrelacement. Ces deux opérations exécutent un entrelacement en bloc avec une permutation des colonnes. Les paramètres réglant la permutation dépendent de l'intervalle de transmission *TTI* utilisé.
- Multiplexage. Chaque 10 ms, les blocs des divers canaux de transport appartenant à la même trame radio, sont délivrés à l'unité de multiplexage. Cette unité prépare ainsi le message final, contenant des informations appartenant à tous les canaux de transport. Ce message, nommé *Coded composite transport channel* ('*CCTrCH*'), sera ensuite transmis physiquement.
- Segmentation. La segmentation se fait en fonction du nombre de canaux physiques. Si l'application envisage l'utilisation simultanée de plusieurs

canaux physiques (*PhCH*), les bits appartenant à ce dernier message (*CCTrCH*) sont répartis uniformément entre ces canaux.

- Passage au canal physique. Les bits d'entrée sont transférés au canal physique de manière à pouvoir être envoyé, les uns après les autres, par transmission sans fil.

3.3 Exemple d'application: le service de parole Adaptive-Multi-Rate à bande étroite (AMR-NB)

La description du système de codage *UMTS* [Ts25212] met en évidence la stratégie adoptée lors de la définition de ce type de standard, c'est à dire la création de systèmes disposant d'un haut niveau de souplesse. Cette souplesse permet l'adaptation du système de transmission aux diverses exigences des applications, présentes et futures.

La définition des paramètres de transmission et le choix des opérations de codage sont déterminés parallèlement au développement et à la réalisation de l'application. Le document *3GPP* [Tr21904] regroupe ainsi de manière détaillée les informations des diverses fonctionnalités et les gammes de paramètres correspondant. Les paramètres se répartissent en deux classes: les paramètres obligatoires et les paramètres spécifiques aux services envisagés. L'équipement, afin d'être reconnu à l'intérieur du réseau *UMTS*, doit obligatoirement supporter une gamme de fonctionnalités et de paramètres de fonctionnement de base. L'accès à certains services et/ou applications peut requérir des fonctionnalités et des paramètres de fonctionnement supplémentaires.

En ce qui concerne le cas spécifique du service de parole *AMR* à bande étroite (*AMR-NB*) [Ts26073] [Ts26101], la Table 3-1 décrit les fonctionnalités et les gammes de paramètres les plus importantes, qui concernent les opérations de codage et de multiplexage. Ce service utilise trois canaux de transport (Figure 3-3) de manière à pouvoir adapter la protection aux diverses sensibilités aux erreurs des bits d'informations. Un quatrième canal est utilisé pour la transmission des signaux de contrôle.
La stratégie d'une protection différenciée naît de l'expérimentation apparaissant dans la littérature (exemples [Hage88] [Cox91]) et de l'expérience acquise avec les technologies *GSM* [ETSI909]. L'estimation des sensibilités des bits d'information de la trame *AMR-NB* à 12.2kbps (Figure 3-4) montre des valeurs non-uniformes: l'altération des valeurs des bits d'information formant la trame provoque une dégradation du signal de parole

reconstruit, dégradation qui varie fortement selon la fonction du bit d'informations.

Opérations	Commentaires
Détection d'erreurs	Tous les équipements doivent supporter la possibilité d'un codage en bloc *CRC* avec 8, 12 et 16 bits de parité. Par contre, le codage en bloc avec 24 bits de parité est optionnel.
Codage de canal	Tous les équipements doivent supporter le codage convolutif (avec un rendement du code 1/3 et 1/2). Les autres variantes (codage Turbo et absence de codage) sont optionnelles.
Multiplexage	Dans le cas d'un service de parole *AMR* utilisant un canal dédié (*Dedicated channel, 'DCH '*), tous les équipements doivent supporter au minimum 4 canaux de transmission (3 pour les données avec *TTI* de 20 ms et 1 pour la transmission des signaux de contrôle).
Etalement *(Spreading*)	Dans le cas d'un seul service de parole *AMR*, tous les équipements doivent supporter les facteurs d'étalement *SF*=64 (qui signifie 600 bits/trame radio [Ts25211]), *SF*=128 (300 bits/trame radio) et *SF*=256 (150 bits/trame radio).

Table 3-1: extrait des fonctionnalités relatives à l'accès au service de parole *AMR* [Tr21904].

Selon la structure de protection adoptée (Figure 3-3), les bits de la trame *AMR-NB* à 12.2 kbps sont groupés en trois classes, indiquées par les lettres 'A', 'B' et 'C', en fonction de leur sensibilité aux erreurs [Ts26101]. Dans la classe *A*, on trouve les bits d'information les plus importants dont l'utilisation nécessite l'absence totale d'erreurs. Par conséquent, le codage des bits d'information utilise le code convolutif le plus performant (R_c=1/3) et la qualité du décodage est contrôlée par l'utilisation d'un code en bloc *CRC* systématique. La classe *B* renferme les bits de valeur, dont l'éventuelle altération ne requiert pas la prise de mesures spéciales de recouvrement d'erreurs (*Error Concealment*). La classe *C* contient tous les bits présentant une sensibilité mineure aux erreurs: leur niveau de protection est réduit par l'utilisation d'un code convolutif avec rendement de 1/2.

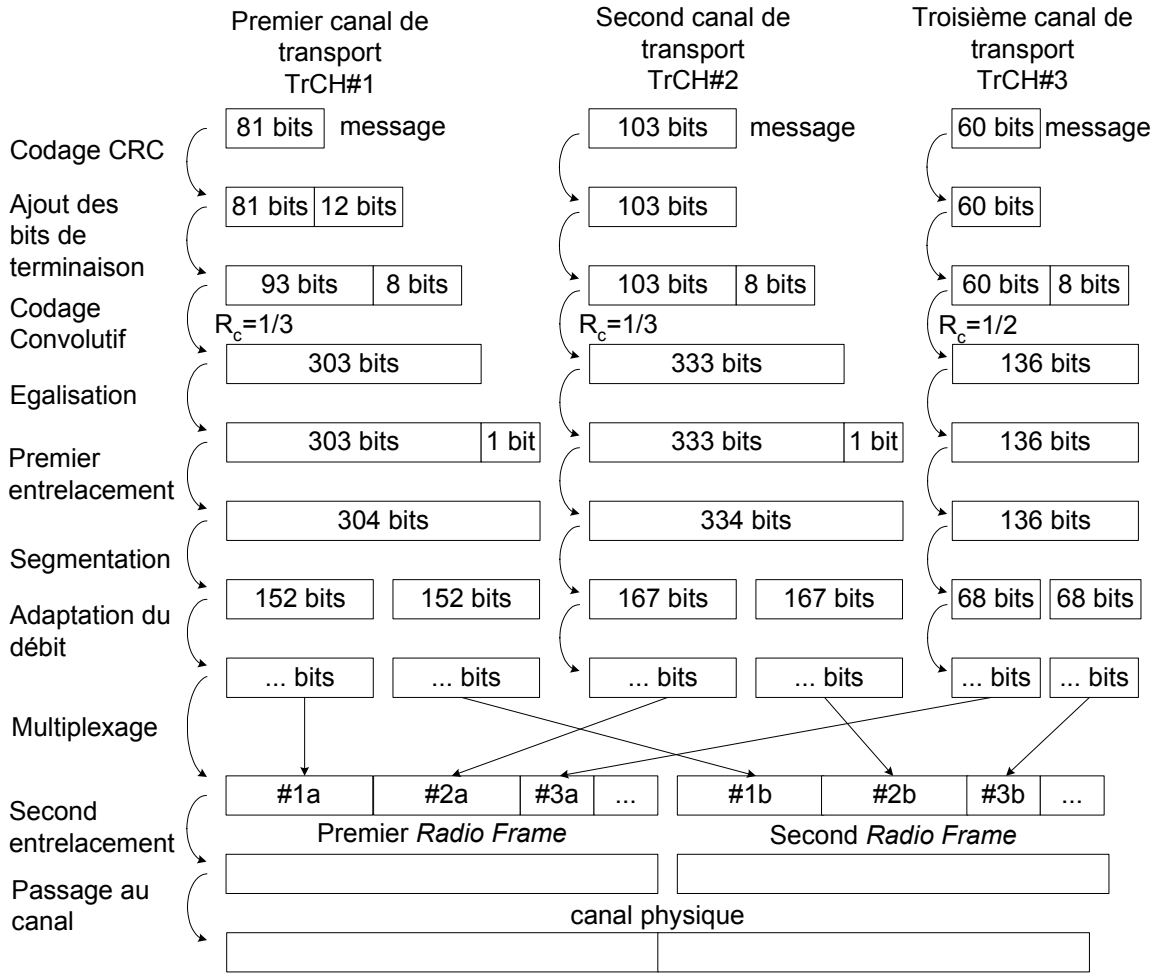

Figure 3-3: représentation détaillée de la séquence d'opérations nécessaires au codage de canal, proposée par [TrR104] pour le service de parole *AMR-NB* à 12.2kbps (*uplink*, mode *FDD*).

Bien que les paramètres du service de parole *AMR-NB* (Table 3-2) puissent encore subir des modifications[13], cette structure (Figure 3-3) sera utilisée pour l'évaluation pratique des implantations software des algorithmes. En effet, cette structure a le mérite de supporter les contextes de codage les plus intéressants: une protection par l'utilisation conjointe d'un code en bloc *CRC* et d'un code convolutif et le simple codage convolutif avec rendement R_c=1/3 ou R_c=1/2.

[13] Au moment de l'établissement de la structure de protection utilisée pour les simulations softwares.

Nombre de canaux de transport (*TrChs*)	3
Tailles des canaux de transport	81, 103 et 60 bits
Codage *CRC*	Uniquement utilisé pour le premier canal de transport : CRC avec 12 bits de parité
Codage de canal	Codage convolutif avec rendement du code de 1/3 pour les deux premiers canaux de transport et 1/2 pour le dernier
TTI	20 ms

Table 3-2: caractérisation de la structure de protection pour le service de parole *AMR-NB* à 12.2kbps (*uplink*, mode *FDD*), structure proposée par [TrR104].

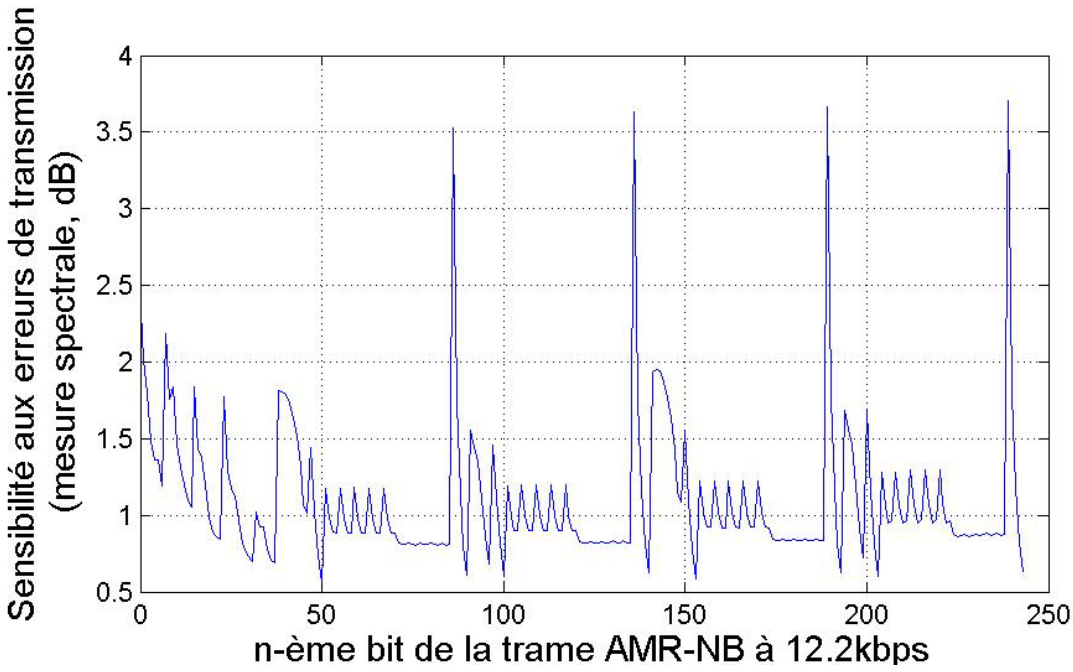

Figure 3-4: mesure objective de la sensibilité aux erreurs des bits d'information constituant la trame du service *AMR-NB* à 12.2kbps. La sensibilité est estimée en altérant systématiquement le bit analysé, avant la reconstruction des signaux de parole (base de données *TIMIT* et *Bdsons*). La dégradation des signaux est évaluée à l'aide de la *mesure spectrale* (*spectral measure*). Les fonctions de chaque bit de la trame *AMR-NB* à 12.2 kbps sont définies dans le standard [Ts26101].

3.4 Conclusions

Ce chapitre porte sur les objectifs des standards de communication numérique *UMTS*.
Parmi les aspects présentés, ce chapitre a illustré la structure particulière du codage de canal prévue par ces technologies de la *3G*. Son environnement de travail ainsi que les opérations de codage et de multiplexage ont été passés en revue.

La stratégie de base adoptée par ces standards *UMTS* est l'adaptation de la qualité de protection en fonction des exigences des applications. La structure de codage du service de parole *AMR-NB* à 12.2 kbps a été utilisée à titre d'exemple. Grâce à ses caractéristiques de protection, cette structure sera utilisée pour l'étude des implantations software de méthodes de décodage convolutif (Chapitre 6).

Références

[Cox91] R. V. Cox, J. Hagenauer, N. Seshadri, et C.-E. W. Sundberg, “Subband Speech Coding and Matched Convolutional Channel Coding for Mobile Radio Channels”, *IEEE Transactions on Signal processing*, Vol. 39, No. 8, août 1991, pp. 1717-1731.

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grand Bretagne, 2002.

[Hage88] J. Hagenauer, ”Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications”, *IEEE Transaction on Communications*, Vol. 36, No. 4, avril 1988, pp. 389-400.

[Tr21904] 3GPP, *UE capability requirements*, document 3GPP TR 21.904, version 3.3.0.

[Tr21905] 3GPP, *Vocabulary for 3GPP Specifications*, document 3GPP TR 21.905, version 0.1.0.

[TrR104] 3GPP, *Channel Coding and multiplexing examples*, document 3GPP TR R1.04, version 0.0.1.

[Ts25201] 3GPP, *Physical layer – General Description*, document 3GPP TS 25.201, version 2.1.0.

[Ts25211] 3GPP, *Physical channels and mapping of transport channels onto physical channels (FDD)*, document 3GPP TS 25.211, version 4.1.0.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Ts26073] 3GPP, *AMR speech Codec: C-source code*, document 3GPP TS 26.073, version 3.0.0.

[Ts26101] 3GPP, *AMR speech Codec: Frame structure*, document 3GPP TS 26.101, version 1.6.0.

4 Transmission de données par réseaux cellulaires sans fil

Ce chapitre traite de la transmission de données par réseaux cellulaires sans fil.

Les facteurs dont dépend la qualité de la transmission sans fil sont tout d'abord présentés. La qualité de la transmission est influencée par des facteurs techniques, tels que le type de modulation, la stratégie de répartition du territoire en plusieurs zones ('cellules'), le type d'antenne des systèmes de communication et les moyens actifs de réduction des distorsions du signal, ainsi que par des facteurs environnementaux, tels que la dimension et la géométrie de la cellule concernée, ainsi que la distance entre la station de base et l'utilisateur et la mobilité de ce dernier.

Ce chapitre traite également de la modélisation des systèmes de transmission. A partir des exigences d'évaluation, de simulation et des types d'erreurs reproductibles, deux modélisations simples du moyen de transmission sont présentées. Il s'agit des modèles "Binary symmetric channel" (BSC) et "Additive white gaussian noise" (AWGN).

4.1 Introduction

Les unités de codage et de décodage de canal font partie d'un système complexe, dont le but est d'assurer la transmission des données par réseaux cellulaires sans fil. L'objectif spécifique du codage de canal est le contrôle des erreurs de transmission des symboles, erreurs causées principalement par la variation des nombreux facteurs influençant la transmission.

La structure simplifiée d'un téléphone mobile (Figure 4-1) contient deux grosses unités de base. La première traite les signaux dans leur bande de base. Elle est responsable de l'adaptation du message à la transmission sans fil ainsi

que de la reconstruction et de la livraison du message reçu au destinataire. Cette unité est dénommée "bande de base". La seconde unité (RF) se charge de la transmission des signaux vers et provenant du/vers le réseau cellulaire sans fil.

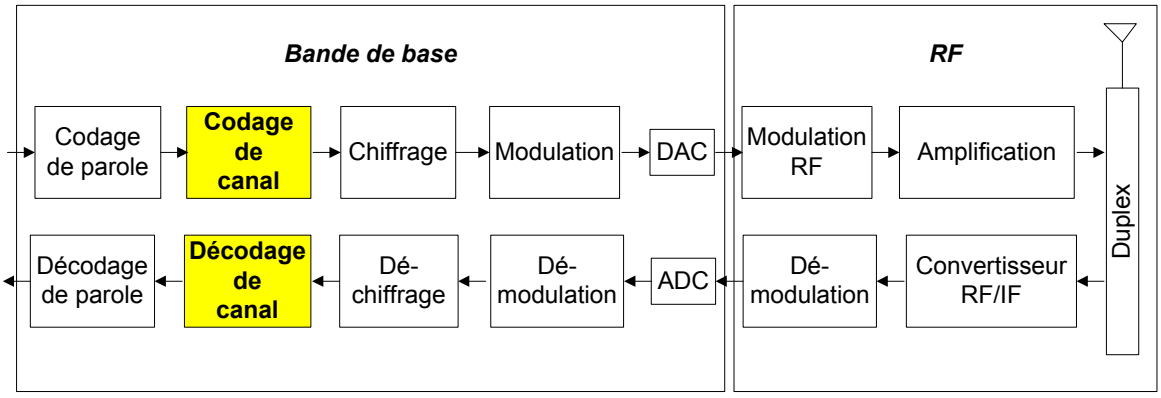

Figure 4-1: schéma très simplifié de la structure d'un téléphone mobile générique.

En considérant cette structure de téléphone mobile, il faut considérer le fait que les caractéristiques des symboles reçus par l'unité de décodage de canal sont influencées par toutes les étapes de traitement et de transmission situées entre le codage et le décodage de canal.

4.2 Facteurs influençant la transmission de données par réseaux cellulaires sans fil

Les caractéristiques de la transmission de données par réseaux cellulaires sans fil dépendent de plusieurs facteurs, aussi bien techniques qu'environnementaux.

En raison des grandes dimensions du territoire couvert par les transmissions aériennes ainsi que du nombre élevé d'utilisateurs, il est impossible de considérer l'utilisation d'une station unique de base pour couvrir toute la surface du territoire. La stratégie adoptée se base sur la répartition du territoire en plusieurs zones dénommées 'cellules', chacune gérée par une station de base (Figure 4-2). Cette stratégie cellulaire s'avère très efficace, aussi bien pour la réduction de la puissance de transmission que pour une meilleure utilisation des canaux de transmission (réutilisation des bandes des fréquences).

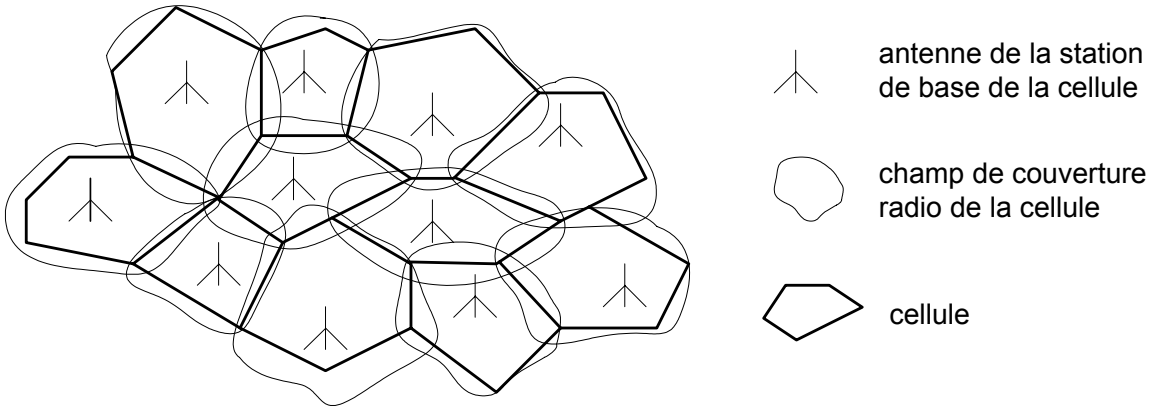

Figure 4-2: stratégie de répartition du territoire en cellules.

Le premier facteur influençant la qualité de la transmission sans fil est la dimension et la géométrie de la cellule, à l'intérieur de laquelle se produit la transmission (Figure 4-3). En effet, la distance entre l'antenne de la station de base de la cellule et celle de l'équipement mobile détermine une atténuation de la puissance de réception, qui varie selon le facteur

$$\frac{1}{\text{distance}^2} \tag{4.1}$$

cf. [Moul92]. Donc, lorsque la dimension de la cellule diminue, la puissance d'émission nécessaire pour une transmission fiable dans la cellule entière peut être ainsi réduite.

Le dimensionnement des cellules doit tenir compte du fait que chaque cellule utilise une bande de fréquences limitée, supportant un nombre fixe de communications simultanées (*Spectrum scarcity*). Par conséquent, l'utilisation de cellules de taille inférieure réduit l'atténuation des signaux transmis et offre un meilleur rapport entre le nombre de communications supportées et la surface du territoire couvert (Figure 4-3). Les opérateurs peuvent également recourir à la création de micro-cellules, afin d'augmenter le nombre d'utilisateurs potentiels dans une partie restreinte du territoire. Les limites inférieures de la dimension des cellules sont déterminées principalement par des aspects économiques. En effet, à partir d'un certain facteur de réduction, les avantages que l'on peut retirer ne justifient pas les coûts supplémentaires de gestion et d'installations.

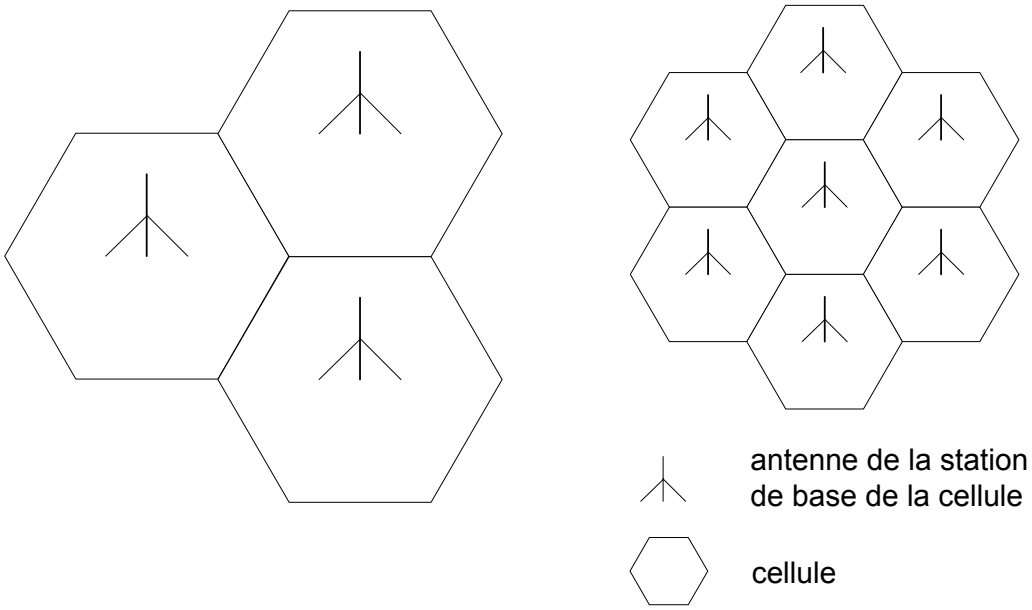

Figure 4-3: répartition du territoire en cellules. Les cellules de taille plus réduite augmentent le rapport entre le nombre de canaux à disposition et le territoire couvert.

Une des raisons de l'efficacité de la stratégie de répartir le territoire couvert en plusieurs cellule est la réutilisation des bandes de fréquences: les cellules non-adjacentes réutilisent les même bandes de fréquences, augmentant le nombre de canaux radio à disposition. Malheureusement, la répartition du territoire en cellules de petite taille implique le rapprochement des cellules utilisant la même bande de fréquences. Ainsi, les transmissions à l'intérieur d'une cellule peuvent directement interférer avec celles des autres utilisant la même bande de fréquences (Figure 4-4). Ces interférences ne peuvent pas être empêchées par des opérations de filtrage et sont appelées *interférences co-canal*.
Un moyens efficace pour réduire l'importance de ces interférences est l'utilisation d'antennes directionnelles: selon la structure des cellules et la répartition géographique du territoire, il est possible d'atténuer les interférences co-canal, en dirigeant intelligemment les émissions des antennes des stations de base (Figure 4-5). L'utilisation de ces antennes directionnelles s'adapte également bien aux situations dans lesquelles des obstacles modifient la puissance de réception des signaux jusqu'à créer des situations d'ombrage.

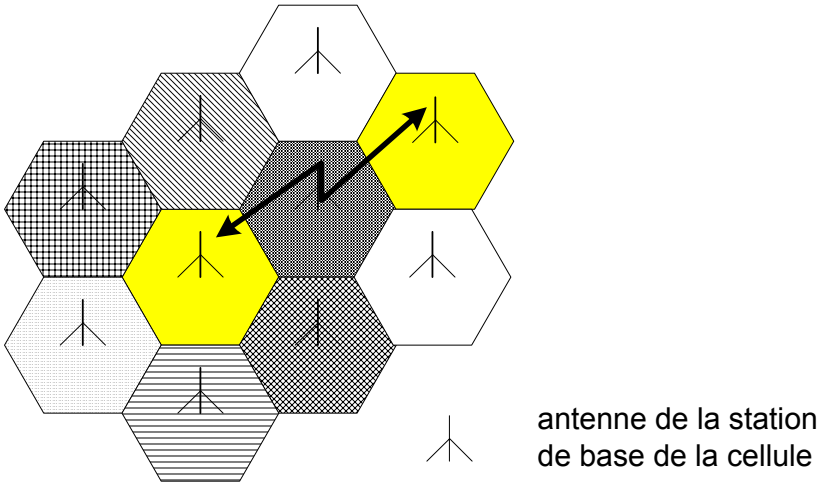

Figure 4-4: situation d'interférence co-canal. En utilisant la même bande de fréquences, les transmissions à l'intérieur d'une cellule interfèrent avec celle de l'autre.

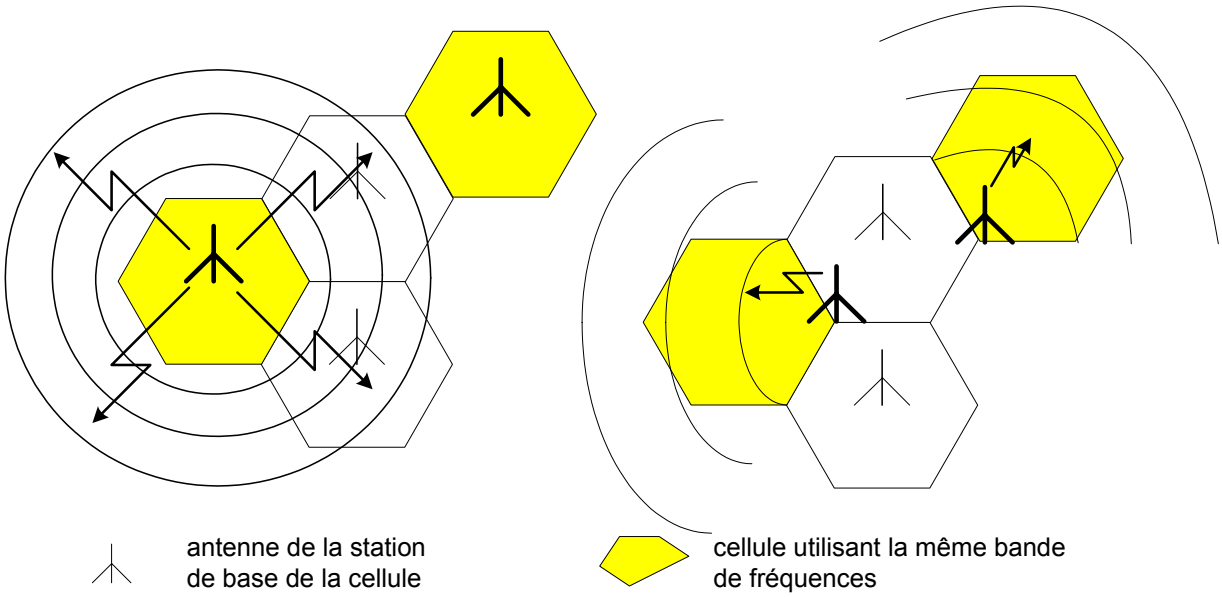

Figure 4-5: exemple de stratégie d'émission et réception des stations de base. L'utilisation des antennes directionnelles permet d'atténuer les interférences co-canal.

Dans le cas de complexes urbains densément peuplés, les transmissions sans fil deviennent encore plus critiques. Etant donné la forte densité d'obstacles, la qualité des transmissions varie sans cesse et avec des différences locales pouvant devenir très importantes. Par conséquent, l'atténuation de la puissance de réception (4.1) peut ainsi atteindre pratiquement un facteur valant:

$$\frac{1}{\text{distance}^4} \tag{4.2}$$

cf. [Moul92].
Si loin des agglomérations, la transmission implique normalement la réception d'un seul signal, dans ces espaces densément construits l'antenne reçoit plusieurs signaux temporellement décalés (*Signal Multipath*). De plus, les multiples réflexions du signal contre les obstacles créent des signaux mutuellement déphasés, qui, s'additionnant de manière destructive, dégradent la qualité de la communication (*Signal Fading*).

Un facteur influençant les caractéristiques de transmission est la mobilité de l'équipement. La transmission est également affectée par des distorsions, dont l'importance est fonction de la vitesse de l'équipement mobile (effet *Doppler*), de la dimension des cellules impliquées et de l'environnement (installations urbaines ou campagne).

Enfin, des facteurs techniques (principalement le type de modulation et les moyens de réduction des distorsions) ont un effet direct sur la qualité de la transmission sans fil. Parmi ces moyens on trouve les techniques d'étalement de spectre (qui permet de réduire les interférences co-canal, [Ts25213]), l'utilisation de plusieurs antennes réceptrices (*Multiple Antennas*, [R1-1218]) et l'utilisation d'une recombinaison de plusieurs signaux reçus (exemple: *Maximal Ratio Combination* [Gath02]).

4.3 Modélisation du moyen de transmission

Le standard *UMTS* [Ts25212] définit la structure et la séquence des moyens mis à disposition pour la protection des données contre les erreurs de transmission dans le cadre des technologies de la *3G*. Toutefois, le standard n'impose aucune procédure de décodage.
Afin de permettre l'évaluation des performances de diverses stratégies de décodage, il faut analyser le comportement du décodeur face à des signaux perturbés. Ces signaux peuvent être obtenus soit par acquisition, soit en les générant par simulation.

Une méthode de génération consiste à manipuler des bits contenus dans les trames radio (*Radio Frames*) de manière à obtenir des séquences de symboles présentant certains types de distorsions (Figure 4-6). Idéalement, cette manipulation doit reproduire fidèlement les effets causés par une situation de

transmission donnée, et tenant compte de l'influence des autres éléments du système situés entre le codage et le décodage de canal.
La reproduction d'une situation réelle implique sa modélisation précise et la définition exacte de tous les paramètres et de tous les éléments, qui la décrivent. Etant donné le nombre élevé des facteurs intervenant dans la transmission aérienne et leur variabilité, la réalisation et la mise au point d'un tel système sont très complexes et coûteuses.

Une alternative efficace à la modélisation est l'utilisation de séquences réelles d'erreurs, obtenues par mesure. La transmission de messages connus et l'acquisition successive des séquences correspondantes de symboles à l'entrée du décodage de canal permettent l'obtention de divers modèles d'erreurs. Cette approche nécessite un système de communication *UMTS* opérationnel ainsi que d'un équipement d'acquisition.

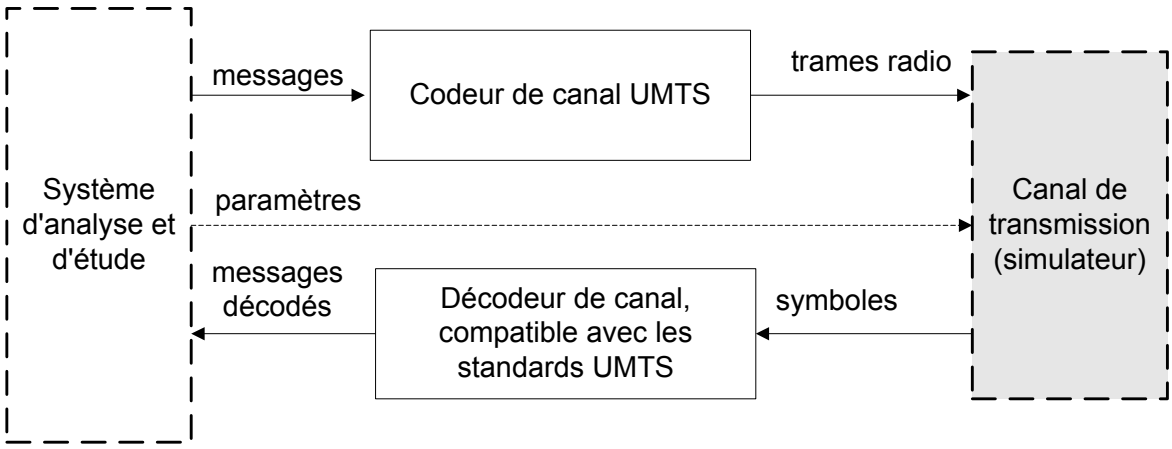

Figure 4-6: système d'analyse manipulant le contenu des trames radio (*Radio Frames*).

4.3.1 Système de perturbation artificielle des signaux

Lors du développement d'une nouvelle technologie, l'étude de base de la qualité de protection utilise généralement une modélisation simple du moyen de transmission.

Types d'erreurs

La définition d'un système simple de modélisation doit tenir compte de la réaction du système de codage *UMTS* [Ts25212] en présence des divers types d'erreurs de transmission. Il existe deux principaux types d'erreurs: le *Random-error* et le *Burst-error* [Lust00].

Le *Random-error* caractérise les moyens de transmission qui ne présentent pas d'effets de mémoire (*Memoryless channels*). Le bruit affecte chaque symbole de manière indépendante, générant des erreurs non-corrélées dans la séquence de symboles reçus. Les transmissions spatiales en sont des exemples typiques.

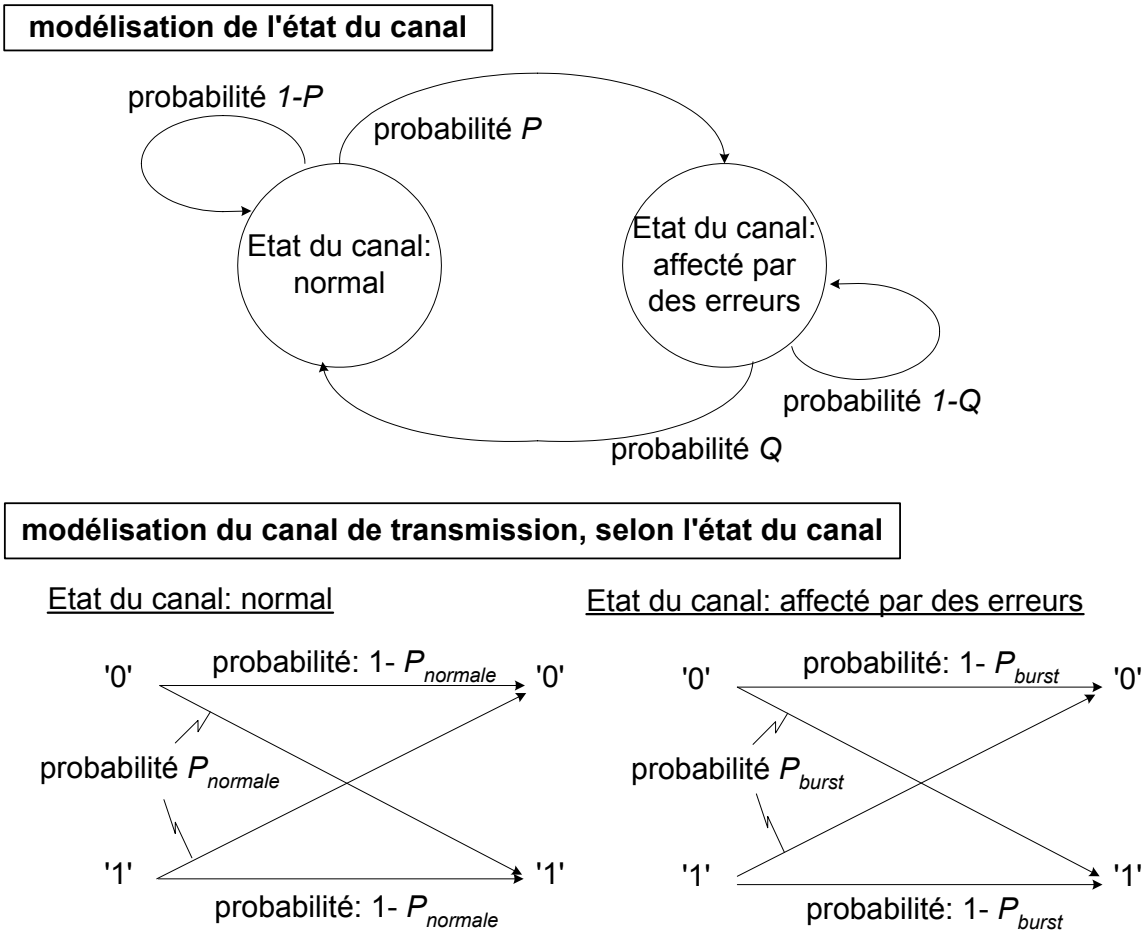

Figure 4-7: modélisation des erreurs de type *Burst* au moyen du model de canal "*Gilbert Elliot Channel*" (*GEC*) [ITU191]. Selon l'état du canal (normal ou affecté par des erreurs), ce modèle représente la transmission des données par moyen du modèle *Binary symmetric channel* ('*BSC*', Figure 4-8). Les probabilités *P* et *Q* déterminent l'occurrence de deux états du canal. Les probabilités d'erreurs $P_{normale}$ et P_{burst} permettent l'adaptation du modèle *BSC* aux deux états du canal. A titre d'exemple, l'application *Error Insertion Device* de [ITU191] utilise $P_{normale}$=0 et P_{burst}=0,5.

Les transmissions radio terrestres présentent parfois de fortes corrélations entre les apparitions d'erreurs. Ces transmissions souffrent temporairement des effets de la réception simultanée de plusieurs copies du même signal

décalées dans le temps ("*Fading*"). Par conséquent, les erreurs ne sont pas distribuées uniformément dans la séquence de symboles reçus, mais elles affectent des groupes de symboles consécutifs (Figure 4-7). Ainsi ils sont appelés *burst-errors*.

La présence des deux étages d'entrelacement dans le système de codage de canal *UMTS* [Ts25212] réduit la concentration des erreurs, en les répartissant dans tout le message[14]. Par conséquent, le signal affecté par ce type d'erreur est perçu par le décodeur de canal *UMTS* comme un message présentant des erreurs non-corrélées, mais possédant un débit d'erreurs (temporairement) plus élevé.

Modélisations simples du moyen de transmission: le *Binary symmetric channel* (*BSC*) et le *Additive white gaussian noise* (*AWGN*) *channel*.

Les deux modèles les plus simples, mais toutefois efficaces pour l'évaluation de la qualité de protection de canal, sont le *Binary symmetric channel* et le *Additive white gaussian noise channel*.

Le *Binary symmetric channel* modélise un canal de transmission qui ne présente pas d'effets de mémoire, à l'aide d'une situation de décision ferme (Figure 4-8).

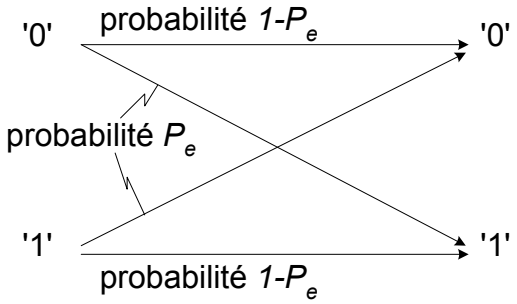

Figure 4-8: diagramme de transition du *Binary symmetric channel* (*BSC*) avec probabilité d'erreur P_e.

Cette modélisation est décrite par la probabilité de croisement P_e, décrite par

[14] La qualité de protection du codage convolutif est affaiblie en présence de concentrations des erreurs de transmission (*burst errors*, cf. Chapitre 2): les techniques d'entrelacement permettent de faire face à ces situations, évitant une baisse temporaire de la qualité de la protection.

$$P_e = \Pr\left[r_i \middle| x_i \right] r_i \neq x_i \text{ et } r_i, x_i \in \left[0,1\right]. \tag{4.3}$$

A partir de la probabilité P_e, le *BER* de la séquence de bits reçus est directement obtenu.

Le modèle *Additive white gaussian noise channel* est utile pour l'évaluation de systèmes exploitant une décision souple. A chaque instant i, ce modèle[15] modifie le symbole $\tilde{x}_i$ par l'addition d'un bruit blanc gaussien v_i:

$$\tilde{r}_i = \tilde{x}_i + v_i, \tag{4.4}$$

où le symbole $\tilde{x}_i$ est une représentation antipodale[16] du bit d'entrée x_i. Le bruit constitutif v_i suit la densité de probabilité gaussienne

$$p\left(v_i \middle| \tilde{x}_i \right) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} e^{-\frac{\left(v_i - \tilde{x}_i\right)^2}{2 \cdot \sigma^2}}, \tag{4.5}$$

caractérisée par la variance σ^2 (Figure 4-9) [Proa95]. Dans le contexte du décodage des codes convolutifs, il est d'usage d'exprimer le niveau de perturbation affectant la transmission par le rapport 'signal sur bruit par bit' donné par E_b/N_0 (*Signal-to-noise ratio per bit,* Figure 4-9). Cette unité de mesure considère l'énergie nécessaire à la transmission d'un bit d'information E_b par rapport à l'efficacité de la perturbation. Cette efficacité est représentée par la *densité spectrale latérale de la puissance* N_0 (*One-sided power spectral density*).

[15] Version à temps discret du modèle *Additive white gaussian noise channel*.

[16] Ce modèle considère que le symbole $\tilde{x}_i$ est le i-ème symbole livré par l'opération de démodulation, étant donné une transmission binaire du i-ème bit x_i.

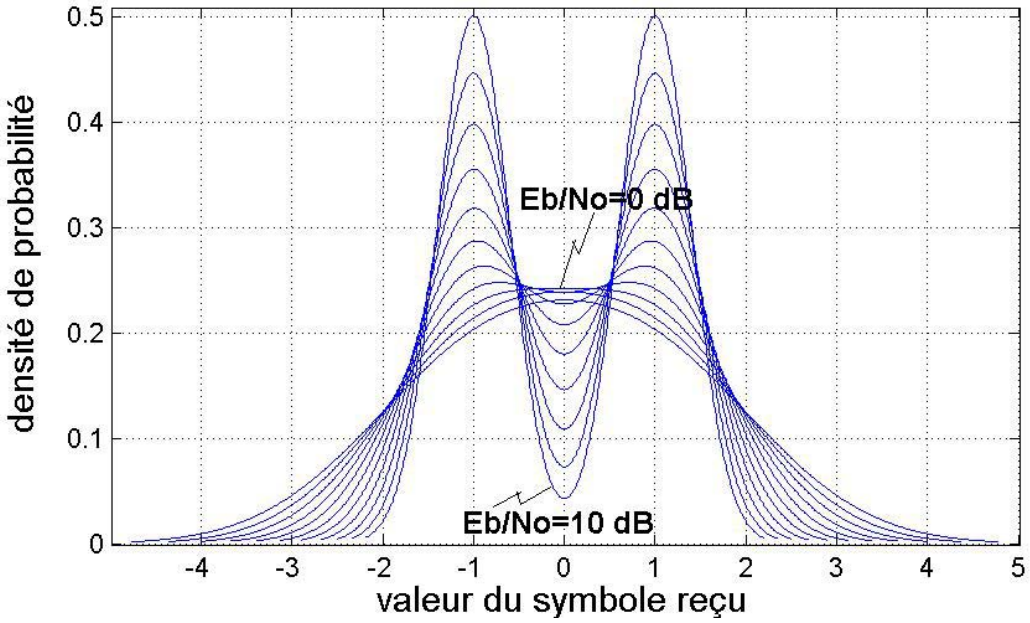

Figure 4-9: densité de probabilité des symboles affectés par l'addition d'un bruit blanc gaussien (rapport *signal sur bruit par bit* E_b/N_0, de 0 dB à 10 dB). Les symboles sont modulés en amplitude de manière antipodale (+1/-1 *PAM*).

4.3.2 Modèle de canal utilisé dans la suite des études

Le système de perturbation choisi pour les simulations de perturbations dues à la transmission sans fil se base sur l'addition d'un bruit blanc gaussien (Figure 4-10). Ce système permet d'analyser le comportement du décodeur selon les décisions de type ferme et pondéré, ainsi que de bénéficier des résultats mentionnés dans la littérature.
Par rapport aux paramètres du modèle *AWGN*, il faut remarquer que la relation existant entre la densité spectrale N_0 et la variance σ^2 est

$$\sigma^2 = \frac{N_0}{2}. \tag{4.6}$$

La relation entre l'énergie de transmission de chaque symbole E_s et celle de chaque bit d'information E_b est

$$E_s \cdot \text{nombre de symboles } n = E_b \cdot \text{nombre de bits d'information } k. \tag{4.7}$$

Ces deux relations (4.6) et (4.7) permettent d'exprimer le rapport entre la variance σ^2 et le rapport E_b/N_0 de la manière suivante:

$$\frac{E_b}{N_0} = \frac{E_s \cdot n}{2 \cdot k \cdot \sigma^2}. \tag{4.8}$$

Par rapport à la génération de valeurs de bruit blanc gaussien v_i, les implantations logicielles utilisent les techniques mentionnées dans [Pres92].

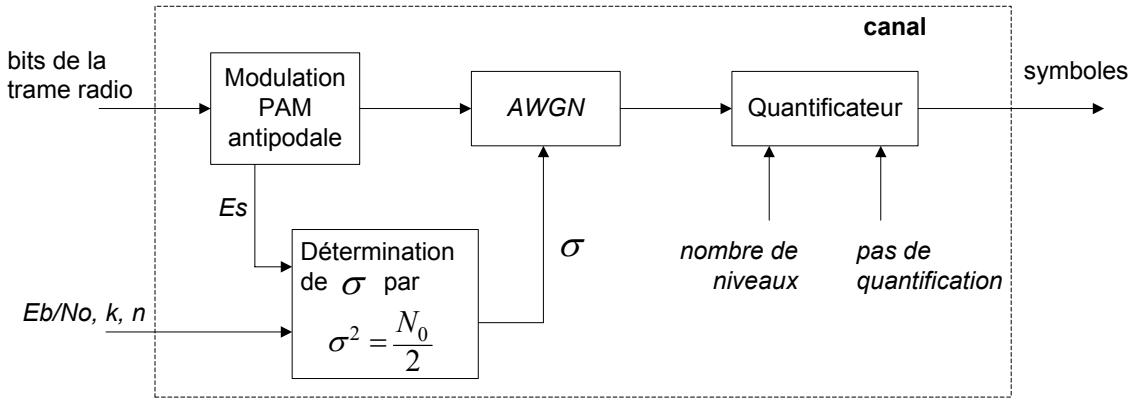

Figure 4-10: schéma du canal utilisé, se basant sur l'addition d'un bruit blanc gaussien (*AWGN*) à temps discret. La modulation antipodale *PAM* adoptée transforme le bit d'entrée prenant les valeurs '0' et '+1' en un bit prenant respectivement les valeurs '1' et '–1'.

4.4 Conclusions

Ce chapitre a traité la problématique de la transmission de données par réseaux cellulaires sans fil. Les facteurs techniques et environnementaux influençant la transmission ont été brièvement illustrés.
Suite à la description de ces facteurs, ce chapitre a discuté l'aspect de la modélisation du moyen de communication. En considérant les exigences d'évaluation et de simulation ainsi que les types d'erreurs reproductibles, deux modélisations simples ont été présentées: le modèle *Binary symmetric channel* (*BSC*) et le *Additive white gaussian noise* (*AWGN*). Ce dernier a été retenu pour modéliser le moyen de transmission servant à l'analyse des méthodes de décodage des Chapitres 6 et 7.

Références

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grand Bretagne, 2002.

[ITU191] International Telecommunication Union ITU-T, *Software Tools Library*, Série G: Transmission systems and Media, document ITU-T G.191-STL-96, 1996.

[Lust00] F. Lustenberger, "On the Design of Analog VLSI Iterative Decoders", Dissertation *ETH* No. 13879, *Serie in Signal and Information Processing: Volume 2*, Hartung Gorre, Konstanz, Allemagne, novembre 2000.

[Moul92] M. Mouly, M.–B. Pautet, *The GSM System for Mobile Communications*, Cell & Sys, Telecom Publishing, 1992.

[Pres92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, et B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, Cambridge University Press, 1992.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[R1-1218] 3GPP, *Preliminary link level results for HSDPA using multiple antennas,* Lucent Technologies, 3GPP TSG RAN WG1, document TSGR1#16(00)1218.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Ts25213] 3GPP, *Spreading and modulation (FDD)*, document 3GPP TS 25.213, version 3.1.1.

5 Méthodes pour le décodage des codes convolutifs

Ce chapitre présente les principes de décodage de codes convolutifs et en décrit les algorithmes les plus représentatifs.

Les notions de base nécessaires à la compréhension des principes et des algorithmes sus-mentionnés sont introduites. Puis la représentation numérique des symboles reçus par le décodeur, le critère d'optimisation utilisé pour le décodage et le type d'information de sortie souhaité sont traités.

Un aperçu historique des principales étapes ayant conduit aux méthodes de décodage actuelles est donné. Ces méthodes sont présentées selon leur critère d'optimisation, soit les méthodes recherchant le message le plus probable et celles estimant les symboles d'informations les plus probables. Dans la première classe de méthodes on trouve le "Décodage séquentiel", le "List Decoding" et "l'Algorithme de Viterbi". Parmi la seconde classe de méthodes, on trouve le "Symbol-by-symbol Maximum A Posteriori Algorithm", le "Max-Log-MAP", le "Log-MAP" et le "Soft Output Viterbi Algorithm".

Les algorithmes de Viterbi et de List Decoding constituent la base des méthodes améliorées "List Viterbi Algorithm" et "List Decoding intégrant la validation CRC" présentées au Chapitre 6, et sont à la base de la réalisation matérielle décrite au Chapitre 7. Les descriptions algorithmiques des autres méthodes existantes sont données en Annexe A.

Ce chapitre se termine par une comparaison et une discussion des caractéristiques des méthodes traitées.

5.1 Notions de base

Les points déterminant le choix et la mise au point d'une méthode de décodage sont la configuration et les exigences propres au système dans lequel le décodeur est appelé à travailler.
Il s'agit plus précisément:

- du type de représentation numérique des symboles reçus;
- du critère d'optimisation de la méthode de décodage;
- de la demande d'information supplémentaire requise pour évaluer la fiabilité de l'opération de décodage.

Ces trois aspects sont brièvement passés en revue ci-dessous.

5.1.1 Représentations numériques des symboles reçus

Les symboles reçus sont représentés numériquement en utilisant l'une des deux techniques suivantes:

- *Hard Decision*. La représentation de chaque symbole reçu est soumise à une décision 'ferme': les symboles sont quantifiés en utilisant la même représentation numérique que celle des symboles du message codé.
- *Soft Decision*. La représentation de chaque symbole reçu est soumise à une décision 'souple' ou 'pondérée': la représentation numérique dispose d'un nombre de niveaux de quantification supérieur par rapport à la représentation des symboles du message codé. L'utilisation d'une quantification moins rigide augmente le flux d'informations mis à disposition du décodeur.

Chaque type de décision présente ses avantages. La quantité d'informations plus importante, fournie au décodeur grâce à une décision souple, permet un décodage de qualité supérieure. L'application d'une décision ferme réduit les ressources de stockage et de calcul nécessaires aux opérations de traitement du signal.

5.1.2 Critères de décodage

L'objectif du codage de canal est l'utilisation de techniques de traitement du signal permettant de protéger les séquences de symboles d'informations ('les messages') contre les perturbations qu'elles subissent pendant leur transmission. Dans le cas particulier des codes convolutifs, la protection est obtenue en passant le message à travers un système ('le générateur du code' ou 'codeur') constitué d'un registre à décalage et d'un nombre fini de fonctions

linéaires algébriques (*Linear Finite-State Shift Register* [Proa95], voir la Sous-Section 2.3.2). Le déroulement de la procédure de codage convolutif peut être représenté soit par un diagramme en arbre (Figure 2-14), soit par un diagramme en treillis (Figure 2-16). Bien qu'il existe plusieurs principes et méthodes de décodage, tous les algorithmes exploitent un de ces deux diagrammes afin de modéliser l'opération de codage.
La succession des changements d'états de la mémoire du codeur convolutif est usuellement désigée par le terme "*chemin*" et chaque transition entre deux états par le terme "*branche*".

Faisant référence à la séquence d'opération du décodeur, on peut classifier les méthodes de décodage en deux classes:

1. les méthodes qui cherchent à déterminer le message produisant les symboles de sortie, ressemblant le plus aux symboles reçus;
2. les méthodes qui se concentrent sur l'identification de chaque symbole d'information le plus probable.

En raison de la diversité des procédés de décodage, l'aspect temporel des diagrammes en arbre et en treillis ne coïncide pas nécessairement avec le déroulement séquentiel de l'opération de décodage. Par conséquent, la notion de *temps* des représentations graphiques est remplacée par celle de *niveau de profondeur*.

5.1.3 Informations de décodage disponibles

L'accès aux informations de l'opération de décodage caractérise la configuration de travail du décodeur.

A l'origine, le décodage convolutif se chargeait de reconstruire le message par correction des éventuelles erreurs de transmission. Aujourd'hui, certaines applications et certains systèmes de décodage nécessitent des informations relatives à la fiabilité de la procédure de décodage de chaque symbole d'information. En présence de messages affectés par des erreurs de transmission, ces applications peuvent utiliser ces informations dans le but d'atténuer les dégâts dus aux erreurs. Des exemples sont le décodage 'joint' entre le décodage de canal et de source (*Source-channel Decoding*) et les techniques de mitigation d'erreurs affectant les messages (*Error concealement*).

La mise à disposition de ce type d'information n'est intrinsèque qu'aux méthodes se concentrant sur le décodage individuel de chaque symbole du message (Section 5.3).

5.2 Méthodes de décodage recherchant le message le plus probable

Après avoir introduit les notions de base, les méthodes les plus représentatives de chaque principe de décodage sont illustrées dans la suite du chapitre. Selon les principales étapes historiques ayant mené aux méthodes exploitées actuellement, les méthodes prise en considération sont le *décodage séquentiel*, le *List Decoding*, l'*Algorithme de Viterbi*, le *Symbol-by-symbol Maximum A Posteriori Algorithm* ("*MAP*"), le *Bidirectional Soft Output Viterbi Algorithm*, le *Max-Log-MAP* et le *Log-MAP*. Du point de vue historique, cette section présente tout d'abord les méthodes se basant sur le critère d'optimisation le plus ancien, à savoir la recherche du message le plus probable étant donné les symboles reçus. Pratiquement, ces méthodes consistent en une recherche du *chemin* dont les symboles de redondance ressemblent le plus à ceux reçus par le décodeur. Le message peut être ainsi construit en *décodant* ce chemin.

Les principes des trois méthodes les plus représentatives de cette classe sont présentées ici: le *décodage séquentiel*, le *List Decoding* et l'*Algorithme de Viterbi*. Comme la méthode *List Decoding* et l'algorithme de Viterbi sont traités dans la suite de ce rapport, ces méthodes sont décrites ici en détail. Les particularités algorithmiques de la méthode *décodage séquentiel* sont par contre décrites en Annexe *A*.

La présentation des algorithmes susmentionnés est précédée par celle de la fonction de métrique utilisée par ces algorithmes pour la recherche du message le plus probable.

Recherche du chemin à l'aide de la fonction de métrique

Les algorithmes recherchant le message le plus probable analysent la séquence des changements d'états S que la mémoire du codeur convolutif subit pendant l'opération de codage. L'objectif de ces algorithmes est l'identification de la séquence $\hat{S}$ statistiquement la plus probable, par rapport à la séquence de symboles reçus R. $\hat{S}$ est donnée par [Vale98]:

$$\begin{aligned}\hat{S} = \arg\left\{\max_S \Pr\left[\mathrm{S}|\mathrm{R}\right]\right\} &\overset{\text{Règle de Bayes}}{=} \arg\left\{\max_S \frac{\Pr\left[\mathrm{R}|\mathrm{S}\right]\cdot\Pr\left[\mathrm{S}\right]}{\Pr[\mathrm{R}]}\right\} \\ &= \arg\left\{\max_S \Pr\left[\mathrm{R}|\mathrm{S}\right]\cdot\Pr\left[\mathrm{S}\right]\right\}.\end{aligned} \tag{5.1}$$

La succession des symboles d'information, qui produisent une telle séquence est ensuite identifiée grâce à Ŝ.

Dans le but d'obtenir une solution à l'équation (5.1), ces algorithmes nécessitent une fonction mesurant la ressemblance entre la redondance générée par le chemin considéré *i* et la séquence de symboles reçue. Cette fonction est appelée *fonction de métrique*.
La fonction de métrique du *i*-ème chemin S_i est représentée dans ce document par le symbole $PM^{(S_i)}$. Elle permet de reformuler le critère d'optimisation (5.1) selon:

$$\hat{S} = \arg\left\{ \max_{S_i \in S, \forall i} PM^{(S_i)} \right\}, \tag{5.2}$$

où

$$PM^{(S_i)} \propto \Pr\left[\mathrm{S_i} | \mathrm{R} \right]. \tag{5.3}$$

Afin de réduire la dynamique des valeurs de métrique, le critère d'optimisation est souvent formulé dans le domaine logarithmique comme suit:

$$\hat{S} = \arg\left\{ \max_{S_i \in S, \forall i} \log(\Pr\left[\mathrm{S_i} | \mathrm{R} \right]) \right\} \tag{5.4}$$

et la fonction de métrique est reformulée de la manière suivante:

$$PM^{(S_i)} \propto \log(\Pr\left[\mathrm{S_i} | \mathrm{R} \right]). \tag{5.5}$$

Une propriété fondamentale de la fonction de métrique telle que définie ci-dessus, est la propriété récursive suivante:

$$PM^{(S_i)} = PM_{B-1}^{(S_i)} + \mu_B^{(S_i)} = \sum_{np=1}^{B} \mu_{np}^{(S_i)}.$$

$PM_{np}^{(S_i)}$: valeur de métrique ne considérant que les premières *np* transitions du chemin *i*; (5.6)

$\mu_{np}^{(S_i)}$: contribution à la mesure de fidélité de la transition entre les niveaux de profondeur *np* -1 et *np*

Cette propriété facilite (et permet) la réalisation de plusieurs méthodes de décodage, introduisant la notion de *métrique cumulée* $PM_{np}^{(S_i)}$ et de *métrique de branche* $\mu_{np}^{(S_i)}$. Le calcul de la métrique de branche doit tenir compte de la représentation numérique des symboles reçus ainsi que de la modélisation du moyen de communication. L'adoption d'une décision ferme permet l'établissement d'une fonction de métrique basée sur la distance de Hamming (*Hamming Distance*), alors qu'une décision souple implique l'adoption d'une distance euclidienne.

La fonction de métrique selon l'équation (5.6) permet d'envisager plusieurs principes de recherche du chemin le plus prometteur. On peut identifier trois genres de recherches différentes [Ande84], qui sont:

- Le genre *Depth-First*, où le chemin le plus prometteur est cherché en suivant le chemin montrant la valeur de métrique cumulée la plus favorable. En cas d'accumulation temporelle d'erreurs (*Burst errors*), le chemin présentant la métrique cumulée la plus favorable peut s'éloigner du chemin globalement le plus prometteur. Par conséquent, ce genre de recherche prévoit la possibilité de revenir en arrière sur le chemin parcouru, afin de reprendre un meilleur chemin: la procédure de recherche parcourt le diagramme en arbre de manière bidirectionnelle.
- Le genre *Breadth-First*, où la recherche est unidirectionnelle et synchronisée par rapport au niveau de profondeur du diagramme en arbre ou en treillis. Cette stratégie se base sur la comparaison des métriques cumulées de plusieurs chemins au même niveau de profondeur.
- Le genre *Metric-First*, où la recherche est unidirectionnelle dans le diagramme en arbre. Cette recherche est réglée par les valeurs des métriques cumulées des chemins déjà analysés. Au cours de la recherche, l'algorithme considère un nombre fini de chemins atteignant des niveaux de profondeur différents. Ce genre de recherche est le moins performant des trois [Ande84].

A l'aide des notions introduites ici, les algorithmes les plus représentatifs sont présentés.

5.2.1 Décodage Séquentiel

La première méthode utilisée pour le décodage des séquences protégées par des codes convolutifs, a été l'algorithme séquentiel proposé par Wozencraft en 1957.

Cette méthode a été reprise et modifiée par Fano [Proa95]. La stratégie adoptée par Fano consiste en la recherche du chemin le plus probable à l'intérieur du diagramme en arbre (Figure 2-14), en examinant une transition de mémoire à la fois. Ce type de recherche appartient au genre de recherche *depth-first* [Ande84] [Schl97]. La recherche du chemin le plus prometteur est conduite à l'aide d'une fonction de métrique *PM* (5.5) adaptée à ce genre de recherche (voir Annexe *A*, Section *A.1*).

Procédure pour la détection des "faux" chemins

Le genre de recherche *depth-first* est extrêmement sensible à l'accumulation temporelle d'erreurs de transmission. En effet, une accumulation rapide d'erreurs peut temporairement entraîner le rejet du chemin globalement le plus probable par rapport aux symboles reçus.

Le moyen choisi pour faire face à cet inconvénient est l'adoption d'un seuil dynamique de métrique qui évalue l'exactitude de la recherche du chemin le plus probable (Figure 5-1) [Joha99] [Proa95] [Schl97]. Dans le cas où le chemin choisi dépasserait ce seuil, le chemin sera suspecté d'être incorrect en raison d'une forte accumulation d'erreurs: la procédure de recherche de l'algorithme reviendra en arrière sur les nœuds précédents, en cherchant un nouveau chemin qui satisfasse la comparaison avec le seuil. Si aucun chemin conforme n'est trouvé, la valeur du seuil sera réduite et la procédure de recherche recommencera en utilisant le chemin qui a été suspecté d'être incorrect. Ce moyen permet de se prémunir d'une boucle de recherche infinie.

Le nombre d'itérations, la précision de la procédure de recherche dans les nœuds antécédents et, par conséquent, la qualité de correction d'erreurs fournie par l'algorithme dépendent strictement des modifications apportées à la valeur du seuil de référence. Pour cette raison, la procédure et les paramètres d'adaptation du seuil sont identifiées à l'aide de l'expérience accumulée ainsi que de nombreuses simulations.

Observations

La méthode de *décodage séquentielle* est caractérisée par l'exécution séquentielle et bidirectionnelle de la recherche du chemin le plus prometteur. La conséquence de cette stratégie de recherche est tout d'abord un temps de décodage variable et fortement influencé par le nombre et le type d'erreurs de transmission. Ensuite, la qualité de protection est aussi fonction du réglage des paramètres du seuil dynamique de référence. D'autre part, la fonction de métrique doit être établie de manière à permettre une comparaison équitable

entre chemins se situant à différents niveaux de profondeurs (Annexe *A*, Sous-section *A.1.1*).

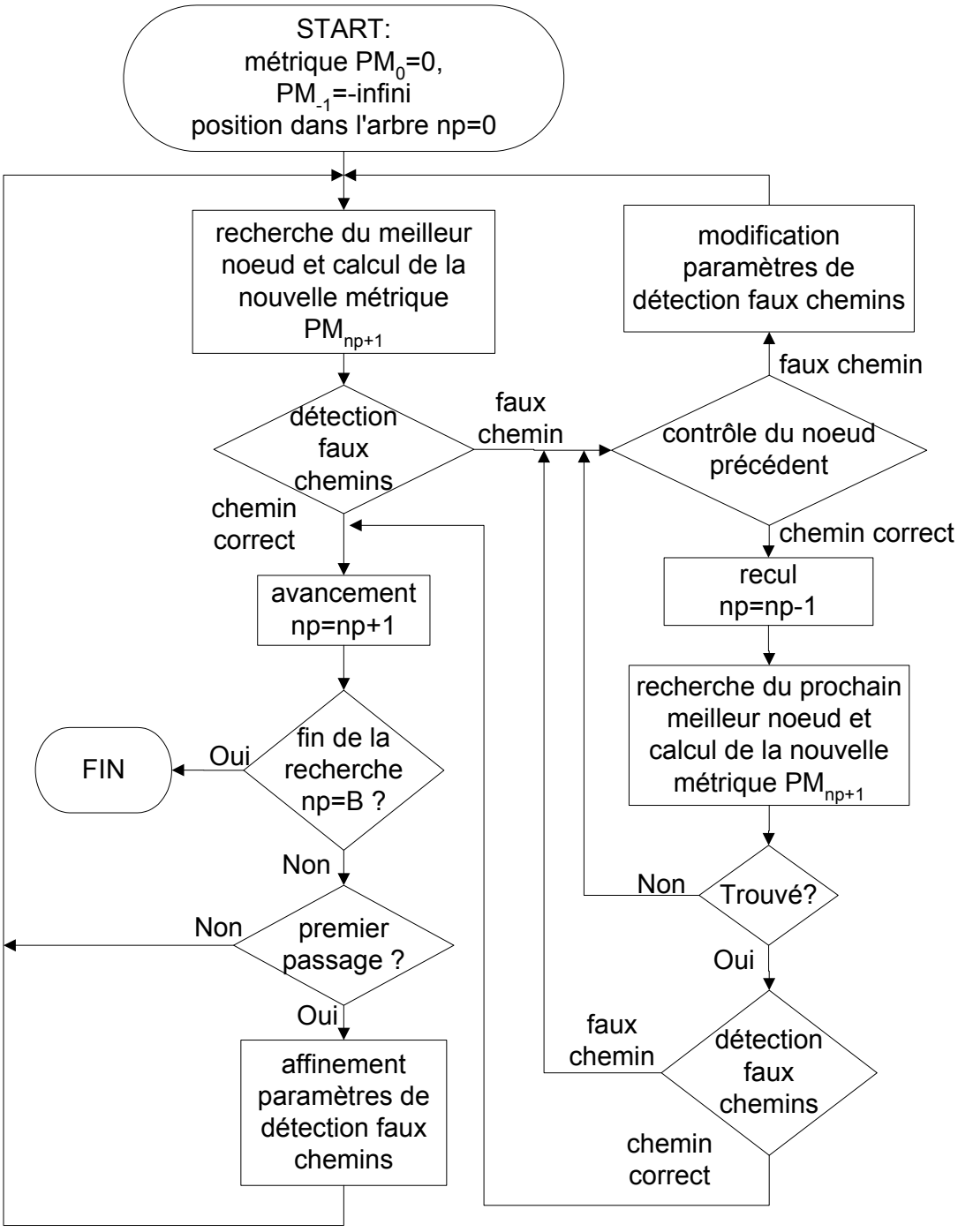

Figure 5-1: diagramme de flux simplifié de l'algorithme de Fano (adapté selon [Proa95]).

La variabilité et la non-prévisibilité du nombre d'opérations rendent ce genre de recherche inadaptée à un contexte de codage en temps réel. Cet

inconvénient peut être éliminé en changeant la nature de la recherche, en la remplaçant par le genre *breadth-first*. En effet, les algorithmes utilisant ce genre de recherche ont une complexité de calcul indépendante des conditions de transmission du signal, et un temps de décodage prévisible.

5.2.2 List Decoding

A la différence des méthodes de recherche précédentes, le genre de recherche *breadth-first* analyse et traite les chemins par groupes. Les algorithmes *List Decoding* [Joha99], *M-Algorithm* [Ande84] [Schl97] et *T-Algorithm* [Schl97] sont des méthodes qui exploitent ce genre de recherche.
Le principe de recherche de ces méthodes peut être ainsi partagé en deux parties. D'abord, un nombre de chemins les plus prometteurs est sélectionné parmi les chemins atteignant le niveau de profondeur traité. Puis, les chemins sélectionnés sont propagés au niveau de profondeur suivant.

Description de l'algorithme *List Decoding*

Etant donné une recherche de type *breadth-first*, l'analyse et le contrôle des chemins sont synchronisés par rapport au niveau de profondeur dans l'arbre de recherche. La possibilité de revenir sur les nœuds précédents n'est pas donnée (Figure 5-2).

L'algorithme *List Decoding* peut être réparti en quatre étapes principales:

1. Insertion du nœud de départ dans la liste contenant les chemins les plus prometteurs (initialisation de la liste).
2. Détermination des nouveaux chemins à partir des chemins contenus dans la liste des chemins les plus prometteurs. Mise à jour des métriques relatives cumulées.
3. Si la fin de l'arbre de recherche est atteinte, décodage du chemin possédant la meilleure métrique finale.
4. Si la fin de l'arbre n'est pas atteinte, effacement du contenu de la liste des chemins les plus prometteurs et sélection des L chemins présentant les métriques cumulées les plus favorables. Ces chemins sont ensuite inscrits dans la liste des chemins les plus prometteurs et la procédure de recherche retourne à l'étape 2.

Les chemins analysés ayant la même longueur, la fonction de métrique *PM* (5.5) peut ignorer tous les termes en commun à toutes les métriques. La fonction de métrique n'est ainsi formée que par l'accumulation des

contributions de chaque transition (*branche*) dans le diagramme en arbre, c'est-à-dire des métriques de branche μ_{np} (5.6):

$$PM_{\text{np}}^{(S_i)} = PM_{\text{np-1}}^{(S_i)} + \mu_{\text{np}}^{(S_i)} \tag{5.7}$$

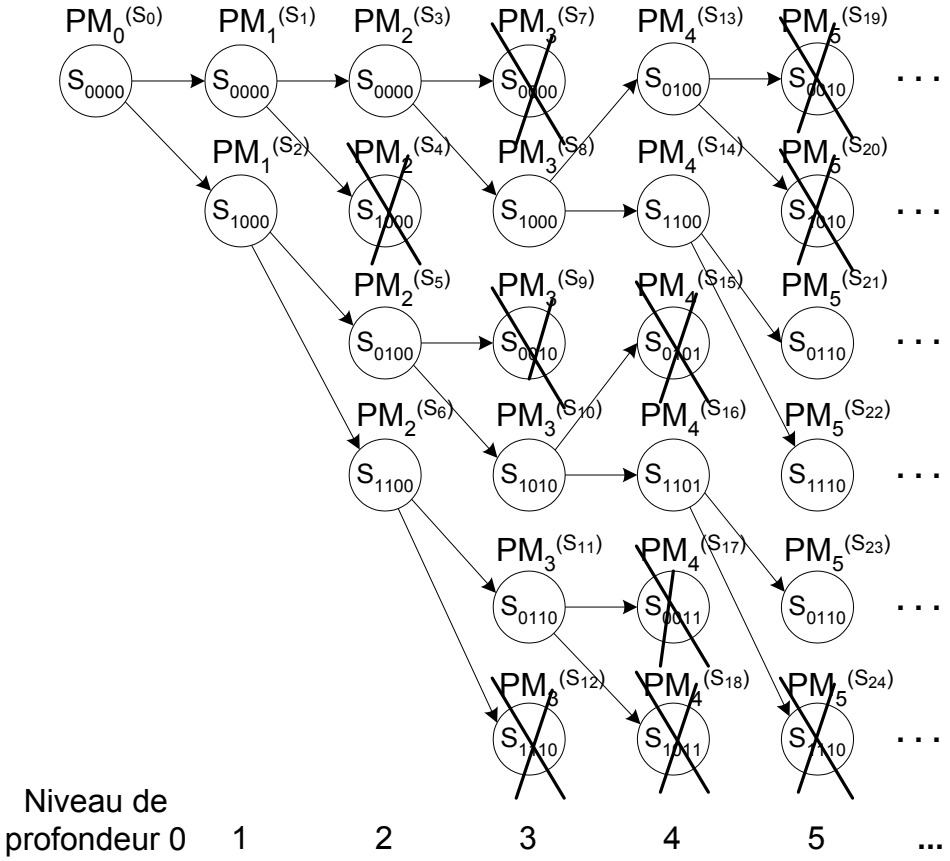

Figure 5-2 : exemple illustrant le genre de recherche de la méthode *List Decoding* ($L = 3$).

Importance du nombre *L* de chemins propagés

Le nombre L de chemins retenus à chaque transition est normalement défini à priori à l'exception de l'algorithme *T-Algorithm* qui sélectionne dynamiquement les chemins candidats. Cet algorithme sélectionne les chemins en se basant sur les différences existant entre les métriques cumulées [Schl97].

Le problème typique affectant ce type d'algorithme est la possibilité de rejet du chemin correct de la liste (temporaire/finale) des L chemins les plus favorables, à cause d'une concentration d'erreurs de transmission. Le rejet du chemin correct comporte un décodage livrant un message incorrect (Figure

5-3). Le nombre d'erreurs dans le message est fonction du nombre de transitions d'état différents entre le chemin choisi et le chemin correct.

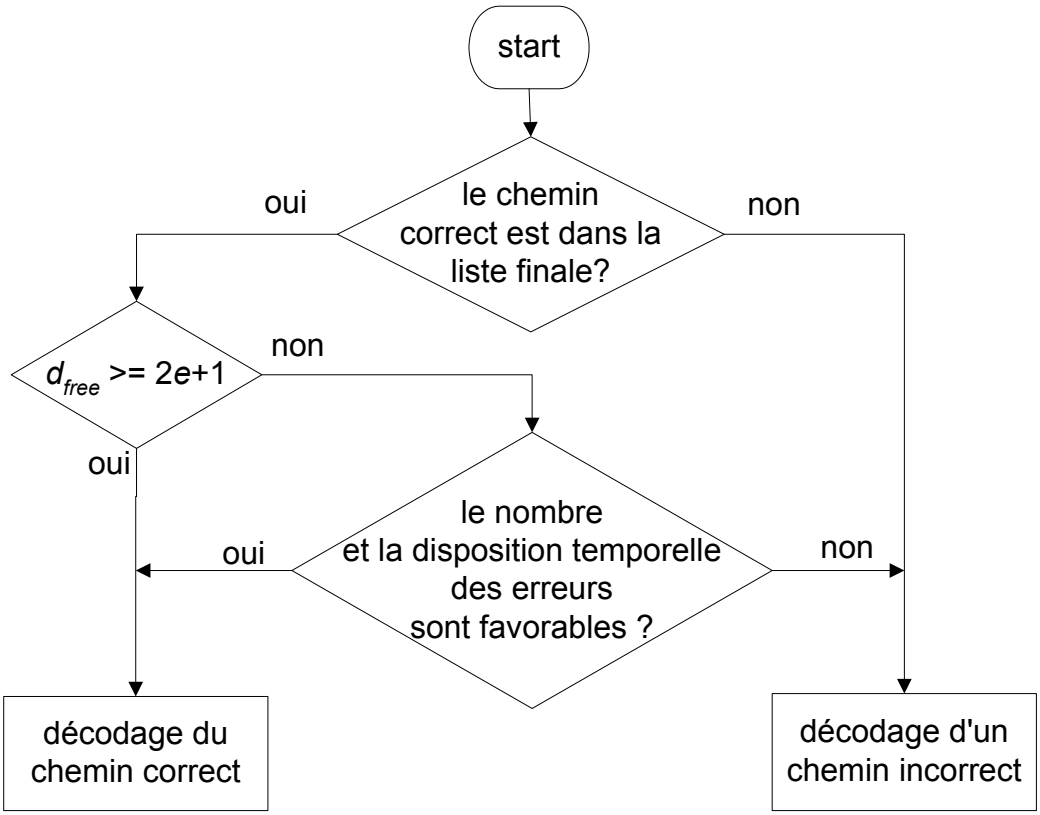

Figure 5-3 : représentation graphique des relations entre les paramètres influençant la performance du décodage, où *e* indique le nombre d'erreurs affectant la transmission du message, et où d_{free} indique la distance libre du code convolutif.

L'augmentation du nombre *L* permet de réduire l'occurrence du rejet due à une accumulation (temporaire et défavorable) d'erreurs de transmission. Toutefois, cette solution augmente la charge de calcul de la méthode de décodage, charge qui est fonction du nombre des chemins propagés et de l'utilisation d'opérations de sélection.

Observations

La qualité de décodage de l'algorithme *List Decoding* est fortement influencée par le possible rejet du chemin correct de la liste des chemins les plus prometteurs. L'occurrence dépend d'une part de la quantité et de la disposition temporelle des erreurs de transmission, et d'autre part du nombre *L* des chemins retenus dans la liste.

Ce paramètre L détermine non seulement la qualité de protection contre les erreurs de transmission, mais aussi la charge de calcul de la procédure de décodage. En effet, la nature de recherche de cet algorithme rend la charge de calcul indépendante du nombre et de la disposition temporelle des erreurs de transmission. A chaque niveau de profondeur, l'algorithme exécute la prolongation des L chemins contenus dans la liste, met à jour les $2^b \cdot L$ métriques cumulées et sélectionne ensuite les L meilleurs chemins[17].

De par sa charge de calcul indépendante de la longueur de contrainte du code et de par la nature des erreurs de transmission, l'algorithme *List Decoding* a été retenu comme noyau pour la nouvelle méthode software de décodage itérative appelée ici "*List Decoding intégrant la validation du CRC*", et décrite en Section 6.7. Cette méthode utilise de manière itérative les résultats de l'application de la méthode *List Decoding* et les informations supplémentaires fournies par la validation *CRC*.

5.2.3 Algorithme de Viterbi

L'algorithme de Viterbi a été présenté en 1967 par A. J. Viterbi comme décodeur des codes convolutifs [Vale98]. Le principe de cet algorithme consiste à estimer les transitions, qui se sont produites dans la mémoire du codeur convolutif pendant le codage. L'estimation se base sur le maximum de vraisemblance (*Maximum Likelihood*), fonction qui permet d'identifier le message globalement le plus probable [Hell71] [Forn73] [Thit93] [Proa95] [Schl97] [Vale98] [Joha99] [Vuce00].

L'importante contribution apportée par l'algorithme de Viterbi est la possibilité d'exploiter tout le potentiel de correction d'erreurs, mis à disposition par le code, sans devoir contrôler individuellement chacun des chemins possibles. En effet, la recherche du chemin globalement le plus probable par analyse des 2^{bB} chemins possibles devient rapidement impraticable avec l'augmentation du nombre des symboles du message.
Les méthodes existant avant l'algorithme de Viterbi se limitaient au contrôle d'un nombre important mais toutefois limité de chemins: la correspondance entre le chemin choisi et le chemin globalement le plus probable dépendait de plusieurs paramètres, dont le nombre et la disposition temporelle des erreurs de transmission, la qualité de protection offerte par le code convolutif et le nombre des chemins contrôlés par la méthode.

[17] b indique le nombre de bits formant le symbole d'information à l'entrée du codeur de canal.

Viterbi développe son algorithme en établissant cinq critères permettant d'identifier le chemin globalement le plus probable:

- la recherche du chemin globalement le plus probable par moyen d'une fonction de métrique;
- La modélisation de la procédure de codage convolutif par un processus de Markov (processus échantillonné dans le temps, avec un nombre d'états finis);
- un genre de recherche de type *breadth-first*;
- l'hypothèse d'un bruit 'sans mémoire' (*memoryless*) perturbant le canal;
- une procédure de recherche exploitant la structure du diagramme en treillis.

Établissement de la fonction de métrique

La modélisation de la procédure de codage par un processus de Markov offre la possibilité d'exploiter ses propriétés dont en particulier les deux suivantes [Vale98]:

- «*La probabilité qu'un processus de Markov soit dans un état particulier, étant donné tous les états précédents, est égale à la probabilité que le processus soit dans cet état, étant donnée l'état précédent*» (traduction) [Vale98]:

$$\Pr[s_{np+1} \,|\, s_0, s_1, \ldots, s_{np}] = \Pr[s_{np+1} \,|\, s_{np}]. \qquad (5.8)$$

- «*Dans un environnement perturbé par un bruit blanc, la probabilité de la np-ème observation particulière r_{np} étant donné la succession complète des transitions entre les états S, est égale à la probabilité de l'observation étant donné uniquement la transition entre la profondeur np-1 et np*» (traduction) [Vale98]. On a donc

$$\Pr[r_{np} \,|\, S] = \Pr[r_{np} \,|\, s_{np-1} \to s_{np}]. \qquad (5.9)$$

L'application de ceux deux propriétés à la formule (5.1) décrivant le critère d'optimisation, permet d'établir l'expression suivante [Vale98]:

$$\hat{S} = \arg\left\{ \max_{S} \prod_{np=1}^{B} \Pr[r_{np} \,|\, s_{np-1} \to s_{np}] \prod_{np=1}^{B} \Pr[s_{np} \,|\, s_{np-1}] \right\}. \qquad (5.10)$$

Le transfert de l'équation (5.10) dans le domaine logarithmique (5.4) permet de substituer le produit de probabilités par leur somme ce qui simplifie le calcul et réduit la dynamique des nombres. On a alors

$$\hat{S} = \arg\left\{ \max_{S} \sum_{np=1}^{B} \left(\log\left(\Pr[r_{np} \left| s_{np-1} \to s_{np} \right.] \right) + \log\left(\Pr[s_{np} \left| s_{np-1} \right.] \right) \right) \right\}. \quad (5.11)$$

A partir de l'équation (5.11), la fonction de métrique *PM* est introduite dans la description du critère d'optimisation:

$$\hat{S} = \arg\left\{ \max_{S} PM^{(S)} \right\} = \arg\left\{ \max_{S} \sum_{np=1}^{B} \mu_{np}^{(S)} \right\},$$

$$PM^{(S_i)} = \sum_{np=1}^{B} \mu_{np}^{(S_i)} \text{ et} \quad (5.12)$$

$$\mu_{np}^{(S_i)} = \log\left(\Pr[r_{np} \left| s_{np-1} \to s_{np} \right.] \right) + \log\left(\Pr[s_{np} \left| s_{np-1} \right.] \right),\ s_{np-1} \text{ et } s_{np} \in S_i$$

où les métriques finales $PM^{(S_i)}$ et de branche $\mu_{np}{}^{(S_i)}$ se réfèrent aux transitions du *i*-ème chemin.
L'exécution de la tâche de recherche (5.12) peut être allégée en ne considérant que les transitions effectivement possibles, transitions qui sont définies par la structure du codeur. Si tous les messages sont équiprobables, le second terme de la métrique de branche (5.12) n'apporte aucune contribution de discrimination: la formulation de la métrique de branche peut être ainsi simplifiée et donnée par:

$$\mu_{np}^{(S_i)} = \log\left(\Pr[r_{np} \left| s_{np-1} \to s_{np} \right.] \right). \quad (5.13)$$

Le maximum de vraisemblance devient alors l'unique paramètre servant à déterminer le meilleur chemin.
Les hypothèses de travail de l'algorithme présenté en 1967 par Viterbi considéraient une telle situation d'équiprobabilité [Vale98], qui est vérifiée lorsque la redondance des messages est réduite au minimum, ou admise lorsque les probabilités a priori $\Pr[s_{np}|s_{np-1}]$ ne sont pas connues.

Notion de *survivant*

L'introduction de la notion de *survivant* (*survivor*) décrite ci-après, permet de réduire le nombre d'opérations nécessaires à l'exploitation de tout le potentiel de correction d'erreurs à disposition. Cette notion se base sur le concept de *non-optimalité* (*nonoptimality*).

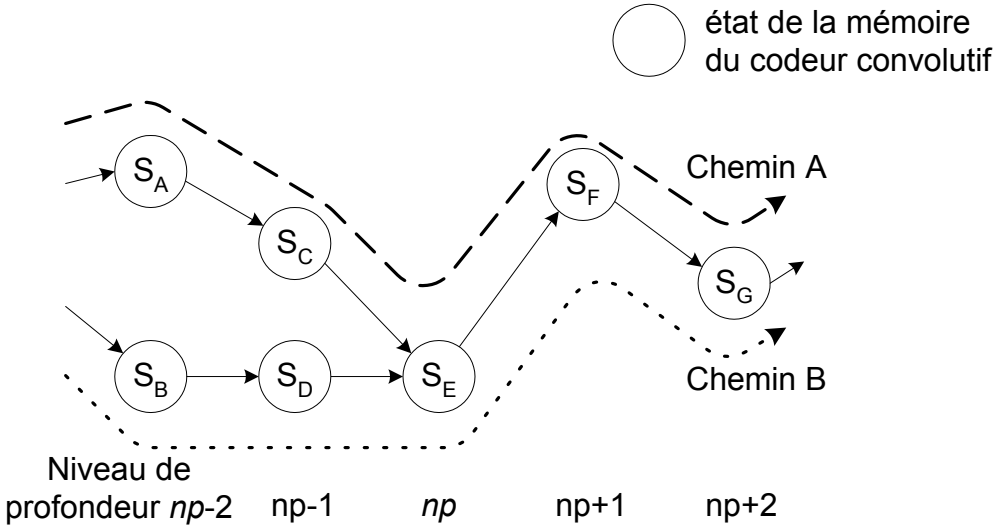

Figure 5-4 : représentation graphique de la convergence entre deux chemins

La notion de survivant peut être expliquée sur la base d'un exemple comprenant deux chemins *A* et *B* dans la situation suivante (Figure 5-4):

- les métriques cumulées de deux chemins *A* et *B*, sont comparées à la profondeur *np+2*;
- au niveau de profondeur *np+2*, le chemin *A* présente une métrique cumulée meilleure que le chemin *B*:

$$PM_{np+2}^{(A)} > PM_{np+2}^{(B)}; \tag{5.14}$$

- à partir du niveau de profondeur *np*, les deux chemins convergent en un même chemin, c'est-à-dire qu'à partir du niveau de convergence *np*, les deux chemins indiquent des états identiques de la mémoire du codeur.

A l'aide de la propriété récursive de la fonction de métrique *PM* (5.12), les métriques cumulées PM_{np+2} des deux chemins (5.14) peuvent être ainsi réécrites

$$PM_{np+2}^{(S_A)} = \sum_{prof=1}^{np} \mu_{prof}^{(S_A)} + \sum_{prof=np+1}^{np+2} \mu_{prof}^{(S_A)}$$

$$PM_{np+2}^{(S_B)} = \sum_{prof=1}^{np} \mu_{prof}^{(S_B)} + \sum_{prof=np+1}^{np+2} \mu_{prof}^{(S_B)}. \qquad (5.15)$$

Cette formulation distingue la contribution des métriques de branches, par rapport au nœud de convergence des deux chemins. On peut ainsi constater que les branches qui suivent le nœud de convergence apportent la même contribution aux deux métriques cumulées $PM_{np+2}^{(S_A)}$ et $PM_{np+2}^{(S_B)}$:

$$\sum_{prof=np+1}^{np+2} \mu_{prof}^{(S_A)} = \sum_{prof=np+1}^{np+2} \mu_{prof}^{(S_B)}. \qquad (5.16)$$

A partir de ce nœud de convergence, la relation existant entre les deux métriques cumulées ne change pas parce que les deux chemins suivrent le même parcours (propriétés des processus de Markov, (5.8) et (5.9)):

$$PM_{np}^{(S_A)} > PM_{np}^{(S_B)} \quad \Leftrightarrow \quad PM_{np+x}^{(S_A)} > PM_{np+x}^{(S_B)}, \ \mathrm{x} = 1, 2. \qquad (5.17)$$

Cette relation reste inaltérée en choisissant un autre future parcours[18]: après le nœud de convergence, les métriques cumulées du future parcours apportent la même contribution aux deux chemins. Par conséquent, au niveau de profondeur *np*, l'opération de recherche sait que le chemin convergeant le moins performant 'S_B' ne pourra jamais coïncider avec le chemin globalement le plus probable.

Cette propriété permet d'anticiper la sélection entre deux chemins concurrents à la profondeur où les deux chemins se réunissent pour la première fois. Le théorème de la non-optimalité (*theorem of nonoptimality path*) [Joha99] [Schl97] définit ainsi que: «*la procédure de réunion des nœuds, qui correspondent à des états identiques de l'encodeur, et successivement la suppression des chemins avec les métriques cumulées les moins favorables n'éliminent jamais le chemin avec la ressemblance maximale*»[19] (traduction) [Schl97]. De point de vue de la nomenclature, le chemin retenu après la comparaison est appelé chemin *survivant*.

[18] Parcours qui démarre depuis le nœud de convergence des deux chemins.

[19] Selon ce théorème, dans le cas où les chemins convergeant en un même nœud présenteraient une métrique identique, l'un des deux chemins peut être éliminé de manière aléatoire.

L'application systématique de cette procédure de sélection anticipée réduit considérablement la complexité de calcul de procédure de recherche et rend ainsi faisable l'identification du chemin globalement le plus probable.

Description de l'algorithme

La sélection du chemin survivant, en considérant les divers états de mémoire possibles, permet le remplacement d'une recherche basée sur un diagramme en arbre par une recherche exploitant le diagramme en treillis. En représentant les états de mémoires par les nœuds de la représentation, le diagramme en treillis facilite l'exécution de l'opération de sélection.

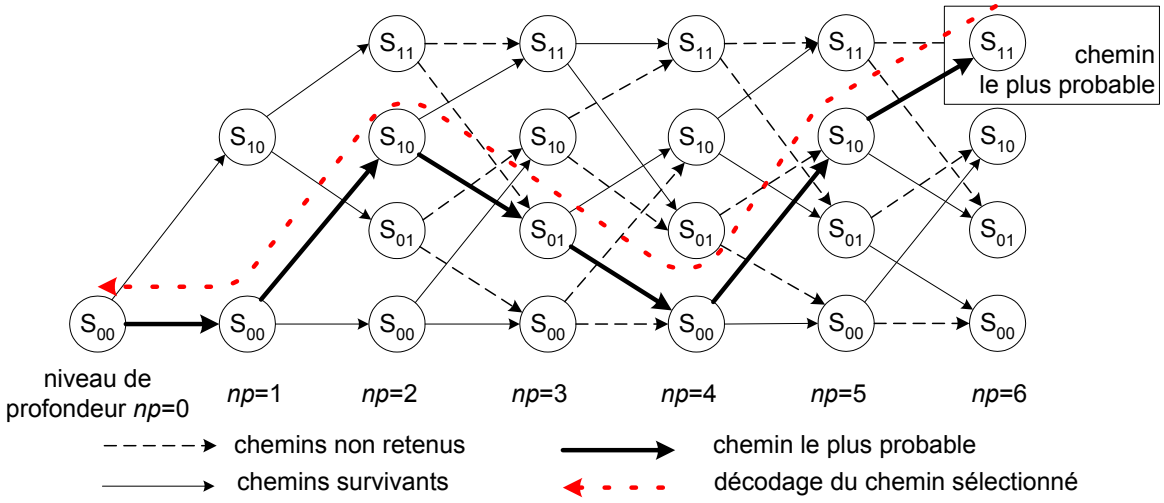

Figure 5-5 : exemple de décodage d'un message de 6 bits par l'algorithme de Viterbi, message qui a été protégé avec un code convolutif, possédant une longueur de contrainte de 3.

Le déroulement de l'algorithme de Viterbi implique ainsi les étapes suivantes (Figure 5-5):

1. Les métriques cumulées de tous les états sont initialisées selon les connaissances (a priori) des conditions de départ du processus analysé.
2. La valeur de la variable *np*, qui indique le niveau de profondeur dans le treillis, est initialisée à zéro.
3. Pour chaque nœud à la profondeur *np+1* :
    a. les chemins convergeant en ce nœud sont identifiés;
    b. leurs métriques sont mises à jour à l'aide des métriques de branche;
    c. le chemin le plus probable (survivant) est déterminé;
    d. la métrique cumulée du chemin survivant est assignée au nœud;

e. les informations nécessaires à la reconstruction du chemin survivant sont sauvegardées.

4. Lorsque la fin du treillis de recherche est atteinte, l'algorithme décode le chemin avec la meilleure métrique, selon les informations sauvegardées.
5. Sinon, le niveau de profondeur *np* est incrémenté de 1.
6. On revient à l'étape 3.

Observations

La propriété principale de cet algorithme est d'offrir une analyse efficace et exhaustive de tout l'espace de codage, grâce à la notion de survivant. Les ressources nécessaires à cette analyse, qui doit prendre en compte les 2^{bB} chemins possibles, sont réduites à:

- $B \cdot 2^{b} \cdot 2^{(K-1)}$ mises à jour de métriques cumulées;
- $B \cdot 2^{(K-1)}$ sélections du chemin survivant.

L'indépendance de la complexité de calcul par rapport au nombre et à la disposition temporelle des erreurs de transmission est assurée par le genre de recherche *breadth-first*. Par contre, la longueur de contrainte *K* du code détermine la charge de calcul pour le traitement de chaque bit d'information.

La notion de survivant peut être aussi servir à l'amélioration de la qualité de décodage des méthodes *List Decoding*. La performance de l'algorithme *List Decoding* est en effet perturbée par la conservation possible de chemins présentant un même état de mémoire (voir l'exemple de la Figure 5-2, au 5ème niveau de profondeur 5). Cette situation perturbatrice dégrade l'efficacité de la recherche du chemin le plus probable, en réduisant le nombre de chemins effectivement utiles à cette recherche. L'élimination des chemins inutiles implique une augmentation de la complexité de calcul de la méthode, non seulement pour l'exécution de l'opération de sélection, mais aussi en raison d'un type de recherche basée sur le diagramme en arbre.

L'algorithme de Viterbi, grâce à son analyse efficace et exhaustive de tout l'espace de codage, sera un élément crucial des travaux décrits dans les Chapitres 6 et 7.
Le sujet de l'implantation software (Chapitre 6) sera introduit par l'analyse de cet algorithme qui est massivement utilisé dans les systèmes de la *2G*. L'algorithme de Viterbi est ensuite à la base d'une des deux méthodes itératives de décodage software qui seront proposées dans la suite de l'étude du sixième chapitre. Cette méthode, le '*List Viterbi Algorithm*', utilise de manière itérative les informations fournies par l'algorithme de Viterbi et par le

codage supplémentaire *CRC*, améliorant la qualité de protection contre les erreurs de transmission.
Le Chapitre 7 traitera le sujet de la réalisation *ASIC* d'une méthode de décodage de codes convolutifs, basée entièrement sur l'algorithme de Viterbi. Le système de décodage de base sera ainsi décrit en décomposant cet algorithme en ses éléments constituants.

5.3 Méthodes de décodage estimant les symboles les plus probables

Après avoir introduit les méthodes se concentrant sur le message le plus probable, cette section présente les méthodes agissant au niveau des symboles d'information du message.

La méthode pionnière de cette classe de décodeurs est l'algorithme *Symbol-by-symbol Maximum A Posteriori Algorithm* ("*MAP*"). Les avantages fournis par cet algorithme sont la minimisation de la probabilité d'erreur de chaque symbole du message et la mise à disposition d'une estimation de la fiabilité du décodage. Toutefois, cette première méthode souffre de problèmes de représentation numérique des variables, de ressources de stockage et d'une charge de calcul importante. Pour réduire ces inconvénients, différents algorithmes ont été ultérieurement proposés. Les algorithmes *Max-Log-MAP Algorithm*, *Log-MAP Algorithm* et *Soft Output Viterbi Algorithm* ("*SOVA*") sont des exemples représentatifs.

Dans cette section, on illustre ainsi les principes de décodage des méthodes *MAP*, *Max-Log-MAP Algorithm*, *Log-MAP Algorithm* et *Bidirectional SOVA*. Les détails algorithmiques de ces méthodes sont disponibles dans l'Annexe *A*.

Critère d'optimisation

Le critère d'optimisation de cette classe de méthodes se base sur l'estimation de chaque bit (b=1) ou symbole (b>1) d'information du message, étant donné la séquence de symboles reçus R:

$$\widehat{\text{info Bit}}_{np} = \arg\left\{ \max_{\text{info Bit}_{np}} \Pr\left[\text{info Bit}_{np} | \text{R} \right] \right\}, \qquad (5.18)$$

info Bit_{np} : np-ème symbole d'information.

Ce critère détermine les symboles les plus probables, indépendamment de la corrélation existant entre les symboles reçus et ceux générés par le message entier.

La résolution de l'équation (5.18) implique la détermination des probabilités de décodage de chaque symbole du message, informations qui peuvent être utiles à l'opération suivant le décodage convolutif. Cette classe d'algorithmes ne fournit ainsi pas le message, mais une estimation de la fiabilité du décodage de chaque symbole.
Dans le contexte d'un codage binaire (b=1), cette estimation est livrée au moyen de la *fiabilité* Λ:

$$\Lambda\left(\text{info bit}_{\text{np}}\right)=\log\frac{\Pr\left[\text{info bit}_{\text{np}}=1\middle|\text{S}\right]}{\Pr\left[\text{info bit}_{\text{np}}=0\middle|\text{S}\right]}, \tag{5.19}$$

qui se base sur la probabilité *a-posteriori* (*APP*) des symboles transmis:

$$\Pr\left[\text{info bit}_{\text{np}}=\text{i}\middle|\text{S}\right],\ \ i=0,1\ . \tag{5.20}$$

Le message peut ainsi être extrait en regardant le signe de la fiabilité Λ:

$$\text{info bit}_{\text{np}}=\begin{cases}1, & \text{si } \Lambda\left(\text{info bit}_{\text{np}}\right)\geq 0\\ 0, & \text{autrement}\end{cases}. \tag{5.21}$$

5.3.1 Symbol-by-symbol Maximum A Posteriori Algorithm

L'algorithme *symbol-by-symbol Maximum A Posteriori* ('*MAP*') a été présenté formellement en 1974 dans la publication [Bahl74], comme une solution alternative pour le décodage de codes convolutifs [Vale98].
Contrairement aux méthodes précédentes, le critère de décodage de cet algorithme est la minimisation de la probabilité d'erreur de chaque symbole du message (5.18). L'algorithme est ainsi capable de fournir en plus une estimation de la fiabilité du décodage, qui se base sur la probabilité a posteriori du bit concerné [Bahl74] [Proa95] [Robe95] [Schl97] [Vale98] [Vuce00].

Bien que le nom de la méthode rappelle la fonction de maximisation, cette fonction mathématique n'est pas à la base de la méthode. Par conséquent, dans

certaines publications, la méthode est citée sous d'autres noms, souvent sous le terme *A Posteriori Probability Algorithm* [Vale98]. Afin de maintenir une cohérence avec la plupart des publications, dans la suite de ce rapport cette méthode continuera à être mentionnée avec son nom original.

Objectif de la méthode pionnière *MAP*

En modélisant le problème au moyen d'un processus de Markov (échantillonné dans le temps, avec un nombre fini de niveaux), la méthode générale permet l'estimation des probabilités a posteriori ('*APP*') des transitions

$$\Pr\left[s_{np-1} = m',\ s_{np} = m \middle| R \right] = \frac{\Pr\left[s_{np-1} = m',\ s_{np} = m, R \right]}{\Pr\left[R \right]} \tag{5.22}$$

et des probabilités des états du processus

$$\Pr\left[s_{np} = m \middle| R \right] = \frac{\Pr\left[s_{np} = m,\ R \right]}{\Pr\left[R \right]} \tag{5.23}$$

Dans le cas particulier du décodage convolutif, l'objectif poursuivi par la méthode *MAP* est l'estimation de la probabilité

$$\Pr\left[\text{info bit}_{\text{np}} = i \middle| R \right] = \frac{\Pr\left[\text{info bit}_{\text{np}} = i, R \right]}{\Pr\left[R \right]}, \tag{5.24}$$

probabilité qui est ensuite délivrée au moyen de la valeur de fiabilité Λ. Ainsi, au moyen de la probabilité $\sigma(.,.)$

$$\begin{aligned} \sigma_{np}\left(m', m\right) &= \Pr\left[s_{np-1} = m',\ s_{np} = m,\ R \right] \\ &= \alpha_{np-1}\left(m'\right) \cdot \gamma_{np}\left(m', m\right) \cdot \beta_{np}\left(m\right), \end{aligned} \tag{5.25}$$

où

$$\alpha_{np}\left(m\right) = \Pr\left[s_{np} = m,\ R_1^{np} \right] \tag{5.26}$$

$$\beta_{np}\left(m\right) = \Pr\left[R_{np+1}^{B} \middle| s_{np} = m \right] \tag{5.27}$$

$$\gamma_{np}\left(m', m\right) = \Pr\left[s_{np} = m,\ r_{np} \middle| s_{np-1} = m' \right], \tag{5.28}$$

l'algorithme *MAP* accomplit sa tâche en estimant la fiabilité

$$\Lambda\left(\text{info bit}_{\text{np}}\right)=\frac{\Pr\left[\text{info bit}_{\text{np}}=1,R\right]}{\Pr\left[\text{info bit}_{\text{np}}=0,R\right]}=\frac{\sum_{\substack{\text{transition } (m',m)\\ \text{impliquant } \textbf{info bit}_{\textbf{np}}=1}}\sigma_t(m',m)}{\sum_{\substack{\text{transition } (m',m)\\ \text{impliquant } \textbf{info bit}_{\textbf{np}}=0}}\sigma_t(m',m)} \quad (5.29)$$

Le calcul de la probabilité σ(.,.) (5.25) bénéficie des propriétés itératives des probabilités α(.) (5.26)

$$\alpha_{np}\left(m\right)=\sum_{m'}\alpha_{np-1}\left(m'\right)\cdot\gamma_{np}\left(m',m\right) \quad (5.30)$$

et β(.) (5.27)

$$\beta_{np}\left(m\right)=\sum_{m'}\beta_{np+1}\left(m'\right)\cdot\gamma_{np+1}\left(m,m'\right). \quad (5.31)$$

Les détails de la dérivation mathématique ainsi que le déroulement algorithmique de la méthode *MAP* sont disponibles dans l'Annexe *A* (Section *A.2*).

Par rapport à la représentation graphique de la procédure de décodage, cette méthode utilise un diagramme en treillis (Figure 5-6), similairement à l'algorithme de Viterbi. A chaque nœud du diagramme en treillis est assignée la probabilité *APP* d'un état

$$\Pr\left[s_{np}=m\middle|R\right] \quad (5.32)$$

et à chaque branche la probabilité *APP* d'une transition de mémoire

$$\Pr\left[s_{np-1}=m',\ s_{np}=m\middle|R\right]. \quad (5.33)$$

Observations

L'analyse du déroulement de l'algorithme *MAP* met tout de suite en évidence l'importante charge de calcul et la forte demande de ressources de stockage ainsi qu'un décodage qui ne supporte pas des messages infinis (*block-oriented*).

Par conséquent, des méthodes ont été ensuite proposées afin de pallier à ces inconvénients, tout en gardant le même critère d'optimisation. Parmi ces méthodes, on trouve les algorithmes *Max-Log-MAP* et *Log-MAP*, qui

proposent des solutions réduisant la complexité de calcul [Robe95] [Vale98] [Vuce00].

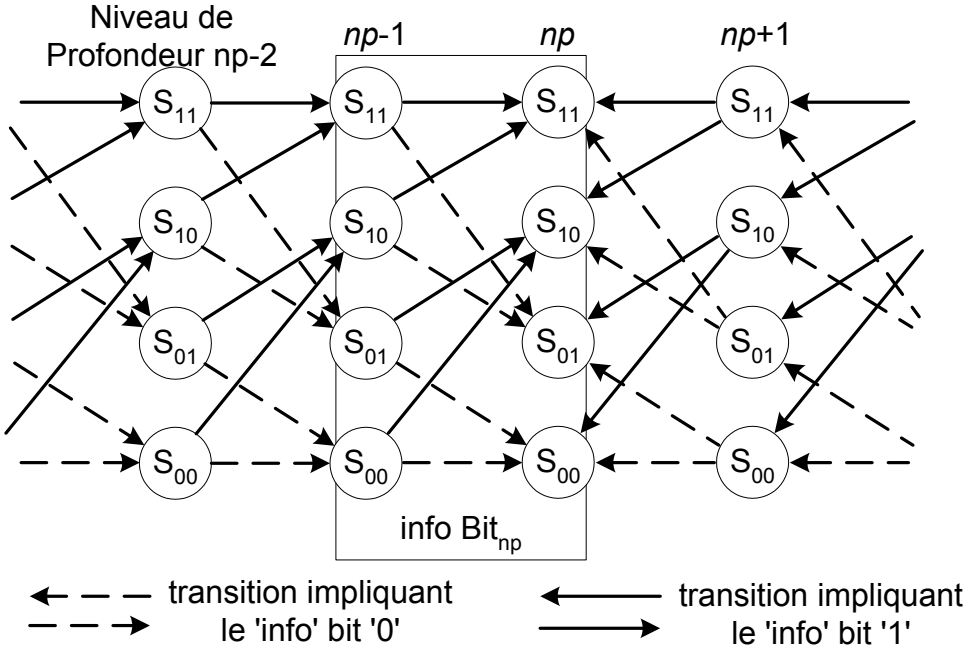

Figure 5-6: vue d'ensemble de la procédure pour l'estimation de la fiabilité du décodage du *np*-ème bit par les algorithmes *MAP* et *Log-MAP*.

5.3.2 Algorithme 'Max-Log-MAP'

L'algorithme *MAP* est indiscutablement la solution optimale pour l'estimation des probabilités des états et des transitions de mémoire du codeur convolutif [Robe95]. Cependant, il souffre de sévères inconvénients [Vale98]:

- Une représentation numérique des probabilités qui doit couvrir une vaste dynamique de valeurs.
- Une utilisation intensive des ressources de stockage et de calcul.

L'algorithme *Max-Log-MAP* propose de réduire ces inconvénients.

Le problème de la dynamique des nombres est résolu en exécutant les estimations des probabilités dans le domaine logarithmique [Robe95] [Vale98] [Vuce00]. Les probabilités $\alpha_t(m)$, $\beta_t(m)$, $\gamma_t(m',m)$ et $\gamma_t^i(m',m)$ sont ainsi remplacées par les probabilités équivalentes $\overline{\alpha_t}(m), \overline{\beta_t}(m)$, $\overline{\gamma_t}(m',m)$ et $\overline{\gamma_t^i}(m',m)$ dans le domaine logarithmique:

$$\overline{\gamma_t^i}\left(m',m\right)=\log\left(\gamma_t^i\left(m',m\right)\right) \tag{5.34}$$

$$\overline{\alpha_t}\left(m\right)=\log\left(\alpha_t\left(m\right)\right)=\log\left(\sum_{m'}e^{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t}\left(m',m\right)}\right) \tag{5.35}$$

$$\overline{\beta_t}\left(m\right)=\log\left(\beta_t\left(m\right)\right)=\log\left(\sum_{m'}e^{\overline{\beta_{t+1}}\left(m'\right)+\overline{\gamma_{t+1}}\left(m,m'\right)}\right). \tag{5.36}$$

La fiabilité Λ peut être ainsi reformulée:

$$\Lambda\left(\text{info bit}_\text{t}\right)=\log\frac{\sum_{(m',m)}e^{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t^1}\left(m',m\right)+\overline{\beta_t}\left(m\right)}}{\sum_{(m',m)}e^{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t^0}\left(m',m\right)+\overline{\beta_t}\left(m\right)}} \tag{5.37}$$

Ensuite, si on considère l'approximation

$$\log\left(e^{\delta_1}+e^{\delta_1}+\ldots+e^{\delta_n}\right)\approx\max_{i\in\{1,2,..,n\}}\delta_i, \tag{5.38}$$

les expressions (5.35) et (5.36) peuvent être simplifiées d'une manière analogue

$$\overline{\alpha_t}\left(m\right)\approx\max_{m'}\left\{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t}\left(m',m\right)\right\} \tag{5.39}$$

$$\overline{\beta_t}\left(m\right)\approx\max_{m'}\left\{\overline{\beta_{t+1}}\left(m'\right)+\overline{\gamma_{t+1}}\left(m,m'\right)\right\}, \tag{5.40}$$

ce qui permet une réduction de la charge de calcul [Robe95] [Vale98] [Vuce00]. L'évaluation de la fiabilité Λ (5.37) peut aussi être simplifiée:

$$\begin{aligned}\Lambda\left(\text{info bit}_\text{t}\right)\approx&\max_{(m',m)}\left\{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t^1}\left(m',m\right)+\overline{\beta_t}\left(m\right)\right\}\\&-\max_{(m',m)}\left\{\overline{\alpha_{t-1}}\left(m'\right)+\overline{\gamma_t^0}\left(m',m\right)+\overline{\beta_t}\left(m\right)\right\}\end{aligned} \tag{5.41}$$

Observation

En analysant l'expression (5.41), on en constate que le nombre des chemins participant à la mesure de la fiabilité est réduit et que l'algorithme ressemble à celui de Viterbi.

Pour l'estimation de la fiabilité de chaque bit, l'algorithme *Max-Log MAP* utilise seulement deux chemins, dont la sélection est analogue au principe du chemin *survivant* de l'algorithme de Viterbi. L'un des deux chemins coïncide toujours avec le chemin globalement le plus prometteur, ce qui détermine le signe de l'estimation Λ [Vuce00] (Figure 5-7). Par conséquent, le message extrait par les signes des fiabilités Λ coïncide avec celui livré par l'algorithme de Viterbi.

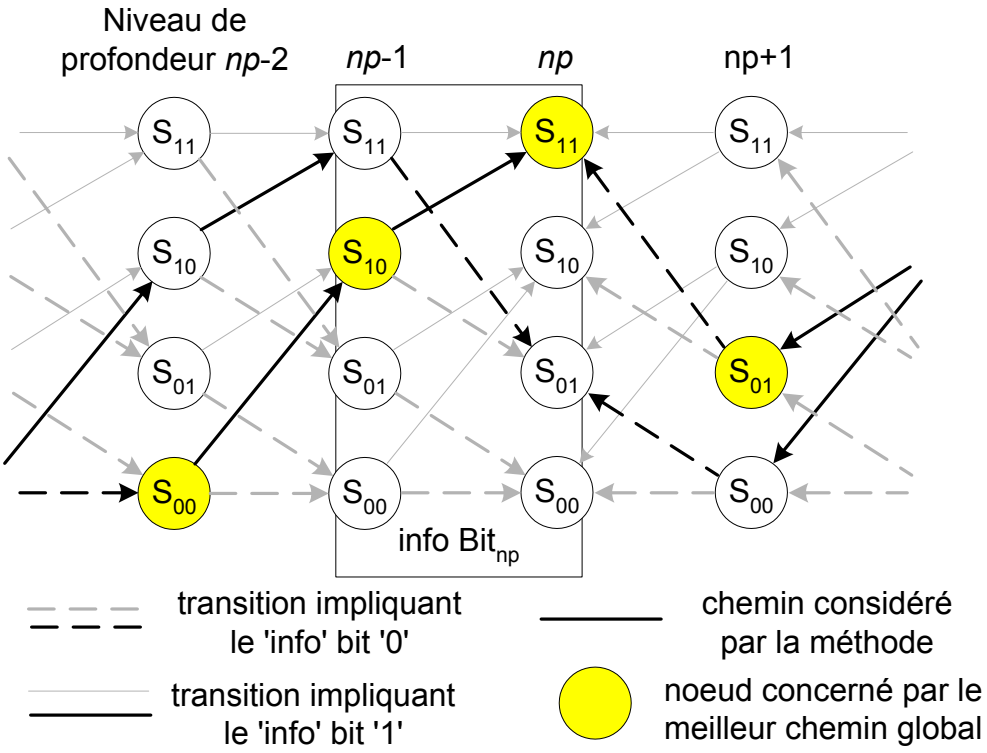

Figure 5-7: vue d'ensemble de la procédure pour l'estimation de la fiabilité du décodage du *np*-ème bit par l'algorithme *Max-Log-MAP*.

5.3.3 Algorithme 'Log-MAP'

La réduction de la complexité de calcul de l'algorithme *Max-Log-MAP*, obtenue à l'aide de l'utilisation des approximations, est contre-balancée par la dégradation de la qualité de l'estimation Λ. Pour parer à ce problème, l'algorithme *Log-MAP* utilise la notion du '*Jacobian Algorithm*' de manière à améliorer la qualité des approximations.

Le *Jacobian Algorithm* comporte les relations suivantes

$$\begin{aligned}\ln\left(e^{\delta_1}+e^{\delta_2}\right) &= \max\left(\delta_1,\delta_2\right)+\ln\left(1+e^{-\left|\delta_2-\delta_1\right|}\right)\\ &= \max\left(\delta_1,\delta_2\right)+\mathrm{f}_c\left(\left|\delta_1-\delta_2\right|\right)\end{aligned} \tag{5.42}$$

et

$$\ln\left(\underbrace{e^{\delta_1}+e^{\delta_2}+\ldots}_{e^{\delta}}+e^{\delta_n}\right)=\ln\left(e^{\delta}+e^{\delta_n}\right)=\max\left(\delta,\delta_n\right)+\mathrm{f}_c\left(\left|\delta-\delta_n\right|\right), \tag{5.43}$$

où $\mathrm{f}_c(.)$ est une fonction de correction [Robe95] [Vuce00].

Observations

L'utilisation de la fonction de correction $f_c(.)$ permet de conserver la qualité de l'estimation Λ de l'algorithme *MAP*, au détriment de la complexité de calcul par rapport à l'algorithme *Max-Log-MAP*.
En choisissant une implantation partielle de cette fonction de correction à l'aide d'une table de valeurs pré-calculées (*8 values Pre-computed One-Directional Table* [Robe95]), un compromis peut être atteint entre l'augmentation de complexité de calcul et la dégradation de la qualité d'estimation par rapport à celle de l'algorithme original [Robe95].

5.3.4 Une curiosité: le Soft Output Viterbi Algorithm ('SOVA')

Le point fort des algorithmes dérivés de l'algorithme *MAP* est la livraison de la fiabilité Λ. Toutefois, ces méthodes souffrent d'une charge de calcul très importante par rapport à l'algorithme de Viterbi, qui est reconnu comme méthode de décodage convolutif de référence.
En guise de solution, le concept de *Soft Output Viterbi Algorithm* (*SOVA*) a été ainsi proposé [Forn73] [Hage95] [Vale98][Vuce00]. Le concept se base sur la modification de l'algorithme de Viterbi de manière à produire une estimation de la fiabilité Λ de chaque bit du message, en utilisant le chemin le plus probable.

Bien qu'il existe plusieurs méthodes *SOVA*, les estimations de la fiabilité Λ suivent dans les grandes lignes le même principe. Toutes les méthodes d'estimation du *np*-ème bit se basent sur la relation existant entre les métriques cumulées

- du chemin globalement le plus probable, et
- du meilleur chemin, qui s'est écarté au niveau de profondeur *np* (Figure 5-8).

.

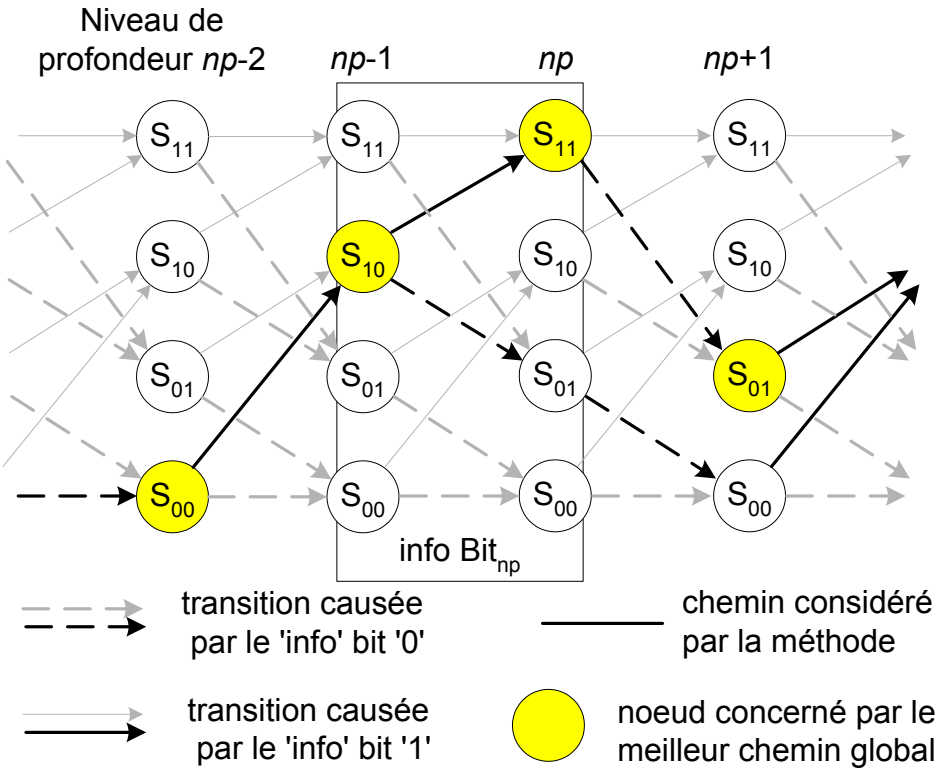

Figure 5-8: chemins concernés par la procédure d'estimation de la fiabilité du décodage du *np*-ème bit, par les méthodes *Soft Output Viterbi Algorithm*.

En généralisant, la complexité des méthodes appartenant à la classe des algorithmes *SOVA* peut être estimée entre 1 et 2 fois la complexité de calcul de l'algorithme de Viterbi. En Annexe *A* (Section *A.3*), on trouve la description de l'algorithme *Bidirectional Soft Output Viterbi Algorithm* ('*bidirectional SOVA*') [Vuce00]. L'avantage de cet algorithme est sa simplicité de compréhension et d'implantation.

5.4 Analyse critique des méthodes présentées

Du point de vue historique, le premier critère de décodage a être exploité par les méthodes de décodage convolutif est l'estimation du message le plus probable (Table 5-1a).

La première méthode (le décodage séquentiel élaboré principalement par Wozencraft, Reiffen, Fano et Jelinek [Joha99] [Proa95] [Schl97]) utilise ce critère, en modélisant les transitions d'états de la mémoire de l'encodeur à

l'aide d'un diagramme en arbre. Populaire dans les systèmes à basse vitesse de transmission [Schl97], cette méthode souffre de trois inconvénients majeurs: l'exécution séquentielle de l'algorithme, un temps variable de décodage et la fonction de métrique dépendante de la longueur du chemin analysé (voir Sous-section 5.2.1 et Section A.1).

Méthode	Critère de décodage		Qualité de:			Principaux paramètres influençant la complexité de calcul	Notes:
	message	symboles	Estimation du message	Estimation des symboles	Estimation de la fiabilité		
Algorithme séquentiel (Wozencraft 1957) § 5.2.1 § A.1	X		sous optimale	-	-	condition de transmission des signaux, procédure pour la détection des chemins incorrects	première méthode proposée
List Decoding § 5.2.2 § B.2	X		sous optimale	-	-	nombre L des états considérés pour le décodage	
Algorithme de Viterbi (Viterbi 1967) § 5.2.3	X		optimale	-	-	longueur de contrainte K du code	méthode utilisée communément pour le décodage des codes convolutifs (où $K<11$)

Table 5-1a: vue d'ensemble des caractéristiques des méthodes les plus représentatives présentées dans ce chapitre.

Ces inconvénients ont été éliminés en changeant la nature de la recherche. La recherche unidirectionnelle et synchronisée du chemin le plus probable (*Breadth-first*) rend la complexité de calcul indépendante des conditions de transmission du signal et fixe le temps de décodage. La structure algorithmique de ce type de recherche est fortement parallèle. Cette dernière caractéristique est indispensable pour une implantation *VLSI*. Dans la classe de méthodes utilisant ce type de recherche on trouve le *List Decoding* (Sous-section 5.2.2 et Section B.2) et l'algorithme de Viterbi (Sous-section 5.2.3).

Méthode	Critère de décodage		Qualité de:			Principaux paramètres influençant la complexité de calcul	Notes:
	message	symboles	Estimation du message	Estimation des symboles	Estimation de la fiabilité		
symbol-by-symbol MAP (Bahl 1974) § 5.3.1 § A.2		X	-	optimale	optimale	longueur de contrainte *K* du code	
Soft-Output Viterbi Algorithm (Hagenauer 1989) § 5.3.4 § A.3	X		optimale	-	sous optimale	longueur de contrainte *K* du code	méthode basée sur l'algorithme de Viterbi
Max-Log-MAP (Koch 1990) § 5.3.2		X	-	sous optimale	sous optimale	longueur de contrainte *K* du code	méthode basée sur l'algorithme *MAP*
Log-MAP (Robertson 1995) § 5.3.3		X	-	Dépendante de l'implantation	Dépendante de l'implantation	longueur de contrainte *K* du code	méthode basée sur l'algorithme *MAP*

Table 5-1b (cont.): vue d'ensemble des caractéristiques des méthodes les plus représentatifs présentées dans ce chapitre (Section 5.3).

Du point de vue historique, le dernier algorithme de cette classe de méthodes estimant le message le plus probable est l'algorithme de Viterbi. La nouveauté apportée par cet algorithme est la possibilité d'exploiter tout le potentiel de correction d'erreurs à disposition, sans devoir recourir à une analyse exhaustive de tous les possibles messages. Malgré cette optimisation des ressources, la complexité de calcul croît toutefois exponentiellement en fonction de la longueur de contrainte K du code. Pour cette raison, la méthode est communément utilisée pour le décodage de codes convolutifs dont la longueur de contrainte est limitée ($K \leq 10$) [Proa95].

L'autre critère de décodage (historiquement plus récent) se concentre sur le décodage séparé de chaque symbole du message (Table 5-1b).
L'algorithme pionnier est l'algorithme *symbol-by-symbol Maximum A Posteriori* (*MAP*), qui est présenté comme une solution alternative à l'algorithme de Viterbi [Vale98]. Les avantages fournis par ce critère sont la

minimisation de la probabilité d'erreur de chaque symbole et la livraison d'une estimation de la fiabilité du décodage. Cet algorithme souffre d'inconvénients importants, tels que la représentation numérique des variables, la demande de ressources de stockage et l'importante charge de calcul (Sous-section 5.3.1 et Section A.2) [Robe95] [Vale98][Vuce00].
Pour réduire ces inconvénients, différents algorithmes ont été proposés ultérieurement. Les exemples les plus caractéristiques sont les algorithmes *Max-Log-MAP Algorithm* (Sous-section 5.3.2), *Log-MAP Algorithm* (Sous-section 5.3.3) et *Soft Output Viterbi Algorithm* (Sous-section 5.3.4 et Section A.3) [Robe95]. Au niveau de l'estimation de la fiabilité Λ (Table 5-1), ces méthodes diffèrent principalement dans l'utilisation des ressources à disposition [Vuce00] (Figure 5-6, Figure 5-7 et Figure 5-8).

Par rapport à la complexité de calcul de ces méthodes, on peut tout de suite constater que:

- La performance et la complexité de calcul des méthodes de décodage séquentiel et *List Decoding* sont influencées par le choix des paramètres impliqués dans l'implantation.
- La complexité de calcul des méthodes basées sur l'algorithme de Viterbi et *MAP* dépend de la longueur de contrainte *K* du code convolutif utilisé.
- Bien qu'il en existe plusieurs versions, la méthode la plus simple d'évaluation de la fiabilité Λ appartient à la classe de méthodes *SOVA*. Puisque la structure de l'algorithme est basée normalement sur l'algorithme de Viterbi et sur une procédure parallèle pour l'estimation de la fiabilité, on peut raisonnablement s'attendre à ce que la complexité soit inférieure au double de la complexité de l'algorithme de Viterbi [Vale98] [Vuce00].
- La complexité de calcul de la méthode *Max-Log-MAP* est approximativement estimée au double de celle des méthodes *SOVA* [Robe95] [Vuce00].
- La méthode *Log-MAP* est caractérisée par une complexité de calcul estimée entre 2 et 3 fois celle des méthodes *SOVA* [Vale98] [Vuce00].

5.5 Conclusions

Dans ce chapitre, les principes et les algorithmes les plus représentatifs utilisables pour le décodage de codes convolutifs ont été présentés.

Les notions ainsi que les principes de décodage discutés représentent la base des études qui seront présentées par la suite. Le chapitre suivant traitera en effet le décodage software d'un message protégé contenant des bits de parité du *CRC*. Ce sujet sera introduit par l'évaluation des performances d'une implantation classique des opérations de décodage, en utilisant l'algorithme de Viterbi. Deux méthodes itératives seront ensuite proposées. La première méthode "*List Viterbi Algorithm*" exploite les propriétés de l'algorithme de Viterbi, alors que la seconde "*List Decoding intégrant la validation CRC*" incorpore l'algorithme *List Decoding*.
Le sujet du Chapitre 7 est l'implantation *ASIC* d'une méthode de décodage de codes convolutifs. Afin de mieux répondre aux exigences actuelles de la communication mobile, la méthode de décodage choisie pour l'étude est l'algorithme de Viterbi. Le système de décodage de base sera ainsi décrit en décomposant l'algorithme en ses éléments constituants.

Références

[Ande84] J. B. Anderson, S. Mohan, "Sequential Coding Algorithms : A Survey and Cost Analysis", *IEEE Transactions on Communications*, Vol. COM-32, No. 2, février 1984, pp. 169-176.

[Bahl74] L. R. Bahl, J. Cocke, F. Jelinek, et J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory*, Vol. 20, mars 1974, pp. 284-287.

[Forn73] G. D. Forney Jr., "The Viterbi Algorithm", *Proceedings of the IEEE*, Vol. 61, No. 3, mars 1973, pp.268-278.

[Hage95] J. Hagenauer, "Source-Controlled Channel Decoding", *IEEE Transactions on Communications*, Vol. 43, No. 3, septembre 1995, pp. 2449-2457.

[Hell71] J. A. Heller, I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication", *IEEE Transactions on Communication Technology*, Vol. COM-19, octobre 1971, pp. 835-847.

[Joha99] R. Johannesson, K. S. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Series on Digital and Mobile Communication, Wiley-IEEE Press, Etats-Unis d'Amérique, 1999, chapitres 4-6, pp. 163-315.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[Robe95] P. Robertson, E. Villebrun, et P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", *Proc. ICC'95*, Seattle, juin 1995, pp. 1009-1013.

[Schl97] C. Schlegel, *Trellis Coding*, IEEE Press, Etats-Unis d'Amérique, 1997.

[Thit93] P. Thitimajshima, *Les codes Convolutifs Récursifs Systématiques et leur application à la concaténation parallèle*, Thèse de Doctorat en Electronique, Université de Bretagne Occidentale, France, 1993.

[Vale98] M. C. Valenti, *Iterative Detection and Decoding for Wireless Communications*, A Preliminary Review of Initial Research and Proposal for Current and Future Work towards Doctor of Philosophy degree, Virginia Polytechnique Institute and State University, Blacksburg, Virginia, Etats-Unis d'Amérique, 1998.

[Vuce00] B. Vucetic, J. Yuan, *Turbo Codes, Principles and Applications*, Kluwert Academic Publishers, Etats-Unis d'Amérique, 2000.

6 Architectures "software" utilisant un processeur pour le traitement numérique du signal

---

Ce sixième chapitre traite de l'implantation software de méthodes de décodage de codes convolutifs, dans le contexte de la troisième génération (3G) des systèmes de communication numériques, cellulaires et sans fil. Les codes convolutifs considérés dans la suite sont conformes aux standards UMTS.

Ce chapitre débute avec l'analyse des potentialités offertes par la structure de protection des standards UMTS, tout en considérant une architecture comprenant un processeur spécialisé pour le traitement numérique du signal. Deux situations de codage sont ainsi envisageables: à savoir le codage convolutif direct du message, ou le codage convolutif d'un message précédemment traité par un codage en bloc CRC.

Du point de vue du décodage, le cas le plus intéressant est le second: le décodage d'un message protégé contenant en plus des bits de parité CRC. Ce sujet est introduit par l'évaluation des performances d'une implantation classique des opérations de décodage, en utilisant l'algorithme de Viterbi et la validation des bits de parité. Deux méthodes itératives sont ensuite proposées: les méthodes "List Viterbi Algorithm" (nouvelle réalisation) et "List Decoding intégrant la validation CRC" (nouvelle méthode). En s'appuyant sur l'algorithme de Viterbi et sur la méthode List Decoding, ces deux méthodes utilisent de manière itérative les informations supplémentaires fournies par le codage en bloc, ce qui permet d'améliorer le rapport entre la qualité de correction et la complexité d'exécution.

Les évaluations de la complexité de calcul et de la qualité de codage utilisent la structure de protection du service de parole AMR-NB à 12,2 kbps, proposée dans le cadre du standard UMTS.

6.1 Introduction

Le marché de la téléphonie mobile de la seconde génération est actuellement caractérisé par une importante utilisation de processeurs spécialisés pour le traitement numérique du signal ('*DSP*'). Cette prédominance est due à l'expérience et à la maturité acquise aujourd'hui par cette technologie *2G* [Gath02].

Les arguments qui au début empêchaient l'utilisation des *DSP* ont été réfutés suite à la maturation de la technologie *2G*. Les dernières années ont ainsi montré une intensification de l'utilisation des architectures exploitant les *DSP* avec le perfectionnement des standards *2G*. Bien qu'il ne soit pas possible de prédire la tendance qui sera suivie dans le cadre *3G*, les points forts qui ont valorisé l'utilisation d'une telle architecture peuvent être analysés.

6.1.1 Points forts des DSP

Lors du débat pour le choix d'une architecture optimale dans l'exploitation des technologies *2G*, l'aspect de la faible consommation a favorisé le développement de téléphones mobiles à l'aide de circuits *ASIC* (*Application Specific Integrated Circuits*). L'inversion de la tendance a commencé avec la reconnaissance et la revalorisation des qualités des *DSP* [Gath02], telles que:

- La vitesse d'évolution des générations des *DSP*. En raison de la dynamique importante du marché des *DSP*, leur développement bénéficie d'une équipe de travail normalement plus nombreuse par rapport à celle chargée du design d'un *ASIC*.
- Avec l'évolution des générations de *DSP*: extension des fonctionalités et de la vitesse d'horloge. L'évolution des *DSP* porte non seulement sur l'augmentation de vitesse d'horloge mais aussi sur l'amélioration des performances et de la fonctionnalité.
- La souplesse d'emploi. Le *DSP* offre tous les avantages et potentialités d'un dispositif programmable à large usage et à traitement multitâche (Figure 6-1).
- Réduction des coûts de développement des applications. Le *DSP* est une solution à faible risque, grâce à la rapidité des modifications et des corrections des implantations software.
- Possibilités de maintenance et de mise à jour des applications. L'utilisation d'un dispositif programmable étend les possibilités de maintenance et de mise à jour des applications.

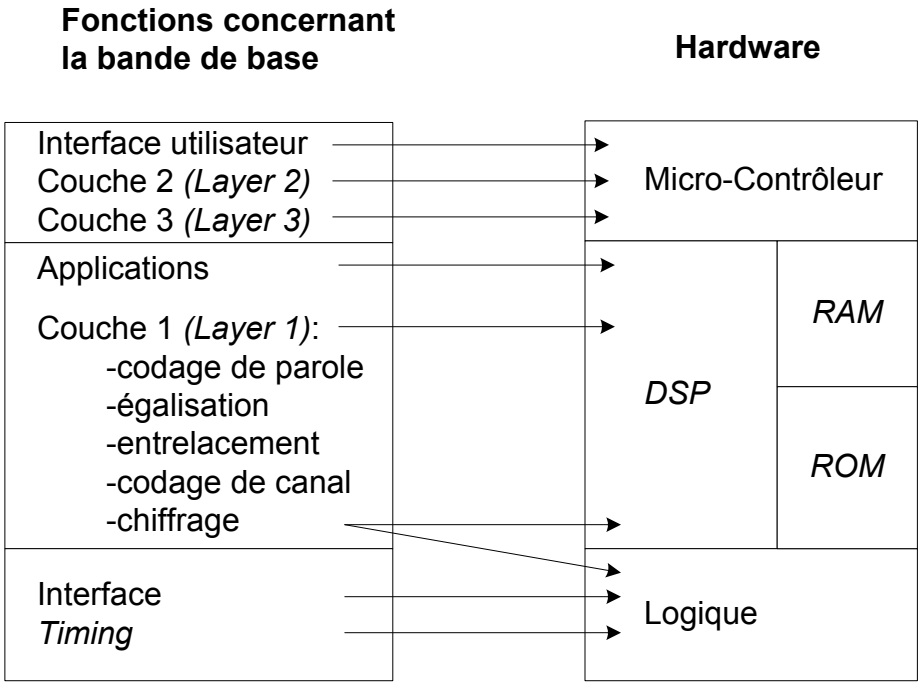

Figure 6-1: répartition classique des fonctions concernant la bande de base (*Baseband*) dans un téléphone mobile *GSM* [Gath02].

6.1.2 Exemple du standard GSM

L'évolution du standard *GSM* (*Global System for Mobile Communication*) pour la *2G* a été caractérisée par une inversion importante de tendance.

L'évolution de ce standard a été caractérisée par l'introduction de deux nouveaux codeurs de parole (le *Half Rate* et le *Enhanced Full Rate* [ETSI726]) ainsi que par des modifications de la partie responsable de la transmission sans fil des données. Par conséquent, chaque série et génération de téléphone mobile présentaient systématiquement de petites modifications et améliorations par rapport à la précédente. Lorsqu'on considère le développement d'architectures basées entièrement sur les *ASIC*, ces modifications et améliorations ont impliqué l'emploi d'importantes ressources et des coûts supplémentaires considérables. Le besoin de souplesse a rendu de plus en plus attractive l'exploitation des *DSP*.

De plus, en considérant que [Gath02]:

- le téléphone *GSM* a graduellement élargi l'assortiment des services offerts (tendance prévue dans la *3G*, Figure 6-2);

- la flexibilité est une nécessité, particulièrement lorsque la durée de vie du produit devient de plus en plus courte (l'espérance de vie d'un téléphone *GSM* s'est graduellement réduite de 2,5 à 1 année);
- la puissance de calcul non-exploitée du *DSP* est mise à disposition pour l'implantation d'autres services;

il apparaît que la différence de consommation en puissance électrique entre les *DSP* et les *ASIC* n'est plus le facteur déterminant pour le choix du type d'architecture.

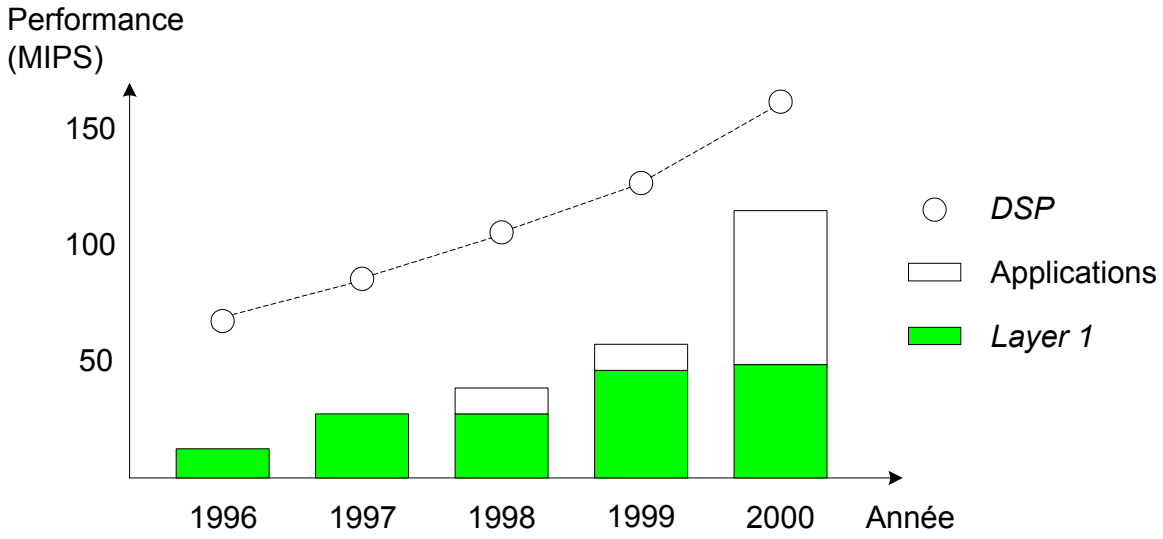

Figure 6-2: évolution de l'assortiment des services offerts par la technologie *GSM* [Gath02].

6.1.3 Situation actuelle

La souplesse d'emploi des dispositifs programmables ainsi que la tendance à la réduction de la différence de consommation entre les *DSP* et les *ASIC* rendent les *DSP* très attractifs. Ces derniers peuvent ainsi être chargés de l'exécution des fonctions de la bande de base et des applications de la nouvelle génération *3G*.
Toutefois, dans le cadre d'un produit mobile alimenté par batterie, la consommation reste toujours un facteur central. Cette réalité est perceptible dans la tendance et dans les efforts pour le développement de *DSP* à haute performance mais à consommation réduite (*Low-Power DSP*). Les fabricants de *DSP* poursuivent la réduction de la consommation d'énergie en améliorant les procédures de conception et de fabrication des *DSP*. L'élargissement de l'ensemble des instructions, le développement des périphériques efficaces et la

mise à disposition de moyens de gestion des ressources permettent la réduction ultérieure de la consommation d'énergie.

6.2 Contexte de codage UMTS

L'objectif des standards *UMTS* est la réalisation d'une nouvelle génération de technologies pour la communication mobile. Cette nouvelle génération vise à une communication indépendante des facteurs concernant la localisation des personnes, le type d'équipement utilisé, les moyens de transmission (par câble ou sans-fil) et le choix de la technologie [Ts22101].

L'une des nouveautés apportées par ces standards est la définition d'un protocole de codage de canal unique pour toutes les applications actuelles et futures. Pour réaliser cet objectif, une série de codeurs sont mis à disposition, offrant une protection optimale à un spectre plus large d'applications.

6.2.1 Protection de canal des standards UMTS

La structure de codage du standard *UMTS* [Ts25212] permet une protection adaptée aux exigences propres aux applications (Chapitre 3). Cet objectif est poursuivi en permettant tout d'abord la répartition du message en plusieurs groupe de symboles selon leur sensibilité aux erreurs. Ensuite, les standards *UMTS* permettent d'appliquer à chaque groupe de symboles:

1. un codage CRC suivi par un codage convolutif;
2. un codage convolutif;
3. un codage turbo;
4. aucun codage.

Cette thèse étant dédiée à l'étude du décodage des codes convolutifs, la suite de l'étude se concentrera sur les deux premières configurations de codage. Par rapport à ces configurations, les standards *UMTS* mettent à disposition deux codes convolutifs avec une longueur de contrainte K de 9 et de rendement R_c=1/3 et 1/2 (Table 6-1).
En comparant ces codes avec ceux exploités par la *2G*, on peut remarquer une importante augmentation de la longueur K. A titre d'exemple, le standard *GSM* propose un code convolutif avec $K=5$ pour les services de parole *Full Rate* et *Enhanced Full Rate*. Pour le service *Half Rate* il est prévu un code avec une longueur supérieure K=7 [ETSI726]. L'objectif poursuivi par l'augmentation de la longueur de contrainte des codes *UMTS* est

l'amélioration de la protection contre les erreurs de transmission, en gardant le rendement R_c inchangé.

Rendement du code	Longueur de contrainte	Polynômes [octal]	Distance libre
R_c=1/2	K=9	G_0=561, G_1=753	d_{free} =12
R_c=1/3	K=9	G_0=557, G_1=663, G_2=711	d_{free} =18

Table 6-1: caractéristiques principales des deux codeurs convolutifs proposés par le standard *UMTS*.

6.2.2 Configuration de travail du décodeur

L'opération de décodage convolutif est analysée en utilisant des simulations qui prennent en compte la structure de codage *UMTS* (Figure 6-3). Les potentialités ainsi que les diverses configurations de protection sont évaluées en utilisant la structure de codage prévue pour le service de parole *AMR-NB* à 12,2kbps (Section 3.3). Cette structure de protection présente trois contextes de codage intéressants: un codage convolutif enchaîné avec un codage *CRC*, un codage convolutif avec R_c=1/3 et un autre caractérisé par un rendement R_c=1/2 (Figure 3-3).

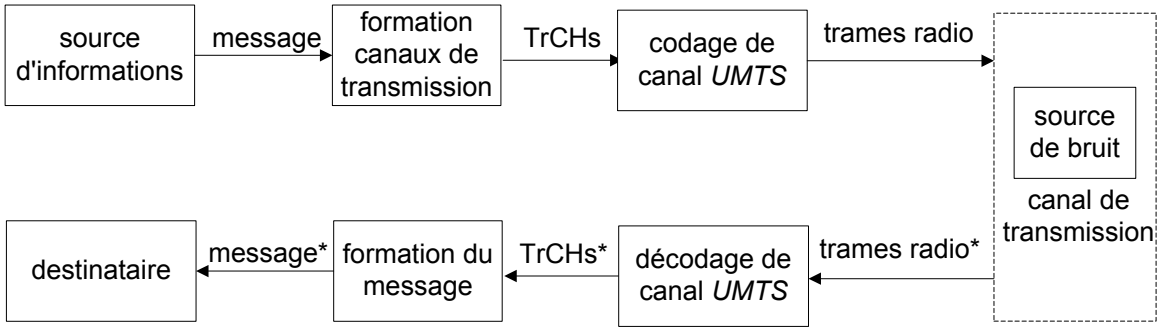

Figure 6-3: système de simulation du codage et décodage de canal *UMTS*.

Dans le but d'analyser et de reproduire facilement les résultats des études, la séquence de symboles reçue (*'trames radio*'*: Figure 6-3) est soumise à une décision ferme et le décodage livre le message décodé sans informations supplémentaires sur l'opération de décodage (*Bit True*).
Bien que les discussions et les résultats soient concentrés sur cette configuration de travail, il faut souligner que les principes présentés dans ce chapitre sont également efficacement applicables dans des systèmes utilisant

une décision pondérée. La principale conséquence du passage d'un type de décision à l'autre est la variation du niveau de la qualité de décodage.

6.3 Critères de sélection des solutions fonctionnelles

Un algorithme peut être considéré comme intéressant s'il offre un rapport optimal entre la qualité et le temps d'exécution (complexité de calcul) de la tâche établie.

Pour plusieurs applications de la *3G*, la puissance de calcul mise à disposition pour une application en temps réel est limitée. Par conséquent, à intervalles de temps fixes, chaque opération dispose d'une certaine puissance maximale de calcul. Une utilisation rationnelle de cette ressource permet ainsi soit la mise à disposition de la puissance restante pour d'autres applications, soit la réduction de la consommation d'énergie.

6.3.1 Evaluation de la qualité de protection

En fonction du contexte de codage, l'évaluation de la qualité de correction d'erreurs peut être effectuée soit à l'aide du débit des symboles erronés du message (*Bit Error Rate*, '*BER*'), soit à l'aide du débit des messages erronés *Frame Error Rate* (*FER*). Les deux mesures permettent une évaluation objective de la qualité de protection par rapport aux erreurs de transmission.

Si, pendant les simulations, les deux mesures sont réalisables en comparant les messages à protéger avec les messages fournis par l'opération de décodage (Figure 6-3), dans une application en temps réel, seul le *FER* peut être déterminé par la validation du code *CRC*[20] attaché au message.
Toutefois, il est envisageable d'utiliser la relation non-linéaire existant entre ces deux mesures (Figure 6-4) afin d'obtenir une estimation du *BER* à partir de la valeur du *FER* fournie par le codage en bloc. Le rapport existant entre ces deux mesures est principalement fonction des distances de Hamming entre les messages codés, de la méthode de décodage et des caractéristiques du bruit affectant la transmission. Le nombre important de paramètres rend ainsi difficile l'établissement d'un modèle simple et efficace de distribution des erreurs dans le message décodé (Figure 6-5).

[20] Etant donné une capacité de détection d'erreurs suffisante, soit pour empêcher les fausses validations, soit pour les réduire à un nombre insignifiant.

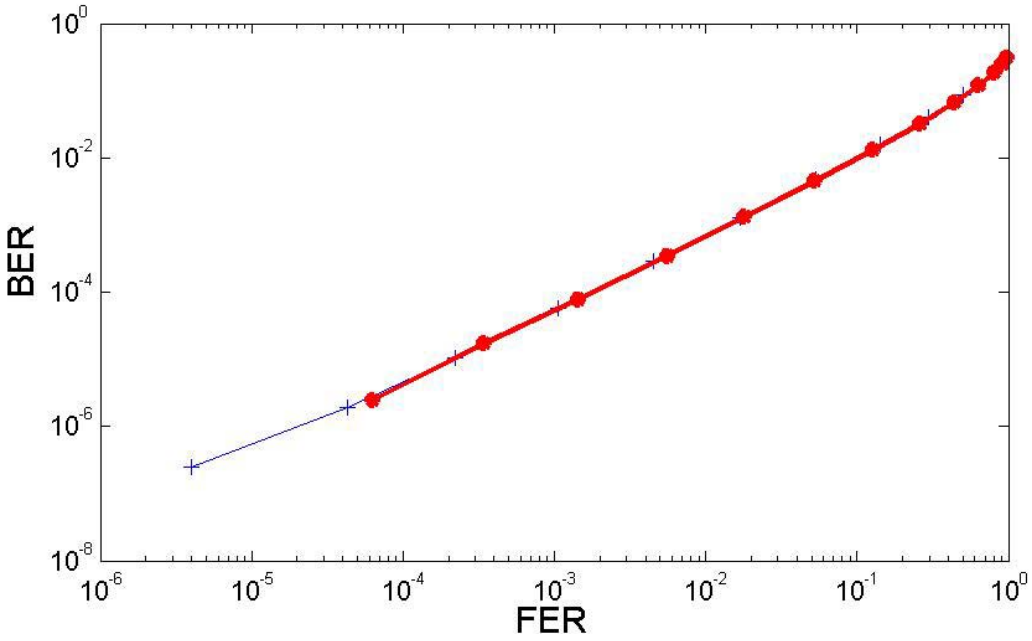

Figure 6-4: le rapport entre le *FER* et le *BER* obtenu par simulations, en appliquant une décision ferme (graphe rouge en trait gras) et une décision souple idéale (sans quantification, graphe bleu en trait fin). Simulations basées sur 500'000 trames du codeur de parole *AMR-NB* à 12.2kbps, algorithme de Viterbi.

6.3.2 Evaluation de la complexité de calcul

Si l'évaluation objective de la qualité de protection ne pose pas de gros problèmes, celle du temps nécessaire au *DSP* pour terminer l'opération de décodage est plus compliquée. De manière générale, le temps d'exécution dépend du nombre (et des types) d'opérations de la tâche, de l'efficacité du code binaire généré[21], ainsi que de la vitesse d'exécution du jeu d'opérations impliqué.

De ces trois facteurs, seul le premier caractérise la tâche à accomplir, alors que les deux autres sont des attributs du processeur choisi. Une évaluation objective de la complexité de calcul doit toutefois considérer les potentialités d'exécution spéciales, qui caractérisent les processeurs optimisés pour le traitement numérique du signal. Par conséquent, les méthodes de décodage seront jugées selon leur charge de calcul (nombre et type d'opérations), en

[21] L'efficacité est fonction du degré d'exploitation des ressources de calcul (classiques et spéciales) du *DSP*.

considérant les potentialités d'exécution qui sont les caractéristiques générales de cette grande famille de processeurs.

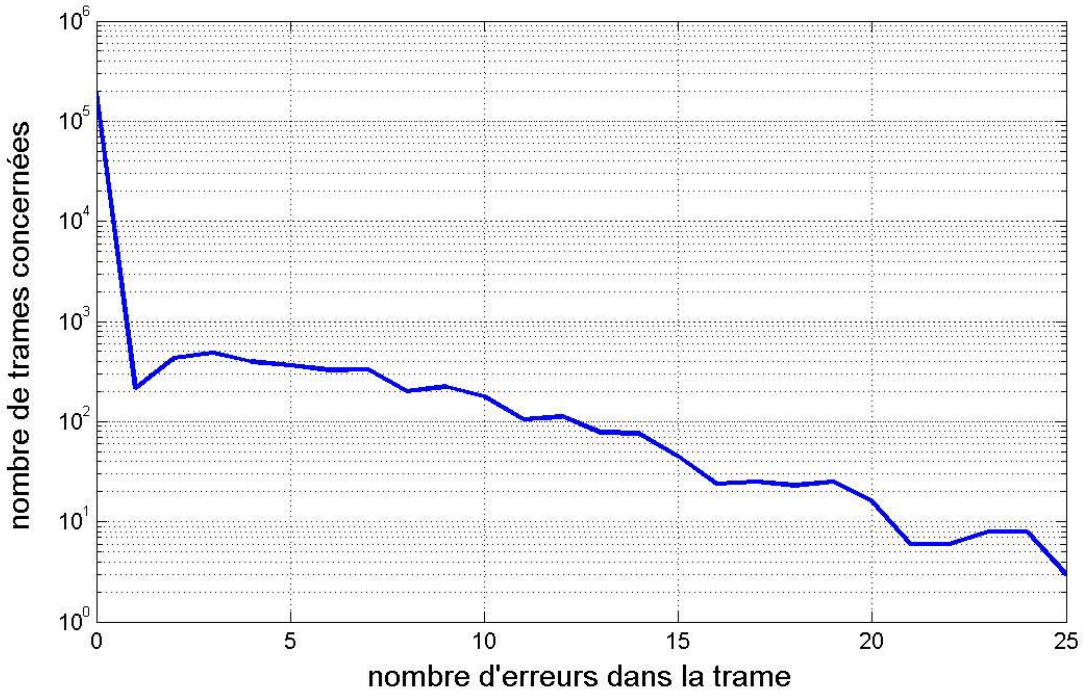

Figure 6-5: statistique des erreurs de transmission dans les trames reçues. Simulations basées sur 2 millions de trames du codeur de parole *AMR-NB* à 12.2kbps, classe *A* avec *CRC* de 12 bits, *SNR* de 4.0 dB, algorithme de Viterbi. Remarque: le nombre de trames qui ne sont pas affectées par erreurs s'élève à 1'962'952.

Influence du niveau de langage utilisé pour la description des opérations

Il existe deux principaux niveaux de langage pour la description des tâches à exécuter: le langage à haut niveau et le langage assembleur propre au *DSP* (Figure 6-6).

L'utilisation d'un langage à haut niveau réduit le temps de développement. Ce langage offre en plus une bonne portabilité de la description des tâches, description qui peut être rapidement adaptée aux autres plates-formes d'exécution. L'optimisation du code binaire est ainsi à la charge de la chaîne de compilation: les ressources et les spécialités architecturales des *DSP* sont rarement exploitées de manière exhaustive. L'utilisation des fonctions

intrinsèques (*intrinsics*)[22] permet de parer à cet inconvénient, réduisant par contre la portabilité de la description.
Le langage assembleur permet l'exploitation optimale de toutes les caractéristiques, structures et outils mis à disposition par le *DSP*. L'exécution de la tâche peut ainsi bénéficier d'une charge de calcul optimisée. Malheureusement, la portabilité d'un tel code se limite normalement à une seule famille/type de *DSP*, en plus d'un temps de développement supérieur requis par rapport à celui d'une description par langage à haut niveau.

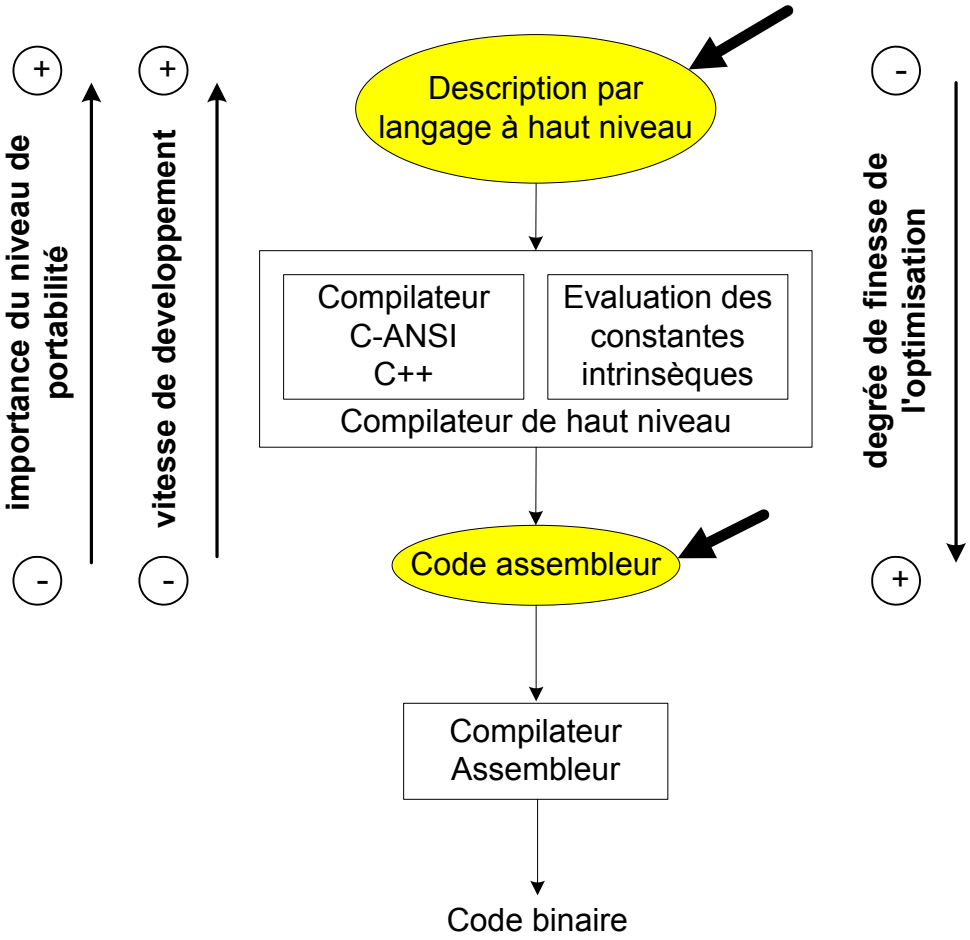

Figure 6-6: chaîne de compilation pour la génération d'un code binaire exécutable sur un *DSP*, soit à partir d'une description utilisant un langage de niveau supérieur, soit de l'assembleur.

Etant donné que cette étude ne se concentre pas sur une seule famille de *DSP*, les méthodes de décodage sont décrites en utilisant le langage *ANSI-C* en

[22] Les fonctions intrinsèques sont des fonctions décrites en code assembleur mais qui sont appelées comme les fonctions décrites en langage C/C++.

virgule fixe, afin de bénéficier de la rapidité d'exécution et de la portabilité du langage à haut niveau.

Stratégie d'évaluation adoptée

Les caractéristiques structurelles des *DSP* sont pris en compte en étendant le langage de programmation avec un ensemble de nouvelles fonctions dédiées. L'idée est de représenter (et de compter) toutes les opérations caractéristiques à la plupart des *DSP*, indépendamment de leur mode d'accès, soit par code assembleur, soit par fonctions intrinsèques.

Nom de l'opération	**Opération**	**Taille des variables [bits], signées**		**Facteur de poids** (ITU/ETSI)
		Entré	Sortie	
add	Addition 16 bits avec contrôle de dépassement et saturation	16/16	16	1
sub	Soustraction 16 bits avec contrôle de dépassement et saturation	16/16	16	1
abs_s	Valeur absolue (16 bits)	16	16	1
shl	Décalage arithmétique avec saturation (gauche)	16/16	16	1
shr	Décalage arithmétique avec saturation (droite)	16/16	16	1
mult	Multiplication 16 bits x 16 bits avec mise à l'échelle	16/16	16	1
L_mult	Multiplication 16 bits x 16 bits avec un décalage à gauche	16/16	32	1
negate	Négation 16 bits	16	16	1
extract_h	Extraction de 16 MSB	32	16	1
extract_l	Extraction de 16 LSB	32	16	1
round	Arrondi des 16 LSB avec saturation, extraction des 16 MSB	32	16	1
L_mac	*Mac*: *L_mult* avec *L_add*	32/16/16	32	1
L_msu	*Msu*: *L_mult* avec *L_add*	32/16/16	32	1
L_macNs	*Mac* sans saturation	32/16/16	32	1
L_msuNs	*Msu* sans saturation	32/16/16	32	1

Table 6-2a: les *Basic Operations* définis par les organisations *ITU* et *ETSI* [ITU729].

Le point de départ est l'extension du code *C* par les *Basic Operations* définis par les organisations *ITU* et *ETSI* (Table 6-2a et Table 6-2b), dans le contexte du développement des codeurs de parole. Son objectif est l'optimisation des

implantations, toujours en gardant la rapidité, l'efficacité et la portabilité de la description.

Nom de l'opération	**Opération**	**Taille des variables [bits], signées**		**Facteur de poids** (ITU/ETSI)
		Entré	Sortie	
L_add	Addition 32 bits avec contrôle de dépassement et saturation	32/32	32	2
L_sub	Soustraction 32 bits avec contrôle de dépassement et saturation	32/32	32	2
L_add_c	Addition 32 bits avec carry et sans saturation	32/32	32	2
L_sub_c	Soustraction 32 bits avec carry et sans saturation	32/32	32	2
L_negate	Négation 32 bits	32	32	2
mult_r	Multiplication avec arrondi	16/16	16	2
L_shl	Décalage arithmétique avec saturation (gauche)	32/16	32	2
L_shr	Décalage arithmétique avec saturation (droite)	32/16	32	2
shr_r	Décalage arithmétique avec arrondi (droite)	16/16	16	2
mac_r	*Mac* avec arrondi	32/16/16	16	2
msu_r	*Msu* avec arrondi	32/16/16	16	2
L_deposit_h	La valeur à l'entrée forme les 16 MSB de la valeur à la sortie	16	32	2
L_deposit_l	La valeur à l'entrée forme les 16 LSB de la valeur à la sortie	16	32	2
L_shr_r	Décalage arithmétique avec arrondi (à droite)	32/16	32	3
L_abs	Valeur absolue (32 bits)	32	32	3
L_sat	Saturation (32 bits)	32	32	4
norm_s	Calcul du nombre de décalage (à gauche) nécessaire à la normalisation	16	16	15
div_s	Division 16 bits	16/16	16	18
norm_l	Calcul du nombre de décalage (gauche) nécessaire à la normalisation	32	16	30

Table 6-2b (cont.): les *Basic Operations* définis par les organisations *ITU* et *ETSI* [ITU729].

Cette extension permet l'évaluation de la complexité de calcul d'une implantation sur *DSP* en surveillant le nombre d'exécutions des fonctions "classiques" et "étendues" (Table 6-2 et Table 6-3). Etant donné une gamme de fonctions couvrant de manière efficace les exigences d'une implantation

sur *DSP*, l'occurrence de ces fonctions est en effet un bon indicateur de la charge de calcul de l'application.
La complexité de calcul peut être ainsi estimée en liant à chaque type d'opération une valeur (facteur de '*poids*') indiquant la difficulté de son exécution par le *DSP* considéré. Plus ces valeurs se rapprochent des caractéristiques d'exécution du jeu d'instructions du *DSP*, plus l'évaluation de la complexité devient précise. Dans le cas des *Basic Operations*, les facteurs de poids établis par l'*ITU*/*ETSI* représentent les performances idéales d'un *DSP* générique à virgule fixe possédant une structure à 16 bits (Table 6-2 et Table 6-3).

Nom de l'opération	**Opération**	**Bits**	**Facteur de poids** (ITU/ETSI)
move16	Déplacement de données	16	1
move32	Déplacement de données	32	2
logic16	Opération logique (non-signée)	16	1
logic32	Opération logique (non-signée)	32	2
test	Contrôle		2

Table 6-3: liste établie par l'*ITU*/*ETSI* des compteurs d'occurrence des opérations classiques.

Dans cette étude, la gamme des fonctions surveillées doit être adaptée au contexte du codage de canal, contexte qui privilégie les opérateurs logiques et les opérations génériques agissant sur variables non-signées. La gamme de fonctions de l'*ITU*/*ETSI* (Table 6-2 et Table 6-3) a été ainsi élargie en ajoutant des compteurs d'occurrence supplémentaires: la Table 6-4 montre la liste complète des compteurs d'occurrence utilisés pour l'évaluation de la complexité de calcul des méthodes de décodage.
Dans la suite de ce chapitre, les estimations de la complexité de calcul indiqueront seulement l'occurrence des fonctions surveillées. Ces choix permettent une évaluation algorithmique des méthodes, indépendante des caractéristiques des familles *DSP*. A l'aide des valeurs de *poids*, cette stratégie offre toujours la possibilité d'une analyse à posteriori adaptée à chaque type de processeur.

6.3.3 Considérations sur l'implantation software

Les diverses méthodes analysées sont décrites en utilisant le langage *ANSI-C* en virgule fixe. La description des méthodes a obéi à deux des points forts des implantations software: la rapidité de développement et la description paramétrable. Cette approche ne compromet pas l'exactitude des

comparaisons entre les méthodes, ni du point de vue de la qualité de protection, ni de celui de la complexité de calcul.

Bien que la charge de calcul augmente en raison d'une description de la méthode souple et paramétrable, l'application de la même approche à toutes les descriptions permet une comparaison correcte entre les méthodes. L'importance de la perte d'efficacité causée par l'approche souple et par la qualité de la description [BDTi5] sera ensuite évaluée (Chapitre 8).

Nom de l'opération	**Opération**	**Bits**
move8	Déplacement de données	8
move16	Déplacement de données	16
move32	Déplacement de données	32
logic8	Opération logique (non-signée)	8
logic16	Opération logique (non-signée)	16
logic32	Opération logique (non-signée)	32
test	Contrôle	
shiftl8	Décalage à gauche (non-signée)	8
shiftr8	Décalage à droite (non-signée)	8
shiftl16	Décalage à gauche (non-signée)	16
shiftr16	Décalage à droite (non-signée)	16
shiftl32	Décalage à gauche (non-signée)	32
shiftr32	Décalage à droite (non-signée)	32
incr8	Incrémentation	8
incr16	Incrémentation	16
incr32	Incrémentation	32
division8	Division	8
division16	Division	16
division32	Division	32
pointerAssign	Assignation d'un pointeur	
pointerOp	Arithmétique concernant un pointeur	
testLOOP	Contrôle concernant la gestion d'une boucle	
incr8LOOP	Incrémentation concernant la gestion d'une boucle	8
incr16LOOP	Incrémentation concernant la gestion d'une boucle	16
incr32LOOP	Incrémentation concernant la gestion d'une boucle	32
move8LOOP	Déplacement de données concernant la gestion d'une boucle	8
move16LOOP	Déplacement de données concernant la gestion d'une boucle	16
move32LOOP	Déplacement de données concernant la gestion d'une boucle	32

Table 6-4: liste complète des compteurs d'occurrence des opérations qui sont utilisés pour l'évaluation de la complexité de calcul des méthodes de décodage convolutif. Le fond gris met en évidence les compteurs définis par les organisations *ITU* et *ETSI*.

Bien qu'une description par langage à haut niveau permette de se distancer des caractéristiques fines des *DSP*, le choix de la taille des variables (16 bits ou 32 bits) a une influence sur le temps de traitement. La vitesse d'exécution peut varier en fonction de l'architecture interne du *DSP* (voir Annexe *B*, Section *B.3*). Par rapport aux implantations de méthodes de décodage convolutif, la représentation numérique (*Range*) des variables à 16 bits est suffisante pour tout l'ensemble des opérations. Le choix des tailles peut s'effectuer en considérant l'architecture interne du *DSP*.

6.4 Approche classique: l'algorithme de Viterbi

Conformément à la rapidité d'évolution du marché de la communication mobile, la méthode de décodage la plus répandue dans la *2G* est tout d'abord implantée: l'algorithme de Viterbi. L'emploi de cet algorithme est prédominant pour le décodage des codes convolutifs, dont la longueur de contrainte est réduite ($k \leq 10$) [Proa95].

6.4.1 Fonction de métrique

L'implantation de l'algorithme de Viterbi implique la définition de la fonction de métrique ainsi que de la stratégie de décodage du message (*Traceback*).

La tâche de cette fonction de métrique est la mesure de l'affinité entre le message sous analyse et les symboles reçus. Son implantation doit considérer les trois aspects suivants:

- la représentation numérique des symboles reçus;
- le risque de dépassement de la valeur maximale représentable par les variables utilisées (*Overflow*);
- l'initialisation des valeurs de métriques cumulées, selon les conditions de départ de l'opération de codage.

La représentation numérique des symboles reçus influence l'implantation de la fonction de métrique et détermine la qualité de protection de la méthode de décodage. Une décision souple livre au décodeur une quantité supérieure d'informations, ce qui permet une amélioration remarquable de la qualité de protection (Figure 6-7 et Figure 6-8). Le gain de codage (*Coding Gain*) offerte par l'application d'une décision souple idéale[23] peut atteindre 2dB

[23] C'est-à-dire une décision souple bénéficiant d'un nombre infini de niveaux de quantification.

[Proa95]. Comme déjà indiqué, cette étude considère la situation de décodage dans le contexte d'une décision ferme.

Représentation numérique des métriques

Cette fonction se base sur le concept de métrique cumulée et de métrique de branche (5.6). Dans le contexte d'une décision ferme, les représentations numériques de ces métriques utilisent la même unité de base qui est la distance de Hamming.

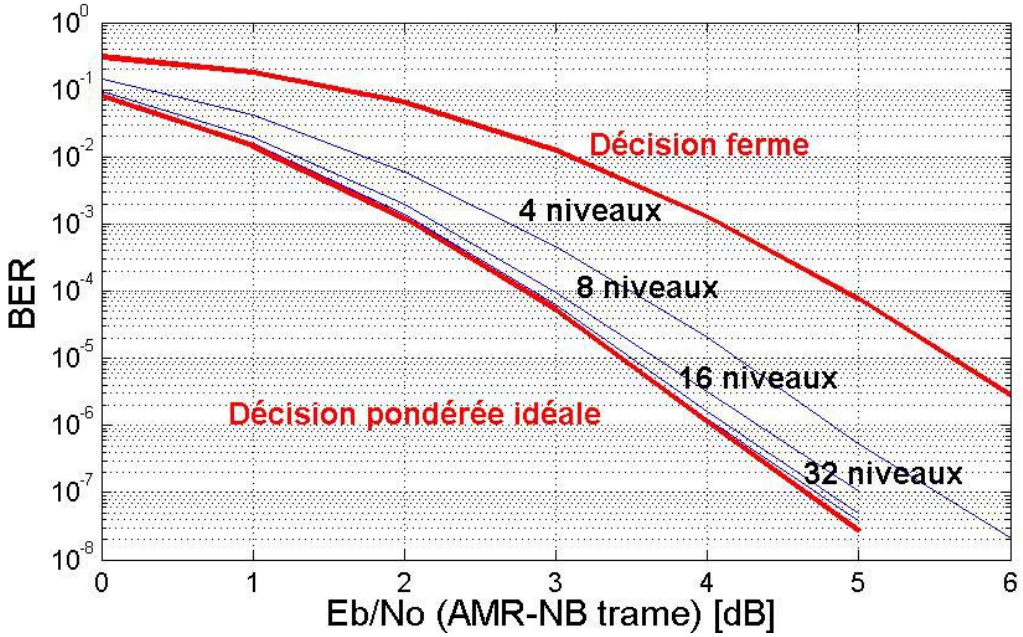

Figure 6-7: *BER* obtenus en fonction d'une décision ferme, pondérée avec 4, 8, 16 et 32 niveaux, et d'une décision pondérée idéale (sans quantification: représentation en virgule flottante). Les symboles sont modulés par amplitude de manière antipodale (+1/-1 PAM): les niveaux de quantification sont répartis de manière uniforme entre l'échelle $[-1.5\cdot\sigma^2-1\ldots1.5\cdot\sigma^2+1]$ en fonction de la valeur de la variance σ^2 du bruit gaussien (Figure 4-9). Simulations: service *AMR-NB* à 12.2 kbps, 2 millions de trames, classe *A* avec 12 bits de parité *CRC*.

La représentation numérique en virgule fixe (*Fixed Point Arithmetic*) peut causer des effets de dépassement des capacités des variables (*Overflow*). L'exécution de l'algorithme de Viterbi exige que la représentation des métriques cumulées reproduise correctement les décalages entre toutes les métriques cumulées (appartenant au même niveau de profondeur): le dépassement des capacités représentables ne garantit évidemment plus le déroulement correct de l'algorithme. Par conséquent, l'implantation de la

fonction de métrique doit empêcher le dépassement soit par l'utilisation de variables suffisamment grandes, soit par l'introduction de procédures de normalisation des valeurs de métriques cumulées. Le standard de codage *UMTS* [Ts25212] limite le nombre de bits contenus dans le message à coder à 505 bits: l'utilisation de variables à 16 bits assure ainsi la représentation correcte de toutes les métriques cumulées, sans effets de dépassement des capacités des variables[24].

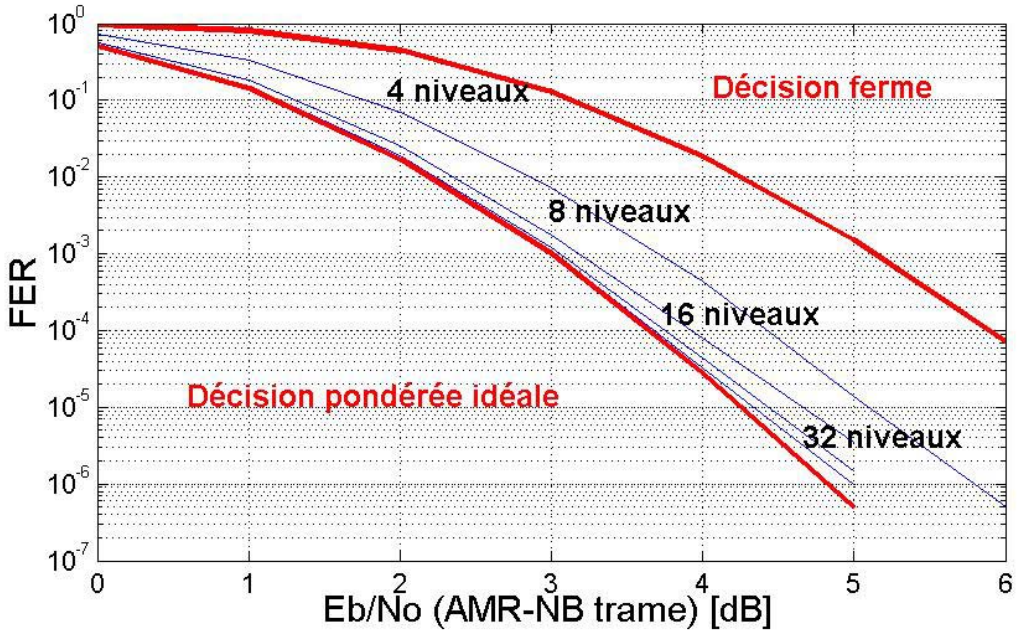

Figure 6-8 : *FER* obtenus selon le type de décision appliqué. Les paramètres de simulation sont les mêmes que ceux de la simulation précédente (Figure 6-7).

Initialisations des valeurs de métrique

Un dernier aspect de l'implantation de la fonction de métrique est l'initialisation des variables représentant les métriques cumulées, selon les conditions de départ de l'opération de codage convolutif.

Le standard *UMTS* [Ts25212] établit une remise à zéro de la mémoire du codeur convolutif avant la procédure de codage. Pratiquement, cette opération détermine l'état de départ de la mémoire du codeur.
La procédure de décodage peut ainsi utiliser cette information, améliorant ainsi la qualité de protection des premiers bits du message (Figure 6-9).

[24] L'unité de la représentation des métriques cumulées est la distance de Hamming (décision ferme).

Il existe deux principes d'initialisation de la procédure de décodage:

1. L'exécution de l'algorithme de Viterbi à partir de l'état de départ défini par le contexte de codage. Une procédure de contrôle des chemins est ainsi nécessaire dans les phases initiales de l'algorithme. La valeur de métrique cumulée de l'état de départ est initialisée à zéro.
2. Une initialisation diversifiée des valeurs de métrique dans le but de rendre défavorables les chemins démarrant d'autres états. La stratégie envisagée est de provoquer l'élimination 'naturelle' de ces chemins incorrects, pendant les phases initiales de l'algorithme de Viterbi.

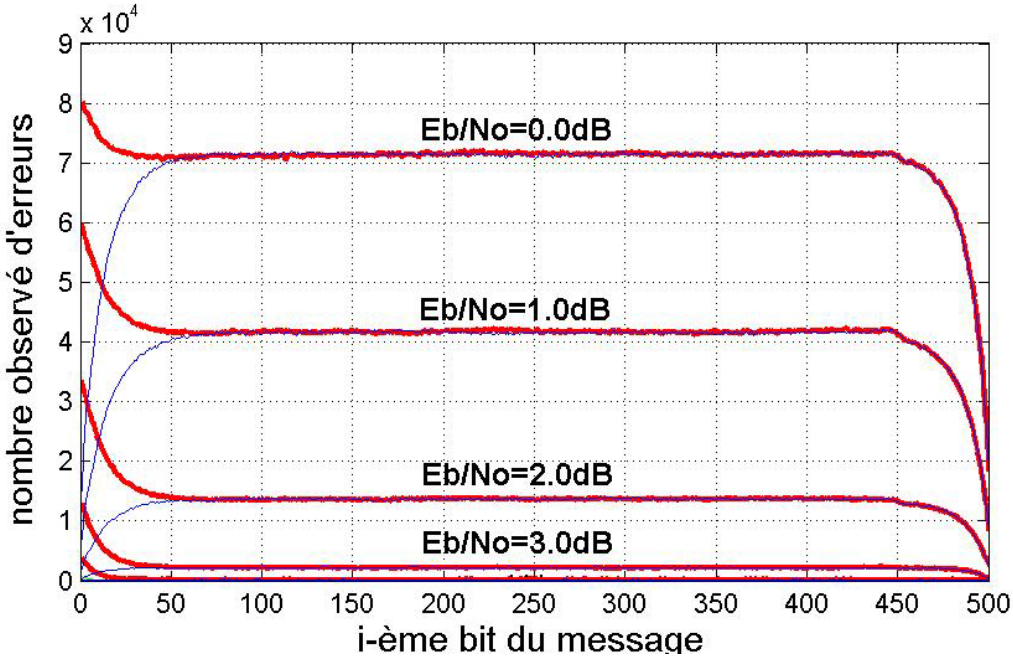

Figure 6-9: exemple de distribution des erreurs de décodage dans un message. Le graphe rouge en trait gras indique la protection offerte par une initialisation par remise à zéro de toutes les métriques cumulées. La qualité procurée par une initialisation exploitant une procédure de contrôle est représentée en graphe bleu en trait fin. L'augmentation de la qualité de protection des derniers bits du message est due à l'utilisation de bits de terminaison (*Tail Bits*). Contexte: codage convolutif avec remise à zéro de la mémoire du codeur convolutif, 200'000 messages, messages de 500 bits, 8 bits de terminaison, code convolutif avec R_c=1/3 du standard *UMTS*, E_b/N_0 de 0 à 4 dB.

Le premier principe initialise à zéro la métrique cumulée de l'état de départ. Ensuite, l'algorithme démarre en ne considérant que les chemins démarrant depuis cet état. L'exploitation optimale des informations des conditions de

départ est toutefois contre-balancée par l'introduction d'une procédure de contrôle du déroulement de l'algorithme. Bien que dans la phase initiale cette procédure permette une gestion optimale des ressources, ces opérations de contrôle supplémentaires causent l'augmentation de la complexité totale du décodage. L'importance de l'augmentation dépend de la souplesse de la description et du style de programmation. Les implantations software de l'algorithme de Viterbi, qui apparaissent dans la suite de l'étude, adoptent ce concept.

Le second principe prévoit l'initialisation différenciée des métriques cumulées, de manière à garantir le déroulement correct de l'algorithme, sans l'introduction de procédures supplémentaires de contrôle. Ce principe se base sur une initialisation défavorisant les chemins qui ne démarrent pas de l'état de départ établi (Figure 6-10). En raison de l'initialisation de la mémoire du codeur avant le codage, ces chemins sont en effet inexistants.

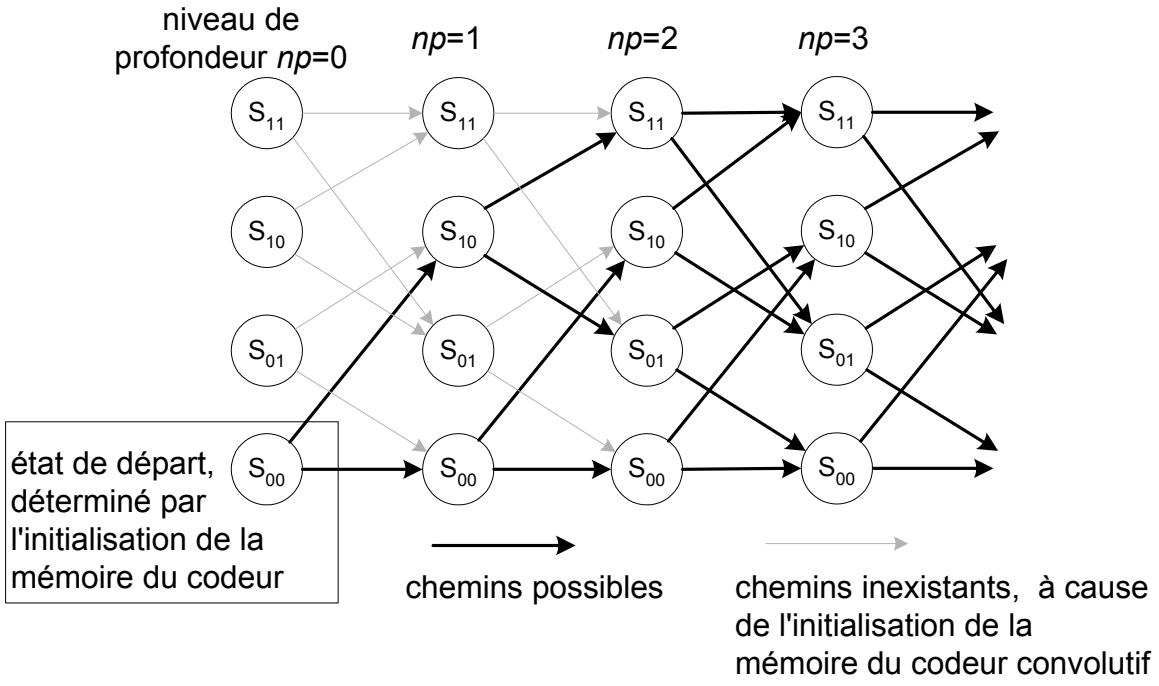

Figure 6-10: états des chemins dans la phase initiale, selon l'initialisation de la mémoire de l'encodeur convolutif (longueur de contrainte de 3).

Selon ce principe, l'initialisation différenciée doit prendre en compte:

- la contribution maximale apportée par la métrique de branche ('*MaxContribBranche*');
- la valeur maximale représentable par les variables des métriques cumulées ('*ValMaxMetrique*');

- la taille de la mémoire du codeur ($K-1$): cette valeur identifie le niveau de profondeur, à partir duquel tous les états de mémoire sont atteints par des chemins possibles (Figure 6-10).

Ainsi, une valeur comprise entre

$$(K-1)\cdot MaxContribBranche$$

et

$$ValMaxMetrique-\left[(K-1)\cdot MaxContribBranche\right]$$

peut être utilisée pour l'initialisation différenciée des métriques cumulées. Cette initialisation permet le déroulement correct de l'algorithme, assurant la qualité de protection offerte par l'initialisation de la mémoire du codeur (Figure 6-9).

6.4.2 Mécanisme de décodage du meilleur chemin

Selon l'algorithme proposé par Viterbi, la reconstruction du message commence après l'identification du chemin (complet) le plus probable.
Cette stratégie devient peu fonctionnelle si on considère:

- son incompatibilité dans des contextes de codage continu;
- l'important retard algorithmique de la procédure de décodage, en raison de l'attente de la réception de tous les symboles avant son exécution;
- les nombreuses ressources impliquées pour le stockage des informations qui sont nécessaires à la reconstruction des chemins.

Une solution à ces inconvénients est une prise de décision anticipée de la valeur du np-ème bit du message, en utilisant le meilleur chemin partiel à la profondeur $np+\delta$ (Figure 6-11). Cette approximation de l'algorithme[25] anticipe ainsi le début de la reconstruction du message, en exécutant de manière itérative la prise de décision sur un/plusieurs bits d'informations pas encore décodés.

[25] La décision de la valeur du np-ème bit est prise en n'analysant que la séquence incomplète des observations $R_0^{np+\delta}$.

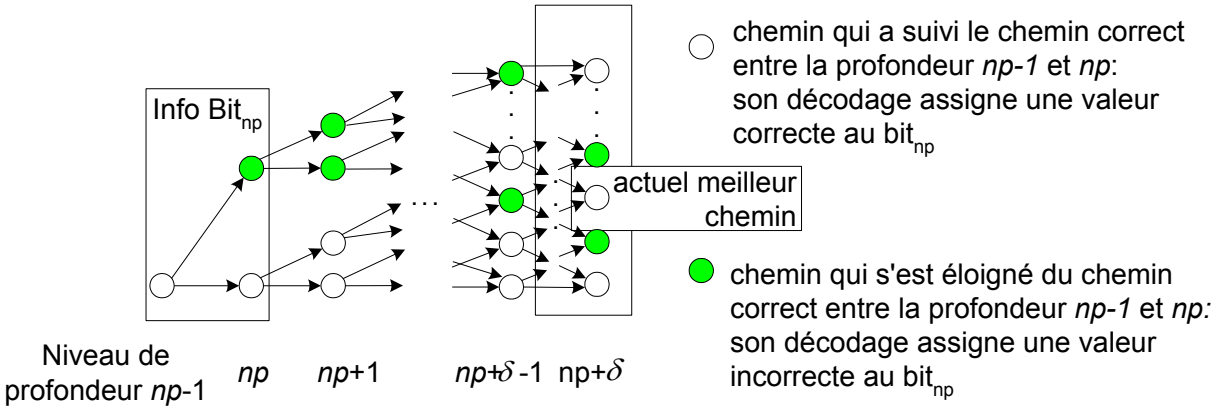

Figure 6-11: principe à la base du décodage avec troncation des chemins.

Cette approche réalise en pratique une *troncation* de la mémoire du décodeur, ce qui permet d'établir a priori les exigences de stockage d'information, indépendamment du nombre effectif de bits constituants le message.

On est ainsi confronté à une répétition systématique de la procédure de décodage, répétition qui augmente la charge de calcul de l'algorithme (Table 6-6). Puisqu'il s'agit d'une approximation de l'algorithme, on doit en plus s'attendre à une dégradation de la qualité de protection, dont l'importance est fonction du paramètre δ. L'assignation d'une valeur à la distance δ est ainsi un compromis entre la dégradation de la qualité de décodage et l'utilisation minimale des ressources de stockage.

Distance minimale δ_{min}

La relation entre la distance δ et la dégradation de la qualité de protection dépend des caractéristiques du codeur convolutif ainsi que du nombre d'erreurs de transmission.

Pour décrire l'évolution de la métrique cumulée d'un chemin incorrect, le concept de *croissance minimale de la métrique* (*Minimum Distance Growth*) a été introduit par Anderson [Ande89]. Ce terme (qui sera représenté par l'abréviation *MinMetrCroissance* dans les formules) indique l'augmentation minimale de la métrique cumulée d'un chemin incorrect, normalisée par le nombre de symboles analysés.
Grâce à ce concept, la contribution minimale de métrique de branche à la métrique cumulée d'un chemin incorrect est de

$$n \cdot MinMetrCroissance, \tag{6.1}$$

(étant donné l'absence d'opérations d'adaptation du débit de transmission[26]). Ainsi, afin de garantir la correction (minimale) de

$$e_{\min} = \left\lfloor \frac{d_{free} - 1}{2} \right\rfloor \tag{6.2}$$

erreurs, la distance minimale δ_{min} ne doit pas être inférieure à[27]

$$\left\lceil \frac{e_{\min}}{n \cdot MinMetrCroissance} \right\rceil. \tag{6.3}$$

Cette distance assure qu'après δ_{min} branches, tous les chemins qui provoquent une prise de décision incorrecte montrent des métriques cumulées défavorables[28] (Figure 6-11). Le chemin le plus probable au niveau de profondeur $np+\delta$ coïncidera ainsi avec la première partie, soit du chemin correct, soit d'un chemin qui ne s'est éloigné qu'après le niveau de profondeur np (Figure 6-11).

De façon générale, on peut déterminer la distance δ minimale en fonction du nombre d'erreurs, qui perturbent les dernières δ observations ($\delta \cdot n$ symboles), de sorte que la dégradation de la qualité de protection soit rendue négligeable:

$$\delta > \frac{\text{nombre d'erreurs}_{np \to np+\delta}}{n \cdot MinMetrCroissance}. \tag{6.4}$$

Similairement à la distance libre d_{free} (Table 6-1), la valeur de la croissance minimale de la métrique peut fournir des indications sur la qualité de protection du code convolutif exploité (Table 6-5). Cette valeur fixe en effet la limite supérieure du nombre d'erreurs de transmission qui peuvent être (potentiellement) supportées par le codage convolutif.

---

[26] Le standard UMTS [Ts25212] prévoit la possibilité d'une adaptation du débit de transmission (voir chapitres précédents). Dans le cas d'une utilisation, le paramètre n doit être adapté au débit de poinçonnage.

[27] Par rapport à une représentation des métriques basée sur la seule distance de Hamming.

[28] En considérant un nombre d'erreurs inférieur à e_{min}.

Rendement du code	Croissance minimale (normalisée) de la métrique d'un chemin incorrect [distance de Hamming/symboles]	Croissance moyenne (normalisée) de la métrique d'un chemin incorrect [distance de Hamming/symboles]
***Random Tree Codes* [Ande89]**		
R_c=1/2	0.110	-
R_c=1/3	0.174	-
Simulations des codes *UMTS*		
R_c=1/2 [29]	0.154	0.341
R_c=1/3 [30]	0.207	0.408

Table 6-5: valeurs de référence concernant la croissance minimale et moyenne de la métrique d'un chemin incorrect. Par rapport aux codes *UMTS*, l'approche pratique utilisée se base sur l'algorithme pour la détermination de la distance libre d_{free}, publié dans [Cede89].

Détermination de la distance δ par simulations

La détermination de la distance δ par l'inéquation (6.4) n'est pas aisée en raison des paramètres impliqués. Une règle a ainsi été déterminée empiriquement en se basant sur l'expérience et les résultats de nombreuses simulations effectuées (exemple: Figure 6-12). Cette règle fixe la gamme de valeurs de distance δ entre *4* et *6* fois la longueur de contrainte K du code [Joha99] [Proa95] [Thit93].

Sauvegarde des informations pour la reconstruction des chemins

Le dernier aspect du mécanisme de décodage est la définition du système de sauvegarde des informations qui sont indispensables pour la reconstruction des chemins.
L'algorithme de Viterbi se base sur la reconstruction itérative des états précédents de la mémoire du codeur (*décodage du chemin* le plus probable), jusqu'à pouvoir extraire la valeur d'un (ou de plusieurs) bit du message.

[29] Par rapport au calcul de la croissance moyenne, la simulation a considéré tous les chemins incorrects qui possèdent une distance de Hamming inférieure/égale à w_i=32 (361'849'305 chemins analysés). Un contrôle systématique de tous les chemins partiels rencontrés a permis la détermination empirique de la valeur minimale de croissance.

[30] Le calcul de la croissance moyenne a analysé 338'890'207 chemins possédant une distance de Hamming inférieure/égale à w_i=53. Comme pour le cas précédent, la détermination de la valeur minimale a bénéficié d'un contrôle systématique de tous les chemins partiels rencontrés lors du calcul de la valeur moyenne.

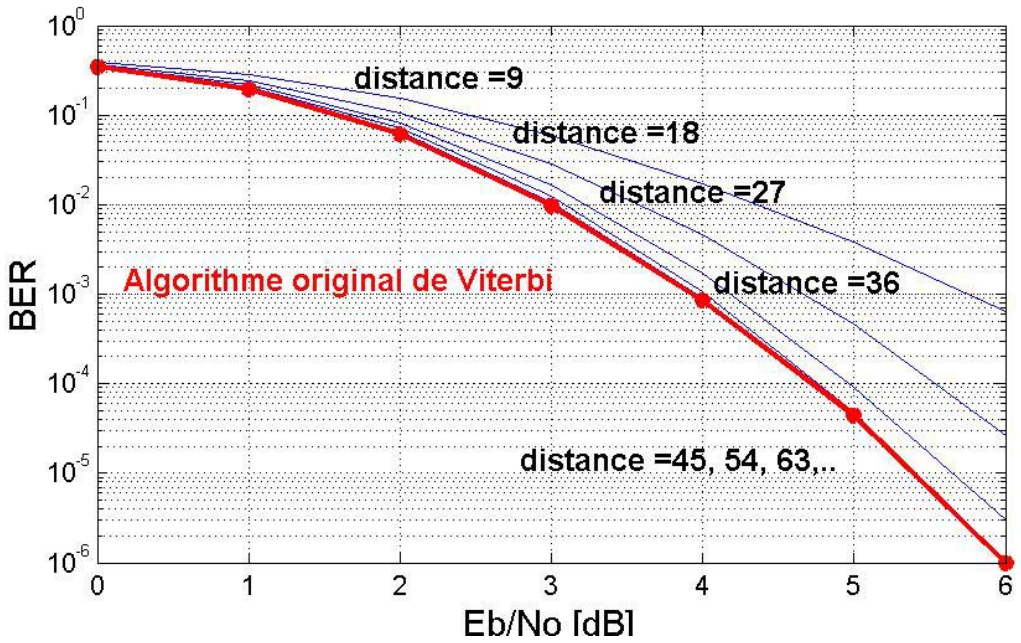

Figure 6-12: résultats des simulations ayant pour objectif la détermination de la valeur optimale de la distance δ. Paramètres des simulations: message de 1'000 bits, 8 bits de terminaison, 1 million de messages, code convolutif avec R_c=1/3 du standard *UMTS*, distances δ multiples de la longueur de contrainte K du code.

Le système le plus économe en mémoire sauvegarde seulement le bit, qui est perdu lors de la transition de la mémoire du codeur convolutif (Figure 6-13). Par contre, le système le plus rapide et donc le moins complexe sauvegarde directement la valeur de l'état précédent du chemin survivant. Afin de réduire la complexité de calcul, cette dernière stratégie a été adoptée pour l'implantation software de l'algorithme de Viterbi.

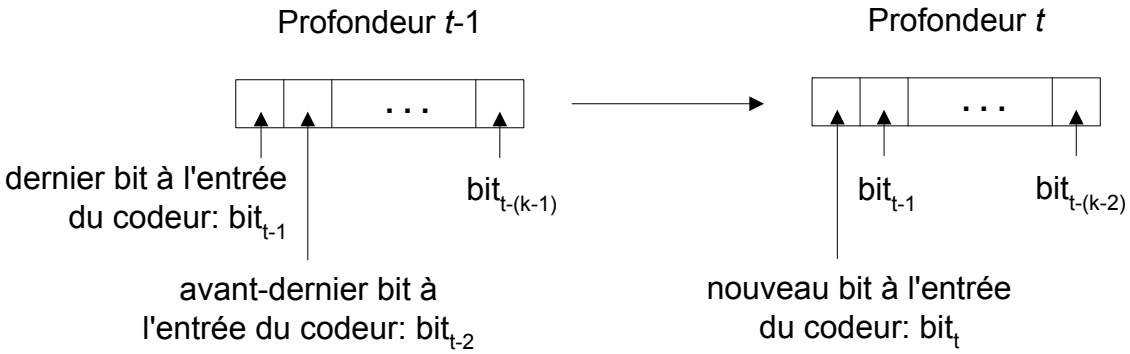

Figure 6-13: comportement de la mémoire du code convolutif lors d'une transition.

Bits de terminaison

Le standard *UMTS* [Ts25212] prévoit l'utilisation de bits de terminaison (Sous-section 2.3.2) afin de garantir une protection adéquate aux derniers éléments du message à transmettre.

Similairement aux stratégies d'initialisation de la procédure de décodage, la phase finale de l'opération de décodage peut bénéficier de ces informations:

1. Soit en ne traitant pas les chemins qui sont exclus par le codage de ces bits de terminaison. Dans les phases finales, les valeurs (figées) de ces bits limitent les états possibles de la mémoire du codeur.
2. Soit en traitant tous les chemins, mais en modifiant la procédure de sélection du chemin (partial/final) le plus probable. Dans ce cas, une procédure de contrôle supplémentaire est introduite dans l'opération de sélection.

Les implantations de l'algorithme de Viterbi de cette étude adoptent la première stratégie, c'est-à-dire le contrôle du déroulement de l'algorithme lors de la réception des symboles générés par le codage des bits de terminaison.

6.4.3 Performances de l'algorithme de Viterbi

La qualité de protection de l'algorithme de Viterbi a déjà été présentée dans les Figures 6-7, 6-8, 6-9 et 6-12. Dans la suite de l'étude, ces valeurs seront utilisées comme référence pour le jugement de la qualité de protection des autres méthodes de décodage.

Les Tables 6-6 et 6-7 montrent les évaluations de la charge de calcul de l'algorithme. Les effets de l'augmentation de la longueur de contrainte du code à K=9 sont immédiatement perceptibles: la charge de calcul du décodage est très importante. Le traitement d'un bit d'information demande en effet une charge de calcul supérieure à 7k opérations (256 opérations de sélection du chemin survivant) et nécessite presque 800k opérations pour le décodage de 93 bits d'information et de 8 bits de terminaison.

La Table 6-6 montre l'augmentation de la charge de calcul due à la stratégie de troncation de la mémoire, augmentation qui est quantifiable autour du 8 pour cent. Ensuite, la Table 6-7 prouve l'indépendance de la complexité de calcul des caractéristiques des erreurs de transmission.

	Nombre moyen d'opérations observées					
	Algorithme sans Troncation			***Algorithme avec troncation à 6 · longueur de contrainte***		
	Format des variables			Format des variables		
	32 bits	*16 bits*	***Total***	*32 bits*	*16 bits*	***Total***
Additions	44541	9	**44550**	44451	9	**44550**
Incrémentations	94	1212	**1306**	93	1212	**1305**
Soustractions	111	104	**215**	11	152	**263**
Multiplications	0	0	**0**	0	0	**0**
Divisions	0	0	**0**	0	0	**0**
Transferts de données	115659	45653	**161312**	118520	45653	**164173**
Logique	130219	2424	**132643**	130271	2424	**132695**
Décalages	70942	0	**70942**	70894	0	**70894**
Contrôles		102588	**102588**		126074	**126074**
Pointeur: assignations		611	**611**		659	**659**
Pointeur: arithmétique		513	**513**		3056	**3056**
Boucle: initialisations	24167	101	**24268**	24215	101	**24316**
Boucle: contrôles			**97723**		110810	**110810**
Boucle: incrémentations	72646	808	**73454**	85637	808	**86445**
Grand total			**710125**			**765240**

Table 6-6: évaluation détaillée de la complexité de calcul de l'implantation software de l'algorithme de Viterbi. Paramètres: structure de codage pour la classe *A* du codeur *AMR-NB* à 12.2kbps (codeur avec K=9 et R_c=1/3), moyennes sur 500 trames, valeurs avec écarts maximaux inférieurs à 1%. Le fond gris met en évidence les compteurs d'occurrence de l'*ITU*/*ETSI*.

L'implantation et les simulations software de l'algorithme de Viterbi ont utilisé la configuration suivante:

- L'étude utilise le système de canal *UMTS uplink*.
- La structure de protection prise en considération est relative au codage des bits appartenant à la classe *A* [Ts26101] (aussi nommé premier canal de transport [TrR104]), dans le cadre du service de parole *AMR* à 12.2 kbps [TrR104]. Les opérations de validation des bits de parité *CRC* sont illustrées dans l'Annexe *B* (Section *B.1*).
- Les signaux sont modulés de manière antipodale (+1/-1) et sont perturbés au niveau des trames radio (*Radio Frames*) par l'addition d'un bruit blanc gaussien (de 0.0 dB à 6.0 dB). En conséquence, l'indication E_b/N_0 s'adresse au codage des trois classes (canaux de transport).

	Nombre moyen d'opérations observées	
	Algorithme sans Troncation	
	Eb/No=0.0dB	***Eb/No=6.0dB***
Additions	44550	44550
Incrémentations	1306	1306
Soustractions	215	215
Multiplications	0	0
Divisions	0	0
Transferts de Données	161312	161312
Logique	132643	133292
Décalages	70942	70942
Contrôles	102588	102588
Pointeur: assignations	611	611
Pointeur: arithmétique	513	513
Boucle: initialisations	24268	24268
Boucle: contrôles	97723	97723
Boucle: incrémentations	73454	73454
Grand total	**710125**	**710774**

Table 6-7: évaluation de la complexité de calcul de l'algorithme de Viterbi dans deux conditions d'erreurs de transmission opposées. Paramètres: structure de codage pour la classe *A* du codeur *AMR-NB* à 12.2kbps (codeur avec K=9 et R_c=1/3), moyennes sur 500 trames, valeurs avec écarts maximaux inférieurs à 1%. Le fond gris met en évidence les compteurs de l'*ITU*/*ETSI*.

- Les signaux reçus sont représentés numériquement par une décision ferme.
- Bien que l'étude n'utilise pas l'outil de poinçonnage (*Puncturing*)[31], l'implantation de l'algorithme supporte de manière optimale ce service. Les symboles reçus sont ainsi représentés à l'aide de trois valeurs (+1/-1/0).
- La métrique cumulée de l'état de départ est initialisée à zéro et la structure de l'algorithme ne prend en considération que les chemins démarrant depuis cet état.
- Par rapport au traitement des bits de terminaison, la structure de l'algorithme considère seulement les états possibles.
- Les informations relatives à la reconstruction des chemins sont sauvegardées sous la forme des valeurs des états précédents.
- L'implantation software de l'algorithme de Viterbi s'oriente vers des plates-formes à 32 bits.

[31] Elimination d'un nombre de bits des messages codés afin de s'adapter au débit de transmission disponible (adaptation du débit de transmission, Sous-section 3.2).

6.4.4 Considérations sur cette approche classique

Conformément aux exigences du marché de la communication mobile, la méthode de décodage implantée est l'algorithme qui est prédominant dans les technologies *2G*: l'algorithme de Viterbi.

La nouvelle implantation de l'algorithme a révélé immédiatement un alourdissement de la charge de calcul assigné au *DSP*. Comme il était prévisible, le changement du contexte de codage de la *3G* demande des charges de calcul de décodage plus importantes par rapport à l'ancienne technologie *2G*. L'agrandissement de la longueur de contrainte des codes *UMTS* exige tout d'abord une complexité de calcul de $30 \cdot 2^{K-1}$ *wOP* par bit d'information décodé[32]. Ensuite, l'utilisation de débits de transmission supérieurs augmente la charge de calcul nécessaire à l'exécution en temps réel de l'opération de décodage.

La complexité de calcul de l'implantation software de l'algorithme de Viterbi dépend premièrement de la longueur de contrainte des codes convolutifs utilisés ainsi que du débit de transmission requis par l'application concernée. Par conséquent, l'augmentation de la longueur de contrainte et l'élargissement de la gamme des débits de transmission des standards *UMTS* provoquent une importante variabilité au niveau des ressources de calcul nécessaires à l'opération de décodage. La charge de calcul peut varier entre une trentaine de *wMOPS* (demandé par un service de parole à 12.2 kbps) jusqu'à des valeurs approchant le millier de *wMOPS* (exigé par les applications les plus gourmandes soutenues par les technologies *UMTS*). Par conséquent, on envisage l'analyse de la structure de codage de l'*UMTS* afin de développer des méthodes de décodage améliorant le rapport entre la qualité de protection et la charge de calcul demandée. L'objectif envisagé est l'élargissement du spectre de solutions software efficaces, afin de mieux répondre aux exigences particulières des applications.

6.5 Décodage des codes convolutifs incluant la validation CRC

Le système de codage de canal du standard *UMTS* [Ts25212] dispose de plusieurs moyens de codage, afin de permettre une protection efficace et

[32] Valeur observée. La perte d'efficacité due à une description souple et paramétrable de l'algorithme sera traitée dans le Chapitre 8.

optimale contre les erreurs de transmission pour une vaste gamme d'applications. Il existe ainsi deux configurations principales de codage convolutif: le codage direct du message et le codage d'un message traité précédemment par un codage en bloc *CRC* (Sous-section 6.2.1).

Le cas le plus intéressant est l'enchaînement d'un code convolutif avec un code en bloc (Figure 6-14), structure qui revient régulièrement dans les récents standards de communication numérique mobile (*GSM* [ETSI909], *DCS1800* [Kühn97], *UMTS* [Ts25212]). A la différence du simple codage convolutif, cette configuration dispose d'informations supplémentaires fournies par le codage *CRC*, qui peuvent être exploitées de manière exhaustive.

6.5.1 Idée: exploitation des informations du codage concaténé

L'objectif d'une concaténation de deux codes est la création d'un code plus performant, en bénéficiant des avantages de chaque code [Proa95]. Dans le cas des récents standards de communication, le code en bloc est chargé du contrôle de l'intégrité du message, qui a été protégé par un codage convolutif.

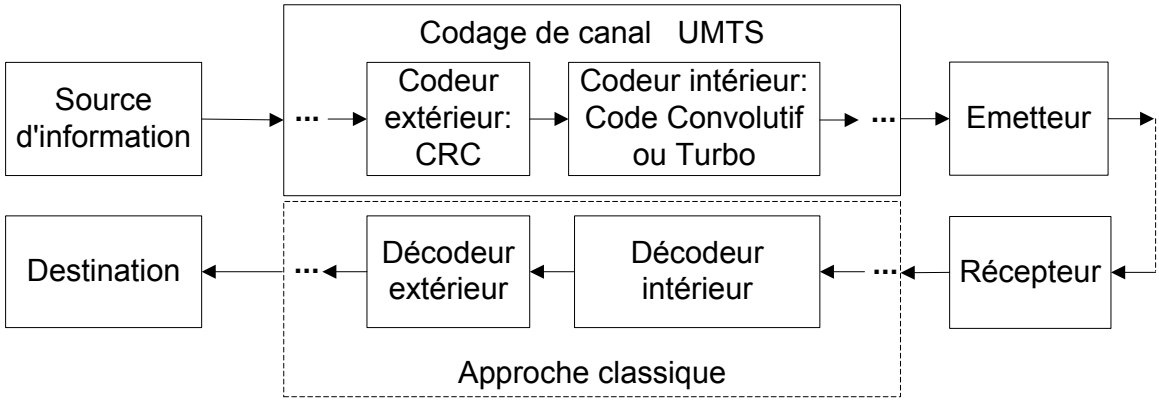

Figure 6-14: diagramme des codes concaténés en série, dans le contexte du standard *UMTS*.

Lors du décodage convolutif, l'intégrité du message reconstruit est difficile à confirmer, sans réduire sensiblement la potentialité de protection du code utilisé. Le codage en bloc élimine ainsi cet inconvénient, en vérifiant l'éventuelle présence d'erreurs dans le message (Section 2.4). Cet enchaînement est demandé par les applications sensibles, pour lesquelles l'utilisation d'un message corrompu provoque des effets désastreux. Dans le cas de la détection d'un message corrompu, il est ainsi prévu que le message soit retransmis (mode de contrôle d'erreurs *ARQ*) ou qu'il y ait une prise de

mesures spéciales de recouvrement d'erreurs (*Error Concealment*, mode de contrôle d'erreurs *FEC*).

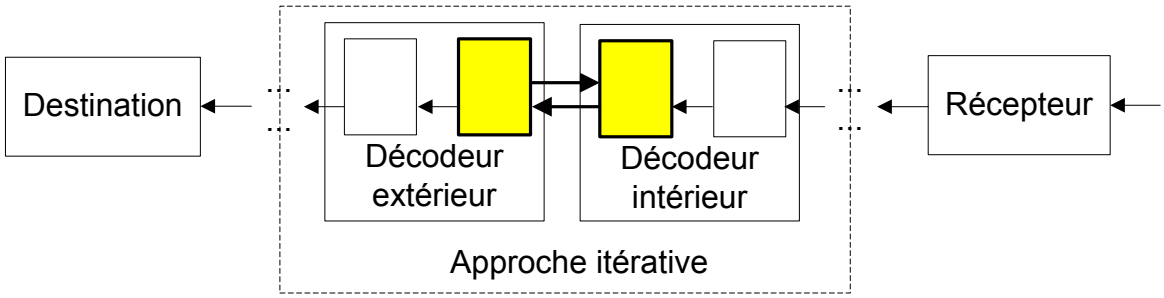

Figure 6-15: principe général à la base des décodages itératifs**.**

Le décodage de cette double protection est normalement exécuté de manière séquentielle, ce qui consiste pratiquement en la reconstruction du message suivie par sa vérification (Figure 6-14). L'avantage de pouvoir développer individuellement les deux procédures de décodage est contre-balancé par une utilisation sous-optimale des ressources disponibles. Une interaction entre les deux décodeurs permettrait de mieux exploiter les informations mises à disposition par le code concaténé (Figure 6-15).

6.5.2 Principe de décodage: livraison d'une liste de messages

La livraison d'une série de messages au décodeur extérieur, qui les examine dans l'intention d'identifier l'éventuel message correct [Kühn97], est un principe de décodage intéressant.
Dans la suite de l'étude, ce principe sera utilisé pour l'exploration des potentialités offertes, d'une part à cause de la souplesse d'une architecture basée sur les *DSP*, et d'autre part en raison de la structure de codage de canal établi par le standard *UMTS*.

En envisageant l'amélioration du rapport entre qualité et complexité de l'opération de décodage, ce principe itératif peut être exploité pour:

- Améliorer la qualité de protection. Les informations supplémentaires de la validation *CRC* sont utilisées afin d'améliorer la protection contre les erreurs, en limitant l'augmentation de la charge de calcul totale.
- Réduire la complexité de calcul de l'opération de décodage. Dans ce cas, l'exploitation des informations envisage la réduction de la charge de calcul du décodage convolutif, tout en gardant inchangée la qualité de protection contre les erreurs.

Le facteur indispensable est la présence d'un code en bloc possédant des capacités de détection d'erreurs suffisantes à l'exécution correcte d'un tel décodage itératif. Puisque le cas d'une erreur non détectée (*undetected error*) doit être considéré comme désastreux, l'application de ce principe doit être absente de fausses validations de messages. L'emploi de ce principe itératif dans le cadre de la téléphonie mobile *GSM* a été en effet limité par le nombre considérable de fausses validations, causées par une capacité de détection insuffisante [Kühn97].
L'évaluation des potentialités des quatre codes *CRC* des standards *UMTS* (Annexe *B*, Section *B.1*) a tout d'abord confirmé des capacités de détection supérieures à celles des standards *GSM*, ce qui rend ce principe de décodage attrayant. Les nombreuses simulations dans le contexte de codage *AMR-NB* à 12.2kbps ont ensuite prouvé la faisabilité de ce principe de décodage, en ne révélant aucun cas de fausse validation.

Dans la suite du chapitre, deux méthodes de base sont proposées pour prouver les potentialités et la faisabilité de ce principe de décodage dans le contexte de l'*UMTS*. Les deux méthodes améliorent le rapport entre la qualité de protection et la charge de calcul, l'une en augmentant la qualité de protection, l'autre en réduisant la complexité du décodage convolutif. Les potentialités de ces deux méthodes seront comparées avec celles de l'algorithme de Viterbi. L'objectif de ce décodage itératif étant la livraison d'un message intègre, la qualité de protection sera évaluée par le débit de messages erronés (*FER*).

6.6 'List Viterbi Algorithm'

6.6.1 Introduction

Le principe de ce décodage est la livraison itérative des *L* messages les plus probables au décodeur *CRC*, afin d'identifier le message correct.

Le principe de décodage par livraison d'une liste de messages permet l'identification du message correct dans une liste de messages prometteurs. Du point de vue de la qualité de décodage, l'obtention d'une série de *L* messages globalement les plus probables représente le cas idéal: la limite inférieure de qualité de ce décodage itératif (*L*=1: livraison du message globalement le plus probable) coïncide ainsi avec celle de l'algorithme de Viterbi.

Le point de départ des stratégies pour l'obtention des *L* messages globalement les plus probables est l'algorithme de Viterbi, qui garantit une analyse efficace

et complète de l'espace de codage. Par contre, l'algorithme de Viterbi ne permet pas la coexistence de chemins globalement les plus probables et, de plus, il est incapable de fournir une série de messages[33]. L'inadéquation de l'algorithme à fournir une série de messages les plus probables peut être compensée par:

- La prise en considération à chaque nœud des *L* meilleurs chemins. Les algorithmes appartenant à cette classe sont nommés *Parallel List Viterbi Algorithm* ('parallèle *LVA*') ou *Parallel Generalized Viterbi Algorithm* [Sesh94]. Selon ce principe, chaque nœud reçoit 2^bL chemins de manière à pouvoir ensuite en sélectionner les *L* meilleurs [Sesh94]. Ces chemins seront ensuite transmis aux nœuds suivants (Figure 6-16).
- L'exploitation (à posteriori) des différences entre la métrique cumulée des chemins survivants et celle des perdants, à chaque nœud de chaque niveau de profondeur du treillis (*Serial List Viterbi Algorithm* ou *Serial Generalized Viterbi Algorithm* [Nill95] [Sesh94] [Kühn97]). En utilisant correctement ces différences, les *L* meilleurs chemins globaux peuvent être ainsi identifiés et ensuite reconstruits.
- L'utilisation des estimations de la fiabilité Λ offerte par les méthodes *SOVA*. L'exploitation de ces informations permet la génération (à posteriori) des *L* messages retenus les plus probables (*List SOVA* [Nill95] [Kühn97]). En raison d'une génération de la liste de chemins dépendante de la qualité de l'estimation de la fiabilité, cette stratégie ne garantit pas nécessairement l'identification de *L* chemins globalement les plus probables.

En raison d'un contexte d'exécution en temps réel, le facteur déterminant pour le choix de la stratégie est sa charge de calcul.

Malgré l'absence dans la littérature d'évaluations précises de la complexité de calcul de ces stratégies, il est évident que les techniques basées sur les algorithmes *parallèles LVA* et *SOVA* comportent une augmentation considérable de la charge de calcul nécessaire à leur exécution. On peut estimer que l'utilisation d'algorithmes *parallèles LVA* comporte une augmentation de complexité d'un facteur de *L* [Sesh94]. Les stratégies exploitant les algorithmes *SOVA* impliquent aussi une importante augmentation de la complexité. Le chapitre précédent montre que leur charge

[33] L'algorithme de Viterbi adopte le thèoreme de la non-optimalité (*theorem of nonoptimality path*) afin de livrer le message globalement le plus probable (Sous-section 5.2.3).

de calcul est de 1,5 à 2 fois celle requise par l'algorithme de Viterbi (Section 5.4).

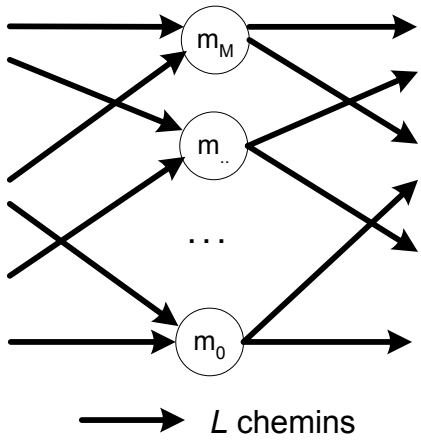

Figure 6-16: principe à la base du parallèle *List Viterbi Algorithm* (LVA).

La stratégie d'une exploitation des différences des métriques cumulées est par contre très intéressante. L'augmentation de la complexité n'est déterminée que par la création de la liste de messages après l'exécution de l'algorithme de Viterbi.

6.6.2 Méthode Serial List Viterbi Algorithm

Un avantage considérable de cette stratégie de décodage, qui sera utilisée dans cette étude et sera nommée simplement *List Viterbi Algorithm* ('*LVA*'), est la possibilité de séparer le développement du décodeur intérieur en deux parties. La première partie s'occupe principalement de l'exécution de l'algorithme de Viterbi. La seconde partie se concentre sur la génération de la série de *L* messages, en utilisant les tables contenant les différences des métriques cumulées et les informations pour la reconstruction des chemins. Le nombre de chemins qui peuvent être reconstruits n'est limité que par les exigences de l'application: cette stratégie permet en effet la reconstruction de tous les messages souhaités ainsi que la détermination de leurs métriques finale. Ces valeurs de métrique sont déterminées à partir de la métrique du meilleur chemin, par la formule:

$$\text{métrique finale}_{\substack{\text{meilleur}\\\text{chemin}}} + \sum_{np=B}^{1} \begin{cases} 0, & \text{si le chemin à cette profondeur } np \text{ est le survivant} \\ \Delta métrique_{np}, & \text{autrement} \end{cases} \tag{6.5}$$

où Δ*métrique* représente la différence des métriques entre le chemin survivant et celui perdant mais choisi pour la reconstruction.

L'exécution de cette méthode de décodage itératif peut être ainsi caractérisée par les étapes suivantes:

1. L'algorithme de Viterbi est tout d'abord exécuté. Pendant son exécution, les différences entre les métriques cumulées des deux chemins (qui se réunissent dans chaque nœud à chaque profondeur) sont systématiquement sauvegardées dans une table.
2. Après l'accomplissement de l'algorithme de Viterbi, la variable responsable du contrôle du nombre des messages générés est initialisée (l=1).
3. Le l-ème chemin est tout d'abord identifié et ensuite décodé. Ces deux opérations utilisent les informations relatives aux différences de métrique et à la reconstruction des chemins.
4. Le message obtenu est analysé par le décodeur extérieur: si le message est validé, le décodage itératif est terminé.
5. Si les L meilleurs messages globaux n'ont pas encore été tous contrôlés, le décodage itératif continue: la variable l est incrémentée et on passe au point 3. Sinon, la procédure de décodage est interrompue sans pouvoir délivrer un message valide.

6.6.3 Performances de cette méthode itérative

L'implantation de cette méthode est conforme à la configuration de base présentée pour l'implantation de l'algorithme de Viterbi.

La réalisation de cette méthode se base sur la même implantation de l'algorithme de Viterbi, qui est utilisé à titre de référence. Toutefois, en raison de la reconstruction de plusieurs chemins, le principe de la troncation de la mémoire (responsable de la sauvegarde des informations nécessaires à la reconstruction des chemins) ne peut pas être exploité.

Par rapport à l'obtention de la liste de L messages, l'implantation de la procédure de création des messages suit les critères suivants:

- Les messages délivrés sont les L meilleurs messages globaux.
- Le nombre maximal de messages délivrés (pour l'étude) est: L=2, 4, 8, 16, 32, 64, 128, 256 et 512 messages;

- Les messages sont délivrés selon le critère d'une probabilité décroissante et la procédure itérative s'arrête si le message est validé par le décodeur *CRC*. Les détails de l'opération de validation *CRC* sont illustrés dans l'Annexe *B* (Section *B.1*).
- L'implantation software de cette méthode s'oriente vers des plates-formes à 16 bits.
- Une surveillance de l'occurrence de fausses validations est effectuée par un contrôle systématique de l'exactitude de la coïncidence entre le message correct et celui identifié par le décodeur *CRC*.

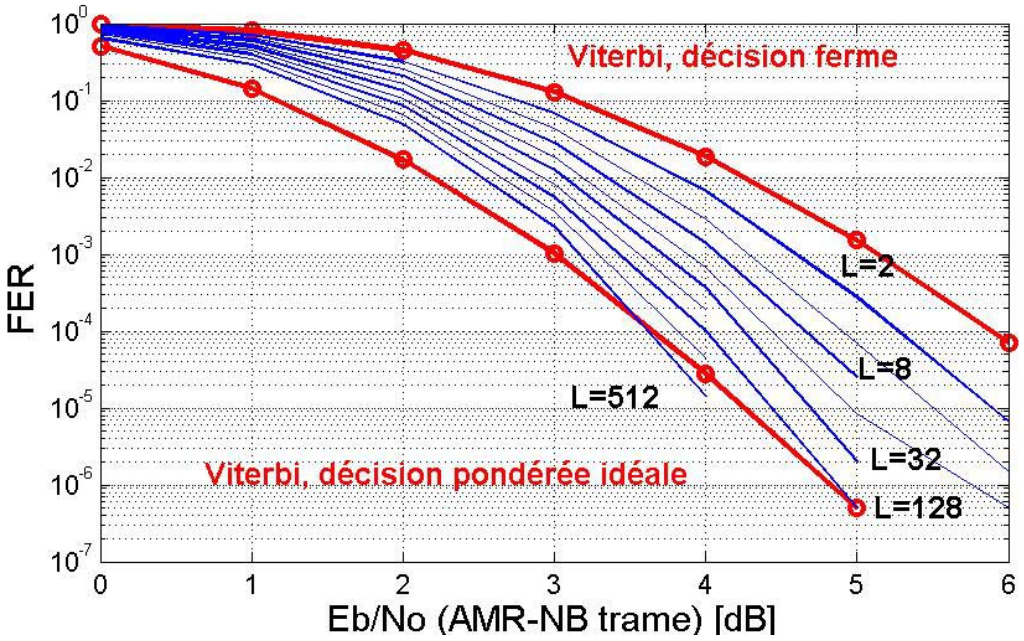

Figure 6-17: qualité de protection contre les erreurs de transmission offerte par la méthode itérative *List Viterbi Algorithm*. Le graphe rouge en trait gras se rapporte à la qualité de référence obtenue par l'algorithme de Viterbi. Simulation basée sur 2 millions de trames du codeur de parole *AMR-NB* à 12.2 kbps.

Qualité de protection contre les erreurs de transmission

Les résultats de la qualité de protection (utilisant le service de parole *AMR-NB* à 12.2 kbps) sont illustrés en Figure 6-17. Les potentialités de protection de la méthode *List Viterbi Decoding* sont prouvées: l'augmentation du nombre de messages délivrés au décodeur *CRC* améliore sensiblement la qualité de protection. Par contre, la relation entre le gain de codage (*Coding Gain*) et le nombre maximal de messages délivrés n'est pas linéaire (Table 6-8), en raison de la probabilité décroissante des messages livrés. La Figure 6-17 met aussi

en évidence l'influence des conditions de transmission sur le gain de codage. Les résultats de cette simulation montrent que le gain de codage augmente avec l'amélioration des conditions de transmission.

Aucune fausse validation n'a pu être identifiée lors des simulations effectuées. Bien que la probabilité de rencontrer des situations de fausses validations n'est jamais nulle, ces résultats quantitatifs prouvent que la capacité de détection d'erreurs du code en bloc utilisé est suffisante à garantir l'exécution correcte de cette méthode itérative.

Nombre maximal de messages L	Gain de codage à FER=10^{-3} [dB]	Nombre maximal de messages L	Gain de codage à FER=10^{-3} [dB]
2	0.5	64	1.6
4	0.8	128	1.7
8	1.0	256	1.9
16	1.3	512	2.0
32	1.5		

Table 6-8: vue d'ensemble du gain de codage par rapport à la qualité de protection de l'algorithme de Viterbi.

Complexité de calcul de cette méthode itérative

Afin de permettre l'analyse correcte de la complexité de calcul de cette méthode, la nouvelle réalisation de la tâche concernant la génération des messages doit être tout d'abord discutée.

La génération des L meilleurs messages globaux utilise deux tables: la première contenant les informations nécessaires à la reconstruction des chemins survivants, la seconde comprenant les différences de métriques cumulées.

A partir de ces deux tables, la stratégie implantée pour la génération des messages les plus probables suit les étapes suivantes:

1. Le message globalement le plus probable (l=1) est obtenu par décodage du chemin indiqué par l'algorithme de Viterbi. En analysant les chemins perdants rencontrés lors de la génération de ce message (Figure 6-19), une table contenant toutes les informations nécessaires à la reconstruction des L-1 chemins (temporairement) les plus probables est construite (Figure 6-18). Les informations sont la valeur de la métrique et les indications des niveaux de profondeur, où le chemin suit des chemins perdants. Cette table est générée et

gérée selon la méthode d'insertion des nouvelles informations *Straight Insertion* [Pres92].

2. Le message globalement le plus probable (l=1) est ensuite contrôlé par le décodeur *CRC*. Si le message est validé, la procédure s'arrête.
3. La variable l est incrémentée (l=l+1)
4. Le chemin le plus probable de la table est utilisé pour la construction du l-ème message globalement le plus probable. Ce message est obtenu en suivant les informations de reconstruction des chemins et les indications des niveaux de profondeur des chemins perdants (Figure 6-19). Après la dernière déviation depuis un chemin survivant, la table des L-l meilleurs chemins est mise à jour (*Straight Insertion*) en analysant les métriques des chemins perdants rencontrés (Figure 6-19). Cette stratégie évite de considérer des chemins perdants précédemment déjà analysés.
5. Ce l-ème message est ensuite contrôlé par le décodeur *CRC*. Si le message est validé, cette procédure s'arrête.
6. Si les L messages ne sont pas encore reconstruits, on passe à l'étape 3. Sinon, la procédure s'arrête sans pouvoir livrer un message valide.

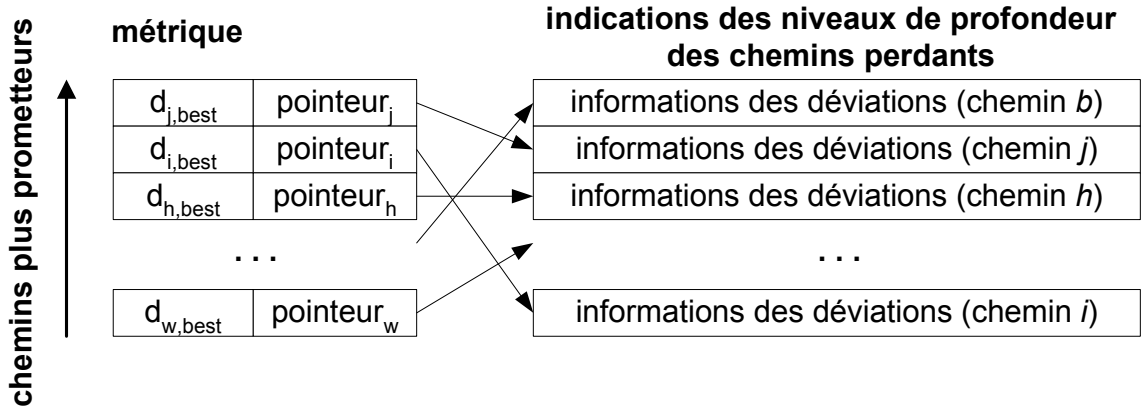

Figure 6-18: table contenant toutes les informations nécessaires à la reconstruction des chemins temporairement les plus probables. L'utilisation des pointeurs permet de réduire les coûts de mise à jour de la table par insertion et déplacement des éléments (*Straight Insertion* [Pres92]).

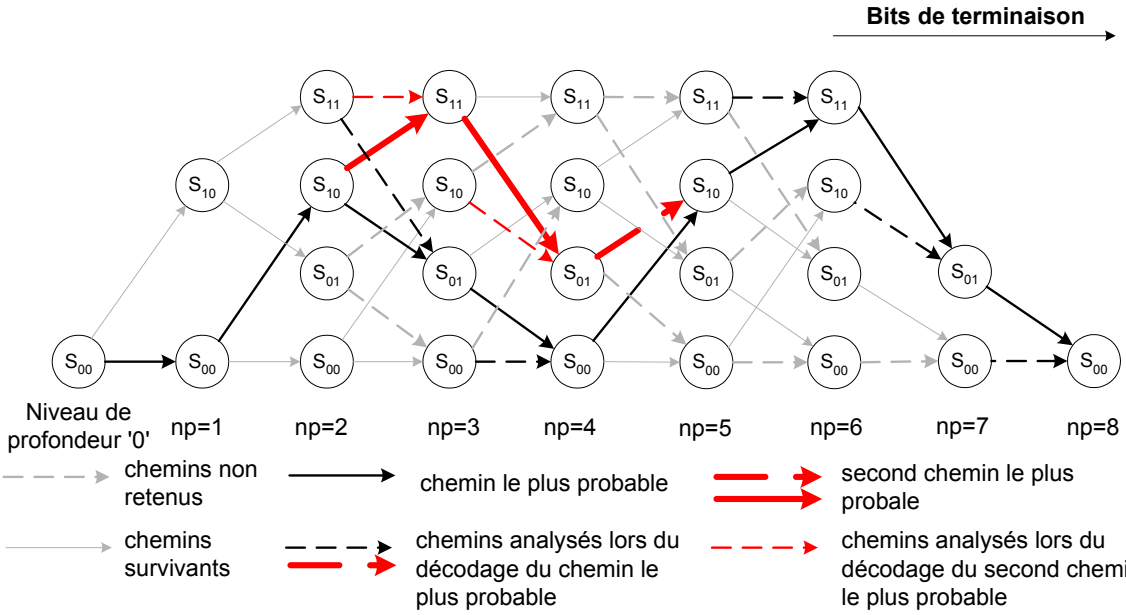

Figure 6-19: exemple de procédure de reconstruction des deux meilleurs messages globaux avec l'analyse des chemins perdants.

Les estimations de la charge de calcul de la méthode itérative sont résumées dans la Table 6-9, Table 6-10 et Table 6-11. La première examine dans le détail la méthode *List Viterbi* livrant au maximum 8 messages au décodeur extérieur (L=8). La Table 6-10 et la Table 6-11 résument les résultats concernant la livraison de 2, 4, 8, 16, 32, 64, 128, 256 et 512 messages.

Les résultats qui en découlent sont conformes aux attentes. La structure algorithmique de cette méthode itérative permet en effet de limiter l'augmentation de la complexité, la rendant intéressante du point de vue du rapport qualité/complexité de calcul.
La séparation de la méthode en deux parties principales a rendu l'implantation de la tâche de génération des L messages responsable de l'augmentation de complexité. L'importance de l'augmentation dépend ainsi du positionnement du message correct dans la série de messages (Table 6-9) et du nombre maximal L de messages livrables (Table 6-10 et Table 6-11).

	Nombre moyen d'opérations observées							
	Sans erreurs de transmission				**Exécution exhaustive de l'algorithme** (imposé)			
	Format des variables [bits]				Format des variables [bits]			
	32	*16*	*8*	***Total***	*32*	*16*	*8*	***Total***
Additions	44540	146	0	**44686**	44540	760	0	**45300**
Incrémentations	0	1322	0	**1322**		2092	0	**2092**
Soustractions	22525	121	0	**22646**	22525	842	0	**23367**
Multiplications	0	0	0	**0**	0	0	0	**0**
Divisions	0	0	0	**0**	0	0	0	**0**
Transferts de données	11	45847	0	**164979**	119146	46752	0	**165898**
Logique	12750	2613	13	**130131**	127733	3929	104	**131766**
Décalages	70721	282	1	**71004**	71127	2249	8	**73384**
Contrôles			102761	**102761**			106329	**106329**
Pointeur: assignations			654	**654**			690	**690**
Pointeur: arithmétique			716	**716**			2182	**2182**
Boucle: initialisations	24166	112	0	**24278**	24166	133	0	**24299**
Boucle: contrôles			97782	**97782**			98573	**98573**
Boucle:	72545	952	0	**73497**	72545	1722	0	**74267**
Grand total				**734456**				**748147**

Table 6-9: complexité de calcul détaillée de la méthode itérative basée sur le *List Viterbi Algorithm* exploitant au maximum *L*=8 messages. Les 500 trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les écarts observés sont inférieurs à 1,4%. Le fond gris met en évidence les compteurs d'occurrence définis par les organisations *ITU* et *ETSI*.

Avec l'augmentation du nombre de chemins *L*, l'importance du positionnement du message correct s'accroît remarquablement. Le principal responsable de la hausse importante de la complexité est la gestion de la table des informations des chemins les plus prometteurs: à partir de *L*=64 messages, la stratégie de gestion de la table par *Straight Insertion* perd son efficacité, en exigeant un nombre élevé d'opérations. L'emploi de cette méthode de décodage itérative avec un nombre élevé de messages implique la modification de cette stratégie de gestion.

	Nombre moyen d'opérations observées					
	Viterbi avec val. *CRC* (*)	**List Viterbi Algorithm: Exécution exhaustive de l'algorithme**(imposé)				
Contexte	E_b/N_0=0.0dB	L=2	L=4	L=8	L=16	L=32
Additions	44551	44655	44866	45300	46216	48240
Incrémentations	1320	1432	1652	2092	2972	4732
Soustractions	225	22742	22951	23367	24199	25863
Multiplications	0	0	0	0	0	0
Divisions	0	0	0	0	0	0
Transferts de données	161317	165052	165326	165898	167138	170002
Logique	132694	130382	130830	131766	133681	137441
Décalages	71003	71344	72024	73384	76104	81544
Contrôles	102615	103137	104193	106329	110697	119817
Pointeur: assignations	612	626	644	690	830	1302
Pointeur: arithmétique	513	928	1346	2182	3854	7198
Boucle: initialisations	24269	24275	24283	24299	24331	24395
Boucle: contrôles	97734	97849	98087	98573	99593	101825
Boucle: incrémentations	73464	73574	73801	74267	75247	77399
Grand total	**710317**	**735996**	**740003**	**748147**	**764862**	**799758**
Complexité relative[34]	1.00	1.04	1.04	1.05	1.08	1.13
		List Viterbi Algorithm: Sans erreurs de transmission				
Grand total		**734179**	**734217**	**734456**	**735263**	**738406**
Complexité relative[34]		1.03	1.03	1.03	1.04	1.04

Table 6-10: vue d'ensemble de la complexité de calcul de la méthode itérative basée sur le *List Viterbi Algorithm* exploitant au maximum *L*=2, 4, 8, 16 et 32 messages. Les 500 trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les écarts observés sont inférieurs à 1,4%. (*)L'opération de validation CRC est illustrée dans l'Annexe *B* (Section *B.1*).

[34] On rappelle que la méthode de référence est l'algorithme de Viterbi. La complexité relative est obtenue en divisant la complexité de calcul (nombre moyen d'opérations observées: 'grand total') de la méthode analysée par celle de l'algorithme de Viterbi (donnée en Table 6-6).

	Nombre moyen d'opérations observées					
	Viterbi avec val. *CRC* (*)	**List Viterbi Algorithm: Exécution exhaustive de l'algorithme** (imposé)				
Contexte	E_b/N_0=0.0dB	L=32	L=64	L=128	L=256	L=512
Grand total	**710317**	**799758**	**875765**	**1053806**	**1514077**	**2850308**
Complexité relative[34]	1.00	1.13	1.23	1.48	2.13	4.01
		List Viterbi Algorithm: Sans erreurs de transmission				
Grand total		**738406**	**750806**	**797065**	**899840**	**1105428**
Complexité relative[34]		1.04	1.06	1.12	1.27	1.56

Table 6-11: vue d'ensemble de la complexité de calcul de la méthode itérative basée sur le *List Viterbi Algorithm* exploitant au maximum *L*=32, 64, 128, 256 et 512 messages. Les 500 trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les écarts observés sont inférieurs à 1%. (*)L'opération de validation CRC est illustrée dans l'Annexe *B* (Section *B.1*).

6.7 'List Decoding intégrant la validation du CRC'

Les résultats de la méthode précédente prouvent la faisabilité d'une exploitation intensive des informations générées par le codage en bloc.
La méthode *List Viterbi Algorithm* améliore la qualité de protection de l'algorithme de Viterbi au moyen d'une utilisation plus exhaustive de ces informations. La méthode montre un bon rapport entre la qualité de protection et la complexité de calcul. Toutefois, la complexité de calcul est fortement influencée par l'incorporation de l'algorithme de Viterbi dans la méthode itérative: la complexité croît de manière exponentielle avec l'augmentation de la longueur de contrainte *K* du code convolutif.

6.7.1 Nouvelle stratégie de décodage

L'efficacité du principe de décodage itératif permet d'envisager une exploitation exhaustive des informations reçues dans le but de réduire la charge de calcul de l'opération entière de décodage, tout en conservant la même qualité de protection que l'algorithme de Viterbi.

Ce nouvel objectif peut être atteint tout d'abord en choisissant une méthode de décodage convolutif moins exigeante en termes de charge de calcul. En raison

de l'efficacité de l'algorithme de Viterbi, une réduction de la complexité de calcul ne peut être obtenue qu'au détriment d'une analyse complète de l'espace de codage. La perte inévitable de qualité de protection (Table 6-12) peut être ensuite compensée par l'apport d'informations supplémentaires du codage *CRC*.

Nombre de chemins survivants pris en compte	**Facteur de réduction de l'espace de codage**	**Rapport signal sur bruit E_b/N_0 produisant un *BER* de 10^{-3}**	**Gain de codage**
Tous (Algorithme de Viterbi)	-	4.1 dB	-
128 chemins (états)	2	4.2 dB	-0.1 dB
64 chemins (états)	4	4.6 dB	-0.5 dB
32 chemins (états)	8	5.1 dB	-1.0 dB
16 chemins (états)	16	5.6 dB	-1.5 dB

Table 6-12: vue d'ensemble de la dégradation introduite en raison d'une analyse incomplète de l'espace de codage. Contexte de codage: codeur *UMTS* avec rendement R_c=1/3.

Le même principe de décodage livrant une série de messages au décodeur *CRC* est à nouveau adopté. L'algorithme utilisé pour la génération des messages n'est plus l'algorithme de Viterbi, mais un algorithme qui favorise l'aspect de la faible complexité de calcul: l'algorithme *List Decoding* (Figure 6-20).

La charge de calcul de la méthode *List Decoding* est en effet fonction du nombre *L* de chemins considérés à chaque niveau de profondeur. De plus, une liste des chemins les plus prometteurs est intrinsèquement disponible à tous les niveaux de profondeur (Sous-section 5.2.2). La série de messages est générée en décodant les chemins contenus dans la liste finale des *L* chemins les plus prometteurs. La génération des messages bénéficie de la non-application du principe du survivant, ce qui permet la libre coexistence des meilleurs chemins (Figure 6-21). Cette caractéristique est indispensable pour obtenir une liste dont les messages possèdent de bonnes probabilités.

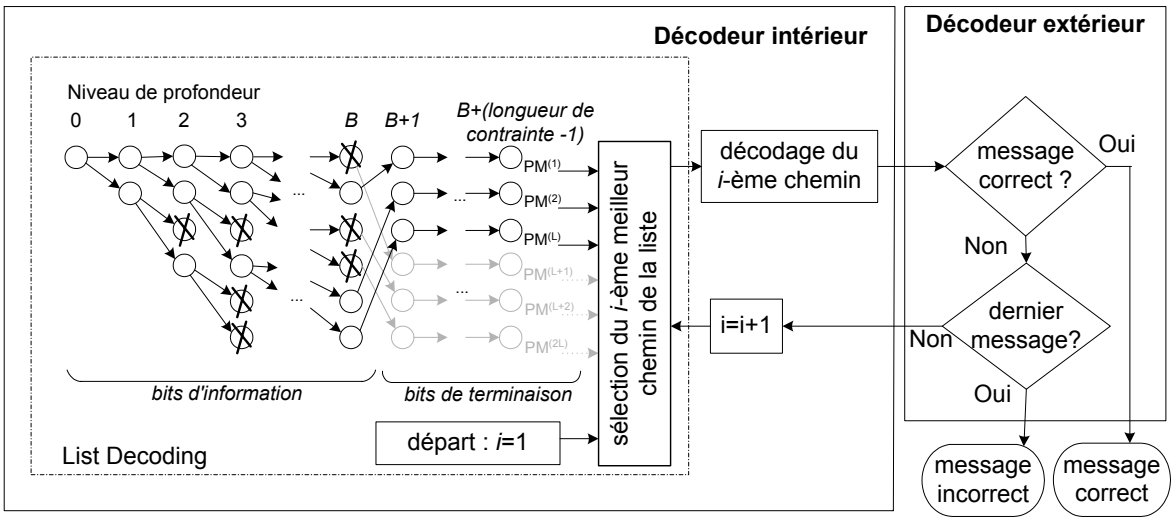

Figure 6-20: représentation graphique du principe de cette méthode itérative *List Decoding intégrant la validation CRC*.

6.7.2 Considérations sur l'implantation

L'implantation de cette méthode utilise la même représentation numérique des métriques et le même système de sauvegarde des informations de reconstruction des chemins que ceux qui ont été adoptés pour l'implantation de l'algorithme de Viterbi.

Par rapport à la sauvegarde des informations de reconstruction, l'algorithme *List Decoding* permet deux stratégies opposées de sauvegarde des informations. Le système le plus économe en mémoire sauvegarde seulement un bit ayant fonction d'indicateur (*flag*). Il signale si le chemin a été inséré dans la liste des L chemins qui seront ensuite prolongés. Grâce à cette indication et au positionnement du chemin dans la liste des prolongations, les états précédents des chemins peuvent être reconstruits de manière itérative. Les détails et les explications sont disponibles dans le livre de Johannesson [Joha99]. De manière cohérente avec l'implantation de l'algorithme de Viterbi, le système le moins complexe (qui sauvegarde directement la valeur de l'état précédent) a été choisi pour l'implantation de cette méthode.
En raison d'une reconstruction d'une large quantité de messages, la méthode de troncation de la mémoire n'est pas applicable.

6.7.3 *Effet de compensation par l'exploitation exhaustive du codage CRC*

La qualité de protection de cette méthode itérative est déterminée par la probabilité de trouver le message correct dans la liste de messages livrés au décodeur *CRC*. Cette probabilité est évidemment fonction du nombre L de messages livrés ainsi que des affinités entre les symboles reçus et ceux générés par les messages.

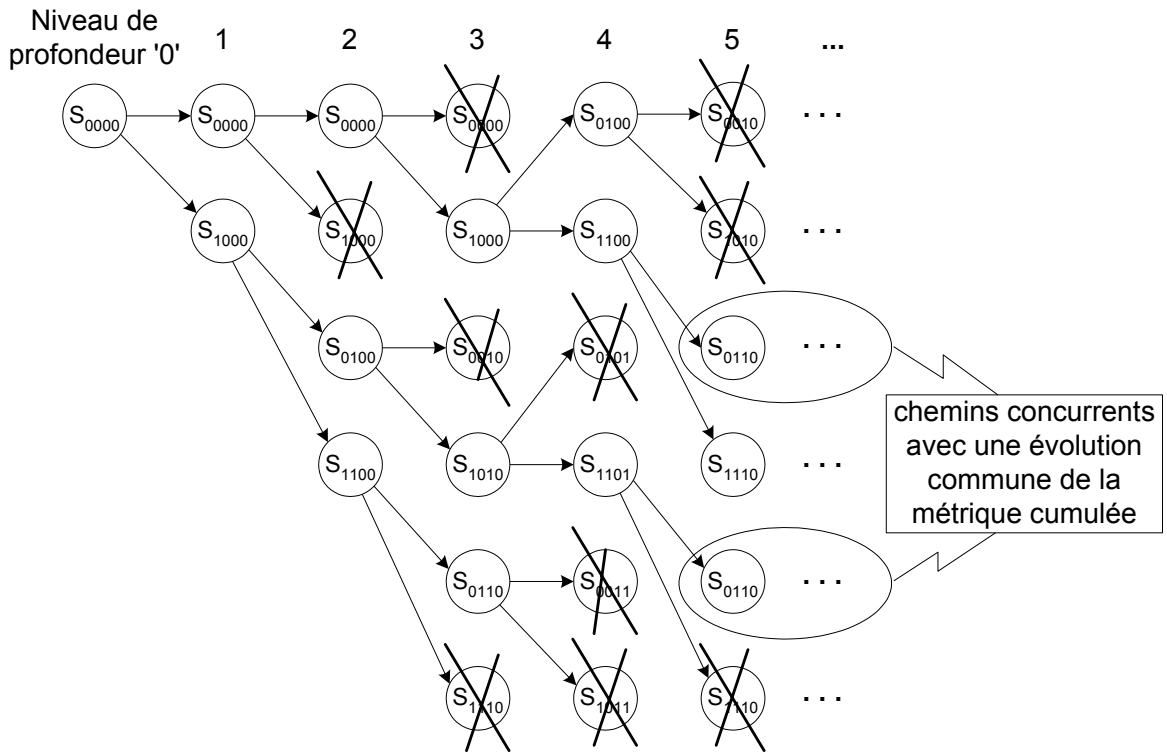

Figure 6-21: l'algorithme *List Decoding* permet la libre coexistence de chemins "parallèles" qui ont une évolution commune de la métrique cumulée. Ceci est dû à l'inapplication du critère du survivant.

A la différence de la méthode *List Viterbi Decoding* qui livre les L messages globalement les plus probables, l'algorithme *List Decoding* traite les L chemins les plus prometteurs de chaque niveau de profondeur. Cet algorithme peut être ainsi affecté par le phénomène gênant du rejet, c'est-à-dire la disparition du(des) chemin(s) globalement le(s) plus probable(s) de la liste des L chemins (Section 5.2.2). Par conséquent, les L messages livrés ne correspondent pas nécessairement aux L messages globalement les plus probables.

Caractéristiques de l'implantation de cette méthode itérative

L'implantation de cette solution suit les critères suivants:

- Le nombre maximal de messages délivrés coïncide avec le nombre *L* des messages les plus prometteurs de la méthode *List Decoding*. A titre d'étude, ce nombre sera aussi étendu à *2·L*, en n'exécutant pas la dernière sélection avant le traitement des bits de terminaison (Figure 6-20).
- Les messages sont délivrés selon une affinité décroissante (c'est-à-dire selon une valeur de métrique croissante) et la procédure itérative s'arrête si le message a été validé. Les détails de l'opération de validation *CRC* sont illustrés dans l'Annexe *B* (Section *B.1*).
- Une surveillance de l'occurrence de fausses validations est effectuée pendant les simulations. Le message identifié par le décodeur *CRC* est systématiquement comparé avec le message envoyé.
- L'implantation software de cette méthode s'oriente vers des plates-formes à 16 bits.

Effet de compensation

Les qualités de protection de cette méthode *List Decoding intégrant la validation CRC* démontrent la validité du principe de la compensation de la dégradation de la protection due à l'utilisation de l'algorithme *List Decoding* (Figure 6-22).

Cette étude montre que la perte de qualité due à l'analyse incomplète de l'espace de codage peut être compensée en délivrant plusieurs messages au décodeur *CRC*. La qualité de protection n'est maintenant influencée que par la perte possible du chemin correct par l'algorithme *List Decoding*: si ce chemin apparaît dans la liste finale des *L* chemins, la procédure itérative permet l'identification et la livraison du message correct.

L'importance de l'amélioration apportée par la livraison des messages dépend strictement du nombre *L* de chemins par l'algorithme *List Decoding*. Un nombre *L* élevé diminue les risques d'un rejet du chemin correct causé par une accumulation (soit constante, soit temporairement défavorable) d'erreurs de transmission (Table 6-13). L'extension du nombre de messages de *L* à *2L* par la non-exécution de la dernière sélection des chemins n'apporte en effet pas d'amélioration sensible (Figure 6-22). Cette procédure génère *L* messages supplémentaires, qui possèdent des probabilités très réduites d'être le message correct.

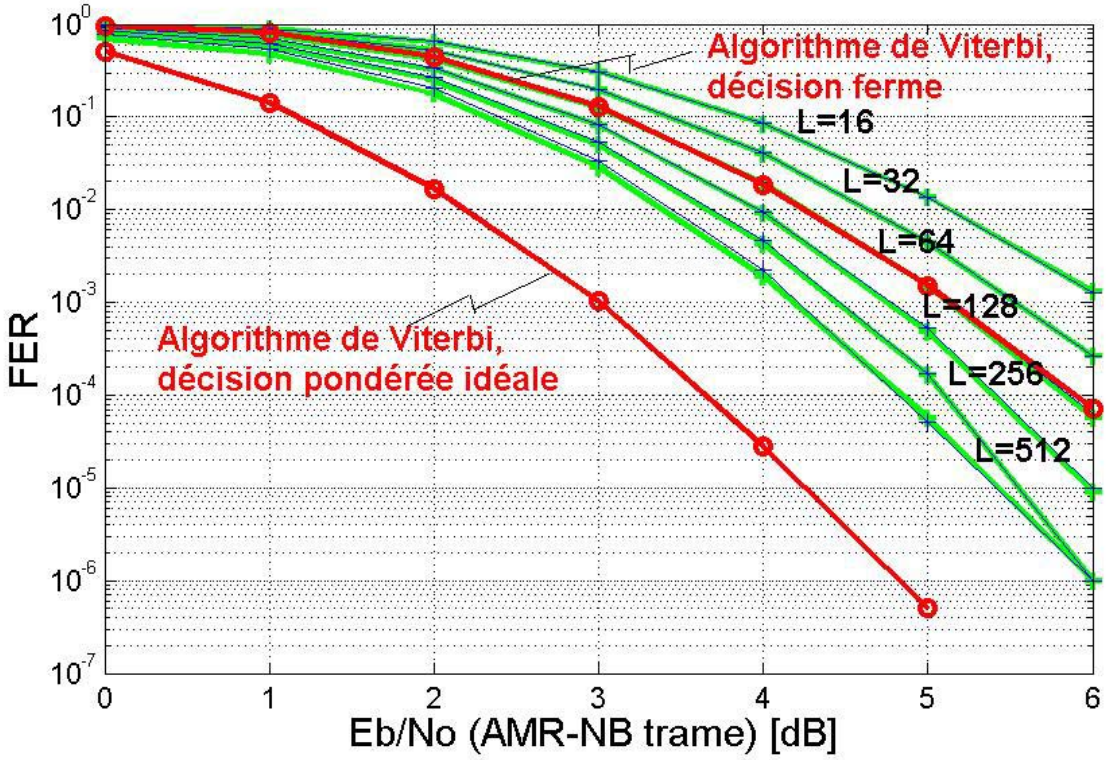

Figure 6-22: qualité de protection de la méthode itérative *List Decoding intégrant la validation CRC* et de l'algorithme de Viterbi. Le graphe rouge en trait gras représente la qualité de protection de l'algorithme de Viterbi, le graphe bleu en trait fin celle de la méthode itérative livrant L messages. Le graphe vert en trait gras indique la qualité de protection obtenue par la livraison de $2 \cdot L$ messages (élimination de la dernière sélection des L messages). Simulation basée sur 2 millions de trames du codeur de parole *AMR-NB* à 12.2 kbps.

Nombre de chemins L	Importance du bruit causant un *FER* de 10^{-3} [E_b/N_0]		
	Algorithme List Decoding	***Méthode itérative List Decoding intégr. CRC***	**Algorithme de Viterbi**
32	5.57 dB	**5.53 dB**	**5.13 dB**
64	5.27 dB	**5.14 dB**	
128	5.16 dB	**4.78 dB**	

Table 6-13: exemple d'amélioration des performances observé dans le cadre du service de parole *AMR-NB* à 12.2 kbps, grâce à la méthode itérative.

En ce qui concerne les simulations (service de parole *AMR-NB* à 12.2 kbps), le contrôle systématique de la qualité de détection d'erreurs n'a révélé aucune irrégularité dans le jugement effectué par le décodeur *CRC* (cas de fausses validations). Dans le cas d'une perturbation par l'addition d'un bruit blanc gaussien (de 0.0 à 6.0 dB), ce décodage itératif avec L=64 messages offre une qualité de protection comparable à celle de l'algorithme de Viterbi (Figure 6-22). Dans des conditions de transmission très mauvaises (E_b/N_0<1dB), on peut même envisager de réduire ultérieurement le nombre de messages L livrables.

6.7.4 Complexité de calcul

Après avoir démontré la faisabilité d'une compensation de la qualité de protection par la livraison d'une série de messages, cette sous-section considère la réduction de la charge de calcul de la méthode de décodage.

La complexité de calcul des méthodes itératives de décodage considérées (Figure 6-15) dépend tout d'abord de la méthode de décodage convolutif et ensuite de la procédure itérative de génération et de contrôle de la validité des messages. L'utilisation de l'algorithme *List Decoding* réduit la complexité de cette procédure itérative, mettant intrinsèquement à disposition une liste de chemins dont le décodage permet de générer une série de messages. Du point de vue de la complexité de calcul, le point crucial devient ainsi l'implantation de la méthode *List Decoding*.

Charge de calcul de l'algorithme *List Decoding*: la fonction de sélection

A chaque niveau de profondeur, cet algorithme procède à la prolongation de L chemins, à la mise à jour de $2 \cdot L$ métriques cumulées et à la sélection des L meilleurs chemins (Sous-section 5.2.2). Contrairement à l'algorithme de Viterbi, le déroulement de cet algorithme nécessite l'intervention d'une fonction de tri pour la sélection des chemins dont la complexité de calcul dépend évidemment du nombre de chemins impliqués dans l'opération.

Les fonctions de tri classiques, telles que les méthodes *Straight Insertion* et *Heap Sort*, nécessitent un nombre d'opérations dont l'ordre de grandeur varie entre $(2 \cdot L)^2$ et $2 \cdot L \cdot log_2(2 \cdot L)$ [Pres92]. Avec l'augmentation du nombre L de chemins, les exigences de ces fonctions deviennent trop importantes et compromettent la réduction de la charge de calcul envisagée par l'analyse partielle de l'espace de codage. Il devient ainsi essentiel d'utiliser une solution alternative à ces fonctions de tri classiques.

Procédure de sélection développée

Une procédure de sélection qui utilise la statistique des valeurs des métriques cumulées a été développée et ensuite implantée. La fonction de tri peut ainsi être remplacée par la détermination du seuil délimitant les deux sous-ensembles de métriques (Figure 6-23). L'ordre de grandeur de la complexité de calcul de cette procédure est de $2 \cdot L$. Les détails sont présentés dans l'Annexe *B* (Section *B.2*).

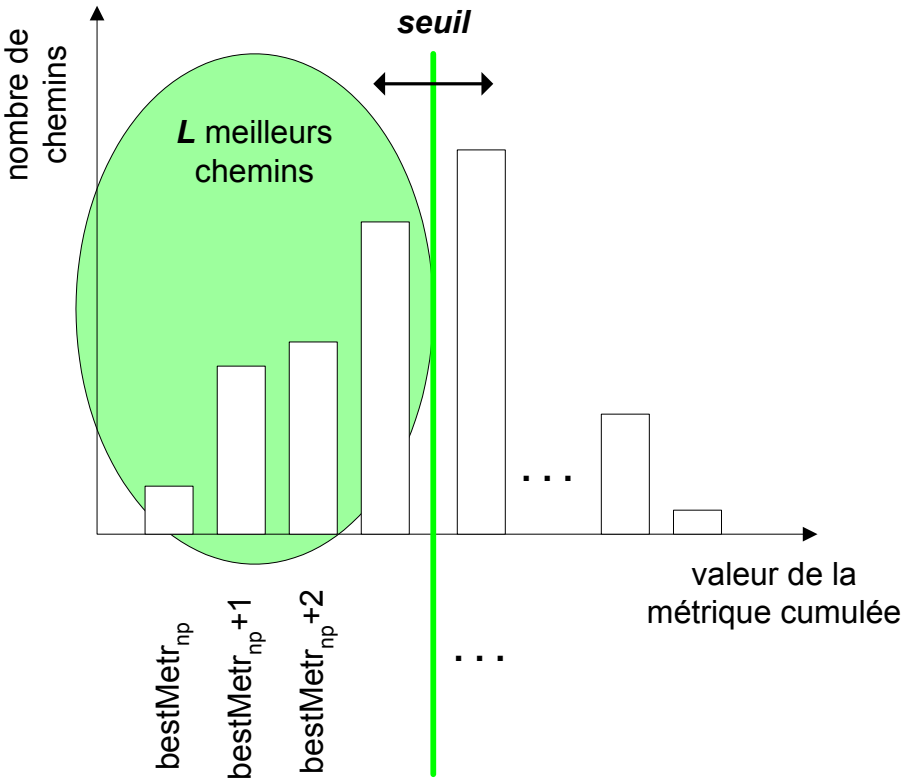

Figure 6-23: recherche du seuil qui permettra ensuite la sélection des *L* meilleurs chemins par simple comparaison. Cette représentation considère des métriques se basant sur la pure distance de Hamming.

Cette procédure s'inspire de l'observation d'une stabilité de la différence maximale entre les valeurs des métriques cumulées (Figure 6-24). Bien que les valeurs aient tendance à augmenter en présence d'erreurs de transmission, la différence maximale entre les métriques *dmm* reste pratiquement constante:

$$dmm = MaxContribBranche \cdot \left\lceil \log_2\left(2 \cdot L\right) \right\rceil, \qquad (6.6)$$

où *MaxContribBranche* est la contribution maximale de la métrique de branche[35] (voir Annexe *B*, Section *B.2*). On peut ainsi constater que les *2·L* valeurs de métriques cumulées sont concentrées dans un écart réduit à *dmm* valeurs.

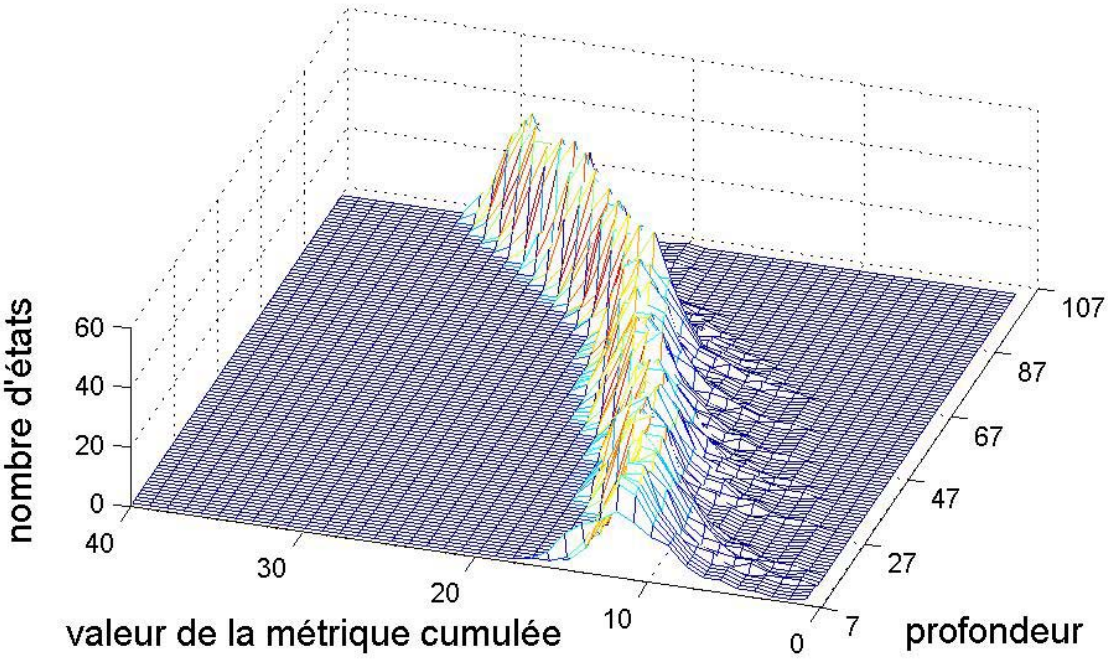

Figure 6-24: exemple de la dynamique relative aux valeurs des métriques cumulées. La distance de Hamming est utilisée comme métrique de branche. Configuration: *List Decoding* avec *L*=64, codeur convolutif avec R_c=1/3 du standard *UMTS* [Ts25212], intensité du bruit de E_b/N_0=6dB, la partie initiale de la procédure de décodage n'est pas affichée.

La forte accumulation des valeurs permet de développer une procédure de sélection exploitant la distribution statistique des métriques cumulées. Le principe est l'établissement (à chaque niveau de profondeur) de la statistique de distribution des métriques cumulées, afin de déterminer le seuil qui permet la sélection des *L* meilleurs chemins (Figure 6-23).

[35] On rappelle que l'unité des métriques de branche est la distance de Hamming. Les valeurs de métriques cumulées indiquent ainsi le nombre d'erreurs rencontrées par les divers chemins.

Du point de vue de la complexité de calcul, cette méthode peut être résumée de la manière suivante:

- Les (*dmm* + *MaxContribBranche*+1) compteurs d'occurrence nécessaires pour établir la statistique des métriques cumulées[36] sont remis à zéro.
- Pendant les opérations de prolongement des chemins, ces compteurs sont incrémentés selon les valeurs des nouvelles métriques cumulées.
- Le seuil est ensuite établi par l'addition des valeurs de ces compteurs d'occurrence, jusqu'à parvenir à l'isolement des L_s chemins les plus prometteurs (où $L_s \geq L$):

$$seuil = \arg\left\{ \min_{seuil} \sum_{stat=0}^{seuil} compteur_{stat} \geq L \right\} \qquad (6.7)$$

- Grâce à ce seuil et à un compteur supplémentaire (au cas où $L_s > L$), les L métriques cumulées les plus favorables sont identifiées.

Complexité de calcul observée pour la méthode *List Decoding intégrant la validation du CRC*

La Table 6-14 et la Table 6-15 montrent les charges de calcul observées pour la méthode itérative *List Decoding intégrant la validation du CRC*.

La première table illustre la complexité nécessaire à l'exécution de cette méthode avec L=64 messages. Deux situations sont traitées en détail: la première présente une situation de transmission de données sans erreurs, alors que la seconde considère l'exécution exhaustive de la méthode en raison d'une transmission très mauvaise. Dans cette dernière situation, l'algorithme est forcé (artificiellement) à fournir et à examiner tous les L messages.

La Table 6-15 donne une vue d'ensemble de la complexité de cette méthode itérative. Elle résume les résultats obtenus en utilisant L=16, 32, 64, 128 et 256 messages. Ces résultats mettent en évidence la relation entre la charge de calcul et les conditions de transmission des données. En raison d'un nombre important de messages potentiellement livrables, la position du message valide dans la liste influence la charge de calcul demandée par la méthode.

[36] Les limites inférieure et supérieure des métriques cumulées sont estimées à partir des valeurs de métriques cumulées obtenues au niveau de profondeur précédent. On doit ainsi considérer une différence maximale des métriques plus large (pour les détails, voir Annexe *B*). Chaque compteur d'occurrence surveille une valeur précise de métrique.

Par rapport aux simulations relatives au service de parole *AMR-NB* à 12.2 kbps, l'emploi de méthode itérative avec *L*=64 messages implique une complexité relative[34] qui varie entre 0.39 (absence d'erreurs) et 0.49 (analyse de tous les *L* messages).

	Nombre moyen d'opérations observées							
	Sans erreurs de transmission				**Exécution exhaustive de l'algorithme** (imposé)			
	Format des variables [bits]				Format des variables [bits]			
	32	*16*	*8*	***Total***	*32*	*16*	*8*	***Total***
Additions	0	12703	0	**12703**	0	12974	0	**12974**
Incrémentations	0	37610	0	**37610**	0	44497	0	**44497**
Soustractions	0	98	0	**98**	0	728	0	**728**
Multiplications	1	0	0	**1**	1	0	0	**1**
Divisions	0	0	0	**0**	0	0	0	**0**
Transferts de données	12465	33786	0	**46251**	12528	42081	0	**54609**
Logique	11910	14387	13	**26310**	14143	26231	832	**41206**
Décalages	6	12040	1	**12101**	3714	24129	64	**27907**
Contrôles			85075	**85075**			93731	**93731**
Pointeur: assignations			615	**615**			678	**678**
Pointeur: arithmétique			584	**584**			6821	**6821**
Boucle: initialisations	0	387	0	**387**	0	513	0	**513**
Boucle: contrôles			26636	**26636**			33251	**33251**
Boucle: incrémentations	0	26249	0	**26249**	0	32738	0	**32738**
Grand total				**274621**				**349654**

Table 6-14: complexité de calcul détaillée de la méthode itérative '*List Decoding* intégrant la validation *CRC*' avec *L=64* . Les 500 trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les écarts observés sont inférieurs à 1%. Le fond gris met en évidence les compteurs définis par l'organisations *ITU* et *ETSI*.

Ces résultats montrent que l'objectif établi a été atteint. L'analyse incomplète de l'espace de codage de la méthode *List Decoding* et la nouvelle fonction de sélection des *L* chemins diminuent la complexité de calcul de la méthode itérative *List Decoding intégrant la validation CRC*. Le point crucial est l'opération de sélection des *L* chemins dans la méthode *List Decoding*, qui conditionne la réduction de sa charge de calcul. L'utilisation de fonctions classiques de tri exige en effet une charge de calcul trop importante, ce qui va à l'encontre de l'obtention d'une méthode efficace du point de vue de la complexité. Il a été ainsi nécessaire de développer une fonction de sélection adaptée aux caractéristiques de l'algorithme *List Decoding*.

	Nombre moyen d'opérations observées					
	Viterbi et val. *CRC* (*)	**List Decoding intégrant la validation CRC Exécution exhaustive** (imposé)				
Contexte	Eb/No=0.0dB	*L*=16	*L*=32	*L*=64	*L*=128	*L*=256
Additions	44551	3771	6879	12974	25010	48992
Incrémentations	1320	11908	22695	44497	87853	167660
Soustractions	225	250	409	728	1367	2646
Multiplications	0	1	1	1	1	1
Divisions	0	0	0	0	0	0
Transferts de données	161317	16943	29743	54609	103668	201485
Logique	132694	12322	21993	41206	79386	155227
Décalages	71003	7014	13989	27907	55656	111091
Contrôles	102615	31314	52105	93731	176160	327185
Pointeur: assignations	612	630	646	678	742	870
Pointeur: arithmétique	513	2071	3654	6821	13156	25827
Boucle: initialisations	24269	419	450	513	640	895
Boucle: contrôles	97734	10571	18258	33251	62734	120947
Boucle:	73464	10152	17808	32738	62094	120052
Grand total	**710317**	**107365**	**188631**	**349654**	**668467**	**1282877**
Complexité relative[34]	1.00	0.15	0.27	0.49	0.94	1.81
		List Decoding intégrant la validation CRC Sans erreurs de transmission				
Grand total		**89711**	**151934**	**274621**	**516245**	**974710**
Complexité relative[34]		0.13	0.21	0.39	0.73	1.37

Table 6-15: vue d'ensemble de la complexité de calcul de la méthode itérative basée sur *List Decoding*. Les 500 trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les écarts observés sont inférieurs à 1%. (*)L'opération de validation CRC est illustrée dans l'Annexe *B* (Section *B.1*).

6.8 Comparaisons des méthodes itératives proposées

Le principe à la base de ces décodages itératifs est la livraison d'une série de messages au décodeur extérieur dans le but d'identifier le message correct. La probabilité de la livraison du message correct correspond ainsi à la somme des probabilités des *L* messages livrables.

Le cas idéal est représenté par la livraison des *L* messages globalement les plus prometteurs, ce qui maximise la probabilité de l'obtention du message correct. Cette stratégie est adoptée par la méthode itérative *List Viterbi*

Algorithm ('*LVA*'), méthode qui offre ainsi une qualité de protection supérieure à celle de l'algorithme de Viterbi (Figure 6-25).

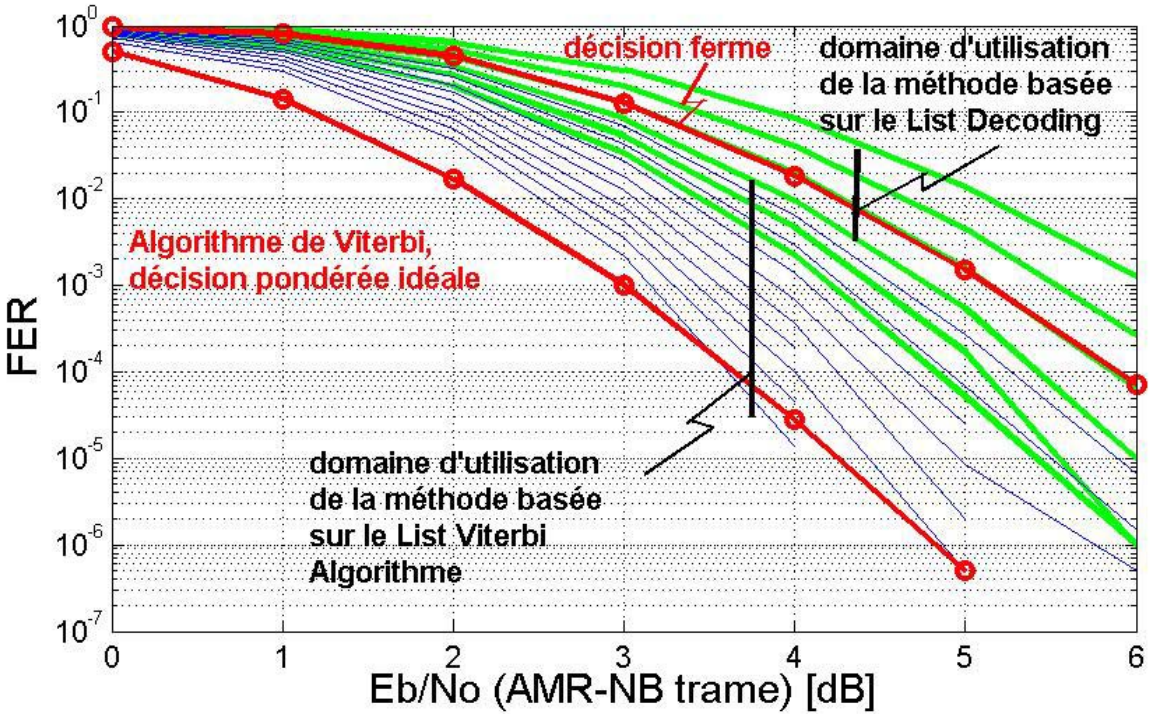

Figure 6-25: vue d'ensemble des potentialités offertes par les deux méthodes itératives. Le graphe vert en trait gras représente la qualité de protection des méthodes itératives *List Decoding intégrant la validation CRC*, le graphe bleu celle des algorithmes *List Viterbi*. Les valeurs de référence de l'algorithme de Viterbi sont indiquées au moyen du graphe rouge en trait gras. Simulation basée sur 2 millions de trames du codeur de parole *AMR-NB* à 12.2 kbps. La figure indique aussi le domaine d'utilisation dans lequel chacune des méthodes fournit une protection de qualité et efficace.

En raison de l'intégration de l'algorithme de Viterbi dans la méthode itérative de décodage, sa complexité de calcul est évidemment supérieure à l'approche classique. Une implantation soigneuse de l'opération de génération des messages (Sous-section 6.6.3) limite l'augmentation de complexité, rendant cette méthode très intéressante du point de vue du rapport entre l'amélioration de la qualité de protection et la complexité de calcul (Figure 6-25 et Table 6-16).

Le principe de livraison d'une série de messages peut être aussi exploité pour développer une méthode de décodage efficace mais moins exigeante en termes de charge de calcul: la méthode *List Decoding intégrant la validation CRC* (Figure 6-25 et Table 6-16). La livraison de plusieurs messages permet

de compenser ainsi la dégradation de la qualité de protection, dégradation qui est causée par un algorithme de décodage convolutif moins exigeant et moins précis que l'algorithme de Viterbi. L'analyse partielle de l'espace de codage et la nouvelle fonction de sélection limitent la charge de calcul demandée par le décodage convolutif *List Decoding*. La perte de qualité de protection est ensuite compensée par la recherche itérative du message valide.

Algorithmes	**Nombre moyen d'opérations observées (exécution exhaustive)**	***WCwOPS*(*)**	
		Pour le traitement d'une trame	**Pour le traitement d'un bit du message (valeur indicative)**
Viterbi, E_b/N_0=0.0dB, sans troncation de la mémoire	710125	616144	6100
Détection du CRC	192	291	4
Méthode itérative basée sur l'algorithme '*List Decoding*'			
L= 32	188631	199919	1979
L= 64	349654	369384	3657
L= 128	668467	705261	6983
Méthode itérative basée sur l'algorithme '*List Viterbi*' (*LVA*)			
Max. 2 messages (*L*=2)	735996	641881	6355
Max. 4 messages (*L*=4)	740003	646035	6396
Max. 8 messages (*L*=8)	748147	654465	6480
Max. 16 messages (*L*=16)	764862	671704	6651
(*) *Worst-Case weighted Operations*: nombre maximal d'opérations impliquées dans l'exécution de la procédure analysée, pondéré selon le modèle proposé par l'*ITU*/*ETSI*. Il comprend aussi les opérations nécessaires à l'adressage (*Array Index*), qui ne sont normalement pas considérées lors de ce type d'évaluation.			

Table 6-16: vue d'ensemble de la complexité de calcul des méthodes les plus représentatives. Les trames utilisées sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps.

La Figure 6-25 marque les champs d'emploi complémentaires des deux méthodes itératives. La méthode *LVA* est envisageable afin d'améliorer la qualité de protection de l'algorithme de Viterbi. En variant le nombre *L* des messages livrables, la méthode atteint ainsi différents gains de codage. Toutefois, un nombre important de messages (*L*>64) réduit l'efficacité de l'implantation de la méthode (Table 6-16), en raison de la stratégie adoptée pour la gestion des chemins les plus probables.
L'emploi envisageable par la méthode *List Decoding intégrant la validation CRC* est la réduction de la charge de calcul de l'opération de décodage, tout en

gardant le même niveau de qualité de protection que celui de l'algorithme de Viterbi. Dans ce cas, il s'agit ainsi de déterminer le nombre *L* de chemins qui offre le rapport désiré entre la qualité de protection et la charge de calcul.

L'utilisation de ces méthodes nécessite (impérativement) une capacité de détection d'erreurs suffisante pour accomplir correctement l'opération d'identification [Kühn97]. Par rapport au cas pratique du service de parole *AMR-NB* à 12.2 kbps, les nombreuses simulations ont démontré la faisabilité et l'efficacité de ces méthodes itératives dans le contexte de codage *UMTS*.

6.9 Conclusions

Ce chapitre a traité le sujet de l'implantation software de méthodes de décodage de données protégées par les codes convolutifs du standard *UMTS* [Ts25212]. L'analyse de la structure de codage de l'*UMTS* a démontré encore une fois l'importance du codage convolutif dans les standards de communication numérique mobile. Les récents standards exploitent massivement ce codage, individuellement, concaténé en série et en parallèle. Cette étude s'est concentrée sur la seconde configuration.

Conformément à la rapidité d'évolution du marché actuel de la communication mobile, l'algorithme de Viterbi est la méthode qui été tout d'abord implantée. L'importante charge de calcul qui en découle (Table 6-17), ainsi que l'importante variabilité du débit de transmission, ont poussé l'étude vers l'exploration des potentialités d'exécution offertes par la structure de protection du standard *UMTS*. En jugeant les méthodes en termes de rapport entre la qualité de protection et la complexité de calcul, deux méthodes ont ainsi été proposées: le *List Viterbi Algorithm* (nouvelle réalisation) et le *List Decoding intégrant la validation CRC* (nouvelle méthode). Ces méthodes itératives se basent sur une exploitation plus exhaustive des informations supplémentaires provenant du codage *CRC*.

La première méthode présentée a été la méthode *List Viterbi Algorithm*. Cette méthode offre une qualité de protection supérieure à celle de l'algorithme de Viterbi (Table 6-17). Une nouvelle implantation de cette méthode, conforme aux critères d'une exécution en temps réel, a confirmé non seulement la faisabilité de la méthode itérative dans le contexte de codage *UMTS*, mais aussi les bons rapports entre qualité de protection et complexité de calcul.

Service de parole *AMR-NB* à 12.2kbps (class *A*)				
		Qualité: gain de codage à *FER*=10^{-3}	**Nombre moyen d'opérations observée: complexité relative[34]**	
			Exécution exhaustive (imposé)	**Sans erreurs de transmission**
Algorithme de Viterbi (référence)		- (E_b/N_0=5.13 dB)	1.00 (710k opérations)	
LVA	*L*=2	+ 0.5 dB	1.04	1.03
	L=8	+ 1.0 dB	1.05	1.03
	L=32	+ 1.5 dB	1.13	1.04
	L=128	+ 1.7 dB	1.48	1.12
List Decoding avec CRC	*L*=32	- 0.4 dB	0.27	0.21
	L=64	0.0 dB	0.49	0.39
	L=128	+ 0.4 dB	0.94	0.73

Table 6-17: résumé des résultats obtenus par les simulations du service de parole *AMR-NB* à 12.2kbps (class *A*) [TrR104], service proposé dans le cadre du standard *UMTS*. Pour les détails, voir les tables: Table 6-8, Table 6-10, Table 6-11, Table 6-13 et Table 6-15.

Une amélioration du rapport qualité/complexité a été ensuite poursuivie en envisageant une réduction de la complexité de calcul de la méthode de décodage. La nouvelle méthode *List Decoding intégrant la validation CRC* a été ainsi développée et proposée. Sa stratégie est la liaison itérative d'une méthode de décodage efficace dans l'analyse partielle de l'espace de décodage (la méthode *List Decoding*) avec la validation du message *CRC*. Cette nouvelle méthode itérative a été possible grâce à une nouvelle réalisation de l'algorithme *List Decoding* qui a permis une réduction importante de sa charge de calcul. Cette réalisation revalorise l'algorithme *List Decoding* aussi dans le contexte d'un codage convolutif direct du message. Les simulations utilisant le codage du service de parole *AMR-NB* à 12.2kbps (classe *B*) confirment la bonne combinaison entre l'analyse partielle de l'espace de codage et la charge de calcul (Table 6-18).

Cette étude a démontré la possibilité pratique d'exploiter de manière itérative les informations supplémentaires provenant de l'enchaînement avec un code en bloc, afin d'améliorer le rapport entre qualité et complexité de l'opération de décodage (Table 6-17). Les simulations exploitant le contexte de codage

du service de parole *AMR-NB* à 12.2kbps (classe *A*) ont confirmé la faisabilité des deux méthodes itératives. Elles ont donné en plus des indications numériques sur les améliorations envisageables (Table 6-17). Par exemple, la méthode *List Viterbi Algorithm* (*LVA*) avec L=8 offre un gain de codage de 1.0 dB, en chargeant le *DSP* d'un nombre d'opérations supérieur de 5% par rapport à l'algorithme de Viterbi. Le réseau *UMTS* peut réduire la puissance d'émission des signaux (de E_b/N_0=5.1 à E_b/N_0=4.1), en garantissant la même qualité de protection des données (FER=10^{-3}). Dans ce même contexte, le remplacement de l'algorithme de Viterbi par la méthode *List Decoding intégrant la validation CRC* permet de réduire la charge de calcul de l'opération de décodage, en gardant inchangée la qualité de la protection. L'utilisation de L=64 messages offre une protection similaire à celle de l'algorithme de Viterbi, mais en réduisant de moitié le nombre d'opérations nécessaires au décodage.

Service de parole *AMR-NB* à 12.2kbps (classe *B*)			
		Qualité: gain de codage à *BER=10^{-3}*	**Nombre moyen d'opérations observée: complexité relative[34]**
Algorithme de Viterbi (référence)		- (E_b/N_0=4.1 dB)	1.00 (788k opérations)
List Decoding	128 chemins	- 0.2 dB	0.73
	64 chemins	- 0.5 dB	0.39
	32 chemins	- 1.0 dB	0.21

Table 6-18: résumé des résultats obtenus par simulations du service de parole *AMR-NB* à 12.2kbps (class *B*) [TrR104].

Enfin, l'un des plus importants résultats de cette étude est la prise de connaissance de la variété des méthodes efficaces de décodage. Ces méthodes offrent une vaste gamme de qualités. Grâce à la souplesse et de la rapidité de développement des implantations software, il est ainsi envisageable de créer une librairie de méthodes de décodage afin de mieux s'adapter aux différents contextes de codage.

Références

[Ande89] J. B. Anderson, "Limited Search Trellis Decoding of Convolutional Codes", *IEEE Transactions on Information Theory*, Vol. 35, No. 5, septembre 1989, pp. 944-955.

[BDTi5] Berkeley Design Technology, Inc (BDTi), *Separating Reality from Hype in Processors' DSP Performance*, présentation, présentée à Embedded Systems Conference (ESC), mars 2002, page Internet accédée au printemps 2002: www.bdti.com

[Cede89] M. Cedervall, R. Johannesson, "Fast Algorithm for Computing Distance Spectrum of Convolutional Codes", *IEEE Transactions on Information Theory*, Vol. 35, No. 6, novembre 1989, pp. 1146-1159.

[ETSI726] ETSI, *Enhanced Full Rate (EFR) Speech Transcoding*, document ETSI ETS 300 726 GSM 06.60, version 5.2.0.

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grande-Bretagne, 2002.

[ITU729] ITU, *basic_op.h*, C-code du standard G.729a, www.itu.int

[Joha99] R. Johannesson, K. S. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Series on Digital and Mobile Communication, Wiley-IEEE Press, Etats-Unis d'Amérique , 1999, chapitres 4-6, pp. 163-315.

[Kühn97] V. Kühn, *Applying List output Viterbi Algorithms to a GSM-based Mobile Cellular Radio System*, présentation à International Conference on Universal Personal Communications ICUPC'97, San Diego, Etats-Unis d'Amérique, 1997.

[Nill95] C. Nill, C.-E. W. Sundberg, "List and Soft Symbol Output Viterbi Algorithms: Extensions and Comparisons", *IEEE Transactions on Communications*, vol. 43, No. 2/3/4, février/mars/avril 1995, pp. 277-287.

[NttD99] NTT DoCoMo, *TSGR1#5(99)689*, TSG-RAN Working Group1, meeting #5, Cheju, Korea, 1-4 Juni 1999.

[Pres92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, Cambridge University Press, 1992.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[Sesh94] N. Seshadri, C.-W. Sundberg, "List Viterbi Decoding Algorithms with Applications", *IEEE Transactions on Communications*, Vol. 42 , No. 2/3/4, février/mars/avril 1994, pp. 313-323.

[Thit93] P. Thitimajshima, *Les codes Convolutifs Récursifs Systématiques et leur application à la concaténation parallèle*, Thèse de Doctorat en Electronique, Université de Bretagne Occidentale, France, 1993.

[TrR104] 3GPP, *Channel Coding and multiplexing examples*, document 3GPP TR R1.04, version 0.0.1.

[Ts22101] 3GPP, *Service principles*, document 3GPP TS 22.101, version 5.4.0.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Ts26071] 3GPP, *AMR Speech Codec: General Description*, document 3GPP TS 26.071, version 4.0.0.

[Ts26101] 3GPP, *AMR Speech Codec Frame Structure*, document 3GPP TS 26.101, version 1.6.0.

[Will99] R. N. Williams, "A Painless Guide to CRC Error Detection Algorithm", version 3, page Internet accédée au printemps 1999: ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt

7 Architectures "hardware" basées sur circuits ASIC

La plupart des architectures utilisées dans le cadre de la communication mobile tirent profit aussi bien du support d'un DSP que de l'utilisation des circuits ASIC. Le sujet de ce chapitre est l'analyse des difficultés pratiques et l'évaluation des potentialités offertes par une implantation ASIC d'une méthode de décodage de codes convolutifs.

Afin de mieux répondre aux exigences actuelles de la communication mobile, la méthode de décodage choisie pour l'étude est l'algorithme de Viterbi. Le système de décodage de base est ainsi décrit en décomposant cet algorithme en ses éléments constituants. Les paramètres, la conception, les variantes possibles et la réalisation matérielle de chaque élément sont traités. Parmi les variantes, on étudie une méthode alternative d'initialisation de l'opération de prise de décision[37] du bit d'information le plus vieux pas encore livré. Cette variante prévoit le décodage d'un chemin quelconque, pour autant qu'il y ait une distance suffisante entre le niveau de profondeur atteint par le chemin et celui du bit d'information. Une stratégie innovatrice pour l'exécution de l'opération de prise de décision est ensuite proposée. Cette stratégie prévoit une réalisation sous forme pipeline, ce qui augmente le débit de traitement du système. Enfin, l'analyse de l'opération de prise de décision présente les avantages d'une stratégie de prise de décision exploitant la redondance des opérations de reconstruction.

Les analyses et les solutions proposées se basent sur les résultats obtenus par la synthèse du système de base en utilisant l'outil de synthèse logique "Design Compiler" de Synopsys (technologie

---

[37] Opération de reconstruction itérative des états de mémoire du codeur convolutif lors du codage, opération qui permet de déterminer la valeur des symboles (bits) formant le message. Cette opération sera aussi nommée simplement 'décodage'.

UMC25-0.25µm). Le chapitre se termine avec un résumé des informations de synthèse sur la vitesse d'exécution et la surface de silicium de chaque élément, ainsi que sur les taux d'activité[38] des signaux liant les éléments constituant le système de décodage.

7.1 Introduction

La situation des technologies de la troisième génération (*3G*) est actuellement caractérisée par la phase de développement des réseaux et des premiers équipements. Les degrés d'implantation de ces technologies *3G* dans les divers pays européens, américains et asiatiques sont variable et dépendent des stratégies nationales ainsi que des économies locales.
Aujourd'hui, le choix d'une architecture optimale pour l'exploitation de ces nouvelles technologies ne peut bénéficier ni d'une expérience acquise, ni de la maturité des standards *3G*. Par conséquent, ce choix ne peut se baser que sur des évaluations et des suppositions, que seule la maturation de ces technologies *3G* pourra successivement soit valider soit réfuter.

7.1.1 Marché actuel des technologies 3G

Quoi que les récentes années aient montré l'augmentation du nombre de tâches assignées au *DSP* avec le perfectionnement des standards *2G*, les autres approches ne doivent pas être exclues du spectre des solutions possibles dans le domaine de la *3G*. On ne dispose pas actuellement d'une approche globale et optimale pour le traitement numérique du signal dans la bande de base.
Le marché et les premiers débats pour le choix d'une architecture optimale montrent une revalorisation de la contribution de circuits *ASIC* au fonctionnement des équipements *3G*. Les informations disponibles sur le développement des premiers réseaux et équipements confirment qu'un nombre élevé de fonctionnalités de la *3G* exploite des circuits spécifiques *ASIC*[39].

---

[38] Par taux d'activité, on entend la fréquence des changements d'état logique d'un signal binaire.

[39] «lors du passage de la 2G à la 3G, le pourcentage de l'exécution des tâches relatives au *Physical Layer* par le *DSP* passe de 100% pour le cas du *GSM* à 10% dans le cas du *WCDMA*» (traduction) [Gath00].

Par rapport au cas du décodage de codes convolutifs, le marché actuel de la communication mobile propose des solutions s'appuyant sur les deux approches software et hardware.
D'un côté, on perçoit un effort des principaux fabricants de *DSP* pour l'amélioration des performances de leurs récentes familles de processeurs, en tenant compte des caractéristiques algorithmiques de la méthode de Viterbi (Figure 7-1 et Table 7-1). L'exécution de cet algorithme peut être facilitée soit par la modification de l'architecture du *DSP*, soit par l'addition de nouveaux blocs fonctionnels. Un exemple est la famille de *DSP* C55x de Texas Instruments qui propose une unité spéciale '*Compare Select and Store*' [Spru312] [Spru393].

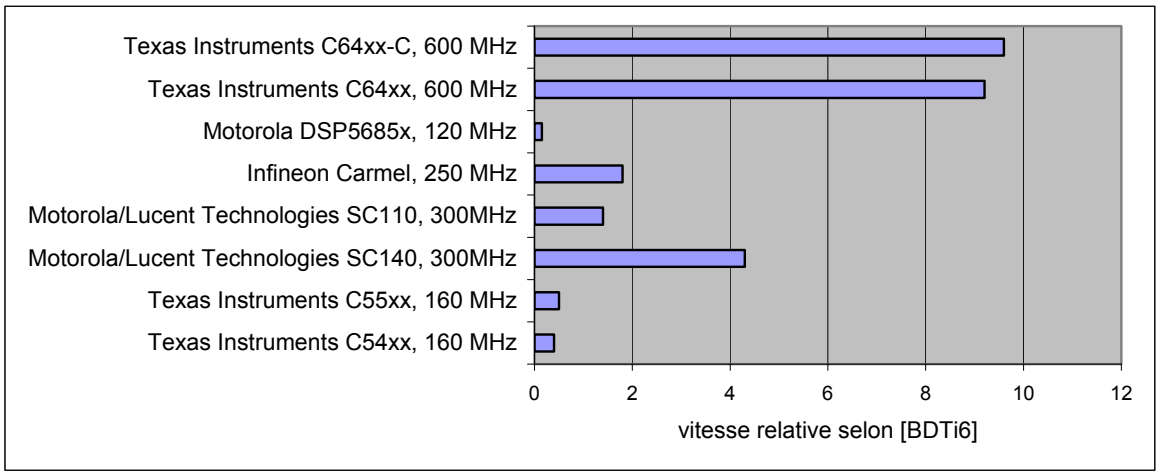

Figure 7-1: exemple d'évaluation de performances des *DSP* (disponibilité 2001) par rapport à l'implantation de l'algorithme de Viterbi. Cette évaluation met en évidence la particularité de cet algorithme qui s'adapte difficilement au caractéristiques 'classiques' des *DSP*. Evaluation effectuée par la société *Berkeley Design Technology* [BDTi3]. Les détails de la méthode d'évaluation utilisés sont disponible dans l'article [BDTi6].

De l'autre coté, le marché offre aussi bien des *ASIC* spécialisés que des *DSP* à haute performance possédant un coprocesseur voué à l'exécution de l'algorithme de Viterbi. A titre d'exemple, on peut nommer l'architecture paramétrable et synthétisable du 'Core*Pool* FHG_VITERBI' de Fraunhofer Institut [FhG01] et le *DSP* TMS320C6416 de Texas Instruments [Spra750] [Spru533].

Cette offre du marché dénote l'augmentation généralisée de la charge de calcul demandée par les technologies *3G* ainsi que par les particularités algorithmiques de la méthode de Viterbi (Figure 7-1 et la Table 7-1).

Fonction	Description	Exemples d'applications
Real Block FIR	Filtre à réponse impulsionnelle finie traitant des blocs de données.	Traitement de signaux de parole.
Single-Sample FIR	Filtre à réponse impulsionnelle finie traitant un échantillon à la fois.	Traitement de signaux de parole, filtrage en général.
Complex Block FIR	Filtre à réponse impulsionnelle finie traitant des blocs de données complexes.	Egalisation d'un canal Modem
LMS Adaptive FIR	Filtre adaptatif *Least-mean-square* traitant un échantillon.	Egalisation de canal, codage prédictif linéaire.
Two-Biquad IIR	Filtre à réponse impulsionnelle infinie traitant un échantillon.	Traitement de signaux audio, filtrage en général.
Vector Dot Product	Produit scalaire	Convolution, corrélation, multiplication matricielle.
Vector Add	Addition de deux vecteurs.	Graphique, combinaison de signaux.
Vector maximation	Recherche de la valeur et de la position de l'élément le plus grand du vecteur.	Protection de données.
Viterbi Decoder	Décodage d'un signal encodé par un code convolutif.	Dans le domaine de la communication.
Control	Série d'opérations (artificielles) concernant le contrôle et la manipulation des bits.	Toutes les applications *DSP* impliquent ces types d'opération.
256-Point FFT	Transformation de Fourier (*Fast Fourier Transformation*).	Radar, sonar, compression audio de type *MPEG*, analyse spectrale.
Bit Unpack	Extraction de mots (de tailles différentes) à partir d'une séquence continue de données.	Décompression de signaux audio et de parole.

Table 7-1: ensemble de fonctions utilisées pour l'évaluation des performances de *DSP*, utilisé par la société *Berkeley Design Technology* (*BDTI*) [BDTi1].

7.1.2 Exigences des technologies UMTS

Les standards de communication mobile *3G* sont le résultat de l'effort d'un grand nombre de sociétés mondiales, effort qui a commencé au milieu des années 90 [Gath02].
Selon la stratégie adoptée lors de leur développement, les systèmes appartenant à cette technologie *3G* sont prévus pour supporter une large variété de services, en offrant un débit de transmission (*Data Rate*) fortement

variable. Ce débit peut atteindre 144 kbps considérant le contexte d'une utilisation dans des véhicules (*Vehicular Outdoor Environment*), 384 kbps pour une utilisation pédestre (*Pedestrian Outdoor Environment*), et jusqu'à 2 Mbps à l'intérieur des bâtiments (*Indoor Environment*). Comme introduit dans la Section 3.1, l'interface radio a été prévue selon un protocole à trois couches:

- Le *Physical Layer* (ou *Layer 1*). Il est responsable de la transmission sans fil des données.
- Le *Data link layer* (ou *Layer 2*). Sa tâche est la détermination des caractéristiques des données en transmission, la gestion du flux de ces données et la qualité du service. Le passage d'informations entre ce niveau et le premier est géré par le *Medium Access Layer* (*MAC*).
- Le *Network layer* (ou *Layer 3*). Cette dernière couche gère les échanges entre l'équipement portable et le réseau *UTRAN* (*UMTS Terrestrial Radio Access Network*). Le *Radio Ressource Controller* (*RRC*) est l'élément de contact avec la première couche du protocole.

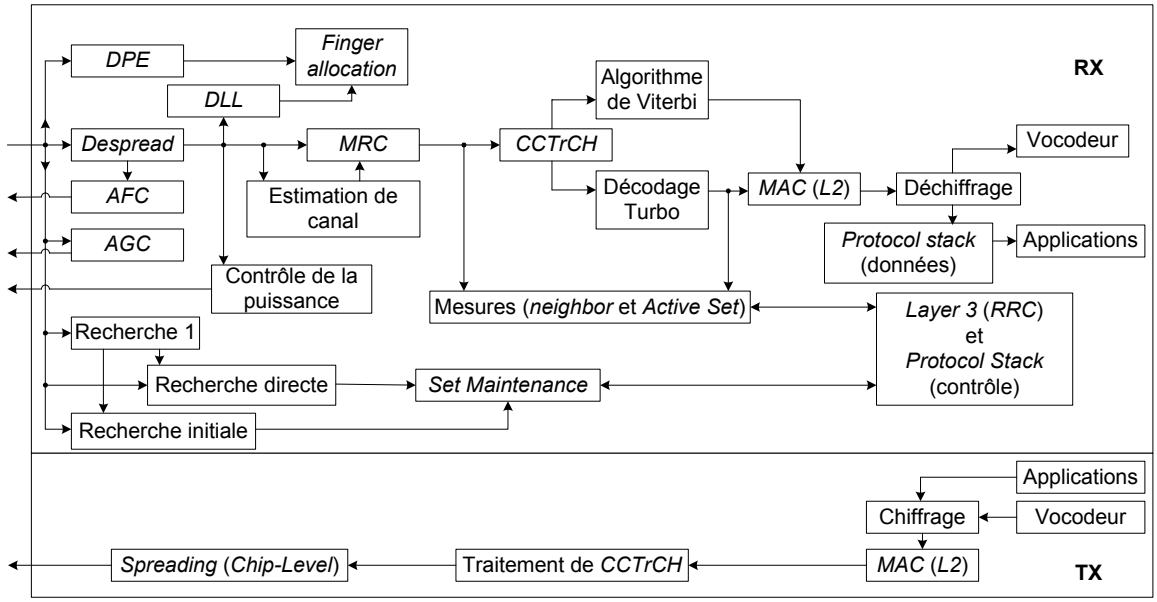

Figure 7-2: vue d'ensemble des composants de la couche physique (*Physical Layer*) réalisant le traitement digital du signal dans la bande de base [Gath02].

La couche qui nous intéresse est la première. En ne considérant que cette couche, la Figure 7-2 donne la vue d'ensemble de ses composants qui accomplissent le traitement numérique du signal dans la bande de base.
Du point de vue de la complexité de calcul, cette première couche est la plus exigeante en termes de ressources hardware et software. Parmi ses

composants les plus exigeants, on trouve [Gath02]: le traitement des données appartenant aux différents canaux de transport (*CCTrCH Processing*), le décodage de canal, la procédure de recherche de la cellule, l'opération de *Despreading*, le *Maximal Ratio Combination* (*MRC*)[40] et le *Multipath Search* (ou *Delay Profile Estimation*, *DPE*)[41] (Figure 7-3).

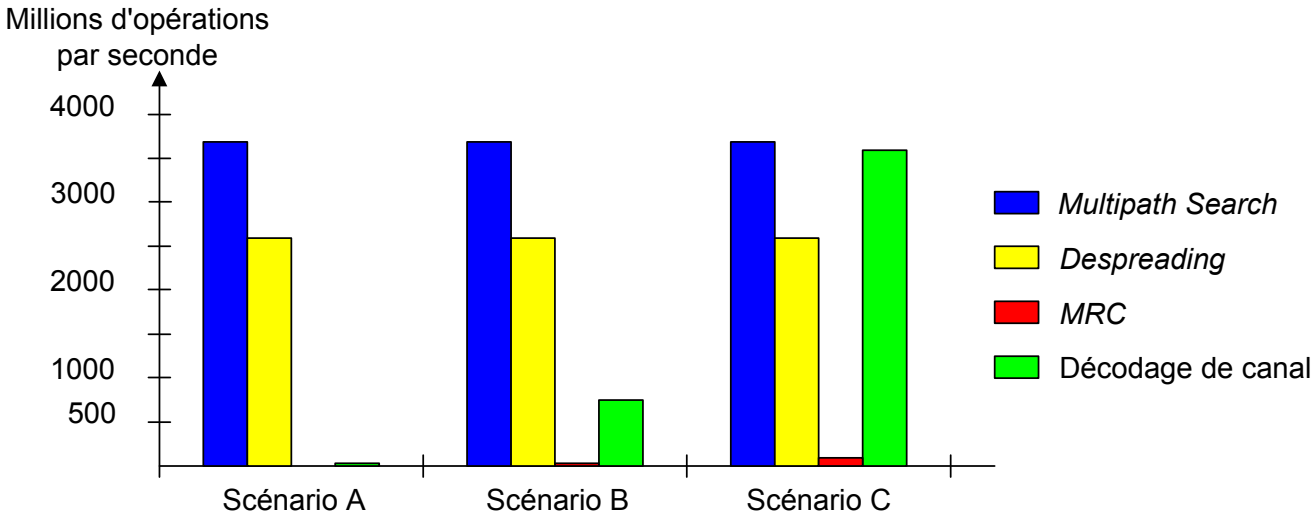

Figure 7-3: estimation des ressources de calcul demandées par les tâches les plus exigeantes, selon trois scénarios représentatifs [Gath02]. Scénario A: service de parole à 8 kbps. Scénario B: service de parole à 12,2 kbps et service vidéo à 384 kbps. Scénario C: service générique à 2 Mbps.

L'exemple de la Figure 7-3 montre que la puissance de calcul nécessaire à l'exécution de certaines tâches peut changer de manière importante, non seulement en fonction du débit de transmission, mais aussi selon le nombre de services offerts, les caractéristiques de la transmission sans fil, ou encore le nombre de cellules dans le voisinage [Gath02]. Parmi ces tâches, on trouve le décodage de canal.

[40] Procédure de combinaison de signaux reçus de manière à augmenter la protection contre les distorsions de type *fading*.

[41] La recherche du signal, qui s'approche le plus du signal le plus fort, afin d'exécuter le *maximal ratio combination*.

Répartition des modules des terminaux *3G* entre implantation software et réalisation hardware

Le développement des terminaux mobiles *3G* est actuellement caractérisé par un besoin évident de souplesse, par une augmentation importante de la puissance de calcul demandée et par la variété des services envisagés par la *3G*.

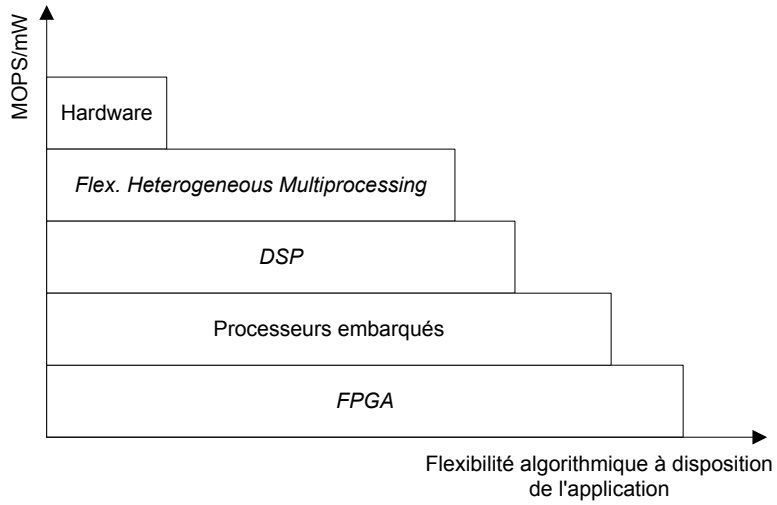

Figure 7-4: compromis entre la consommation d'énergie et la flexibilité de la réalisation [BDTi4]

En raison de la forte demande en puissance de calcul, une répartition des modules des terminaux entre implantations software et réalisations matérielles est généralement nécessaire.
En général, le choix de l'approche suit les règles suivantes (Figure 7-4):

- Si le module exige un nombre d'opérations très élevé ou si les caractéristiques du traitement s'adaptent difficilement aux ressources mises à disposition par le *DSP*, une approche hardware est en général une solution optimale.
- Si la nécessité de souplesse est prédominante, la meilleure approche est l'implantation software.
- Dans certain cas, le choix du type d'approche est déterminé par des critères de basse consommation d'énergie et/ou par la particularité du service concerné [Gath02].

La Figure 7-5 montre une répartition possible des divers modules du *Physical Layer* [Gath02], en suivant les critères ci-dessus.

Par rapport au décodage de codes convolutifs, cette opération demande une charge importante de calcul par bit d'information: l'implantation paramétrable de l'algorithme de Viterbi présente une complexité de calcul de $30 \cdot 2^{K-1}$ wOP par bit d'information (Chapitre 6)[42]. Cette charge doit être ensuite pondérée par le débit de transmission de l'application concernée, débit qui peut varier de quelques dizaines jusqu'à plusieurs milliers de kbps.
L'importante variabilité en termes de puissance de calcul qui en découle ne permet pas d'établir à priori une stratégie générale et optimale dans ce contexte *UMTS* (Figure 7-3). Par conséquent, le spectre des solutions possibles doit considérer les deux approches: software et hardware.

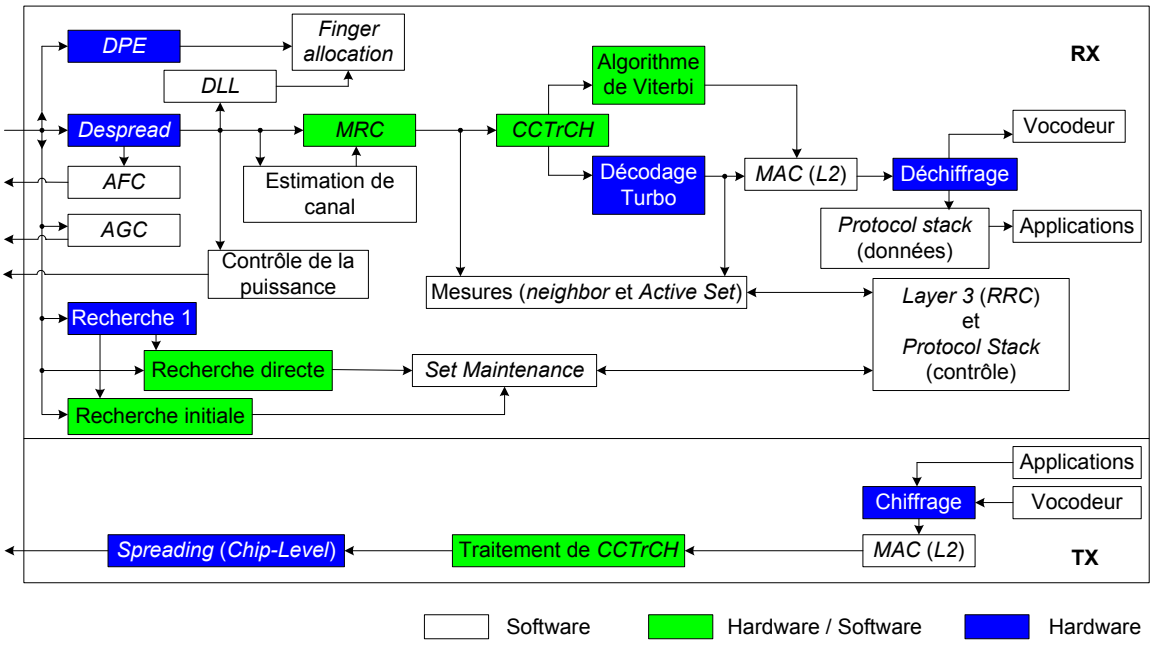

Figure 7-5: répartition possible des composants de la couche physique pour le traitement digital du signal dans la bande de base [Gath02].

7.2 Considérations sur l'implantation matérielle

7.2.1 Définition du système dans lequel le décodeur travaille

La configuration de base reste identique à celle considérée lors de l'implantation software de méthodes de décodage. Les symboles reçus sont soumis à une décision ferme et l'étude considère la simple livraison du

[42] La perte d'efficacité due à cette description souple et paramétrable sera ensuite traitée dans le Chapitre 8.

message sans informations supplémentaires (*Bit True*). Cette configuration envisage un système de décodage le plus général possible, en facilitant la reproduction et la comparaison des résultats.
Ainsi, en contradiction partielle avec le contexte de codage établi dans le standard *UMTS* [Ts25212], ce chapitre abordera la situation suivante:

- le décodage de séquences infinies de symboles, et
- l'absence de procédures d'adaptation du débit de transmission (par répétitions et poinçonnages de symboles).

L'adaptation de ce système aux standards de télécommunications *UMTS* sera discutée dans l'Annexe *C* (Section *C.1*).

7.2.2 Description HDL du décodeur

La contrainte principale de la description *HDL* (*Hardware Description Language*) est une description compréhensible et facile à modifier, afin de permettre une adaptation rapide aux divers standards de communications.
La conception de cette procédure est réalisé selon les notions de modularité et de paramétrage, permettant ainsi une rapide modification et substitution de tous les blocs de traitement.

Par rapport au type d'architecture, on envisage la conception d'un système autonome de base, tout en permettant la conversion rapide dans une solution coprocesseur (Annexe *C*, Section *C.2*). Par conséquent, la conception de ce système de base s'oriente vers une architecture distribuée (*Distributed Architectures* [Gath02]).
Cette architecture prévoit une autonomie et une indépendance de chaque module, en diminuant les ressources partagées. Du point de vue de la consommation d'énergie, cette stratégie permet une gestion optimale du mode d'économie d'énergie et une fréquence d'horloge adaptée à chaque module.

La description du système de base est validée au moyen de quatre codes convolutifs et par l'implantation du code sur *FPGA* (*Field Programmable Gate Array*, [Gras01]). Les caractéristiques de ces codes convolutifs sont illustrées dans la Table 7-2.

Codeur	Longueur de contrainte	Nombre d'états différents	Polynômes générateurs [octal]			Distance libre d_{free}	Exemple d'utilisation dans le contexte des télécommunications
			G_0	G_1	G_2		
$Codeur_{4états}$	3	4	5	7	7	8	-
$Codeur_{16états}$	5	16	25	33	37	12	*GSM 6.0 kbps Data TC* [Etsi909]
$Codeur_{64états}$	7	64	133	145	175	15	*GSM Half Rate Speech TC* [Etsi909]
$Codeur_{256états}$	9	256	557	663	711	18	*UMTS Channel Coding* [Ts25212]

Table 7-2: caractéristiques des codes convolutifs, avec rendement R_c=1/3, utilisés pour l'évaluation et la validation de la conception du décodeur. Les acronymes indiqués dans la première colonne du tableau seront utilisés dans la suite de ce document pour identifier le codeur convolutif utilisé.

7.2.3 Méthode de décodage: l'algorithme de Viterbi

Contrairement au cas précédent d'une implantation software, une seule méthode de décodage est considérée pour l'évaluation des potentialités de l'approche hardware: l'algorithme de Viterbi.

Les raisons de ce choix sont liées aux caractéristiques de cette méthode de décodage qui facilitent l'approche hardware, notamment par sa structure algorithmique parallèle et très régulière ainsi que par son indépendance algorithmique du flux des données. Ce dernier aspect est important afin de réduire les risques de modifications dues à la maturation des standards et à l'expérience acquise avec les technologies *3G*.
Afin de permettre le décodage d'une séquence continue de symboles, l'algorithme de Viterbi doit s'appuyer sur une procédure de troncation de la mémoire. Bien que cette procédure emploie un paramètre réglant le fonctionnement de la méthode (la distance δ), la longue expérience acquise permet de réduire au minimum les coûts de réglage de ce paramètre.
Enfin, sur le plan de qualité de protection, cet algorithme est retenu communément comme la méthode de référence en raison de sa bonne protection contre les erreurs.

7.3 Système de base

7.3.1 Vue d'ensemble

Conformément aux directives d'une décomposition de l'algorithme de Viterbi en ses éléments constituants, le système est tout d'abord partagé en deux grandes unités fonctionnelles (Figure 7-6).

La première unité est responsable de la sélection des chemins par mise à jour des valeurs des métriques cumulées. Cette unité est divisée en deux modules. Le premier module (indiqué par l'acronyme *BMu*) est chargé du calcul des métriques de branches, en considérant les symboles reçus r_{np}. Ces métriques sont utilisées par le second module (*ACSu*) qui effectue la mise à jour des valeurs des métriques cumulées. Par comparaison de ces valeurs, les chemins survivants sont ensuite sélectionnés. Les informations sur les chemins survivants ainsi que leurs métriques cumulées sont mises à disposition des autres modules.

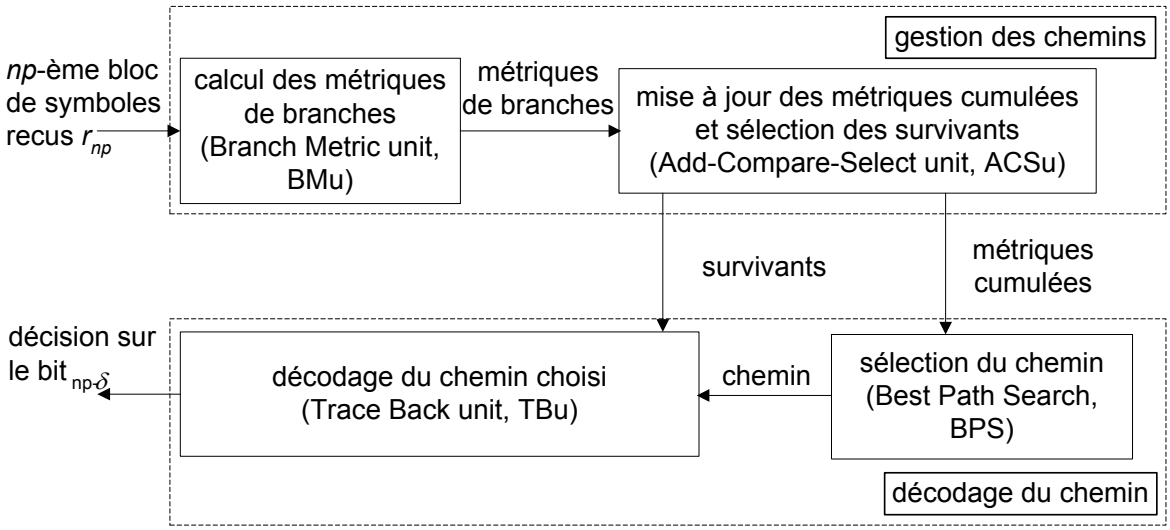

Figure 7-6: architecture du système de base décrit en *HDL*.

La tâche de la seconde unité est la prise de décision du plus vieux bit, qui n'a pas encore été délivré. Cette procédure implique le choix d'un chemin, dont le décodage permet une prise de décision correcte. Le module nommé *BPS* s'occupe du choix du chemin, alors que la prise de décision par décodage du chemin est exécutée par le module *TBu*.

Les paramètres utilisés dans la description *HDL* du système de base peuvent être répartis en deux classes: ceux qui décrivent les caractéristiques du codeur

convolutif et ceux qui règlent la conception du système. Dans la première classe, on trouve les paramètres décrivant les fonctions génératrices G_i, la longueur de contrainte du code K et le nombre d'états différents que la mémoire du codeur peut atteindre. La distance δ de troncation de la mémoire, le nombre de bits utilisés pour les représentations des métriques sont les paramètres de la seconde classe.

7.3.2 Synchronisation et répartition entre blocs combinatoires et non-combinatoires

Après la présentation de l'architecture du système de base, cette section traite les aspects de synchronisation et de répartition entre blocs combinatoires et non-combinatoires (Figure 7-7).

Selon le flux des données, le module *BMu* est abordé en premier. Ce module, qui est responsable du calcul des métriques de branches, n'est implanté que par de la logique purement combinatoire, sans l'utilisation d'aucune mémoire tampon (*Buffer*) à l'entrée.

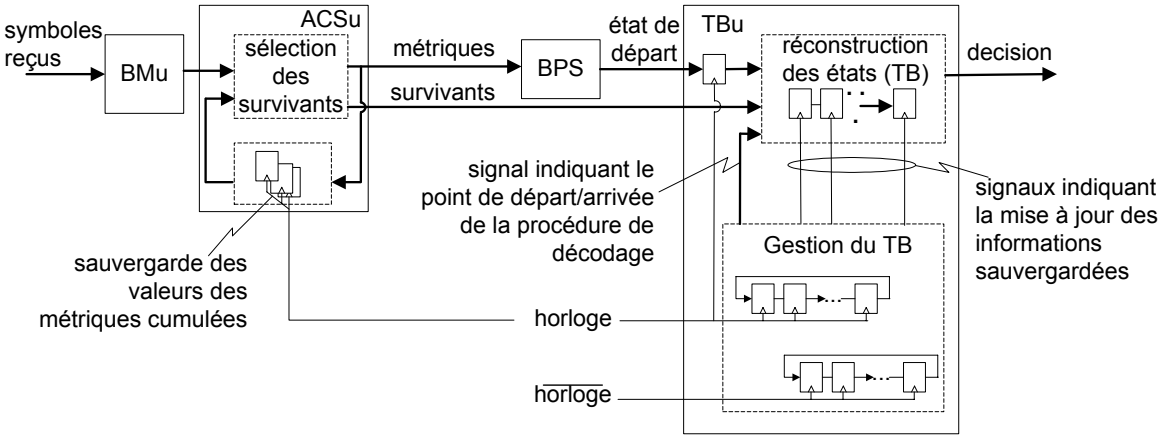

Figure 7-7: gestion de la synchronisation relative à la réalisation physique du système.

La réalisation physique du module suivant (le *ACSu*) est caractérisée par deux parties fonctionnelles: l'une est responsable de la sauvegarde des dernières valeurs des métriques cumulées et l'autre de la sélection des chemins survivants par la mise à jour de leurs métriques cumulées.
L'opération de sélection utilise les valeurs de métriques fournies par le module précédent et celles sauvegardées dans le module même. A partir de ces valeurs, les chemins survivants sont sélectionnés par de la logique combinatoire. En raison du stockage des valeurs de métrique, cette opération de sélection est synchronisée par le signal d'horloge du système.

Parallèlement, les informations des chemins survivants ainsi que leurs métriques cumulées sont mises à disposition des modules *BPS* et *TBu*.

En suivant le flux principal de données, le prochain module est le *BPS*. Sa tâche est la détermination du chemin le plus conforme à une prise de décision correcte: cette opération implique uniquement de la logique combinatoire.

Le dernier module est le module *TBu* qui est chargé de l'opération de prise de décision du bit d'information. Cette opération implique tout d'abord le stockage et la gestion des informations permettant la reconstruction des chemins survivants. Ce module reçoit régulièrement les nouvelles informations au sujet des chemins survivants, qui, selon une stratégie de mémoire circulaire, remplacent les anciennes informations devenues désormais inutiles. A partir du chemin indiqué par le module *BPS*, ces informations sont ensuite utilisées pour reconstruire de manière itérative les états antécédents de mémoire du codeur, jusqu'à extraire la valeur du (np-δ)-ème bit d'information. Les interférences entre les deux tâches du module sont évitées en synchronisant l'opération de sauvegarde des informations et le début de la procédure de reconstruction du chemin.
La réalisation physique de ce module *TBu* utilise aussi bien des blocs combinatoires que non-combinatoires. Ces derniers sont utilisés pour la sauvegarde des informations et pour la gestion des signaux qui indiquent le point de départ d'et arrivée de la procédure de reconstruction des chemins.

Par rapport aux contraintes sur le signal d'entrée (Figure 7-7), le signal des symboles reçus doit rester stable durant une période suffisante pour assurer:

- le stockage correct des nouvelles métriques cumulées dans le module *ACSu*;
- la sélection du chemin le plus favorable pour la prise de décision, chemin qui est mémorisé dans le module *TBu*.

7.3.3 Calcul des métriques des branches

La tâche de ce module est le calcul de toutes les métriques de branches qui seront ensuite utilisées pour la mise à jour des métriques cumulées lors de la procédure de sélection des chemins survivants (Figure 7-8).

Selon le contexte de codage binaire (b=1), la procédure de sélection de l'algorithme de Viterbi nécessite les valeurs de métriques de branches des

couples de chemins convergeant en chaque état du treillis (Figure 7-9): ce module est chargé de la livraison de ces couples de métriques.

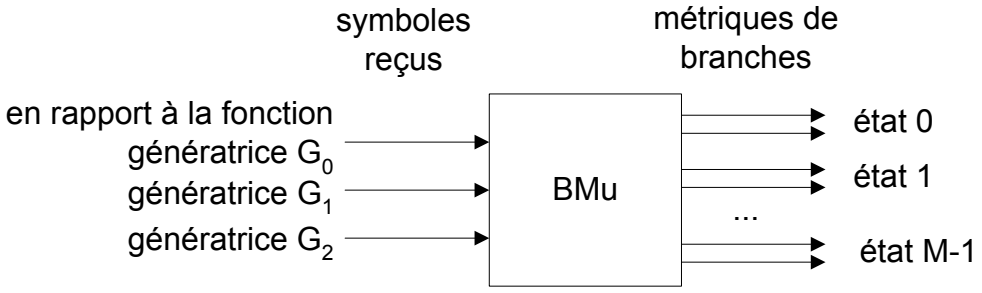

Figure 7-8: calcul des métriques de branches.

Les valeurs des métriques de branches coïncident avec la distance de Hamming existant entre les symboles reçus et ceux générées par les transitions d'états. En présence de codeurs convolutifs avec rendement R_c=1/3, la représentation numérique de ces métriques demande 2 bits.

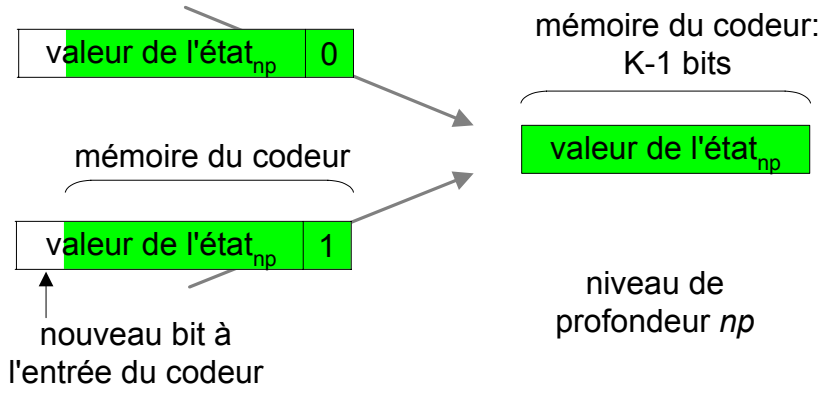

Figure 7-9: représentation graphique des deux transitions qui mènent au même état de mémoire du codeur convolutif.

En raison d'une approche paramétrable, les $2 \cdot M = 2 \cdot 2^{K-1}$ métriques de branches sont décrites à l'aide des 3 fonctions génératrices G_i et de l'état de mémoire du codeur. L'opération d'optimisation des calculs est déléguée aux compilateurs de synthèse.

La conception de ce module dépend des paramètres décrivant le codeur convolutif, notamment le rendement R_c et les matrices génératrices G_i. En raison de cette forte dépendance aux changements de contexte de codage, les approches coprocesseurs (Annexe *C*, Section *C.2*) délèguent la tâche de ce module aux processeurs.

7.3.4 Sélection des chemins survivants

Le module *ACSu* est chargé de la sélection des nouveaux chemins survivants à l'aide des récentes valeurs de métriques cumulées (Figure 7-10). Les informations de parcours des chemins survivants et les valeurs de leurs métriques cumulées doivent être ensuite mises à disposition des autres modules, de manière à permettre la reconstruction des chemins les plus intéressants.

Répartition du module en deux parties

En raison de sa tâche, ce module est composé de deux parties fonctionnelles: la première responsable du stockage des métriques cumulées et la seconde de la sélection des 2^{K-1} chemins survivants (Figure 7-11).

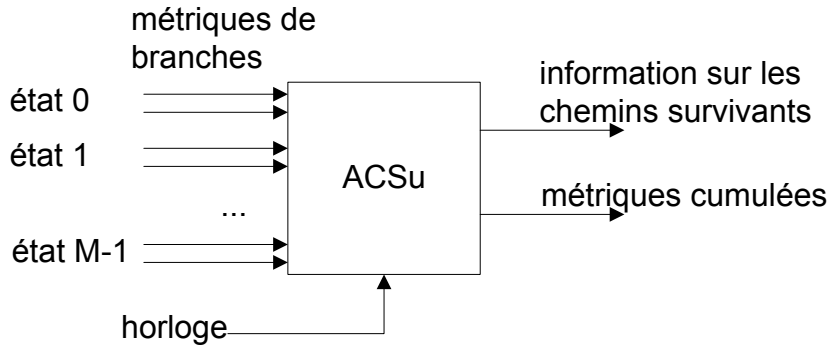

Figure 7-10: sélection des chemins survivants.

Les deux paramètres qui agissent sur ce module sont:

- Le nombre de bits utilisé pour la représentation des valeurs des métriques de branches. Ce paramètre influence la représentation numérique des métriques cumulées.
- Le nombre d'états différents *M* que la mémoire du codeur peut montrer. Ce paramètre concerne la description du codeur convolutif et sa valeur dépend de la longueur de contrainte du code ($M= 2^{K-1}$).

La conception du module est ainsi moins sensible aux modifications du contexte de codage. Par exemple, en considérant la structure de codage de l'*UMTS* [Ts25212], la conception reste inchangé, indépendamment de l'utilisation d'un des deux codes convolutifs à disposition.

Représentation numérique des métriques cumulées

La stratégie du traitement des métriques cumulées doit considérer les aspects suivants:

- L'opération de sélection des chemins survivants nécessite la représentation correcte des différences entre les valeurs des métriques cumulées.
- La sélection du chemin le plus prometteur (pour la prise de décision) est communément basée sur l'identification de la métrique cumulée la plus favorable.
- En raison d'une métrique accumulant le nombre d'erreurs rencontrées par les chemins, la transmission erronée des données implique la croissance des valeurs des métriques cumulées.
- Le décodage de séquences infinies de symboles ne permet pas de fixer une valeur limite aux métriques cumulées.
- La représentation numérique des métriques cumulées influence la conception, les performances et les caractéristiques de tous les blocs, qui sont responsables des opérations de mise à jour, de comparaison et de stockage des métriques cumulées.

En considérant ces aspects, la stratégie choisie envisage une représentation minimaliste des métriques cumulées avec la contribution d'une fonction de normalisation des valeurs de métriques. L'objectif est de garantir la relation correcte entre les valeurs des métriques cumulées, tout en minimisant les ressources nécessaires aux opérations d'addition, de comparaison et de stockage de ces métriques.

Première étape: la détermination du nombre minimal de bits

L'établissement de la représentation numérique des métriques cumulées doit tout d'abord prendre en considération le nombre minimal de bits nécessaires à la description correcte des écarts entre les métriques.

Comme déjà montré dans le chapitre précédent, la différence maximale entre les valeurs des métriques cumulées (au même niveau de profondeur) est

$$différence_{\max} = MaxContribBranche \cdot \left(K \text{-} 1 \right). \qquad (7.1)$$

En considérant que l'opération de normalisation suit les opérations de mise à jour des métriques cumulées et de sélection du chemin survivant (Figure 7-11), l'écart (7.1) est ainsi élargi

$$\begin{aligned} dyn_{\max} &= différence_{\max} + MaxContribBranche \\ &= MaxContribBranche \cdot K, \end{aligned} \tag{7.2}$$

afin d'assurer l'exécution correcte des opérations de sélection. Par conséquent, le nombre minimal de bits nécessaire à la représentation correcte de l'écart est

$$bitsDynamique_{\min} = \lceil \log_2 (dyn_{\max} + 1) \rceil \tag{7.3}$$

Seconde étape: la procédure de normalisation

La stratégie adoptée vise à garantir une représentation correcte des différences entre les métriques à l'aide d'une fonction de normalisation (Figure 7-11).

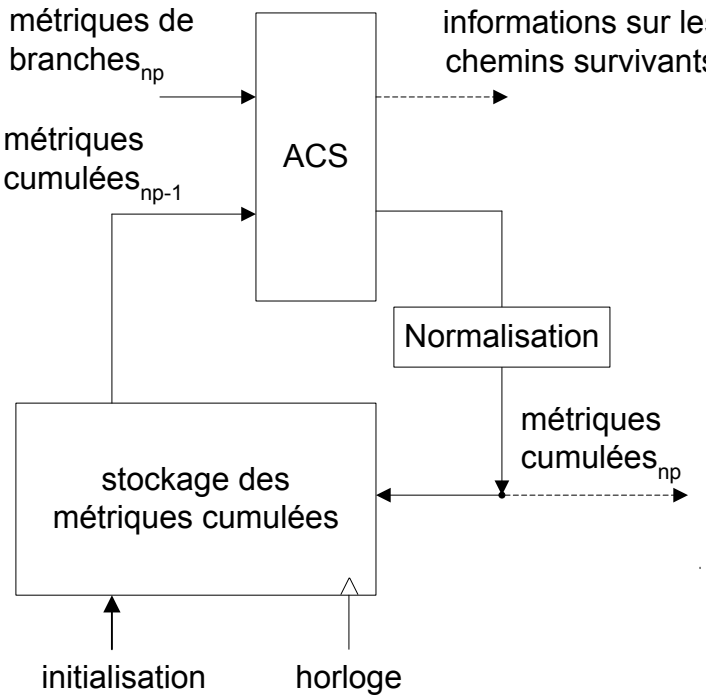

Figure 7-11: structure du module *ACSu* montrant la stratégie choisie pour la représentation numérique des métriques cumulées.

Le principe classique de normalisation prévoit la soustraction d'une valeur déterminée à priori, lorsque toutes les métriques cumulées atteignent ou dépassent un seuil fixé [Min91]. La valeur soustraite peut être soit égale (cas idéal), soit inférieure à la métrique cumulée la plus basse, afin d'éviter le dépassement de la valeur minimale représentable (*Underflow*).

Bien que la détermination du seuil de normalisation ne soit soumise à aucune contrainte, elle influence inévitablement le nombre de bits nécessaires à la représentation des métriques cumulées ainsi que la conception de l'opération de normalisation.
Évidemment, la représentation des variables doit permettre le dépassement correct du seuil par toutes les métriques cumulées, sans générer de situations de dépassement de la valeur maximale représentable (*Overflow*): la réprésentation numérique doit ainsi couvrir l'espace

$$\left[0... \text{ seuil} + dyn_{\max} - 1\right]. \tag{7.4}$$

La valeur du seuil influence aussi la réalisation physique de la tâche de surveillance de son dépassement. La solution idéale est l'utilisation d'un seuil de métrique dont la représentation numérique coïncide avec une puissance de *2*. Dans ce cas, le contrôle de l'égalisation/dépassement du seuil se concentre sur un seul bit de la représentation numérique des métriques cumulées.
Un autre avantage qui en découle est la simplification de l'opération de soustraction: en fixant la valeur de normalisation égale au seuil, l'opération se réduit à la simple remise à zéro du bit "surveillé" (Figure 7-12).

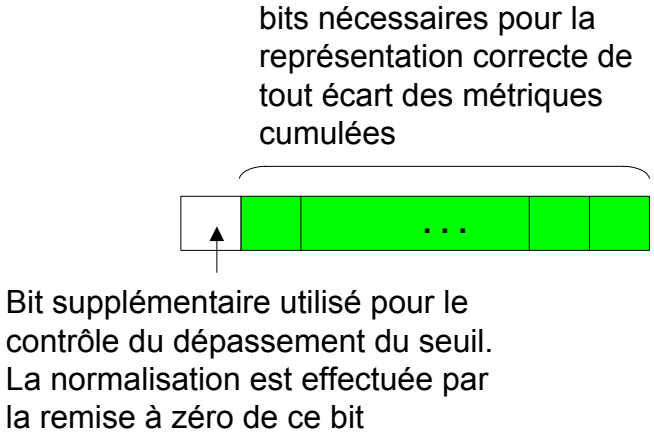

Figure 7-12: stratégie adoptée pour la représentation numérique des métriques cumulées.

Solution adoptée

La stratégie choisie emploie une représentation numérique utilisant ($bitsDynamique_{min}$ *+1*) bits (Table 7-3). Ce bit supplémentaire représente pratiquement la valeur du seuil de normalisation ainsi que la valeur de soustraction (Figure 7-12), notamment la valeur de

$$2^{bitsDynamique_{\min}}. \qquad (7.5)$$

Ce choix permet de limiter le nombre de bits de la représentation numérique des métriques cumulées et de simplifier la réalisation physique de l'opération de normalisation (contrôle et soustraction).

Initialisation des métriques cumulées

Après avoir défini la représentation numérique des valeurs des métriques cumulées, on aborde le sujet de leur initialisation.

L'exploitation de l'état de départ de la mémoire du codeur peut se réaliser par l'initialisation différenciée des métriques cumulées. L'objectif poursuivi est de rendre défavorables les chemins démarrant d'autres états de départ. De manière analogue à l'implantation software (Sous-section 6.4.1), on peut ainsi utiliser une valeur d'initialisation entre

$$MaxContribBranche \cdot (K-1)$$

et

$$ValMaxMetrique - \left[MaxContribBranche \cdot (K-1) \right].$$

Comme déjà expliqué dans le chapitre précédent, cette initialisation permet d'exploiter les informations de départ du codage convolutif sans exiger aucune procédure supplémentaire de contrôle.

Codeur	Longueur de contrainte K	Métriques de branches (R_c=1/3)		Taille de la représentation des métriques cumulées [bits]
		Taille de la représentation [bits]	Valeur maximale [distance de Hamming]	
$Codeur_{4états}$	3	2	3	5
$Codeur_{16états}$	5	2	3	5
$Codeur_{64états}$	7	2	3	6
$Codeur_{256états}$	9	2	3	6

Table 7-3: vue d'ensemble du nombre de bits utilisés pour la représentation des métriques cumulées dans les divers contextes de codage.

Sélection des 2^{K-1} chemins survivants

La tâche de ce module est la sélection des $M=2^{K-1}$ chemins survivants par comparaison des métriques cumulées des couples de chemins qui convergent en un même état du treillis (Figure 7-13). La structure algorithmique de la procédure de sélection des M chemins survivants s'avère ainsi très parallèle et régulière, ce qui favorise une réalisation matérielle.

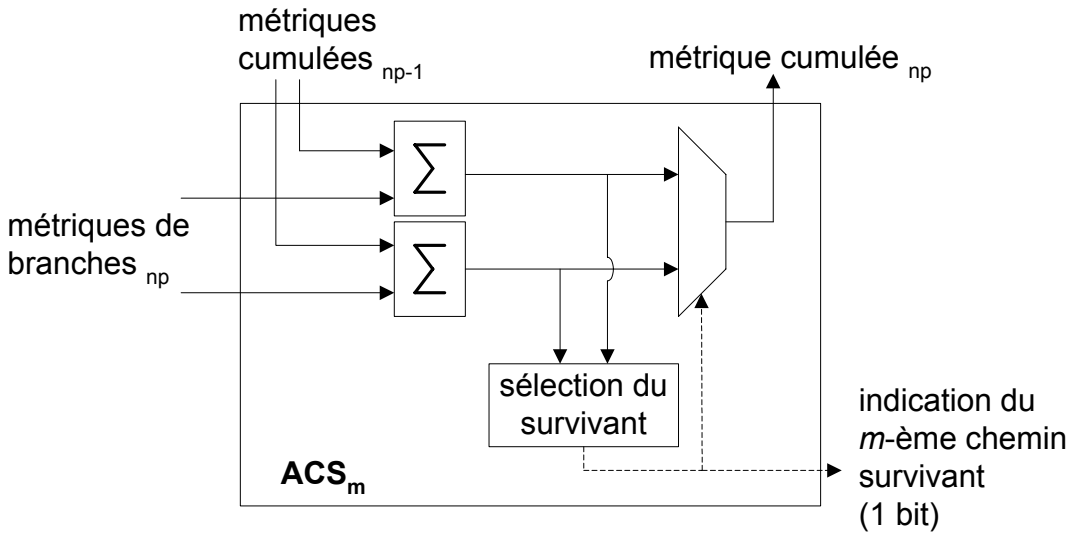

Figure 7-13: description graphique de l'opération de sélection du m-ème chemin survivant.

Contrairement aux implantations software, une réalisation matérielle permet d'exploiter exhaustivement le parallélisme algorithmique de la tâche de sélection. Généralement, si l'exécution séquentielle offre tous les avantages d'une surface réduite [Kapp92], la stratégie parallèle permet l'exploitation optimale de la structure algorithmique de l'opération de sélection, améliorant le débit du traitement.
Si on considère une réalisation matérielle de l'algorithme original de Viterbi (la prise de décision n'est effectuée qu'à la fin du traitement de tous les symboles reçus), le temps d'exécution de cette opération de sélection des M survivants détermine le débit du système de décodage. Pour cette raison, la réalisation de cette unité a fait l'objet de nombreuses études [Fett90] [Min91] [Kapp92].

Afin de prendre connaissance des potentialités d'une réalisation matérielle, cette étude prend en compte la stratégie d'une exploitation exhaustive du parallélisme disponible (Figure 7-14).

Par rapport à la conception de l'opération de sélection d'un chemin survivant (Figure 7-13), l'opération de mise à jour des métriques cumulées, la comparaison et la sélection du meilleur chemin sont exécutées de manière séquentielle.

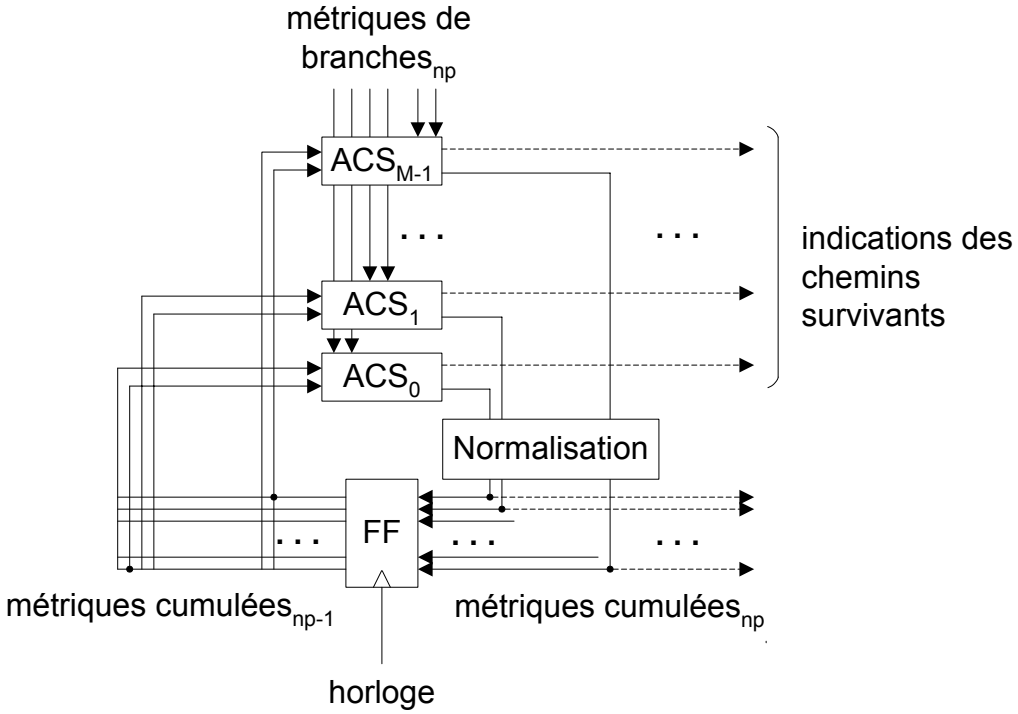

Figure 7-14: structure parallèle du module responsable de la sélection des chemins survivants.

7.3.5 Prise de décision

La seconde grande unité du système de décodage (Figure 7-6) se charge de la prise de décision et de la livraison du bit le plus ancien, pas encore remis au destinataire.
La stratégie communément exploitée est la troncation de la mémoire (Sous-section 6.4.2), qui prévoit la sélection du chemin à la profondeur actuelle *np*, dont le décodage permet une prise de décision correcte du (*np*-δ)-ème bit d'information (Figure 7-15). Cette stratégie est indispensable pour effectuer une prise de décision dans un contexte de codage qui envisage la protection d'une séquence infinie de bits d'information.

Décodage du chemin: la reconstruction des états de la mémoire du codeur

A l'aide du décodage du chemin indiqué par le module précédent *BPS*, le module *TBu* se charge de la prise de décision du bit le plus "vieux", pas encore livré (Figure 7-15).

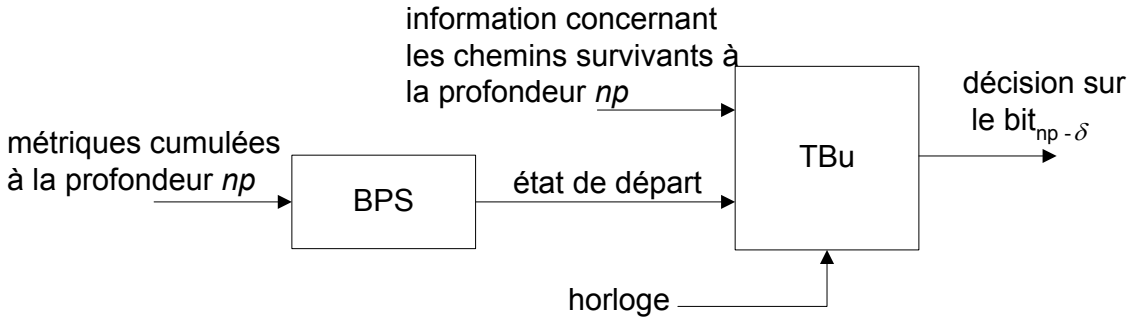

Figure 7-15: opération de prise de décision sur la valeur du $bit_{np-\delta}$.

Dans le contexte d'un codage convolutif non-récursif, la prise de décision du $bit_{np-\delta}$ s'exécute en estimant l'état de la mémoire du codeur à un niveau de profondeur compris entre (np-δ) et (np-δ+$(K-2)$). Grâce à la caractéristique du codeur de garder en mémoire les valeurs des derniers (k-1) bits du message, la valeur du bit concerné peut être ainsi extraite (Figure 7-16).

La reconstruction des δ_p états précédents nécessite le stockage des dernières δ_p informations sur les chemins survivants. Ce module est ainsi chargé du stockage et de la gestion des informations qui sont nécessaires à la reconstruction des états des mémoires des chemins survivants. En raison de la stratégie de troncation de la mémoire, ces informations sont sauvegardées dans une mémoire circulaire (Figure 7-17).

Gestion des procédures de reconstruction des états et de stockage des informations

La réalisation physique de l'opération de prise de décision doit assurer le déroulement correct de deux opérations fondamentales:

- la sauvegarde des informations des nouveaux chemins survivants;
- le départ de la procédure de reconstruction des états du chemin.

Afin d'assurer la succession correcte de ces deux opérations, un bloc fonctionnel a été expressément développé et chargé de la synchronisation entre les deux opérations (Figure 7-17).

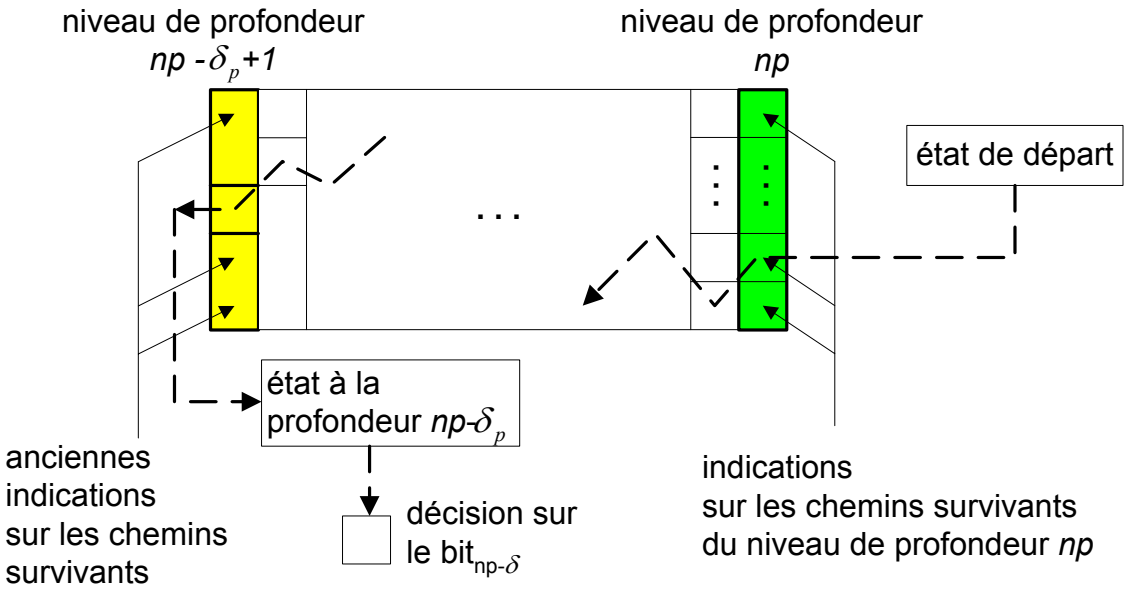

Figure 7-16: reconstruction des états et extraction de la valeur estimée du (*np*-δ)-ème bit d'information. En vert sont représentées les nouvelles informations sur les chemins survivants. En jaune, les informations qui, après l'exécution de la prise de décision, vont devenir inutiles. La distance δ_p est comprise entre les valeurs δ et (δ-*K*+*2*).

Ce bloc est constitué par deux chaînes circulaires de δ_p bascules. Chaque chaîne est sensible à un flanc différent du signal d'horloge et seule la sortie d'une bascule par chaîne présente un signal actif. Les signaux actifs des deux chaînes sont utilisés pour l'activation des bascules de stockage des informations et pour la sélection de l'unité de départ de la procédure de reconstruction (Figure 7-17). Cette stratégie permet la synchronisation des deux opérations, garantissant une réalisation physique exempte de phénomènes de *glitches* [Gras01].

Paramètres du module *TBu*

Les deux paramètres qui agissent sur ce module sont le nombre d'états *M* de la mémoire du codeur ($M= 2^{K-1}$) et la distance δ_p.
La valeur de ce dernier paramètre dépend de la stratégie de sélection du chemin qui sera utilisé pour la prise de décision: soit le chemin le plus probable, soit un chemin quelconque.

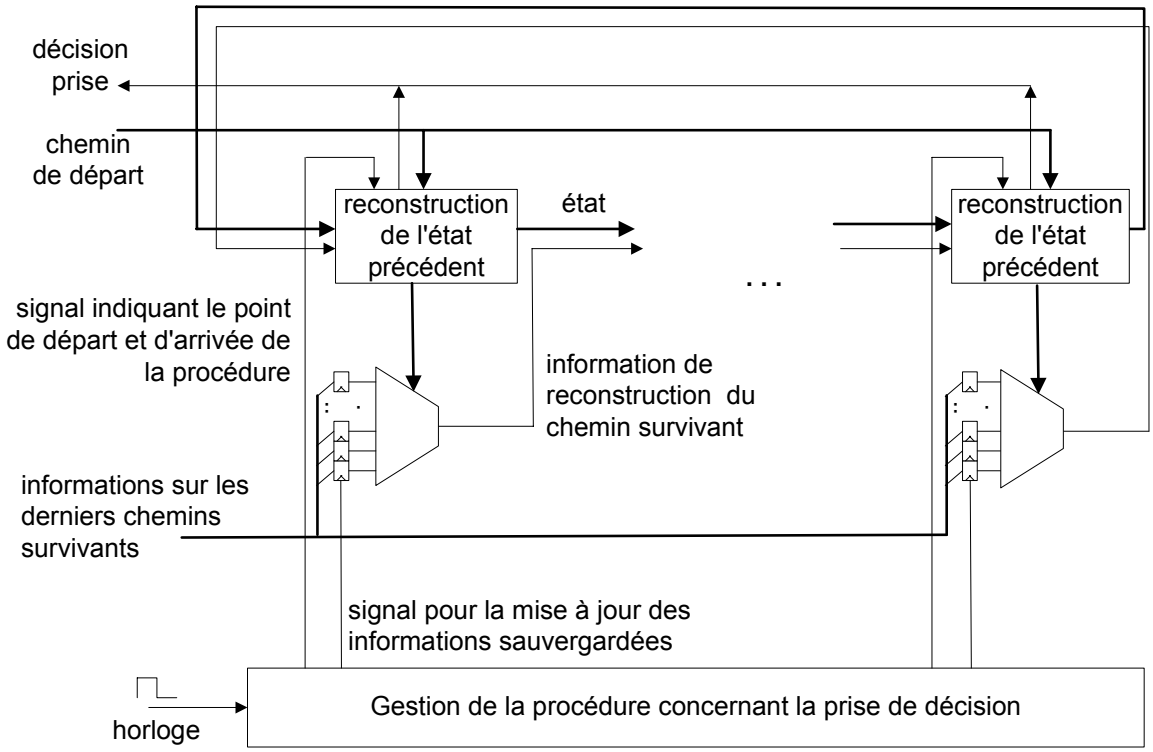

Figure 7-17: illustration de la structure pour la procédure de prise de décision. Le signal *chemin de départ* indique le chemin dont le décodage permet la prise de décision, qui sera ensuite communiquée par le signal *décision prise*.

7.3.6 Désignation du chemin permettant une prise de décision correcte

L'opération de prise de décision emploie la stratégie de la troncation de la mémoire, stratégie qui prévoit le décodage d'un chemin à la profondeur actuelle *np* jusqu'à extraire la valeur du (*np*-δ)–ème bit d'information (Figure 7-16).

Stratégie classique de sélection du chemin

Le critère classique prévoit la sélection du chemin qui possède la métrique cumulée la plus favorable. Comme illustré dans la Sous-section 6.4.2, cette approximation de l'algorithme de Viterbi permet d'anticiper la prise de décision sans provoquer une dégradation sensible de la qualité de protection, à condition d'utiliser une distance δ appropriée au contexte de codage:

$$\delta = (4 \sim 6) \cdot K \qquad (7.6)$$

Le désavantage de cette stratégie est la composante séquentielle de l'opération de sélection, qui s'accroît avec le nombre de chemins à vérifier (Figure 7-18). On utilise en effet *K-1* stades de comparaisons.
Bien que cette stratégie ait été adoptée pour la conception du système de décodage de base, l'exploration des potentialités de cette approche matérielle stimule l'analyse d'une solution alternative.

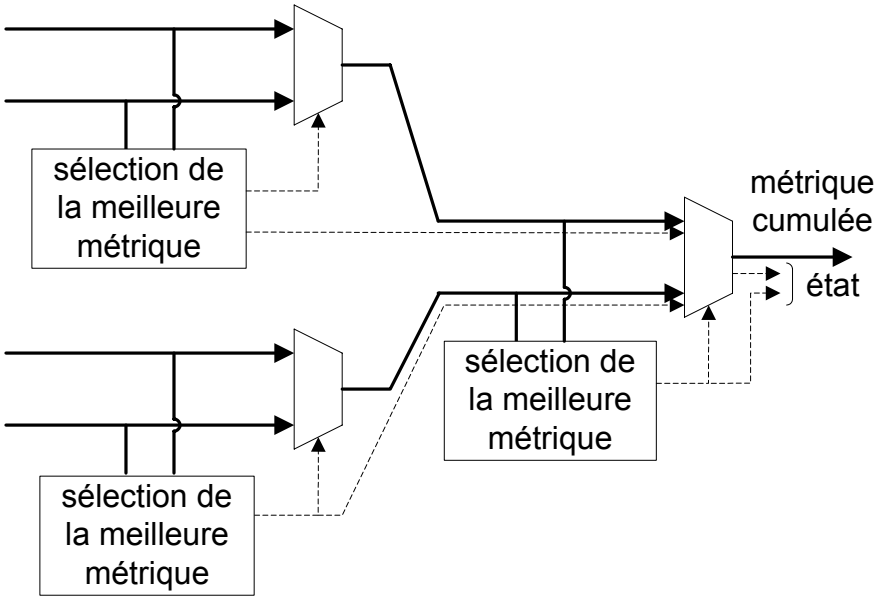

Figure 7-18: exemple de réalisation de l'unité *BPS* pour la détermination du chemin le plus favorable en analysant les métriques cumulées des quatre états (code convolutif avec longueur de contrainte de 3).

Solution alternative: la désignation d'un chemin quelconque

Les désavantages d'une exécution séquentielle peuvent être évités en changeant la stratégie de l'opération sélection: la désignation d'un chemin quelconque pour la prise de décision.

Cette stratégie exploite le phénomène de l'aggravation des valeurs de métriques cumulées de tous les chemins dont le décodage comporte une prise de décision erronée[43]. La stratégie prévoit le choix d'un chemin à un niveau de profondeur $np+\delta_{zp}$ [44] suffisamment grand, qui ne peut être attendu que par les

[43] Ces chemins seront nommés par la suite *chemins incorrects*.
[44] Dans ce cas, la distance δ est nommée δ_{zp} de manière à mieux séparer les deux stratégies: d'une part celle utilisant le chemin le plus probable (distance δ) et de l'autre celle choisissant un chemin quelconque de départ (distance δ_{zp}).

chemins comportant une décision correcte du *np*-ème bit d'information[45] (Figure 7-19).

Détermination de la distance δ_{zp}

Afin de rendre négligeable la dégradation introduite par cette méthode, la détermination de la distance δ_{zp} doit considérer la relation entre:

- l'augmentation minimale des métriques des chemins incorrects, et
- la métrique cumulée la plus défavorable des chemins corrects au niveau de profondeur $np+\delta_{zp}$ (Figure 7-19).

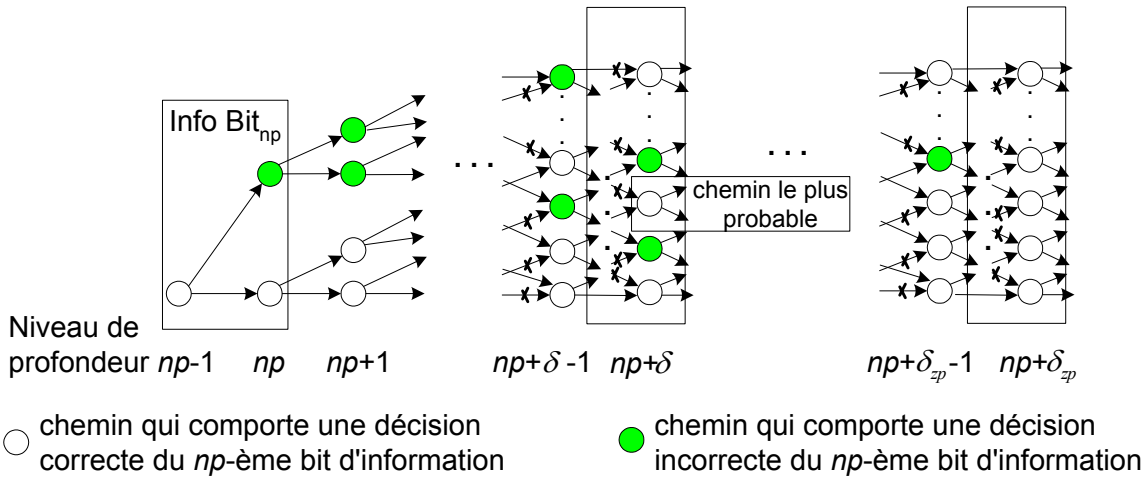

Figure 7-19: illustration graphique des deux stratégies de sélection du chemin utilisé pour l'opération de prise de décision du *np*-ème bit d'information bit_{np}. Cas simplifié.

Pour la détermination de la distance δ_{zp}, on suppose ainsi que:

- A chaque transition, l'augmentation minimale des métriques cumulées des chemins incorrects est de

$$n \cdot MinMetrCroissance. \qquad (7.7)$$

- La métrique la moins favorable des chemins corrects est estimée à l'aide de la différence maximale existant entre les valeurs extrêmes des métriques cumulées ($différence_{max}$ (7.1)). En supposant que la métrique la plus favorable coïncide avec le nombre exact d'erreurs de transmission, la valeur de métrique la plus défavorable est:

[45] Dans la suite du paragraphe, ces chemins seront appelés "chemins corrects".

$$\text{nombre d'erreurs}_{0\rightarrow np+\delta_{zp}} + \textit{différence}_{\max} \tag{7.8}$$

A partir de ces assertions, l'établissement de la distance δ_{zp} est ainsi soumis à l'inéquation

$$\delta_{zp} > \delta_{zp,\min} = \frac{\text{nombre d'erreurs}_{\text{np}\rightarrow\text{np}+\delta_{zp,\min}} + \textit{différence}_{\max}}{n \cdot \textit{MinMetrCroissance}}, \tag{7.9}$$

afin d'éviter une dégradation de la qualité de protection offerte par l'algorithme de Viterbi.
Par rapport à la stratégie classique, cette stratégie doit atteindre évidemment une profondeur δ_{zp} plus grande ($\delta_{zp} > \delta$), dont l'augmentation peut être estimée en groupant les inéquations (6.4) et (7.9):

$$\delta_{zp,\min} \approx \delta_{\min} + \frac{\textit{MaxContribBranche}}{n \cdot \textit{MinMetrCroissance}} \cdot (K-1). \tag{7.10}$$

Distance δ_{zp} à l'aide de simulations

L'utilisation de l'inéquation (7.9) nécessite la connaissance précise de plusieurs caractéristiques techniques qui sont propres au code convolutif utilisé. Par conséquent, on souhaite disposer d'une règle empirique qui facilite la détermination d'une distance δ_{zp} adaptée au contexte de codage.

Les résultats de nos simulations montrent qu'une distance δ_{zp} supérieure à 6-8 fois la longueur de contrainte K suffit à rendre négligeable la dégradation de la qualité de protection due à cette stratégie (exemple: Figure 7-20). Les simulations effectuées montrent ensuite une relation entre les dégradations de la qualité des deux stratégies, relation qui suit empiriquement l'équation[46]

$$\underset{\text{meilleur chemin}}{\text{dégradation}}(\delta) \approx \underset{\text{chemin quelconque}}{\text{dégradation}}\left(\delta_{zp} = \delta + 2 \cdot K\right). \tag{7.11}$$

Stratégie du décodage d'un chemin quelconque: conclusions

L'avantage de disposer d'une solution simple et rapide est contrebalancé par l'augmentation de la distance δ_{zp} de troncation de la mémoire. Cette solution

[46] En comparant cette relation (7.11) avec l'équation (7.10), on peut établir une meilleure correspondance entre les deux en remplaçant la valeur de croissance maximale par la valeur moyenne (Table 6-5).

implique l'augmentation du retard algorithmique ainsi que des ressources de stockage des informations des chemins survivants. Par conséquent, l'utilisation de cette stratégie est strictement soumise à l'analyse des coûts supplémentaires provoqués par l'augmentation de la distance de troncation .

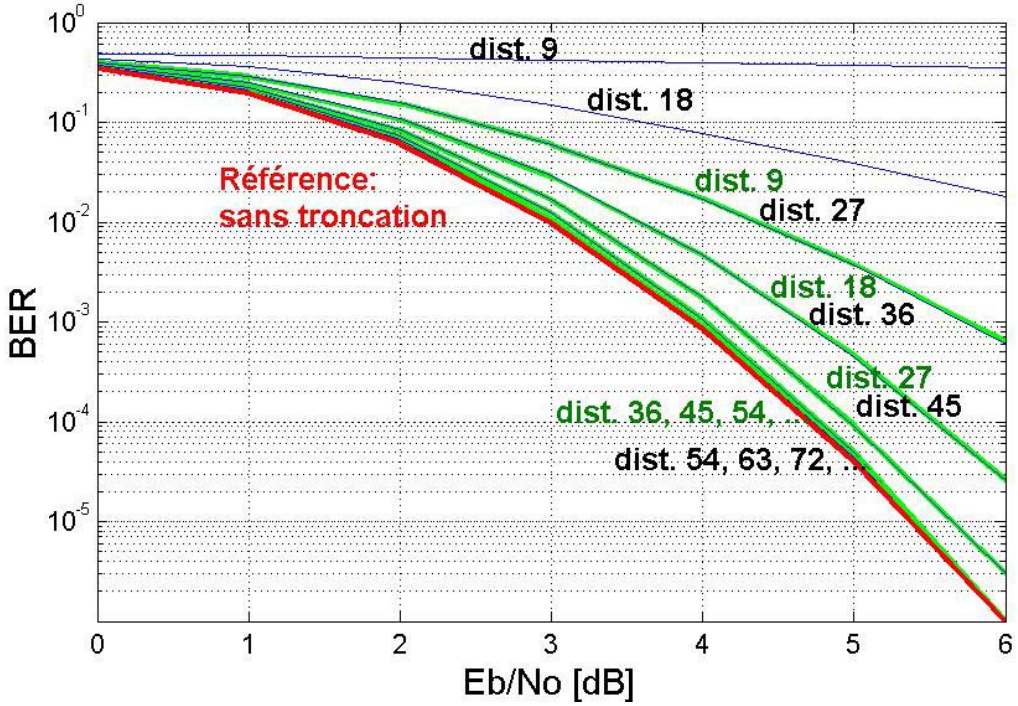

Figure 7-20: dégradations causées par la troncation de la mémoire (distances δ et δ_{zp}). Le graphe rouge représente la qualité de protection offerte par l'algorithme original de Viterbi. Le graphe vert en trait gras indique la qualité de protection obtenue selon la stratégie classique du choix du chemin le plus prometteur. Le graphe bleu en trait fin indique la qualité de protection obtenue en décodant un chemin quelconque (dans ce cas les chemins liés à l'état 'zéro'). Paramètres de simulation: $\text{codeur}_{256\text{états}}$, messages de 1000 bits, 8 bits de terminaisons, 1 million de messages, distances multiples de la longueur de contrainte du code.

7.4 Synthèse du système de base

Cette section illustre les résultats de synthèse de chaque module du système de base à l'aide du compilateur *Design Compiler* (*Synopsys*). La procédure de synthèse a suivie une stratégie *Bottom-up* et utilisée la librairie *Virtual Silicon Technology VST25* (*UMC25*, 0.25µm). La liste et la description détaillée des paramètres utilisés sont illustrés par la Table 7-4.

7.4.1 Stratégie adoptée

Le but de la stratégie de synthèse adoptée (Table 7-4, Figure 7-21) est la prise de connaissance des caractéristiques et des potentialités de chaque module du système de base, sans orienter la procédure de synthèse vers un but particulier.

Cette stratégie de synthèse n'implique ainsi aucune procédure d'optimisation (sur le plan de l'architecture du système, d'implantation logique ou du choix des cellules), ce qui permet une vue d'ensemble des caractéristiques propres à chaque module. La marge de manœuvre qui en découle peut être ensuite exploitée pour améliorer ultérieurement les potentialités du système de base.

Compilateur	*Synopsys Design Compiler* (*Synopsys DC*), version 2000.05-1	
Langage	*VHDL* (*Very High-Speed Integrated Circuit Hardware Description Language*)	
Librairie	*Virtual Silicon Technology VST25*, *UMC25* (0.25µm) (et *ALTERA APEX20k* pour la validation pratique)	
Stratégie de synthèse	*Bottom-Up*	
Paramètre	Argument	Commande pour *Synopsys DC*
Effort dans la synthèse	moyen (valeur par défaut)	*-map_effort medium*
Modèle capacitif des fils	10k	*-set_wire_load "suggested_10K"*
Conditions d'exploitation	Pire des cas	*-set_operating_conditions "WORST"*
Couverture de test	95% (valeur par défaut)	*-set_min_fault_coverage 95 –timing_critical*
Critère de l'approche	*structured*	*-set_flatten_false -design* *-set_structure_true -design*
Horloge	50ns (valeur par défaut)	*-create_clock –period 50 –waveform {" 0 " " 25 "}*

Table 7-4: Contexte de synthèse du système de base.

7.4.2 Synthèse des modules

Module *BMu*: le calcul des métriques de branches

La réalisation physique de ce module est purement combinatoire (Table 7-5). Comme déjà expliqué, la stratégie adoptée pour la conception de ce module est la description paramétrable de l'opération de calcul des $2 \cdot M = 2 \cdot 2^{K-1}$ métriques de branches. L'opération d'optimisation des calculs est ainsi déléguée au compilateur de synthèse (approche *structured*, Table 7-4).

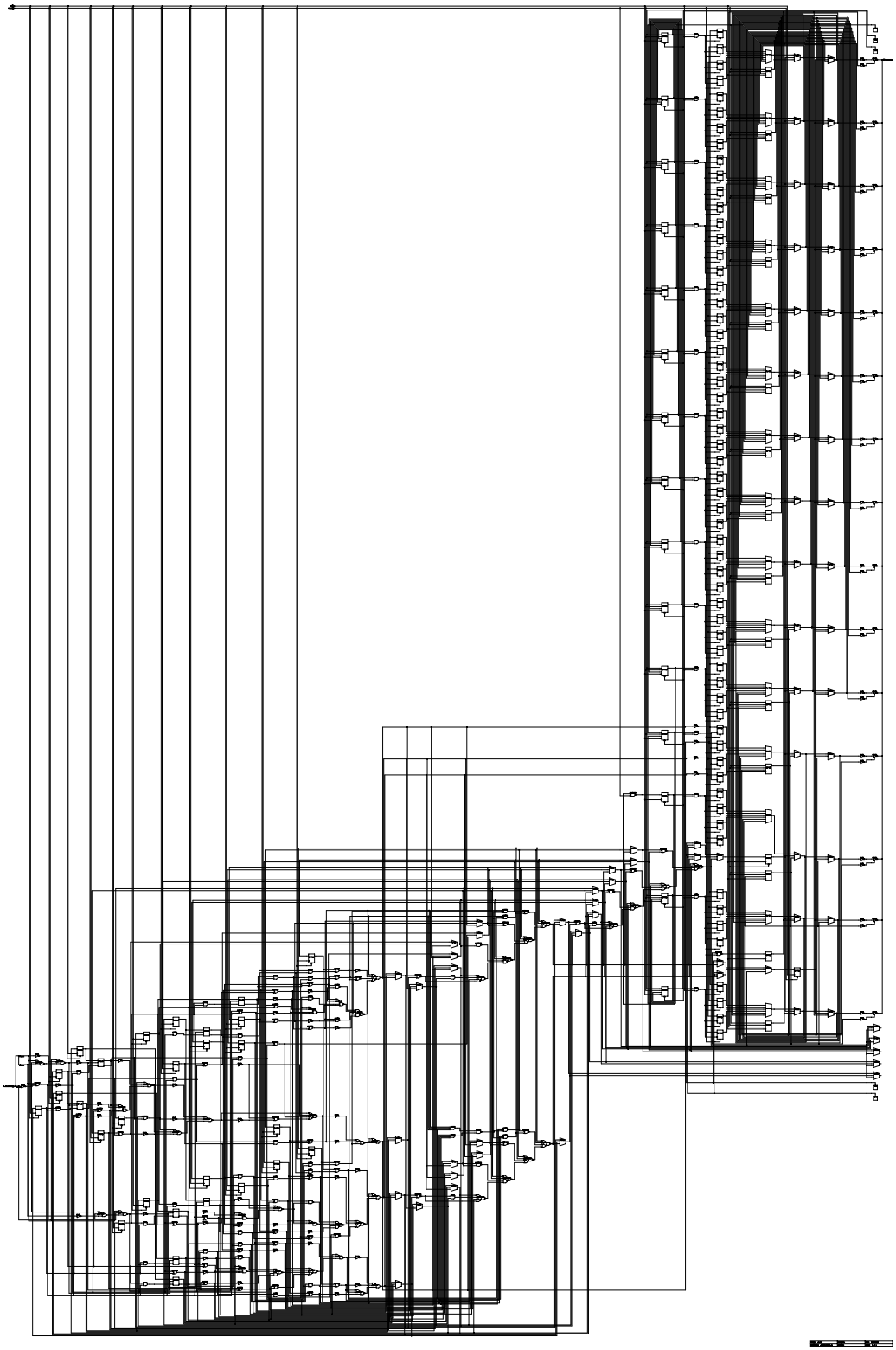

Figure 7-21: exemple de synthèse du système de base à l'aide du compilateur *Design Compiler* de *Synopsys*. Le contexte de codage est le $codeur_{4états}$.

Cette stratégie permet une adaptation rapide aux autres contextes de codage, sans nuire sensiblement à la qualité de la réalisation physique.

Contexte de codage	Virtual Silicon Technology VST25, UMC25			
	Nombre de cellules	Surface	Nombre indicatif de portes logiques (NAN2D1)	Temps d'exécution
$Codeur_{4états}$	11	356.4 μm^2	15	0.39 ns
$Codeur_{16états}$	23	617.8 μm^2	26	0.92 ns
$Codeur_{64états}$	49	1'037.5 μm^2	44	1.58 ns
$Codeur_{256états}$	135	2'613.6 μm^2	110	1.81 ns

Table 7-5: caractéristiques relatives à la réalisation physique du module *BMu*. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Module *ACSu*: la sélection des chemins survivants

Les Table 7-6 et Table 7-7 résument les caractéristiques principales de la synthèse du module *ACSu* qui est chargé de la sélection des $M=2^{K-1}$ chemins survivants.

Les principes suivis pour la conception du module sont la sélection parallèle des 2^{K-1} chemins survivants, une représentation des métriques cumulées minimisant le nombre de bits nécessaires, l'emploi d'une fonction de normalisation de ces métriques, ainsi qu'une initialisation des métriques permettant l'exploitation des conditions de départ de l'opération de codage (remise à zéro de la mémoire du codeur).

Contexte de codage	Virtual Silicon Technology VST25, UMC25					
	Nombre de cellules	Surface [μm^2]			Nombre d'éléments	
		Partie combin.	Partie non-combin.	Total	Portes logiques (*)	Bascules
$Codeur_{4états}$	174	6'462.7	3'484.8	9'947.5	272	20
$Codeur_{16états}$	679	25'280.6	13'939.2	39'219.8	1064	80
$Codeur_{64états}$	3'324	121'754.2	66'908.2	188'662.3	5124	384
$Codeur_{256états}$	13'504	490'715.3	267'632.7	758'347.9	20653	1536
(*) Nombre indicatif de portes logiques (NAN2D1) de la partie combinatoire						

Table 7-6: caractéristiques relatives à la réalisation physique de ce module *ACSu*. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Contexte de codage	Virtual Silicon Technology VST25, UMC25	
	Temps d'exécution	Fréquence (indicative) après synthèse
$Codeur_{4états}$	2.37 ns	400 MHz
$Codeur_{16états}$	2.57 ns	370 MHz
$Codeur_{64états}$	3.40 ns	280 MHz
$Codeur_{256états}$	4.32 ns	220 MHz

Table 7-7: : ordre de grandeur des performances relatives à la réalisation physique de ce module *ACSu*. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Module *BPS*: la sélection du chemin pour la prise de décision

La stratégie classique de prise de décision utilise le chemin partiel qui, au moment de l'exécution de cette opération, est estimé le plus probable.

Le principal désavantage de cette stratégie est la forte composante séquentielle de l'opération de sélection du chemin montrant la métrique cumulée la plus favorable (Figure 7-18). La réalisation physique de cette unité (Table 7-8) met en évidence cette problématique: le nombre de stades de comparaison s'accroît parallèlement à l'augmentation du nombre de chemins impliqués dans la sélection.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm				
	Nombre de cellules (blocs de comparaison)	Surface [μm^2]	Nombre indicatif de portes logiques (NAN2D1)	Stades de comparaison	Temps [ns]
$Codeur_{4états}$	5 (3)	2'265.1	95	2	2.24
$Codeur_{16états}$	21 (15)	12'687.8	534	4	5.42
$Codeur_{64états}$	89 (63)	68'270.5	2'873	6	10.60
$Codeur_{256états}$	393 (255)	301'086.7	12'672	8	15.35

Table 7-8: réalisation physique du module *BPS* par blocs de comparaisons (Table 7-9). Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm			
	Nombre de cellules	Surface Combinatoire	Nombre indicatif de portes logiques (NAN2D1)	Temps
$Codeur_{4états}$	16	744.5 μm^2	31	1.07 ns
$Codeur_{16états}$	18	839.5 μm^2	35	1.32 ns
$Codeur_{64états}$	23	1'077.1 μm^2	45	1.64 ns
$Codeur_{256états}$	25	1'172.2 μm^2	49	1.76 ns

Table 7-9: réalisation physique du bloc responsable de la comparaison de deux métriques cumulées, de manière à déterminer la meilleure. Ce bloc représente l'unité de base pour la construction du module *BPS*. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Module *TBu*: la prise de décision

La réalisation matérielle des modules *BPS* et *TBu* a suivi l'approche la plus classique: la détermination du chemin partiel le plus probable afin d'extraire la valeur du bit le plus vieux pas encore livré.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm					
	Nombre de cellules (nombre de blocs de mémoire: δ_p)	Surface [μm^2]			Nombre d'éléments	
		Partie combinatoire	Partie non-combinatoire	Total	Portes logiques (*)	Bascules
$Codeur_{4états}$	52 (15)	4'625.3	17'131.0	21'756.3	195	95
$Codeur_{16états}$	107 (25)	26'373.6	80'997.8	107'371.4	1110	455
$Codeur_{64états}$	337 (35)	116'154.7	406'224.7	522'379.4	4889	2316
$Codeur_{256états}$	1210 (45)	488'806.6	2'027'797.3	2'516'604.0	20573	11616
(*) Nombre indicatif de portes logiques (NAN2D1) de la partie combinatoire						

Table 7-10: réalisation physique de la procédure complète de prise de décision à partir de blocs chargés du stockage des informations et de reconstruction de l'état précédent de la mémoire (Table 7-11). La distance δ_p est égale à $(5 \cdot K)$. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Les résultats de la synthèse du module *TBu* sont illustrés à l'aide de la Table 7-10. La réalisation du module se base sur la concaténation de δ_p blocs opérationnels (Table 7-11), chacun chargé:

1. du stockage des informations des M chemins survivants appartenant au même niveau de profondeur, et
2. de la reconstruction de l'état précédent du chemin survivant sélectionné.

Par rapport au paramètre de distance δ_p, l'implantation considère la reconstruction itérative et successive de $\delta_p=(5 \cdot K)$ états de mémoire.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm					
	Nombre de cellules	Surface [μm^2]			Nombre d'éléments	
		Partie combinatoire	Partie non-combinatoire	Total	Portes logiques (*)	Bascules
$Codeur_{4états}$	11	301.0	736.6	1'037.5	13	4
$Codeur_{16états}$	43	1'029.6	2'827.4	3'857.0	43	16
$Codeur_{64états}$	147	3'215.5	11'191.0	14'406.4	135	64
$Codeur_{256états}$	514	10'486.1	44'645.0	55'131.1	441	128
(*) Nombre indicatif de portes logiques (NAN2D1) de la partie combinatoire						

Table 7-11: réalisation physique du bloc responsable du stockage des informations concernant les chemins survivants à une certaine profondeur. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

7.4.3 Surface de silicium

La Figure 7-22 montre une vue d'ensemble de la surface demandée par chaque module. Les deux modules les plus exigeants sont le module *ACSu* et le *TBu*. Indépendamment du nombre d'états considérés, ces deux modules représentent en effet plus de 90% de la surface du système de base entier (Figure 7-23).

Etant chargé de la sélection parallèle des $M=2^{K-1}$ chemins survivants, la surface du module *ACSu* dépend linéairement du nombre d'états représentables par la mémoire du codeur. Les opérations de sélection sont accomplies par M unités parallèles et les M métriques cumulées sont sauvegardées dans le module même.
La surface du module *TBu* dépend non seulement du nombre M d'états, mais aussi de la valeur de la distance δ_p. Le module est constitué de l'enchaînement de δ_p unités fonctionnelles, chacune responsable de la reconstruction d'un état de mémoire en plus du stockage des informations liées à M chemins survivants.
Si on considère que les paramètres M et δ_p dépendent tous les deux de la longueur de contrainte K, on obtient un module dont la demande de surface

s'accroît rapidement avec l'augmentation de la taille de mémoire du codeur convolutif (Figure 7-23).

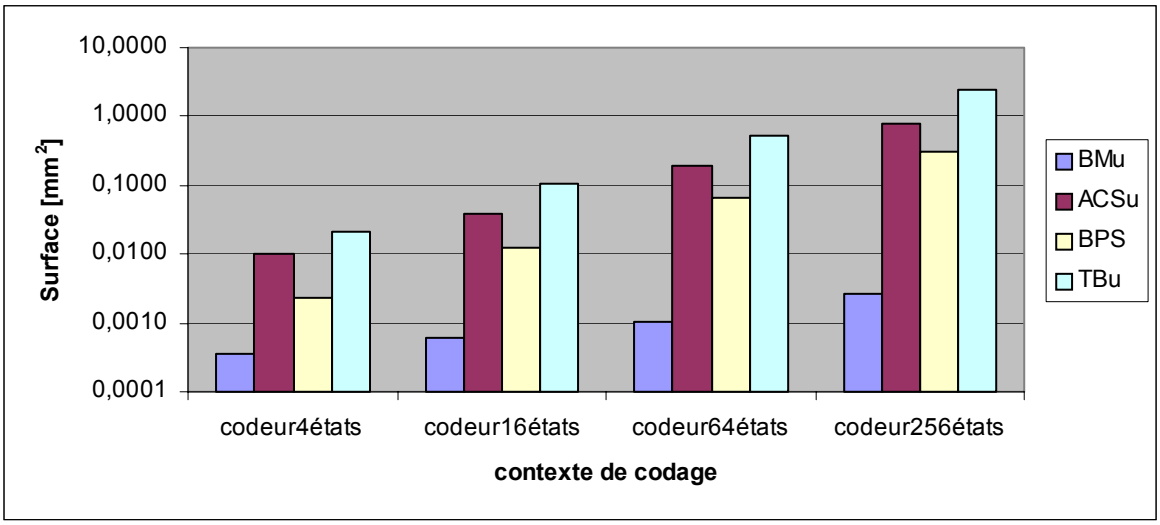

Figure 7-22: vue d'ensemble des valeurs de surface de chaque module.

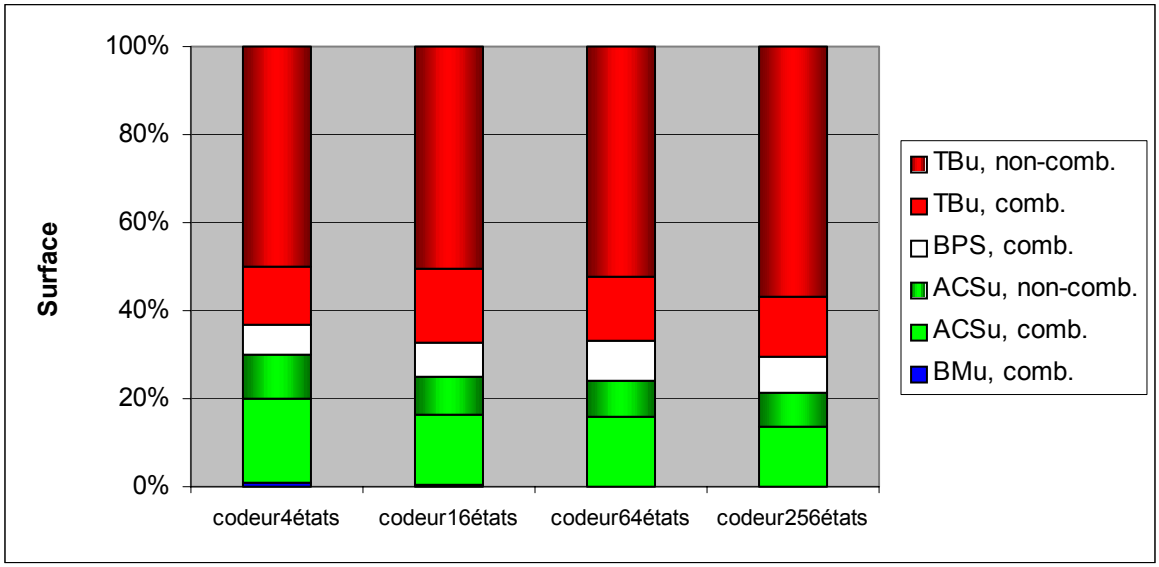

Figure 7-23: vue d'ensemble de la répartition de la surface entre les modules. Légende: *comb.*: partie combinatoire du module, *non-comb.*: partie non-combinatoire du module.

7.4.4 Vitesse d'exécution

La Table 7-12 donne une vue d'ensemble des temps d'exécution des diverses tâches, valeurs obtenues par la synthèse du système de base. Les estimations de la vitesse d'exécution après routage sont obtenues en réduisant d'un facteur deux les valeurs de vitesse résultant après synthèse.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm				
	Temps d'exécution des unités selon *Design Compiler* de *Synopsys* (après synthèse)				Estimation après routage
	Unité *BMu*	Unité *ACSu*	Unité *BPS*: détermination du meilleur chemin	Unité *TBu*: structure séquentielle (δ_p=5·*K*)	Débit de traitement du système (*)
$Codeur_{4états}$	0.39 ns	2.37 ns	2.24 ns	13.08 ns	38 Mbits/s
$Codeur_{16états}$	0.92 ns	2.57 ns	5.42 ns	59.95 ns	8 Mbits/s
$Codeur_{64états}$	1.58 ns	3.40 ns	10.60 ns	88.17 ns	6 Mbits/s
$Codeur_{256états}$	1.81 ns	4.32 ns	15.35 ns	113.57 ns	4 Mbits/s
(*) L'estimation après routage est obtenue par la pondération d'un facteur deux des valeurs de synthèse.					

Table 7-12: vue d'ensemble des valeurs concernant la vitesse d'exécution des divers modules.

Ces résultats montrent que les performances des divers modules réagissent différemment à la variation du contexte de codage. Si les performances des modules *ACSu* et *BMu* sont relativement stables, le débit de traitement des deux autres modules se dégrade parallèlement à l'augmentation de la longueur de contrainte *K*.

En raison de la tâche assignée, ces deux modules montrent en effet des composants séquentiels, dont l'importance dépend directement du paramètre *K*: *K-1* stades de comparaisons (module *BPS*) et la concaténation séquentielle de δ_p=5·*K* unités opérationnelles (module *TBu*). Ces composants séquentiels agrandissent la profondeur logique de la réalisation matérielle de ces opérations, ralentissant le temps d'exécution des tâches assignées. Ces composants séquentiels influencent évidemment les performances du système entier: les performances de l'opération de prise de décision déterminent le débit de traitement maximal du système de décodage.
Bien que les performances de l'opération de prise de décision dégradent parallèlement à l'augmentation de la longueur de contrainte *K* du code, ce système de décodage assure un débit de traitement suffisant aux exigences des

standards *UMTS*: l'estimation du débit de traitement indique une valeur de 4 millions de bits d'information traités par seconde (Table 7-12).

7.4.5 Taux d'activité des signaux liant les modules

On termine la discussion sur les aspects de la synthèse du système en analysant les taux d'activité des signaux[38] qui lient les différents modules du système (Figure 7-24 et Table 7-13).

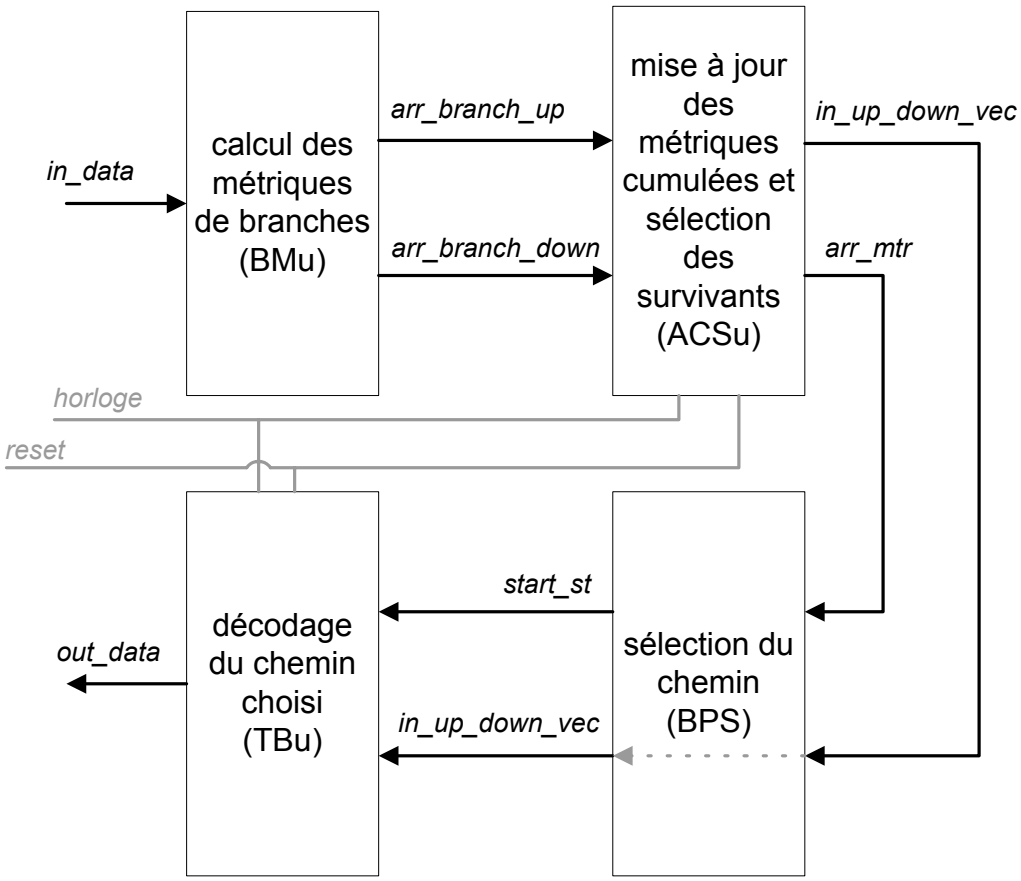

Figure 7-24: signaux liant les modules du système (Table 7-13).

Outil et simulations

Les taux d'activité sont déterminés à l'aide de l'outil *ModelSim* (*Model Technology*) qui, à partir d'une séquence donnée de symboles d'entrée, simule le comportement du système entier (décrit en *VHDL*). Ce faisant, cet outil permet la caractérisation des signaux liant les divers modules, caractérisation qui peut être ensuite utilisée par *Design Compiler* (*Synopsys*) pour une évaluation plus précise de la consommation d'énergie de chaque module (Chapitre 8).

Signal	***Description***
horloge	Signal d'horloge
reset	Remise à zéro
in_data	Symboles reçus
arr_branch_up/down	Métriques de branches
in_up_down_vec	Informations sur les chemins survivants
arr_mtr	Métriques cumulées
start_st	État de départ de la procédure de décodage
out_data	Décision sur l'ancien $bit_{t-\delta}$

Table 7-13: liste des signaux utilisés pour la description *VHDL* du système.

Deux sont les cas considérés pour la prise de connaissance des taux d'activité des signaux. La première classe de simulations considère une transmission numérique de données absente d'erreurs (exemple: Table 7-14). Dans ce cas, les symboles sont crées en codant un flux de bits d'information aléatoire.
L'autre classe de simulations prend en compte, évidemment, des situations de transmission extrêmement désastreuses en termes d'erreurs. Dans ces simulations, les symboles d'entrée du décodeur sont entièrement générés de manière aléatoire (exemple: Table 7-15).

Résultats des simulations

A partir des résultats de ces simulations, dont la Table 7-14 et Table 7-15 montrent deux exemples représentatifs, on peut analyser et commenter le comportement des modules constituant le système de décodage.

Le module *BMu* calcule les diverses métriques de branches à partir des symboles reçus. L'activité de ce module est strictement dépendante de l'activité des symboles reçus (*in_data*). A titre d'exemple, le codage d'une séquence infinie de zéros en l'absence d'erreurs de transmission génère une situation stationnaire dans le module *BMu*. Dans cette situation extrême, on assiste à une non-transition des éléments logiques du module.

Le module *ACSu* se partage en deux parties. La première partie, étant responsable des opérations permettant la sélection des chemins survivants, est entièrement réalisée par logique combinatoire. Par contre, la seconde partie est chargée du stockage des valeurs des métriques cumulées des chemins survivants. L'activité de ce module est ainsi influencée d'une part par les taux d'activité des métriques de branches et de l'autre par ceux des métriques cumulées.

ModelSim SE PLUS 5.5e Revision2001.10												
in_data			***arr_branch_up/ arr_branch_down***			***arr_mtr***			***start_st***			***out_data***
bit	P[1]	Activité	*bit*	P[1]	Activité	*bit*	P[1]	Activité	*bit*	P[1]	Activité	P[1]=0.48
2 MSB	0.51	0.47	*1 MSB*	0.50	0.50	*5 MSB*	0.00	0.00	*6 MSB*	0.49	0.45	Activité =0.47
1 MSB	0.49	0.55	*0 LSB*	0.51	0.50	*4*	0.00	0.00	*5*	0.49	0.45	
0 LSB	0.50	0.47				*3*	0.93	0.11	*4*	0.50	0.45	
						2	0.54	0.49	*3*	0.49	0.45	
						1	0.47	0.49	*2*	0.49	0.45	
						0 LSB	0.50	0.49	*1*	0.49	0.45	
									0 LSB	0.49	0.45	
						in_up_down_vec						
						P[1]= 0.40 Activité=0.47						

Table 7-14: taux d'activité des signaux liant les modules (Figure 7-24, Table 7-13) du système $Codeur_{256états}$. Les taux d'activité sont obtenus à l'aide du programme *Modelsim*[47]. La simulation considère la réception d'une séquence de symboles créé par le codage de 1'000 bits d'information. Ces bits ont été générés de manière aléatoire. Le signal de remise à zéro n'a été utilisé qu'une fois, au début de la simulation. Légende: 'P[1]' indique la probabilité que le bit soit égal à l'état '1', 'activité' représente le nombre de transition par cycle d'horloge.

Les taux d'activité des métriques de branche sont stable et ne changent que dans une situation (artificielle) de codage stationnaire en absence d'erreurs. Par contre, l'activité des métriques cumulées (*arr_mtr*) est fonction de la stratégie adoptée pour sa représentation (nombre de bits et fonction de normalisation) et des conditions de transmission. Etant donné que la métrique cumulée indique le nombre d'erreurs rencontré par le chemin, son taux d'activité est évidemment influencé par le débit d'erreurs de transmission (Table 7-14 et Table 7-15).
L'analyse du taux d'activité du signal d'information *in_up_down_vec* prouve l'utilisation d'une fonction de sélection qui favorise (en cas de métriques équivalentes) le chemin dont l'état précédent possède le *LSB* égal à zéro. Ce signal, qui indique le *LSB* de l'état de provenance du chemin survivant, montre en effet la prédominance de l'état logique 'zéro' (Table 7-14 et Table 7-15).

[47] *ModelSim SE/EE PLUS 5.5e* de *Model Technology Incorporated*, *Mentor Graphics* (Rev. 2001.10).

ModelSim SE PLUS 5.5e Revision2001.10										
in_data			***arr_branch_up/ arr_branch_down***			***arr_mtr***			***start_st***			***out_data***
bit	P[1]	Activité	*bit*	P[1]	Activité	*bit*	P[1]	Activité	*bit*	P[1]	Activité	P[1]=0.45
2 MSB	0.49	0.43	*1 MSB*	0.50	0.44	*4 MSB*	0.06	0.08	*1 MSB*	0.32	0.42	Activité=0.48
1 MSB	0.51	0.44	*0 LSB*	0.50	0.51	*3*	0.49	0.13	*0 LSB*	0.49	0.49	
0 LSB	0.49	0.45				*2*	0.51	0.27				
						1	0.51	0.46				
						0 LSB	0.50	0.54				
						in_up_down_vec						
						P[1]=0.32 Activité=0.41						

Table 7-15: le taux d'activité des signaux liant les modules (Figure 7-24, Table 7-13) du système $Codeur_{4états}$. Les taux d'activité sont obtenus à l'aide du programme *Modelsim*. Cette analyse se base sur la génération aléatoire de 1'000 symboles d'entrée (signal *in_data*). Le signal de remise à zéro n'a été utilisé qu'au début de la simulation. Légende: voir Table 7-14.

Le module *BPS* est chargé du choix du chemin partiel le plus probable. Par conséquent, la fonction de sélection est influencée par le contenu du message et par les erreurs de transmission. Dans la situation d'une équiprobabilité des messages et d'absence d'erreurs de transmission, le chemin le plus favorable peut montrer de manière équitable tous les états de mémoire (signal *start_st*, Table 7-14). Par contre, en présence d'un fort débit d'erreurs, l'opération de sélection du chemin est confrontée à des situations d'équivalence des métriques cumulées. Dans cette situation, le taux d'activité de ce signal de sortie montre les traces de l'utilisation de fonctions de sélection inéquitable en cas d'équivalence des métriques (Table 7-15). Ces fonctions sont les opérations de sélection du chemin survivant (module *ACSu*) et les unités de comparaison des chemins survivants (module *BPS*).

Le dernier module traité est le *TBu*. L'analyse de son signal de sortie, qui représente la prise de décision sur le plus vieux bit pas encore livré, ne permet pas la caractérisation du taux d'activité de ce module.
Les informations qu'on peut obtenir sur son taux d'activité dérivent de la conception du module. En effet, la stratégie de réalisation implique qu'à chaque période d'horloge:

- un des δ_p éléments de sauvegarde stocke les nouvelles informations des M chemins survivants;

- quatre bascules des deux chaînes circulaires de synchronisation (Figure 7-17) changent d'état;
- l'exécution de la prise de décision, c'est à dire la reconstruction successive de δ_p états de mémoire du chemin sélectionné.

7.5 Stratégie 'pipeline' pour la prise de décision

7.5.1 Motivation

L'analyse des résultats de synthèse montre que les performances des divers modules réagissent différemment à la variation du contexte de codage (Table 7-12). Si les débits de traitement des modules *ACSu* et *BMu* sont relativement stables, les performances des deux autres modules se dégradent avec l'augmentation de la longueur de contrainte *K*.
En raison des tâches assignées, les deux modules chargés de la prise de décision montrent de fortes composantes séquentielles dont l'importance dépend du paramètre *K*:

- *K-1* stades de comparaisons pour la sélection du chemin le plus probable et,
- la concaténation séquentielle de $\delta_p=5\cdot K$ unités opérationnelles pour la prise de décision.

Ces composants séquentiels dégradent non seulement les débits de traitement des modules mais aussi les performances du système entier (Table 7-12). Si la stratégie d'une prise de décision exploitant un chemin quelconque permet de minimiser le temps d'exécution du module *BPS*, la réalisation du module *TBu* reste l'élément crucial de notre système.
Dans le contexte d'une exploitation exhaustive du parallélisme algorithmique disponible, il devient intéressant d'explorer une stratégie *pipeline* pour l'exécution de l'opération de prise de décision. Son principe est l'exécution simultanée de plusieurs opérations de prise de décision, ce qui permet l'amélioration du débit de traitement de l'unité *TBu*.

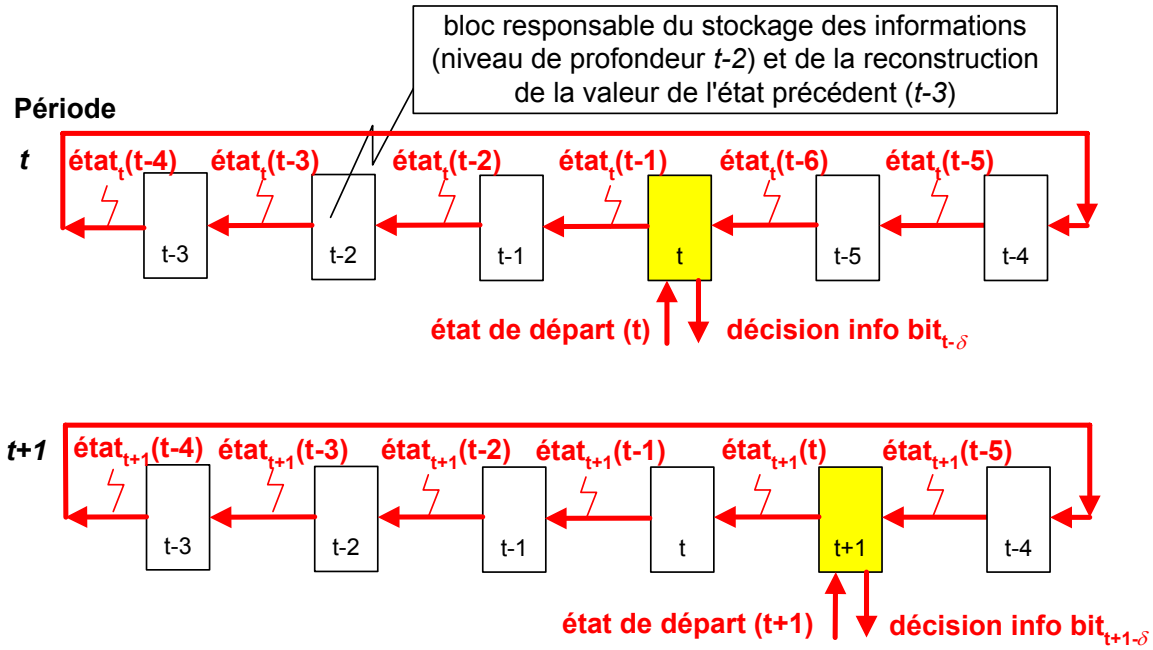

Figure 7-25: opérations de prise de décision. L'approche suivie est la reconstruction séquentielle des δ_p états précédents de la mémoire du codeur, à partir du chemin sélectionné. L'abréviation *état$_i$(t)* indique l'état de la mémoire du *i*-ème chemin de départ par rapport à la période *t*.

7.5.2 Approche pipeline

La procédure de prise de décision peut être exécutée en δ_p étages: à chaque étage, l'état précédent de la mémoire du codeur est reconstruit à l'aide des informations sauvegardées (Figure 7-26). La vitesse d'exécution d'un de ces blocs fondamentaux de reconstruction devient l'élément indiquant les potentialités du module *TBu* (Table 7-16).

En raison d'une exécution répartie en δ_p étages, cette approche demande le double du nombre de blocs fondamentaux qui sont nécessaires au stockage des informations ainsi qu'à la reconstruction d'états précédents (Figure 7-26). Evidemment, elle entraîne l'augmentation du retard algorithmique de l'opération de prise de décision, retard qui passe ainsi de δ_p à $2{\cdot}\delta_p$ périodes.

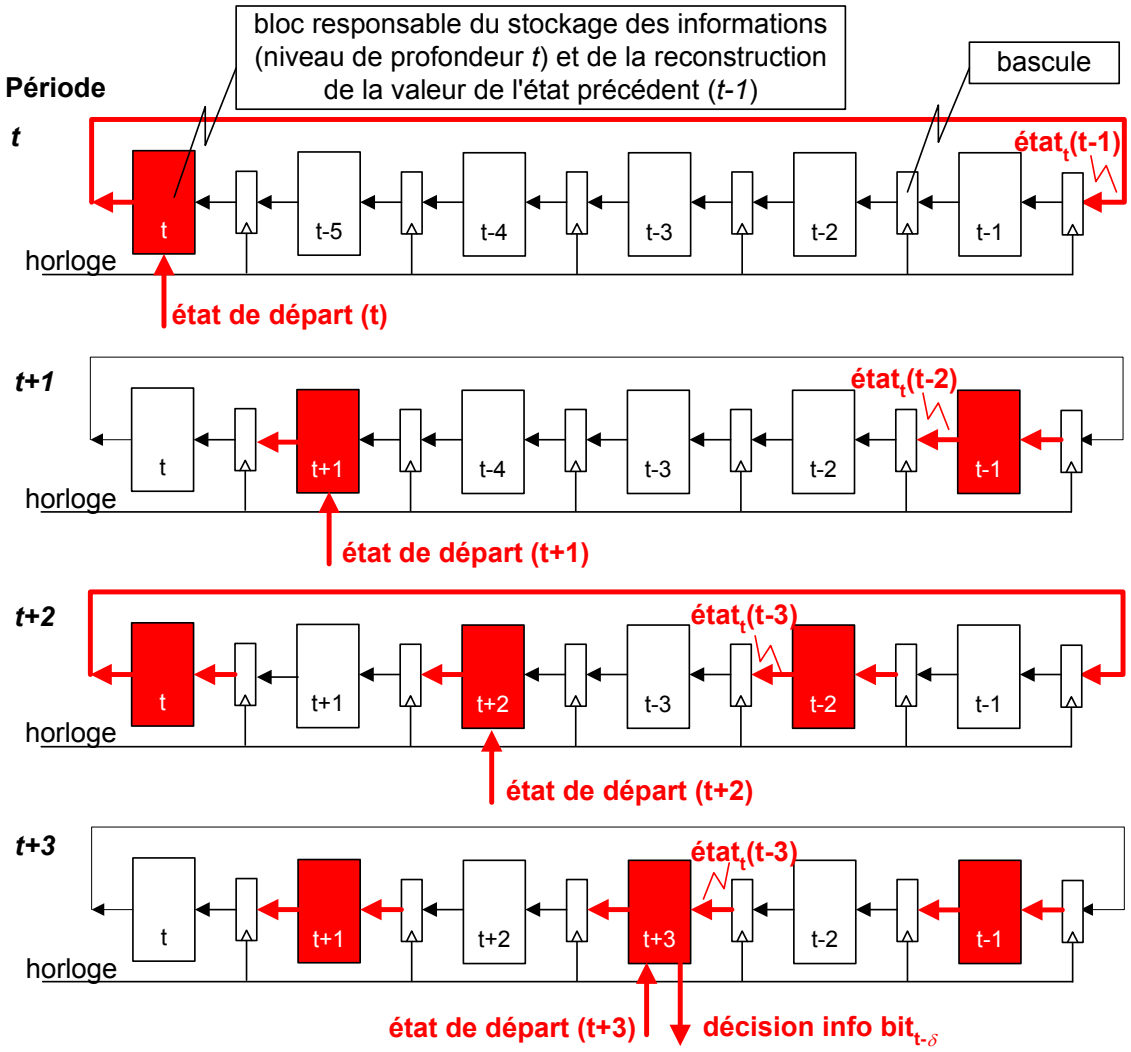

Figure 7-26: représentation graphique du déroulement de l'opération de prise de décision exploitant une structure *pipeline* (δ_p=3). L'abréviation *état*$_i$*(t)* indique l'état de la mémoire du *i*-ème chemin de départ par rapport à la période *t*.

7.5.3 Amélioration de la stratégie pipeline

L'analyse d'efficacité de cette stratégie montre une gestion insatisfaisante des opérations de reconstruction: les $2 \cdot \delta_p$ blocs fondamentaux sont en effet chargés de la reconstruction parallèle de 'seulement' δ_p chemins.

Cette structure pipeline (Figure 7-26) souffre de la reconstruction parasitaire de δ_p chemins, qui n'apporte aucune contribution à l'opération de prise de décision. Sur le plan de la consommation d'énergie, cette situation est intolérable.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm (valeurs après synthèse)				
	procédure complète de prise de décision			**bloc responsable du stockage des informations et reconstruction d'état**	
	$\delta_p = 5 \cdot K$	Temps	Fréquence indicative	Temps	Fréquence indicative
$Codeur_{4états}$	15	13.08 ns	75 MHz	0.61 ns	1'600 MHz
$Codeur_{16états}$	25	59.95 ns	15 MHz	1.16 ns	860 MHz
$Codeur_{64états}$	35	88.17 ns	11 MHz	1.87 ns	530 MHz
$Codeur_{256états}$	45	113.57 ns	8 MHz	2.09 ns	470 MHz

Table 7-16: ordre de grandeur des performances de la réalisation du module *TBu*. Les indications concernant le bloc fondamental impliquent l'opération de sélection, la lecture des informations sauvegardées et la reconstruction des états précédents du chemin sélectionné. Les paramètres et les options de synthèse sont énumérés dans la Section 7.4.

Une solution possible est une intervention sur le signal d'horloge (*Gated Clock*) des bascules interposées entre les blocs, de manière à activer seulement les δ_p blocs opérationnels qui sont impliqués dans les opérations demandées. Par conséquent, le signal d'horloge liant toutes les bascules (Figure 7-26) peut être remplacé par deux signaux d'horloge liant les bascules de manière alternée (Figure 7-27).

L'aspect de la faible consommation d'énergie peut aussi tirer profit de cette solution. Cette structure *pipeline* permet de limiter la consommation d'énergie en évitant la répétition d'opérations identiques de reconstruction. La reconstruction d'états du chemin actuellement sélectionné peut en effet bénéficier des opérations précédemment effectuées. Le bénéfice qu'on peut obtenir est proportionnel au nombre de transitions d'états que les deux chemins ont en commun.

7.5.4 Redondance des opérations de reconstruction d'états

Indépendamment de la stratégie d'exécution de l'opération de prise de décision, cette sous-section présente les bénéfices d'une élimination des répétitions (successives) d'opérations identiques de reconstruction.

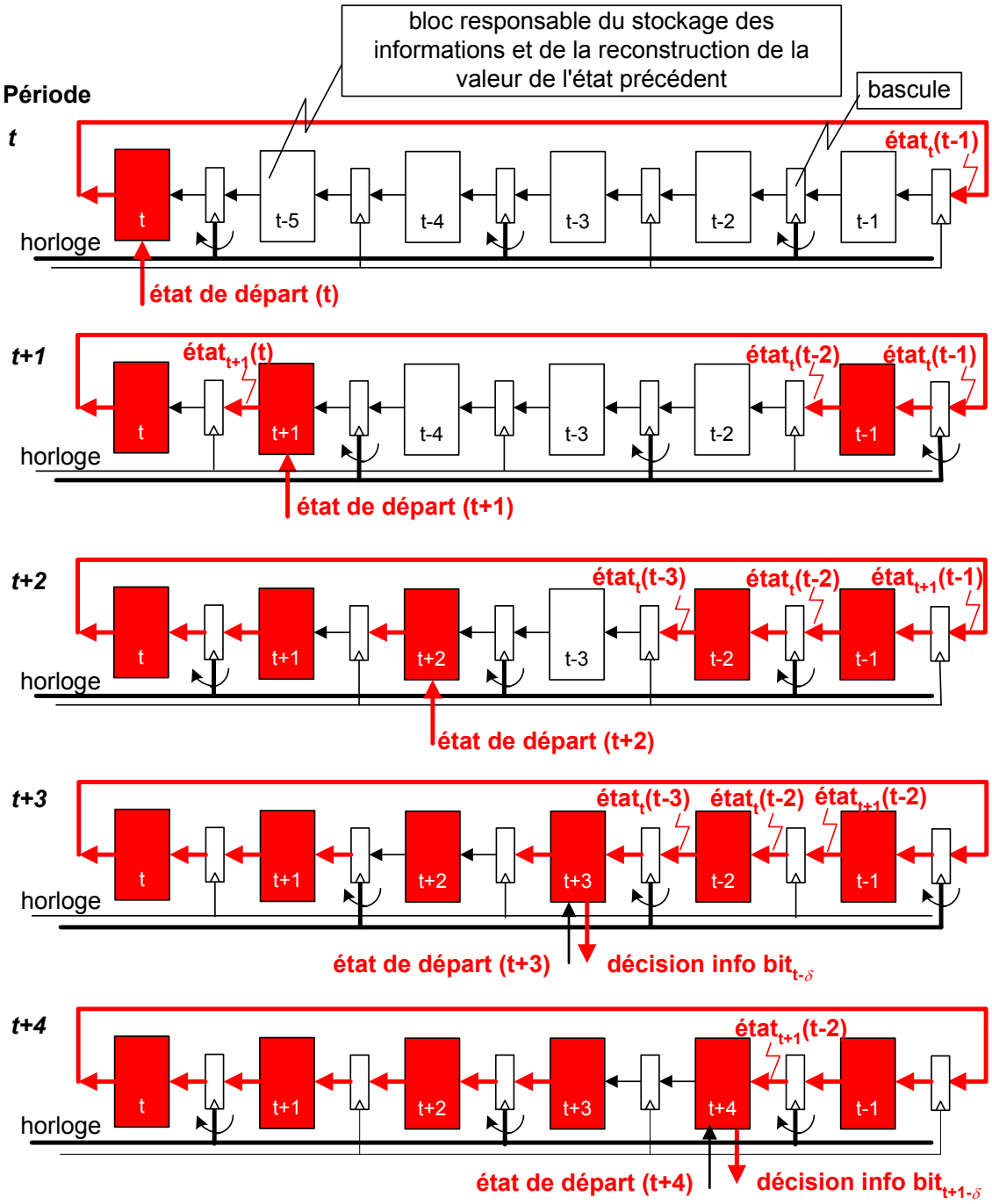

Figure 7-27: optimisation de la structure *pipeline* de la tâche de prise de décision (δ_p=3). La structure est gérée par deux signaux d'horloge qui sont activés de manière alternée (l'activité est illustrée par la taille de la ligne représentant le signal). L'abréviation $état_i(t)$ indique l'état de la mémoire du i-ème chemin de départ par rapport à la période t.

L'importance de la réduction de consommation est évidemment déterminée par le nombre de transitions que les chemins choisis ont en commun. Par

conséquent, la réduction la plus importante est atteinte lorsque le nouveau chemin sélectionné est la prolongation du chemin précédemment utilisé. Dans ce cas, après une seule opération de reconstruction, on peut utiliser entièrement tous les résultats de la reconstruction précédente.
A partir de ces évidences, on s'attend à ce que l'importance de la réduction dépende du critère de sélection du chemin, de la distance δ_p, ainsi que des erreurs de transmission.

Prise de décision par le décodage du chemin le plus probable

Dans le cas de l'utilisation du chemin le plus probable, la redondance des opérations est influencée par le nombre d'erreurs et par leur distribution temporelle.

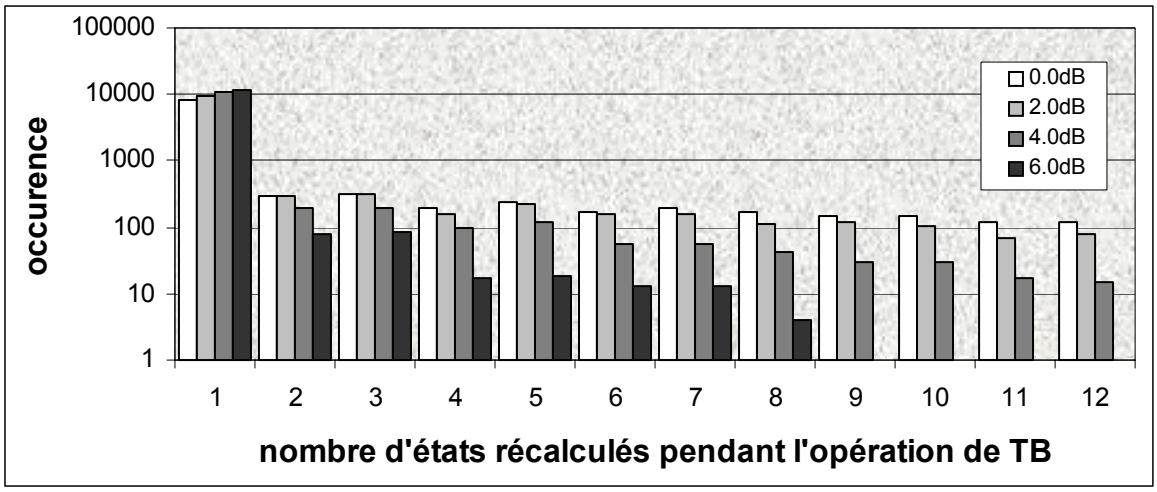

Figure 7-28: statistique du nombre d'états recalculés lors de l'opération de prise de décision, qui utilise les chemins les plus probables. Paramètres de la simulation: contexte de codage $codeur_{16états}$ et 11.8k opérations de choix du chemin le plus probable.

Le cas idéal est représenté par la sélection d'un chemin qui est la prolongation du précédent, ce qui demande la reconstruction d'un seul état de mémoire. La sélection d'un autre chemin provoque l'augmentation du nombre d'états qui doivent être recalculés, nombre qui est compris entre 2 et δ_p (Figure 7-28 et Table 7-17). L'occurrence de ce dernier cas augmente (évidemment) avec la dégradation des conditions de transmission. Dans les deux cas, l'exploitation des précédentes opérations de reconstruction (Table 7-17) permet une réduction sensible de la consommation d'énergie.

Contexte de codage	Distance δ_p	Moyenne du nombre d'états observés qui sont effectivement recalculés lors de chaque opération de prise de décision						
		E_b/N_0: 0.0 dB	1.0 dB	2.0 dB	3.0 dB	4.0 dB	5.0 dB	6.0 dB
$Codeur_{4états}$	15	2.27	2.06	1.78	1.50	1.27	1.12	1.05
$Codeur_{16états}$	25	4.26	3.38	2.46	1.73	1.31	1.12	1.05
$Codeur_{64états}$	35	7.76	5.80	3.66	2.16	1.44	1.16	1.06
$Codeur_{256états}$	45	11.81	8.67	4.86	2.41	1.46	1.15	1.05

Table 7-17: nombre moyen d'états effectivement recalculés lors d'une opération de prise de décision (TB). Simulations: contexte de codage *codeur*$_{256états}$, choix du chemin le plus probable, 1'000 opérations de prise de décision (*TB*) par trame, 10'000 trames.

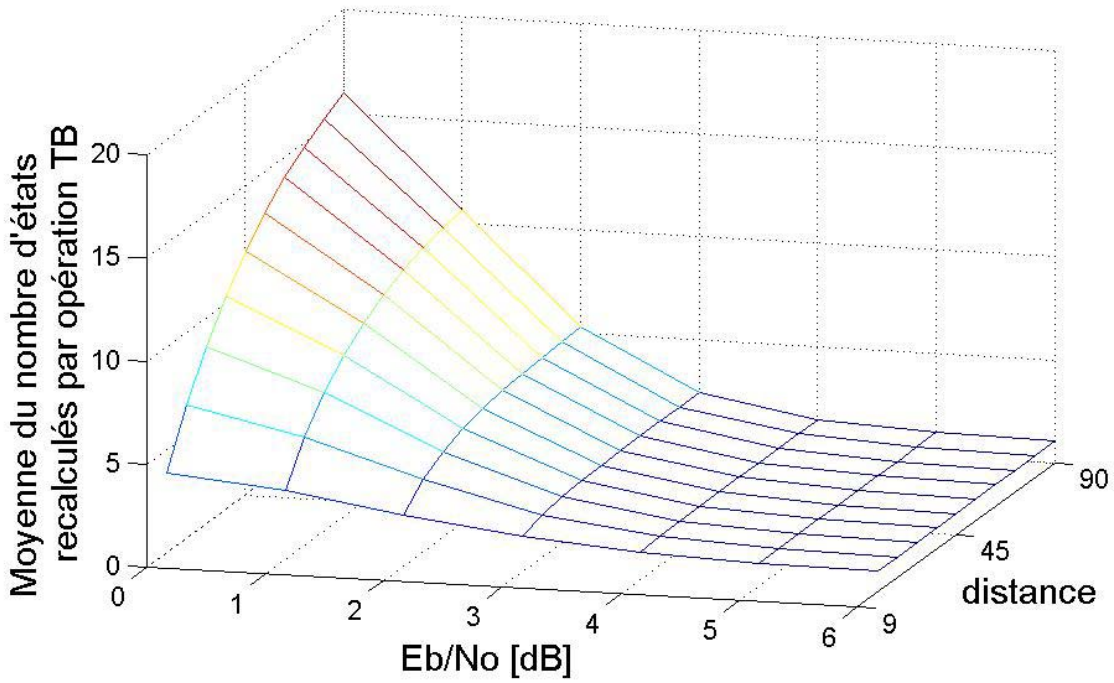

Figure 7-29: influence de la distance δ_p sur le nombre d'états recalculés. Simulation software: contexte de codage *codeur*$_{256états}$, 1'000 opérations de choix du chemin le plus prometteur par trame, 10'000 trames.

L'influence de la distance δ_p sur le nombre d'états à recalculer est par contre plus marginale (Figure 7-29). Sa contribution se restreint à la limitation du nombre d'états recalculés dans le cas où le chemin sélectionné divergerait du précédent par un nombre d'états supérieurs à δ_p. Cet événement se produit

dans de mauvaises conditions de transmission: dans le contexte de codage traité par la Figure 7-29, l'influence de la distance δ_p est observable au-dessous d'un rapport E_b/N_0 de 2dB.

Reconstruction des états de mémoire d'un chemin quelconque

L'utilisation d'un chemin quelconque offre un niveau de souplesse supérieur à l'opération de sélection, souplesse qui pourrait être ainsi exploitée afin de réduire la consommation d'énergie. On pourrait envisager de choisir (autant que possible) un chemin qui est la prolongation du précédent. Ce faisant, bien que la possibilité de survie des prolongations du dernier chemin soit fonction des conditions de transmission et du contenu du message, la consommation d'énergie peut être minimisée. Toutefois, cette solution nécessite l'exécution d'une tâche de sélection complexe, ce qui contredit les avantages (précédemment) envisagés par l'utilisation de cette méthode alternative.

Conformément aux motivations qui sont à la base de la proposition, on analyse la situation de sélection la plus simple: le choix systématique du même état de départ.
Par rapport à l'aspect de la consommation d'énergie, deux états de mémoire montrent des caractéristiques intéressantes lorsqu'ils sont utilisés comme états de départ de l'opération de prise de décision:

- L'état 'zéro' qui représente la situation d'une mémoire du codeur convolutif contenant uniquement des bits d'informations 'zéros', et
- L'état qui représente la situation d'une mémoire contenant uniquement des bits d'information 'un'.

En effet, seuls ces deux états rendent possible le choix d'un chemin qui soit la prolongation du chemin choisi à la profondeur précédente (Figure 2-15).
La probabilité de cet événement peut être ensuite renforcée par la réalisation de l'opération de sélection du chemin survivant (module *ACS*). La réalisation peut en effet assurer que, dans une situation d'équiprobabilité des métriques cumulées, la prolongation du chemin (précédemment choisi) sera sélectionnée. La contribution apportée par cette simple modification[48], qui ne modifie pas le déroulement correct de l'algorithme (voir le théorème de la *non-optimalité*), est facilement observable dans un contexte de décodage utilisant une décision ferme (Table 7-18).

[48] Dans le cas de l'utilisation de l'état de départ 'zéro', l'équivalence des métriques cumulées avantage le chemin provenant d'un état de mémoire dont le *MSB* est égal à zéro.

Contexte de codage	Longueur de contrainte du code	Message [bit]	Bits de terminaisons	Nombre de messages	Troncation [δ_p]	Occurrence observée du choix de la prolongation du chemin précédent	
						Min.	Max.
$Codeur_{4états}$	K=3	1'000	2	200'000	K,2K,..,9K	65,0%	65,5%
$Codeur_{256états}$	K=9	1'000	8	10'000	K,2K,..,9K	62,1%	65,6%

Table 7-18: analyse du cas où le chemin actuellement choisi est la prolongation du chemin choisi à la profondeur précédente. Simulations: choix systématique du chemin de départ 'zéro', opération de sélection (module *ACSu*) favorisant le choix de la prolongation du chemin précédent, conditions de transmission E_b/N_0 de 0 à 6 dB.

Sur le plan du nombre nécessaire d'opérations de reconstruction, on doit s'attendre à ce que cette stratégie alternative de sélection soit plus gourmande que l'approche classique de sélection du chemin le plus probable. Les résultats des simulations (Table 7-19) prouvent en effet qu'un nombre supérieur d'états est recalculé pendant chaque opération de prise de décision.

Contexte de codage	Distance δ_p	Moyenne du nombre d'états observés, qui sont effectivement recalculés lors de chaque opération de prise de décision						
		E_b/N_0:						
		0.0 dB	1.0 dB	2.0 dB	3.0 dB	4.0 dB	5.0 dB	6.0 dB
$Codeur_{4états}$	21	3.38	3.23	3.00	2.71	2.43	2.21	2.07
$Codeur_{16états}$	35	5.86	5.21	4.46	3.81	3.37	3.11	2.96
$Codeur_{64états}$	49	11.98	10.19	8.08	6.42	5.46	4.98	4.72
$Codeur_{256états}$	63	14.75	12.53	9.47	7.21	6.06	5.51	5.21

Table 7-19: nombre moyen d'états effectivement recalculés lors d'une opération de prise de décision (*TB*). Simulations: contexte de codage $codeur_{256états}$, 1'000 opérations de prise de décision par trame, 10'000 trames, état de départ 'zéro'.

La méthodologie de sélection du chemin est une facteur pénalisant, surtout dans des situations de transmission favorables: en sélectionnant soit la prolongation du chemin précédent, soit un chemin qui peut se réunir avec le précédent qu'après *K* transitions (Figure 7-30), le nombre moyen d'états à recalculer est superier par rapport à l'approche classique (Table 7-17 et Table 7-19).

L'influence du paramètre de la distance δ_p sur le nombre d'états à recalculer est similaire à l'approche classique.

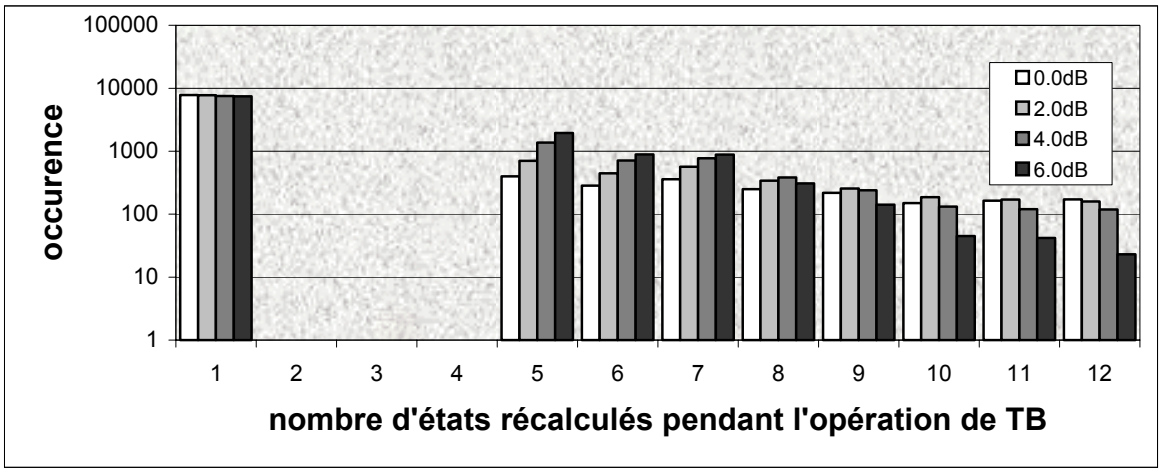

Figure 7-30: statistique du nombre d'états recalculés lors de l'opération de prise de décision (*TB*), avec l'utilisation de l'état de départ 'zéro'. Simulations: contexte de codage $codeur_{16états}$, 11.8k opérations de *TB* analysées.

7.5.5 Considérations finales

En raison des exigences établies, la réalisation du système de base est fortement influencée par la stratégie de l'opération de prise de décision. Une étude approfondie de cette opération a permis d'évaluer ses potentialités en exploitant une approche *pipeline*. Cette stratégie permet d'améliorer le débit de traitement supporté par cette opération, en éliminant les limitations dues à sa forte composante séquentielle. Les performances du système entier deviennent ainsi fonction de l'opération fondamentale de sélection des chemins survivants.
Le prix à payer est le doublement des unités de reconstruction d'état ainsi que l'augmentation de la consommation d'énergie due à la gestion des multiples opérations parallèles.

Sur le plan de la consommation d'énergie, cette étude a montré les avantages d'une stratégie de prise de décision exploitant la redondance des opérations de reconstruction. Indépendamment de la modalité de sélection, une telle stratégie permet d'envisager une réduction du nombre d'opérations de reconstruction, nombre qui passe ainsi de δ_p à des valeurs s'approchant de l'unité (Table 7-17 et Table 7-19). L'importance de la réduction est évidemment influencée par les conditions de transmission.

7.6 Conclusions

Ce septième chapitre a exploré les potentialités d'une réalisation *ASIC* d'une méthode de décodage se basant exclusivement sur l'algorithme de Viterbi.

Après l'analyse de la situation actuelle des technologies *3G*, le système de décodage a été décrit en décomposant l'algorithme de Viterbi dans ses éléments constituants. Ces blocs opérationnels sont: le bloc responsable du calcul des métriques de branche (*BMu*), celui de la mise à jour des métriques cumulées (*ACSu*), le bloc responsable de la sélection du chemin (*BPS*) et le bloc de prise de décision par le décodage (*TBu*) du chemin indiqué par le bloc précédent.

Virtual Silicon Technology*, *VST25*, *UMC25*, 0.25µm** **Contexte de codage:** $\text{Codeur}_{256\text{états}}$ Condition des analyses: ***condition de fonctionnement difficile (VDD 2.25V, 70˚C) analyse simple (*analysis_effort low*)			
	estimation après synthèse (*Design Compiler* de *Synopsys*)		
	Temps d'exécution	Dissipation intérieure par bit traité	Surface
BMu	1.81 ns	5.0 pJ	$2.6 \cdot 10^{-3}$ mm^2
ACSu	4.32 ns	1.0 nJ	$7.6 \cdot 10^{-1}$ mm^2
BPS	15.35 ns	0.2 nJ	$3.0 \cdot 10^{-1}$ mm^2
TBu (δ_p=45)	113.57 ns	2.9 nJ	2.5 mm^2

Table 7-20: estimation des caractéristiques techniques de la synthèse des blocs constituant le décodeur de Viterbi. Le contexte de codage concerne le code convolutif proposé par le standard *UMTS* avec rendement R_c=1/3.

Cette décomposition a permis de traiter séparément chaque élément, illustrant ainsi ses paramètres, leurs implications dans l'exécution de l'algorithme, les options et les variantes de conception. Une réalisation générale et paramétrable a été choisie pour l'évaluation des potentialités d'adaptation et des coûts d'emploi du système (Table 7-20).

L'utilisation de la technologie *VST25-UMC25* (0.25µm) ainsi que des outils *Synopsys Design Compiler* (synthèse logique) et *Modelsim* de *Model*

Technology Incorporated (outil de simulation) a permis l'analyse critique de chaque bloc constituant, ce qui a favorisé la proposition et l'étude de méthodes alternatives aux approches communément utilisées.
Une méthode alternative d'initialisation de l'opération de prise de décision (module *BPS*) a été ainsi étudiée. La forte composante séquentielle de l'opération de sélection du chemin le plus probable a induit l'exploration d'un autre principe de sélection plus simple: la désignation d'un chemin quelconque de départ. L'étude a ensuite montré la faisabilité de cette solution, à condition d'une adaptation de la distance δ_p de l'opération de prise de décision. La réalisation du module *TBu* nécessite ainsi un nombre supérieur d'unités de reconstruction d'états, qui provoque l'augmentation de la surface de silicium et du retard algorithmique.

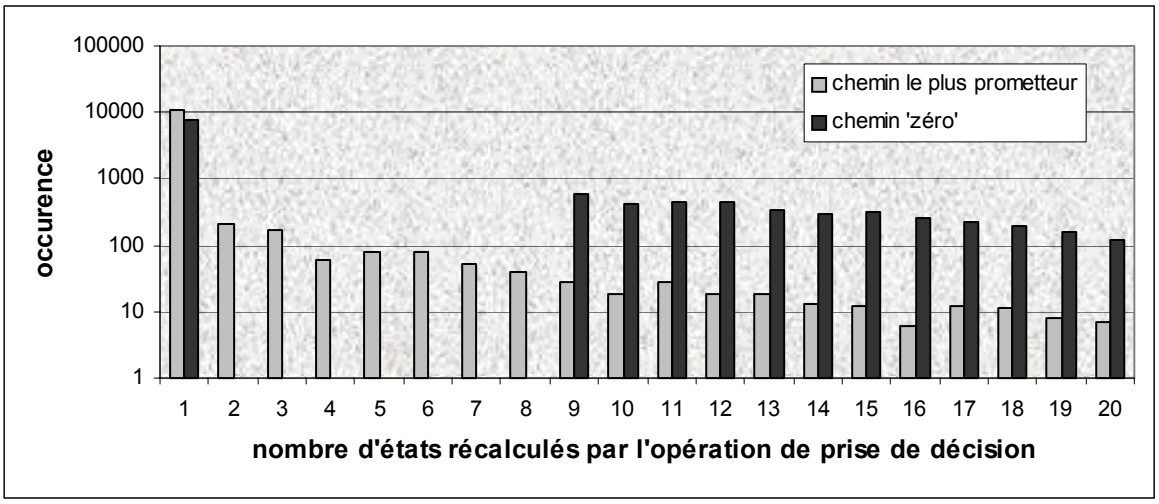

Figure 7-31: nombre d'états qui sont effectivement recalculés lors de l'opération de prise de décision (*TB*). Condition de la simulation: code convolutif proposé par le standard *UMTS* avec rendement R_c=1/3, condition de transmission E_b/N_0=4dB et 11.8k opérations de prise de décision analysées.

Ce chapitre propose ensuite une stratégie innovatrice pour l'exécution de l'opération de prise de décision. Cette stratégie *pipeline* prévoit une exécution parallèle de plusieurs opérations de reconstruction. Le partage de l'opération de prise de décision en δ_p étages de reconstruction augmente le débit de traitement du module *TBu* ainsi que du système entier. Les coûts de cette stratégie sont le doublement des unités de reconstruction d'état, l'augmentation du retard algorithmique et la consommation d'énergie demandée par la gestion des multiples opérations parallèles.
L'analyse de l'opération de prise de décision a successivement montré les avantages d'une stratégie de prise de décision exploitant la redondance des

opérations de reconstruction. Indépendamment de la modalité de sélection, on peut envisager une réduction du nombre d'opérations de reconstruction exécutées, nombre qui passe ainsi de δ_p (reconstruction complète) à des valeurs s'approchant de l'unité (Figure 7-31).

Ce chapitre a prouvé la faisabilité et les potentialités d'une réalisation matérielle de l'algorithme de Viterbi. Cet algorithme possède des caractéristiques importantes: la description de l'algorithme utilise un nombre réduit de paramètres et elle ne nécessite d'aucuns systèmes de contrôle ou de réglage de la qualité de l'opération de décodage (situation d'indépendance algorithmique des conditions de transmission des données).
Grâce à ces caractéristiques de l'algorithmique de Viterbi, les principaux inconvénients et limites de souplesse des réalisation *ASIC* n'interfèrent que marginalement avec la conception du système. CE chapitre a montré aussi qu'une approche coprocesseur augmente la souplesse d'emploi de la réalisation matérielle de l'algorithme de Viterbi.

Références

[BDTi1] Berkeley Design Technology, Inc (BDTi), *The BDTImark2000™: A Measure of DSP Execution Speed*, livre blanc, 1997-2001, page Internet accédée au printemps 2002: www.bdti.com

[BDTi3] Berkeley Design Technology, Inc (BDTi), *Evaluating the Latest DSPs for Portable Communications Applications*, présentation de J. Bier à Aalborg University, Danemark, avril 2001, page Internet accédée au printemps 2002: www.bdti.com

[BDTi4] Berkeley Design Technology, Inc (BDTi), *Alternatives to DSP Processors for Communications Applications*, présentation de J. Bier à Aalborg University, Danemark, avril 2001, page Internet accédée au printemps 2002: www.bdti.com

[BDTi6] Berkeley Design Technology, Inc (BDTi), *The BDTiMark2000: A Summary Measure of Signal Processing Speed*, septembre 2004, page Internet accédée en automne 2004: www.bdti.com

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Fett90] G. Fettweis, *Parallelisierung des Viterbi-Decoders: Algorithmus und VLSI-Architektur*, rapport de recherche (Fortschritt-Berichte), série 10: Informatik/ Kommunikationstechnik, VDI-Verlag, Düsseldorf, 1990.

[FhG01] Fraunhofer Institut Integrierte Schaltungen, *CorePool FHG_VITERBI*, *Databook*, version 1.5, juillet 2001.

[Gath00] A. Gatherer, T. Stetzler, M. McMahan, E. Auslander, "DSP-based Architectures for Mobile Communications: Past, Present and Future", *IEEE Communications Magazine*, janvier 2000, pp.84-90.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grande-Bretagne, 2002.

[Gras01] L. Grasso, *Implementation of a parameterized family of Viterbi Decoders*, rapport final de projet, IMT, Université de Neuchâtel, Neuchâtel, décembre 2001.

[Kapp92] H. Kapp, *CMOS-Schaltungstechnik für integrierte Viterbi-Decoder*, Thèse de Doctorat ès sciences techniques, Technische Universität Berlin, 1992.

[Min91] B.–K. Min, *Architecture VLSI pour le decodeur de Viterbi*, Thèse de Doctorat, Ecole Nationale Supérieure des Télécommunications, Paris, 1991.

[Spra750] Texas Instruments, *Using TMS320C6416 Coprocessors: Viterbi Coprocessor (VCP)*, *Application Report*, document SPRA750, juin 2001.

[Spru312] Texas Instruments, *TMS320C55x DSP Functional Overview*, document SPRU312, juin 2000.

[Spru393] Texas Instruments, *TMS320C55x Technical Overview*, document SPRU393, février 2000.

[Spru533] Texas Instruments, *Viterbi Decoder Coprocessor User's Guide*, document SPRU533, mai 2001.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

8 Comparaison des caractéristiques des deux architectures 'software' et 'hardware'

Les chapitres précédents ont présenté l'aspect du développement d'un système de décodage, en considérant les caractéristiques et les points forts des deux approches, l'une software et l'autre par un circuit supplémentaire ASIC. Ce chapitre se concentre sur les coûts d'emploi de tels systèmes, indépendamment de leurs difficultés de réalisation et du temps nécessaire à leur développement. Les aspects clés sont la vitesse de traitement, la consommation d'énergie et la surface de silicium.

Ce chapitre traite séparément chacun de ces aspects, de manière de mettre en évidence les caractéristiques et les potentialités de chaque approche. L'analyse critique de chaque aspect examine les deux types d'approches software et hardware de manière indépendante et se conclue avec une comparaison de leurs potentialités. La méthode de décodage utilisée pour l'analyse critique est l'algorithme de Viterbi.

Les potentialités des deux approches sont estimées à partir des résultats présentés dans ce document ainsi que par les informations disponibles du marché des DSP et de la technologie de synthèse UMC25-0.25µm [UMC25].

8.1 Introduction

Dans le domaine du traitement numérique du signal et de la télécommunication mobile, la comparaison des coûts d'emploi d'un système prend principalement en compte trois aspects. Ces aspects sont le débit de traitement, l'énergie nécessaire pour le traitement d'un bit d'information et la surface de silicium utilisée par le circuit. Par conséquent, 'le coût total'

d'utilisation d'un système peut être défini par une contribution égale de ces trois aspects principaux [Feld02].

Dans ce chapitre, chacun de ces aspects sera traité séparément, de manière à mieux comprendre les caractéristiques et les potentialités propres à chaque approche.

8.2 Débit du traitement

Le premier aspect traité est le débit du traitement. Cet aspect est fondamental pour situer les champs d'application des deux approches (hardware et software) dans le contexte de la communication mobile. Les standards *UMTS* prévoient un débit de transmission extrêmement variable, qui peut atteindre en effet 144 kbps lors d'une utilisation dans des véhicules en mouvement, 384 kbps pour une utilisation pédestre, et 2 Mbps à l'intérieur des bâtiments.

Afin de permettre une analyse correcte, les comparaisons des deux approches concernent l'implantation et la réalisation matérielle de l'algorithme de Viterbi. Toutefois, les performances de l'implantation software de l'algorithme *List Decoding* seront également estimées.

8.2.1 Approche software

Le débit de traitement des *DSP* est estimé à partir des charges de calcul des implantations et des directives de complexités des opérations de l'*ETSI* (Sous-section 6.3.2).
A l'aide d'estimations successives, le nombre de cycles d'instructions ainsi que le temps d'exécution nécessaire peuvent être évalués, en considérant les caractéristiques et les potentialités propres aux familles *DSP*.

Algorithme de Viterbi

La première étape est la caractérisation et la modélisation de la complexité de calcul de l'algorithme de Viterbi.

L'algorithme de Viterbi peut être modélisé par la complexité de calcul de l'opération de sélection d'un chemin survivant (Table 8-1). La normalisation de la complexité observée par le nombre de répétitions de cette opération permet d'établir l'importance de cette opération dans l'exécution de l'algorithme de Viterbi (Table 8-1).

En comparant cette valeur normalisée avec la complexité de calcul effectivement nécessaire à cette opération de sélection, sa prédominance est évidente: les complexités de calcul des autres tâches (notamment la détermination des métriques de branches, la sélection du chemin le plus prometteur et l'opération de prise de décision) ne contribuent que de manière minoritaire (moins de 20%) à la complexité totale du calcul. Contrairement à l'approche hardware, la contribution de la tâche de prise de décision se limite à 10%. Ceci est dû à la simplicité de l'exécution de cette tâche qui consiste dans la lecture successive des δ_p états précédents du chemin sélectionné (Sous-section 6.4.2).

	Facteur de poids des opérations (*ITU*/*ETSI*)	Observation de l'exécution de la tâche de décodage		Complexité de calcul de la seule procédure de sélection d'un chemin survivant (code C) (*)
		Nombre moyen d'opérations observées	Complexité moyenne normalisée par le nombre d'opérations de sélection (24'830 opérations)	
	[*WCwOP*/ opération]	[opérations]	[*WCwOP*]	[*WCwOP*]
Additions	1	49'670	2.0	2.0
Incrémentations	1	1'435	0.1	0.0
Soustractions	1	293	0.0	0.0
Transf. de données	1	182'888	7.4	6.0
Logique	1	147'769	6.0	5.5
Décalages	1	78'584	3.2	3.0
Contrôles	2	142'734	11.5	8.0
Point.: assignations	0	729	0.0	0.0
Point.: arithmétique		3'637		
B.: initialisations		26'906		
B.: contrôles		124'270		
B.: incrémentations		97'305		
Grand total		**856230**	**30.2**	**24.5**

Table 8-1: vue d'ensemble de la complexité de calcul de l'algorithme de Viterbi. Contexte: 500 messages utilisés, codage de canal conforme à la classe *B* du service de parole *AMR-NB* à 12.2 kbps, troncation de la mémoire δ=54. (*) Cette procédure implique la mise à jour des deux métriques cumulées, leur comparaison et la sélection du chemin survivant.

Cette méthodologie de modélisation permet une comparaison correcte des performances entre l'approche software et *ASIC*, en limitant la forte influence

du type d'exécution (séquentielle/parallèle) de la tâche de sélection. Cette méthodologie offre en plus un moyen d'estimation du débit de traitement dans d'autres contextes de codage, où, non seulement le nombre de bits du message change, mais aussi les longueurs de contrainte K [49].

Effets de la description souple des méthodes de décodage

Afin de permettre une comparaison équitable avec l'approche hardware, les implications dues à une description souple de l'algorithme doivent être quantifiées.

La stratégie souple de description de l'algorithme permet l'adaptation aux autres contextes de codage par la modification du paramètre de longueur de contrainte K et des fonctions génératrices G_i. Cette souplesse entraîne une augmentation de la charge de calcul, due à la difficulté d'optimisation du code qui en découle.
Une évaluation du facteur d'augmentation est possible en comparant cette complexité avec celle potentiellement réalisable par une implantation idéale de la tâche de sélection. Pour cette dernière, qui représente la limite inférieure de complexité de cette tâche, on considère l'exécution de la séquence d'opérations suivante:

- chargement des anciennes métriques cumulées appartenant aux deux états précédents (deux opérations de *Transferts de données*);
- chargement des actuelles métriques de branches qui sont couplées aux deux transitions traitées (deux opérations de *Transferts de données*);
- mise à jour des deux métriques cumulées (deux opérations d'addition) ;
- sélection du chemin survivant par l'identification de la meilleure métrique cumulée (une opération de *contrôle*);
- stockage des valeurs de métrique cumulée et de l'information nécessaire à la reconstruction du chemin survivant (deux opérations de *Transferts de données*).

Selon les indications des organisations *ITU* et *ETSI* (*Basic Operations*, Table 8-1), la complexité de cette séquence d'opérations est de 10 *weighted operations* (*wOP*). En considérant une contribution minimale des autres tâches[50] estimée à 10% de la complexité totale, la limite inférieure de complexité de l'algorithme de Viterbi peut se situer autour de 11 wOP/état. En

[49] Il faut se rappeler que la longueur de contrainte K du code détermine le nombre d'opérations de sélection du chemin survivant qui sont impliquées dans le traitement de chaque bit d'information.
[50] On considère seulement la contribution de l'opération de prise de décision.

comparant cette limite inférieure avec la complexité de calcul observée, la perte de compétitivité de la description paramétrable de l'algorithme de Viterbi (Sections 6.4 et 6.6) peut être estimée à un facteur 2,5 (Table 8-1).

Algorithme *List Decoding*

Dans ce paragraphe, on caractérise l'algorithme *List Decoding*, ce qui permet une meilleure compréhension de ses potentialités.

De manière analogue à l'algorithme de Viterbi, le premier aspect traité est l'importance de l'opération caractérisant l'algorithme *List Decoding*: la propagation de deux chemins à partir d'un chemin appartenant à la liste des chemins les plus prometteurs. Cette opération implique la détermination des états de mémoire des deux chemins, ainsi que le calcul de leurs métriques cumulées.

	Complexité moyenne observée, normalisée par le nombre total de chemins prolongés			Complexité de calcul de la procédure de propagation (code C) (*)
	List Decoding, L=128 (12'671 propagations)	*List Decoding*, L=64 (6'527 propagations)	*List Decoding*, L=32 (3'423 propagations)	
	[*WCwOP*]	[*WCwOP*]	[*WCwOP*]	[*WCwOP*]
Additions	2.1	2.1	2.2	2.0
Incrémentations	6.5	6.4	6.2	6.0
Soustractions	0.0	0.0	0.0	0.0
Multiplications	0.0	0.0	0.0	0.0
Transferts de données	7.5	7.8	8.3	5.0
Logique	4.3	4.5	4.7	4.0
Décalages	2.1	2.0	2.0	2.0
Contrôles	27.7	28.8	31.0	14.0
Grand total	**50.2**	**51.6**	**54.4**	**33.0**

Table 8-2: vue d'ensemble de la complexité de calcul de l'algorithme *List Decoding*. Contexte: 500 messages utilisés, codage de canal conforme à la classe *B* du service de parole *AMR-NB* à 12.2kbps, sans procédure de troncation de la mémoire. (*) cette procédure implique la détermination de deux nouveaux états et le calcul des métriques cumulées relatives.

L'analyse de la charge de calcul (Table 8-2) montre que la prédominance de l'opération fondamentale de propagation est affaiblie par les opérations de

gestion de la liste des *L* chemins les plus prometteurs. Les résultats illustrés par la Table 8-2 montrent un rapport entre la complexité totale normalisée et celle de l'opération de propagation de 3:2.
Contrairement à l'algorithme de Viterbi, la structure algorithmique de la méthode *List Decoding* décompose la procédure de recherche en deux parties distinctes et non-négligeables. La première partie comprend les opérations fondamentales de propagations des chemins. Cette partie est la plus complexe (autour de 2/3 de la complexité totale de calcul, Table 8-2) et peut être implantée de manière parallèle. La seconde partie est responsable de la gestion de la liste de chemins ainsi prolongés. L'exécution de cette tâche est caractérisée par une utilisation séquentielle et massive d'opérations de *contrôles* et de *transferts de données*. La complexité de calcul de cette partie demande un tiers de la complexité totale de calcul.

Limite inférieure de complexité de calcul de l'opération de propagation

La détermination de la limite inférieure de complexité de l'opération de propagation est un élément utile pour la caractérisation algorithmique de la méthode. Dans ce cas, l'opération minimaliste de propagation peut être représentée par la séquence d'opérations suivante:

- chargement de l'état de mémoire d'un chemin contenu dans la liste des chemins à prolonger, ainsi que sa métrique cumulée (une opération de *contrôle* et deux opérations de *Transferts de données*);
- détermination des valeurs des deux nouveaux états, qui sont atteints par la propagation du chemin concerné (deux opérations de *Transferts de données*);
- chargement des métriques de branches, qui sont couplées aux deux transitions impliquées dans la propagation (deux opérations de *Transferts de données*);
- mise à jour des deux métriques cumulées (deux opérations d'addition);
- stockage des valeurs des métriques cumulées et des informations nécessaires à la reconstruction des chemins (quatre opérations de *Transferts de données*).

La complexité de cette séquence d'opérations est de 14 *wOP*. En considérant la contribution des autres opérations constituant l'algorithme (estimée à 1/3 de la complexité totale), la limite inférieure de complexité de la méthode *List Decoding* peut être ainsi estimée à 21 *wOP/chemin propagé*.
Cette limite permet d'estimer la perte de compétitivité de la description paramétrable de l'algorithme *List Decoding* (Section 6.7) à un facteur 2 (Table 8-2).

Nombre de cycles d'instructions demandés par l'implantation software

En tenant compte de la modélisation des algorithmes, on procède à la seconde étape de l'estimation du débit de traitement. Cette étape implique l'estimation du nombre minimal de cycles d'instructions, qui sont nécessaires au *DSP* pour l'accomplissement de l'opération de décodage.

L'élément crucial de cette estimation est la modélisation du rapport existant entre les opérations définies par la description des algorithmes et le nombre effectif d'instructions. Vu que les complexités (typiques) de chaque opération ont déjà été prises en considération par les facteurs de pondérations de l'*ITU/ETSI*, il reste à définir le degré d'efficacité de la génération du code.

Le degré d'efficacité exprime le rapport entre la complexité de calcul et le nombre de cycles d'instructions effectivement nécessaires au *DSP* (*wMOPS*/*MIPS*). Par conséquent, le degré d'efficacité n'est pas uniquement influencé par les caractéristiques techniques du *DSP*, mais aussi par le type de langage utilisé par le mode de description (langage à haut ou bas niveau), par le style de programmation et par les performances des compilateurs. Pour ces raisons, les évaluations algorithmiques du Chapitre 6 se sont limitées au comptage des divers types d'opérations, afin de ne pas se concentrer sur une situation de développement particulière et/ou une famille *DSP* spécifique.

Contexte de codage	**Estimation**			
	Estimation de la complexité minimale de calcul par opération fondamentale	Limite inférieure de complexité de calcul par bit du message	Degré d'efficacité de la génération du code	Nombre de cycles d'instructions nécessaires au DSP par bit d'information
	[wOP/ opération]	[wOP/bit]	[wMOPS/MIPS]	[MIPS/bit]
$Codeur_{4états}$	11	44	1 :1	$0.044 \cdot 10^{-3}$
$Codeur_{16états}$	11	176	1 :1	$0.18 \cdot 10^{-3}$
$Codeur_{64états}$	11	704	1 :1	$0.70 \cdot 10^{-3}$
$Codeur_{256états}$	11	2816	1 :1	$2.8 \cdot 10^{-3}$

Table 8-3: exemple d'estimation du nombre d'instructions nécessaire pour l'exécution de l'algorithme de Viterbi. Les valeurs sont obtenues à partir de l'estimation de la complexité minimale de calcul par opération fondamentale et du degré d'efficacité empirique de 1:1 *wMOPS/MIPS*.

Malgré la difficulté à définir le degré d'efficacité sans spécifier un environnement de développement, un rapport de conversion *wMOPS*/*MIPS* de 1:1 a été choisi pour les comparaisons avec l'approche *ASIC*. En utilisant ce rapport, qui représente une description et une conversion optimale du code software en code binaire, la Table 8-3 montre les estimations du nombre de cycles d'instructions nécessaires à l'exécution de l'algorithme de Viterbi, dans les quatre contextes de codage (Sous-section 7.2.2).

Comparaison des deux algorithmes: *Viterbi vs. List Decoding*

La modélisation des deux algorithmes permet la comparaison des exigences de calcul, indépendamment du contexte de codage et de la qualité de la description de l'algorithme (Table 8-4).

Débit de l'application	Exemple d'application dans le contexte *UMTS*	Contexte de décodage: algorithme utilisé	**Estimation**			
			Estimation de la complexité de calcul par opération fondamentale (*) [wOP/opération]	Limite inférieure de complexité de calcul par bit du message [wOP/bit]	Grade d'efficacité de la génération du code [wMOPS/MIPS]	Nombre de cycles d'instructions pour l'exécution en temps réel
320kbps	Débit maximal par *DPDCH*: 960kbps (codes R_c=1/3)	Viterbi	11	2816	1 :1	900 MIPS
		L.Decoding, *L*= 64	21	1344	1 :1	430 MIPS
12.2 kbps	Service de parole à 12.2kbps	Viterbi	11	2816	1 :1	34 MIPS
		L.Decoding, *L*= 64	21	1344	1 :1	16 MIPS

(*) L'opération fondamentale de l'algorithme de Viterbi est l'opération de sélection du chemin survivant, l'opération de propagation celui de la méthode *List Decoding*.

Table 8-4: exemple d'estimation du nombre d'instructions demandées par les deux algorithmes, dans le contexte du standard *UMTS* [Ts25212]. Les valeurs sont obtenues à partir de l'estimation de la limite inférieure de complexité de calcul par opération fondamentale et d'un degré d'efficacité de la génération du code de 1:1 *wMOPS/MIPS*.

A la différence de l'algorithme de Viterbi, le *List Decoding* contre-balance l'avantage d'un réglage du nombre d'opérations de propagation par l'augmentation de la complexité de calcul de son opération fondamentale. Le

changement du type de recherche (du critère de sélection du chemin survivant à la propagation contrôlée des chemins) implique le doublement de la complexité de calcul nécessaire au traitement de chaque chemin. Dans le contexte de codage *UMTS* (Chapitre 6), la diminution d'un facteur 4 du nombre de chemins traités permet à l'algorithme *List Decoding* de réduire ainsi de moitié les exigences de calcul.

Vitesse d'exécution: les potentialités offertes par le *DSP*

La dernière étape de l'estimation est la conversion du nombre de cycles d'instruction dans le nombre de périodes d'horloge du *DSP*. Cette étape de l'estimation implique la prise en compte des caractéristiques et des potentialités propres à chaque *DSP*.

	Familles *DSP* (hiver 2002-03)				
	StarCore SC140	Texas Instruments TMS320C64x	Texas Instruments TMS320C62x	Texas Instruments TMS320C54x	Analog Devices 219x
Fréquence d'horloge maximale	300 MHz	600 MHz	300 MHz	160 MHz	160 MHz
Nombre d'unités fonctionnelles	4	8	8	1	1
MIPS maximal (déclaré)	1200 MIPS	4'800 MIPS	2'400 MIPS	160 MIPS	160 MIPS
SIMD	2x16-bit	4x8-bit	4x8-bit	-	-
Unités ou instructions facilitant l'exécution de la tâche de décodage	Instructions spécifiques facilitant l'implantation de l'algorithme de Viterbi	TMS320C6416: Coprocesseur dédié à l'exécution de l'algorithme de Viterbi ('VCP')	-	Unité spéciale *'Compare, Select and Store' Unit* ('CCSU')	-

Table 8-5: vue d'ensemble des caractéristiques et des potentialités de certaines familles *DSP*. L'exploitation optimale de la redondance des unités fonctionnelles et l'exploitation des unités spéciales nécessite l'emploi du langage à bas niveau propre au *DSP*. Les données de cette table proviennent des sites Internet des fabricants (situation: hiver 2002-03).

L'exécution des opérations de décodage est influencée par l'architecture interne du *DSP* ainsi que par la redondance des unités fonctionnelles et par la disponibilité d'unités spéciales. Les structures des deux algorithmes permettent en effet une exécution parallèle d'opérations ainsi que la concaténation d'instructions.
Au contraire des bénéfices apportés par les architectures classiques des *DSP* (qui sont considérés à l'aide des fonctions *Basic Operations*), les contributions apportées par la redondance des unités et la disponibilité d'unités spéciales ne sont pas facilement généralisables.

Exemples caractéristiques

Afin de prendre connaissance des potentialités des familles *DSP*, un groupe restreint de *DSP* sont utilisés à titre d'exemple (Table 8-5). Ce groupe de *DSP* est caractérisé par différentes fréquences d'horloge, ainsi que par une variété en termes de redondance et d'offre d'unités spéciales.
L'article de M. Du [DuAz00] traite de l'implantation optimale de l'algorithme de Viterbi exploitant les potentialités de ces *DSP* (Table 8-6). Bien que le contexte de codage considéré dans l'article (code convolutif avec longueur K=5 et rendement R_c=1/2) diffère de ceux traités dans cette étude, les résultats de cet article permettent une prise de connaissance des contributions des moyens spéciaux mis à disposition par certains *DSP* (Table 8-7).

L'efficacité des moyens spéciaux des *DSP* est prouvée en observant les résultats de l'estimation de la Table 8-6, tout en considérant une complexité de calcul minimale de 10 *wOP* pour l'opération de sélection d'un chemin survivant.
La disponibilité d'unités redondantes et de fonctions *SIMD* permet tout d'abord une implantation parallèle des opérations, en tirant profit de la structure régulière et parallèle des algorithmes. On trouve par exemple le *DSP StarCore SC140* et les familles *DSP* de Texas Instruments *TMS320C62x* et *C64x*.
Une réduction du nombre de cycles est également réalisable en disposant d'unités spéciales de traitement (exemple le *DSP StarCore SC140* et la famille *DSP* de Texas Instruments *TMS320C54x*). Toutefois, ces unités s'adressent spécifiquement à l'exécution de l'algorithme de Viterbi: l'implantation d'autres méthodes de décodage ne bénéficie que partiellement de ces moyens spéciaux.

DSP	**Nombre d'unités fonctionnelles**	**Nombre indicatif de périodes d'horloge pour la procédure de sélection d'un seul chemin survivant (*) [DuAz00]**	**Technologie**	**Tension d'alimentation du *core***	**Vitesse maximale de la famille *DSP* [BDTi2]**	**Estimation du temps nécessaire à l'exécution d'une opération de sélection (estimation opérations/s)**
TMS320 C64x	8	(*) 0.88	0.12 µm	1.4 V	600 MHz	1.47 ns (680 MHz)
StarCore SC140	4	(*) 0.63		1.5 V	300 MHz	2.10 ns (470 MHz)
TMS320 C62x	8	(*) 2.39	0.15 µm	1.5 V	300 MHz	7.97 ns (125 MHz)
TMS320 C54x	1	(*) 3.84		1.6 V	160 MHz	24.0 ns (40 MHz)

Table 8-6: vue d'ensemble des potentialités offertes par l'utilisation optimale des ressources structurelles et fonctionnelles de certaines familles *DSP*. (*) Ces valeurs sont indicatives: elles ont été extraites d'un autre contexte de codage (code convolutif avec longueur de contrainte K=5 et rendement R_c=1/2) présenté par l'article de M. Du [DuAz00].

DSP	**Estimation:** Temps nécessaire au traitement d'un bit d'information (estimation bits/s)			
	$Codeur_{4états}$	$Codeur_{16états}$	$Codeur_{64états}$	$Codeur_{256états}$
TMS320C64x	5.9 ns (170 Mbps)	23 ns (43 Mbps)	94 ns (11 Mbps)	375 ns (2.6 Mbps)
StarCore SC140	8.4 ns (120 Mbps)	34 ns (29 Mbps)	134 ns (7.5 Mbps)	538 ns (1.9 Mbps)
TMS320C62x	32 ns (31 Mbps)	127 ns (7.9 Mbps)	510 ns (2.0 Mbps)	2'040 ns (0.5 Mbps)
TMS320C54x	96 ns (10 Mbps)	384 ns (2.6 Mbps)	1536 ns (0.7 Mbps)	6'150 ns (0.2 Mbps)

Table 8-7: estimation des potentialités de traitement qu'on peut obtenir en exploitant les ressources structurelles et fonctionnelles des *DSP*. L'estimation se base sur les informations extraites de l'article de M. Du (Table 8-6) [DuAz00].

La fréquence d'horloge des *DSP* influence ensuite le temps d'exécution des opérations de décodage. Ceci est évident en comparant le temps de traitement estimé du *DSP StarCore SC140* et du *TMS320C64x*: si le premier *DSP* optimise le nombre de périodes en exploitant au mieux tous les moyens structurels et hardwares disponibles, le second atteint de meilleures performances en bénéficiant d'une fréquence d'horloge nettement supérieure.

8.2.2 *Approche ASIC*

Les potentialités de débit de traitement d'une réalisation *ASIC* sont déterminées par les performances de l'unité *ACSu*, qui est chargée de la sélection des chemins survivants.

Contexte de codage	Virtual Silicon Technology VST25, UMC25, 0.25µm				
	Temps d'exécution des unités selon *Synopsys Design Compiler* (après synthèse) (estimation du nombre de bits traité par seconde après routage)(*)				
	Unité *BMu*	Unité *ACSu*	Unité *BPS*: détermination du meilleur chemin	Unité *TBu*:	
				Structure séquentielle, δ_p =45	Structure *pipeline,* **estimation**
$Codeur_{4états}$	0.39 ns (1300 Mbits/s)	2.37 ns (210 Mbits/s)	2.24 ns (220 Mbits/s)	13.08 ns (38 Mbits/s)	0.61 ns (820 Mbits/s)
$Codeur_{16états}$	0.92 ns (540 Mbits/s)	2.57 ns (190 Mbits/s)	5.42 ns (92 Mbits/s)	59.95 ns (8 Mbits/s)	1.16 ns (430 Mbits/s)
$Codeur_{64états}$	1.58 ns (320 Mbits/s)	3.40 ns (150 Mbits/s)	10.60 ns (47 Mbits/s)	88.17 ns (6 Mbits/s)	1.87 ns (270 Mbits/s)
$Codeur_{256états}$	1.81 ns (280 Mbits/s)	4.32 ns (120 Mbits/s)	15.35 ns (33 Mbits/s)	113.57 ns (4 Mbits/s)	2.09 ns (240 Mbits/s)
(*) Perte de compétitivité estimée à un facteur 2					

Table 8-8: vue d'ensemble des valeurs concernant le temps d'exécution des divers modules.

Les performances du circuit de base sont tout d'abord déterminées par la réalisation des unités responsables de la prise de décision (Table 8-8). Bien que l'étude de la Section 7.3 se concentre sur l'exécution séquentielle et continue de cette opération de décision, des solutions envisageant une augmentation du débit de traitement sont réalisables.
Une prise de décision multiple, l'utilisation des structures *pipeline* et les techniques utilisant le décodage d'un chemin survivant quelconque permettent en effet de réduire sensiblement le temps d'exécution des tâches de ces unités *TBu* et *BPS*. Ce faisant, la réalisation de l'unité *ACSu* devient l'élément crucial caractérisant le potentiel de débit de traitement du système.

La réalisation de l'unité *ACSu* du système emploie une stratégie classique:

- L'opération de mise à jour des deux métriques cumulées et la comparaison suivante sont exécutées de manière individuelle, l'une après l'autre, et
- La réalisation physique utilise seulement des blocs de logiques 'standards'.

Une amélioration ultérieure du débit de traitement peut être atteinte par d'autres stratégies de réalisation plus performantes et par la concaténation des opérations de sélection du chemin survivant ([Fett90], [Kapp92] et [Saou02]).

8.2.3 Comparaison des potentialités de débit de traitement des deux approches

Le premier facteur causant une perte importante de compétitivité est la divergence entre la modalité d'exécution (parallèle ou séquentielle) et les caractéristiques algorithmiques de la méthode de décodage. L'utilisation d'une technologie plus performante favorise ensuite la rapidité de traitement.

Les deux méthodes de décodage montrent de fortes composantes parallèles ainsi que séquentielles. En ce qui concerne l'algorithme de Viterbi, l'opération de sélection des $2^{(K-1)}$ chemins survivants montre une structure régulière et parallèle, alors que l'opération de prise de décision demande une exécution séquentielle.
L'analyse du temps nécessaire à l'exécution de l'opération de sélection d'un chemin survivant illustre la perte de compétitivité due à une exécution entièrement séquentielle de cette opération fondamentale (Figure 8-1). La disponibilité d'un traitement parallèle, indépendamment du type d'approche hardware et software, permet l'exploitation du parallélisme implicite dans l'opération. Aux extrêmes, on trouve ainsi l'approche *ASIC* (qui exploite exhaustivement le parallélisme disponible) et la famille de *DSP AD219x* (qui ne dispose ni de traitements redondants, ni d'unités spéciales facilitant l'implantation de l'algorithme). La Figure 8-1 montre aussi la contribution de la technologie sur le temps d'exécution de l'opération (TMSC64x: technologie 0.12, TMSC62x: technologie 0.15, Table 8-6).

Le débit de traitement du système est déterminé par l'exécution des $2^{(K-1)}$ opérations de sélection du chemin survivant ainsi que de l'opération de prise de décision (Figure 8-2).

Grâce à sa nature, l'approche *ASIC* peut exploiter exhaustivement le parallélisme de l'algorithme, réduisant ainsi le temps d'exécution des $2^{(K-1)}$ opérations de sélection du chemin survivant.
Les approches software peuvent réduire la perte de compétitivité d'une exécution séquentielle des opérations en utilisant des *DSP* qui disposent d'unités fonctionnelles redondantes, d'instructions *SIMD*, d'unités de traitement spéciales ainsi que de performances supérieures dues à une technologie plus compétitive. Le prix à payer est le renoncement progressif à une description souple de l'algorithme et à l'utilisation d'un langage à haut niveau.

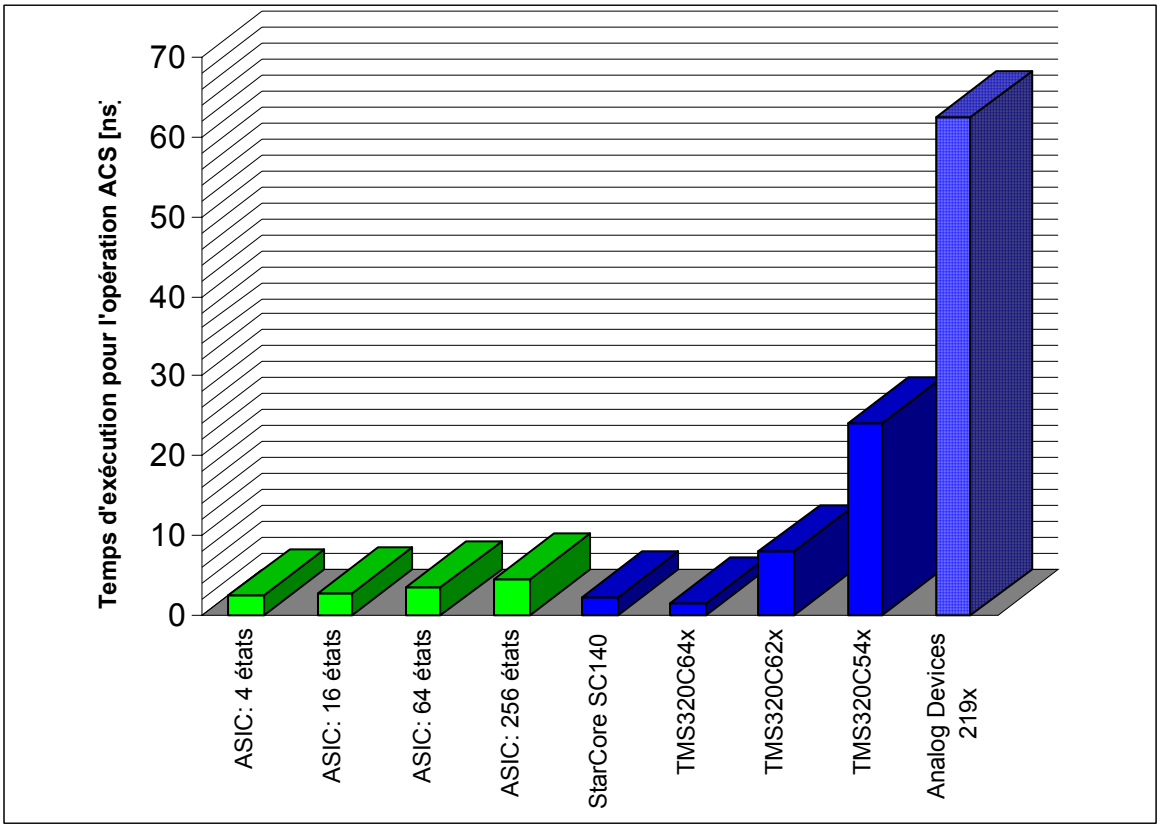

Figure 8-1: comparaison du temps nécessaire à l'exécution d'une tâche de sélection (*ACS*) en fonction de l'approche suivie. En vert sont représentées les réalisations *ASIC* (système de base), en bleu les évaluations d'implantations software sur *DSP* (basées sur les implantations assembleur de l'article [DuAz00], Table 8-6). Le temps d'exécution relatif à la famille de *DSP 219x* de Analog Devices se base sur l'estimation d'une implantation hypothétique de l'opération *ACS* (10 *wOP*, 1:1 *wMOPS/MIPS*), implantation qui ne dispose ni d'unités spéciales, ni de fonctions dédiées facilitant l'implantation de l'algorithme de Viterbi.

La Figure 8-2 met en évidence les limites du système de décodage hardware décrit dans le Chapitre 7. L'exécution séquentielle de l'opération de prise de décision limite le débit de traitement du système entier (Table 8-8), diminuant l'écart de performances avec les implantations software les plus compétitives. Une réalisation différente de cette opération (par exemple une prise de décision multiple ou une stratégie d'exécution *pipeline*) permet à nouveau de bénéficier des importantes potentialités de l'approche matérielle (Figure 8-2).

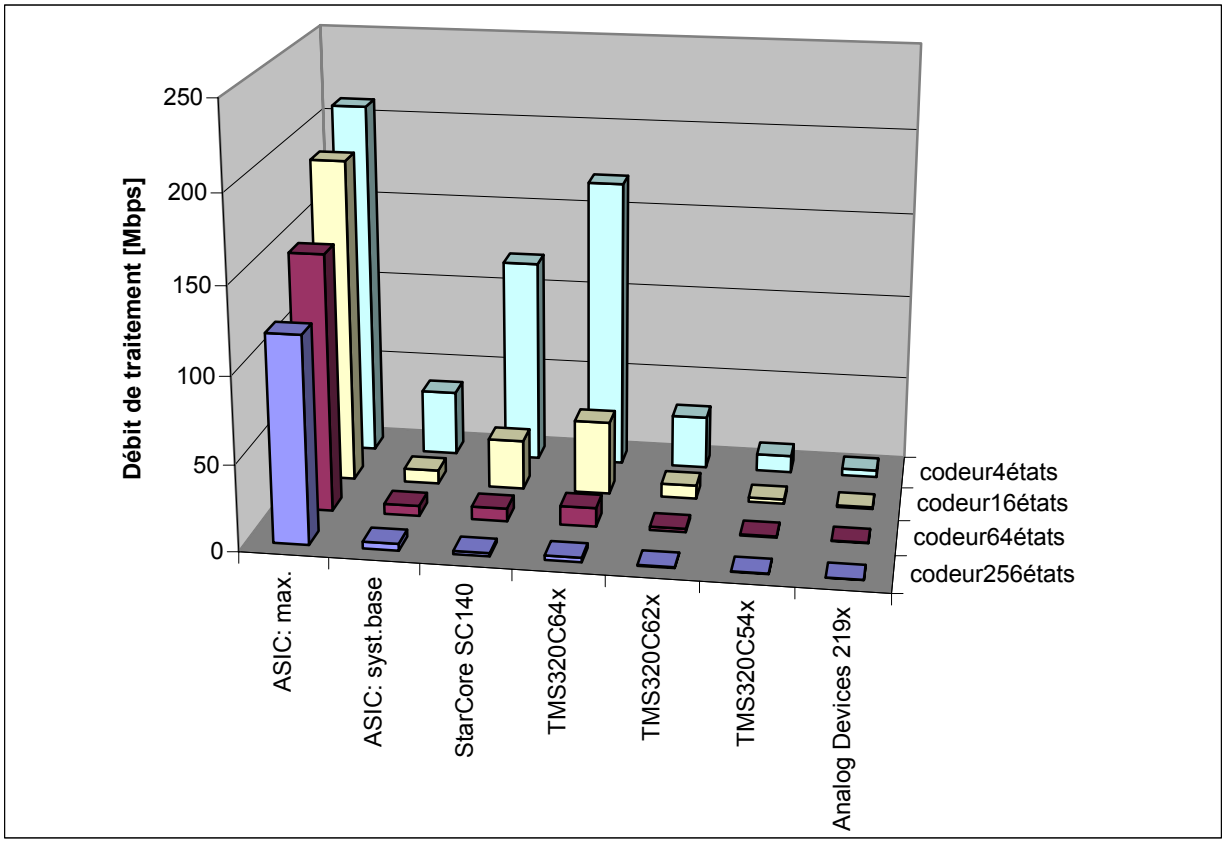

Figure 8-2: vue d'ensemble des potentialités de débit de traitement. Les solutions évaluées sont une réalisation *ASIC* caractérisée seulement par l'opération *ACS* [51] ('max'), le système de base et les implantations software sur *DSP* de l'article de M. Du [DuAz00] (Table 8-8). Les potentialités de la famille de *DSP 219x* de Analog Devices se basent sur l'estimation d'une implantation hypothétique de l'opération *ACS* de 10 *wOP* avec une efficacité de 1:1 *wMOPS/MIPS*.

[51] Solution impliquant, soit une prise de décision multiple de plusieurs bits d'informations (Annexe C, Figure C-8), soit l'exécution *pipeline* de l'opération de prise de décision.

8.3 Dissipation d'énergie par symbole traité

Cette section traite de la dissipation d'énergie. L'estimation des valeurs de dissipation de l'approche software se base toujours sur l'évaluation du nombre de cycles d'instructions, qui sont nécessaires au *DSP* pour accomplir la tâche attribuée. Par contre, les estimations relatives à l'approche hardware utilisent les résultats de synthèse du système de base.

8.3.1 Approche software

Une estimation de la dissipation d'énergie nécessite la connaissance du nombre d'instructions exécutés ainsi que les valeurs de dissipation typique de chaque classe d'instruction du *DSP*.

Par conséquent, deux éléments entrent en jeu dans ce type d'estimation: les instructions du code binaire et les caractéristiques de consommation des opérations du *DSP*. Si le premier élément peut être estimé par le comptage des *Basic Operations*, la caractérisation de la consommation des différentes opérations est difficile. Non seulement il s'agit d'informations sensibles (et donc difficiles à obtenir des fabriquants et aussi difficiles à vérifier), mais elles varient sensiblement de *DSP* en *DSP*.

Évaluation simplifiée

Ne disposant pas de ces informations, la méthode d'évaluation choisie estime le nombre total de cycles d'instructions nécessaires à l'exécution de la tâche, nombre qui est ensuite pondéré par une valeur typique de dissipation d'énergie. Le point critique de cette évaluation est uniquement la détermination de la valeur de dissipation typique par cycle d'instruction. Par contre, la simplification de la méthode d'évaluation rend impossible l'appréciation d'une implantation utilisant des instructions à faible consommation.

Afin d'explorer de manière correcte les potentialités des *DSP*, on analyse la consommation typique des familles de *DSP* à basse consommation. Plus précisément, la consommation d'énergie par instructions *Multiply-Accumulate*

('*MuAcc*')[52] [Gath02] de la classe de *DSP* à basse consommation du fabricant Texas Instruments (Figure 8-3).

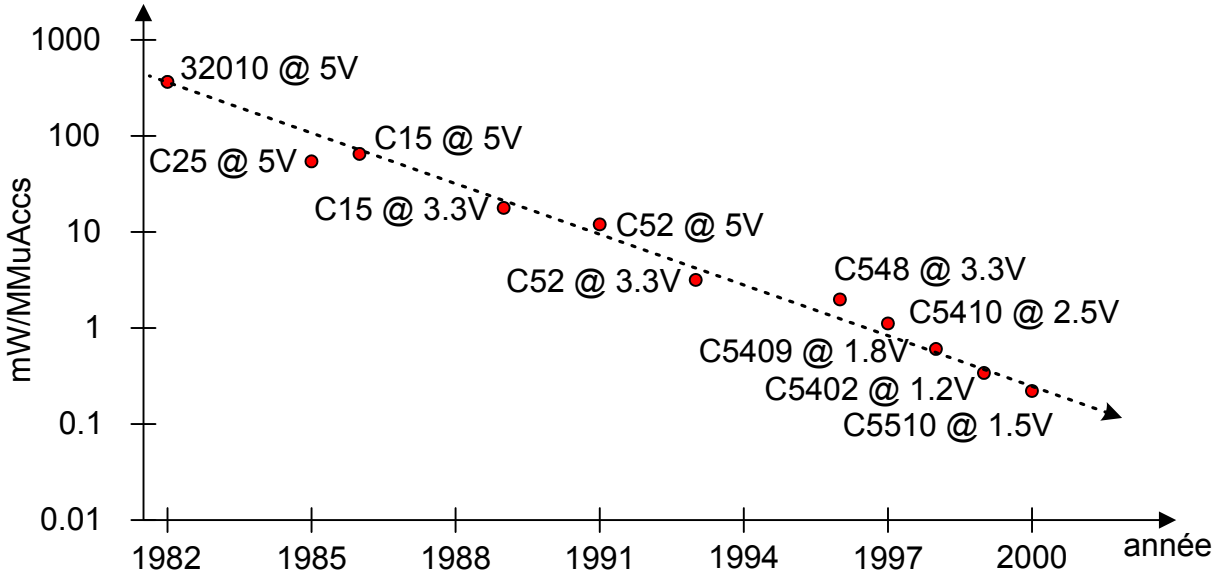

Figure 8-3: tendance à la réduction de la dissipation d'énergie par million d'instructions *MuAccs* de *DSP* à basse consommation de Texas Instruments [Gath02].

L'observation de ces valeurs de dissipation met en évidence la tendance à la réduction de moitié de la consommation d'énergie (par instructions *Multiply-Accumulate*) chaque 18 mois. En supposant ainsi une valeur de dissipation typique par instruction égale à la consommation de l'instruction *Multiply-Accumulate*, une valeur de 0.1mW/MIPS peut être utilisée à titre de référence, indépendamment de la puissance de calcul offerte par cette classe de *DSP* à basse consommation.
L'analyse des Table 8-1 et Table 8-2 montre que les descriptions algorithmiques des algorithmes de Viterbi et de *List Decoding* n'utilisent pas d'opérations coûteuse comme la multiplication. On pourrait ainsi supposer que l'utilisation généralisée de cette valeur de référence puisse mener à une estimation surdimensionnée de la consommation d'énergie. Par conséquent, on prévoit une seconde évaluation utilisant une valeur de référence moins onéreuse.

[52] Une opération *Multiply-Accumulate* consiste en le chargement de deux opérandes depuis la mémoire, l'exécution de la multiplication (*Multiply and Accumulate*, '*MuAcc*') et la sauvegarde successive du résultat dans la mémoire [Gath02].

Modèles d'estimation de la consommation à borne maximale et à borne minimale

La Table 8-9 montre les résultats de deux modèles d'estimation de la dissipation d'énergie par opération de sélection du chemin survivant (algorithme de Viterbi) et de propagation d'un chemin (*List Decoding*). Ces modèles se basent sur l'évaluation du nombre d'instructions exécutées par le *DSP*, nombre qui est ensuite pondéré par l'estimation de la consommation d'énergie typique par instruction.

Classe de *DSP*	Algorithme	Estimation de la complexité de calcul de l'opération fondamentale (*) [wOP/ opération]	Degré d'efficacité de l'implantation [wMOPS/ MIPS]	Dissipation typique par cycle d'instruction exécuté [mW/ MIPS]	Dissipation d'énergie de l'opération fondamentale (*) [nJ/ opération]	Modèle d'estimation de la consommation
DSP à basse consommation conçu pour un large usage	Viterbi	10	1:1	0.1	1.00	'A borne maximale' ('max.')
	List Dec.	14			1.40	
DSP à basse consommation, offrant des instructions *SIMD* et des unités spéciales	Viterbi	10	4:1	0.05	0.13	'A borne minimale' ('min.')
	List Dec.	14			0.18	
(*) L'opération fondamentale de l'algorithme de Viterbi est l'opération de sélection du chemin survivant, l'opération de propagation celui de la méthode *List Decoding*.						

Table 8-9: Estimations des valeurs de dissipation d'énergie pour une implantation software des opérations fondamentales des deux algorithmes: l'opération de sélection du chemin survivant et de propagation.

Le premier modèle d'estimation (dénommé 'à borne maximale') représente le contexte d'une implantation optimale des opérations fondamentales sur un *DSP* classique à basse consommation. Dans ce contexte, le *DSP* ne dispose ni de fonctions *SIMD*, ni d'unités spéciales pour le traitement de l'algorithme de Viterbi. La redondance des unités arithmétiques n'influence pas la consommation totale d'énergie.

Le second modèle d'évaluation prend en compte le contexte d'un *DSP* qui dispose de moyens facilitant l'exécution des opérations fondamentales (fonctions *SIMD* et unités spéciales). Par conséquent, l'estimation du nombre d'instructions a été réduite d'un facteur 4, en supposant que le degré d'efficacité d'implantation s'élève à 4:1 *wMOPS*/*MIPS*. Ce facteur indique l'ordre de grandeur de la réduction réalisable par l'utilisation d'instructions *SIMD* et d'unités spéciales (observation des cas illustrés par la Table 8-6).
En effet, les algorithmes considérés ne comprenant pas d'opérations coûteuses en consommation, il est possible de réduire la valeur de dissipation de référence. La valeur de dissipation est ainsi réduite (arbitrairement) de moitié, en reconnaissant la difficulté d'estimation de la réduction apportée par l'utilisation d'unités spéciales.

Algorithme	**Opération fondamentale (1) (partie parallèle)**			**Partie séquentielle des algorithmes (*)**			**Système entier**	
	Complexité	Dissipation d'énergie		Complexité normalisée par nombre d'opérations fondamentales (1)	Dissipation d'énergie normalisée par nombre d'opérations fondamentales (1)		Dissipation d'énergie normalisée par nombre d'opérations fondamentales (1)	
		Max.	Min. (2)		Max.	Min. (3)	Max.	Min.
Viterbi	10wOP	1.0 nJ	0.13 nJ	1 wOP/ACS	0.10 nJ	0.05 nJ	1.1 nJ	0.18 nJ
List Dec.	14wOP	1.4 nJ	0.18 nJ	7 wOP/prop.	0.70 nJ	0.35 nJ	2.1 nJ	0.53 nJ

(1) L'opération fondamentale de l'algorithme de Viterbi est l'opération de sélection du chemin survivant, l'opération de propagation celui de la méthode *List Decoding*.
(2) 4:1 wMOPS/MIPS, 0.05 mW/MIPS (3) 1:1 wMOPS/MIPS, 0.05 mW/MIPS

Table 8-10: Estimation des valeurs de dissipation d'énergie demandée par les implantations des algorithmes de Viterbi et *List Decoding*. Les paramètres de ces estimations sont illustrés dans la Table 8-9. (*) L'exécution de ces opérations ne peut pas bénéficier de moyens permettant une réduction du nombre d'instructions, tels que des unités spéciales et des fonctions *SIMD*.

A partir de ces modèles d'estimation, la Table 8-10 présente l'estimation de la dissipation totale d'énergie par bit d'information.
Ces estimations mettent en évidence les avantages de l'exploitation des composantes parallèles des algorithmes. Le parallélisme algorithmique permet l'utilisation de fonctions *SIMD*, ce qui réduit le nombre d'instructions nécessaires au traitement d'un bit d'information. Les performances de l'algorithme *List Decoding* sont en effet pénalisées par son importante composante séquentielle. Ensuite, sa différence algorithmique empêche

l'exploitation des unités spéciales, dédiées au traitement de l'algorithme de Viterbi.

Il faut toujours se rappeler que ces estimations (Table 8-10) se basent sur le nombre minimal d'instructions nécessaires à l'exécution des opérations fondamentales et sur le rapport (observé) entre la complexité de calcul de ces opérations et la complexité totale du système (Table 8-1 et Table 8-2).

8.3.2 Approche ASIC

Contrairement à l'approche software, l'estimation de la dissipation d'énergie de l'approche *ASIC* dispose d'informations sur la structure, le nombre, les types et les valeurs de dissipation de tous les éléments logiques et de stockage utilisés.

Contexte de codage: Codeur$_{256états}$ ***Virtual Silicon Technology VST25*, *UMC25*, 0.25µm** Condition de l'analyse: **condition de fonctionnement difficile** (VDD 2.25V, 70˚C), **analyse simple** (*analysis_effort low*) **Cycle d'horloge** de **50 ns** (20 MHz)					
		Estimation de *Synopsys Design Compiler*	Estimation de *Design Compiler* utilisant les résultats des simulations effectuées avec *ModelSim*		
			Sans erreurs	*Erreurs*	*Seulement' 0'*
Dissipation intérieure	*BMu*	0.1 mW	0.1 mW	0.1 mW	0.0 mW
	ACSu	19.0 mW	21.7 mW	22.4 mW	10.5 mW
	BPS	4.9 mW	5.9 mW	5.2 mW	0.1 mW
	TBu	57.8 mW	39.9 mW	36.5 mW	2.1 mW
	Total	**82 mW**	**68 mW**	**64 mW**	**13 mW**
Courant de fuite statique (*Leakage*)					**2.1 mW**

Table 8-11: estimations de la dissipation d'énergie dans le contexte de codage *codeur$_{256}$*. Les conditions et les hypothèses de travail des diverses estimations sont décrites dans la Table 8-12.

Incertitudes: les taux d'activités des blocs et la contribution des interconnexions

Une incertitude affecte (potentiellement) l'estimation de la dissipation d'énergie: la variation des taux d'activité des divers modules. Comme cité

dans le chapitre précédent, les taux d'activité varient en fonction de plusieurs facteurs, tels que le taux d'activité du symbole reçu, les fonctions génératrices, la représentation numérique des métriques de branche et cumulées, la présence d'erreurs de transmission et certains choix techniques (fonctions de sélection, procédure de prise de décision).
Etant donné que certains facteurs sont fixés lors de la conception du circuit, on veut analyser l'influence des facteurs externes sur la dissipation d'énergie. Ainsi, quatre évaluations ont été effectuées afin de situer l'importance des variations de dissipation d'énergie (Table 8-11 et Table 8-12).

Estimation	**Conditions et hypothèses des estimations**
Synopsys	Estimation effectuée par *Synopsys Design Compiler* en l'absence d'information sur les taux d'activité des blocs/signaux. La situation suivante a été considérée: Signal de remise à zéro (*reset*) changeant de valeur chaque 2ème cycles d'horloge (P[1]=0.5, Activité=0.5); Bits du symbole reçu (*in_data*) changeant de valeur chaque 2 ème cycles d'horloge (P[1]=0.5, Activité=0.5).
Estimation effectuée par *Synopsys Design Compiler* en se basant sur les résultats d'une simulation effectuée à l'aide de l'outil *Modelsim* (*Model Technology Incorporated*)	
Sans erreurs	Codage d'un flux de 1'000 bits d'information qui ont été générés de manière aléatoire (P[1]=0.48, Activité=0.47). Absence d'erreurs de transmission. Le signal de remise à zéro n'a été utilisé qu'une fois au début de la simulation.
Erreurs	Génération de symboles d'entrée (*in_data*) de manière aléatoire et indépendante: *premier bit (LSB):* P[1]=0.49, Activité=0.45; *second bit*: P[1]=0.51, Activité=0.44; *troisième bit (MSB):* P[1]=0.49, Activité=0.43. Le signal de remise à zéro n'a été utilisé qu'une fois au début de la simulation.
Seulement '0'	Codage d'un flux de 1'000 bits d'information, bits constitués seulement par des zéros (P[1]=0.0, Activité=0.0). Absence d'erreurs de transmission. Le signal de remise à zéro n'a été utilisé qu'une fois au début de la simulation.

Table 8-12: conditions et hypothèses prises en compte par les estimations de dissipation d'énergie effectuées. Légende: 'P[1]' symbolise la probabilité que le bit soit égal à '1', 'activité' indique le nombre de transitions par cycle d'horloge.

Quatre situations particulières sont considérées (Table 8-12):

- une situation d'absence d'informations sur les signaux d'entrée (évaluation nommée '*Synopsys*');
- une transmission de données en l'absence d'erreurs ('*Sans erreurs*');

- une transmission extrêmement perturbée et dominée par les erreurs de transmission ('*Erreurs*');
- le codage d'une séquence de bits d'information impliquant une invariabilité du symbole de sortie ('*Seulement '0'*').

Il en découle que l'absence d'information sur l'activité des symboles reçus et du signal de remise à zéro a généré une évaluation ('*Synopsys*') plus sévère par rapport aux autres cas simulés (Table 8-11). Par contre, l'influence des erreurs de transmission sur la dissipation d'énergie est faible, l'augmentation de la consommation d'énergie du module *ACSu* étant limitée par rapport à la consommation totale.
Dans la dernière situation considérée (estimation '*Seulement '0'*'), l'activité des modules se réduit au minimum, avec des valeurs de dissipation d'énergie extrêmement basses. Cette situation d'invariabilité des symboles générés est bien entendu réelle, toutefois rare si l'on suppose l'équiprobabilité des messages transmis.

Pour prendre en considération un plus large spectre d'applications, les valeurs de dissipation de l'évaluation plus sévère (évaluation 'Synopsys', Table 8-12) sont utilisées dans la suite du document (Table 8-13).

		***Virtual Silicon Technology VST25*, *UMC25*, 0.25µm** Condition de l'analyse: **condition de fonctionnement difficile** (VDD 2.25V, 70˚C), **analyse simple** (*analysis_effort low*) **Cycle d'horloge** de **50 ns** (20 MHz)			
		Codeur$_{4états}$	**Codeur$_{16états}$**	**Codeur$_{64états}$**	**Codeur$_{256états}$**
Dissipation intérieure	*BMu*	8.1 µW	20.3 µW	55 µW	0.1 mW
	ACSu	243.5 µW	1.1 mW	4.9 mW	19.0 mW
	BPS	23.9 µW	0.2 mW	0.9 mW	4.9 mW
	TBu	570.6 µW	2.2 mW	10.2 mW	57.8 mW
	Total	**846 µW**	**3 mW**	**16 mW**	**82 mW**
Consommation due au courant de fuite statique (*Leakage*)		**21 µW**	**97 µW**	**466 µW**	**2.1 mW**

Table 8-13: dissipations d'énergie des modules selon les divers contextes de codage. Evaluation '*Synopsys*' (Table 8-12).

Actuellement, on dispose de toutes les informations des éléments de conception du système de décodage. Afin d'effectuer une comparaison objective, on doit considérer aussi les effets dus au placement des éléments et

du routage. En ne disposant pas de résultats de routage, les effets des interconnexions sont pris en compte avec une pondération arbitraire par un facteur deux (Table 8-14).

Contexte de codage: Codeur$_{256états}$ ***Virtual Silicon Technology VST25*, *UMC25*, 0.25µm** Condition de l'analyse: **condition de fonctionnement difficile** (VDD 2.25V, 70˚C), analyse simple (*analysis_effort low*)				
Module		**Estimation de *Synopsys Design Compiler*** Cycle d'horloge de 50ns	Dissipation d'énergie par cycle d'horloge	Estimation de la dissipation d'énergie de la réalisation matérielle (*x*2)
Dissipation intérieure	*BMu*	0.1 mW	5.0 pJ	10 pJ
	ACSu	19.0 mW	1.0 nJ	1.9 nJ
	BPS	4.9 mW	0.2 nJ	0.5 nJ
	TBu	57.8 mW	2.9 nJ	5.8 nJ
	Total	**82 mW**	**4.1 nJ**	**8.2 nJ**

Table 8-14: estimation de la consommation d'énergie par cycle d'horloge, à partir des valeurs de dissipation intérieure. Evaluation '*Synopsys*' (conditions: Table 8-12).

Aspect de la faible consommation

Les moyens pour atteindre une faible consommation sont d'abaisser autant que possible la fréquence de travail, la capacité équivalente et, surtout, la tension d'alimentation [Heub97]. Ainsi, en bénéficiant de la relation existante entre la fréquence de travail et la tension d'alimentation, le renoncement (partiel) à une vitesse d'exécution élevée en réduisant la tension d'alimentation est une solution extrêmement rentable en termes d'économie d'énergie.

Cette stratégie permet de réduire efficacement la dissipation d'énergie dynamique par bit: la Table 8-15 montre l'évaluation de l'économie d'énergie obtenue par une réduction de moitié de la tension d'alimentation.
En ce qui concerne sa faisabilité, les débits de traitement montrés par le système de base (Table 8-8) permettent d'envisager concrètement cette solution, surtout en considérant les importantes potentialités apportées par l'approche *pipeline* dans la réalisation du module *TBu*.

	***Virtual Silicon Technology VST25*, *UMC25*, 0.25µm** Condition de l'analyse: **condition de fonctionnement difficile** (VDD 2.25V, 70˚C) **analyse simple** (*analysis_effort low*)	
	Dissipation d'énergie par période	
	Evaluation de *Synopsys Design Compiler* pondérée par un facteur 2 (estimation des effets dus au routage)	Estimation d'une réduction de moitié de la tension d'alimentation (1.12 V)
$Codeur_{4états}$	0.08 nJ	0.02 nJ
$Codeur_{16états}$	0.3 nJ	0.08 nJ
$Codeur_{64états}$	1.6 nJ	0.4 nJ
$Codeur_{256états}$	8.2 nJ	2 nJ

Table 8-15: Estimation de la variation de la dissipation d'énergie lors d'une réduction de la tension d'alimentation. Ces évaluations se basent sur l'hypothèse qu'une réduction de moitié de la tension d'alimentation permet de baisser de la consommation d'un facteur quatre.

8.3.3 Comparaison des dissipations d'énergie des deux approches software et hardware

La comparaison des deux approches montre que le seul moyen que l'approche software possède pour se rendre plus concurrentiel est la génération d'un code binaire optimal. Ceci est possible à l'aide de fonctions *SIMD* et d'unités dédiées au traitement de l'algorithme de Viterbi (Figure 8-4 et Figure 8-5).

Opération fondamentale de sélection du chemin survivant ('*ACS*')

La structure algorithmique de l'algorithme de Viterbi se concentre autour de cette opération particulière. La réalisation matérielle ainsi que l'implantation software de cette opération caractérisent les systèmes correspondants.

L'opération de sélection du chemin survivant est simple autant que particulière: il s'agit en fait d'une double addition, suivie par une comparaison des valeurs obtenues. L'implantation software se trouve ainsi dans une situation défavorable, en raison d'une discordance entre la particularité de cette opération et le mode d'exécution classique des *DSP*. Cette situation implique un nombre supérieur d'instructions qui, dans le contexte d'une opération simple, entraîne une perte de compétitivité et une dissipation supérieure d'énergie.

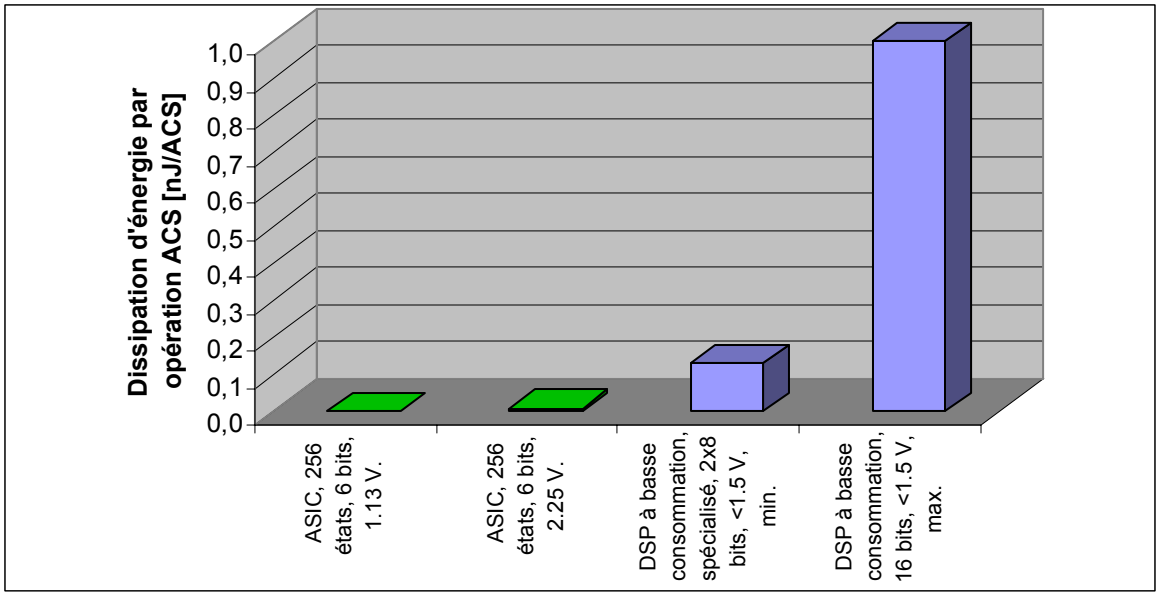

Figure 8-4: comparaison des valeurs estimées de dissipation d'énergie des approches software (Table 8-10) et *ASIC* (Table 8-14). L'opération analysée est l'opération *ACS* de l'algorithme de Viterbi. Les informations de ces opérations indiquent l'approche, la situation considérée, le nombre de bits assignés à la représentation des métriques cumulées, la tension d'alimentation et le type d'estimation.

Consommation d'énergie du décodeur de Viterbi

L'algorithme de Viterbi est caractérisé par une séries d'opérations indépendantes de sélection des chemins survivants suivies par une opération itérative de prise de décision. En termes de débit de traitement et de dissipation d'énergie, un circuit programmable à large usage est évidemment moins performant à un circuit adapté aux particularités de ce type d'algorithme.
Sur le plan de la réduction de la consommation d'énergie, cette perte initiale de compétitivité des approches softwares peut être partiellement comblée tout d'abord par la disponibilité (et le constant renouvellement) de *DSP* à basse consommation réalisés avec des technologies toujours plus avancées. Ensuite, les unités spéciales de traitement et les instructions *SIMD* permettent une exécution plus efficace des opérations particulières et répétitives constituant l'algorithme de Viterbi (Figure 8-4).

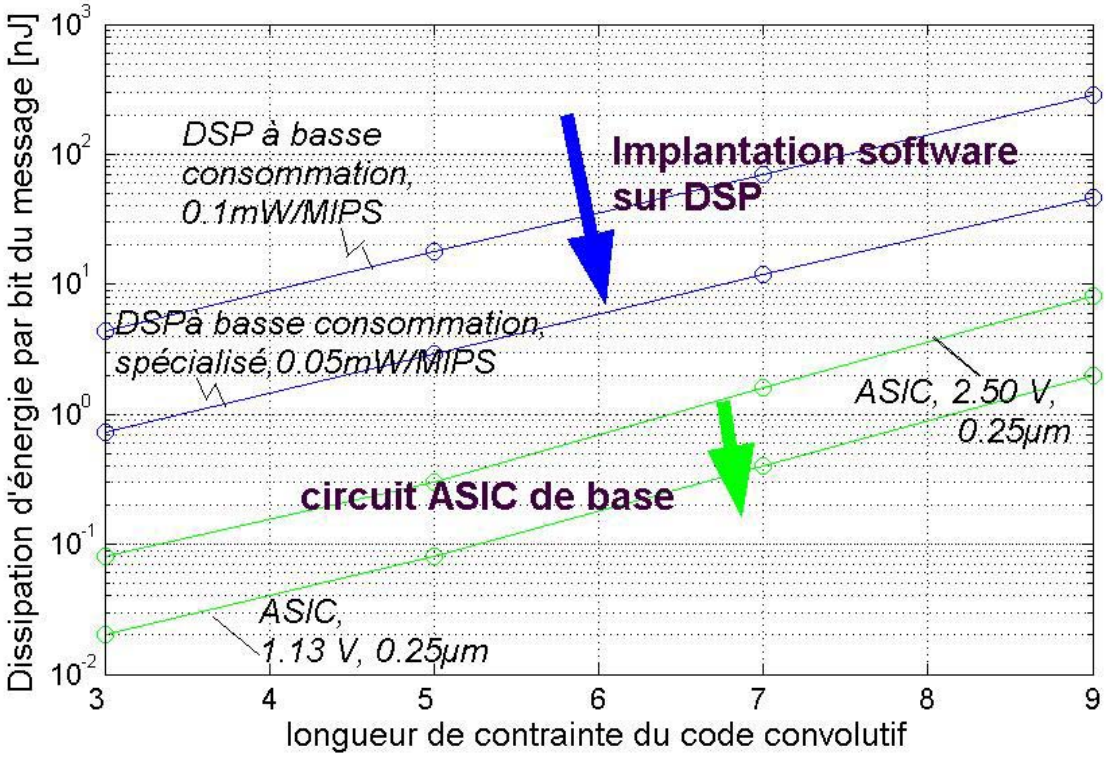

Figure 8-5: tendance de la dissipation d'énergie de l'algorithme de Viterbi.

Il est ainsi évident que, grâce à une adaptabilité optimale aux caractéristiques particulières de l'algorithme de Viterbi, l'approche *ASIC* offre toujours les meilleures potentialités en termes de réduction de consommation (Figure 8-5). Cette caractéristique est surtout appréciée (et dans certains cas indispensable) dans le contexte de la communication mobile (Table 8-16).

Accumulateur	Tension [V]	Capacité typique [Ah]	**Estimation du nombre de bits potentiellement traitables, en disposant de manière exclusive de toute l'énergie** [bits]		
			Approche *DSP*: consommation 'Max'	Approche *DSP*: consommation 'Min'	Approche *ASIC*
Ni-MH type AA	1.2	1.5	$0.02 \cdot 10^{12}$	$0.1 \cdot 10^{12}$	$3 \cdot 10^{12}$
Ni-MH type AAA	1.2	0.7	$0.01 \cdot 10^{12}$	$0.7 \cdot 10^{12}$	$1 \cdot 10^{12}$

Table 8-16: potentialités et performances indicatives des deux approches, en considérant le contexte de codage de la communication mobile *UMTS*. Les valeurs de consommation ont été extraites de la Table 8-10 et la Table 8-15).

8.4 Surface de silicium

Le dernier aspect traité est la surface de silicium des circuits, qui a une incidence sur les coûts de fabrication et sur la facilité d'incorporer la fonction dans des systèmes intégrés (System On Chip, '*SoC*').
L'objectif principal de cette analyse est la comparaison de la surface de silicium nécessaire à la réalisation d'un circuit dédié, qui exploite exhaustivement le parallélisme de l'algorithme de Viterbi, avec la surface d'un processeur versatile et générique, qui montre un degré de parallélisme réduit et limité au strict nécessaire.

Fabriquant	***DSP***	**Normalement disponible comme (*chip*/*core*)**
Agere Systems	*DSP16xxx*	*chip*
Analog Devices	*ADSP-21xx, ADSP-219x, ADSP-2153x, ADSP-TS101S*	*chip*
ARM	*ARM7, ARM9, ARM9E*	*core*
3DSP	*SP-5*	*core*
DSP Group	*OakDSPCore, TeakLite, Teak, PalmDSPCore*	*core*
Hitachi	*SH-DSP, SH3-DSP*	*chip*
Hitachi/ STMicroelectronics	*SH4/ST40, SH5/ST50*	*chip* et *core*
Infineon	*TriCore 1*	*chip* et *core*
	Carmel 10xx	*core*
LSI Logic	*LSI140xX*	*chip* et *core*
Motorola	*DSP563xx, DSP568xx,DSP5685x*	*chip*
StarCore	*MSC8101, StarPro2000*	*chip*
Texas Instruments	*TMS320C2xx, TMS320C28xx, TMS320C54xx, TMS320C55xx, TMS320C62xx, TMS320C64xx*	*chip*

Table 8-17: processeurs à virgule fixe livrables en 2002 [BDTi2].

8.4.1 Processeurs DSP

La caractérisation de la surface typique des *DSP* est d'abord rendue difficile par la non-homogénéité des nombreux *DSP* présents dans le marché (Table 8-17). Cette non-homogénéité est causée principalement par la diversité des approches architecturales, la taille de la mémoire intégrée, les technologies de réalisation physique, les outils mis à disposition et le nombre d'unités fonctionnelles redondantes du *DSP*.
L'opération de caractérisation est ensuite entravée par la disponibilité réduite d'informations facilement accessibles sur les valeurs de surface de silicium.

Cœur synthétisable	Technologie	Surface	Fréquence d'horloge	Performances	Notes
ARM7EJ-S	0.18 µm	0.95 mm^2	80-110 MHz	1.0 MIPS/MHz	Basse consommation, surface réduite et extension instructions *DSP*
	0.13 µm	0.42 mm^2	100-133 MHz		
ARM9E-S	0.25 µm	2.7 mm^2	160 MHz	1.1 MIPS/MHz	Extensions traitement du signal, unité *MAC* de16x32 (cycle unique) et pipeline avec 5 étages
	0.18 µm	-	200+ MHz		

Table 8-18: caractéristiques de deux cœurs synthétisables de *ARM* [ARM7] [ARM9]. Ces cœurs sont des processeurs-*DSP* de type *RISC* possédant une structure à 32 bits.

Par conséquent, en raison de leur simplicité architecturale et de la richesse en informations, les processeurs *DSP* de *ARM* (Table 8-18) sont utilisés à titre de référence pour cette analyse.
La famille de processeurs *ARM7* comprend les cœurs *RISC* à 32 bits, destinés à des applications basse consommation et à faible coût. Le cœur synthétisable *ARM7EJ-S* met à disposition toutes les potentialités d'un microcontrôleur et d'un *DSP* [ARM7].
Le cœur *ARM9E-S* offre les capacités d'un *DSP* exploitant un cœur de processeur *RISC* à 32 bits. Ses arguments de vente sont la surface réduite de silicium, la simplicité architecturale, la basse consommation et la rapidité de réalisation des solutions (*time-to-market*) [ARM9].

8.4.2 Système de base ASIC

Dans le contexte d'une exploitation exhaustive du parallélisme de l'algorithme de Viterbi, la surface de silicium du système dépend évidemment de la longueur de contrainte *K* du code convolutif (Table 8-19).

La surface de silicium croît parallèlement à la redondance des unités fonctionnelles qui est fonction de la longueur de contrainte *K* du code convolutif envisagé (Table 8-19). Les deux modules les plus exigeants sont: celui responsable de la sélection parallèle des chemins survivants (*ACSu*) et celui chargé de la prise de décision du bit d'information (*TBu*).

Virtual Silicon Technology VST25, UMC25, 0.25µm				
Blocs	**Surface en mm^2 (après synthèse)**			
	Contexte de codage:			
	$Codeur_{4états}$	$Codeur_{16états}$	$Codeur_{64états}$	$Codeur_{256états}$
BMu, combinatoire	$3.6\cdot10^{-4}$	$6.2\cdot10^{-4}$	$1.0\cdot10^{-3}$	$2.6\cdot10^{-3}$
ACSu, combinatoire	$6.5\cdot10^{-3}$	$2.5\cdot10^{-2}$	$1.2\cdot10^{-1}$	$4.9\cdot10^{-1}$
ACSu, non- combinatoire	$3.5\cdot10^{-3}$	$1.4\cdot10^{-2}$	$6.7\cdot10^{-2}$	$2.7\cdot10^{-1}$
BPS, combinatoire	$2.3\cdot10^{-3}$	$1.3\cdot10^{-2}$	$6.8\cdot10^{-2}$	$3.0\cdot10^{-1}$
TBu, combinatoire	$4.6\cdot10^{-3}$	$2.6\cdot10^{-2}$	$1.2\cdot10^{-1}$	$4.9\cdot10^{-1}$
TBu, non- combinatoire	$1.7\cdot10^{-2}$	$8.1\cdot10^{-2}$	$4.1\cdot10^{-1}$	2.0
Total	$\mathbf{3.4\cdot10^{-2}}$	$\mathbf{1.7\cdot10^{-1}}$	$\mathbf{7.9\cdot10^{-1}}$	**3.6**

Table 8-19: vue d'ensemble des valeurs de surface de chaque bloc appartenant au système de base développé.

8.4.3 Comparaison des demandes en surface de silicium des deux approches software et hardware

En comparant les valeurs de surface de silicium, il est évident le coût supplémentaire en silicium dû à l'exploitation exhaustive du parallélisme de l'algorithme de Viterbi (Table 8-19). La stratégie d'une exécution séquentielle des opérations ainsi que la souplesse d'emploi des deux cœurs *ARM* permettent d'obtenir un rapport optimal entre la surface de silicium et la gamme de fonctions disponibles.

Dans le contexte de codage *UMTS*, l'exploitation exhaustive du parallélisme algorithmique entraîne des forts coûts en termes de surface de silicium, en raison de l'importante longueur de contrainte des codes convolutifs (K=9). La réalisation matérielle nécessite ainsi plusieurs mm^2 de silicium pour intégrer l'important nombre d'unités fonctionnelles: 256 unités 'mise à jour-comparaison-sélection' et 45 blocs de sauvegardes-reconstruction d'états du chemin.
Il faut souligner que la réalisation physique du système de base n'était pas soumise à des contraintes relatives à la surface de silicium. A titre d'exemple, la conception du module *TBu* a utilisé systématiquement des bascules pour la sauvegarde des informations nécessaires à la reconstruction des chemins. En raison de la typologie de cette tâche, une réduction de la surface de silicium peut être obtenue en remplaçant les bascules par des verrous (*Latches*).

	Technologie	$Codeur_{4états}$		$Codeur_{16états}$		$Codeur_{64états}$		$Codeur_{256états}$	
		Surface [mm^2]	Débit de traitem. [Mbits/s]	Surface [mm^2]	Débit de traitem. [Mbits/s]	Surface [mm^2]	Débit de traitem. [Mbits/s]	Surface [mm^2]	Débit de traitem. [Mbits/s]
Processeur-*DSP* ARM7EJ-S	0.18µm	$9.5 \cdot 10^{-1}$	2.5	$9.5 \cdot 10^{-1}$	$6.1 \cdot 10^{-1}$	$9.5 \cdot 10^{-1}$	$1.6 \cdot 10^{-1}$	$9.5 \cdot 10^{-1}$	$3.9 \cdot 10^{-2}$
Processeur-*DSP* ARM9E-S	0.25 µm	2.7	4.0	2.7	$9.7 \cdot 10^{-1}$	2.7	$2.5 \cdot 10^{-1}$	2.7	$6.3 \cdot 10^{-2}$
Système de base (δ_p=5·K)	*UMC25* 0.25µm **(synthèse)**	$3.4 \cdot 10^{-2}$	76	$1.7 \cdot 10^{-1}$	17	$7.9 \cdot 10^{-1}$	11	3.6	8.8
	Estimation après routage	$7 \cdot 10^{-2}$	38	$3 \cdot 10^{-1}$	8	2	6	7	4

Table 8-20: comparaison des valeurs de surface entre deux processeurs-*DSP* de type *RISC* et les quatre décodeurs utilisés pour l'étude (Table 8-18, Table 8-3 et Table 8-19). La comparaison tient aussi compte des potentialités de débit de traitement. Le système de base est caractérisé par les valeurs après synthèse: les potentialités après routage sont estimées par une pondération par un facteur deux.

8.5 Conclusions: DSP ou ASIC ?

Ce chapitre a permis de situer les coûts d'emploi d'un circuit *ASIC* et d'une implantation software sur *DSP* pour l'exécution de l'algorithme de Viterbi. Les aspects traités ont été le débit du traitement, l'énergie nécessaire pour le traitement de chaque bit d'information et la surface de silicium des circuits. Les potentialités des deux approches ont été jugées à partir des résultats et de l'expérience mûrie avec les implantations et la réalisation matérielle décrite dans les chapitres précédents.
Si l'implantation software offre une solution rapide, souple et bon marché, la réalisation matérielle offre des potentialités de vitesse d'exécution et d'économie de consommation incomparables (Table 8-21). Les coûts en termes de surface de silicium dépendent de la redondance des unités de traitement incluse dans les circuits *DSP* et *ASIC*.

	Débit de traitement		Consommation		Surface	
	DSP TI320C64x(*)	*ASIC* (**) *pipeline*(***)	*DSP "min"*	*ASIC* (**) *2.25V* (****)	*DSP ARM7EJ-S*	*ASIC* (**) *Syst.base*
$Codeur_{4états}$	170 Mbps	210 Mbps	0.72 nJ/bit	0.08 nJ/bit	0.95 mm^2	$3.4 \cdot 10^{-2}$ mm^2
$Codeur_{16états}$	43 Mbps	190 Mbps	2.88 nJ/bit	0.3 nJ/bit	0.95 mm^2	$1.7 \cdot 10^{-1}$ mm^2
$Codeur_{64états}$	11 Mbps	150 Mbps	11.52 nJ/bit	1.6 nJ/bit	0.95 mm^2	$7.9 \cdot 10^{-1}$ mm^2
$Codeur_{256états}$	2.6 Mbps	120 Mbps	46.08 nJ/bit	8.2 nJ/bit	0.95 mm^2	3.6 mm^2

(*) estimation à partir des résultats de [DuAz00]
(**) valeur après synthèse pondérée par un facteur 2
(***) estimation de la stratégie pipeline avec décodage d'un chemin quelconque
(****) estimation de *Synopsys Design Compiler* (VDD 2.25V, 70ºC, analyse simple)

Table 8-21: vue d'ensemble des potentialités des approches software et hardware. Les potentialités sont estimées séparément l'une des autres (Table 8-7, Table 8-8, Table 8-10, Table 8-13, Table 8-15, Table 8-18 et Table 8-19).

Algorithme de Viterbi

Cet algorithme est caractérisé par la répétition massive d'une opération simple autant que particulière (la sélection du chemin survivant), suivie par une opération demandant une exécution séquentielle (prise de décision). En raison d'une configuration de travail impliquant l'exécution répétitive[53] de cette dernière opération, l'algorithme montre de fortes composantes parallèles ainsi que séquentielles.

Implantation software sur *DSP*

Les principales limites de l'implantation software sont la nécessité de disposer d'instructions supplémentaires pour l'exécution des tâches (liées aux transferts des données), le manque d'adaptabilité des tailles des variables et la redondance limitée des unités fonctionnelles.

L'implantation de l'opération de sélection du chemin survivant est tout de suite grevée par la nécessité d'instructions supplémentaires. Ce handicap initial est ensuite amplifié par la répétition massive de cette opération fondamentale de l'algorithme de Viterbi. Cette perte de compétitivité ne peut être comblée que par l'utilisation de solutions et de technologies plus récentes: la disponibilité d'unités fonctionnelles redondantes, de fonctions *SIMD* et de

[53] Dans le contexte de décodage établi, l'opération de prise de décision implique, à chaque niveau de profondeur, la prise de décision sur un bit d'information.

blocs spéciaux de traitement permet l'amélioration des potentialités de débit de traitement et de consommation d'énergie. Le prix à payer est toutefois l'utilisation d'un *DSP* possédant une plus grande surface de silicium, ainsi que le renoncement graduel à une description souple, à haut niveau et portable de l'algorithme de Viterbi.

La forte composante séquentielle de l'opération de prise de décision convient de manière optimale au mode d'exécution classique des *DSP*. L'exécution répétitive de cette opération ne dégrade pas les potentialités de l'approche software.

Approche *ASIC*

La réalisation physique convient à l'exécution de l'algorithme de Viterbi, grâce à la possibilité d'exploiter exhaustivement les caractéristiques particulières de l'algorithme.

La réalisation de l'opération de sélection du chemin survivant se révèle être encore une fois l'élément crucial, en raison de sa structure simple mais particulière. L'approche *ASIC* permet une réalisation adaptée à la singularité algorithmique de l'opération, en exploitant exhaustivement le parallélisme disponible. L'avantage d'un traitement parallèle de l'algorithme est évident en analysant l'opération de sélection des 2^{k-1} chemins survivants. Les potentialités en termes d'épargne d'énergie et de vitesse d'exécution de cette approche sont ainsi excellentes.
Par contre, l'augmentation de la redondance des unités de traitement et la sauvegarde des informations de reconstruction augmentent les coûts en termes de surface de silicium.
L'importante composante séquentielle de l'opération de prise de décision cause ensuite une perte de compétitivité du système de base *ASIC*. L'exécution séquentielle de cette opération à chaque cycle d'horloge dégrade les performances de débit de traitement du système de base. Les contre-mesures envisageables sont l'adoption de la stratégie *pipeline* ainsi qu'une prise de décision multiple par intervalles (Annexe *C*, Figure C-8).

Domaines d'utilisation complémentaires des deux approches software et hardware

Du point de vue du codage de canal, le changement du contexte de codage et l'augmentation du débit maximal de transmission des standards *UMTS* génèrent une forte variabilité de la charge de calcul des opérations de décodage.

L'utilisation d'un code convolutif avec une longueur de contrainte de 9 [Ts25212] exige tout d'abord une importante charge de calcul pour le décodage de chaque bit d'information. Par rapport aux technologies *2G*, il faut ainsi considérer un facteur d'augmentation entre quatre (si on prend en compte le *GSM Half Rate Speech* [ETSI909][54]) et seize (par rapport au *GSM Enhanced Full Rate Speech* et *Full Rate Speech* [ETSI909][55]).

Période	**Entreprise**	**Modèle (Fabricant)**	**Nombre d'unités rappelées**
Juillet 2000	KDDI/au	C309H (Hitachi)	138'000
Octobre 2000	KDDI/au	C309T (Toshiba)	59'800
Novembre 2000	J-Phone	J-D03 (Mitsubishi)	180'000
Janvier 2001	NTT DoCoMo DDI Pocket	S0502iWM (Sony) PHS-J80 (Sanyo)	31'000 600'000
Février 2001	NTT DoCoMo Astel	P503i (Matsushita Comm.) AJ-51 (Japan Radio)	230'000 2'000
Mai 2001	NTT DoCoMo KDDI/au J-Phone	S0503i (Sony) C101S (Sony) DP-221 (Mitsubishi)	420'000 126'000 2'730
Juin 2001	Tu-Ka	TH461 (Mitsubishi)	1'814
Juillet 2001	NTT DoCoMo KDDI/au KDDI/au	P503i (Matsushita Comm.) C406S (Sony) Casio	100'000 560'000 52'100

Table 8-22: rappels principaux de téléphones mobiles dus à des défaillances software, hardware ou à l'utilisation de protocoles non entièrement conformes aux réseaux. La période considérée se situe entre juillet 2000 et 2001, période qui est caractérisée par l'élargissement de la gamme d'applications offertes (Japon, [Yasu01]).

La grande variabilité du débit de transmission des standards *UMTS* peut ensuite créer des exigences de décodage contradictoires. A cause du caractère à large usage des standards *UMTS*, on peut ainsi se retrouver facilement dans des situations où, soit le *DSP* est incapable de fournir toute la puissance de calcul nécessaire à l'exécution de l'application en temps réels, soit les bénéfices apportés par un circuit supplémentaire *ASIC* ne sont pas suffisants pour justifier ses coûts supplémentaires.

[54] Ce système de transmission exploite un code convolutif avec longueur de 7.
[55] Ces contextes de codage utilisent un code convolutif avec longueur de contrainte de 5.

En considérant des contextes de codage peu exigeants en termes de ressources, l'approche software permet une implantation rapide d'algorithmes, offrant toutes ses qualités qui ont été très appréciées dans le contexte des technologies *2G*. Ces qualités sont très utiles pour minimiser les coûts supplémentaires dus à la modification et à la correction des défaillances des applications et des protocoles ([Yasu01], Table 8-22).
Le choix de la méthode de décodage dépend principalement des exigences de l'application ainsi que de la disponibilité d'outils et unités spéciales de traitement.
Si les exigences en termes de débit de traitement et d'économie d'énergie augmentent, la solution *ASIC* s'impose graduellement. L'approche hardware s'adapte de manière optimale aux caractéristiques de cet algorithme, comportant une meilleure efficacité d'exécution et une réduction de la consommation d'énergie. Le vaste marché estimé pour la *3G* ainsi que la potentialité d'intégration (System On Chip, '*SoC*') permettent d'atténuer les problématiques liées aux coûts supérieurs de cette solution [BDTi4].

Références

[ARM7] ARM Ltd, *ARM7™ Thumb® Family*, informations sur le produit, page Internet accédée en été 2002: www.arm.com

[ARM9] ARM Ltd, *ARM9E-S™ Thumb® Family*, informations sur le produit, page Internet accédée en été 2002: www.arm.com

[BDTi2] Berkeley Design Technology, Inc (BDTi), *Pocket Guide to DSP Processors and Cores: Currently Available Processors*, mai 2002, page Internet accédée en été 2002: www.bdti.com

[BDTi4] Berkeley Design Technology, Inc (BDTi), *Alternatives to DSP Processors for Communications Applications*, présentation de J. Bier à Aalborg University, Danemark, avril 2001, page Internet accédée au printemps 2002: www.bdti.com

[DuAz00] J. Du, J. Falkowski, A. Aziz, J. Lane, "Implementation of Viterbi Decoding on StarCore SC140 DSP", *International Conference on Signal Processing, Applications & Technology* (*ICSPAT 2000*), 16-19 octobre 2000, Dallas, Etats-Unis d'Amérique.

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Feld02] H. T. Feldkämper, T. Gemmeke, H. Blume, T. G. Noll, "Analysis of reconfigurable and heterogeneous architectures in the

communication domain", *1st IEEE International Conference on Circuits and Systems for Communications* (*ICCSC'02*), 26-28 juin 2002, St-Petersburg, Russie.

[Fett90] G. Fettweis, *Parallelisierung des Viterbi-Decoders: Algorithmus und VLSI-Architektur*, rapport de recherche (Fortschritt-Berichte), série 10: Informatik/ Kommunikationstechnik, VDI-Verlag, Düsseldorf 1990.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grande-Bretagne, 2002.

[Gold95] I. Gold, "The OakDSPCore's Viterbi Accelerator Speeds Up Digital Communications", *DSP & Multimedia Technology*, décembre 1995 - janvier 1996, page Internet de *DSP Group* accédée en été 2002: www.dspg.com/website/technology/article/2.htm

[Heub97] A. Heubi, *De l'algorithme de traitement numérique du signal à sa réalisation à faible consommation par un circuit intégré spécifique*, Thèse de Doctorat ès sciences techniques, Ecole Polytechnique Fédérale de Lausanne, Lausanne 1997.

[Kapp92] H. Kapp, *CMOS-Schaltungstechnik für integrierte Viterbi-Decoder*, Thèse Doctorat ès sciences techniques, Technische Universität Berlin, 1992.

[Saou02] Y. Saouter, C. Berrou, "Fast SUBMAP decoders for duo-binary turbo-codes", *1st IEEE International Conference on Circuits and Systems for Communications* (*ICCSC'02*), 26-28 juin 2002, St-Petersburg, Russie.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[UMC25] Virtual Silicon Technology Inc., *Diplomat-25 Standard Cell Library: 0.25µm UMC Process*, rev. 2.1, décembre 1999.

[Yasu01] H. Yasui, *Recall of Mobile Phones Continue*, article du 17 septembre 2001, page Internet de *NeAsia Online*: www.nikkeibp.asiabiztech.com

9 Conclusions

Suite aux modifications apportées dans le nouveau standard de communication mobile *UMTS*, le sujet de la thèse est l'analyse des réalisations software et hardware de l'opération de décodage des codes convolutifs prévus par ce standard. Cette thèse évalue tout d'abord l'efficacité des méthodes classiques de décodage convolutif de la 2ème Génération dans le nouveau contexte du codage *UMTS* et elle propose ensuite d'autres méthodes améliorées de décodage.

Point de départ

En considérant l'architecture et la méthode de décodage prédominantes dans les technologies de la *2G*, une nouvelle implantation software de l'algorithme de Viterbi a mis en évidence un alourdissement de la charge de calcul assigné au *DSP*. Comme on pouvait s'y attendre, l'agrandissement de la longueur de contrainte des codes augmente la charge de calcul nécessaire au décodage de chaque symbole protégé. L'utilisation de l'algorithme de Viterbi nécessite ainsi l'exécution d'au moins 2.8k *wOP* par bit protégé, contre les 176 *wOP* demandés par les services de la *2G GSM Enhanced Full Rate Speech* et *Full Rate Speech*.

Résultats obtenus

Considérant l'importante complexité de calcul demandée par l'algorithme de Viterbi, cette thèse propose des méthodes améliorées pour réaliser le décodage des codes convolutifs. Ces méthodes se révèlent intéressantes à la fois du point de vue de la complexité de calcul et de la qualité du décodage, que des potentialités d'une réalisation matérielle.

Approche software

Dans le cadre de cette thèse, l'analyse de la structure du codage de canal des standards *UMTS* ainsi que des principes du décodage convolutif a permis de proposer deux méthodes alternatives à la solution classique Viterbi: les méthodes itératives *List Viterbi Algorithm* (nouvelle réalisation) et *List Decoding intégrant la validation CRC* (nouvelle méthode). Le principe des deux méthodes itératives est une exploitation exhaustive des informations

supplémentaires fournies par les bits de parité *CRC* qui ont été ajoutés au message.
La méthode *List Viterbi Algorithm* permet d'améliorer le niveau de protection, pour lequel l'algorithme de Viterbi fait référence. Une nouvelle implantation conforme aux critères d'une exécution en temps réel a prouvé la faisabilité de la méthode dans le contexte du codage *UMTS* ainsi que le bon rapport entre la qualité de protection et la charge de calcul de la méthode.
La méthode *List Decoding intégrant la validation CRC* a été développée dans l'optique d'exploiter les informations supplémentaires du codage *CRC* afin de réduire la charge de calcul. Cette méthode bénéficie d'une nouvelle réalisation plus efficace de l'algorithme *List Decoding*, ce qui permet de réduire efficacement la complexité de calcul de la méthode itérative.

Le principe des deux méthodes étant une exploitation exhaustive du codage en bloc concaténé, l'utilisation de ces méthodes s'étend à tous les domaines de la transmission numérique des données fixe, mobile et satellitaire. Ces deux nouvelles méthodes élargissent ainsi la liste de solutions efficaces à disposition, ce qui permet aux designers de mieux répondre aux exigences d'une plus large gamme d'applications. La limite des implantations software reste l'importante charge de calcul due à la répétition massive, soit de l'opération de sélection du chemin survivant (méthodes basées sur l'algorithme de Viterbi), soit de l'opération de propagation (méthodes basées sur le *List Decoding*).

Approche hardware

Ce travail de thèse a traité ensuite la réalisation matérielle d'un décodeur générique pour des codes convolutifs utilisant l'algorithme de Viterbi.
La réalisation du système de base a exploité exhaustivement le parallélisme et la simplicité algorithmique qui caractérisent la méthode de Viterbi. La décomposition du principe de décodage en ses éléments constituants, ainsi qu'une réalisation paramétrable, ont permis d'analyser en détail la réalisation matérielle de chaque élément. L'utilisation de la technologie *Virtual Silicon Technology VST25-UMC25* (0.25µm) et des outils *Design Compiler* (*Synopsys*) et *Modelsim* (*Model Technology Incorporated*) ont permis une évaluation des potentialités de chaque élément constituant, en plus de favoriser la proposition de méthodes alternatives de réalisation.
Parmi les alternatives proposées, on a analysé le principe de l'utilisation d'un chemin quelconque pour la prise de décision du bit d'information (étude pionnière), la nouvelle approche *pipeline* pour la réalisation de l'opération de prise de décision (nouveau principe), ainsi que l'exploitation de la redondance des opérations de reconstruction d'états (nouveau principe). Les premiers

deux principes améliorent le débit de traitement du système de décodage, alors que le troisième réduit la dissipation d'énergie de l'opération de prise de décision.

L'analyse du système de base a permis de mettre en évidence les bonnes potentialités d'une réalisation matérielle de l'algorithme de Viterbi, telles que les aspects de la rapidité de traitement et de la réduction de la consommation d'énergie. L'approche coprocesseur améliore de la souplesse d'emploi qui est la limite caractéristique des réalisations matérielles. Ceci permet l'élargissement de l'utilisation du système de décodage développé à d'autres applications de traitement du signal utilisant l'algorithme de Viterbi (par exemple applications impliquant des fonctions dites de "maximum de vraisemblance").

Observations finales

L'étude met en évidence l'absence actuelle d'une solution globalement efficace pour les standards *UMTS*, contrairement au cas des implantations software de l'algorithme de Viterbi massivement utilisés dans les technologies de la *2G*. La solution optimale dépend des exigences et des caractéristiques propres à l'application utilisant le réseau mobile *UMTS*.
L'utilisation de dispositifs programmables offre des solutions rapides, souples et bon marché, mais les implantations software se révèlent être non-concurrentielle en termes de débit de traitement et de dissipation d'énergie. La réalisation matérielle convient en effet à l'exécution de méthodes de décodage basées sur l'algorithme de Viterbi, grâce à l'exploitation exhaustive des caractéristiques particulières de l'algorithme. Le vaste marché des applications concernées ainsi que la potentialité d'intégration de ces réalisations matérielles permettent d'atténuer les problématiques liées à ses coûts supérieurs.

Possibles travaux futurs

L'étude des méthodes de décodage itératives peut être élargie aux autres services *UMTS* ainsi qu'à d'autres systèmes de communication numérique fixe, mobile et satellitaire. Cette étude permettra de mieux formuler les avantages apportées par ces deux méthodes itératives.

La réalisation matérielle d'un décodeur de Viterbi exploitant la stratégie pipeline permettra de mesurer la consommation d'énergie ainsi que la validation du débit de traitement offertes par cette solution.

Remerciements

Je tiens tout d'abord à remercier le Professeur Fausto Pellandini pour m'avoir accueilli dans son laboratoire de recherche, pour m'avoir donné les moyens de réaliser ce travail de thèse de doctorat, ainsi que pour ses précieux conseils. Je le remercie également pour les excellentes conditions de travail et l'ambiance stimulante régnant au sein de son laboratoire.

Je remercie le Professeur Pierre-André Farine pour m'avoir accordé sa confiance et pour sa participation dans le jury de ma thèse.

Je suis très reconnaissant à PD Dr Michael Ansorge, co-directeur de thèse, pour m'avoir suivi, conseillé et encouragé pendant toutes ces années à l'IMT. Je le remercie également pour toutes les discussions constructives et pour ses remarques critiques qui m'ont permis de progresser professionnellement.

Merci aux membres du jury de thèse. Je remercie Dr Francesca Vatta pour sa disponibilité, pour sa sympathie et pour son encouragement. Merci au Dr Alexandre Heubi, ancien collègue, pour ses remarques et corrections très constructives.

Je tiens à remercier tous les collègues et anciens collègues de l'IMT, ainsi que l'équipe du service administratif, pour l'agréable ambiance et les conditions de travail optimales créées.

Merci également aux étudiants et stagiaires qui ont contribué au domaine dans le cadre de leurs projets.

Finalement, je remercie mes grands-parents, mes parents, mon frère et son épouse pour leurs encouragements et leurs indispensables soutiens. Je remercie également tous mes amis qui contribuent à égayer les moments de libre.

Un merci particulier au Dr Alain Dufaux, au Dr Giuseppina Biundo-Lotito, à Anne-Lise Kissling-Bah et à Sévérine Capt pour la relecture de cette thèse.

Ce travail de thèse a pu être accompli grâce aux connaissances élaborées dans le cadre du projet CTI 4238 "*Joint Wideband Speech and Channel Coding in Wireless Applications*" réalisé en collaboration avec la compagnie *STMicroelectronics N.V.* et du projet "*Implementation of a parameterized family of Viterbi Decoders*" réalisé avec *DSPfactory S.A*. Je suis très reconnaissant à M. Luigi Grasso pour l'important savoir-faire qu'il a eu l'amabilité de partager lors de la réalisation de ce dernier projet.

Annexes

Annexe A: Méthodes pour le décodage des codes convolutifs

A.1 Décodage séquentiel

La première méthode pour le décodage des séquences protégées par les codes convolutifs a été l'algorithme séquentiel proposé par Wozencraft qui a ensuite été reprise et modifiée par Fano.
Dans cette section, on présente la fonction de métrique adoptée par cette méthode ainsi que sa stratégie de recherche (complément à la description de la Sous-section 5.2.1).

A.1.1 Fonction de métrique

La recherche du chemin le plus probable se base sur une fonction de métrique, qui est chargée de quantifier la ressemblance entre les symboles reçus et ceux générés par le chemin sous analyse.
En considérant la stratégie de recherche, la métrique relative à un chemin constitué de B branches est:

$$PM^{(i)} = \sum_{np=1}^{B} \left(\mu_{np}^{(i)} - C \right). \qquad \text{(A.1)}$$

La contribution de chaque transition ('branche') du chemin i à la métrique cumulée $PM^{(S_i)}$ est en effet constituée de deux parties:

- La première est représentée par la métrique de branche 'μ_{np}'. La valeur indique le niveau de ressemblance entre les symboles générés par la transition et ceux reçus par le décodeur.
- La seconde s'identifie avec la soustraction d'une constante positive 'C' [Proa95]. La soustraction d'une constante C est nécessaire afin de permettre une comparaison équitable entre des chemins atteignant des niveaux de profondeurs différents.

La constante C est choisie de manière à mieux différencier le comportement de la métrique cumulée du chemin correct par rapport à celles des autres chemins. L'effet désiré est d'obtenir une métrique cumulée qui aura tendance à augmenter, en suivant seulement le chemin correct.

A.1.2 Stratégie de recherche Depth-First

En comparant, transition après transition, la métrique cumulée d'un chemin candidat avec un seuil de référence dynamique, l'algorithme de Fano incorpore une procédure de détection des chemins qui pourraient être incorrects [Joha99] [Proa95] [Schl97]. Dans le cas où le chemin candidat serait suspecté d'être incorrect, l'algorithme changera la direction de la recherche, en cherchant un chemin plus prometteur.

La Figure A-1 montre un exemple graphique de ce principe de recherche. Dans cet exemple, l'algorithme suit initialement le chemin correct jusqu'au nœud '*A*'. A cet endroit, l'algorithme est incapable de continuer vers le nœud '*D*', en raison du seuil de référence représenté par la valeur th_A.
La procédure de recherche d'un autre chemin est appelée pour la première fois. Après avoir diminué le seuil de référence à la valeur th_1, puis à la valeur th_2, un chemin réputé 'conforme' a ainsi été trouvé par la procédure. Le chemin choisi amène l'algorithme dans l'arbre jusqu'au nœud '*C*', où le chemin est estimé incorrect. La diminution du seuil de référence jusqu'à valeur th_4 permet de reprendre le chemin correct, en passant par les nœuds '*A*', '*D*' et '*E*' [Schl97].

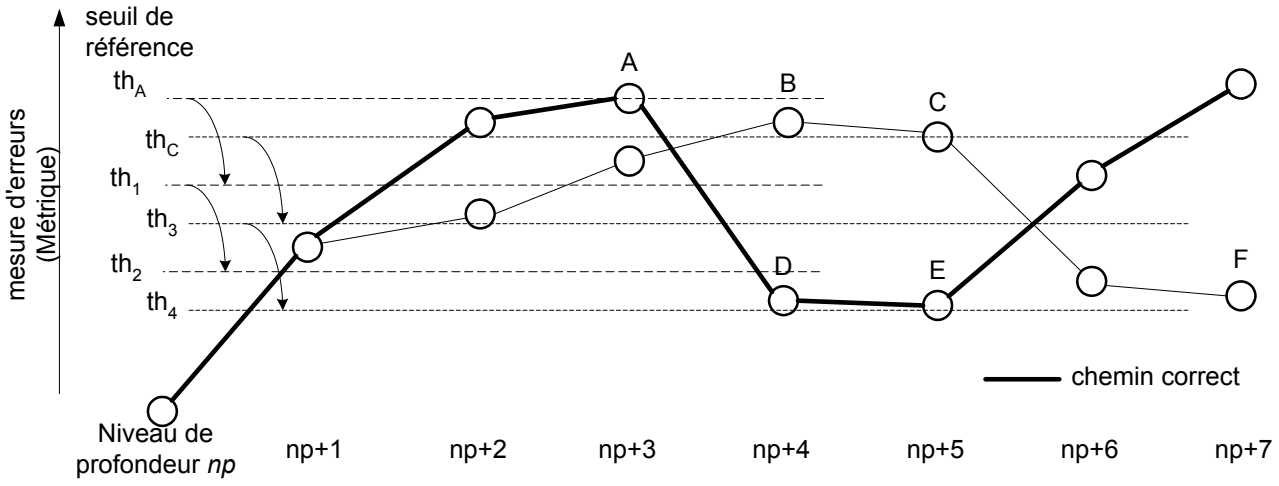

Figure A-1 : exemple de déroulement de l'algorithme de Fano [Schl97].

A.2 Symbol-by-symbol Maximum A Posteriori Algorithm

La méthode appelée *Symbol-by-symbol Maximum A Posteriori* ('*MAP*') a été présentée formellement en 1974 dans la publication [Bahl74], comme une solution alternative pour le décodage de codes convolutifs [Vale98]. Contrairement aux méthodes précédentes, le critère de décodage de cet algorithme est la minimisation de la probabilité d'erreur de chaque symbole formant le message (5.18) [Bahl74] [Proa95] [Robe95] [Schl97] [Vale98] [Vuce00].

A.2.1 Etablissement du problème

En modélisant le problème au moyen d'un processus de Markov (échantillonné dans le temps, avec un nombre fini de niveaux), la méthode permet l'estimation des probabilités des transitions et des états du processus: le décodage de codes convolutifs est une des applications possibles.
Cette sous-section présente tout d'abord le principe général et ensuite les considérations concernant le cas du décodage convolutif.

Comme déjà introduit, cette méthode modélise le problème à résoudre par un processus de Markov. Les transitions entre les états d'un processus de Markov sont réglées par les probabilités de transitions

$$p_{np}\left(m\middle|m'\right)=\Pr\left[s_{np}=m\middle|s_{np-1}=m'\right], \tag{A.2}$$

s_{np}: état considéré au niveau de profondeur np

et les valeurs de la sortie x des transitions (observables) dépendent de la probabilité

$$q_{np}\left(x\middle|m',m\right)=\Pr\left[x_{np}=x\middle|s_{np-1}=m',\ s_{np}=m\right], \tag{A.3}$$

x_{np}: sortie causée par la transition entre les états s_{np-1} et s_{np}

où x appartient à un alphabet fini et discret.

Par rapport à la transmission numérique des données, les effets des distorsions sur le signal transmis causés par un bruit 'sans effet de mémoire' sont évalués à l'aide de la probabilité de transition Tr[.|.] (*Transition probabilities* [Bahl74]):

$$\Pr\left[R\middle|X_1^B\right]=\prod_{np=1}^{B}\mathrm{Tr}\left[r_{np}\middle|x_{np}\right],$$
$$R=R_1^B\left\{r_1,\ ...,r_B\right\}\text{: symboles reçus,} \qquad \text{(A.4)}$$
$$X_1^B=\left\{x_1\ ...,x_B\right\}\text{: symboles transmis.}$$

Selon cette représentation du problème, la méthode *MAP* a pour but l'estimation de la probabilité a posteriori ('*APP*') des états du processus de Markov considéré

$$\Pr\left[s_{np}=m\middle|R\right]=\frac{\Pr\left[s_{np}=m,\ R\right]}{\Pr\left[R\right]} \qquad \text{(A.5)}$$

et de la probabilité *APP* des transitions parvenues entre ces états

$$\Pr\left[s_{np-1}=m',\ s_{np}=m\middle|R\right]=\frac{\Pr\left[s_{np-1}=m',\ s_{np}=m;R\right]}{\Pr\left[R\right]}. \qquad \text{(A.6)}$$

A.2.2 Procédé

Soit une séquence de symboles *R*, la probabilité *Pr[R]* devient ainsi un terme commun aux probabilités (A.5) et (A.6). L'objectif de l'algorithme est ainsi la détermination des deux probabilités suivantes:

$$\lambda_{np}\left(m\right)=\Pr\left[s_{np}=m,\ R\right] \qquad \text{(A.7)}$$
$$\sigma_{np}\left(m',m\right)=\Pr\left[s_{np-1}=m',\ s_{np}=m,\ R\right]. \qquad \text{(A.8)}$$

Ces probabilités peuvent être calculées par l'introduction des trois "nouvelles" probabilités α(.) β(.) γ(.,.) [Bahl74] et [Vuce00]:

$$\lambda_{np}\left(m\right)=\alpha_{np}\left(m\right)\cdot\beta_{np}\left(m\right) \qquad \text{(A.9)}$$
$$\sigma_{np}\left(m',m\right)=\alpha_{np-1}\left(m'\right)\cdot\gamma_{np}\left(m',m\right)\cdot\beta_{np}\left(m\right) \qquad \text{(A.10)}$$

où

$$\alpha_{np}(m) = \Pr\left[s_{np} = m,\ R_1^{np} \right] \tag{A.11}$$

$$\beta_{np}(m) = \Pr\left[R_{np+1}^{B} \middle| s_{np} = m \right] \tag{A.12}$$

$$\gamma_{np}(m', m) = \Pr\left[s_{np} = m,\ r_{np} \middle| s_{np-1} = m' \right]. \tag{A.13}$$

L'algorithme *MAP* permet la détermination des probabilités (A.7) et (A.8) à l'aide de deux probabilités α(.) β(.), qui présentent les propriétés récursives suivantes (Figure A-2 et Figure A-3)

$$\alpha_{np}(m) = \sum_{m'} \alpha_{np-1}(m') \cdot \gamma_{np}(m', m) \tag{A.14}$$

$$\beta_{np}(m) = \sum_{m'} \beta_{np+1}(m') \cdot \gamma_{np+1}(m, m'), \tag{A.15}$$

la probabilité γ(.,.) étant définie par

$$\gamma_{np}(m', m) = \sum_{x} p_{np}(m | m') \cdot q_{np}(x | m', m) \cdot \mathrm{Tr}\left[r_{np} | x \right]. \tag{A.16}$$

A.2.3 Algorithme MAP

Le profil de l'algorithme MAP, ayant pour objectif l'estimation de la fiabilité Λ (5.19) des bits formant le message, est le suivant [Bahl74][Vuce00]:

1. *Initialisation*. Les probabilités $\alpha_0(m)$ et $\beta_B(m)$, où m=0, 1, 2, …, $2^{(k-1)}$-1, sont initialisées selon les conditions de départ et d'arrêt de la procédure de codage.

2. *Forward Recursion*. Après l'arrivée des symboles r_{np} qui ont été générés par le codage du *np*-ème bit d'information:
   a) La probabilité $\gamma_{np}(m',m)$ (A.16) est calculée, en se basant sur la mesure d'affinité entre les symboles reçus et ceux générés par la transition entre les états m' et m.
   b) La probabilité $\alpha_{np}(m)$ (A.14) est ensuite déterminée de manière récursive (Figure A-2).
   c) Les valeurs $\alpha_{np}(m)$ et $\gamma_{np}(m',m)$ sont sauvegardées.
   d) Cette étape 2 est répétée jusqu'à la réception de toute la séquence de symboles *R*.

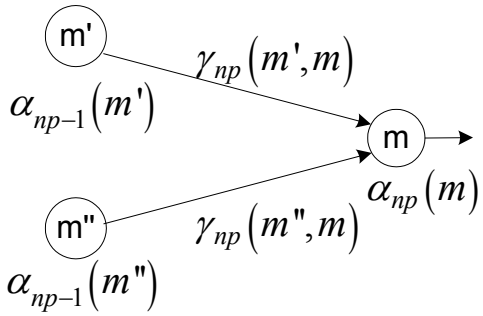

Figure A-2: représentation graphique de la mise à jour de la probabilité α(.) [Vuce00].

3. *Backward Recursion*. Après avoir reçu l'intégralité de la séquence de symboles *R*, on entame cette troisième procédure. Pour *np*=*B*-1, *B*-2, …, 0:
    a) La probabilité $\beta_{np}(m)$ est calculée selon l'équation (A.15) (Figure A-3).
    b) Les probabilités $\lambda_{np}(m)$ et $\sigma_{np}(m',m)$ peuvent être ainsi déterminées selon les formules (A.7) et (A.8).

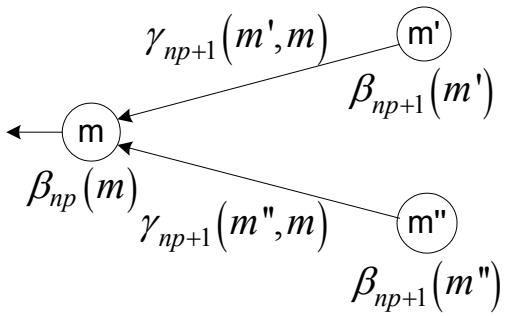

Figure A-3: représentation graphique de la mise à jour de la probabilité β(.) [Vuce00].

A.2.4 Décodage de codes convolutifs

Le contexte du décodage de codes convolutifs implique en particulier que:

- La probabilité de la transition Tr[.¦.] (A.16) tient compte de la modélisation de la distorsion affectant le canal de transmission. Elle peut se baser soit sur la distance de Hamming, soit sur une fonction euclidienne.

- La probabilité de la valeur de sortie x_{np}, associée à la transition entre deux états $q_{np}(x_{np}|m',m)$ (A.16) est normalement une fonction déterministe, invariable dans le temps (*time-invariant*) et dépendante de la transition. Dans ce contexte, cette valeur x_{np} regroupe le *n* bits de sortie du codeur convolutif.
- Si la procédure de codage établit une initialisation de la mémoire du codeur et/ou l'utilisation de bits de terminaison, l'initialisation de la procédure *MAP* doit prendre en considération ces informations:

$$\alpha_0\left(\text{état de départ}\right)=1,\ \text{and}\ \alpha_0\left(m\right)=0,\text{for}\ m\ \neq \text{état de départ}$$
$$\beta_B\left(\text{état finale}\right)=1,\ \text{and}\ \beta_B\left(m\right)=0,\text{for}\ m\ \neq \text{état finale}.$$

- La tâche spécifique du décodage de codes convolutifs est l'estimation des bits ou des symboles contenus dans le message envoyé. La mesure de la fiabilité Λ

$$\begin{aligned}\Lambda\left(\text{info bit}_{\text{np}}\right)&=\log\frac{\Pr\left[\text{info bit}_{\text{np}}=1\middle|\text{S}\right]}{\Pr\left[\text{info bit}_{\text{np}}=0\middle|\text{S}\right]}\\ &\overset{\text{Règle de Bayes}}{=}\log\frac{\Pr\left[\text{info bit}_{\text{np}}=1,\ \text{S}\right]}{\Pr\left[\text{info bit}_{\text{np}}=0,\ \text{S}\right]}\end{aligned}\qquad\text{(A.17)}$$

peut être obtenue par moyen des probabilités

$$\Pr\left[\text{info bit}_{\text{np}}=i,R\right]=\sum_{\substack{\textit{transition}\ (m',m)\\ \textit{impliquant}\ \textbf{info bit}_{\textbf{np}}=i}}\sigma_{np}(m',m).\qquad\text{(A.18)}$$

La fiabilité Λ peut être ainsi formulée [Vuce00]:

$$\Lambda\left(\text{info bit}_{\text{np}}\right)=\log\frac{\sum_{(m',m)}\alpha_{np-1}\left(m'\right)\cdot\gamma_{np}^{1}\left(m',m\right)\cdot\beta_{np}\left(m\right)}{\sum_{(m',m)}\alpha_{np-1}\left(m'\right)\cdot\gamma_{np}^{0}\left(m',m\right)\cdot\beta_{np}\left(m\right)}\qquad\text{(A.19)}$$

où

$$\gamma_{np}^{i}\left(m',m\right)=\Pr\left[\text{info bit}_{\text{np}}=i,\ s_{np}=m,\ r_{np}\middle|s_{np-1}=m'\right]\qquad\text{(A.20)}$$

A.3 Soft Output Viterbi Algorithm (SOVA)

Comme déjà introduit, certaines applications demandent une information supplémentaire sur la procédure de décodage (fiabilité Λ). Cependant, bien que l'algorithme de Viterbi présente de bonnes qualités algorithmiques et de décodage, à l'origine il n'est pas prévu pour cette tâche.
Le concept de *Soft Output Viterbi Algorithm* (*SOVA*) a été ainsi proposé et plusieurs méthodes ont été développées [Forn73] [Hage95] [Vale98] [Vuce00]. Ce concept envisage la modification de l'algorithme de Viterbi de manière à le rendre capable de produire une estimation de la fiabilité Λ.

A.3.1 Méthode Bidirectional Soft Output Viterbi Algorithm

L'algorithme qui est brièvement présenté est la méthode *Bidirectional Soft Output Viterbi Algorithm* ('*bidirectional SOVA*') [Vuce00], qui offre l'avantage d'une simplicité de compréhension et d'implantation.

Cette méthode estime la fiabilité Λ du *np*-ème bit en considérant le rapport existant entre la métrique du chemin globalement le plus probable et celle du meilleur chemin, qui s'est écarté de ce chemin à la profondeur *np-1* (Figure 5-8). Ce second chemin représente le meilleur chemin impliquant l'inversion de la valeur du *np*-ème bit, à partir du chemin globalement le plus probable.
Les deux métriques considérées sont ainsi les suivantes:

$$\begin{aligned} PMbest^{(\text{info Bit}_{np}=0)} &= \max_{\text{chemin} \in \text{bestChemins}_{np-1}} PM^{(\text{chemin}\,|\,\text{info Bit}_{np}=0)} \\ PMbest^{(\text{info Bit}_{np}=1)} &= \max_{\text{chemin} \in \text{bestChemins}_{np-1}} PM^{(\text{chemin}\,|\,\text{info Bit}_{np}=1)}, \end{aligned} \tag{A.21}$$

où la classe $bestChemins_t$ indique les chemins qui suivent le chemin le plus probable jusqu'à la profondeur *np*.
En supposant que les probabilités à posteriori du *np*-ème bit sont fonction de ces métriques

$$\begin{aligned} \log\left(\Pr\left[\text{info bit}_{np}=1\,|\,S\right]\right) &\propto PMbest^{(\text{info Bit}_{np}=1)} \\ \log\left(\Pr\left[\text{info bit}_{np}=0\,|\,S\right]\right) &\propto PMbest^{(\text{info Bit}_{np}=0)}, \end{aligned} \tag{A.22}$$

la fiabilité **Λ** peut être estimée selon la différence qui existe entre les valeurs de ces deux métriques

$$\Lambda\left(\text{info bit}_{np}\right)=\log\frac{\Pr\left[\text{info bit}_{np}=1\middle|S\right]}{\Pr\left[\text{info bit}_{np}=0\middle|S\right]} \qquad (A.23)$$
$$=PM_{Best}^{(\text{info Bit}_{np}=1)}-PM_{Best}^{(\text{info Bit}_{np}=0)}.$$

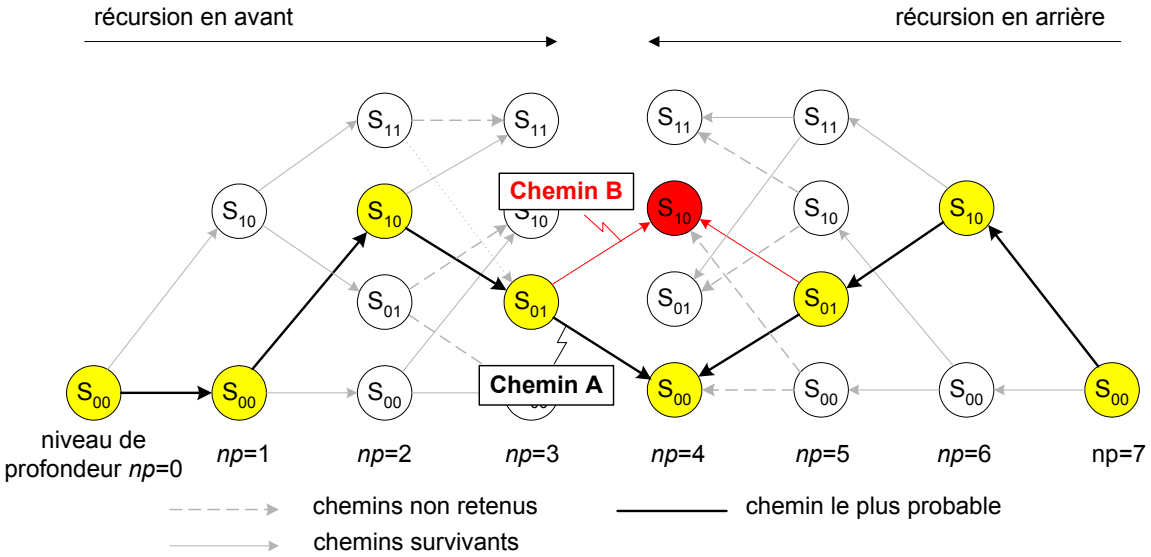

Figure A-4 : exemple graphique de l'estimation de la fiabilité Λ du quatrième bit par l'algorithme *Bidirectional SOVA*. L'estimation utilise ainsi les métriques du chemin *A* (chemin globalement le plus probable) et du chemin *B* (le meilleur chemin qui s'est séparé à partir de la profondeur *np*=3).

A.3.2 Algorithme

La méthode bidirectionnelle peut être résumée par les tâches suivantes [Vuce00]:

1) 'Récursion en avant' (*Forward Recursion*). L'objectif est l'identification du chemin globalement le plus prometteur en exploitant la notion de survivant. Cette récursion coïncide avec l'algorithme de Viterbi.

a) Les métriques cumulées des états sont initialisées selon les connaissances (à priori) des conditions de départ du processus analysé.
b) La variable np_1 est mise à zéro. Cette variable concerne la progression de cette 'récursion' dans le treillis.
c) Pour chaque nœud à la profondeur np_1+1 :
   i) les chemins convergeants dans le nœud sont identifiés;
   ii) leurs métriques cumulées PM_{np1} sont calculées à l'aide des métriques de branche et des anciennes métriques cumulées;
   iii) le chemin survivant est ainsi sélectionné;
   iv) la valeur de la métrique cumulée du chemin survivant est assignée au nœud;
   v) les informations nécessaires à la reconstruction de ce chemin sont sauvegardées.
d) L'étape c) est répétée jusqu'à atteindre la fin du treillis ($t_1=B$), en incrémentant de *1* le niveau de profondeur np_1.

2) 'Récursion en arrière' (*Backward Recursion*). Son but est le calcul des contributions minimales dues à la prolongation des chemins (à chaque niveau de profondeur et pour chaque nœud).
a) Les métriques cumulées des états finaux sont initialisées selon les conditions de terminaison du processus analysé.
b) Le nombre de symboles B du message est assigné à la variable np_2. Cette variable indique la progression de cette 'récursion' dans le treillis.
c) Pour chaque nœud à la profondeur np_2-*1* :
   i) les prolongations du chemin lié au nœud analysé sont identifiées;
   ii) la contribution à la métrique cumulée de chaque prolongation i

$$PM_{np_2 \to B}^{(S_i)} = \sum_{np=np_2+1}^{B} \mu_{np}^{(S_i)} = PM_{(np_2-1) \to B}^{(S_i)} + \mu_{np+1}^{(S_i)} \qquad \text{(A.24)}$$

   est calculée à l'aide de la métrique de branche $\mu^{(S_i)}$;
   iii) cette valeur est retenue et assignée au nœud.
d) L'étape c) est répétée jusqu'à ce que le début du treillis de recherche soit atteint (t_2=0). Le niveau de profondeur np_2 est décrémenté de *1*.

3) Estimation de la fiabilité Λ
a) La variable np_3, responsable du positionnement dans le treillis est initialisé à 1.
b) A la profondeur np_3:
   i) le chemin le plus probable est identifié;

ii) le meilleur chemin, qui s'éloigne au niveau de profondeur np_3 du meilleur chemin, est ensuite déterminé;
iii) sa métrique est calculée en utilisant les valeurs de métrique cumulée PM_{np3}, de la contribution $PM_{(np3+1)\rightarrow B}$ et de la métrique de branche μ_{np3};
iv) la fiabilité Λ du np_3-ème bit est ainsi estimée, en utilisant les valeurs des métriques des deux chemins.

c) Cette étape b) est répétée jusqu'à atteindre la fin du treillis ($np_3=B$), en incrémentant la variable $np_3= np_3+1$.

Normalement, les tâches relatives à la récursion en arrière et à l'estimation de la fiabilité sont exécutées simultanément, réduisant la complexité totale de la méthode [Vuce00].

A.3.3 Complexité de calcul

Du point de vue de la complexité de calcul, une analyse superficielle de cette méthode *SOVA* "bidirectionnelle" met en évidence que:

- La récursion en avant est équivalente à l'algorithme de Viterbi.
- La procédure de récursion en arrière utilise la même structure que l'algorithme de Viterbi, à l'exception de l'opération de stockage des informations pour la reconstruction des chemins survivants. La complexité est donc réduite par rapport à l'algorithme de Viterbi.
- La complexité de l'estimation de la fiabilité peut être réduite en intégrant cette fonction à la récursion en arrière [Vuce00].

Par conséquent, dans le contexte d'une décision ferme, on peut s'attendre à une complexité totale s'élevant approximativement à 1,5 fois la complexité de l'algorithme de Viterbi pour le décodage du même message [Vuce00].

En généralisant, la complexité des méthodes appartenant à la classe des algorithmes *SOVA* peut être estimée entre 1 et 2 fois la charge de calcul de l'algorithme de Viterbi.

A.4 Références

[Bahl74] L. R. Bahl, J. Cocke, F. Jelinek, et J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory*, Vol. 20, mars 1974, pp. 284-287.

[Forn73] G. D. Forney Jr., "The Viterbi Algorithm", *Proceedings of the IEEE*, Vol. 61, No. 3, mars 1973, pp.268-278.

[Hage95] J. Hagenauer, "Source-Controlled Channel Decoding", *IEEE Transactions on Communications*, Vol. 43, No. 3, septembre 1995, pp. 2449-2457.

[Joha99] R. Johannesson, K. S. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Series on Digital and Mobile Communication, Wiley-IEEE Press, Etats-Unis d'Amérique, 1999, chapitres 4-6, pp. 163-315.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[Robe95] P. Robertson, E. Villebrun, et P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", *Proc. ICC'95*, Seattle, juin 1995, pp. 1009-1013.

[Schl97] C. Schlegel, *Trellis Coding*, IEEE Press, Etats-Unis d'Amérique, 1997.

[Vale98] M. C. Valenti, *Iterative Detection and Decoding for Wireless Communications*, A Preliminary Review of Initial Research and Proposal for Current and Future Work towards Doctor of Philosophy degree, Virginia Polytechnique Institute and State University, Blacksburg, Virginia, Etats-Unis d'Amérique, 1998.

[Vuce00] B. Vucetic, J. Yuan, *Turbo Codes, Principles and Applications*, Kluwert Academic Publishers, Etats-Unis d'Amérique, 2000.

Annexe B: Architectures utilisant un processeur pour le traitement du signal

B.1 Codes en bloc CRC des standards UMTS

En raison de l'incapacité des méthodes de décodage convolutif à définir les conditions d'erreurs garantissant un décodage correct (sans limiter le potentiel de correction), le standard *UMTS* [Ts25212] met à disposition une série de codeurs en bloc.

B.1.1 Codage en bloc CRC établi par les standards UMTS

Les codeurs mis à disposition sont quatre codes *CRC* systématiques différents (Table B-1), qui permettent un choix optimal et adéquat à chaque contexte de codage.

Codeur	Longueur [bits]	Polynôme générateur g(X)	Longueur du message codé, longueur du message (n, k) [bits]	Distance minimale de Hamming observée d_{min}
$g_{CRC24}(X)$	24	$X^{24} + X^{23} + X^6 + X^5 + X + 1$	$(2^{23}\text{-}1, 2^{23}\text{-}25)$	4 (1)
$g_{CRC16}(X)$	16	$X^{16} + X^{12} + X^5 + 1$	(32767, 32752)	4 (1)
$g_{CRC12}(X)$	12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$	(2047, 2035)	4
$g_{CRC8}(X)$	8	$X^8 + X^7 + X^4 + X^3 + X + 1$	(127, 119)	4
(1) L'utilisation d'un message de 1'000 bits a permis l'observation d'une distance minimale de 4.				

Table B-1: vue d'ensemble des codes cycliques (*CRC*) mis à disposition pour la structure de codage de canal *UMTS*.

Le standard *UMTS* [Ts25212] modifie par contre la concaténation des bits du code *CRC*, qui doit s'effectuer dans l'ordre inverse. Le but de cette inversion est de garantir le fonctionnement correct de la tâche de *blind rate detection*[56]

[56] La fonction de *Blind rate detection* peut être utilisée pour la détection du format utilisé pour le transport (*Transport Format*), lorsque l'indicateur de la combinaison de format (*Transport Format Combination Indicator*, 'TFCI') n'est pas transmis. Le principe

[NttD99], en empêchant des situations d'incertitude par rapport à la taille du message (Figure B-1).

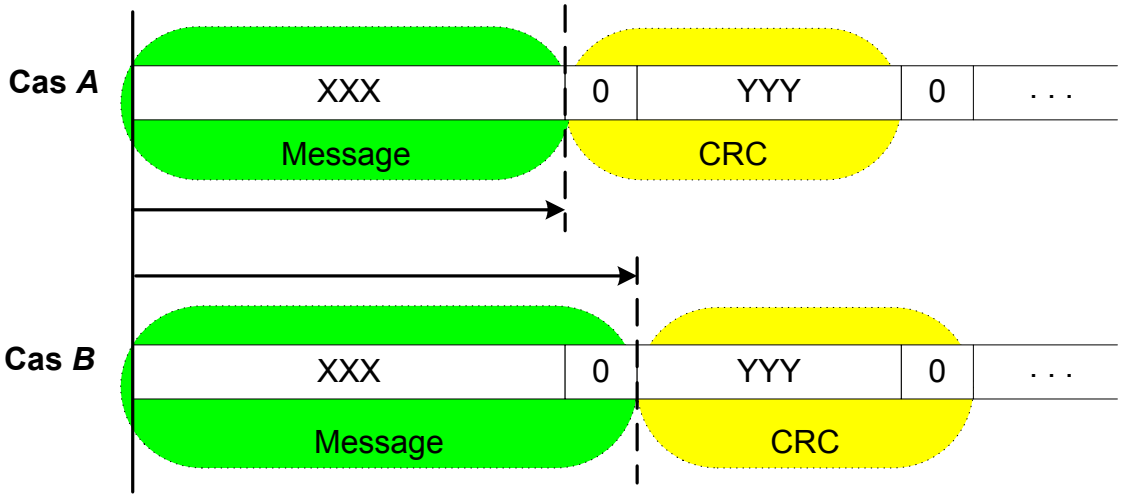

Figure B-1: exemple d'une situation d'incertitude, causée par un attachement de bits de parité de manière décroissante (approche classique).

B.1.2 Stratégie de détection d'erreurs

La détection d'erreurs peut s'accomplir selon deux stratégies [Pres92] [Proa95] [Will99]. La première suit l'approche classique du décodage du message. A partir de la valeur résultante par ce décodage, valeur qui est nommée syndrome (*Syndrome*), l'absence d'erreur peut être soit validée, soit mise en doute. La seconde stratégie exploite par contre la propriété systématique de ces codages: à partir du message sous analyse, les bits de parité sont re-calculés et comparés avec ceux reçus. Cette dernière stratégie a été adoptée pour les implantations software de cette étude.

La complexité liée au calcul des bits de parité (Table B-2) peut être simplifiée par l'utilisation d'une table contenant les valeurs des bits de parité pré-calculées sur 8 bits (encodeur de type *Table-Driven* [Will99]). Cette implantation a été choisie en prévision d'une utilisation successive dans des méthodes itératives de décodage.

---

implique le décodage systématique du message en considérant une gamme de tailles différentes: les bits de parité *CRC* sont ensuite utilisés pour la validation du message.

	Nombre moyen d'opérations observées			
	Format des variables			
	32 bits	*16 bits*	*8 bits*	***Total***
Additions	0	1		**1**
Incrémentations	0	14		**14**
Soustractions	0	10		**10**
Multiplications	0	0		**0**
Divisions	0	0		**0**
Transferts de Données	1	4		**5**
Logique	36	2	13	**51**
Décalages	58	2	1	**61**
Contrôles			27	**27**
Pointeur: assignations			1	**1**
Pointeur: arithmétique				**0**
Boucle: initialisations	0	1		**1**
Boucle: contrôles			11	**11**
Boucle: incrémentations	0	10		**10**
Grand total				**192**

Table B-2: charge de calcul de la tâche de détection d'erreurs (code *CRC* utilisant 12 bits de parité). Paramètres: les 500 messages utilisés sont conformes à la structure de la classe *A* du service de parole *AMR-NB* à 12.2 kbps, les valeurs montrent des écarts maximaux inférieurs à 1%. Le fond gris met en évidence les opérations définies par les organisations *ITU* et *ETSI*.

B.1.3 Capacité de détection d'erreurs des codes CRC du UMTS

Concernant la faisabilité des techniques de décodage itératif, une évaluation des performances des codes *CRC* confirme que les capacités de détection d'erreurs sont supérieures à celles des codes précédemment utilisés dans le standard *GSM* (Table B-3 et Table B-4).
Cette constatation rend les techniques de décodage itératif attrayantes. Toutefois, l'efficacité d'un tel décodage est soumise au rapport existant entre les caractéristiques de la transmission de données (nombre et répartition des erreurs de transmission), la taille des messages, les potentialités de détection du code *CRC* utilisé et le nombre d'itérations de recherche du message correct. Comme déjà cité, le bénéfice d'un tel décodage dans le cadre de la téléphonie mobile *GSM* a été limité par un nombre considérable de cas de fausses validations, causés par une capacité de détection réduite du code en bloc utilisé (d_{min} de 2, Table B-4) par rapport aux caractéristiques de l'application [Kühn97].

Codeur [ETSI909]	Longueur [bits]	Polynôme générateur g(X)	Longueur du message codé, longueur du message (n, k) [bits]	Distance minimale de Hamming observée d_{min}
TCH/EFS	8	$X^8 + X^4 + X^3 + X^2 + 1$	(255, 247)	3
TCH/FS-HS-EFS	3	$X^3 + X + 1$	(7, 4)	2

Table B-3: vue d'ensemble des codes cycliques utilisés par les standards *GSM*.

Messages: 100 bits	**Nombre observé de messages codés avec *poids* w_i de:** 1	2	3	4	5	6	7	8	9
Codeurs CRC *UMTS*									
CRC 8-bit	0	0	0	$2.2 \cdot 10^5$	0	$4.5 \cdot 10^6$	0	$5.5 \cdot 10^8$	0
CRC 12-bit	0	0	0	$1.9 \cdot 10^3$	0	$3.7 \cdot 10^5$	0	$5.0 \cdot 10^7$	0
CRC 16-bit	0	0	0	$2.7 \cdot 10^2$	0	$3.1 \cdot 10^4$	0	$4.4 \cdot 10^6$	0
CRC 24-bit	0	0	0	$7.0 \cdot 10^2$	0	$2.5 \cdot 10^4$	0	$8.9 \cdot 10^5$	0
Codeur CRC *GSM*									
CRC 3-bits	0	$4.6 \cdot 10^2$	$1.2 \cdot 10^4$	$2.4 \cdot 10^5$	$3.9 \cdot 10^6$	$5.1 \cdot 10^7$	$5.7 \cdot 10^8$	…	…
CRC 8-bits	0	0	$8.2 \cdot 10^2$	$2.1 \cdot 10^4$	$4.4 \cdot 10^5$	$7.5 \cdot 10^6$	$1.1 \cdot 10^8$	…	…

Table B-4: vue d'ensemble des performances des codes *CRC* relatifs aux standards *UMTS* [Ts25212] et *GSM* [ETSI909]. Le fond gris met en évidence la distance minimale observée des codes. Paramètres: messages contenant 100 bits d'information, méthode de recherche par ordinateur.

B.2 Procédure développé de sélection des chemins les plus favorables dans méthode List Decoding

Une procédure de sélection exploitant la statistique des valeurs des métriques cumulées a été développée. L'objectif est la détermination du seuil permettant la séparation des valeurs de métrique en deux sous-ensembles égaux (voir Sous-Section 6.7.4, Figure 6-23). Ce principe est basé sur l'observation d'une forte concentration des valeurs de métriques cumulées, due à une différence maximale entre les métriques qui est réduite et stable pendant le décodage (Figure 6-24).

B.2.1 Prédiction des limites des valeurs des métriques cumulées

A partir de la valeur de la métrique cumulée d'un chemin (que dans ce contexte on nomme "générateur"), on peut déterminer la limite inférieure et supérieure des chemins qui se propageront ensuite depuis ce chemin (Figure B-2). La détermination des limites prend en considération deux chemins:

- le chemin qui se propage dans les meilleures conditions de transmission (en l'absence d'erreurs de transmission), et
- celui qui se propage en cumulant le nombre maximal d'erreurs.

Par conséquent, étant donné une métrique de branche basée sur la pure distance de Hamming, la limite inférieure coïncide de manière univoque avec la métrique cumulée du chemin générateur (Figure B-2). De manière analogue, l'estimation de la limite supérieure doit assumer, à chaque transition de niveau, la contribution maximale de la métrique de branche ('*MaxContribBranche*').
Par conséquent, à la profondeur np_1, les métriques de ces chemins sont comprises dans les limites:

$$\left[MetrGen_{np} \;\; ... \;\; MetrGen_{np} + \left(MaxContribBranche \cdot \left(np_1 - np \right) \right) \right], \quad \text{(B.1)}$$

où $MetrGen_{np}$ est la métrique cumulée du chemin générateur.

En considérant la situation de la sélection des L chemins les plus prometteurs dans une liste de $2L$ chemins, la présence d'autres chemins (qui découlent d'autres chemins générateurs à la même profondeur np) ne modifie pas les limites (B.1), si

1. Au niveau de profondeur np: le chemin générateur qui est utilisé pour la détermination des limites montre la métrique cumulée la plus favorable, et;
2. Au niveau de profondeur np_1: le nombre de chemins produits par chaque chemin générateur dépasse ou équivaut à $2L$.

Dû aux effets de l'opération de sélection des chemins les plus favorables, à partir de la profondeur

$$np_2 = np + \left\lceil \log_2 \left(2 \cdot L \right) \right\rceil \quad \text{(B.2)}$$

(point 2), les métriques cumulées des *2L* chemins à sélectionner sont comprises dans les limites (B.1) qui contiennent les valeurs les plus favorables. Ces limites sont évidemment déterminées par le chemin générateur le plus favorable (point 1).

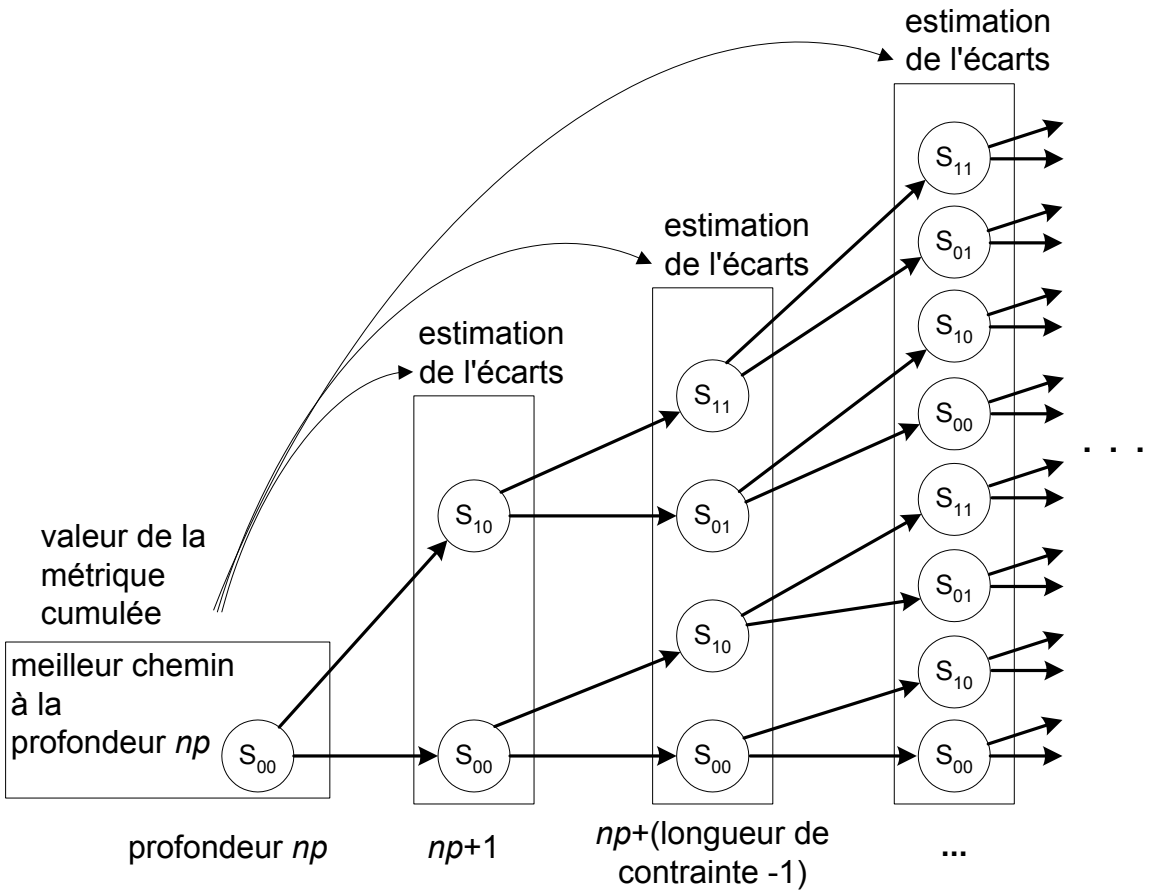

Figure B-2: exemple de propagation des chemins, qui se produit depuis le meilleur chemin à la profondeur *np*. Code avec longueur de contrainte de 3.

B.2.2 Stabilité de la différence entre les limites supérieures et inférieures

Par rapport à une situation dynamique, les limites à la profondeur np_2+1 peuvent être ainsi estimées par la formule (B.1), en utilisant la métrique cumulée soit du meilleur chemin à la profondeur *np*, soit de celui à la profondeur *np+1* (Figure B-3).

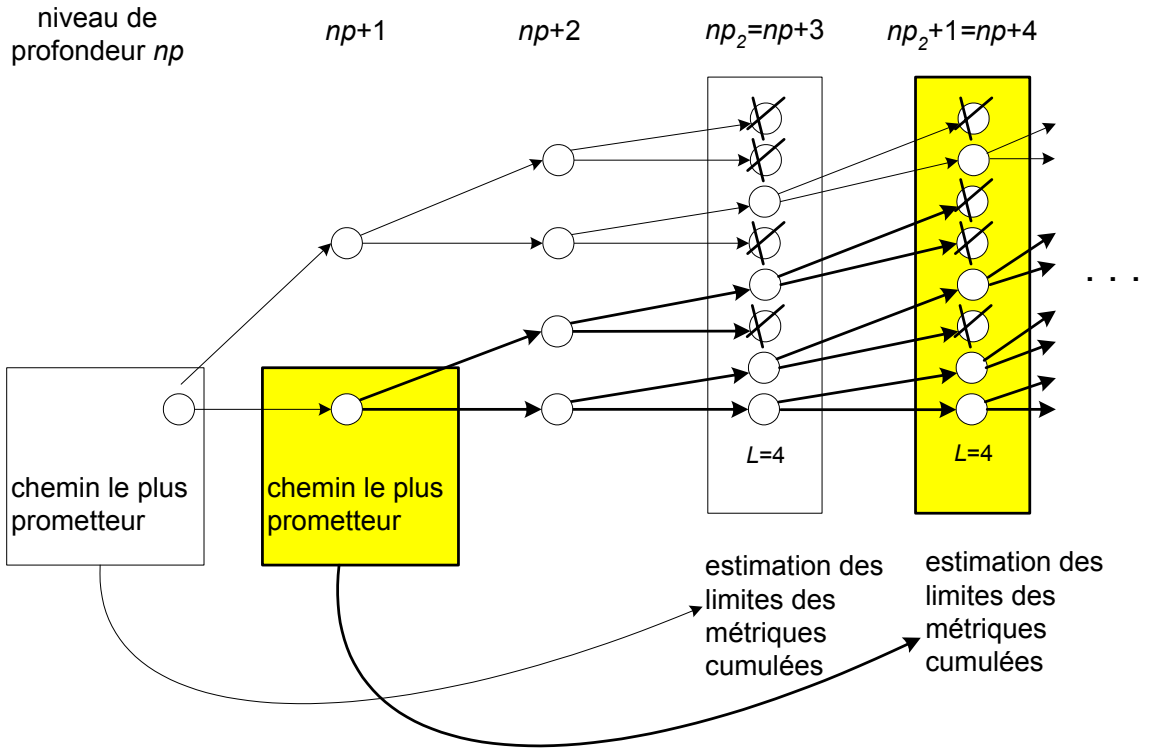

Figure B-3: représentation graphique de la prédiction des limites supérieures et inférieures des métriques cumulées.

L'utilisation des informations de métriques plus récentes permet évidemment d'obtenir une estimation plus précise des limites. Par conséquent, bien que les métriques cumulées aient tendance à augmenter en présence d'erreurs de transmission (Figure 6-24), cette stratégie d'estimation met en évidence la stabilité de la différence maximale entre les métriques. On peut ainsi fixer que la différence maximale entre les métriques cumulées des *2L* chemins à sélectionner est de:

$$dmm = MaxContribBranche \cdot \left\lceil \log_2 \left(2 \cdot L \right) \right\rceil . \qquad \text{(B.3)}$$

B.2.3 Procédure de sélection par établissement d'un seuil de métrique

Les conséquences de cette estimation des limites sont importantes: on sait que les *2L* métriques cumulées montrent un écart réduit, prévisible et indépendant du niveau de profondeur.

A partir de la métrique cumulée momentanément la plus favorable ('$BestMetr_{np}$'), on peut en effet estimer la limite inférieure et supérieure des métriques des *2L* chemins qui atteindront le niveau de profondeur suivant:

$$\left[BestMetr_{np} \ldots BestMetr_t + dnn + MaxContribBranche \right]. \qquad \text{(B.4)}$$

La concentration des valeurs permet ainsi le développement d'une procédure de sélection utilisant la distribution des métriques cumulées. Son principe est la détermination d'un seuil qui délimite les *L* métriques les plus favorables (Figure 6-23), en établissant la statistique des *2L* nouvelles métriques cumulées.

B.3 Affinité entre l'implantation software et l'architecture du processeur

La Table B-5 met en évidence l'importance d'une description par langage à haut niveau adaptée aux caractéristiques architecturales du processeur utilisé. Cette table montre les vitesses d'exécution des méthodes, dont les implantations visent différentes plates-formes. Bien que ces résultats valident les potentialités et les expectatives des diverses méthodes, l'importance du style de la description du code est facilement observable.

Méthodes (sans troncation de la mémoire)		Plates-formes visées [Bits]	Nombre minimal de cycles [cycles]					
			Type de processeur (32 Bits)					
			Pentium, Intel Family 5			*Celeron, Intel Family 6*		
			Nombre de cycles	Normalisé par le nombre d'opérations	Normalisé par le nombre *WCwOPS*(*)	Nombre de cycles	Normalisé par le nombre d'opérations	Normalisé par le nombre *WCwOPS*(*)
Algorithme de Viterbi		32	1'744'035	2.46	2.83	1'260'355	1.77	2.05
Validation CRC		-	2'770	-	-	2'800	-	-
List Viterbi Decoding	*L*=2	mixte 16/32	2'256'022	3.07	3.51	1'833'843	2.49	2.86
	L=4		2'320'739	3.14	3.59	1'909'380	2.58	2.96
	L=8		2'343'000	3.13	3.58	1'936'016	2.59	2.96
	L=16		2'386'972	3.12	3.55	2'050'829	2.68	3.05
L. Decoding intégr. la val. *CRC*	*L*=32	16	609'350	3.23	3.05	519'173	2.75	2.60
	L=64		1'156'162	3.30	3.13	1'002'946	2.87	2.72
	L=128		2'313'910	3.46	3.28	1'919'995	2.87	2.72
Procédure d'acquisition du nombre de cycles (rdTSC)			46			181		

(*) L'évaluation de la complexité de calcul *WCwOPS* (extension de la mesure *wOPS* de l'*ITU/ETSI*) vise des plates-formes à 16 bits.

Table B-5: vue d'ensemble des performances indicatives, en termes de temps d'exécution. Le contexte de codage concerne la structure de protection des bits appartenant à la classe *A* du service de parole *AMR-NB* à 12.2kbps (codes concaténés en série). La procédure d'évaluation se base sur l'utilisation du compteur *read-time stamp counter* (RDTSC) [Inte97].

B.4 Références

[ETSI909] ETSI, *Channel Coding*, document ETSI EN 300 909 GSM 05.03, version 7.1.0.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grande-Bretagne, 2002.

[Inte97] Intel Corporation, *Using the RDTSC Instruction for Performance Monitoring*, notes d'application, page Internet accédée au printemps 2001:

http:/developper.intel.com/drg/pentiumII/appnotes/RDTSCPM1.HTM

[Kühn97] V. Kühn, *Applying List output Viterbi Algorithms to a GSM-based Mobile Cellular Radio System*, présentation à International Conference on Universal Personal Communications ICUPC'97, San Diego, Etats-Unis d'Amérique, 1997.

[NttD99] NTT DoCoMo, *TSGR1#5(99)689*, TSG-RAN Working Group1, meeting #5, Cheju, Korea, 1-4 Juni 1999.

[Pres92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, Cambridge University Press, 1992.

[Proa95] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, Singapour, 1995.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

[Will99] R. N. Williams, "A Painless Guide to CRC Error Detection Algorithm", version 3, page Internet accédée au printemps 1999: ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt

Annexe C: Architectures basées sur circuits ASIC

C.1 Réalisation physique d'un décodeur compatible avec le standard UMTS

Comme décrit préalablement, l'utilisation du système de base dans le contexte de codage *UMTS* nécessite la modification de sa conception, afin de supporter la procédure d'adaptation du débit de transmission et d'exploiter les bits de terminaison [Ts25212].
Ces modifications sont brièvement présentées et discutées.

C.1.1 Adaptation du débit de transmission: la répétition et l'élimination des bits du message codé

Les standards *UMTS* prévoient l'emploi d'une procédure d'adaptation du débit de transmission. La conception du système de base doit ainsi se conformer à ce contexte de décodage, en modifiant la réalisation matérielle du module chargé du calcul des métriques des branches (*BMu*).

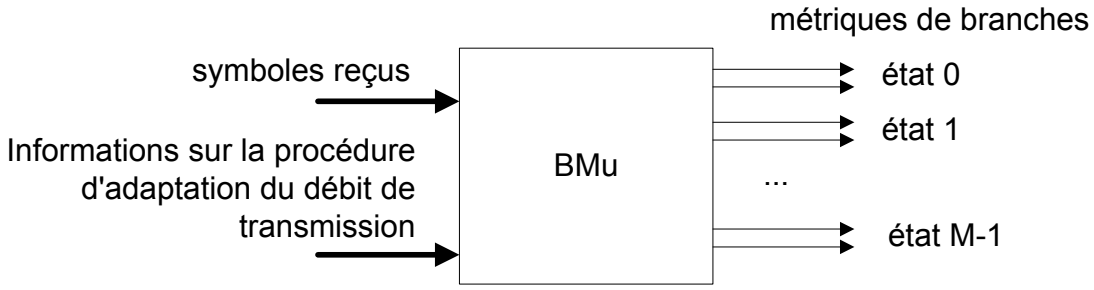

Figure C-1: calcul des métriques de branche dans un contexte de décodage prévoyant l'adaptation du débit de transmission.

Ce module doit fournir les valeurs des métriques cumulées, en suivant les indications relatives à la répétition et à l'élimination ('poinçonnage') des bits de la procédure d'adaptation (Section 3.2). Ainsi, il doit être capable de traiter correctement:

- Un symbole qui n'a subi aucun traitement par la fonction d'adaptation du débit de transmission.
- Un symbole qui a été éliminé. Ce symbole doit être pratiquement ignoré, vu qu'il n'apporte aucune information valable sur la procédure de codage.
- Un symbole répété.

En fonction de la stratégie adoptée (au niveau du système entier de décodage de canal *UMTS*), ces indications peuvent être soit intégrées au moyen d'une nouvelle représentation digitale des symboles, soit transmises au module par un moyen de transmission, physiquement séparé de celui des symboles (Figure C-1).

L'utilisation d'un bloc fonctionnel interposé (entre la chaîne de décodage de canal *UMTS* et le système de base non-modifié) ne permet pas l'exploitation exhaustive des informations d'adaptation du débit de transmission.

C.1.2 Décodage d'un message formé par un nombre fini de symboles et contenant des bits de terminaison (Tail Bits)

Afin de garantir une protection adéquate aux derniers éléments du message, les standards *UMTS* [Ts25212] prévoient l'emploi d'une série de zéros comme bits de terminaison (Figure C-2). Ces valeurs créent des conditions d'arrêt de la procédure de codage et déterminent l'identité de l'état final de la mémoire du codeur (Figure C-3).

Importance de l'état d'arrêt de la procédure de codage

L'exploitation des informations des bits de terminaison ne modifie pas le déroulement de l'algorithme de Viterbi jusqu'au moment du traitement des symboles générés par leur codage. A ce moment, l'algorithme dispose d'informations précises sur la valeur de ces derniers bits, ce qui permet l'amélioration de la qualité de protection des derniers bits d'informations du message (Figure C-2).

Ces informations peuvent bien entendu être négligées (afin d'éviter l'augmentation de la complexité de la tâche de sélection du chemin pour la prise de décision), mais seulement au détriment de la qualité de protection. Le choix d'un chemin inexistant pour le décodage (Figure C-3) augmente en effet les risques d'une prise de décision incorrecte. Le choix crucial est surtout celui

du chemin final, vu qu'il est responsable de la prise de décision des derniers *δ-(longueur de contrainte-2)* bits du message.

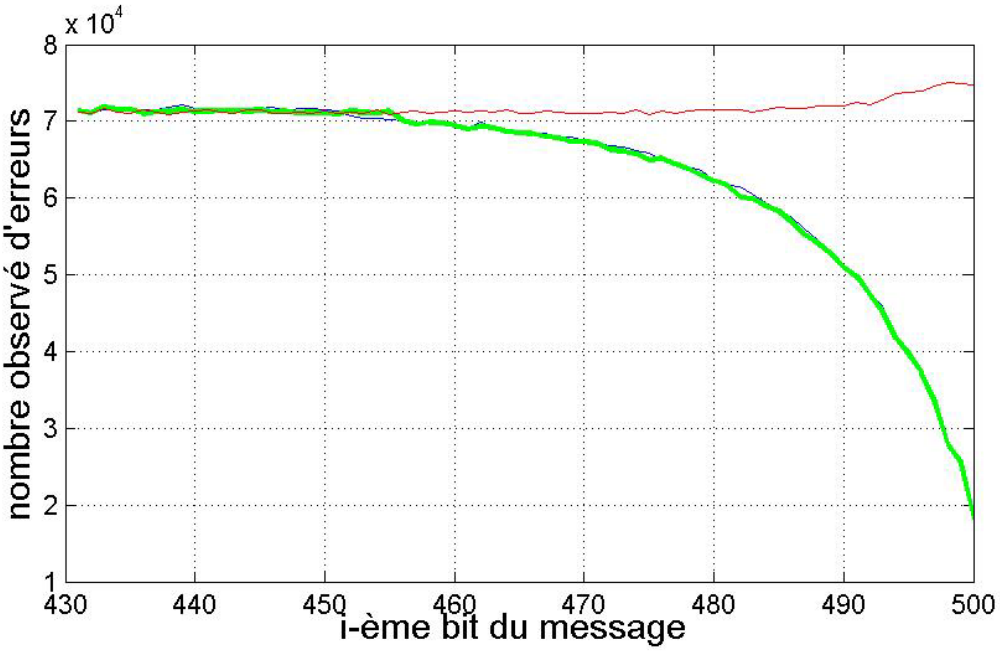

Figure C-2: exemple de la qualité de protection des derniers bits du message. La graphie bleue en trait fin indique la qualité obtenue par le respect des conditions finales de codage. Les autres graphies représentent la qualité obtenue en enfreignant ces conditions: l'inobservance des conditions finales mais l'utilisation de l'état final correct (verte en trait gras) et l'infraction totale des conditions finales de codage (rouge en trait fin). Simulation software utilisant 200'000 messages de 500 bits, 8 bits de terminaison, E_b/N_0=0.0 dB, δ=54, code *UMTS* avec R_c=1/3.

Solution pragmatique

Dans le cas d'une prise de décision basée sur les chemins les plus prometteurs, la conception du système doit prévoir une augmentation de la complexité de la procédure de sélection du meilleur chemin ou une dégradation de la qualité de protection des derniers bits du message.
Un bon compromis est l'infraction partielle des conditions de terminaison de la procédure de codage: l'exécution normale de l'algorithme de Viterbi avec l'utilisation de l'état final correct pour la prise de décision des derniers bits du message. Cette solution peut être réalisée, soit par la répétition de l'opération de prise de décision utilisant le même état final, soit par une prise de décision multiple (Figure C-8) qui fournit toutes les valeurs des derniers *δ-(longueur de contrainte-2)* bits du message.

Dans le cas d'une prise de décision par décodage d'un chemin quelconque, l'utilisation systématique du chemin atteignant l'état 'zéro' garantit une procédure toujours conforme aux conditions d'arrêt établies par le standard *UMTS* (Figure C-4).

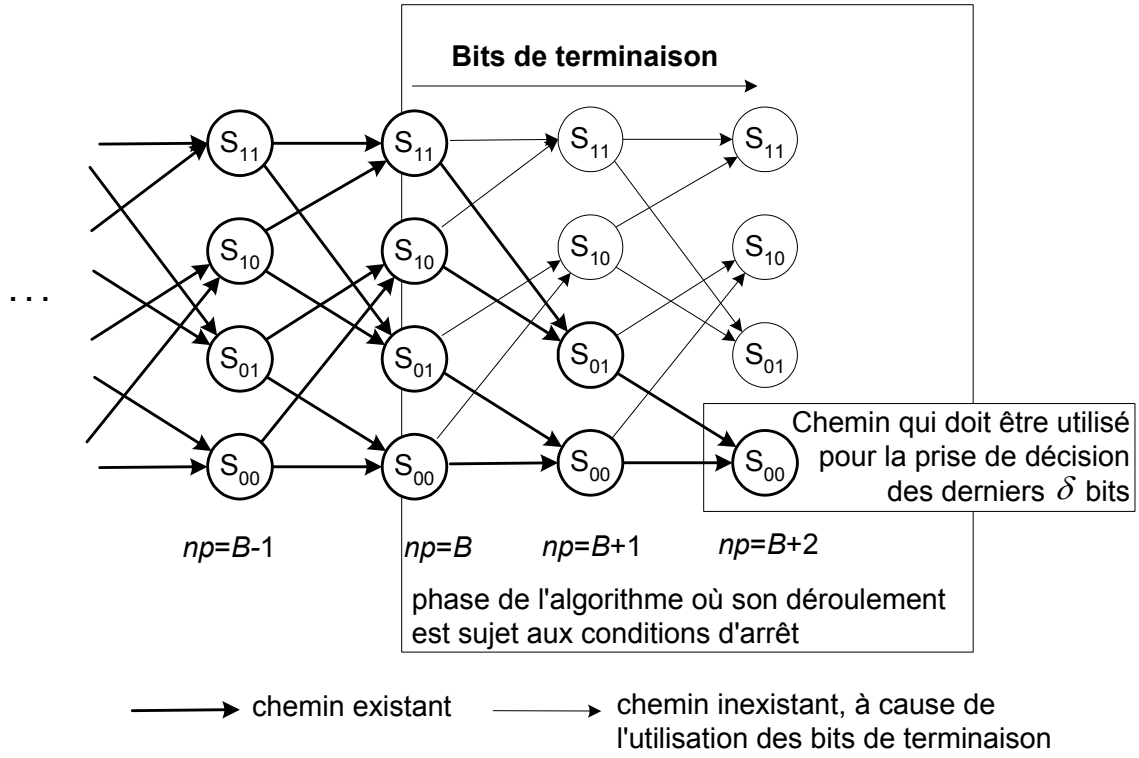

Figure C-3: illustration graphique du déroulement de la procédure de décodage lors de l'utilisation des bits de terminaison. Dans cet exemple, les bits de terminaison sont formés par une série de zéros.

C.2 Approche coprocesseur

C.2.1 Solution intéressante

L'analyse des architectures dédiées à la communication mobile montre qu'une approche coprocesseur est une solution intéressante [Gath02]. Cette solution est surtout intéressante dans la situation actuelle, où l'on demande un degré de flexibilité élevé (jeunesse des standards *3G*) ainsi qu'une importante puissance de calcul. Le circuit *ASIC* peut ainsi apporter la puissance de calcul nécessaire, alors que le *DSP* fournit tous les avantages liés à un dispositif programmable, tels que la flexibilité, la gestion des différentes fonctionnalités, la rapidité d'implantation, d'intégration et de validation.

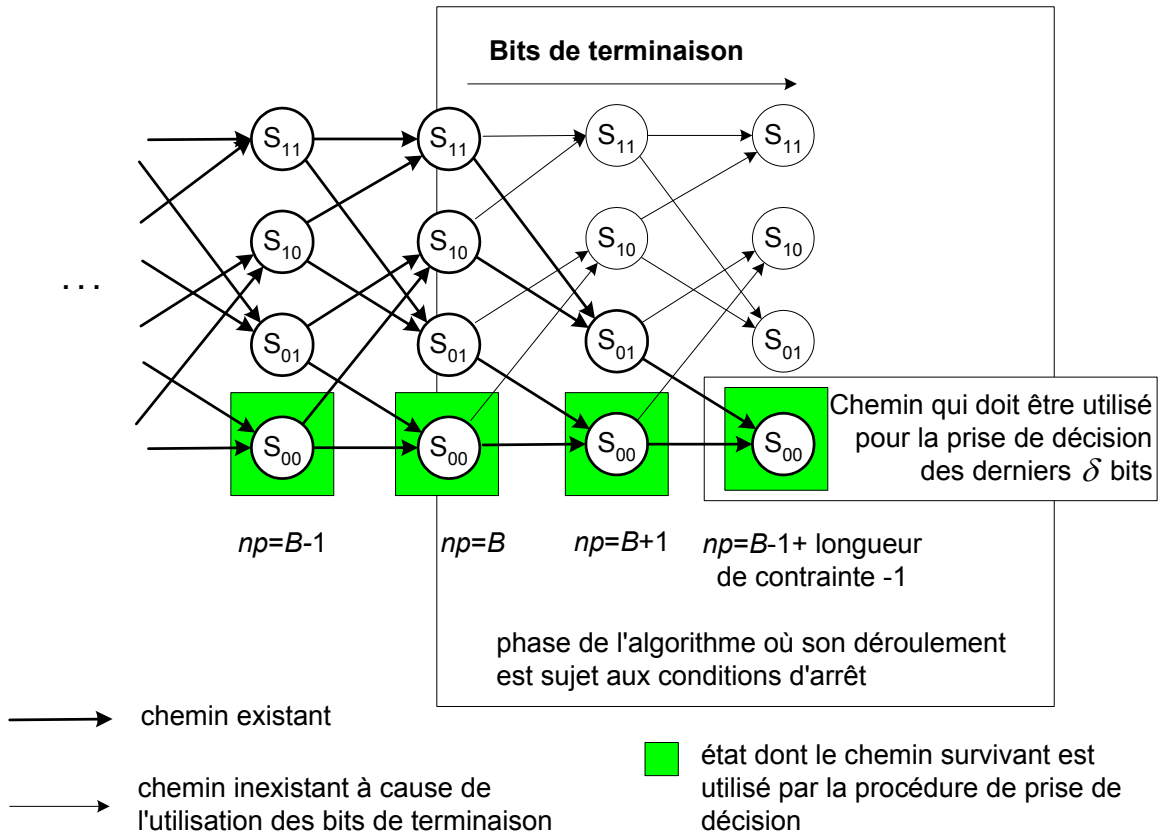

Figure C-4: adaptation de la procédure de détermination d'un chemin quelconque au contexte d'une utilisation des bits de terminaison. Dans cet exemple, les bits de terminaison coïncident avec une série de zéros.

Les deux stratégies d'emploi du coprocesseur sont [Gath02]:

- *Tightly Coupled Coprocessor* (*TCC*). Les tâches exécutées par ces coprocesseurs peuvent être considérées comme des extensions du jeu d'instruction du *DSP*. En effet, le *DSP* initialise une tâche sur le coprocesseur, qui l'exécute en un intervalle de temps correspondant à quelques cycles d'instruction du *DSP*. Pendant ce temps, le *DSP* n'exécute (normalement) aucune autre tâche.
- *Loosely Coupled Coprocessor* (*LCC*). Dans ce cas, le genre de tâche exécutée par le coprocesseur est plus une sous-routine qu'une simple instruction. L'exécution prendra ainsi plusieurs cycles d'instruction du *DSP*. Pour cette raison, l'utilisation efficace d'un tel coprocesseur prévoit le fonctionnement parallèle de deux éléments, à savoir le *DSP* et le coprocesseur.

L'approche coprocesseur permet de franchir les limites de souplesse de la réalisation *ASIC*, en bénéficiant des services d'un dispositif programmable. Cette approche prévoit ainsi la répartition des diverses tâches du décodage entre le *DSP* et le coprocesseur, de manière à charger le processeur des tâches nécessitant un niveau de souplesse supérieur et à attribuer au coprocesseur celles nécessitant un nombre élève d'opérations (Figure C-5).

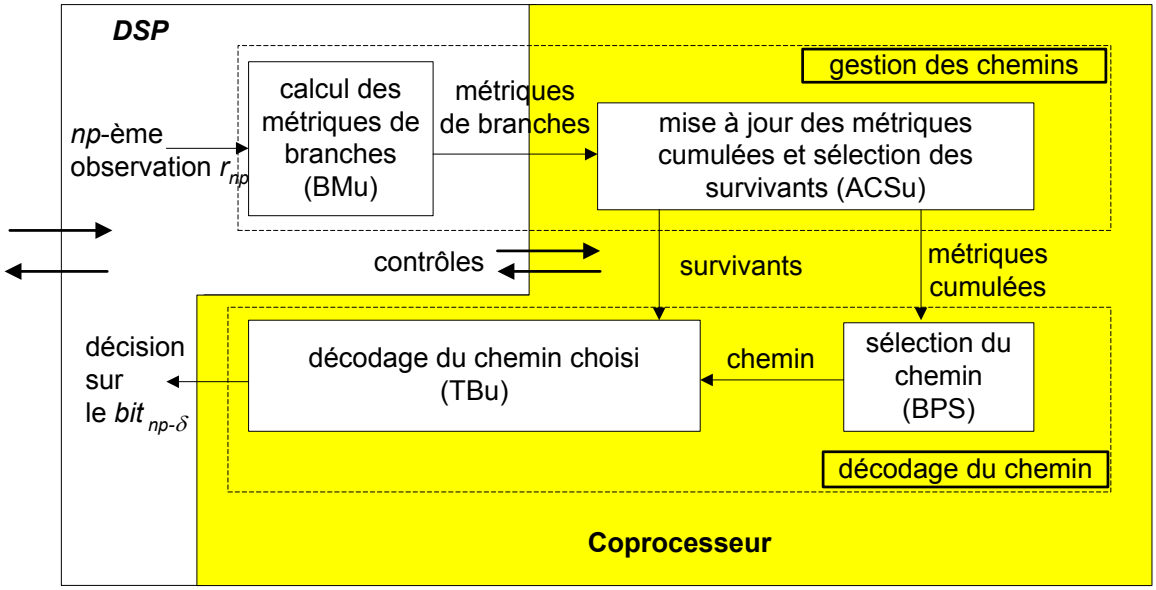

Figure C-5: proposition d'une répartition des tâches entre le *DSP* et le coprocesseur.

C.2.2 Modifications à apporter au système de base

Dans cette sous-section, les implications d'une approche souple sur la conception du système de base sont brièvement traitées. Pour rester dans un discours général, aucun type précis d'approche coprocesseur (TCC ou LCC) ne sera discuté dans cette sous-section.

La Table C-1 rassemble les paramètres du système de base. Ces paramètres augmentent le degré de souplesse de la réalisation matérielle, en l'adaptant au contexte de codage visé.

Calcul des métriques de branche (*BMu*)

C'est le module le plus dépendant des paramètres décrivant le codeur convolutif et la configuration du système dans lequel le décodeur travaille. En effet, tout petit changement de la situation de codage, tel que la modification

du rendement du code ou l'utilisation d'une fonction d'adaptation du débit de transmission, implique la modification de sa conception.

Paramètre	**Module(s) impliqué(s)**
Fonctions génératrices	Unité *BM*
Longueur de contrainte	Unités *BM*, *ACS, BPS* et *TB*
Le rendement du code R_c	Unités *BM* et *ACS*
Adaptation du débit de transmission	Unité *BM*
Condition de départ de la procédure de décodage	Unité *ACS*
Condition d'arrêt de la procédure de décodage	Unité *BPS*
Type de décision (ferme ou pondérée)	Unités *BM* et *ACS*

Table C-1: paramètres permettant l'adaptation de la méthode de décodage aux divers contextes de codage.

En raison de cette forte demande de souplesse, on peut envisager de charger le *DSP* du calcul de toutes les métriques de branches et de leur livraison au coprocesseur, en tenant compte de la configuration du système dans lequel le décodeur travaille, des caractéristiques du codeur convolutif utilisé ainsi que de l'exploitation de méthodes d'adaptation du débit de transmission. Un exemple d'une telle approche est le coprocesseur Viterbi du *DSP* TMS320C6416 de Texas Instruments [Spru533].

Il faut remarquer que le degré de souplesse est déterminé par la représentation numérique des métriques de branches (nombre de bits à disposition et interprétation des valeurs).

Sélection des chemins survivants (*ACSu*)

Les deux paramètres principaux sont le nombre d'unités *ACS* et la représentation numérique des métriques de branches (Figure C-6).

Selon l'approche de sélection parallèle des chemins survivants, le nombre d'unités de sélection ACS détermine la longueur de contrainte maximale des codes convolutifs supportables. L'adaptation aux autres contextes de codage moins exigeants (longueur K inférieure) peut s'effectuer par l'introduction d'un bloc fonctionnel supplémentaire, qui se charge de la bonne gestion des ressources. Sa tâche est l'activation des unités *ACS* effectivement impliquées dans le décodage ainsi que l'assignation correcte des vieilles métriques cumulées (Figure C-6).

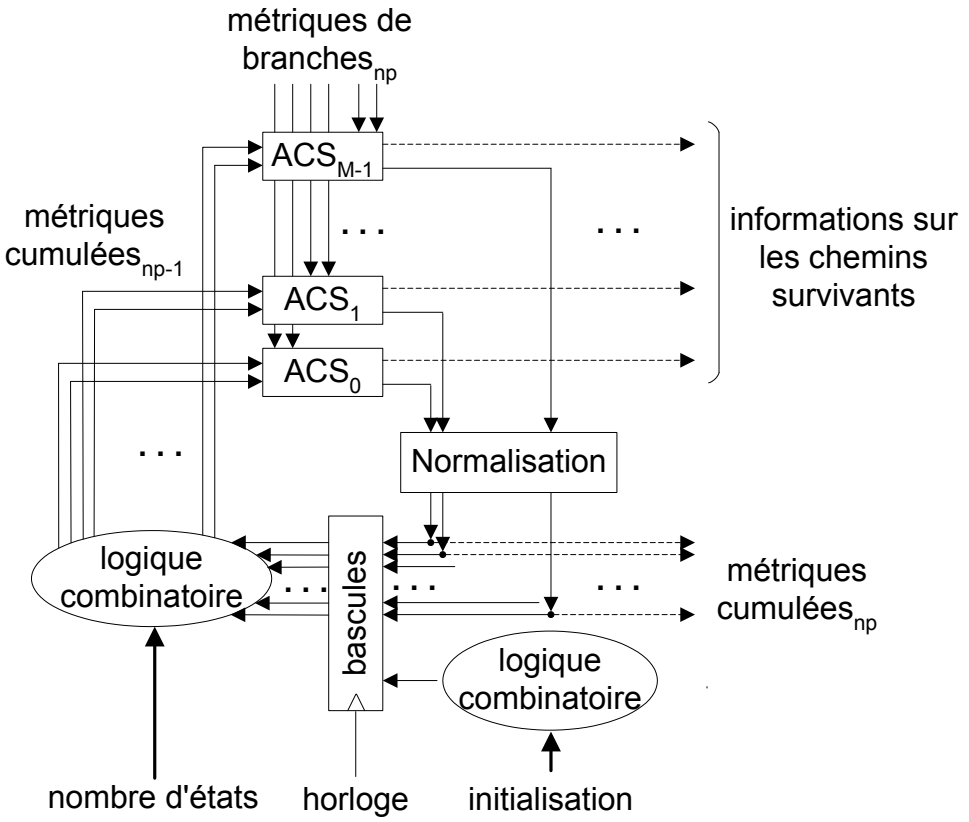

Figure C-6: exemple de structure parallèle utilisable pour une approche souple.

La représentation numérique des métriques cumulées doit être établie en considérant la représentation des métriques de branches et le nombre maximal d'états M que le système peut traiter (Sous-section 7.3.4). L'opération de normalisation des métriques cumulées garantit le déroulement correct de l'algorithme, tout en limitant le nombre de bits de la représentation numérique.
La réalisation de ce module doit aussi permettre l'initialisation des valeurs des métriques cumulées selon les exigences de chaque contexte de codage, ce qui améliore la capacité d'adaptation (exemple [Spra750]).

Désignation du chemin le plus favorable (*BPSu*)

La réalisation physique de ce module dépend principalement du type de critère utilisé pour la désignation du chemin (chemin le plus probable ou un chemin quelconque).
Sa conception doit maintenant considérer une situation caractérisée aussi bien par un nombre variable de chemins survivants que par des conditions d'arrêt différentes de la procédure de codage (exemple: Figure C-7).

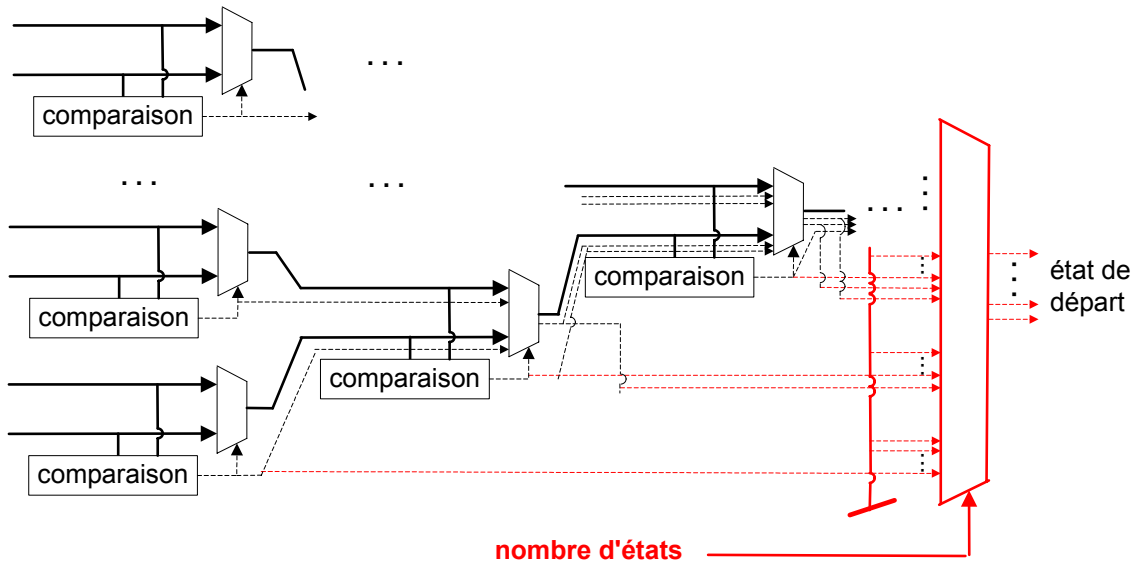

Figure C-7: réalisation du module exécutant la sélection du meilleur chemin, en tenant compte d'un nombre variable d'états. Les modifications apportées par rapport à la réalisation du module *BPS* de base (Figure 7-18) sont signalées en rouge.

L'exploitation exhaustive des conditions d'arrêt complique sérieusement la réalisation physique de ce module, surtout si l'on admet que les valeurs des bits de terminaison peuvent varier selon le standard de communication.
Par conséquent, on peut envisager la solution pragmatique d'une exploitation partielle de ces informations: l'utilisation de l'état final correct pour le décodage des derniers bits d'information. Cette solution tire profit de la présence des bits de terminaison, tout en limitant la complexité de la réalisation physique des modules.

Prise de décision (*TBu*)

La conception de ce module *TBu* doit considérer cinq paramètres principaux: la longueur de contrainte du code visé, la longueur maximale supportée par le système, la distance δ_p, la livraison des résultats au *DSP*, ainsi que l'éventualité d'un contexte de codage impliquant une séquence limitée de symboles.

Longueurs de contrainte des codes convolutifs visés

La complexité maximale (c'est-à-dire la longueur maximale de contrainte) des codes convolutifs que le système est censé décoder détermine les ressources de stockage et de traitement nécessaires au module *TBu*. La conception de ces module doit ensuite considérer la gestion correcte des ressources de stockage

et de reconstruction d'états des chemins, afin d'exécuter la tâche de reconstruction conformément au contexte de codage. L'objectif peut être atteint à l'aide d'une opération de masquage qui permet l'ajustement de la taille de la mémoire reconstruite, sans modifier la conception du système de base.

Distance δ_p

La valeur de ce paramètre dépend strictement de la longueur de contrainte du code convolutif envisagé. Dans ce contexte d'une approche souple coprocesseur, la distance optimale est ainsi potentiellement variable. Deux approches paraissent intéressantes: l'utilisation soit d'une distance optimale par rapport à la longueur *K* maximale, soit d'une distance de compromis.

La première approche assure la qualité de protection offerte par tous les codes convolutifs exploitables. Si la gamme des codes est très large (en termes de longueur de contrainte), l'utilisation d'une distance δ_p adaptée au contexte de codage le plus exigeant nécessite des coûts importants: l'usage de nombreuses ressources ainsi qu'un fort retard algorithmique.

Si les applications envisagées ne tolèrent pas ces coûts, une distance δ_p de compromis doit être établie, en analysant l'importance des coûts d'emploi, parallèlement à la dégradation de la qualité de protection dans les divers contextes de codages.

Modalité de livraison des décisions prises

Si le *DSP* demande la livraison de plusieurs bits décodés à la fois, la méthode de la prise de décision (Figure 7-16) peut être développée et optimisée de manière à extraire simultanément plusieurs valeurs (Table C-2 et Figure C-8) [Fett90] [Min91].

Dans ce cas, l'exécution de la tâche entière de décodage peut être modifiée en exécutant l'opération de prise de décision multiple par intervalle de temps (chaque *N* périodes, Table C-2). Cette opération est ainsi chargée de la prise de décision simultanée de *N* bits d'informations (Figure C-8).

Paramètre	Valeur
Intervalle de temps entre les opérations de décodage [nombre de périodes]	N
Nombre de bits délivrés [bits]	N
Retard algorithmique minimal [nombre de périodes]	δ_p+N-1
Ressources minimales de stockage [nombre d'informations]	$M\cdot(\delta_p+N-1)$
Efficacité: [nombre d'opérations de reconstruction d'état / nombre de bits délivrés]	$(\delta_p+N-1) / N$

Table C-2: vue d'ensemble des caractéristiques liées à la conception de la tâche de décodage.

Une évaluation des points forts d'une prise de décision multiple est illustrée par la Table C-2. Cette table permet de déterminer facilement les performances et les caractéristiques de la prise de décision.

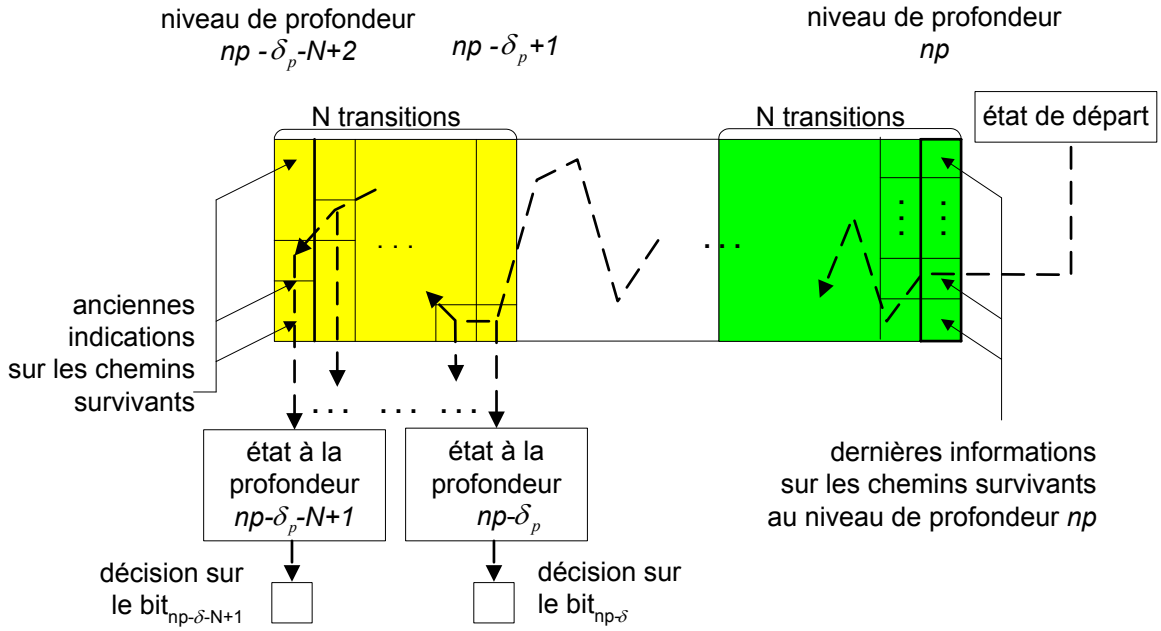

Figure C-8: exemple de procédure pour une prise de décision multiple. La couleur jaune représente les valeurs qui ne sont plus nécessaires pour la prochaine procédure de prise de décision. En vert, les nouvelles valeurs qui ont été sauvegardées depuis la dernière procédure. La distance δ_p est comprise entre les valeurs δ et $(\delta-K+2)$.

Décodage d'un message formé par un nombre fini de symboles

Le coprocesseur peut être confronté à des contextes de codage impliquant la protection de messages formés par un nombre fini de symboles. Dans cette

éventualité, la signalisation de la fin du message et l'état d'arrêt de la procédure de codage doivent être transmis à ce module. A l'aide du chemin survivant lié à cet état d'arrêt, ce module peut ainsi effectuer une prise de décision optimale des (*δ-nombre de bits de terminaison*) bits restants du message.

C.3 Références

[Fett90] G. Fettweis, *Parallelisierung des Viterbi-Decoders: Algorithmus und VLSI-Architektur*, rapport de recherche (Fortschritt-Berichte), série 10: Informatik/ Kommunikationstechnik, VDI-Verlag, Düsseldorf, 1990.

[Gath02] *The Application of Programmable DSPs in Mobile Communications*, édité par A. Gatherer et E. Auslander, John Wiley and Sons, Grande-Bretagne, 2002.

[Min91] B. –K. Min, *Architecture VLSI pour le decodeur de Viterbi*, Thèse de Doctorat, Ecole Nationale Supérieure des Télécommunications, Paris 1991.

[Spra750] Texas Instruments, *Using TMS320C6416 Coprocessors: Viterbi Coprocessor (VCP)*, *Application Report*, document SPRA750, juin 2001.

[Spru533] Texas Instruments, *Viterbi Decoder Coprocessor User's Guide*, document SPRU533, mai 2001.

[Ts25212] 3GPP, *Multiplexing and Channel Coding (FDD)*, document 3GPP TS 25.212, version 3.2.0.

Curriculum Vitae

Davide Manetti

Originaire de Camignolo, Suisse;
Né le 15 août 1972.

Domaines de recherche:

- Systèmes de navigation satellitaires GPS et Galileo.
- Codage de canal: conception d'algorithmes pour implantation sur *DSP* et réalisation de circuits *ASIC* pour appareils mobiles *UMTS*.
- Compression de signaux audio: conception d'algorithmes et implantation sur *DSP*.
- Techniques de traitement numérique destinées à des implantations en temps réel sur *DSP*.

Formation:

1991-1997: Ecole Polytechnique Fédérale de Zürich (ETHZ),
Diplôme d'ingénieur en électronique.

Expérience:

Depuis 1997: **Assistant de recherche**
auprès du Laboratoire d'Electronique et de Traitement du Signal, Institut de Microtechnique (IMT), Université de Neuchâtel (Uni-NE).

Depuis 1998: Définition et gestion de projets d'étudiants (stages, travaux de semestre et de diplôme) pour des étudiants provenant de l'Université de Neuchâtel, de l'Ecole Polytechnique Fédérale de Lausanne et d'autres universités européennes.

Depuis 2001: **Doctorant**
Au sein du Laboratoire d'Electronique et de Traitement du Signal, IMT, Uni-NE.

Décembre 2004: **Soutenance de la thèse de doctorat.**

Publications

Articles de conférences:

D. Manetti, A. Dufaux, E. Meurville, M. Ansorge, F. Pellandini, P. Ryser, D. Wieser, "Audio Coder for Telesurveillance Applications with Real Time Implementation on a Multimedia DSP", *Proceedings of COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, Neuchâtel, Switzerland, 5-7 May 1999, pp. 276-281.

Autres publications:

D. Manetti, *Audio Coder for Remote Surveillance Application*, internal report (confidential), Institute of Microtechnology, University of Neuchâtel, Neuchâtel, Switzerland, August 1999.

D. Manetti, *Channel Coding in Wireless Applications,* internal report (confidential), Institute of Microtechnology, University of Neuchâtel, Neuchâtel, Switzerland, May 2001.

L. Grasso, D. Manetti, *Implementation of a parameterized family of Viterbi Decoders*, internal report (confidential), Institute of Microtechnology, University of Neuchâtel, Neuchâtel, Switzerland, December 2001.

Activités professionnelles

Présentation invitée:

« Evolution of audio coding techniques”**,** presented at *workshop* ***“Solutions numériques pour la vidéo et les télécommunications ?”***, ***ATEME – TI – EPFL*****,** Lausanne, Switzerland, 20 January 2004.

Poster:

Atelier LEA**,** « Le codage de canal utilisé dans les systèmes de communication mobile *UMTS*”**,** Arc et Senans, France, 29-30 septembre 2003.

Révision d'articles:

SCS 2003*, IEEE International Symposium on Signal, Circuits & Systems*, Iasi, Romania, 10-11 July 2003.

EUSIPCO 2002*, XI European Signal Processing Conference*, Toulouse, France, 3-6 September 2002.

SCS 2001*, IEEE International Symposium on Signal, Circuits & Systems*, Iasi, Romania, 10-11 July 2001.

COST 254*, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, Neuchâtel, Switzerland, 5-7 May 1999.

Connaissances informatiques:

Langages de programmation:

C/C++, Assembleur, Matlab.

Conception matérielle:

HDL Designer, Modelsim, Synopsys, Quartus.

Langues:

Italien, Français, Allemand, Anglais.