| | |
|---|---|
| **Titre:** Title: | Algorithmes et architecture de décodeur séquentiel |
| **Auteur:** Author: | Jean Belzile |
| **Date:** | 1990 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Belzile, J. (1990). Algorithmes et architecture de décodeur séquentiel [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58298/ |

UNIVERSITE DE MONTREAL

# ALGORITHMES ET ARCHITECTURES DE DECODEUR SEQUENTIEL

par

JEAN BELZILE
DEPARTEMENT DE GENIE ELECTRIQUE
ECOLE POLYTECHNIQUE

MEMOIRE PRESENTE EN VUE DE L'OBTENTION
DU GRADE DE MAITRE ES SCIENCE APPLIQUEES (M.SC.A.)
(GENIE ELECTRIQUE)

mai 1990

ISBN 0-315-58920-0

Canada

UNIVERSITE DE MONTREAL

ECOLE POLYTECHNIQUE

Ce mémoire intitulé

ALGORITHMES ET ARCHITECTURES DE DECODEUR SEQUENTIEL

présenté par: Jean Belzile
en vue de l'obtention du grade de: M.Sc.A.
a été dûment accepté par le jury d'examen constitué de:

| | |
|---|---|
| M. Yvon Savaria | Ph.D., Président |
| M. David Haccoun | Ph.D., Directeur de recherche |
| Mme. Catherine Rosenberg | Doc. Sc., Membre |

# SOMMAIRE

Le décodage séquentiel est une technique heuristique de recherche d'un chemin dans un arbre d'encodage. Bien que l'effort moyen de calcul soit peu élevé, il est néanmoins extrêmement variable. Dans le but d'obtenir une réalisation matérielle pratique de l'algorithme on doit réduire ou même éliminer totalement cette variabilité sans toutefois sacrifier les performances d'erreur.

Afin d'obtenir une réduction de la variabilité de l'effort de calcul du décodage séquentiel, le développement d'algorithmes et d'architectures bidirectionnelles alliées à un algorithme sous-optimal de recherche en largeur sera étudié.

Lors du décodage séquentiel, la présence de salves de bruit cause la variabilité de l'effort de calcul ou la perte du chemin correct. Bien qu'ayant l'avantage d'éliminer complètement la variabilité de l'effort de calcul, les techniques de recherche en largeur souffrent d'une faible performance d'erreur. Ces techniques, étant sous-optimales, perdent le vrai chemin lors de passages fortement bruités. Les techniques bidirectionnelles convergent sur ces salves réduisant ainsi leur impact sur le reste de l'information. L'utilisation de ces deux techniques permet de bonnes performances d'erreur sans aucune variabilité de l'effort de calcul.

Une étude de réalisation ITGE est présentée pour l'algorithme M-chemins. Une approche parallèle utilisant un réseau de tri sous-optimal et des communications sérielles permet une réalisation sur une puce et des vitesses de décodage de l'ordre des dizaines de millions de bits par seconde.

# ABSTRACT

Sequential decoding is a heuristic tree searching algorithm. While the average computational effort is low, it is extremely variable. Practical decoders however need to reduce or eliminate this computational variability without reducing the error performance.

In order to obtain a reduction in computational variability, the development of bidirectional algorithms and architectures coupled with a sub-optimal search procedure will be studied.

In sequential decoding it is the presence of error bursts that cause computational variability or correct path loss. While eliminating computational variability, breadth first search techniques suffer from poor error performance. These techniques, being sub-optimal, tend to loose the correct path in noisy regions. Bidirectional techniques converge on those regions reducing their impact on the remaining information. The combination of these two techniques permits excellent error performances without any computational variability.

A VLSI implementation for the M-algorithm is presented. A parallel approach using a sub-optimal self sorting network and serial communications permits a single chip realisation and decoding speeds in the order of tens of millions of decoded bits per second.

# REMERCIEMENTS

Je tiens à remercier mon directeur Dr. David Haccoun pour son aide, ses nombreux conseils et son support constant ainsi que Dr. Yvon Savaria pour ses nombreux commentaires et suggestions au niveau de la réalisation pratique de ces algorithmes.

Il serait injuste de passer sous silence les contributions de Guy Bégin et de Pierre Montreuil, ces derniers ont rendu possible plusieurs aspects importants de ce mémoire notamment l'approche bidirectionnelle.

Des remerciements sont également dus à tous mes camarades de laboratoire qui ont contribué à une atmosphère de travail jovial. L'aide apportée par François Gagnon, Jean-François Huard, Nicolas Arel, Zhi-Wei Rong, Claude Thibeault, Pierre Lavoie et Nathalie Dubreuil ne peut être passée sous silence.

Des remerciements doivent également être étendus aux étudiants qui ont contribué à ce mémoire en travaillant sur des réalisations matérielles, notamment les piles et le décodeur multi-chemins.

# TABLE DES MATIERES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# CHAPITRE 1
# INTRODUCTION

Les systèmes de communications numériques modernes exigent des vitesses et une fiabilité de plus en plus élevées [1]-[5]. Afin d'atteindre ces objectifs, les systèmes de communications numériques utilisent des techniques de codage. Ces techniques se classent en deux catégories: Les techniques de détection d'erreurs et de retransmission (ARQ, Automatic Repeat Request), et celles de correction d'erreurs (FEC, Forward Error Correction).

Un système ARQ offre une très faible probabilité d'erreur non-détectée, est très efficace sur les canaux réels et n'est ni très coûteux ou complexe. Cependant, un canal de retour doit être utilisé, le délai de décodage est variable et des problèmes de mise en ordre des trames décodées sont présents.

Les systèmes FEC n'ont pas besoin de canal de retour, et ont un débit effectif constant. Le délai est fonction du décodeur. Par contre, ces systèmes présentent deux inconvénients; une expansion de la largeur de bande nécéssaire à la transmission de l'information ainsi que l'utilisation de décodeurs complexes sensibles aux dégradations du canal.

Ces deux techniques peuvent être combinées pour faire une troisième classe de codage: les systèmes hybrides. Ces systèmes utilisent à la réception un code correcteur d'erreurs suivi d'un code détecteur d'erreurs (FEC/ARQ). Le code correcteur d'erreurs minimise le nombre de retransmissions et améliore le délai au prix d'une augmentation de la complexité et d'une expansion de la largeur de bande.

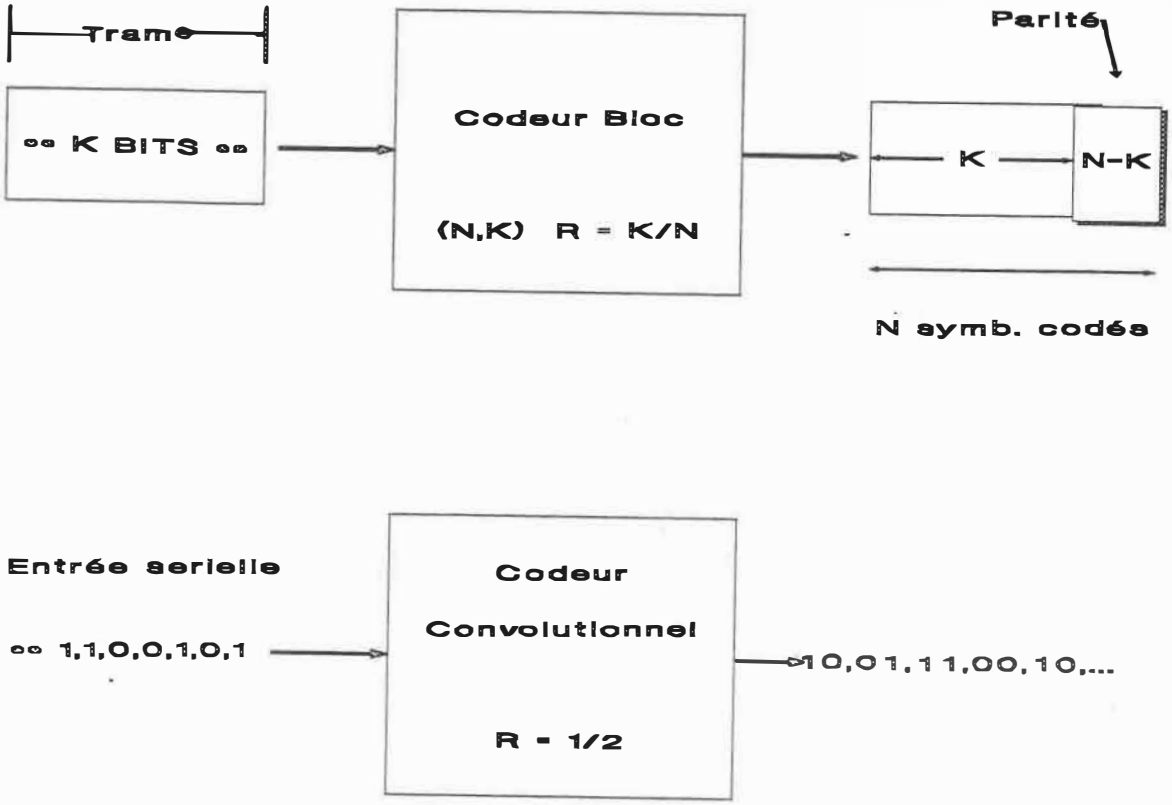


Figure 1.1 Techniques de codage correcteur d'erreurs. Le codage bloc et le codage convolutionnel.

Parmi les systèmes FEC, deux méthodes d'encodage de l'information sont utilisées: le codage bloc et le codage convolutionnel (figure 1.2). Le codeur bloc accepte un message complet (K bits), calcule un certain nombre (N-K) de bits de parité puis transmet le message original suivi des bits de parité calculés. Le code spécifie le nombre de bits de parité et sur quels bits d'information est basé chacun de ces bits.

Le codage convolutionnel accepte les messages bit par bit. Chaque bit est à son tour transformé en plusieurs symboles (2 dans la figure 1.2). Les bits d'information peuvent être ou ne pas être présents dans les symboles transmis selon le code utilisé.

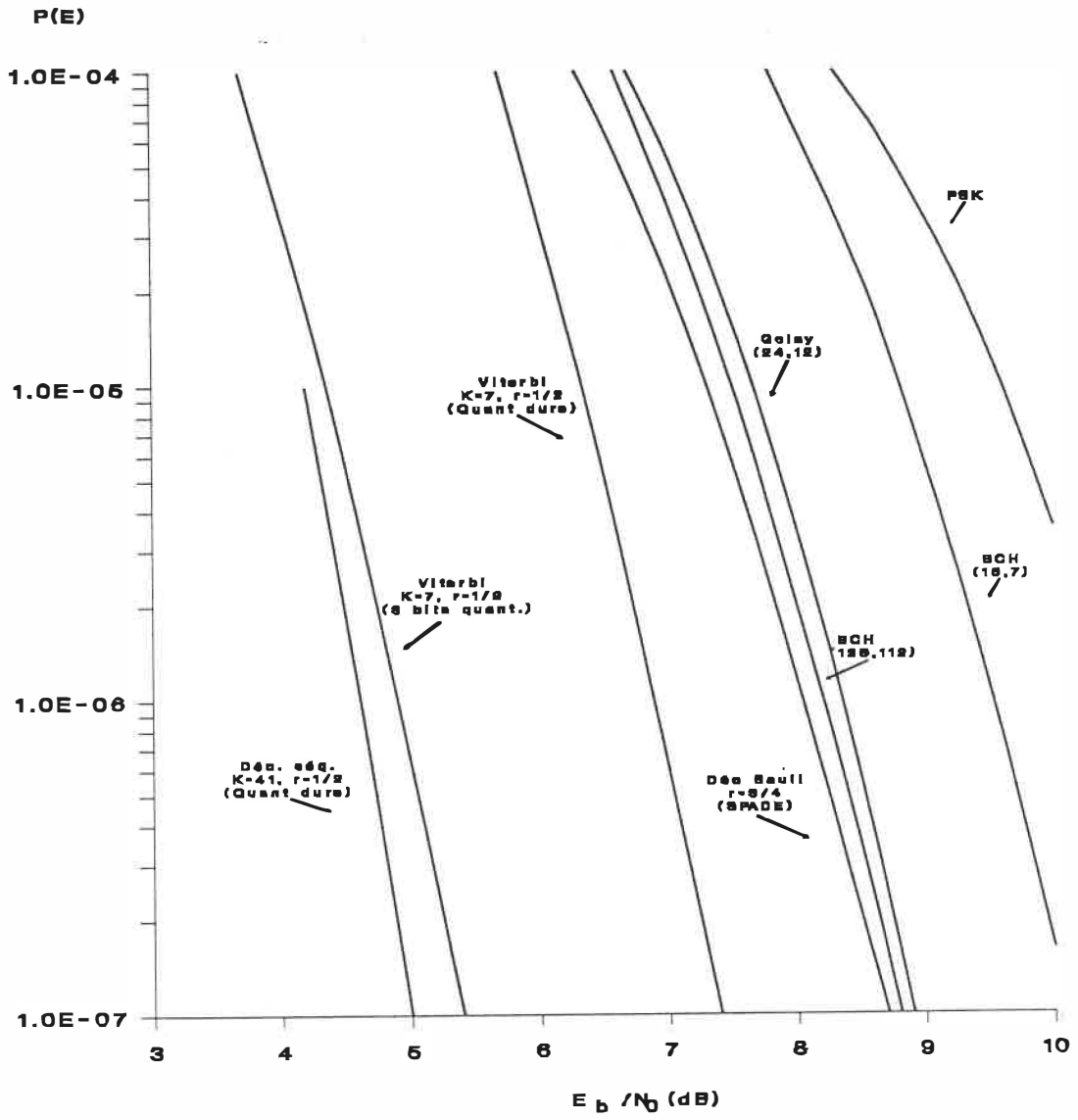


Figure 1.2 Performance d'erreur de quelques systèmes codés.[1]

La figure 1.2 tirée de [1] montre les performances atteintes par divers systèmes de codage . Les 4 courbes inférieures sont obtenues par codage convolutionnel, les 3 suivantes utilisent des codes blocs et finalement, la dernière n'utilise pas de technique de codage.

Le codage convolutionnel offre un meilleur pouvoir correcteur d'erreurs que le codage bloc au prix d'une complexité de décodage plus importante. Ce mémoire porte sur le problème de la réduction de la complexité des décodeurs convolutionnels.

La complexité des algorithmes de décodage est la contrainte principale du codage convolutionnel. Les algorithmes existant demandent une imposante quantité de calculs et/ou de mémoire. De plus certains démontrent une grande variabilité de l'effort de calcul [6]. Bien que les systèmes de communications numériques actuels s'accommodent de ces défauts, les systèmes de communications numériques de l'avenir nécessiteront de meilleurs algorithmes de décodage.

Le but principal de cette recherche est de développer des algorithmes et architectures permettant la réalisation de puissants décodeurs pour codes convolutionnels. Ces algorithmes et architectures doivent être de complexité réduite sans pour autant sacrifier la puissance correctrice du code. La réduction ou l'élimination de la variabilité de l'effort de calcul, sans perte de fiabilité et sans l'introduction d'un grand nombre de calculs, est la première priorité de ce mémoire. Le développement d'algorithmes et d'architectures permettant l'obtention de tels résultats permettera l'augmentation des vitesses de transmissions et l'amélioration de la fiabilité des systèmes de communications numériques.

## 1.1 Contributions

Les recherches, effectuées dans le but d'obtenir une réduction de la complexité des décodeurs de codes convolutionnels, ont permis les contributions suivantes:

La réalisation ITGE (Intégration à très grande échelle) d'une pile systolique à entrées parallèles pour un décodeur séquentiel à pile.

Une pile systolique à entrées parallèles utilisant moins de comparateurs que la pile standard à entrées parallèles.

Une pile semi-systolique utilisant un tri par insertion, obtenant un tri parfait en temps constant.

Une architecture de décodeurs multi-chemins pouvant être utilisée pour des codes ayant une grande longueur de contrainte.

Une technique de décodage bidirectionnel permettant un important gain de codage sans variabilité de l'effort de calcul lorsqu'utilisée avec un algorithme multi-chemins.

Les spécifications et cellules de base pour une réalisation ITGE (Intégration à très grande échelle) de la machine multi-chemins.

## 1.2 Organisation du Mémoire

Cette recherche apporte des solutions à plusieurs problèmes posés par le décodage séquentiel. Le chapitre 2 introduit les principes du codage ainsi que les notions essentielles à la compréhension et à la comparaison des architectures proposées.

Le chapitre 3 couvre les différents algorithmes de décodage ainsi que leurs avantages et désavantages. Les comparaisons sont effectuées selon le pouvoir correcteur du code et selon la variabilité de l'effort de calcul des algorithmes.

Le chapitre 4 présente des approches multiprocesseurs au problème critique de la mise en ordre de la pile dans l'algorithme de décodage séquentiel de Zigangirov-Jelinek (Z-J) classique ainsi que deux nouveaux algorithmes de tri pour la pile.

Le chapitre 5 propose un algorithme multi-chemins qui élimine la variabilité de l'effort de calcul inhérente au décodage séquentiel. L'introduction de la recherche bidirectionnelle de l'arbre d'encodage est également présentée au cours de ce chapitre.

Au chapitre 6, les problèmes rattachés à la réalisation de cet algorithme sont examinés et une solution ITGE est élaborée. Des solutions possibles sont également

définies pour la réalisation matérielle d'un système utilisant une technique de recherche bidirectionnelle.

Les conclusions et des recommandations en vue de recherches futures sont présentées au chapitre 7.

# CHAPITRE 2
# CODAGE CONVOLUTIONNEL

Au chapitre 1, il fut expliqué qu'un système FEC utilise le codage dans le but d'améliorer la performance d'erreur sans avoir recours à des retransmissions. Ce chapitre, dédié au codage, présente le rôle et le fonctionnement de cette technique de contrôle des erreurs.

L'amélioration de la performance d'erreur, fournie par l'emploi d'une technique de codage, est souvent mesurée par le gain de codage (figure 2.1). Le gain de codage est défini comme la différence en dB des rapports signal à bruit pour une probabilité d'erreur donnée entre le système codé et un système sans codage utilisant une modulation PSK cohérente parfaite [1]. Ainsi à la figure 2.1, le gain de codage asymptotique illustré est de 5 dB pour une probabilité d'erreur de $P_E = 10^{-5}$.

L'utilisateur d'un système de communications numériques peut convertir le gain de codage en une combinaison des facteurs suivants: augmentation de la fiabilité, augmentation de la vitesse de transmission ou réduction de l'énergie requise par le système. Dans plusieurs systèmes de communications numériques, notamment les communications par satellites, un gain de vitesse et une réduction de l'énergie de transmission sont des facteurs importants dans le coût du système. Le codage devient alors capital dans plusieurs systèmes de communications.

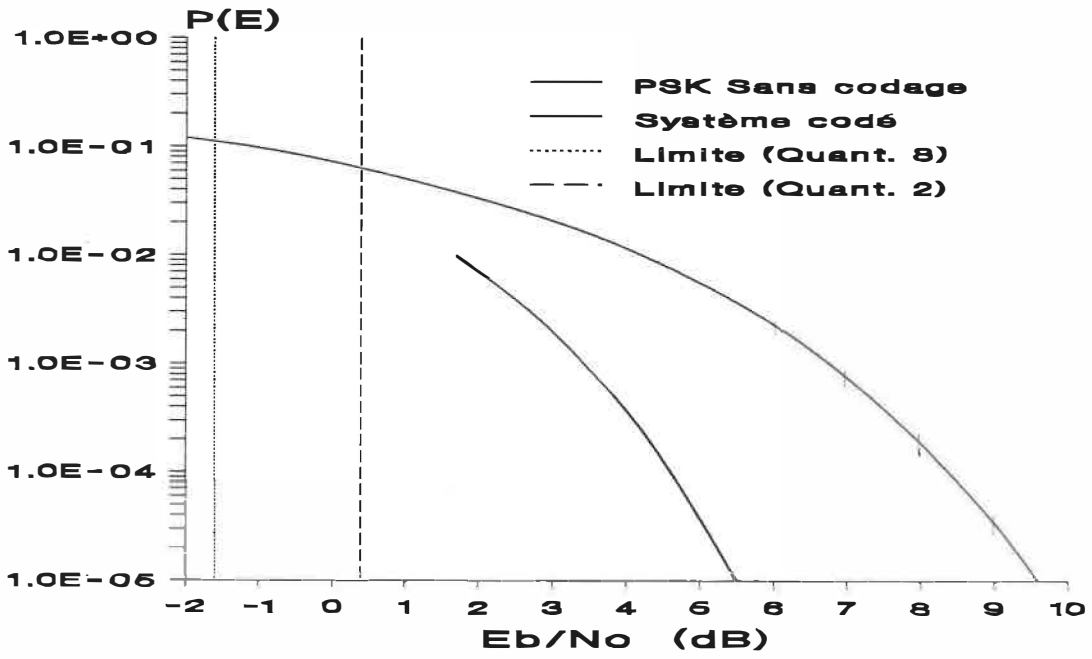


Figure 2.1 Gain de Codage.

## 2.1 Système Codé

Un système de communications numériques (FEC) utilisant une technique de codage est représenté à la figure 2.2. Ce système comprend: source, codeur, modulateur, émetteur, canal physique, récepteur, démodulateur, décodeur, et évidemment la destination. Le bruit est introduit par le canal physique.

La source et la destination ne sont que les usagers du système. La source émet l'information à être transmise à la vitesse de Rs bits par seconde; la destination reçoit cette information au même rythme.

Le codeur accepte le message de la source et l'encode. L'information codée est émise par le codeur à la vitesse de Rc symboles par seconde. Le décodeur effectue l'opération inverse, acceptant Rc symboles par seconde et délivrant Rs bits par seconde.

Etant donné que le système de codage vise une augmentation de la fiabilité, Rc se doit d'être plus grand que Rs. Le taux de codage, R, est ainsi définit par R = Rs/Rc. Plus le taux de codage est faible plus il y aura de redondance dans le signal transmis et, par conséquent, plus il sera possible d'obtenir un gain de codage élevé.

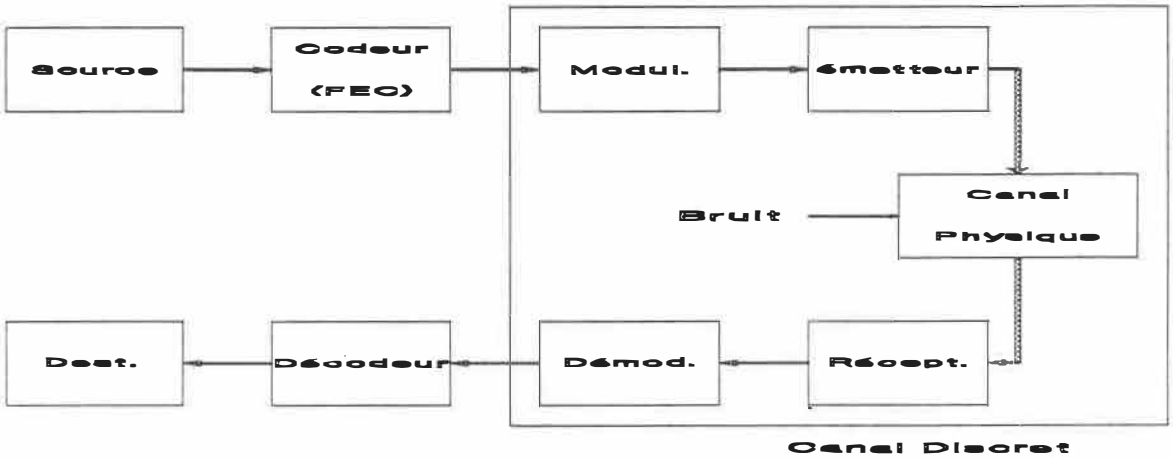


Figure 2.2 Système de communication utilisant une technique de codage et de décodage.

Comme démontré à la figure 2.2, il est possible de combiner ensemble le modulateur, l'émetteur, le canal physique, le récepteur et le démodulateur et de modéliser leurs effets par un canal discret (numérique). Cette opération rend l'analyse du système plus aisé.

Ce nouveau canal possède Q entrées et J sorties, la probabilité de l'entrée i est q(i). Les probabilités des transitions sont fonctions du bruit dans le canal physique et du système de modulation utilisé. Si l'entrée est i, la probabilité d'obtenir une sortie j est donnée par P[i | j]. Un exemple binaire de ce nouveau canal est illustré à la figure 2.3.

Se référant à la figure 2.3, Q = J = 2 et q(0) = q(1) = 1/2. Les probabilités de transitions p et q représentent la probabilité que le canal change la valeur d'un symbole, p = P[1 | 0] alors que q = P[0 | 1]. Si p = q alors le canal est dit symétrique (BSC, Binary

Symmetric Channel) et sa probabilité d'erreur est définie par la probabilité de transition p = P[1 | 0] = P[0 | 1].

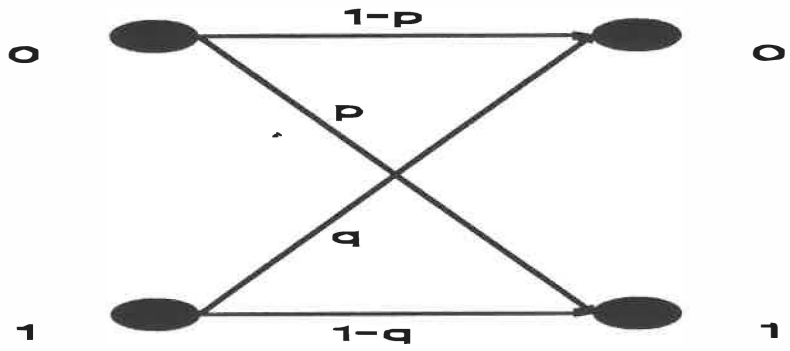


Figure 2.3 Canal Numérique Binaire.

Le bruit dans le canal est modélisé par un bruit gaussien blanc additif de densité spectrale $N_0$. Un bruit blanc est un bruit pour lequel le spectre de puissance est constant dans la bande de fréquences utilisées [1]. Ce type de bruit affecte de façon indépendante les symboles transmis.

Le rapport $E_b/N_0$ est utilisé pour évaluer les performances des systèmes codés. $E_b$ est l'énergie par bit d'information reçue au décodeur. Pour mesurer l'énergie par bit reçue au décodeur il suffit de mesurer l'énergie par symbole, $E_s$, et de diviser par le taux de codage, R.

## 2.2 Codeur Convolutionnel

Les codeurs convolutionnels, en plus d'offrir un important gain de codage (figure 2.1), possèdent une structure simple et régulière propre à une réalisation matérielle aisée.

Un codeur convolutionnel, illustré à la figure 2.4, peut être représenté par un registre à décalage, de longueur K, auquel V additionneurs modulo-2 sont connectés. Ce

sont les connexions entre les additionneurs et le registre à décalage qui spécifient le code utilisé. Toutes les connexions rattachées à un même additionneur modulo-2 composent un vecteur générateur. Ce dernier est spécifié en octal de gauche à droite, une connexion équivaut à un '1' et l'absence de connexion à '0'. L'ensemble des vecteurs générateurs définit le code. A la figure 2.4 le code utilisé est (7,5), soit 2 vecteurs générateurs ayant des valeurs respectives de 7 (111) et 5 (101).

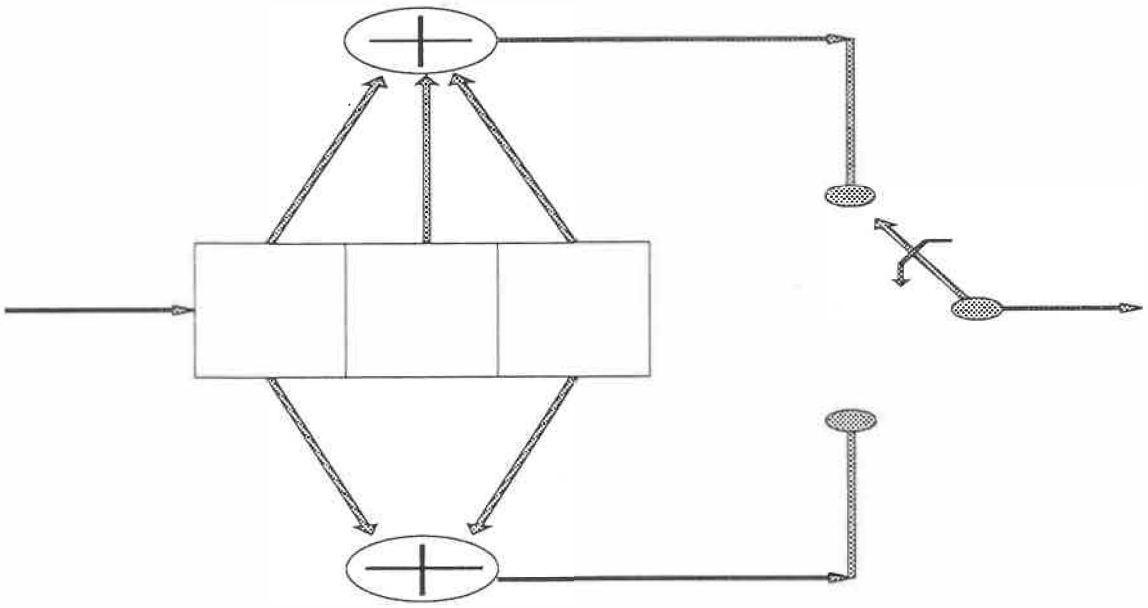

Figure 2.4 Codeur Convolutionnel code = (7,5), K = 3, v = 2.

La longueur de contrainte, K, d'un code est définie comme étant la longueur du registre à décalage. La mémoire du codeur, v, est définie comme les K-1 premières cellules du registre à décalage. A la figure 2.4 K = 3 et v = 2

L'algorithme d'encodage est extrêmement simple et répétitif.

1- Décaler les données du registre à décalage d'un bit vers la droite.
2- Insérer un nouveau bit d'information dans le registre à décalage.
3- Echantillonner à tour de rôle les V sorties des additionneurs.
4- Recommencer jusqu'à l'épuisement des données.

Algorithme 2.1 Algorithme d'encodage.

Il est possible de définir des codeurs acceptant plus d'un bit d'information à la fois. Un tel codeur utilise normalement plus d'un registre à décalage et ceux-ci ne sont pas nécessairement tous de même longueur. Le taux de codage est alors R = b/V, où b est le nombre de bits entrant dans les registres à décalage à chaque cycle de l'algorithme. Ces codeurs ne sont pas discutés dans ce mémoire.

### 2.2.1 Représentation par Diagramme d'état

Le codeur convolutionnel est une machine linéaire à nombre d'états finis. Par conséquent, l'action du codeur convolutionnel de taux R = 1/V est représentée, sous sa forme la plus compacte, par un diagramme d'état (figure 2.5). Le nombre d'états possibles du codeur est défini par la mémoire du codeur, soit $2^v$ états. Le nombre de chemins entrant et sortant de chaque état, ainsi que le nombre de symboles sur chacun de ces chemins, sont définis par le taux de codage. Utilisant un taux de codage de R = 1/V, 2 branches sont issues de chaque noeud et V symboles sont présents sur chacune d'entre elles.

Le diagramme d'état représenté à la figure 2.5, correspond au codeur de la figure 2.4. Ce diagramme est composé de 4 états, soit une mémoire de codeur v = 2 et une longueur de contrainte K = 3. Le taux de codage est R = 1/2.

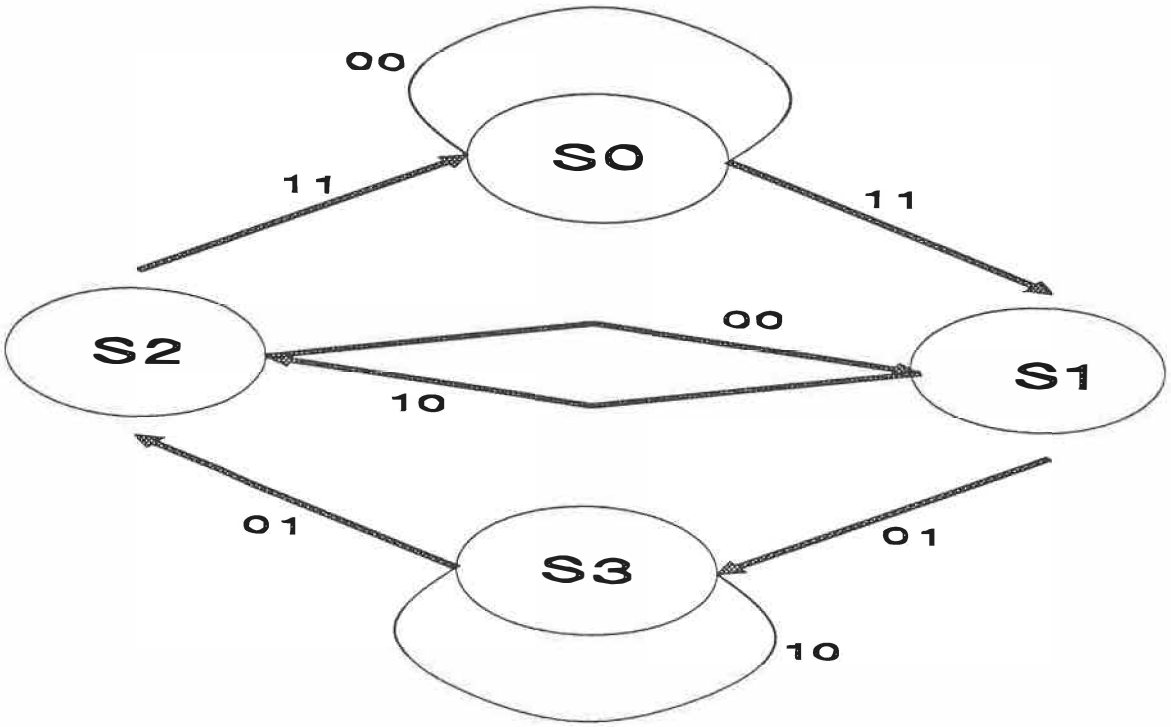


Figure 2.5 Diagramme d'état.

L'encodeur commence à l'état S0 (00). Si le nouveau bit à insérer est un 0, alors l'état demeure inchangé et le codeur émet les symboles 00. Si le nouveau bit d'information est un 1, alors l'état du codeur deviendra S1 (10) et les symboles transmis seront 11. A partir de l'état S1 (10), il est possible d'atteindre l'état S2 (01) ou S3 (11) selon la valeur du prochain bit d'information. A chaque transition, les symboles retrouvés sur les branches du diagramme d'état sont émis par le codeur.

Le diagramme d'état est un outil pratique pour la caractérisation des codes [1]. Dans de mémoire, cette caractérisation des codes n'étant que secondaire, elle n'est pas couverte ici.

## 2.2.2 Treillis d'Encodage

Comme on le verra au chapitre 3, le décodage d'un code convolutionnel consiste à trouver un chemin dans un graphe. Le diagramme d'état, bien que représentant l'action du codeur, n'est pas l'outil idéal pour ce genre de recherche. La représentation graphique de l'action du codeur doit tenir compte du temps de passage des bits d'information.

Un treillis est en fait un diagramme d'état modifié pour représenter le déroulement du temps (voir figure 2.6). Dans le treillis, les états sont maintenant appelés noeuds. Le treillis possède deux dimensions: la longueur et la largeur. La longueur représente le temps, nombre de bits d'information envoyés durant la transmission, et la largeur représente le nombre d'états possibles du codeur, $2^v$. Les chemins allant d'une profondeur à l'autre dans le treillis sont les mêmes chemins que ceux reliant les états dans le diagramme d'état.

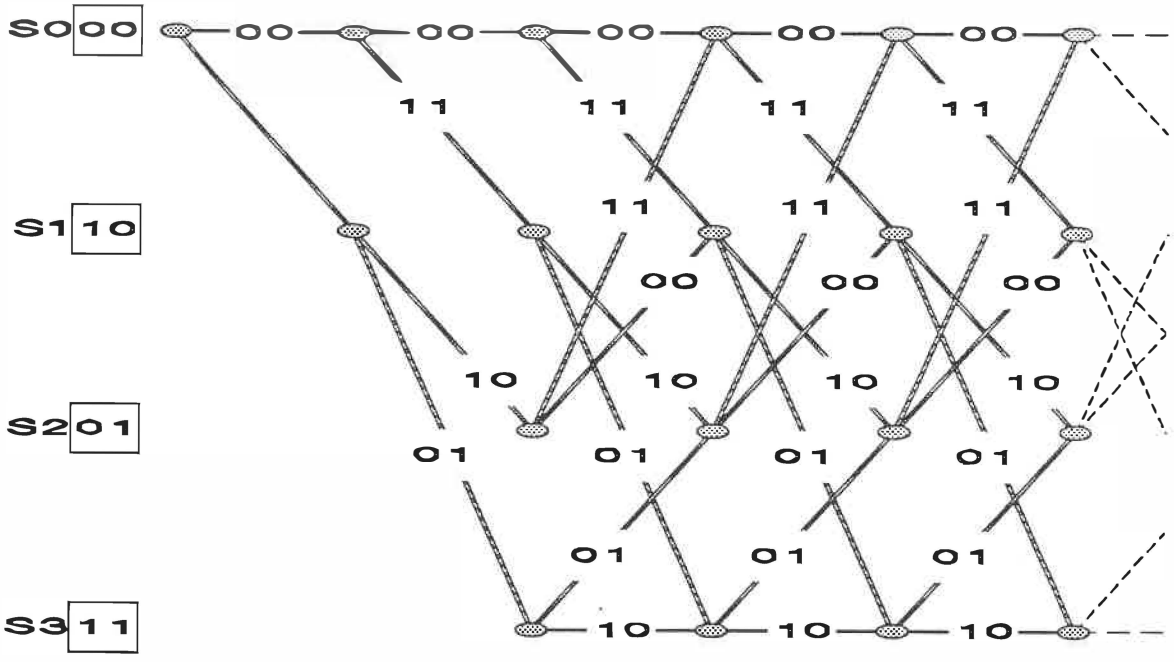


Figure 2.6 Treillis d'encodage.

Etant issu du diagramme d'état, le treillis respecte la même structure que celui-ci. Pour un code de taux R = 1/V, chaque noeud possède 2 branches avec V symboles sur chacune d'entre elles.

Le treillis exploite parfaitement la structure linéaire du codeur. Après une période de départ, le nombre de noeuds à chaque profondeur du treillis demeure constant et égal à $2^V$. Le treillis, représenté à la figure 2.6, tout comme le diagramme d'état de la figure 2.5, correspondent au codeur de la figure 2.4.

### 2.2.3 Arbre d'Encodage

Le treillis, n'est pas la seule représentation possible de l'action du codeur. On peut représenter l'action du codeur par un arbre (figure 2.7).

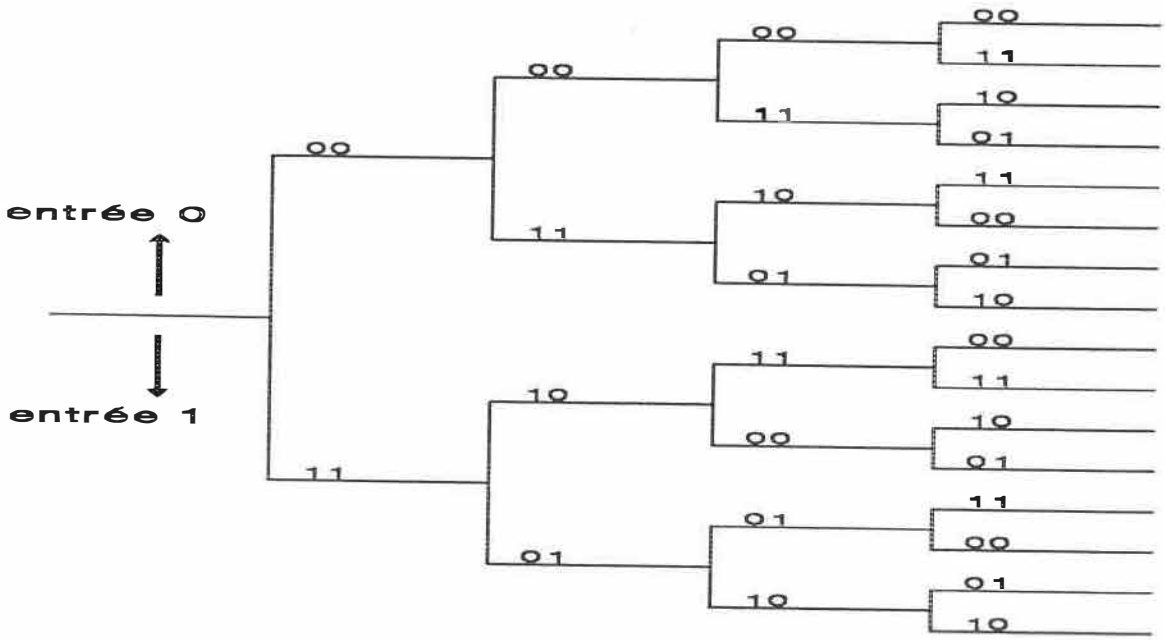


Figure 2.7 Arbre d'encodage.

Compte tenu que chacun des bits d'information entrant dans le codeur est un '1' ou un '0', un arbre de 2 branches peut être construit. Ainsi chaque noeud se sépare en 2 branches, et on retrouve sur chacune d'entre elle les V symboles générés par le codeur.

L'arbre possède une grande redondance. Si le premier bit d'information est '0', l'état du codeur est encore S0 (00) et l'arbre issu de ce premier bit d'information est identique à l'arbre original. Il en va de même pour tous les noeuds à diverses profondeurs pour lesquels il existe un autre noeud de même état.

L'arbre d'encodage, est une représentation plus naturelle de l'action du codeur mais elle est définitivement beaucoup plus coûteuse en termes de chemins à explorer. Pour un message de L bits d'information et un taux de codage R = 1/V, il y a $2^L$ branches (chemins possibles) à l'extrémité de l'arbre d'encodage.

## 2.3 Evaluation de la Performance des Codes

Les codes convolutionnels de même taux, R, sont normalement classés selon leurs longueurs de contrainte, K. En général, plus K est élevé, plus le code possède un grand pouvoir correcteur. Parmi les codes de même taux et de même longueur de contrainte, plusieurs critères sont utilisés pour séparer ces codes entre eux. Les caractéristiques normalement utilisées sont la distance minimale ($d_{min}$), la distance libre ($d_{free}$), le spectre et la fonction de distance de colonne (FDC).

La représentation graphique, utilisée pour représenter l'action du codeur, dicte l'emploi de techniques de recherche différentes. L'analyse de ces techniques de recherche ne requiert pas toutes les caractéristiques, certaines sont plus ou moins importantes [7]. Les deux caractéristiques retenues sont la distance libre ($d_{free}$) et la fonction de distance de colonne (FDC).

Soit le message $\underline{U}$:

$$\underline{U} = (U_1, U_2, \ldots, U_L) \tag{2.1}$$

où $U_i = \{0,1\}$ et correspond aux bits du message. On définit ensuite les symboles codées du message $\underline{U}$ par $\underline{X}(\underline{U})$:

$$\underline{X(U)} = (x_1^1, x_1^2, \ldots, x_1^V, x_2^1, \ldots, x_2^V, \ldots, x_L^1 \ldots, x_L^V) \tag{2.2}$$

où $X_i^j$ est le $j^{ième}$ de V symboles issus du bit $U_i$.

La distance de Hamming entre deux mots de code provenant de deux messages indépendants $\underline{U}$ et $\underline{V}$ est définie comme le nombre de symboles différents entre eux.

$$\begin{aligned} d_H &= \text{distance de Hamming} \\ &= \sum_{i=1}^{L} \sum_{j=1}^{V} x_i^j(\underline{U}) \oplus x_i^j(\underline{V}) \end{aligned} \tag{2.3}$$

où $X_i^j(\underline{U})$ est le $j^{ième}$ symbole codé correspondant au $i^{ième}$ bit d'information du message $\underline{U}$. De façon similaire, $X_i^j(\underline{V})$ correspond au $j^{ième}$ symbole codé du $i^{ième}$ bit d'information du message $\underline{V}$. Le symbole $\oplus$ représente l'addition modulo-2.

La fonction de distance de colonne FDC est alors définie comme la distance de Hamming minimale entre les symboles codés des n premiers bits d'information de deux messages $\underline{U}$ et $\underline{V}$ en supposant que leur premier bit est différent.

$$\begin{aligned} FDC(n) &= \text{fonction de distance de colonne d'ordre n} \\ &= \min \sum_{i=1}^{n} \sum_{j=1}^{V} x_i^j(\underline{U}) \oplus x_i^j(\underline{V}) \end{aligned} \tag{2.4}$$

Le minimum est pris sur toutes les paires de messages possibles $\underline{U}$ et $\underline{V}$ de longueur n dont le premier bit est différent. Il est possible de définir le profil de distance d'un code (PDC) comme étant:

$$PDC = FDC(1), FDC(2), .., FDC(K) \tag{2.5}$$

Le profil de distance de colonne du code K = 20 (2451321,3546713) [8] est (2,3,3,4,4,5,5,6,6,6,7,7,8,8,8,8,9,9,9,10). En général, il est préférable que le profil de

distance de colonne croisse le plus rapidement possible, car les chemins auront ainsi tendance à se démarquer les uns des autres plus rapidement.

La distance libre, $d_{free}$, représente la distance de Hamming, $d_H$, minimale entre deux chemins ayant divergé sur une longueur arbitrairement grande.

$$d_{free} = \lim_{n \to \infty} FDC(n) \qquad (2.6)$$

Plus la valeur de $d_{free}$ d'un code est élevée moins les mots se ressemblent, par conséquent une grande valeur de $d_{free}$ est préférable.

Les caractéristiques présentées ici sont utiles pour comprendre le comportement du décodeur et prédire la puissance d'un code mais l'ultime mesure de performance d'un code demeure sa probabilité d'erreur. Il existe principalement deux mesures de la probabilité d'erreur. La probabilité d'erreur par message, $P_E$ et la probabilité d'erreur par bit $P_B$. La probabilité d'erreur par message, $P_E$, correspond à la probabilité d'avoir au moins un événement erreur dans un message. La durée de l'événement erreur n'est pas importante. De façon théorique [1], pour le canal binaire symétrique de probabilité de transition p (figure 2.3), cette probabilité correspond approximativement à:

$$P_E \approx a[2\sqrt{p(1-p)}]^{d_{free}} \qquad (2.7)$$

où a correspond au nombre de chemins différents du chemin correct après $d_{free}$ branches si la première branche était différente. Cette valeur est spécifique au code utilisé [1].

La probabilité d'erreur par bit, $P_B$, est la probabilité d'avoir un bit en erreur. Toujours pour le canal binaire symétrique de probabilité de transition p, cette probabilité est calculée approximativement par:

$$P_B \approx c[2\sqrt{p(1-p)}]^{d_{free}} \qquad (2.8)$$

où c correspond au nombre de bits en erreur sur les chemins incorrects après $d_{free}$ branches si la première branche était différente. La valeur de c est, elle aussi, spécifique au code. Ces bornes n'utilisent que le premier terme d'une série infinie et ne sont serrées que lorsque p est petit.

Etant donné que ce mémoire utilise des algorithmes de décodage sous-optimaux, et que ces derniers ont la fâcheuse caractéristique d'avoir de très longs événements erreurs, la probabilité d'erreur de message $P_E$ ne donne pas une bonne idée de la qualité de l'algorithme de décodage. C'est pour cette raison que la probabilité d'erreur par bit, $P_B$, sera utilisée pour fin de comparaison au cours du mémoire par opposition à la probabilité d'erreur de message, $P_E$.

# CHAPITRE 3
# DECODAGE

Au chapitre 2, les notions de base de codage furent examinées. L'architecture d'un codeur fut discutée et des représentations graphiques de son comportement furent mises de l'avant. Utilisant ces modèles, ce chapitre montre comment un décodeur peut parvenir à retracer le chemin correct de façon efficace.

La première section de ce chapitre traite des concepts de base applicables à tous les algorithmes de décodage. Les sections suivantes introduisent à tour de rôle les trois grandes méthodes de décodage soit le décodage de Viterbi, le décodage séquentiel et le décodage multi-chemins.

L'algorithme général et quelques variantes de chacune des méthodes de décodage sont décrit. Les avantages et désavantages, sur le plan performance et réalisation, de chaque technique sont examinés.

## 3.1 Concepts de Base

Avant d'examiner en détail des algorithmes de décodage spécifiques, il est important de mentionner quelques notions communes à tous les algorithmes.

### 3.1.1 Partition de l'Information

Lors de l'encodage, les bits d'information sont groupés par trames de longueur L. Ce processus est utilisé pour faciliter le décodage et la demande de retransmission si cela était nécessaire. Bien qu'il soit possible pour certains algorithmes de décoder des séquences infinies, en pratique la plupart des systèmes opèrent par trames.

On voit à la figure 3.1 qu'une trame est définie comme étant L bits d'information suivis d'une queue de v (mémoire du codeur) bits connus par le décodeur

(normalement '0'). La présence de la queue assure la synchronisation du décodeur, c'est-à-dire qu'il soit toujours au même état au début de la trame suivante.

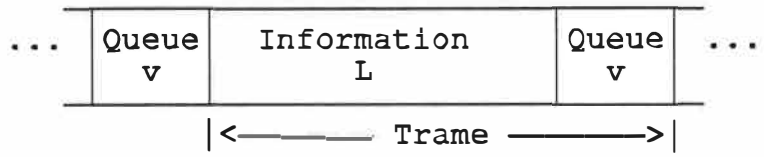


Figure 3.1 Composition d'une trame.

### 3.1.2 Principes de Décodage

Le décodage de codes convolutionnels peut être fait de manière algébrique [9] ou probabiliste [10]. De façon générale, compte tenu que les algorithmes probabilistes obtiennent de meilleures performances, seuls ces derniers sont considérés dans ce mémoire.

Le décodage de codes convolutionnels consiste à rechercher le message transmis, $\underline{U}$, le plus probable étant donné les symboles reçus $\underline{Y}$ (figure 3.2). C'est-à-dire minimiser la distance de Hamming, $d_H$, dans l'arbre ou le treillis entre les symboles du message transmis $\underline{X(U)}$ et ceux du message reçu $\underline{Y}$. Un décodeur qui minimise $d_H$ entre les symboles transmis et les symboles reçus est appelé décodeur à maximum de vraisemblance.

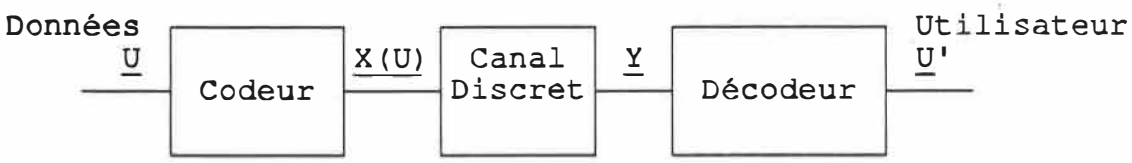


Figure 3.2 Information dans le canal.

Etant donné la fonction de vraisemblance:

$$P[\underline{Y}|\underline{X}(\underline{U})] \qquad (3.1)$$

un décodeur à maximum de vraisemblance choisira $\underline{U}'$ pour lequel:

$$P[\underline{Y}|X(\underline{U}')] > P[\underline{Y}|X(\underline{U})] \quad \underline{U} \neq \underline{U}'. \tag{3.2}$$

Dans un canal discret sans mémoire (DMC), cela revient à dire:

$$\begin{aligned} P[\underline{Y}|\underline{X}(\underline{U})] &= P[Y_1^1|X_1^1(\underline{U})]\ P[Y_1^2|X_1^2(\underline{U})]\ \ldots \\ &= \prod_{i=1}^{L} \prod_{j=1}^{V} P[Y_i^j|X_i^j(\underline{U})]. \end{aligned} \tag{3.3}$$

En définissant la métrique de branche comme étant

$$\gamma_i(\underline{U}) = \sum_{j=1}^{V} \log P[Y_i^j|X_i^j(\underline{U})], \tag{3.4}$$

maximiser la fonction de vraisemblance correspond à maximiser la métrique accumulée $\Gamma_L(\underline{U})$.

$$\begin{aligned} \log(P[\underline{Y}|\underline{X}(\underline{U})]) = \Gamma_L(\underline{U}) &= \gamma_1(\underline{U}) + \gamma_2(\underline{U}) + \gamma_3(\underline{U}) + \ldots \\ &= \sum_{i=1}^{L} \gamma_i(\underline{U}). \end{aligned} \tag{3.5}$$

Un décodeur à maximum de vraisemblance choisira le chemin dans l'arbre, ou le treillis, ayant la métrique maximale. Dans le canal binaire symétrique (BSC), le chemin ayant la métrique maximale est le chemin ayant la distance de Hamming minimale [1]-[3].

La métrique de branche $\gamma_i$, est un nombre réel, cependant dans une réalisation matérielle ou logicielle de l'algorithme cette valeur est habituellement arrondie à un entier. Cette opération accélère les additions et comparaisons requises sans affecter de façon significative l'efficacité de l'algorithme de recherche.

### 3.1.3 Algorithme de Décodage Général

Si la réalisation d'un codeur convolutionnel est relativement facile (chapitre 2), on ne peut pas en dire autant de celle du décodeur. Un décodeur peut être considéré

comme une copie du codeur à laquelle un comparateur et un additionneur sont rattachés. Le comparateur choisit les métriques de branches, $\gamma_i(\underline{U})$, et l'additionneur les ajoute à la métrique accumulée $\Gamma(\underline{U})$. Le décodeur choisit ensuite quels noeuds (chemins) doivent être poursuivis.

Un algorithme de décodage probabiliste général [11] utilise la procédure suivante (R = 1/V):

1- Coder les 2 réponses possibles d'un nombre M, $1 \leq M \leq 2^V$, de noeuds.
2- Comparer les 2M alternatives avec les symboles reçus.
3- Assigner à chaque alternative sa probabilité d'être le chemin correct ($\Gamma(\underline{U}) = \Gamma(\underline{U}) + \gamma_i(\underline{U})$).
4- Eliminer les chemins jugés incorrects.
5- Recommencer jusqu'à la profondeur finale.

Algorithme 3.1 Algorithme de décodage probabiliste général.

Bien que facile à comprendre, cet algorithme cache la complexité de la tâche dans des opérations en apparence simple. Les étapes 1 et 4 en sont de bons exemples. Les difficultés résident dans la quantité imposante de chemins possibles ainsi que dans la décision quant aux chemins à éliminer. Dans le pire cas, les 2 branches de $2^V$ noeuds représentent un nombre de chemins presque impossible à calculer et la décision quant aux chemins à éliminer n'est parfaite que lorsqu'il n'y a pas de bruit dans le canal ou si l'on connaît la réponse.

## 3.2 Décodage de Viterbi

L'algorithme de décodage de Viterbi [12],[13] est un algorithme optimal [14],[15]. Il trouvera la séquence, $\underline{X}(\underline{U}')$, dont la distance de Hamming, $d_H$, est la plus petite par rapport à la séquence reçue, $\underline{Y}$.

L'algorithme de Viterbi utilise le treillis (figure 2.6) comme structure de donnée. Il en parcourt de façon exhaustive tous les états. Les opérations principales

effectuées par un décodeur de Viterbi sont normalement dénotées par ACS (add, compare, select). L'algorithme (taux de codage R = 1/V) est décrit ci-dessous:

1- Coder les 2 possibilités des $2^v$ états.
2- Additionner les métriques de branche aux métriques accumulées ($\Gamma(\underline{U}) = \Gamma(\underline{U}) + \gamma_i(\underline{U})$). (ADD)
3- Comparer les métriques accumulées des 2 branches entrant à chaque noeud. (COMPARE)
4- Choisir la branche de métrique maximale de chaque noeud. (SELECT).
5- Recommencer jusqu'à la profondeur finale.

Algorithme 3.2 Algorithme de décodage de Viterbi.

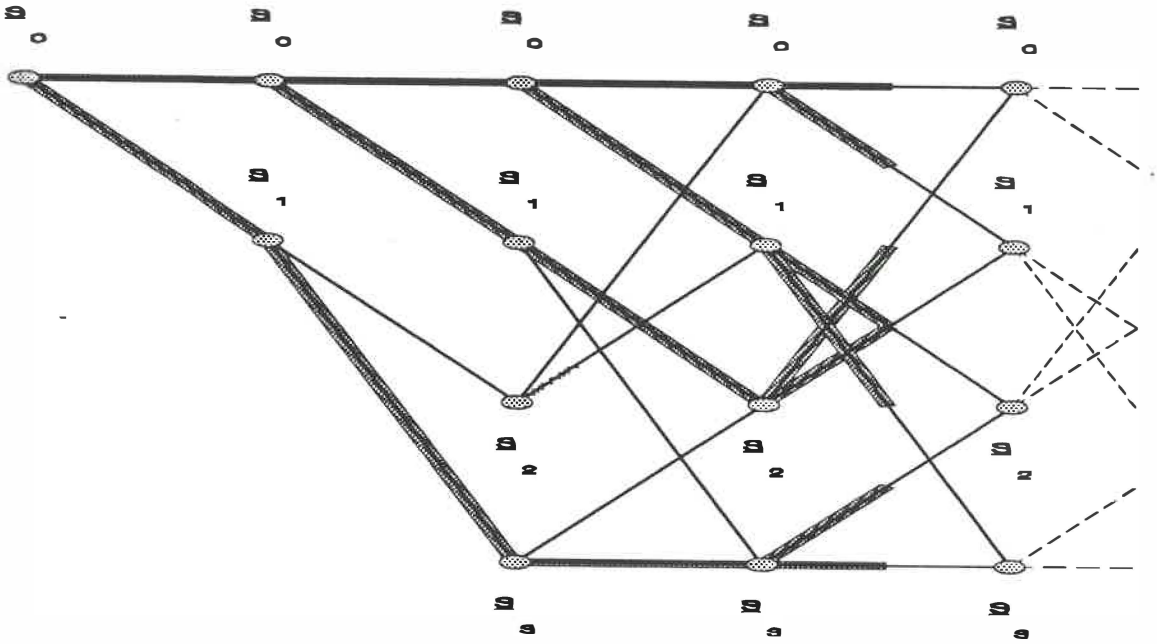

Figure 3.3 Recherche du chemin transmis dans un treillis.

Il est évident que le décodeur de Viterbi est un décodeur à maximum de vraisemblance car à chaque étape il ne rejette que des chemins ne pouvant être meilleurs que les chemins choisis. Si 2 chemins parviennent au même état, après cet état ils ne feront plus qu'un, par conséquent, le chemin le moins probable au point de rencontre ne pourra jamais devenir meilleur que le chemin choisi.

Dans un BSC, les valeurs de métriques de branches utilisées par le décodage de Viterbi sont simplement les distances de Hamming, $d_H$, entre les symboles reçus et les symboles calculés.

En théorie le décodeur doit attendre que toute la trame soit décodée avant de retracer le message; une telle application de l'algorithme nécessite le stockage en mémoire des $L*2^v$ états du treillis. En pratique, on observe qu'après une certaine longueur, tous les chemins se rencontrent en un seul point (figure 3.4). La différence de profondeur entre ce point et la position actuelle du décodeur est appelée longueur de convergence, $L_C$.

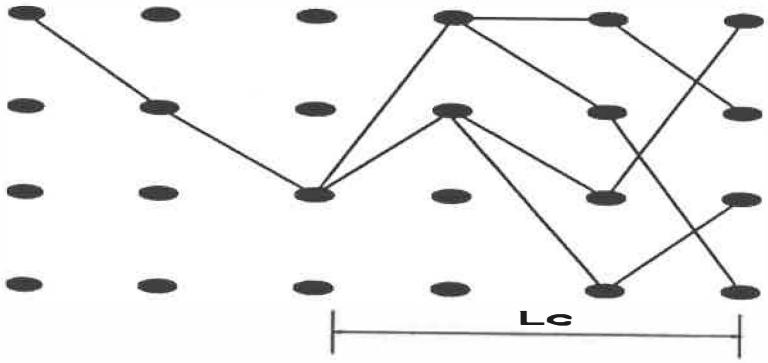


Figure 3.4 Convergence des états dans le treillis.

La performance d'erreur de l'algorithme de Viterbi pour plusieurs longueurs de contrainte sont illustrées à la figure 3.5. Les codes utilisés sont les codes optimaux de taux 1/2 de longueur de contrainte K = 5, 6, 7 et 8 qui sont décris par les générateurs (23,35), (53,75), (133,171) et (247,371). Ces codes sont optimaux pour le décodage de Viterbi [17]. Les simulations furent effectuées avec 10 000 trames et L = 500 bits/trame.

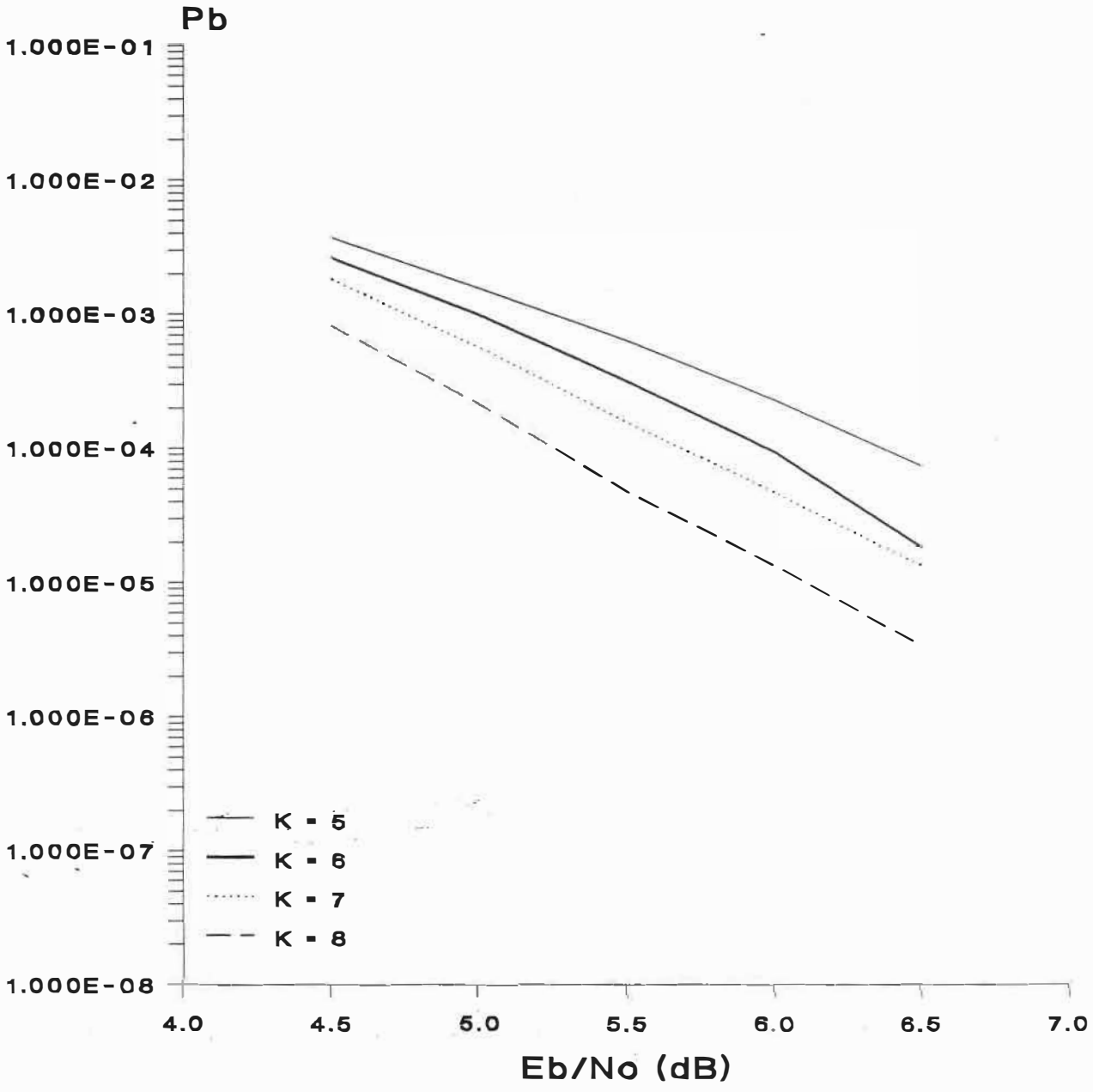


Figure 3.5 Probabilité d'erreur de l'algorithme de Viterbi en fonction de $E_b/N_0$ (dB) R = 1/2, BSC.

Le tableau 3.1 montre le nombre de trames et de bits en erreur pour différents rapports signal à bruit. De façon plus détaillée, on remarque que lorsque le rapport signal à bruit est très faible, $E_b/N_0 = 4.5$ dB, des erreurs sont présentes dans plus de 14% des trames. En décodage de Viterbi ceci correspond à l'élimination du chemin correct. Par contre, la nature exhaustive de la recherche permet au décodeur de retrouver le chemin correct après 6.11 bits en moyenne. On suppose ici comme dans le reste des analyses que le décodeur ne perd le chemin correct qu'une fois par trame. Plus le nombre de chemins est élevé, plus cette supposition est exacte.

| $E_b/N_0$ (dB) | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Durée moy. des erreurs |
|---|---|---|---|---|---|
| 4.5 | 1486 | 9092 | 1.486E-1 | 1.818E-3 | 6 |
| 5.0 | 571 | 2844 | 5.710E-2 | 5.688E-4 | 5 |
| 5.5 | 198 | 763 | 1.980E-2 | 1.526E-4 | 4 |
| 6.0 | 64 | 230 | 6.400E-3 | 4.600E-5 | 4 |
| 6.5 | 19 | 65 | 1.900E-3 | 1.300E-5 | 3 |

Tableau 3.1 Probabilité d'erreur de l'algorithme de Viterbi pour plusieurs valeurs de $E_b/N_0$ (dB)(K = 7, BSC).

Si un calcul est défini comme l'extension d'un noeud en ses branches l'algorithme de décodage de Viterbi effectue tous les calculs du treillis. Cette procédure conduit à l'optimalité et à un effort de calcul constant mais non-négligeable. La probabilité d'erreur d'un code diminue exponentiellement avec K, mais le nombre d'états du treillis augmente aussi dans les mêmes proportions. Ce fâcheux comportement restreint la réalisation de décodeurs utilisant l'algorithme de Viterbi à de petites longueurs de contrainte ($K \leq 7$).

## 3.3 Décodage Séquentiel

Si le décodage de Viterbi souffre d'une croissance exponentielle de l'effort de calcul avec K, il convient donc d'examiner d'autres techniques de recherche afin de surmonter cet obstacle. Il existe deux classes de techniques de recherche: les techniques en largeur (breadth first) comme l'algorithme de Viterbi et les techniques de recherche en profondeur (depth first).

Le décodage séquentiel est un exemple de recherche en profondeur. Il utilise la structure en arbre (figure 2.7), et ne poursuit que le chemin de métrique maximale (figure 3.6). Cette technique a pour but de n'effectuer qu'une infime portion des calculs possibles permettant ainsi l'usage de grandes longueurs de contrainte $K \geq 20$. Par contre lorsque la métrique du chemin exploré chute, le décodage séquentiel permet des retours en arrière dans l'arbre et par conséquent une variabilité de l'effort de calcul est introduite. Cette variabilité de l'effort de calcul force un décodeur à utiliser un tampon d'entrée pour stocker les trames reçues et un tampon de sortie pour adoucir le taux de sortie.

Il existe deux méthodes de décodage séquentiel: l'algorithme de Fano [18] et celui de Zigangirov-Jelinek (Z-J) [19],[20]. L'algorithme de Fano ne conserve pas en mémoire l'ensemble des noeuds terminaux de l'arbre exploré. Pour choisir une autre branche, il doit recalculer son chemin à l'envers. L'algorithme Z-J, par contre, conserve les noeuds terminaux dans une pile (figure 3.7). Afin de choisir la prochaine branche il n'a qu'à sortir le noeud de métrique maximale de la pile. Dans les deux cas le décodage se termine lorsque l'un des chemins explorés atteint la profondeur finale. L'algorithme de Fano requiert donc moins de mémoire, mais au prix d'une complexité accrue du décodeur. Il fut démontré [21] que les deux méthodes sont équivalentes au niveau du chemin décodé.

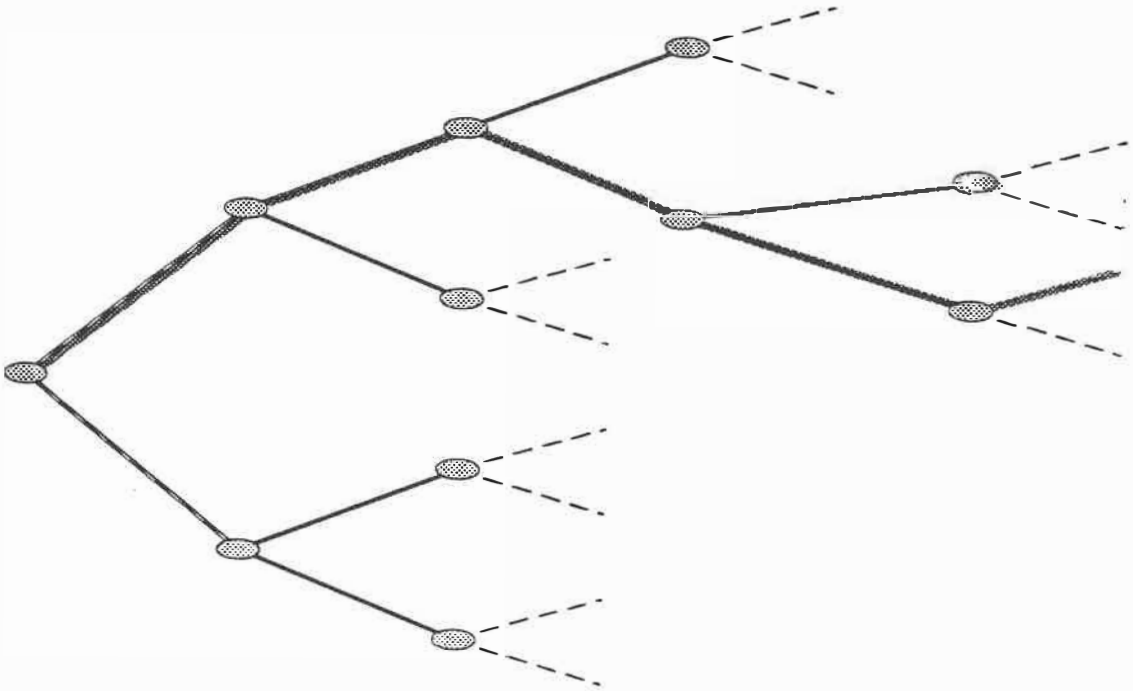

Figure 3.6 Recherche du chemin transmis dans un arbre.

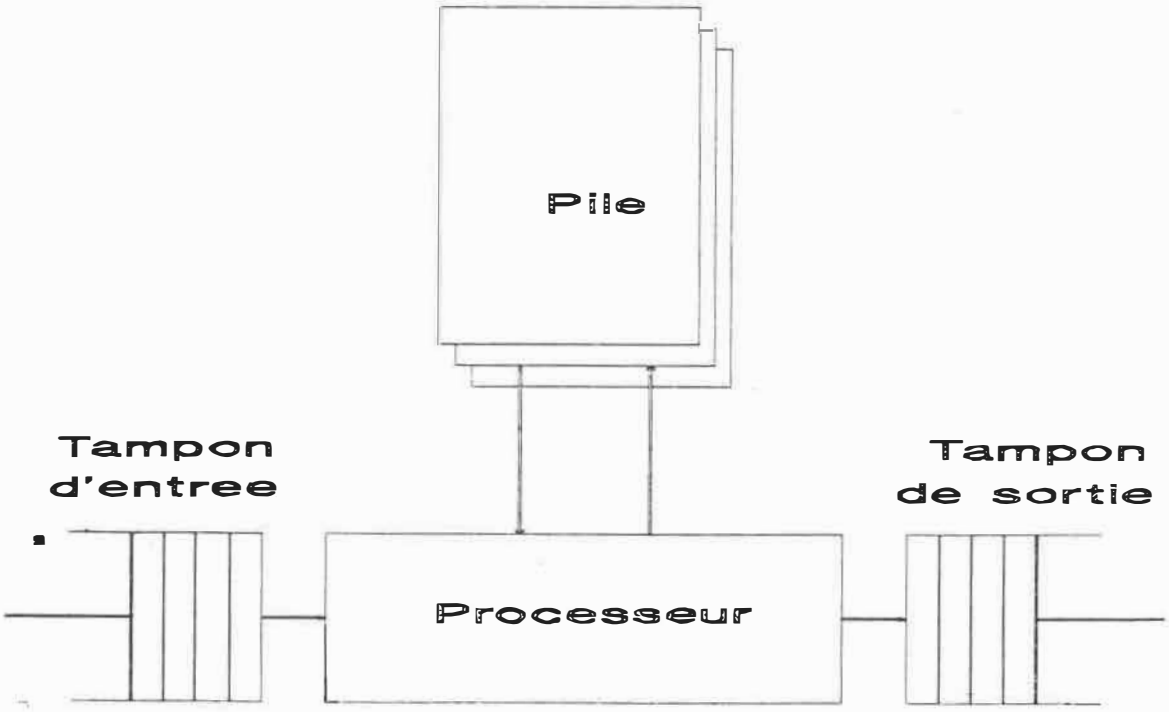


Figure 3.7 Schéma bloc du décodeur Z-J.

De conception plus simple et plus rapide, l'algorithme Z-J a été longtemps inexploité à cause des coûts associés à la mise en ordre de la pile. Cependant l'ITGE (Intégration à très grande échelle) change la valeur de ces coûts; l'algorithme Z-J devient alors attrayant, c'est pourquoi seul ce-dernier sera étudié dans ce mémoire.

L'algorithme de base du décodeur Z-J est le suivant:

1- Faire l'extension des 2 possibilités du meilleur noeud. Recalculer les métriques accumulées.
2- Mettre les nouveaux noeuds dans la pile.
3- Ordonner la pile selon les métriques accumulées.
4- Retirer le noeud ayant la meilleure métrique accumulée.
5- Recommencer jusqu'à ce que la profondeur finale soit atteinte.

Algorithme 3.3 Algorithme de décodage Z-J.

La comparaison des métriques dans l'algorithme de décodage séquentiel se fait sur des noeuds de profondeurs différentes dans l'arbre. La métrique à utiliser n'est donc plus celle développée précédemment. Il a été démontré [18] que la métrique du décodeur séquentiel doit être adaptée de façon à ce que, en moyenne, la métrique du chemin correct augmente avec la profondeur et que celles des chemins incorrects diminuent. Cet ajustement est effectué avec la valeur du taux de codage R et aide à réduire la variabilité de l'effort de calcul.

$$\gamma_i(\underline{U}) = \sum_{j=1}^{V} \left[ \lg \frac{P[Y_i^j | X_i^j(\underline{U})]}{P[Y_i^j]} - R \right] \qquad (3.6)$$

Bien que l'effort moyen de calcul soit petit, normalement un calcul suffit pour décoder 1 bit. L'utilisation d'une technique de recherche en profondeur implique une variabilité de cet effort de calcul. Le nombre de calculs nécessaires pour décoder un bit, C, suit une loi de type Pareto [18],[22],[23]:

$$P[C > x] = \beta x^{\alpha}, \quad 1 << x < \infty \qquad (3.7)$$

où $\beta$ est une constante pouvant être déterminée par simulation et $\alpha$ est l'exposant de Pareto donné par:

$$E_0(\alpha)/\alpha = R \tag{3.8}$$

où $E_0(\alpha)$ est la fonction de Gallager [23]

$$E_0(\alpha) = -\log \sum_{j=1}^{J} \left[ \sum_{i=1}^{Q} q(i) P[j|i]^{1/(1+\alpha)} \right]^{1+\alpha}. \tag{3.9}$$

Dans cette équation J, Q, q(i) et P[j | i] sont définis par le modèle de canal utilisé. Une étude des moments de la cumulative [22] montre que bien que l'effort de calcul soit variable, il est dans certain cas borné. Si $\alpha > 1$ la moyenne est bornée et si $\alpha > 2$ alors la variance est bornée. Si $\alpha \leq 1$, la moyenne n'est plus bornée et le nombre de calcul requis pour décoder une trame devient théoriquement infini. Le comportement de l'effort de calcul du décodeur séquentiel, tel que mesuré par simulations est illustré à la figure 3.8.

La variabilité de l'effort de calcul entraîne quelques fois un débordement du tampon d'entrée (effacement "erasure") ou de la pile (figure 3.7). Il fut démontré [24] qu'il est possible d'éliminer les effacements en restreignant ceux-ci à des débordements de la pile. De tels débordements se produisent dans 2% à 3% des trames à faibles rapports signal à bruit, avec des tailles de pile réalistes d'environ 5000 entrées. En cas de débordement une retransmission doit être demandée, ou la trame en question doit être décodée avec une procédure forçant la marche avant. Cependant cette procédure réduit considérablement la performance d'erreur du décodeur. Des variations de l'algorithme séquentiel de base peuvent aussi éliminer les effacements [25], mais la performance d'erreur est encore une fois dégradée.

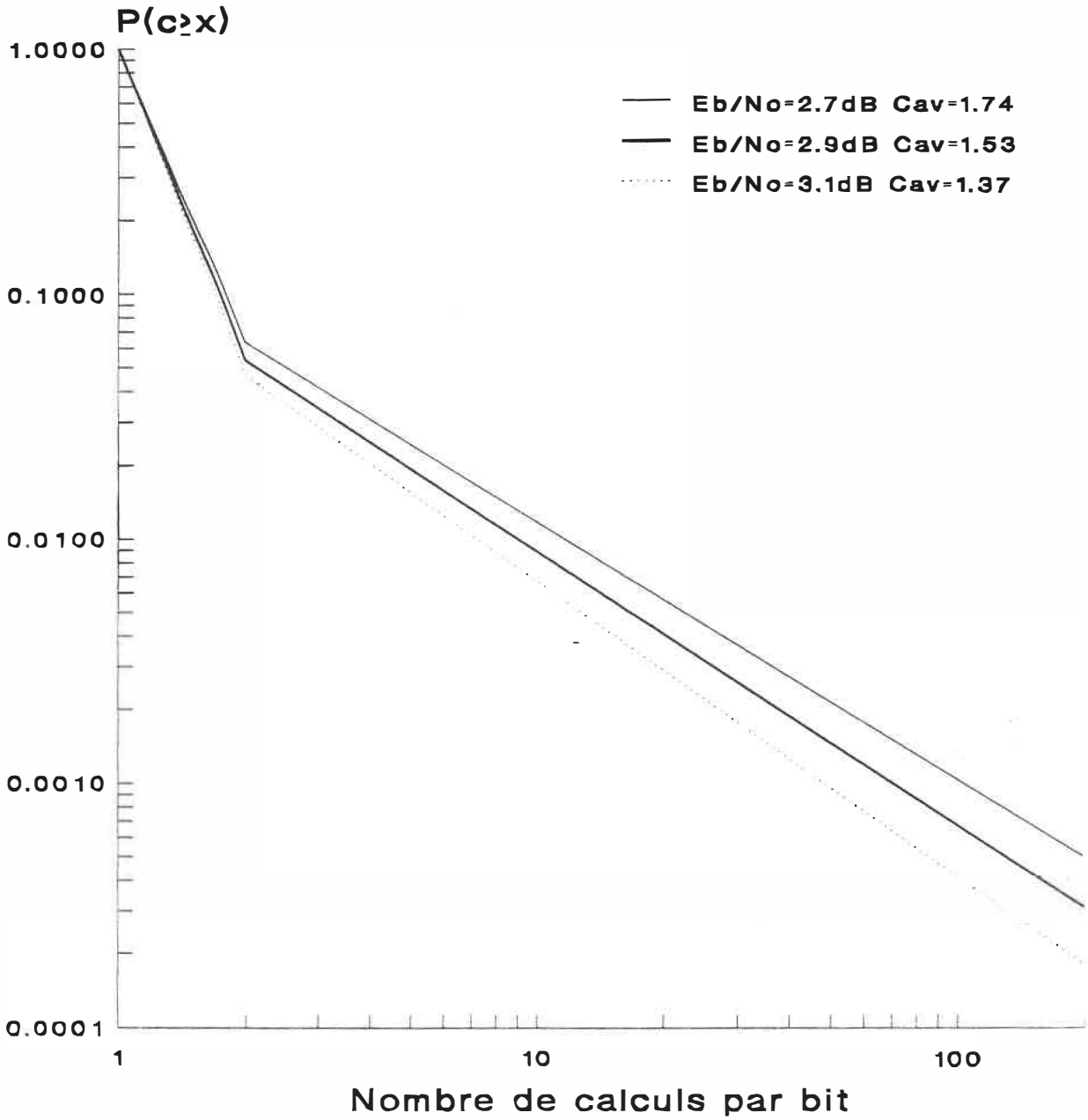


Figure 3.8 Cumulative de l'effort de calcul Pile=20 000, L=500, 3 bits quantification, R=1/2, K=20.

L'utilisation de recherche en profondeur implique également que la réponse choisie sera une bonne réponse mais pas nécessairement 'la réponse'. Le décodeur séquentiel n'est pas optimum car il est possible qu'un autre chemin ait une distance de Hamming inférieure à celle du chemin choisi par rapport à la séquence reçue. Le décodeur séquentiel peut choisir un chemin de poids supérieur s'il existe un chemin incorrect qui, à une profondeur donnée, possède une métrique supérieure à celle du chemin correct (trait pointillé figure 3.9). Si, en poursuivant son parcours, la métrique de ce chemin incorrect ne chute jamais en dessous de la métrique courante du chemin correct le chemin choisi ne serait pas alors le chemin correct et le décodeur aurait fait une ou plusieurs erreurs.

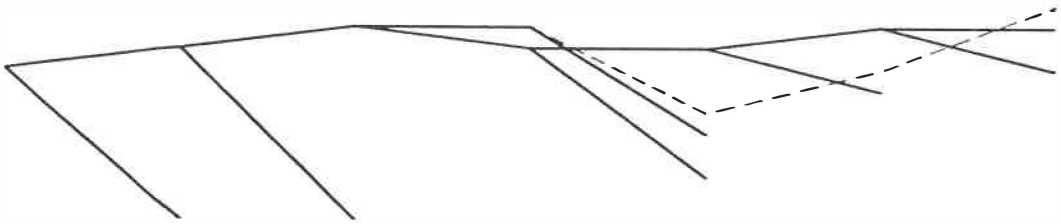

Figure 3.9 Perte du chemin correct en Z-J.

Bien qu'il soit sous-optimal, la probabilité d'erreur du décodeur séquentiel diminue de façon exponentielle avec la longueur de contrainte du code. On montre que pour l'ensemble des codes [1]

$$P_E < L2^{-KR_{comp}/R} \qquad R < R_{comp} \tag{3.10}$$

où $R_{comp}$ (Computational Cut Off Rate) est le taux de coupure défini par:

$$R_{comp} = E_0(1). \tag{3.11}$$

Pour le canal binaire symétrique de probabilité de transition p et pour le canal à bruit gaussien blanc $R_{comp}$ devient respectivement:

$$R_{comp} = 1-\log[1+2\sqrt{p(1-p)}] \tag{3.12}$$

$$R_{comp} = 1-\log[1+e^{-RE_b/N_0}]. \tag{3.13}$$

Dans un système utilisant une technique de décodage séquentiel, le taux de codage est toujours choisi de sorte que $R/R_{comp} < 1$.

En résumé, la sous-optimalité du décodeur séquentiel n'est pas un inconvénient majeur. C'est plutôt l'importante variabilité de l'effort de calcul ainsi que les effacements possibles qui en limite l'usage. Un autre désavantage du décodage Z-J est la mise entre ordre de la pile après chaque extension. Cette opération très coûteuse doit être répétée fréquemment et devient rapidement le goulot d'étranglement d'une réalisation matérielle.

## 3.4 Décodage Multi-chemins

Bien qu'obtenant une très bonne performance d'erreur, les problèmes encourus par le décodage séquentiel ne permettent pas à ce dernier de supplanter l'algorithme de Viterbi dans beaucoup d'applications où une réponse en temps réel est demandée. Il existe donc un besoin pour un algorithme ne possédant que peu ou pas de variabilité de l'effort de calcul et pouvant offrir un meilleur gain de codage que l'algorithme de Viterbi à un coût moindre.

Le décodage multi-chemins tente de répondre à ce besoin. Deux grandes catégories de décodeurs multi-chemins peuvent être identifiées. Les algorithmes permettant les retours en arrière [11] (recherche en profondeur) s'apparentent au

décodage séquentiel et souffrent de variabilité de l'effort de calcul. Les algorithmes ne permettant pas les retours en arrière [26] (recherche en largeur) ressemblent au décodeur de Viterbi et nécessitent beaucoup de chemins pour obtenir une bonne performance d'erreur.

Chacune de ces catégories peut posséder ou non un nombre variable de chemins selon le bruit présent dans le canal; on dira que l'algorithme est adaptatif ou non-adaptatif. Utilisées dans un algorithme multi-chemins, les techniques de décodage adaptative tentent d'obtenir un compromis entre l'effort moyen de calcul et la variabilité de l'effort de calcul [11].

Si le décodeur séquentiel décode en moyenne très bien avec un seul chemin, l'algorithme adaptatif vise à utiliser le moins de chemins possibles durant les passages non-bruités et à décoder avec beaucoup de chemins lors de passages bruités. A la limite, cette approche promet un décodage optimum avec un nombre minimum de chemin. Cependant, il est très difficile pour un décodeur de décider combien de chemins il devrait utiliser. Dans le pire cas, le nombre de chemins peut devenir presque infini. La réalisation d'un système utilisant cette technique de décodage doit tenir compte de cette contrainte.

Les performances des décodeurs multi-chemins réalisables se situent entre celles du décodeur de Viterbi et celles du décodeur séquentiel sur presque tous les points de vue. Pour un code donné, les décodeurs multi-chemins affichent une performance d'erreur intermédiaire, sont sous-optimaux et souffrent à un moindre degré d'une variabilité de l'effort de calcul.

### 3.4.1 Décodage Multi-chemins en Profondeur

Les algorithmes de cette classe sont des descendants rapprochés du décodeur Z-J mais dans lesquels au plus M extensions sont effectuées à chaque étape [11]. Ces décodeurs possèdent donc une pile, un tampon d'entrée et une variabilité de l'effort de

calcul tout comme le Z-J. Cependant, la variabilité de l'effort de calcul est moindre que dans le Z-J traditionnel.

Le décodeur est essentiellement le même que le décodeur Z-J habituel auquel est additionné le contrôle du nombre d'extensions et une mesure de la qualité du canal. Adaptatives ou non, ces approches réussissent à réduire, mais non à éliminer, la variabilité de l'effort de calcul. C'est pour cette raison que cette catégorie d'algorithmes ne sera pas considérée dans ce mémoire.

### 3.4.2 Décodage Multi-chemins en Largeur

Ces algorithmes se rapprochent du décodage de Viterbi. Toutefois, ils n'explorent qu'une partie des états à chaque pas de décodage. La plupart des algorithmes (M,L) [26] ainsi que les "Viterbi réduits" [28] sont de cette catégorie.

Si le nombre d'états est considérablement réduit, la performance d'erreur n'est plus dominée par la longueur de contrainte mais plutôt par le nombre d'extensions utilisées. Même si l'on utilise des codes puissants et un grand nombre de chemins, ces algorithmes souffrent souvent d'une piètre performance d'erreur. Des résultats de simulations [29] indiquent que tout algorithme de complexité réduite aura de la difficulté à conserver le chemin correct en cas de rapport signal à bruit faible. Ces mêmes résultats tendent à démontrer que le nombre minimum de chemins à explorer est de l'ordre de la racine carrée du nombre d'états [30].

Les variantes non-adaptatives de cette classe d'algorithmes éliminent cependant toute variabilité de l'effort de calcul et utilisent un nombre de chemins moindre que le décodeur de Viterbi original. C'est donc un de ces algorithmes qui sera considéré au chapitre 5.

# CHAPITRE 4
# ARCHITECTURE DE PILES

D'un point de vue théorique et pratique, le problème majeur du décodage séquentiel est la variabilité de l'effort de calcul. De plus une réalisation pratique entraîne d'autres difficultés. La principale difficulté de réalisation du décodeur séquentiel réside dans la confection d'une pile de grandeur N dont la mise en ordre s'effectue en temps constant indépendant de N, (ordre 1 O(1)).

Il est donc normal de considérer plus en détail les algorithmes de tri. Il est également important de se rappeler que les très grandes vitesses de décodage exigées par l'industrie requièrent une réalisation matérielle de l'algorithme.

En première analyse, étant donné que le décodeur n'a besoin de retirer que le meilleur noeud, une mise en ordre complète de la pile apparaît être un mauvais investissement des ressources matérielles. Des réalisations sous la forme de logiciels informatiques de la pile [20],[24] utilisent des méthodes de tri partiel très efficaces permettant d'approcher un tri d'ordre O(1). Cependant la réalisation matérielle de ces algorithmes s'avère très difficile voire impossible. Il convient alors d'examiner des algorithmes plus réguliers et plus répétitifs.

D'un point de vue algorithmique, conserver une structure ne connaissant que la position du meilleur noeud requiert O(lgN) comparaisons [31]. Par contre si la pile est ordonnée, l'insertion d'un item dans une liste ordonnée nécessite également O(lgN).

Une mise en ordre partielle ou une insertion, réalisée dans une architecture monoprocesseur, et effectuant une comparaison par cycle, occuperait au mieux O(lgN) cycles, N étant la profondeur de la pile. Ce temps variable est encore plus indésirable que

la variabilité de l'effort de calcul et devient très rapidement le goulot d'étranglement du système.

S'il est impossible de réussir une mise en ordre en temps constant avec une structure monoprocesseur, le but peut être atteint avec une architecture multiprocesseur.

## 4.1 Evaluation de Structures Multiprocesseurs

Parmis les problèmes des architectures multiprocesseurs ont peut mentionner l'évaluation de la complexité et la comparaison des différents algorithmes entre eux. Afin de permettre une comparaison éclairée des différentes solutions proposées et afin de pouvoir les comparer aux algorithmes monoprocesseurs le produit AT sera utilisé. Cette mesure de complexité est définie comme le produit de la surface utilisée, A, par le temps de calcul, T, requis pour l'opération. On suppose ici que tous les processeurs ont une unité de surface unitaire équivalente, qu'ils effectuent tous la même tâche et que le temps d'exécution est constant et identique pour tous les processeurs. Dans un système monoprocesseur $A = 1$ et $T = O(\lg N)$, le but est d'obtenir une architecture multiprocesseur pour laquelle $A = O(\lg N)$, et $T = 1$.

Bien que les systèmes multiprocesseurs soient très puissants, le gain de vitesse obtenu ne dépassera normalement jamais le nombre de processeurs utilisés. C'est-à-dire que pour résoudre un problème dont la complexité AT est d'ordre $O(\lg N)$ en temps constant, $O(1)$, on doit utiliser un nombre de processeur proportionnel a $\lg N$.

Si on ne peut pas trouver d'algorithmes plus performants que les algorithmes pour monoprocesseur en ce qui concerne le produit AT, il est cependant possible d'admettre une certaine inefficacité dans l'utilisation des processeurs pour obtenir un temps constant.

Le produit AT peut alors être utilisé comme indice de la qualité de la solution obtenue. Ce produit représente l'excès de complexité par rapport à un algorithme monoprocesseur. Typiquement, l'algorithme de tri monoprocesseur utilise un processeur et résout le problème en temps O(lgN) donnant un produit AT égale à lgN. Les solutions multiprocesseur qui permettent un temps de traitement constant exigent toutes un ordre de O(N) processeurs ce qui donne un produit AT égale à N. Si le temps de l'algorithme monoprocesseur est normalisé à 1, alors l'indice de performance de l'algorithme multiprocesseur devient N/lgN. Lorsque N devient grand, cette croissance devient à toutes fins pratiques linéaire.

Les réalisations multiprocesseur des algorithmes de tri n'utilisent en général que deux types d'opérations: les comparaisons et les transferts. Compte tenu que les opérations de transfert sont beaucoup plus rapides que les comparaisons, l'unité de temps choisie sera le temps requis pour une comparaison. Puisque la seule différence entre les différents algorithmes est le nombre de comparaisons requis pour obtenir les tris, alors la mesure de complexité matérielle utilisée sera le nombre de comparateurs employé par l'algorithme. Par conséquent, sous ces hypothèses, le produit AT se ramène à l'expression suivante: nombre de comparateurs * nombre de comparaisons.

Cette fonction de coût néglige les coûts de communication et d'entrée/sortie. Les coûts de communication sont négligés à cause de la similitude de ces dernières à travers les architectures présentées. Les coûts d'entrée/sortie ne sont pas considérés car jusqu'à présent le nombre de broches et la vitesse de ces dernières sont suffisantes pour que le goulot d'étranglement d'un système ne se situe pas à ce niveau. Ces hypothèses proviennent de réalisations ITGE (intégration à très grande échelle) effectuées à l'Ecole Polytechnique de Montréal [32].

## 4.2 Concepts Architecturaux

La seule façon économiquement viable de réaliser une pile multiprocesseurs pour un décodeur séquentiel est l'utilisation de puces dédiées. Grâce à l'avancement de l'ITGE, la réalisation de structures multiprocesseurs rapides sur une puce est devenue une réalité. De plus, l'utilisation de techniques d'ITGE permet de conserver un système simple, efficace et rapide [33].

Pour obtenir une solution matérielle efficace à un problème algorithmique, tel que celui du tri, le problème se doit de respecter une certaine structure et doit pouvoir être séparé en plusieurs tâches identiques et indépendantes. Ce n'est que lorsque ces deux conditions sont remplies que l'utilisation d'un système multiprocesseurs devient avantageuse.

Un ensemble de règles régissant les communications entre ces processeurs doit être élaboré. Ces règles dépendent principalement de la structure (arrangement physique) adoptée par les processeurs et servent à classer les architectures. Etant donné le problème à résoudre et les contraintes imposées par l'ITGE, seules les architectures systoliques ou semi-systoliques seront considérées [34].

Une architecture systolique est une architecture multiprocesseur dans laquelle les communications interprocesseurs sont limitées aux voisins immédiats. Cette méthode offre les avantages d'un réseau de connexion facilement réalisable, rapide et occupant une surface presque négligeable par rapport aux autres configurations. La largeur de bande offerte par la multitude de petits bus trouvés dans une architecture systolique est également un avantage.

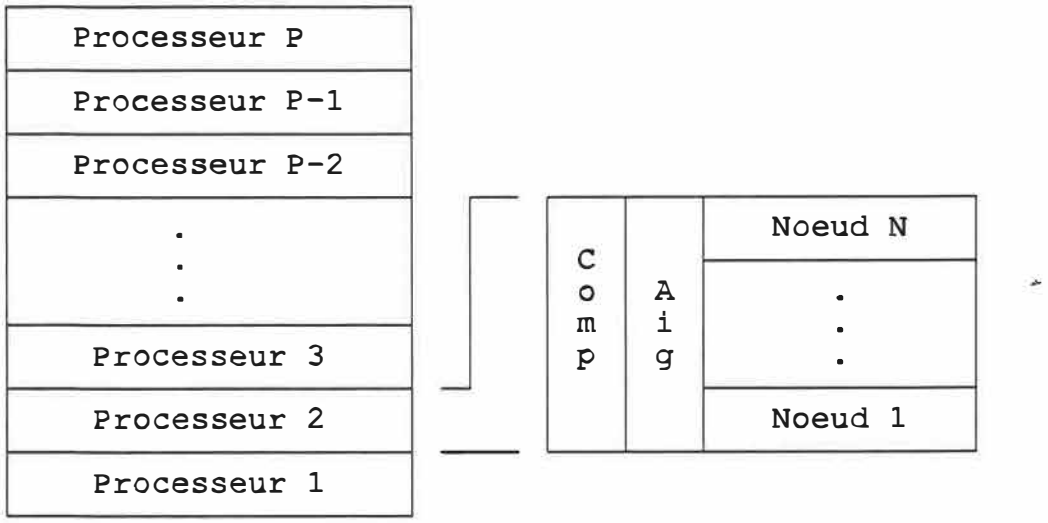


Figure 4.1 Pile utilisant une architecture systolique. Chaque processeur contient au moins 2 noeuds plus un nombre de comparateurs et un aiguilleur.

Bien que les architectures systoliques pures ne contiennent que des interconnexions entre voisins immédiats, il existe une variante, les architectures semi-systoliques, où certaines formes de communications à grande distance sont permises. Comme dans les architectures systoliques pures, les processeurs dans une architecture semi-systolique ne communiquent qu'entre voisins immédiats mais un bus de données globales est utilisé pour communiquer les informations pertinentes à tous les processeurs.

La réalisation d'une pile sous forme d'architecture systolique permet l'utilisation d'autant de processeurs qu'il y a de noeuds dans la pile, et garantit un temps fixe de mise en ordre. L'utilisation de cette structure remplit également les besoins de régularité et de vitesse. Seules les communications entre voisins étant permises, un tri partiel est effectué, ce tri s'avère suffisant pour garantir la sortie du meilleur noeud contenu dans la pile à chaque cycle.

## 4.3 Pile Systolique de Chang et Yao

Les architectures systoliques [35] sont l'objet de nombreuses recherches dans un grand nombre de domaines d'applications. Les problèmes de tri en vue de fournir le meilleur élément d'une liste [36] n'ont pas fait exception à la vague des domaines d'applications couverts par les architectures systoliques. Un algorithme basé sur le principe de diviser pour conquérir fut mis au point. Cet algorithme part du principe que le nombre de comparaisons requis pour ordonner un ensemble d'éléments augmente plus rapidement que le nombre d'éléments. Il s'agit donc de créer plusieurs petits sous-ensembles et de traiter ceux-ci de façon indépendante et simultanée. Chang et Yao [37] reconnurent l'utilité des architectures systoliques dans la réalisation d'un décodeur séquentiel à pile et proposèrent quelques réalisations.

Dans l'architecture définie par Chang et Yao, les opérations d'entrée et de sortie ne suivent pas obligatoirement un cycle prédéterminé. Les deux entrées suivies d'une sortie requise pour le décodage d'un code de taux R = 1/V n'est qu'une des possibilités supportées par leur solution.

Physiquement les noeuds de la pile sont groupés à deux par tranche. En plus des bits de mémoire requis pour conserver les noeuds, la tranche comprend un comparateur et un échangeur, ou aiguilleur. Le comparateur est responsable de décider si le noeud du haut est supérieur à celui du bas, et s'il l'est alors l'aiguilleur échange les deux éléments de façon à ce que le plus grand des deux se retrouve dans la position du bas.

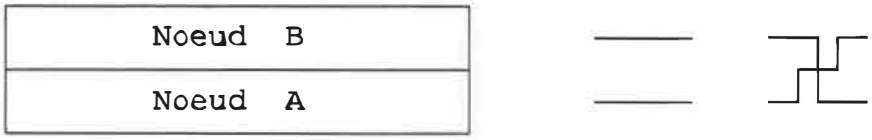


Figure 4.2 Pile de Chang et Yao. Tranche et opérations possibles sur la tranche.

Une pile systolique est donc composée d'une série de tranches. En plus de pouvoir ordonner les éléments tranche par tranche, la pile possède la capacité de décaler les éléments vers le haut et vers le bas d'une position. Dans le mode décalage, la notion de tranche n'existe plus et la pile n'est plus qu'une suite d'éléments indépendants.

Cet algorithme composé de deux parties peut être décrit comme suit.

1- Décaler les éléments d'une position vers le haut.
2- Insérer le nouvel item dans la première position.
3- Effectuer le tri sur toutes les tranches de façon indépendante.

Algorithme 4.1 Entrée d'un item dans l'algorithme de Chang et Yao.

1- Sortir le premier item de la première tranche. C'est l'item maximum.
2- Décaler les éléments d'une position vers le bas.
3- Effectuer le tri sur toutes les tranches de façon indépendante.

Algorithme 4.2 Sortie d'un item dans l'algorithme de Chang et Yao.

Une notion importante pour une bonne compréhension du comportement de la pile systolique est le concept du débordement. En effet, toutes les réalisations pratiques d'une pile possèdent une grandeur finie. Par définition, une pile systolique où on insère deux noeuds pour n'en retirer qu'un seul déborde constamment. A chaque insertion, lors de l'opération de décalage, un noeud est perdu à jamais.

La notion de débordement de la pile conduit naturellement au concept de la qualité du tri. Le tri n'étant que partiel il est alors possible qu'un noeud dont la métrique est de valeur supérieure à celles de quelques un de ses confrères soit perdu alors que ces derniers demeurent dans la pile. Dans un tel cas, il est important de trouver quel pourcentage des N meilleurs noeuds est garanti par la pile.

Bien que la pile puisse contenir au total N noeuds, la borne minimum sur le nombre de noeuds étant parmis les N meilleurs n'est que de N/2. Si à un moment donné la

pile contient les N meilleurs noeuds et que soudainement les nouveaux noeuds entrant dans la pile ne sont plus parmis les N meilleurs, et ce durant N/2 cycles, il faudra alors N/2 cycles avant que le premier de ces mauvais noeuds puisse sortir de la pile grâce à un débordement. La pile à cet instant contient donc N/2 mauvais items et N/2 bons items.

En conclusion, cette pile adaptée au décodage séquentiel de taux R = 1/V contient N/2 comparateurs, requiert 3 comparaisons par comparateur et 3 cycles pour entrer et sortir les items associés à une extension tout en ne garantissant que les N/2 meilleurs noeuds.

## 4.4 Pile à Entrées Parallèles

Même si la pile de Chang et Yao peut être utilisée en décodage séquentiel, elle utilise trois comparaisons par comparateur. Il convient donc de se demander s'il est possible de faire mieux.

En décodage séquentiel de taux R = 1/V, V = 2,3... la pile doit toujours rendre le meilleur noeud et en recevoir deux. Il est donc possible d'imaginer une pile optimisée pour cette application [32]. Cette nouvelle pile possède alors des tranches de trois noeuds où chaque tranche nécessite, en plus de sa mémoire, trois comparateurs et un aiguilleur.

Les trois comparateurs ont pour but de déterminer le plus grand des trois items et l'aiguilleur a comme tâche d'échanger le meilleur noeud avec la position A. Le but final est encore de placer le noeud ayant la plus grande métrique au bas de la tranche. La position relative des deux autres noeuds n'est pas importante.

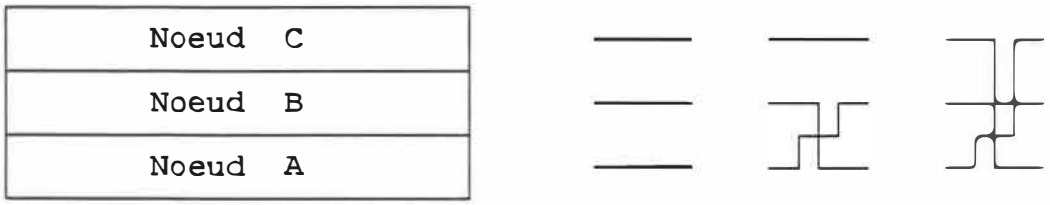


Figure 4.3 Pile à entrées parallèles. Tranche et opérations possibles sur la tranche.

L'algorithme d'entrées et de sorties des noeuds demeure sensiblement le même.

1- Décaler les éléments de deux positions vers le haut.
2- Insérer les nouveaux éléments dans les positions A et B.
3- Effectuer le tri sur toutes les tranches de façon indépendante.
4- Sortir le noeud A de la première tranche. C'est le noeud maximum.
5- Décaler les éléments d'une position vers le bas.
6- Effectuer le tri sur toutes les tranches de façon indépendante.

Algorithme 4.3 Cycle d'entrée et sortie dans l'algorithme de pile à entrées parallèles.

Cette nouvelle approche permet d'obtenir un tri partiel de la pile et de toujours délivrer le meilleur noeud, c'est-à-dire celui possédant la métrique la plus élevée en n'utilisant que deux comparaisons par comparateur. Cependant, comme l'algorithme de Chang et Yao, cette architecture ne permet que de garder qu'une fraction des N meilleurs noeuds. Dans le cas de la pile parallèle ce rapport n'est que de 1/3.

En résumé, cette pile spécialisée pour le décodage séquentiel de taux R = 1/V requiert N comparateurs, 2 comparaisons par comparateur, 1 cycle pour entrer et sortir les items et ne garantit que les N/3 meilleurs noeuds.

## 4.5 Pile à Tranches Ordonnées

Bien que la pile à entrées parallèles offre un gain de vitesse par rapport à l'algorithme de Chang et Yao, ce gain de vitesse survient au prix d'un accroissement de la complexité. Ainsi le nombre de comparateurs passe de N/2 à N et bien que le nombre de

cycles de comparaisons soit réduit, le total de comparaisons passe de 3N/2 à 2N. Une perte substantielle quant à la borne sur le nombre minimum de meilleurs noeuds est également inévitable.

La pile à tranches ordonnées économise sur le nombre de comparateurs et de comparaisons. Elle améliore la qualité du tri sans pour autant réduire la vitesse. Si l'on sait a priori laquelle des deux nouvelles entrées est la meilleure, il ne reste plus qu'à effectuer 2 comparaisons pour déterminer la position du troisième élément de la tranche. L'aiguilleur pourrait alors ordonner parfaitement cette tranche.

Si chaque tranche est ordonnée, à l'insertion, les deux éléments passant d'une tranche à l'autre seront nécessairement ordonnés. Il est donc possible de placer le troisième élément en position et par le fait même recréer des tranches ordonnées. Lors du retrait, les éléments doivent descendre d'une position, et puisque la tranche est ordonnée, les positions A et B sont nécessairement ordonnées. Il ne reste plus qu'à placer une autre fois le troisième noeud à sa position.

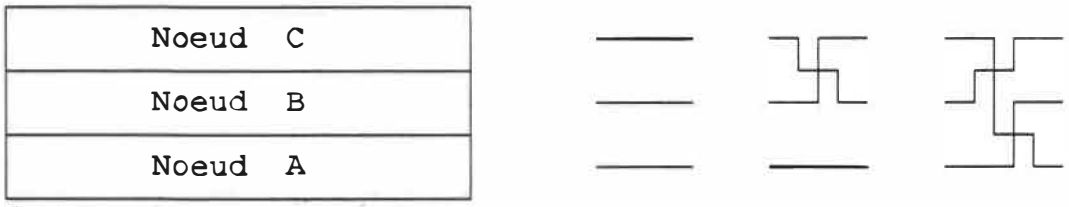


Figure 4.4 Pile à tranches ordonnées. Tranche et opérations possibles sur la tranche.

Afin de déterminer complètement la position du troisième élément, seulement deux comparateurs sont nécessaires. Il s'agit de comparer le troisième élément avec les éléments de la première et de la deuxième position. L'aiguilleur peut ensuite décaler le nombre approprié d'éléments et insérer le troisième élément à sa position.

On note ici que la première tranche de la première puce doit être une tranche spéciale afin d'ordonner parfaitement les noeuds sans connaissance de l'ordre des deux nouveaux noeuds.

Cet algorithme requiert donc 2N/3 comparateurs, 2 comparaisons par comparateur, un cycle pour entrer et sortir les items et il garantit les N/3+1 meilleurs items.

## 4.6 Pile à Insertion

Bien qui la pile à tranches ordonnées réussisse à réduire le nombre total de comparaisons de façon intéressante, la qualité du tri n'est pas améliorée de façon significative. La réduction du nombre de comparateurs est atteinte grâce à un meilleur tri de chaque tranche. Il est possible qu'un grossissement de la tranche jusqu'à l'englobement complet de la pile résulte en une amélioration additionnelle.

Ceci implique un tri complet de la pile. Si la pile est parfaitement ordonnée, le problème se résume au placement des nouveaux noeuds à leurs positions finales de façon à conserver un tri parfait. Lorsqu'un nouvel élément se présente, on le compare avec tous les autres éléments de la pile simultanément. Il décale ensuite tous les éléments de qualité inférieure vers le haut d'une position et on insére le nouvel élément dans la position ainsi libérée. La pile demeure évidemment ordonnée après cette opération.

Il est évident que le nombre de comparateurs ne sera pas réduit, mais par contre, la qualité du tri s'en trouvera améliorée.

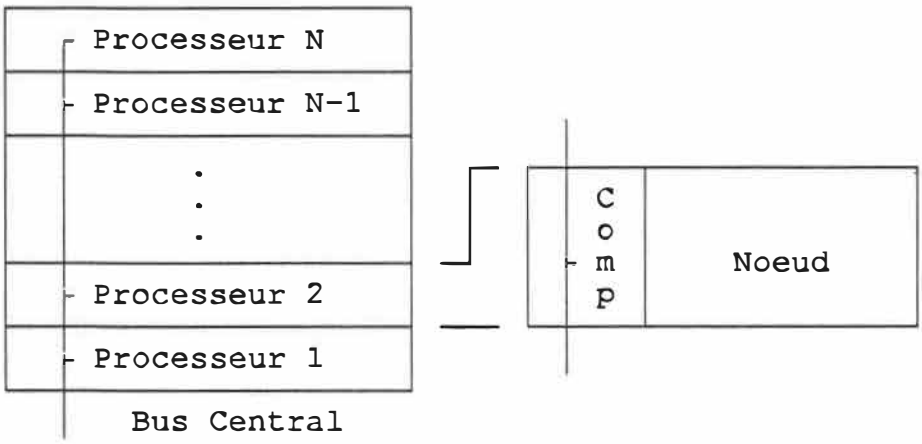


Figure 4.5 Pile à insertion utilisant une entrée séquentielle des données. Les processeurs communiquent également entre voisins immédiats.

L'algorithme de tri par insertion est un algorithme semi-systolique. Tout comme l'algorithme de Chang et Yao, il peut être utilisé dans le cas général pour tous les problèmes de tri. Bien que plus rapide et effectuant un tri d'une qualité supérieure, le prix à payer en complexité et en surface peut être non négligeable. En effet, le nouvel algorithme demande le même nombre de comparateurs que la pile à entrées parallèles et l'introduction d'un bus central. L'introduction de ces deux items augmente la surface et la complexité de la pile.

Une autre considération implique le délai de propagation des signaux dans le bus central. Si le bus central devient long, l'hypothèse originale de l'insignifiance du temps de transfert des données n'est plus valide. Pour palier à ce problème, la pile totale de N entrées ($N \geq 500$) serait composée de plusieurs puce de n entrées ($n \leq 50$). Chaque puce utiliserait l'algorithme de tri par insertion mais au lieu de propager le bus d'une puce à l'autre, le dernier noeud de chaque puce est transmis. La pile totale est donc systolique globalement mais semi-systolique localement.

Le problème avec l'approche hybride est justement l'interface entre les puces. L'algorithme de tri par insertion impose que la totalité de la pile soit ordonnée. La

solution hybride ne conserve pas un tri parfait et entraîne un bris de l'algorithme lorsqu'un noeud ne faisant pas partie des n premiers entre dans la pile. Dans un tel cas, ce mauvais noeud irait se loger dans la position n, les noeuds de qualité supérieure à ce dernier mais occupant des positions supérieures à n ne pourraient plus jamais être retirés.

La solution à ce problème consiste à introduire un registre de sortie. Lors de l'insertion chaque pile vide son registre de sortie. La pile effectue son cycle d'insertion de façon normale en considérant le registre de sortie comme la dernière position de la pile. Lors du décalage vers le bas, le registre de sortie ne fait pas partie de la pile. C'est-à-dire que le meilleur noeud de la puce supérieure entre dans la position inférieure au registre de sortie et que le contenu de ce dernier n'est pas altéré.

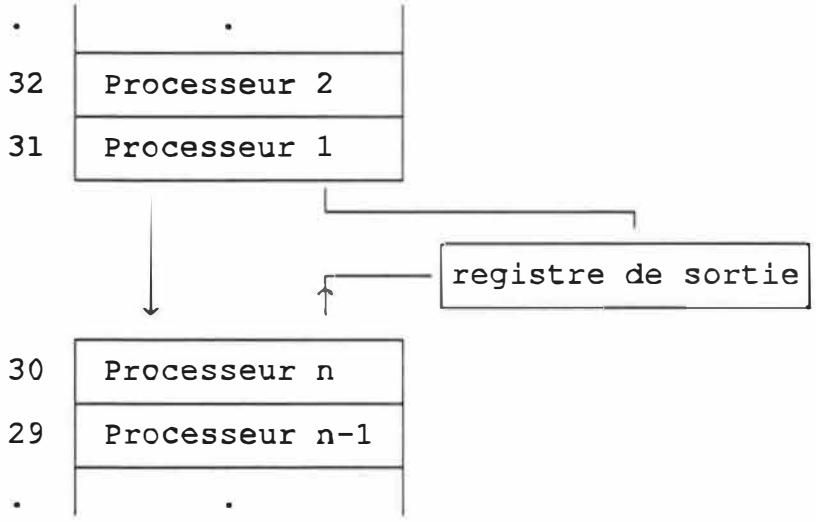


Figure 4.6 Cascade de piles à insertion démontrant l'utilisation du registre de sortie. Les chiffres de gauche indiquent la position du noeud dans la pile totale (n = 30, N >> 30).

Bien que l'algorithme par insertion n'empêche pas une version parallèle de la pile, la réalisation de celle-ci n'est probablement pas aussi attrayante. Une telle pile nécessiterait 2 bus centraux, 2N comparateurs et demanderait une logique de contrôle

capable de décaler les items de 0, 1 ou 2 positions. La logique serait encore plus complexe si on ne connaissait pas la relation entre les deux éléments à insérer.

Bien que trois cycles soient nécessaires dans la version de base de cet algorithme, le troisième n'est que le décalage des éléments vers le bas, par conséquent son temps exécution est négligeable. La version parallèle, ou optimisée, quant à elle ne requiert qu'un cycle.

Dans sa version de base, cet algorithme nécessite N comparateurs et 2N comparaisons, 3 cycles pour entrer et sortir les items mais il garantit les N meilleurs noeuds. La version à entrées parallèles utilise 2N comparateurs et comparaisons, 1 cycle pour faire entrer et sortir les items et il garantit également les N meilleurs noeuds.

## 4.7 Comparaisons des Architectures

Les deux nouvelles variantes proposées pour la pile d'un décodeur séquentiel font clairement apparaître le compromis entre le nombre de comparateurs et la qualité de tri désirée. Les approches utilisées par les deux nouvelles architectures sont passablement différentes: la pile à tranches ordonnées demande légèrement plus de comparateurs que l'algorithme de Chang et Yao mais obtient quand même le meilleur produit AT, alors que la pile triée par insertion obtient le meilleur tri avec un des pires produit AT.

| Architecture | Comparateurs | Comparaisons | AT | Tri |
|---|---|---|---|---|
| Chang et Yao | N/2 | 3 | 3N/2 | N/2 |
| parallèle | N | 2 | 2N | N/3 |
| tranches ord. | 2N/3 | 2 | 4N/3 | 1+N/3 |
| insertion | N | 2 | 2N | N |

Tableau 4.1 Architectures de Piles.

Il devient évident que la sélection d'un algorithme sur la base du produit AT seul est inadéquate étant donné la grande variabilité dans la qualité du tri obtenu. Une fonction de coût doit alors être utilisée pour comparer avec justice les différentes architectures.

Le coût d'une architecture C sera donc défini par le rapport AT/Q où Q est défini comme la qualité du tri; meilleur sera le tri, moins le coût sera élevé. Le nouveau rapport AT/Q devient donc une mesure permettant de comparer les diverses architectures ensemble en tenant compte de la vitesse, complexité et efficacité.

| Architecture | AT | Tri (Q) | C | Cm |
|---|---|---|---|---|
| Chang et Yao | 3N/2 | N/2 | 3 | 3/2<x<3 |
| parallèle | 2N | N/3 | 6 | 3 |
| tranches ord. | 4N/3 | 1+N/3 | 4 | 2 |
| insertion | 2N | N | 2 | 2 |

Tableau 4.2 Rapport AT/Q des architectures.

La pile par insertion est nettement favorisée par cette fonction de coût. Bien que plus raisonnable que le produit AT seul, ce nouvel indice de performance, C, utilise une borne inférieure quant à la qualité du tri. Ce choix favorise la pile par insertion pour laquelle le meilleur et le pire comportement est identique. La fonction de coût moyen, Cm, pallie à cette lacune en utilisant le comportement des piles lorsqu'utilisées dans un décodeur séquentiel. Ce comportement 'moyen' fut trouvé par simulation [38] et semble suggérer que pour les piles utilisant des tranches de trois entrées, le taux d'efficacité du tri soit 2N/3 plutôt que N/3. Le comportement réel de la pile de Chang et Yao, lorsqu'utilisée dans un décodeur séquentiel, n'est pas connu. On peut néanmoins dire que sa fonction de coût, Cm, sera bornée par 3/2 dans le cas ou le tri serait parfait et par 3 s'il n'y a pas d'amélioration par rapport au pire cas. Une valeur de 2 serait plus raisonnable.

La sélection d'une architecture pour la pile systolique devient plus difficile lorsque examinée selon la fonction de coût moyen. Des avantages plus subtils doivent alors départager les différentes architectures. La pile de Chang et Yao est générale mais la qualité de tri varie et cette pile est la plus lente avec trois comparaisons par comparateur. La pile à entrées parallèles est la plus régulière mais elle est la pire au Cm. La pile à tranches ordonnées obtient une meilleur utilisation des comparateurs tout en conservant une architecture systolique pure mais elle possède une qualité de tri variable. La pile par insertion obtient le meilleur tri au prix d'une complexité supplémentaire dans le contrôle.

Le grand nombre de transistors requis par la fabrication de ces puces est l'inconvénient majeur de toutes les architectures présentées ci-haut. Même pour une architecture par insertion, cette mémoire dédiée utilise environ de 30 à 50 transistors par bit de mémoire [32] comparativement à de 1 à 10 pour une mémoire conventionnelle. Même en supposant une structure sur une puce, le nombre de transistors par bit demeure compris entre 10 et 15 [38]. Dans de telles conditions, réaliser une pile de quelques 60+ bits de largeur (15 bit pour représenter la métrique, 24 pour l'état, 11 pour la profondeur, 6 et 4 pour les symboles et la perforation) et 500 mots de profondeur peut dépasser le million de transistors. Dans un système réaliste, il est possible que la pile soit de dimension encore supérieure. Bien que la profondeur de la pile n'ait pas d'influence sur la vitesse à laquelle le tri est effectué, la quantité de matériel peut devenir excessive.

# CHAPITRE 5
# DECODAGE MULTI-CHEMINS

Les notions de base du décodage probabiliste des codes convolutionnels furent introduites au cours du chapitre 3. Les limites et avantages des techniques de décodage de Viterbi et séquentiel ont été examinées et une revue des principes de base des algorithmes multi-chemins a conclu ce chapitre.

Il a été démontré au chapitre 3 qu'on ne peut avoir à la fois une vitesse de décodage constante et une excellente performance d'erreur. La seule technique de décodage permettant de rapprocher ces deux priorités est le décodage multi-chemins en largeur (M-chemins). Le décodage M-chemins offre une solution à la variabilité de l'effort de calcul tout en maintenant un nombre de calculs par itération raisonnable. De plus, l'algorithme M-chemins est un candidat idéal à une réalisation matérielle (chapitre 6).

Dans ce chapitre, le décodage M-chemins et les techniques bidirectionnelles sont introduites. Des résultats sont présentés pour les différentes variantes étudiées.

## 5.1 Décodage Unidirectionnel

Originellement introduit [26] et surtout utilisé [27] pour le codage de source, englobé dans une famille de décodeurs généraux [11] et remis en valeur récemment [30],[29],[40] pour le décodage de codes ayant de grandes longueurs de contrainte ($K > 8$), l'algorithme M-chemins peut se décrire comme suit (codes $R = 1/V$):

1- Faire l'extension des M noeuds courants.
2- Trier les 2M nouveaux noeuds par ordre de métrique.
3- Choisir les M meilleurs noeuds.
4- Recommencer jusqu'à la profondeur finale.

Algorithme 5.1 Algorithme M-Chemins de base pour taux R = 1/V.

La figure 5.1 illustre la progression des chemins (M = 3) dans un treillis généré par un code ayant une longueur de contrainte K = 4.

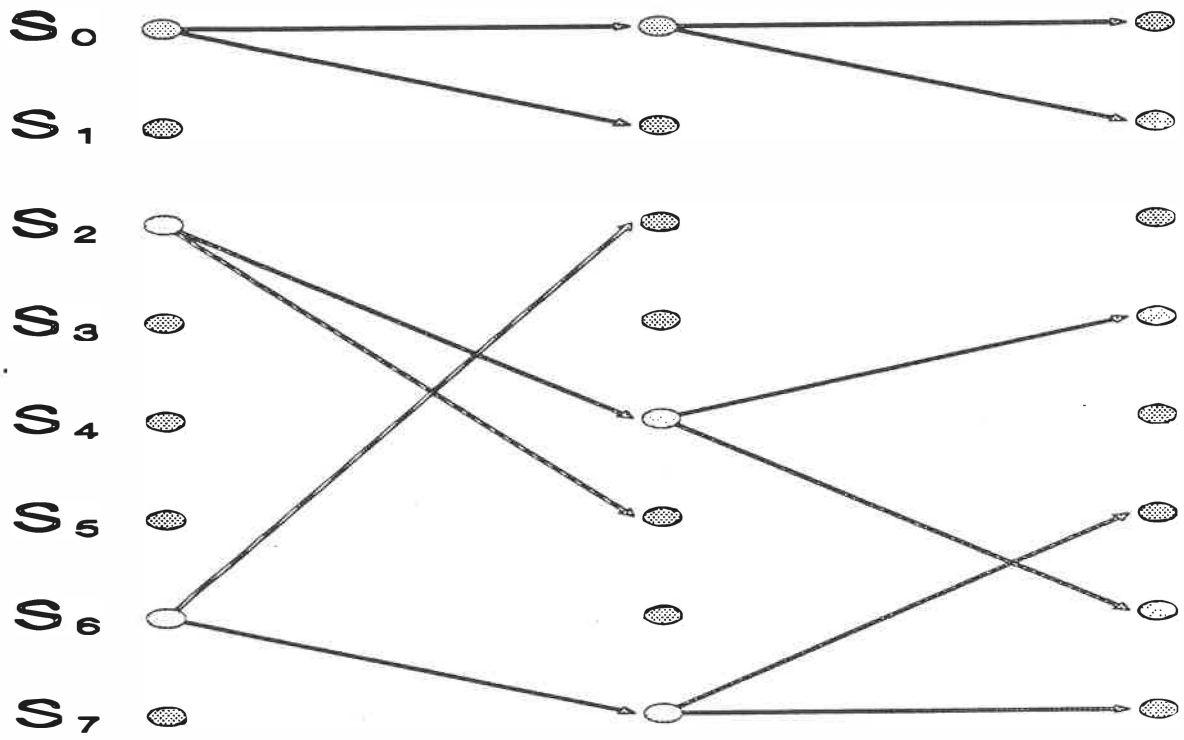


Figure 5.1 Décodage M-chemins en treillis M = 3, K = 4, R = 1/V.

L'algorithme M-chemins est un algorithme de recherche en largeur, par conséquent il ne souffre d'aucune variabilité de l'effort de calcul. L'algorithme M-chemins atteint donc le premier objectif, à savoir une vitesse de décodage constante.

N'étant pas optimal, l'algorithme M-chemins ne fait l'extension que d'un petit nombre de chemins, M, par rapport aux $2^v$ chemins requis par l'algorithme de Viterbi où

v = mémoire du codeur (K-1 si R = 1/V). Il atteint donc le deuxième objectif, soit un effort de calcul réduit. Cependant l'algorithme doit conserver une bonne performance d'erreur.

La performance d'erreur de l'algorithme M-chemins est influencée par deux types d'erreurs. Le premier type est une erreur qualifiée de type Viterbi. Le message reçu est plus près d'un autre message que du message transmis. Le chemin correct est alors éliminé. A la figure 5.2, le chemin correct, en pointillé, est éliminé par un chemin incorrect, en gras, de métrique supérieure.

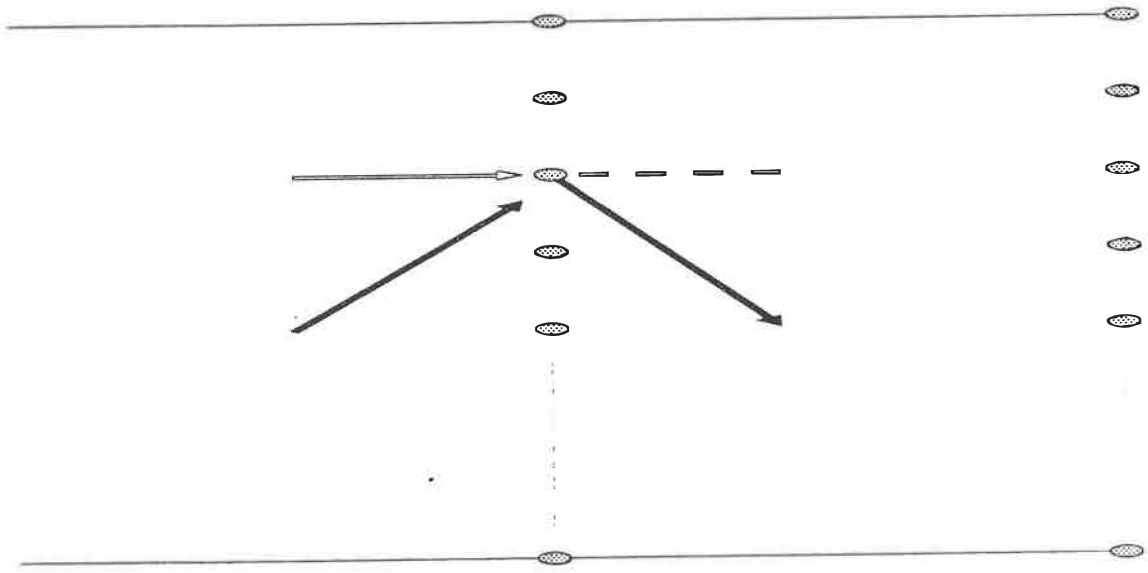

Figure 5.2 Elimination du chemin correct en décodage M-chemins. Erreur de type Viterbi.

Le deuxième type d'erreur se produit lorsque la métrique du meilleur chemin chute tellement que M métriques de chemins incorrects deviennent supérieures à la métrique du chemin correct (figure 5.3). Le chemin correct est perdu. La faible proportion d'états explorés fait que ce type d'erreur domine la probabilité d'erreur.

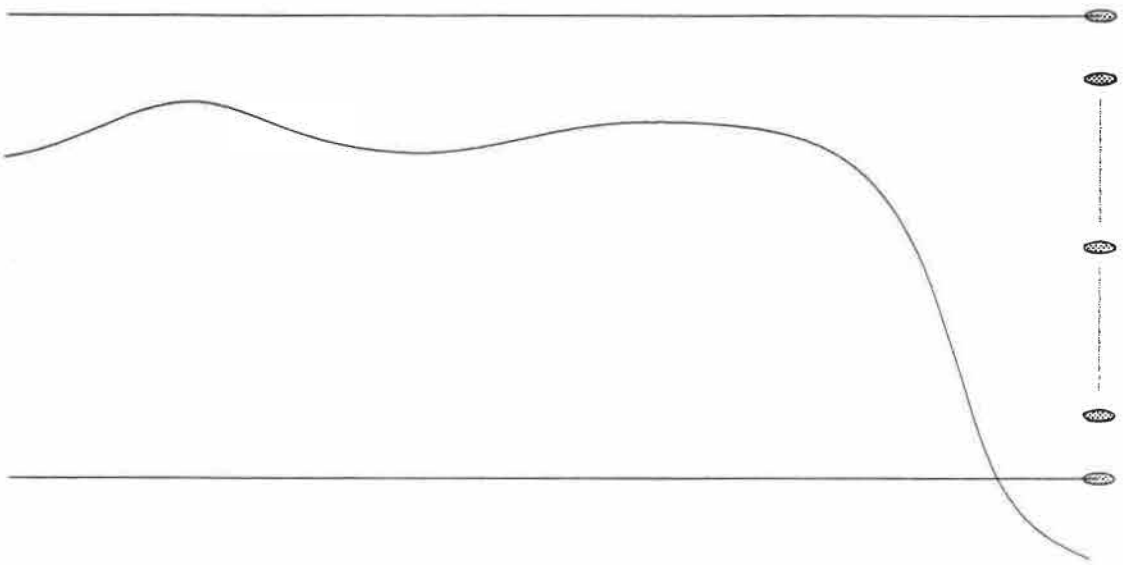

Figure 5.3 Perte du chemin correct en décodage M-chemins.

Tout comme pour l'algorithme de Viterbi, les chemins de l'algorithme M-chemins convergent après une certaine longueur, $L_C$. En décodage M-chemins, $M << 2^v$, le profil de distance de colonne (PDC) influence fortement $L_C$. Plus le PDC augmente rapidement, plus les chemins se démarquent rapidement les uns des autres (chapitre 2).

Si K est grand et que le bruit est négligeable, $L_C$ est petit, par conséquent le nombre de branches a explorer $2^{Lc}$ est également petit. Compte tenu que le bruit réduit la distance de Hamming, $d_H$, $L_C$ et le nombre de branches a explorer augmente. Etant donné un petit nombre fixe de chemins M, il est important d'essayer de conserver $L_C$ le plus petit possible sinon la nature non-exhaustive de la technique de recherche tend à causer des erreurs.

Une $d_H$ réduite implique aussi que les métriques accumulées de ces chemins tendent vers la même valeur et que le décodeur ne peut plus discerner le chemin correct des chemins incorrects. L'*écart*, une importante variable permettant de caractériser le comportement du décodeur M-chemins, est défini comme la différence des valeurs de

métriques accumulées entre le meilleur chemin et le $M^{ième}$ meilleur chemin. Cette différence permet de déterminer si le décodeur est dans une région bruitée ou non, ou encore s'il a perdu le chemin correct. L'écart est une mesure de la capacité du décodeur à absorber une séquence bruitée. Plus l'écart sera grand, plus le décodeur aura la possibilité de résoudre une chute de métrique appréciable.

Lors de décodage dans des conditions normales, l'écart demeure grand et relativement constant. Dès qu'une section du message devient fortement bruitée, l'écart tend vers zéro; la capacité du décodeur à discerner le chemin correct devient nulle. Comme dernier cas, si le décodeur perd le chemin correct, la grande mémoire du code empêche le décodeur de retrouver ce dernier et le décodage devient erroné, l'écart demeure alors petit et constant.

En bref, l'utilisation du décodage M-chemins pose une question fondamentale. Il s'agit de savoir si un code très puissant décodé de façon sous-optimale peut être plus performant qu'un code moins puissant décodé de façon optimale.

### 5.1.1 Présentation et Analyse des Résultats

Le critère d'évaluation de performance choisi sera la probabilité d'erreur par bit, $P_B$, en fonction du nombre de chemins prolongés, M. Ce critère d'évaluation permet une comparaison directe avec l'algorithme de Viterbi. Le nombre de chemins utilisés par l'algorithme de Viterbi est calculé selon la mémoire du code $2^v$.

La figure 5.4 illustre les performances d'un système M-chemins conventionnel pour plusieurs valeurs de M. Ces courbes peuvent être comparées directement avec celles de la figure 3.5.

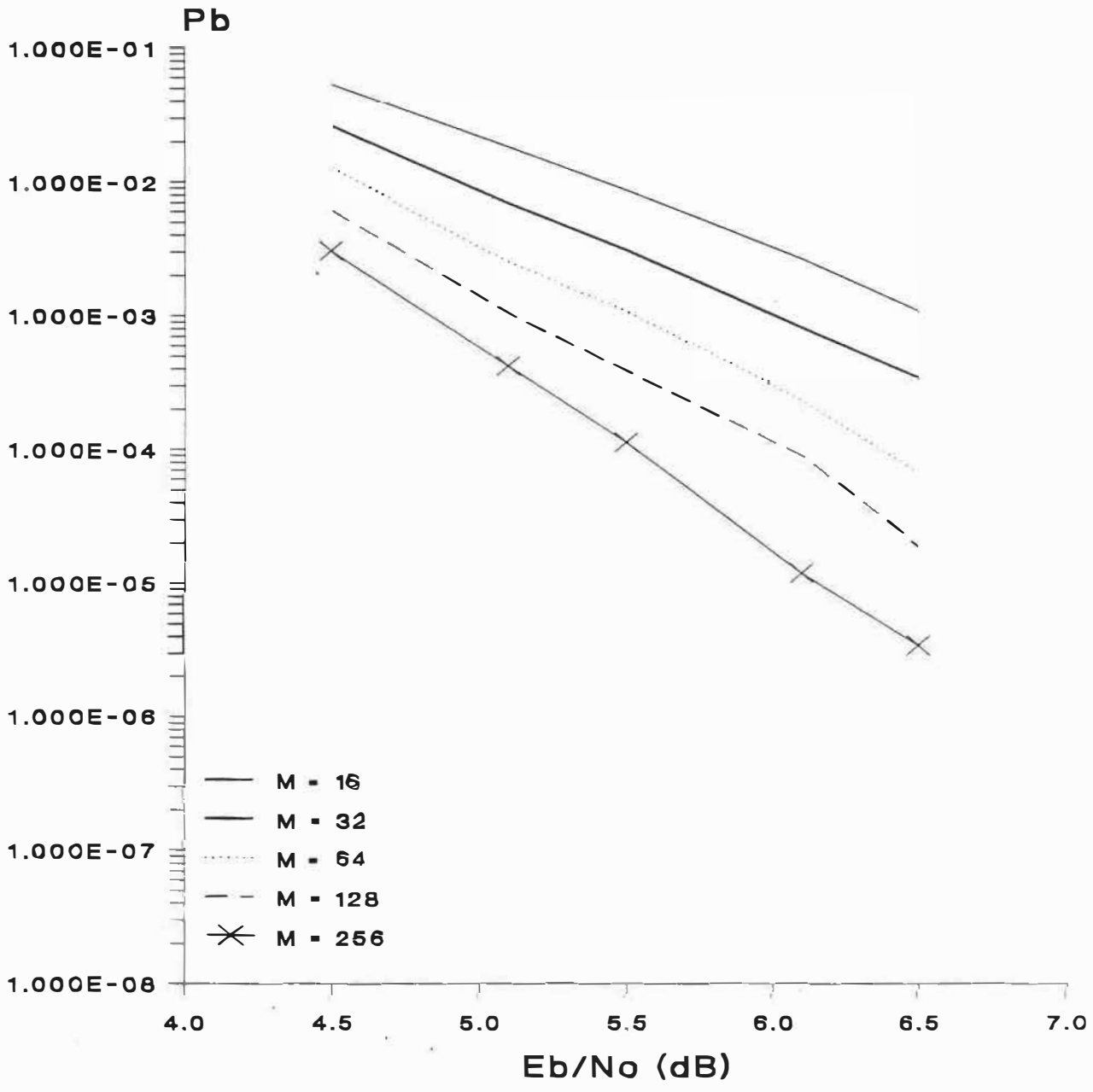


Figure 5.4 Probabilité d'erreur du décodage M-chemins unidirectionnel en fonction de $E_b/N_0$ (dB) K = 20, R = 1/2, BSC.

Le tableau 5.1 démontre bien pourquoi la probabilité d'erreur par bit de ce système est supérieure à celle d'un décodeur de Viterbi. Des 50 000 trames simulées à $E_b/N_0$ = 4.5 dB, 2482 sont en erreur soit 4.964%. De plus, chaque trame en erreur possède en moyenne 130 bits en erreur.

| $E_b/N_0$ (dB) | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Long. moy. des erreurs |
|---|---|---|---|---|---|
| 4.5 | 2482 | 324078 | 4.964E-2 | 1.296E-2 | 130 |
| 5.1 | 514 | 63783 | 1.028E-2 | 2.551E-3 | 124 |
| 5.5 | 212 | 27856 | 4.240E-3 | 1.114E-3 | 131 |
| 6.1 | 50 | 5995 | 1.000E-3 | 2.398E-4 | 120 |
| 6.5 | 16 | 1729 | 3.200E-4 | 6.916E-5 | 108 |

Tableau 5.1 Probabilité d'erreur du décodage M-chemins unidirectionnel pour plusieurs valeurs $E_b/N_0$ (dB) M = 64, K = 20, R = 1/2, BSC.

La figure 5.5 illustre un exemple typique du comportement du décodeur lors d'un événement erreur. Le chemin correct se situe parmi les chemins explorés jusqu'à la région de bruit intense. Le bruit cause la perte du chemin correct et le décodeur ne peut le retrouver si la longueur de contrainte utilisée est suffisamment grande.

Le problème de la perte ou de l'élimination du chemin correct n'aurait pas tant d'impact si le décodeur pouvait, comme le décodeur de Viterbi, retrouver rapidement le chemin correct après un événement erreur. C'est la nature exhaustive du décodeur de Viterbi qui lui donne cette possibilité.

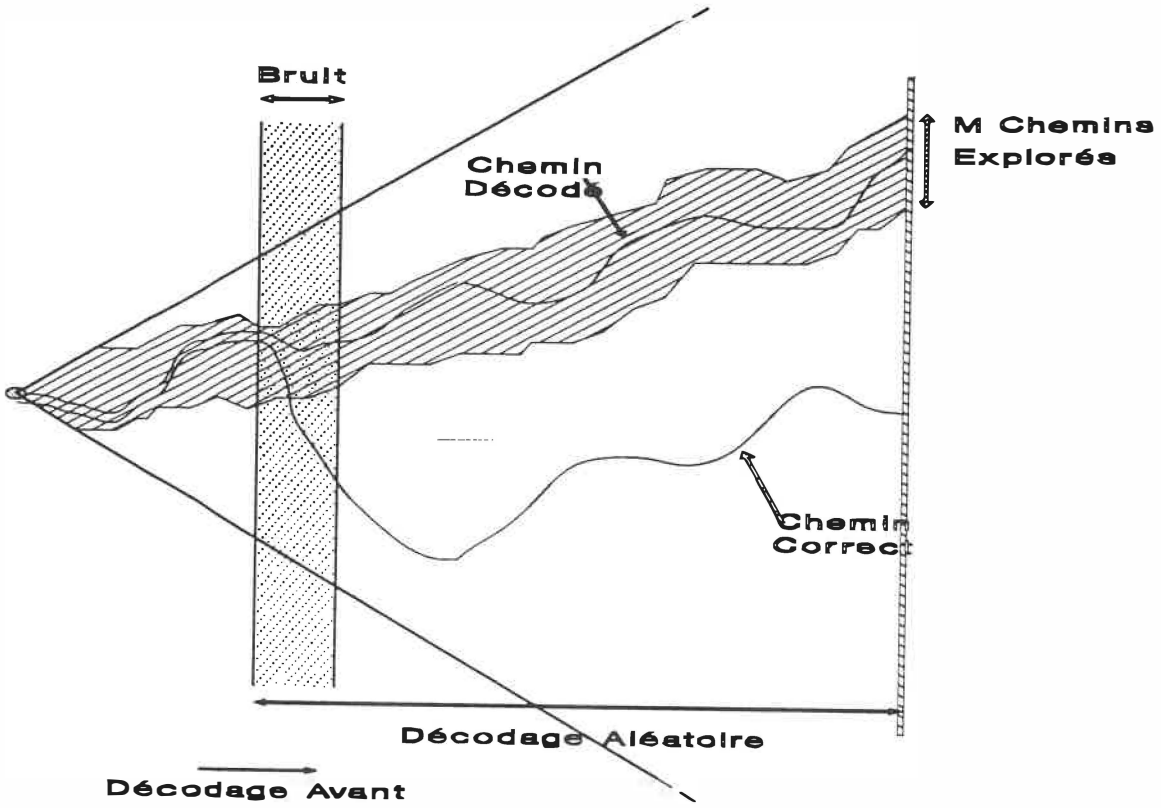


Figure 5.5 Illustration du décodage unidirectionnel.

Si on suppose que les positions des événements erreur sont uniformément distribuées et que le décodage devient aléatoire après la perte du chemin, le nombre moyen de bits en erreur lors de la perte du chemin correct est L/4 où L est la longueur de la trame. Les résultats du tableau 5.1 confirment cette analyse. Comme un événement erreur se produit de façon assez fréquente pour $E_b/N_0 = 4.5$ dB alors, la probabilité de perte du chemin correct de même que la longueur de la séquence de décodage aléatoire doivent être réduites pour espérer une bonne performance d'erreur de l'algorithme M-chemins.

Plusieurs alternatives s'offrent à nous afin de réduire la longueur de la séquence de décodage aléatoire. Premièrement, on peut augmenter le nombre de chemins

explorés et ainsi éviter la perte du chemin correct. Deuxièmement, il est possible de réduire la longueur de la trame transmise et par conséquent la longueur de la séquence de décodage aléatoire. En dernier lieu, il est possible d'essayer de retrouver le chemin correct à l'aide de méthodes algébriques.

Le tableau 5.2 représente l'effet de l'augmentation du nombre de chemins explorés sur la probabilité d'erreur de l'algorithme M-chemins. On constate qu'une augmentation du nombre de chemins ne conduit qu'à une diminution proportionnelle des événements erreurs et non à une diminution de leurs longueurs. Ce gain est nettement insuffisant pour espérer rejoindre la performance d'erreur du code. Les simulations furent effectuées sur 50 000 trames avec $E_b/N_0 = 4.5$ dB.

| M | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Long. moy. des erreurs |
|---|---|---|---|---|---|
| 16 | 10395 | 1.37E6 | 2.079E-1 | 5.481E-2 | 132 |
| 32 | 5099 | 660089 | 1.020E-1 | 2.640E-2 | 130 |
| 64 | 2482 | 324028 | 4.964E-2 | 1.296E-2 | 131 |
| 128 | 1211 | 155266 | 2.422E-2 | 6.211E-3 | 128 |
| 256 | 612 | 77931 | 1.224E-2 | 3.117E-3 | 127 |

Tableau 5.2 Probabilité d'erreur du décodage M-chemins unidirectionnel pour plusieurs valeurs du nombre de chemins étendus K = 20, R = 1/2, BSC, Eb/No = 4.5 dB.

Le tableau 5.3 montre l'effet de la variation de la longueur des trames sur la probabilité d'erreur. Il donne les performances d'un système M-chemins unidirectionnel utilisant 64 chemins et des trames de demi-longueur (L = 250). La figure 5.6 illustre la variation de la probabilité d'erreur en fonction du rapport signal à bruit d'un tel système.

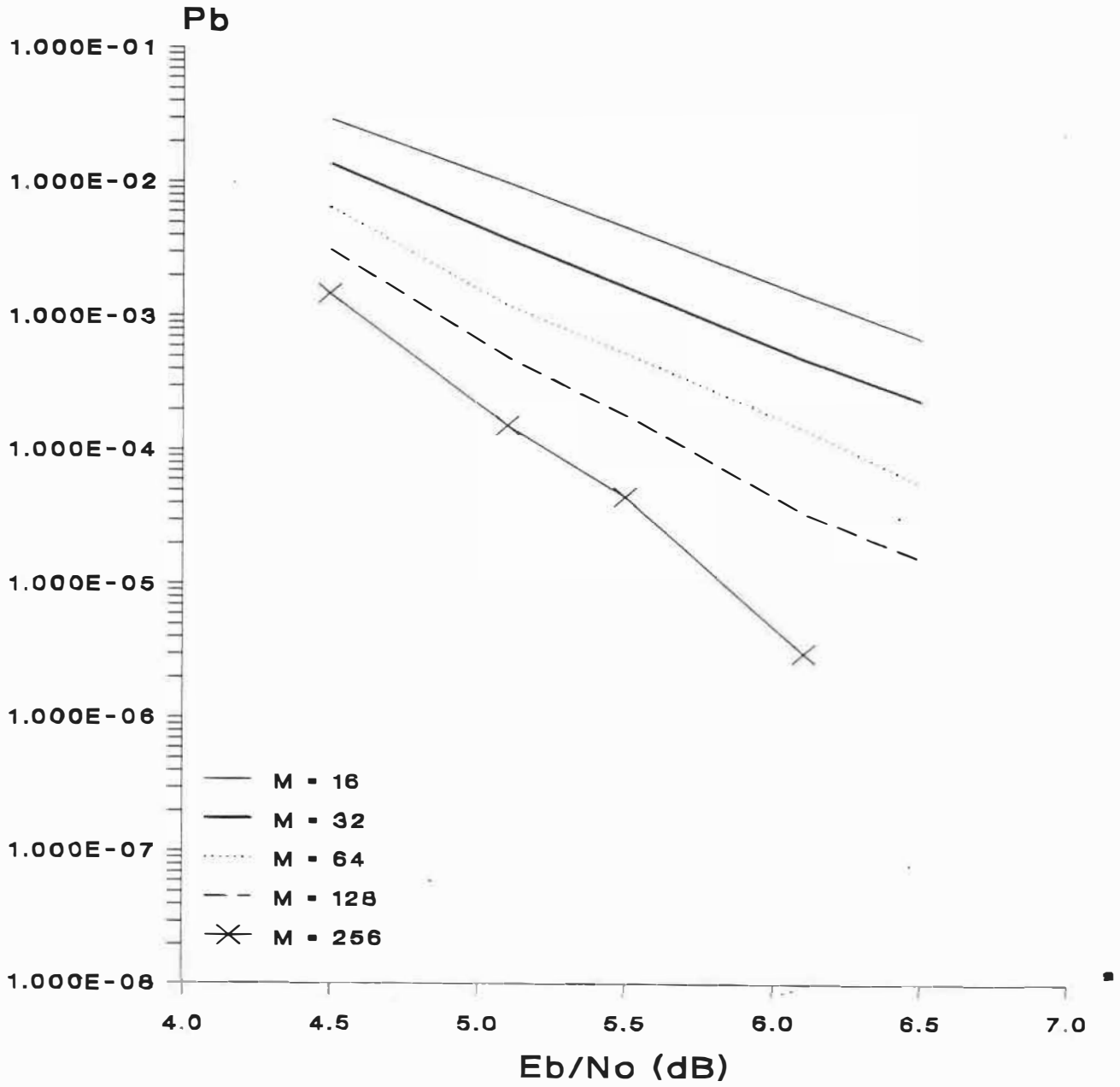


Figure 5.6 Probabilité d'erreur du décodage M-chemins unidirectionnel utilisant des trames de demi-longueur en fonction de $E_b/N_0$ (dB) K = 20, R = 1/2, BSC.

| $E_b/N_0$ (dB) | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Long. moy. des erreurs |
|---|---|---|---|---|---|
| 4.5 | 1233 | 81315 | 2.466E-2 | 6.505E-3 | 66 |
| 5.1 | 240 | 14997 | 4.800E-3 | 1.199E-3 | 62 |
| 5.5 | 107 | 6557 | 2.140E-3 | 5.245E-4 | 61 |
| 6.1 | 28 | 1768 | 5.600E-4 | 1.414E-4 | 63 |
| 6.5 | 9 | 675 | 1.800E-4 | 5.400E-5 | 75 |

Tableau 5.3 Probabilité d'erreur du décodage M-chemins unidirectionnel utilisant des trames de demi-longueur pour plusieurs valeurs de $E_b/N_0$ (dB) M = 64, K = 20, R = 1/2, BSC.

En comparant les tables 5.1 et 5.3 on remarque que la probabilité d'erreur par trame, $P_E$, ainsi que la durée des événements erreur sont réduites de moitié. Une réduction de la longueur de la trame n'améliore donc la probabilité d'erreur que linéairement. Les résultats obtenues par cette technique sont encore inférieurs à ceux obtenus par l'algorithme de Viterbi pour le même nombre de chemins prolongés (figure 3.5 et tableau 3.1).

Retrouver le chemin correct au moyen de méthodes algébriques est complexe avec des codes ayant un profil de distance optimal et ne fonctionne bien que lorsque le canal est peu bruité [30]. La complexité de telles procédures nuit à une réalisation pratique de l'algorithme. De plus une variabilité de l'effort de calcul est introduite. Même en utilisant ces procédures, la performance d'erreur du décodeur de Viterbi n'est pas encore atteinte. Ces procédures ne seront donc pas analysées plus en détail.

Les résultats obtenus dans cette section montrent qu'en moyenne le décodage M-chemins classique ne réussit pas à décoder mieux que le décodeur de Viterbi de complexité équivalente. Le décodeur M-chemins ne peut décoder de façon sous-optimale un code très puissant et obtenir une meilleure probabilité d'erreur qu'un décodeur optimum utilisant un code beaucoup plus faible.

## 5.2 Décodage Bidirectionnel

Le problème majeur du décodage unidirectionnel M-chemins est le décodage erroné du message reçu jusqu'à la fin de la trame si le décodeur perd le chemin correct. Une augmentation du nombre de chemins n'apporte qu'une amélioration proportionnelle à l'augmentation du nombre de chemins. Une réduction de la longueur de la trame n'améliore que linéairement la probabilité d'erreur. Les techniques algébriques pour retrouver le chemin correct possèdent plusieurs caractéristiques indésirables telles que l'introduction de variabilité de l'effort de calcul ainsi qu'un mauvais comportement lors des périodes de faibles rapports signal à bruit [30].

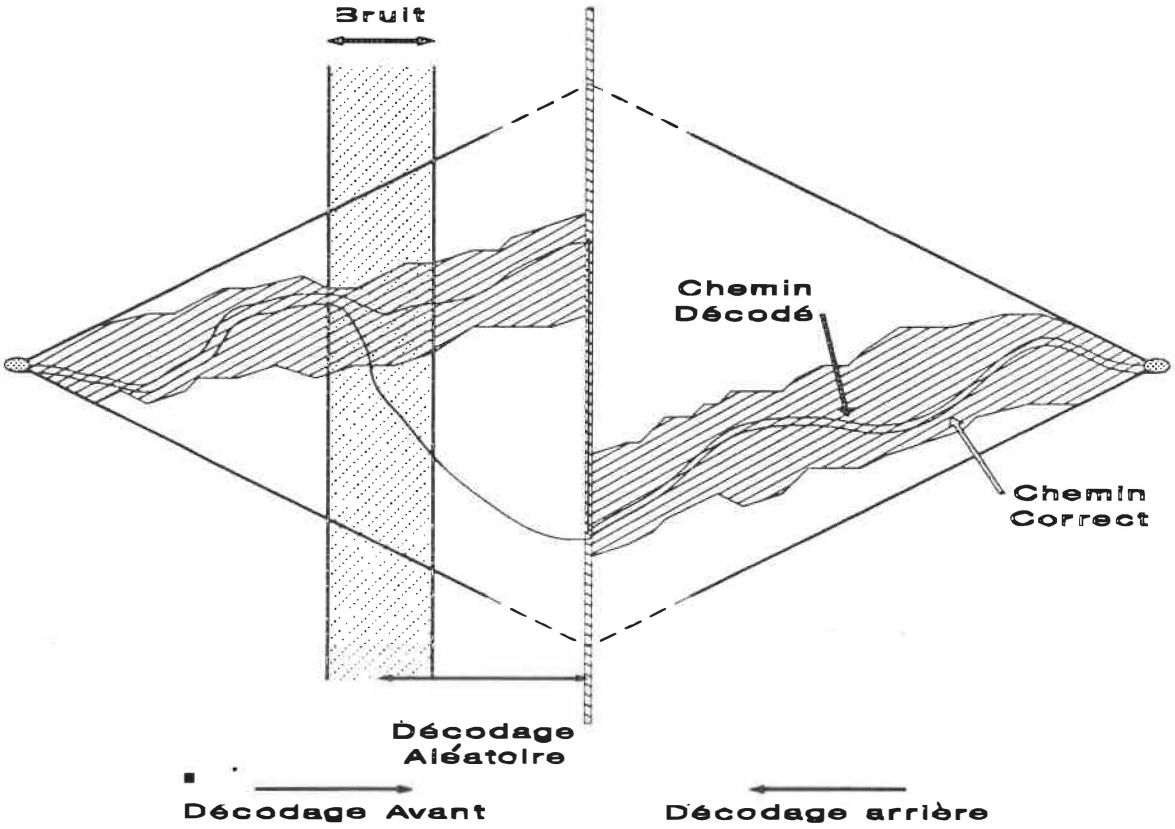


Figure 5.7 Illustration de l'algorithme M-chemins utilisant le décodage bidirectionnel.

Une réponse évidente à ces problèmes consiste à ne pas perdre le chemin correct. A la section précédente, la réduction de la longueur de la trame était la meilleure solution à ce problème. Il faut donc donner au décodeur l'illusion d'une trame plus courte tout en conservant une grande trame transmise. Cette illusion est créée par l'utilisation d'une technique de décodage bidirectionnel.

Tel qu'illustré à la figure 5.7, le décodage bidirectionnel consiste en un décodage du message reçu par le début et la fin de la trame. La longueur apparente de la trame est donc réduite de même que la période de décodage aléatoire.

Le décodage avant estime le message transmis en commençant par le premier bit d'information transmis, soit le premier bit dans le registre à décalage après la transmission de la queue de la trame précédente. Le décodage arrière estime le message transmis en commençant par le dernier bit d'information transmis soit le dernier bit dans le registre à décalage après la transmission de la queue de la trame courante.

Après la transmission de la queue, il ne reste que le dernier bit d'information dans le registre à décalage, il est donc possible de l'estimer. On encode donc les 2 possibilités pour le dernier bit d'information et on compare avec les V derniers symboles reçus. On décale ensuite l'état vers la gauche (normalement vers la droite) et on estime l'avant dernier bit d'information. Cette procédure permet de reconstruire l'information transmise à partir de la fin du message.

### 5.2.1 Jonction des Chemins

Pour que l'algorithme bidirectionnel puisse fonctionner, tous les symboles reçus doivent être utilisés et toute l'information transmise décodée. Dans cette sous-section on démontre d'abord que tous les symboles reçus du canal sont utilisés une fois et une fois seulement. Cette preuve montre que l'algorithme n'effectue pas plus de calculs que ceux requis et qu'aucune information n'est rejetée ou utilisée plus d'une fois.

Une deuxième démonstration prouve que la mémoire des encodeurs avant et arrière, au lieu de jonction, contient les mêmes bits d'information décodés. En plus de conclure que toute l'information est récupérée, cette preuve montre l'existence d'une technique de contrôle de la qualité de l'information décodée.

La sous-section se termine par une discussion des procédures à suivre en cas d'aboutement (mêmes états dans les décodeurs avant et arrière) et de non-aboutement.

**Théorème 5.1:** Tous les symboles reçus sont utilisés une fois et une fois seulement.

L'information à être codée consiste en L bits d'information et une queue de v bits. Cette information est représentée par le vecteur $\underline{U}$:

$$U_1, U_2, U_3, \ldots, U_{L+v-1}, U_{L+v}. \qquad (5.1)$$

où $U_i$ est le $i^{ième}$ bit à entrer dans le codeur. Le codeur utilise un code de taux R = 1/V. Les symboles transmis dans le canal sont représentés par $\underline{X}$:

$$\underline{X(U)} = (X_1^1, X_1^2, \ldots, X_1^V, X_2^1, \ldots, X_2^V, \ldots, X_{L+v}^1, \ldots, X_{L+v}^V). \qquad (5.2)$$

Au décodeur, les symboles transmis, $\underline{X}$, entachés de bruit sont reçus. Le signal reçu est représenté par $\underline{Y}$:

$$\underline{Y} = (Y_1^1, Y_1^2, \ldots, Y_1^V, Y_2^1, \ldots, Y_2^V, \ldots, Y_{L+v}^1, \ldots, Y_{L+v}^V). \qquad (5.3)$$

Les décodeurs avant et arrière décodent respectivement f (forward) et r (reverse) bits de sorte que la somme des bits décodés soit égale au nombre de bits transmis.

$$f + r = L + v \qquad (5.4)$$

En débutant le décodage (f = 1), le décodeur avant calcule à partir des V premiers symboles reçus, le premier bit d'information à entrer dans le codeur soit $U_1$. Le décodeur arrière part de la position r = 1 et à partir des V derniers symboles reçus calcule

le dernier bit d'information à sortir du codeur $U_L$. Le décodeur arrière est alors dans le même état qu'était l'encodeur après l'insertion du dernier bit de la queue, cependant le décodeur décode le dernier bit d'information. Autrement dit, en utilisant les symboles générés par la queue, le décodeur arrière calcule les derniers bits d'information.

Aux niveaux f et r, les décodeurs avant et arrière utilisent respectivement les V symboles reçus suivant:

$$Y_f^1, Y_f^2, \ldots, Y_f^V \tag{5.5}$$

$$Y_{L+v+1-r}^1, Y_{L+v+1-r}^2, \ldots, Y_{L+v+1-r}^V. \tag{5.6}$$

Au point de rencontre, par (5.6) et (5.4) les derniers symboles utilisés par le décodeur arrière sont:

$$Y_{f+1}^1, Y_{f+1}^2, \ldots, Y_{f+1}^V. \tag{5.7}$$

Une simple comparaison de (5.5) et de (5.7) démontre que tous les symboles ont été utilisés une fois et une fois seulement. Il est important de noter que la queue n'est pas décodée, contrairement à tous les algorithmes de décodage unidirectionnel. En effet la queue n'est jamais décodée mais plutôt supposée, ces bits ont servi à initialiser la mémoire de codeur du décodeur arrière.

Maintenant qu'il est établi que les symboles reçus sont tous utilisés une fois et une fois seulement, il convient de démontrer que sauf erreur, les états des deux décodeurs sont identiques car ils sont composés des mêmes bits d'information.

**Théorème 5.2:** Au point de jonction, l'état des décodeurs avant et arrière provient de la même région d'information.

Avant de commencer à décoder, f = r = 0, l'état du décodeur avant contient la queue de la trame précédente alors que celui du décodeur arrière contient la queue de la trame courante, soit respectivement:

$$\overline{U}_0, \overline{U}_{-1}, \overline{U}_{-2}, \ldots, \overline{U}_{-(v-1)} \tag{5.8}$$

$$\overline{U}_{L+v}, \overline{U}_{L+v-1}, \overline{U}_{L+v-2}, \ldots, \overline{U}_{L+1} \tag{5.9}$$

L'information contenue dans les registres à décalage des deux décodeurs est lue de gauche à droite. Comme auparavant, le décodeur arrière commence à la position L et les bits d'information entrent à l'arrière du registre à décalage.

Après f et r calculs les deux décodeurs se retrouvent respectivement dans les états suivants:

$$\overline{U}_f, \overline{U}_{f-1}, \overline{U}_{f-2}, \ldots, \overline{U}_{f-(v-1)} \tag{5.10}$$

$$\overline{U}_{L+v-r}, \overline{U}_{L+v-1-r}, \overline{U}_{L+v-2-r}, \ldots, \overline{U}_{L+1-r} \tag{5.11}$$

En combinant (5.11) et (5.4) on obtient:

$$\overline{U}_f, \overline{U}_{f-1}, \overline{U}_{f-2}, \ldots, \overline{U}_{f-(v-1)} \tag{5.12}$$

Il apparaît alors clairement que la mémoire du registre à décalage avant (5.10) est identique à la mémoire du registre à décalage du décodeur arrière (5.12). Ces concepts sont illustrés à la figure 5.8.

Si M = 1, si le canal n'est pas bruité et si les décodeurs ont calculé conjointement L+v branches alors les deux chemins auront le même état. Si M > 1, alors il existera une paire d'états, corespondant à un chemin de chaque coté, pour laquelle les contenus seront identiques. La probabilité qu'il existe plus d'une paire correspond à la probabilité d'avoir a partir du lieu de jonction deux séquences symétriques de v bits. Cette probabilité est d'environ $2^{-v}$ même en présence de bruit, et par conséquent elle est négligeable.

Par contre si le bruit est intense et qu'au moins un des décodeurs perd le chemin correct, en supposant un comportement aléatoire de ce décodeur, la probabilité qu'une paire de chemins ait un état identique correspond à $[(2^v\text{-}M)!]^2/((2^v\text{-}2M)!2^v!)$.

Cette probabilité est petite lorsque $M << 2^v$, ce qui est généralement le cas lors de l'utilisation de puissants codes correcteurs d'erreurs.

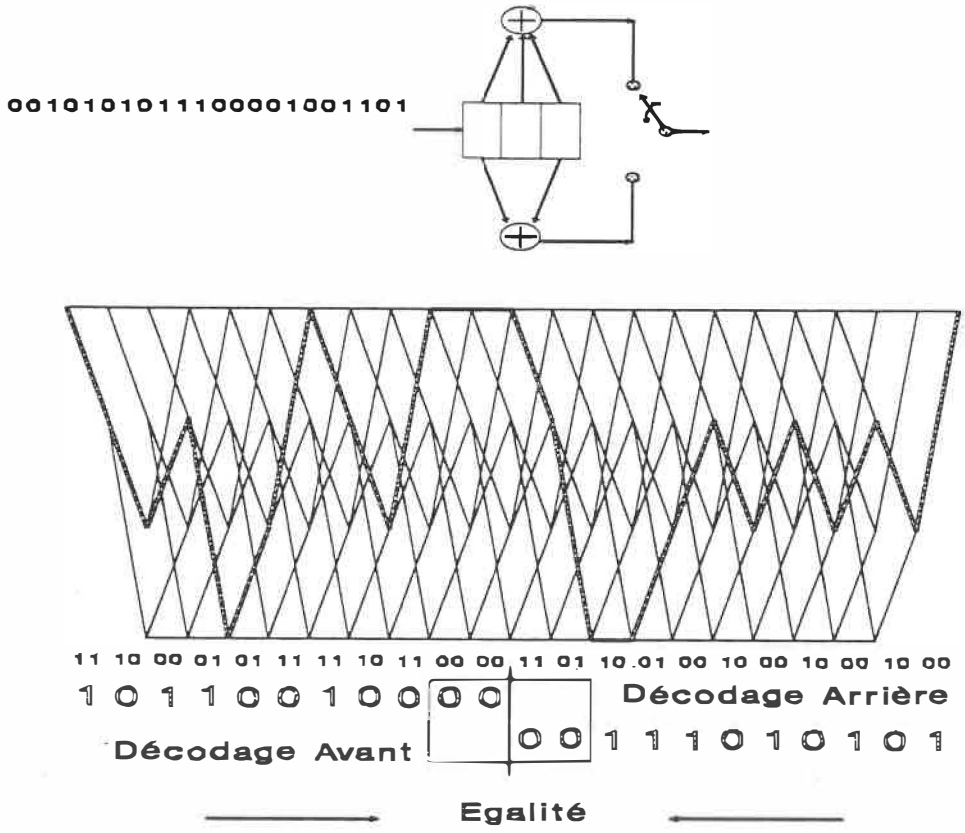


Figure 5.8 Rencontre des chemins en décodage bidirectionnel. Les états sont identiques, la trame est probablement bien décodée. L = 20, v = 2, f = r = 11.

A la jonction, si une paire d'états identiques existe, le message décodé est composé de la concaténation de la séquence avant avec la séquence arrière. La partie commune peut être fournie par l'un ou l'autre des deux décodeurs.

S'il n'y a pas de paire d'états identiques plusieurs options sont disponibles. La plus simple est de déclarer la trame impossible à décoder et de demander une retransmission. Cette approche n'est valide que dans des systèmes où les retransmissions sont permises.

Une option préférable est de favoriser une technique qui tenterait de délivrer un 'meilleur chemin possible' mais en sachant bien que ce chemin contient probablement des erreurs. Un chemin peut être exempt d'erreurs si la procédure de jonction permet l'élimination de certaines erreurs. Il est possible de déclarer 'chemin le plus probable' le chemin composé de la concaténation du meilleur chemin du décodeur avant et arrière, soit les chemins ayant la meilleure métrique accumulée dans chaque direction. Le décodeur possédant la métrique accumulée la plus élevée peut fournir la région commune aux deux décodeurs, éliminant ainsi certaines erreurs. C'est cette technique qui fut utilisée lors des simulations.

Bien qu'efficace et simple, cette technique ne tient pas compte de l'écart des métriques des M chemins lorsque le décodeur éprouve des difficultés. Une meilleure façon de choisir les deux séquences utiliserait cette information. Par exemple, il est possible de choisir le chemin (A) ayant la plus forte métrique dans la direction où l'écart entre les chemins est le plus grand et ensuite de tenter de choisir un chemin (B) de l'autre direction dont la distance de Hamming entre bits d'information avec A est minimale. Le coté où l'écart est maximal est normalement plus fiable que l'autre et le chemin de métrique supérieure de ce coté possède plus de chance d'être le chemin correct. Si le décodeur ayant ses chemins groupés n'a pas perdu le chemin correct depuis longtemps, il est alors concevable que le nombre de bits en erreur soit minimisé.

Une quatrième alternative au problème de l'aboutement consiste à continuer le décodage dans chacune des directions et de vérifier si l'un des chemins d'une des

directions ne rencontrerait pas un des chemins de l'autre direction. Autrement dit, on agrandit la fenêtre d'aboutement mais tout en conservant le même critère de sélection. Bien qu'il soit à peu près impossible que la direction ayant perdu le chemin correct retrouve se dernier et se recombine ainsi avec l'autre décodeur, il est vraisemblable que ceci se produise dans l'autre direction.

Cette dernière méthode introduit une variabilité dans le temps de calcul. Cependant la variabilité est bornée par le temps requis aux décodeurs pour décoder une trame en entier. La difficulté principale de cette technique consiste a comparer les états de tous les chemins d'une direction avec les états de tous les chemins de l'autre direction après chaque calcul.

En terminant, on doit souligner trois concepts importants. Premièrement, si tous les symboles n'ont été utilisés qu'une seule et unique fois et qu'il existe une région commune aux deux décodeurs alors chacun des décodeurs a décodé cette région de façon indépendante. Ceci implique qu'au lieu de jonction, probablement un endroit bruité, les décodeurs ont utilisé en tout 2vV symboles pour décoder v bits.

Deuxièmement au point de jonction, la région d'information décodée étant la même et n'ayant au maximum qu'une paire de chemins identiques, alors le nombre de chemins utilisés pour chercher le chemin correct dans cette région est de 2M-1. Ces deux facteurs influencent de façon positive la performance d'erreur du décodage bidirectionnel.

Troisièmement s'il y a aboutement au lieu de jonction, on peut affirmer avec une très faible probabilité d'erreur que les deux décodeurs n'ont pas requis de plus de M chemins et que, par conséquent le message fut décodé avec toute la puissance du code. Si par contre aucun des chemins avant ne rencontre aucun des chemins arrières, il est possible d'affirmer qu'au moins une des directions a perdu le chemin correct et que le message décodé contient probablement des erreurs.

Bien que tous les types de décodeurs puissent utiliser une technique de décodage bidirectionnel, les avantages ne sont pas les mêmes pour tous. Seule l'architecture M-chemins sera considérée ici.

### 5.2.2 Résultats du Décodage M-Chemins Bidirectionnel à Longueurs Constantes

Un décodeur bidirectionnel sera considéré à longueurs constantes si f et r sont constants. Ces deux longueurs peuvent être biaisées afin de tenir compte d'un certain rapport des profils de distance de colonne du code et du code contraire. Le code contraire est défini comme l'inversion des positions des connexions de chaque générateur du code original.

Les avantages de cette méthode se situent au niveau de sa régularité et sa facilité de réalisation. Deux machines pouvant effectuer du décodage unidirectionnel peuvent être combinées en une machine bidirectionnelle avec facilité. Bien qu'en apparence identique à une machine unidirectionnelle utilisant des trames plus courtes, ce type de décodeur possède encore les avantages d'une détection d'erreurs supérieure, d'un taux de codage effectif plus près du taux désiré et d'un nombre de chemins doublé au lieu de rencontre.

La figure 5.9 illustre la performance d'erreur d'un décodeur M-chemins utilisant le décodage bidirectionnel avec f = r = 250. En comparant celle-ci avec la figure du décodage M-chemins unidirectionnel utilisant des demi-trames (figure 5.6) on constate que la probabilité d'erreur $P_B$ du décodage bidirectionnel est légèrement inférieure. La supériorité de l'algorithme bidirectionnel s'explique par un décodage plus intelligent de la région commune lors de l'absence d'aboutement. Par conséquent, on peut dire que la technique d'aboutement choisie donne un gain d'environ 0.2 dB à des rapports signal à bruit élevés.

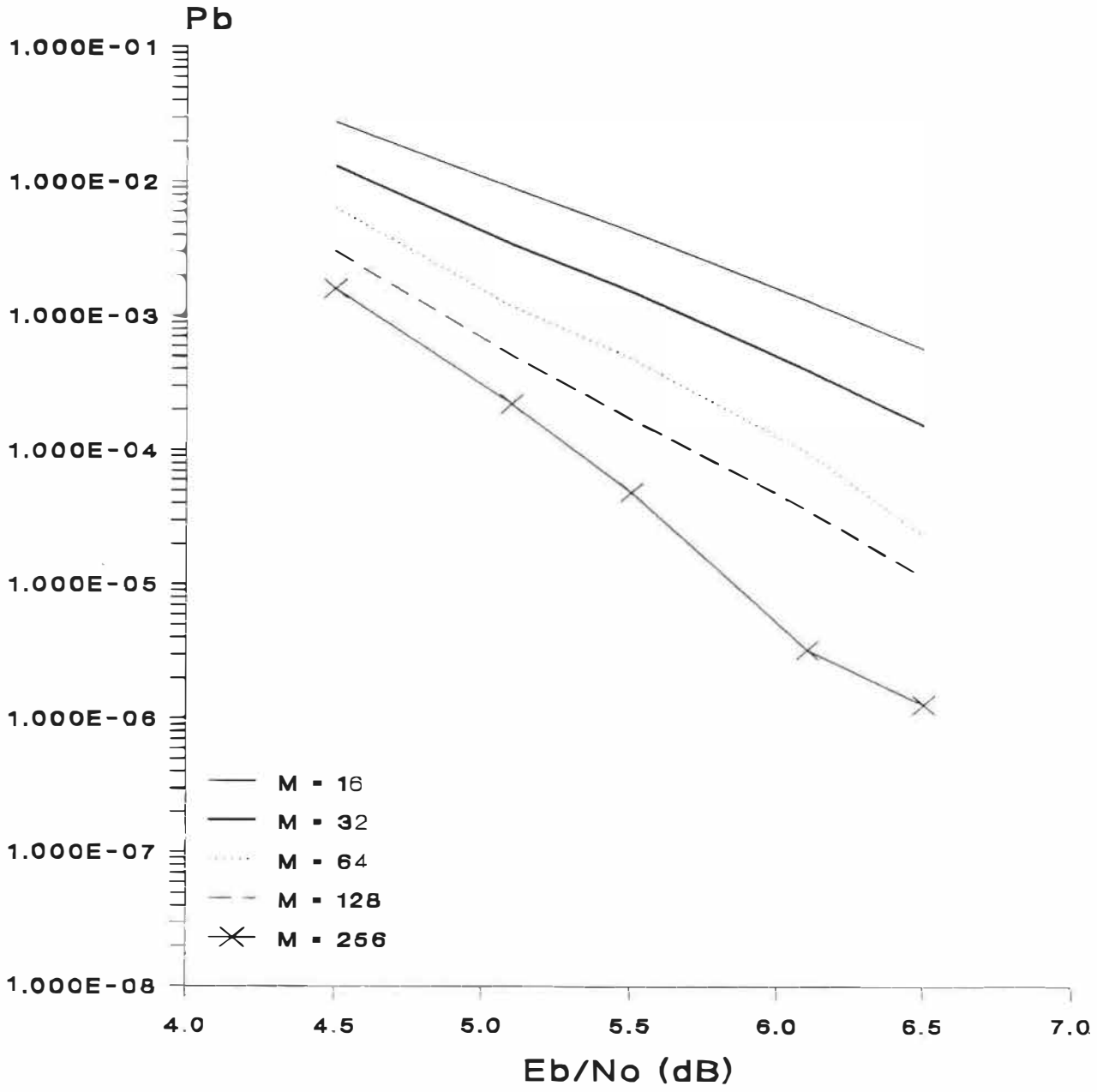


Figure 5.9 Probabilité d'erreur du décodage M-chemins bidirectionnel utilisant des longueurs constantes en fonction de $E_b/N_0$ (dB) (f = r = 250) K = 20, R = 1/2, BSC.

| $E_b/N_0$ (dB) | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Durée moy. des erreurs |
|---|---|---|---|---|---|
| 4.5 | 2594 | 161048 | 5.188E-2 | 6.441E-3 | 62 |
| 5.1 | 520 | 29822 | 1.040E-2 | 1.192E-3 | 57 |
| 5.5 | 209 | 12136 | 4.180E-3 | 4.854E-4 | 58 |
| 6.1 | 48 | 2431 | 9.600E-4 | 9.724E-5 | 51 |
| 6.5 | 18 | 583 | 3.600E-4 | 2.332E-5 | 32 |

Tableau 5.4 Probabilité d'erreur du décodage M-chemins bidirectionnel utilisant des longueurs constantes pour plusieurs valeurs de $E_b/N_0$ (dB) (f = r = 250) M = 64, K = 20, R = 1/2, BSC.

Le tableau 5.4 montre la probabilité d'erreur du décodage M-chemins bidirectionnel pour lequel f = r = 250 pour plusieurs rapports signal-à-bruit. Bien que réduisant la durée d'un événement erreur, la probabilité d'avoir une trame en erreur $P_E$ n'est pas réduite par rapport au décodage unidirectionnel (tableau 5.1). Le gain n'est donc pas suffisamment attrayant pour justifier la complexité qu'amène l'aboutement des chemins.

### 5.2.3 Résultats du Décodage M-Chemins Bidirectionnel à Longueurs Variables

Le décodage bidirectionnel à longueurs constantes réussit à diminuer la région de décodage aléatoire en réduisant la longueur apparente de la trame. Pour réduire encore la région de décodage aléatoire on doit détecter instantanément la perte du chemin correct dans une direction et attendre que l'autre décodeur rejoigne celui qui s'est perdu.

Le décodage bidirectionnel à longueurs variables est basé sur cette idée. La description de l'algorithme est la suivante:

1- Explorer simultanément les lgM premières branches de chaque direction.
2- Calculer la probabilité de chaque direction de suivre le chemin correct.
3- Avancer d'un pas la direction ayant la plus grande probabilité de suivre le chemin correct.
4- Recommencer à l'étape 2 jusqu'à ce que les deux directions se rejoignent.

Algorithme 5.2 Algorithme bidirectionnel utilisant des longueurs variables.

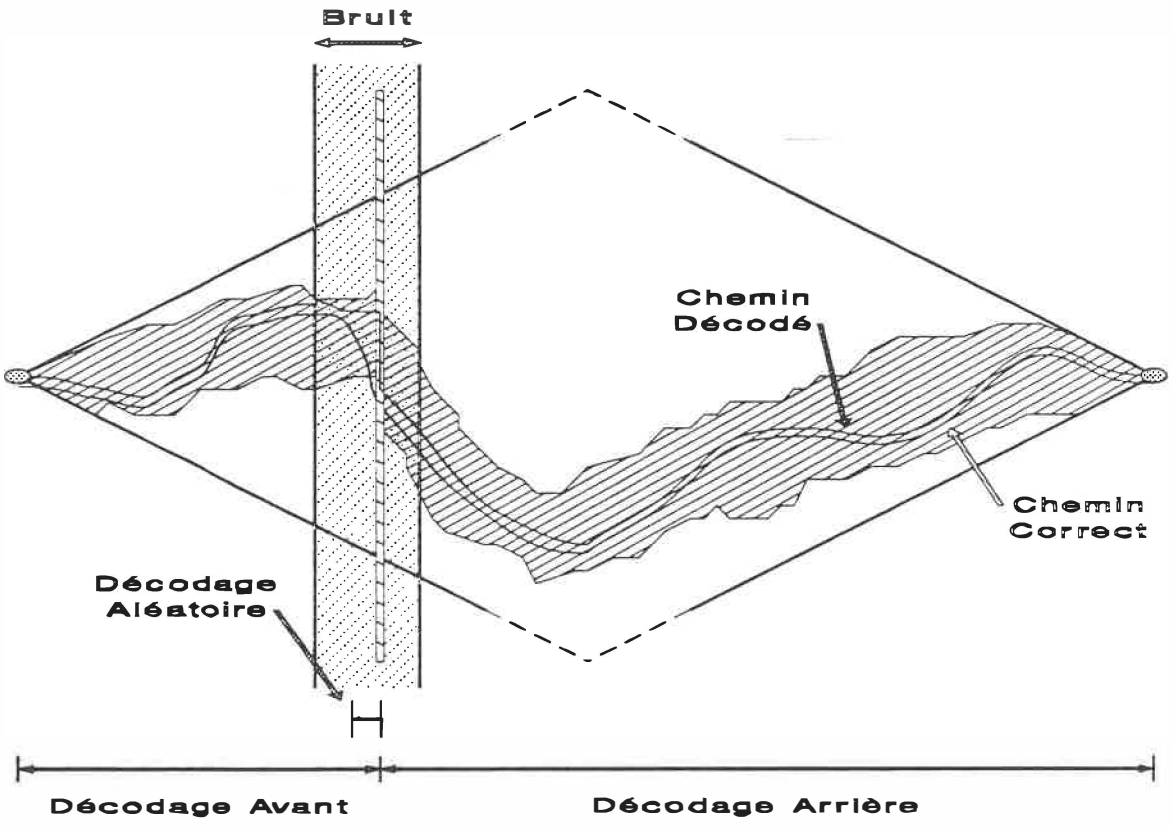


Figure 5.10 Illustration du décodage bidirectionnel à longueurs variables.

La figure 5.10 illustre cette technique. Le décodeur avant ayant calculé une forte probabilité d'événements erreur arrête sa progression. Le décodeur arrière décode alors la majeure partie de l'information. La région de décodage aléatoire est donc extrêmement restreinte.

Comme dans le cas de l'aboutement, plus d'une alternative est possible afin de décider quel décodeur possède la plus forte probabilité de ne pas s'être égaré. La première possibilité est de faire l'extension de la direction ayant la métrique accumulée maximale. Cependant cette décision favorise le chemin le plus long. Ainsi la métrique d'un long chemin qui rencontre une région bruitée prendra longtemps à descendre au niveau du chemin court même si ce dernier n'a pas subit beaucoup de bruit. Une décision indépendante de la longueur respective des chemins est préférable.

La deuxième possibilité, utilisée lors des simulations, est de faire la prochaine extension à partir du décodeur possédant le plus grand écart entre son meilleur et son pire chemin. Tel que mentionné au début de ce chapitre, l'écart peut être vu comme étant la capacité d'absorber du bruit. Il est donc logique de favoriser la direction la plus résistante au bruit.

Une troisième éventualité est dérivée de l'analyse de la longueur de convergence $L_C$. Il est possible d'avancer dans une direction jusqu'à ce que les bits sortant de l'état ne soit plus identiques. A ce point, on peut considérer que le décodeur est en train de perdre le chemin correct. Dans les trois cas, si les directions sont équiprobables, on alterne à chaque pas jusqu'à ce qu'une direction devienne préférable.

Le tableau 5.5 montre la probabilité d'erreur du décodage M-chemins bidirectionnel utilisant des longueurs variables pour plusieurs valeurs de rapport signal à bruit.

| $E_b/N_0$ (dB) | # de Trames en erreur | # de Bits en erreur | $P_E$ | $P_B$ | Durée moy. des erreurs |
|---|---|---|---|---|---|
| 4.5 | 1196 | 17881 | 2.392E-2 | 7.152E-4 | 15 |
| 5.1 | 163 | 1413 | 3.260E-3 | 5.652E-5 | 9 |
| 5.5 | 52 | 344 | 1.040E-3 | 1.376E-5 | 7 |
| 6.1 | 17 | 56 | 3.400E-4 | 2.240E-6 | 3 |
| 6.5 | 6 | 13 | 1.200E-4 | 5.200E-7 | 2 |

Tableau 5.5 Probabilité d'erreur du décodage M-chemins bidirectionnel utilisant des longueurs variables pour plusieurs valeurs de $E_b/N_0$ (dB) M = 64, K = 20, R = 1/2, BSC.

Ce tableau montre la réduction importante de la durée d'un événement erreur. De plus, la probabilité d'avoir une trame en erreur est devenue semblable à celle de trames de demi-longueurs (tableau 5.1). Cette réduction de la probabilité d'erreur par trame est expliquée par la procédure d'aboutement. En faisant correspondre la région fortement bruitée avec la région où le nombre de chemins et de symboles sont doublés, le décodeur évite beaucoup d'erreurs.

La figure 5.11 illustre la performance d'erreur en fonction du rapport signal-à-bruit d'un tel système. On remarque ici l'important gain de codage par rapport à l'algorithme de Viterbi (figure 3.5). On peut également apprécier la pente supérieure de la technique bidirectionnelle qui montre que, plus le rapport signal à bruit devient élevé, moins de chemins sont requis pour atteindre une performance comparable à celle d'un décodeur de Viterbi.

Bien que cet algorithme permet d'identifier plus exactement les endroits fortement bruités et de réduire au maximum la longueur des séquences en erreur, il est plus difficile à réaliser. Par contre, le gain de codage obtenu grâce à cette méthode est très important de sorte qu'à complexité équivalente, il est plus intéressant de réaliser un décodeur M-chemins bidirectionnel à longueurs variables qu'un décodeur de Viterbi.

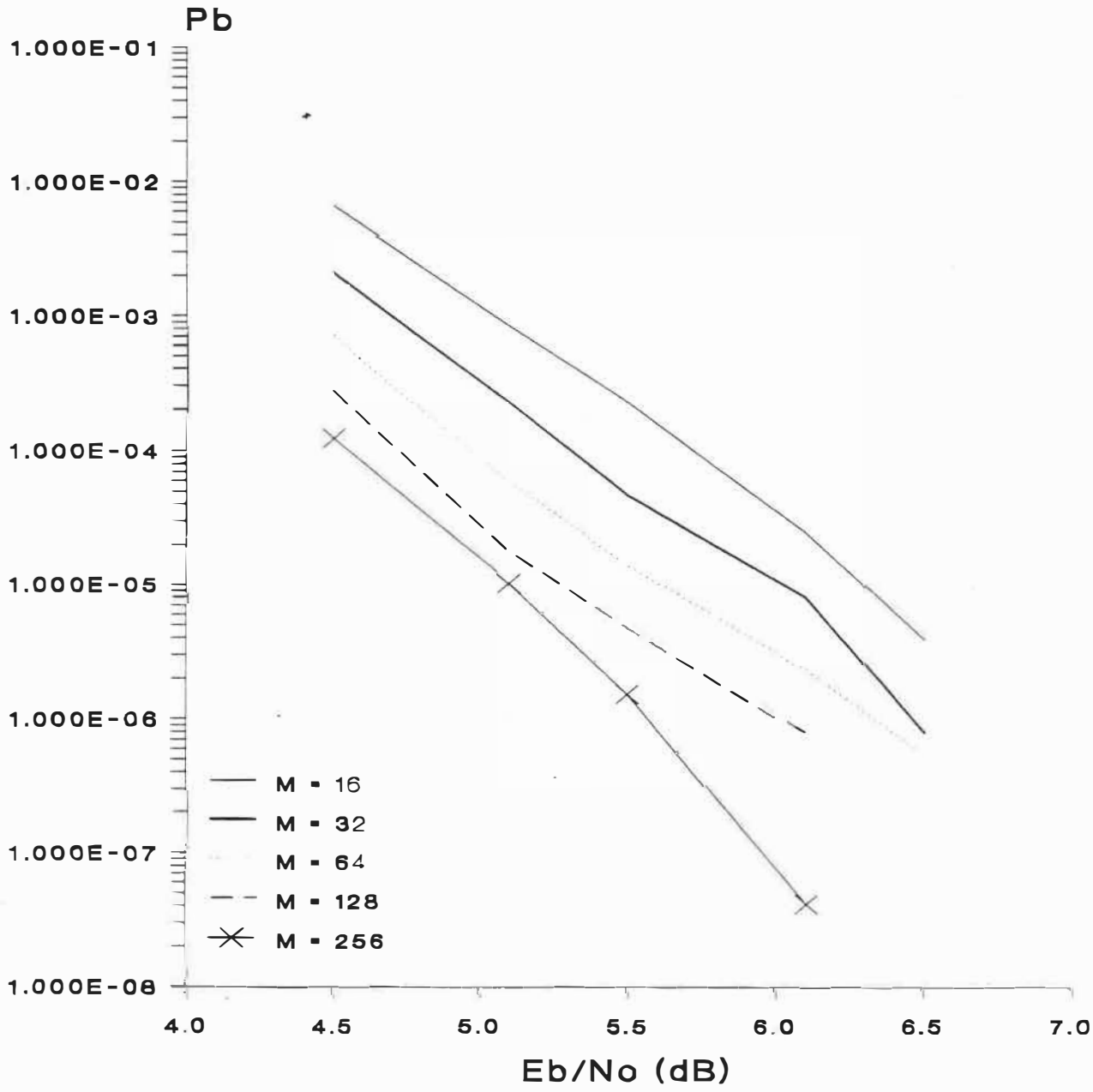


Figure 5.11 Probabilité d'erreur du décodage M-chemins bidirectionnel utilisant des longueurs variables en fonction de $E_b/N_0$ (dB) K = 20, R = 1/2, BSC.

## 5.3 Conclusions

Dans sa version traditionnelle, le décodage M-chemins n'offre pas de bonnes performances d'erreur. Toutefois lorsqu'il est utilisé en conjonction avec une technique de recherche bidirectionnelle, l'algorithme permet un gain de codage substantiel par rapport au décodage de Viterbi.

La figure 5.12 illustre bien, à complexité équivalente (64 chemins explorés), le gain de codage possible de l'algorithme M-chemins bidirectionnel à longueurs variables par rapport à l'algorithme de Viterbi. Un gain de 1 dB est possible pour une probabilité d'erreur de $P_B = 10^{-5}$. La pente des courbes indique également que le gain de codage augmentera si on désire une probabilité d'erreur plus faible.

De façon générale, les avantages du décodage bidirectionnel consistent en une amélioration de la performance d'erreur, et en une méthode presque infaillible pour détecter les trames causant des erreurs. Doubler le nombre de chemins explorés au lieu de rencontre et décoder avec des symboles indépendants une région commune sont d'autres effets secondaires positifs de l'algorithme bidirectionnel.

Un des désavantages du décodage bidirectionnel provient du code utilisé. Une des caractéristiques importantes des codes utilisés en décodage séquentiel est une croissance rapide du profil de distance de colonne (PDC). En décodage bidirectionnel, le code contraire doit être aussi bon que le code original; si tel n'est pas le cas une des directions est défavorisée.

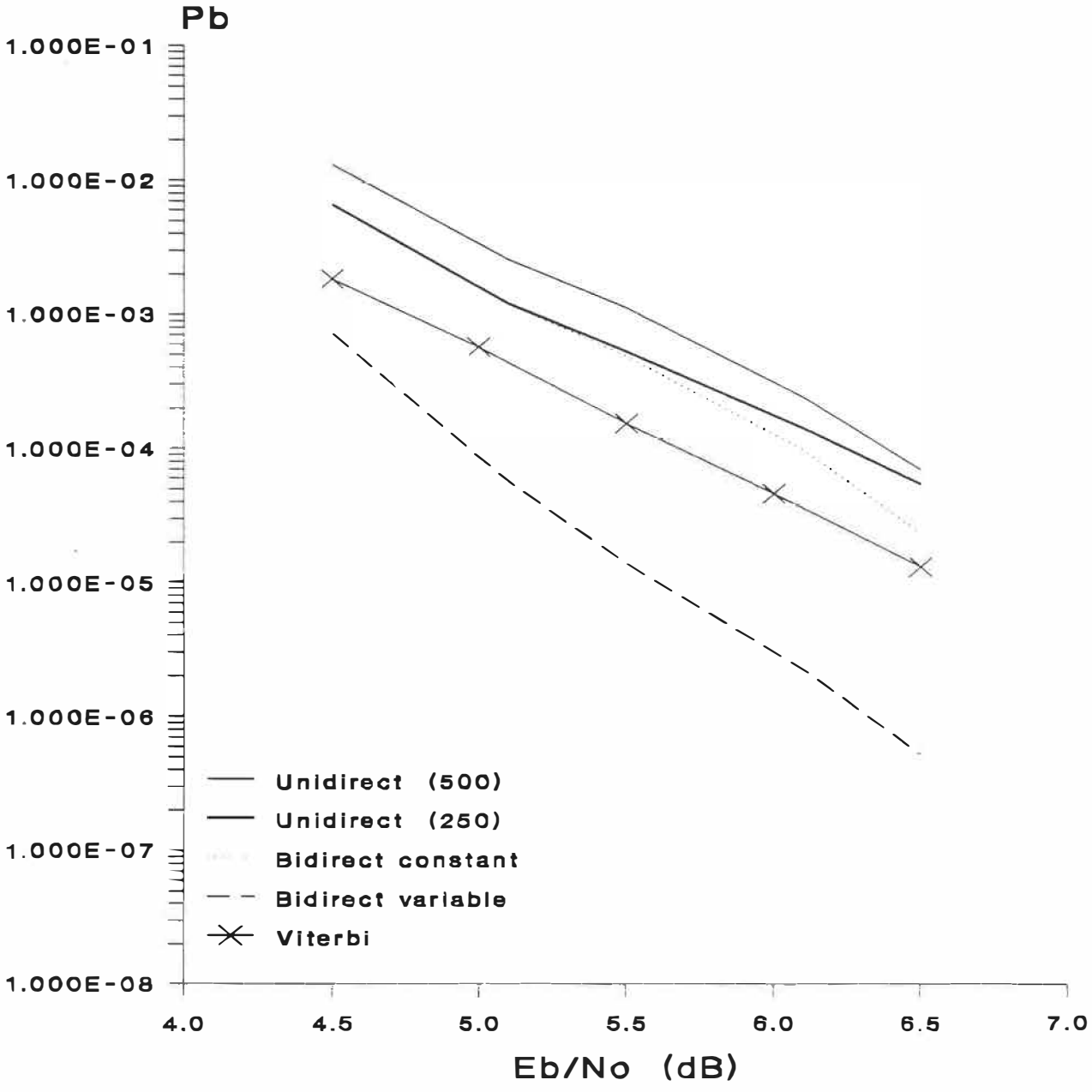


Figure 5.12 Probabilité d'erreur en fonction de $E_b/N_0$ (dB) de plusieurs algorithmes de décodage pour une complexité équivalente M = 64, K = 20, R = 1/2, BSC.

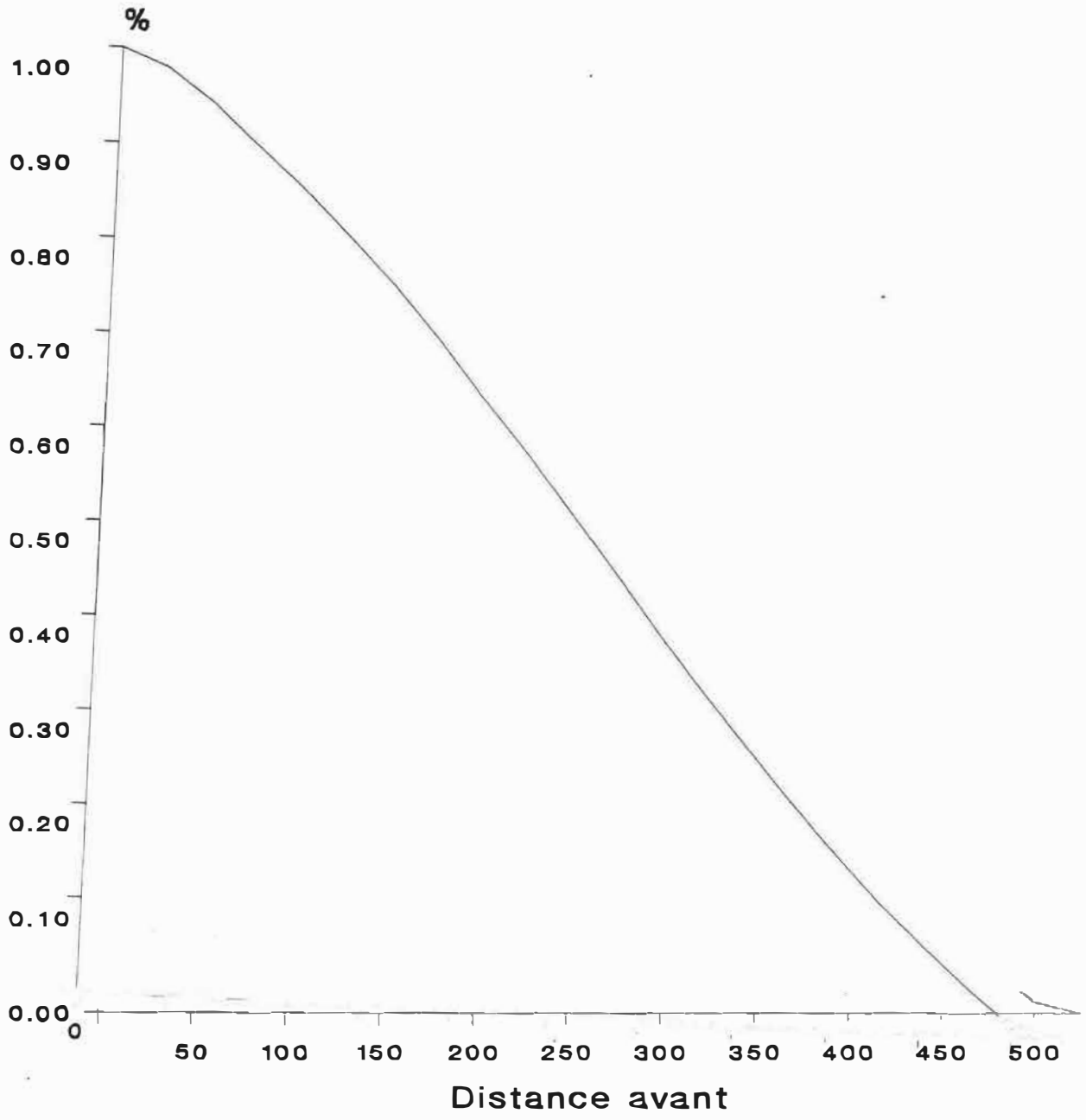


Figure 5.13 Cumulative des longueurs avant pour le décodage bidirectionnel à longueurs variables M = 64, K = 20, R = 1/2, BSC.

La figure 5.13 représente une cumulative des longueurs avant lors du décodage bidirectionnel à longueurs variables. Cette courbe est obtenue pour des trames de longueur 500, une valeur de $E_b/N_0$ de 4.5 dB et 64 chemins explorés.

On voit à la figure 5.13 que, pour le code choisi, la longueur moyenne avant est située autour de 262 alors qu'un code contraire dont le PDC serait identique au PDC de l'original aurait obtenu 250. On remarque aussi que la distribution de la longueur avant est presque uniforme.

Certains codes possèdent un code contraire pour lequel le profil de distance de colonne est presque le même que celui du code original. Le code de Johannesson, K = 20, R = 1/2 (2451311 et 3546713) possède un PDC de (2 3 3 4 4 5 5 6 6 6 7 7 8 8 8 8 9 9 9 10) alors que le code contraire possède (2 3 3 4 4 5 5 6 6 6 6 7 7 7 7 8 8 9 9 9). Il est possible de construire des codes ayant un code contraire avec un PDC identique au code original. Il suffit soit de concaténer un code de longueur de contrainte inférieure et son contraire ou de choisir le deuxième vecteur comme le contraire du premier. Le problème avec ces approches est que la distance libre $d_{free}$ des codes obtenus n'est pas aussi bonne que celle des codes optimaux.

Les problèmes encourus par le décodage M-chemins ne sont pas insurmontables. En alliant le décodage M-chemins avec une technique de décodage bidirectionnel intelligente, un important gain de codage est obtenu par rapport à l'algorithme de Viterbi (1 dB pour $P_B = 10^{-5}$) lorsque comparé au même niveau de complexité (64 chemins explorés).

# CHAPITRE 6
# REALISATION ITGE

Tel que décrit au chapitre 5, les algorithmes multi-chemins ont pour but de combiner les avantages du décodage de Viterbi et du décodage séquentiel sans pour autant présenter les désavantages de ces derniers. Le choix de l'algorithme M-chemins est basé sur les possibilités de l'algorithme tant du point de vue probabilité d'erreur que de celui de réalisation matérielle.

Plusieurs suggestions en vue d'une réalisation ITGE (intégration à très grande échelle) de l'algorithme M-chemins furent avancées [39],[40]. Malheureusement ces architectures n'offrent pas toujours la possibilité d'un grand nombre de processeurs ou d'une vitesse de décodage élevée.

Ce chapitre étudie une architecture multiprocesseur propice à la réalisation ITGE d'un décodeur M-chemins. La première section discute des avantages de l'algorithme au niveau régularité et simplicité ainsi que les principales difficultés de réalisation. La section 6.2 exposera les cellules de base d'une réalisation ITGE de cette machine. Enfin la section 6.3 traitera des implications du décodage bidirectionnel sur la structure présentée à la section 6.2.

## 6.1 Introduction

Bien que la plupart des algorithmes multi-chemins aient les mêmes avantages que le M-chemins du point de vue performance d'erreur, c'est au niveau réalisation que le M-chemins tire sa force. Le M-chemins est un algorithme répétitif dans lequel il ne reste aucune place pour la variabilité et ce, que cette dernière prenne la forme du nombre de chemins étendus ou de retours en arrière du décodeur.

L'algorithme M-chemins de base (section 5.1), fait l'extension simultanée des M chemins, ordonne les métriques en ordre croissant et ne conserve que les M meilleurs. Ce cycle est répété jusqu'à ce que la trame soit terminée.

Les autres algorithmes multi-chemins n'ont pas tous cette régularité. Certains requièrent un montant variable de mémoire et n'ont pas une vitesse de bits décodés constante. D'autres nécessitent un contrôle beaucoup plus élaboré.

Une machine aux opérations si régulières peut facilement profiter d'une approche parallèle pour calculer tous les chemins simultanément. Ce niveau de parallélisme assure une bonne vitesse de décodage, une simplification du contrôle général et facilite une optimisation des parties communes.

L'approche parallèle combinée avec un besoin constant de ressources mémoire rend la réalisation ITGE de l'algorithme M-chemins avantageuse. Cependant, pour tirer avantage des performances rendues possibles par une réalisation ITGE de l'algorithme, ce dernier doit être restreint à un nombre minimum de puces.

Pour rendre l'algorithme parallèle, l'extension des chemins en leurs fils et le tri des chemins par ordre de métrique peut être séparé en deux blocs indépendants. Le premier bloc est chargé de faire les extensions et le second le tri. La figure 6.1 illustre cette séparation des tâches.

Le bloc extenseur peut ensuite être subdivisé. Chaque chemin étant indépendant, son extension en ses deux fils est donc elle aussi indépendante. Par conséquent l'extension peut être réalisée de façon parallèle. Le bloc extenseur serait alors composé de M extenseurs simples. Chacune des cellules serait responsable de calculer les deux fils d'un seul père.

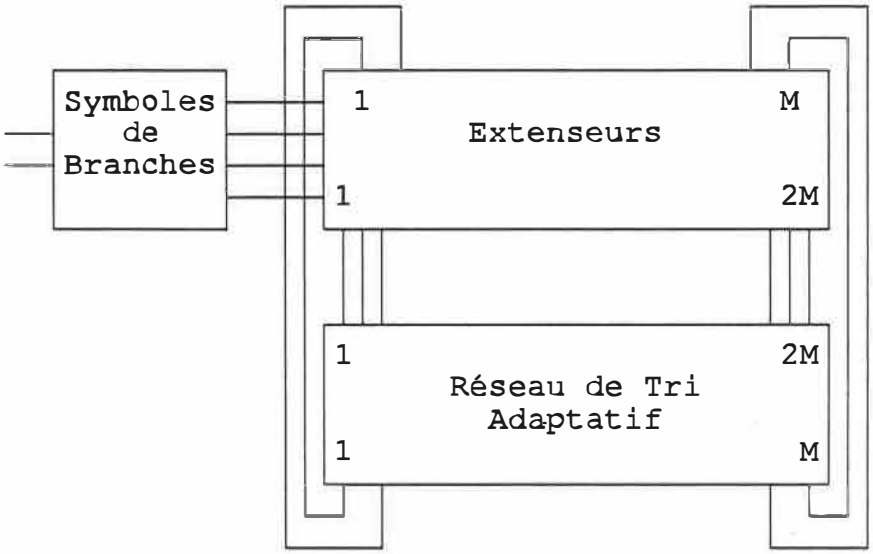


Figure 6.1 Schéma Général d'un décodeur M-chemins.

Le bloc de tri peut lui aussi bénéficier d'une structure parallèle. Compte tenu qu'un certain nombre de comparaisons sont indépendantes, elles peuvent alors être effectuées simultanément. Le bloc de tri serait alors divisé en une mosaïque de comparateurs. La figure 6.2 illustre ce principe.

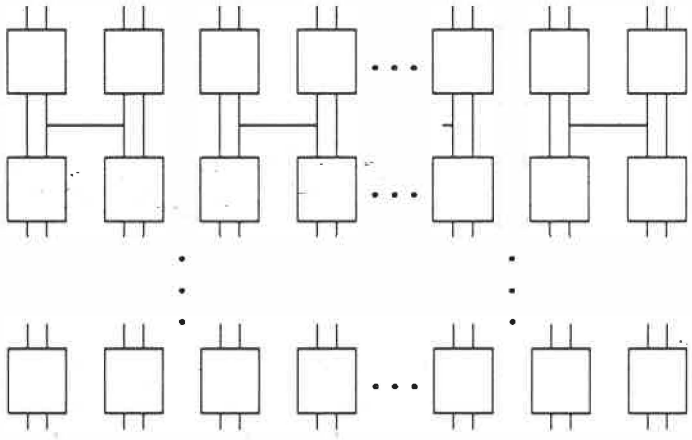

Figure 6.2 Réseau de tri adaptatif. Le réseau utilise M comparateurs en largeur et 1+lg2M en profondeur. La longueur des interconnexions augmente avec la profondeur.

Cependant un certain nombre de difficultés font surface. Les principales difficultés de réalisation de l'algorithme sont le nombre de connexions à l'intérieur comme à l'extérieur des extenseurs ainsi que le tri des noeuds par ordre de métrique.

Le nombre de bits d'information requis par calcul par extenseur impose un grand nombre de connexions. L'extension d'un noeud nécessite la connaissance de l'état du noeud (v = K-1), de la métrique accumulée (15 bits), des symboles reçus (6 bits), ainsi que des métriques de branches (9 bits). Par contre, si tous les extenseurs sont à la même profondeur, ils utilisent tous les mêmes symboles reçus et un sous-ensemble des métriques de branches peut être calculé. C'est ce qui est représenté par le bloc 'symboles de branche' dans la figure 6.1.

Pour M extenseurs, on retrouve environ M(K+29) liens de communications. Pour de bonnes performances, M se situe aux environs de 64 à 128 alors que K est entre 20 et 40. Il est évident qu'une puce possédant 128 bus de 59 bits est présentement irréalisable même dans la plus dense des technologies. Pour une technologie à 1 µm, le placement de ces fils dans un espace minimum occuperait 1 cm de coté.

Effectuer un tri de M éléments de façon parallèle requiert également un nombre de comparateurs beaucoup trop grand, surtout si chacun de ces comparateurs doit effectuer sa comparaison sur tous les bits de la métrique simultanément (15 bits). Un comparateur parallèle relativement lent constitué de 15 comparateurs simples chaînés successivement, utilisé dans un réseau optimum ne requérant que Mlg2M comparateurs nécessiterait au minimum 15 360 comparateurs simples si M = 128 . Un comparateur simple comprend au moins un OU-EXCLUSIF (XOR) et occupe quand même environ 10 000 $\mu m^2$ (100 µm X 100 µm (ref. Annexe A)), ce qui donne une superficie minimale de 1.5 $cm^2$.

## 6.2 Cellules de Base

A la lecture des problèmes mentionnés à la section précédente, il est évident que la mise en oeuvre d'une réalisation intégrée nécessite une réduction du nombre d'interconnexions ainsi qu'une solution astucieuse au problème du tri.

### 6.2.1 Interconnexions

La surface requise par le nombre de connexions demandées augmente rapidement. Si on disposait d'une puce de taille suffisante pour une solution parallèle, elle devrait utiliser des techniques de tolérance aux défaillances. Par contre la complexité et la non-régularité d'un réseau de tri rendent ces techniques moins efficaces. La seule option qui semble pratique aujourd'hui est de réduire la complexité des interconnexions par au moins un ordre de grandeur.

Une approche sérielle aux communications est la seule façon de réduire le nombre de conducteurs d'un ordre de grandeur. Cette technique offre également les avantages d'une simplification et d'une réduction de la surface requise par les éléments de calcul eux-même. Un élément de calcul sériel n'occupe qu'une unité de surface comparativement à une unité de traitement parallèle de N bits qui en occupe au moins N.

Cependant une vitesse moindre est normalement associée à une technique de communications sérielles. Dans un système entièrement sériel, les calculs ne peuvent pas être effectués plus rapidement que le nombre de bits dans le calcul. Si on considère un système ayant 15 bits de métrique (met = 15), et 20 bits d'état (état = 20) et qu'il effectue un calcul par unité de temps, le calcul de la parité sur l'état ainsi que l'addition des métriques requérent S = met+état soit 35 unités de temps. On suppose ici que les liens de communications globaux sont des signaux de contrôle, et que par conséquent ceux-ci sont négligés dans le calcul.

Un système utilisant des communications parallèles et une architecture en arbre pour les opérations de parité et d'addition nécessite moins de temps de calcul. La structure en arbre donne une borne inférieure sur le temps de calcul, soit lgN unités de temps pour N bits. Un système utilisant met = 15 et état = 20 demanderait respectivement un minimum de 4 et 5 unités de temps pour le calcul de la métrique et de la parité.

La perte de performance est donc le rapport entre la vitesse du système sériel par rapport au système parallèle. Si on suppose qu'il est possible d'effectuer les calculs à la vitesse maximale permise par les seuls délais dans la logique combinatoire, alors pour met = 15 et état = 20, ce ralentissement n'est que de (15+20)/(4+5) = 3.89. Une machine parallèle, utilisant une structure en arbre pour un minimum de délai, ne serait donc que 3.9 fois plus rapide. Cette perte de vitesse n'est pas déterminante si l'on considère que l'on a obtenu une réduction de la surface et du nombre de connexions d'au moins S fois (S = 35).

Il est possible d'envisager une gamme de solutions combinant communications sérielles et parallèles. L'étude de l'ensemble de ces solutions dépase malheureusement le cadre de ce travail.

### 6.2.2 Réseau de Tri

Un réseau de tri adaptatif (RTA ou Self Sorting Network SSN) [41] est un réseau dont les connexions sont fixes mais dans lequel les mouvements des données sont contrôlés par les données elles-mêmes (figure 6.2). Ce réseau est composé d'éléments de tri (figure 6.3), chacun possédant deux entrées et deux sorties. Une sortie est définie comme le maximum des deux entrées alors que l'autre est le minimum des entrées. Les communications étant sérielles, les données passent dans les éléments de tri selon l'ordre MSB (Most significant bit) en premier.

Chacun de ces éléments (figure 6.3) possède donc un comparateur (Comp), un échangeur, et deux mémoires d'un bit (REG). La fonction du comparateur est de détecter lorsque les entrées $E_A$ et $E_B$ ne seront plus équivalentes et de fixer la position de l'échangeur pour que le maximum des-deux entrées se retrouve à la sortie indiquée $S_{Max}$. Les mémoires d'un bit sont utilisées pour permettre une structure pipeline du réseau de tri et pour laisser le temps au comparateur d'arriver à une décision.

En utilisant une technique de communications sérielles les comparateurs sont réduits à des unités d'un seul bit. La surface requise par ces derniers, en comptant toujours 10 000 $\mu m^2$ (100 $\mu$m X 100 $\mu$m) par élément et Mlg2M éléments, devient 0.1 $cm^2$. Dorénavant le terme 'comparateur' fera référence à la paire comparateur/échangeur.

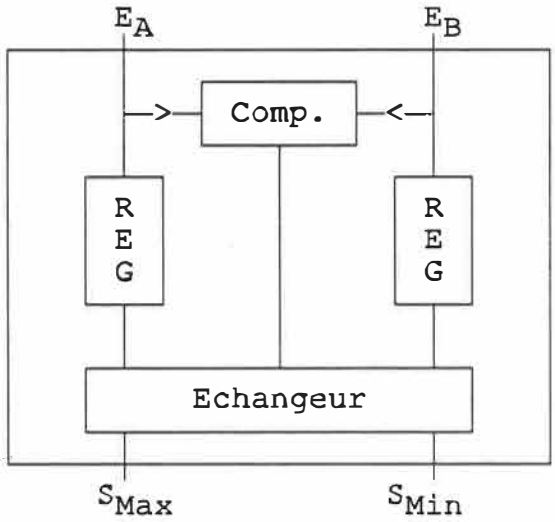


Figure 6.3 Représentation d'un élément de tri.

La cellule de tri représenté à la figure 6.3, bien que relativement simple, demande de 30 à 40 transistors selon l'optimisation désirée. Une économie d'un transistor est multipliée par au moins Mlg2M. Il est donc important que cette cellule soit petite. Le délai dans un réseau de tri adaptatif est le délai d'un élément multiplié par le nombre d'étages du réseau (1+lg2M).

Comme dans le cas de la pile systolique, (chapitre 4) le tri n'a pas besoin d'être complet. Le réseau de tri n'a besoin que de grouper ensemble les M meilleurs parmi 2M entrées sans se préoccuper de l'ordre. Seule la position du meilleur chemin doit être connue.

Il est possible de simplifier encore davantage le problème du tri en stipulant que même les M meilleurs chemins ne sont pas absolument requis mais qu'un ensemble dont la majorité des éléments font partie des M meilleurs est suffisant. Si les éléments choisis ne font pas tous partis de l'ensemble des M meilleurs éléments, les éléments manquant doivent être parmi les moins bons éléments de l'ensemble parfait alors que les éléments choisis ne faisant pas partie de l'ensemble parfait doivent être parmi les meilleurs des éléments normalement rejetés.

La liste 6.1 établit les principaux critères de performance du réseau de tri. Ces critères sont des critères minimaux. Un réseau de tri optimisé pour rencontrer seulement ces critères réduit de façon considérable la complexité de la tâche à effectuer.

1- Le nombre d'étages du réseau doit être de l'ordre de O(lg2M).
2- La position du meilleur item doit être connue.
3- Si le tri n'est pas parfait, les meilleurs items doivent toujours faire partie de la sélection et les pires jamais.

Liste 6.1 Conditions minimums du réseau de tri.

Un certain nombre de facteurs doivent être considérés lors de la réalisation d'un tel réseau. Premièrement, le nombre de comparateurs, donc le délai, devrait être constant sur tous les chemins. Deuxièmement la longueur des interconnexions doit être réduite au minimum tout en conservant la régularité. La longueur des interconnexions devient de plus en plus critique avec l'accroissement de la vitesse de l'horloge.

Des méthodes de tri efficaces existent pour les grands réseaux $N \geq 16$ [31], cependant aucune n'obtient un tri en n'utilisant que lg2M étages. Les critères énoncés

plus haut sont donc incompatibles avec les réseaux de tri conventionnels. En particulier, le fait de limiter le nombre d'étages à lg2M a un impact considérable sur la complexité de la solution.

Une technique classique utilisée pour choisir les M meilleurs chemins parmi 2M consiste à trier de façon indépendante deux ensembles de M items $X_0...X_m$ et $Y_0...Y_m$. et de comparer et échanger $X_0,Y_m$ ... $X_m,Y_0$. Cette approche due à V. E. Alekseyev [31] garantit que les M meilleurs éléments se retrouveront dans les positions $X_0...X_m$.

Les bornes inférieures [31] pour le nombre de comparateurs ainsi que pour le délai existent pour le problème de la sélection. La borne inférieure pour le nombre de comparateurs est Mlg(M+1) et celle du délai est: $lg2M + \lceil lgM \rceil lg\ lg2M + O(log\ log\ log2M)$ lorsque $M \rightarrow \infty$. Si M = 64 ces bornes indiquent que le meilleur réseau de tri possible est composé de 384 comparateurs répartis sur 18 étages. Malgré que le nombre de comparateur soit très raisonnable, les 18 étages introduisent un trop grand délai dans chaque cycle de décodage.

Comme les bornes inférieures de complexité pour une sélection parfaite ne rencontrent pas nos objectifs, une sélection imparfaite doit être étudiée. Il faut remarquer que même si le réseau de tri est imparfait, on peut compter sur l'aide du comportement du décodeur lui-même.

Lors du décodage, le meilleur chemin a souvent tendance à donner naissance à deux noeuds se retrouvant parmi les M meilleurs, il en va de même pour les autres chemins. On constate que les noeuds avant d'être réordonnés ne représentent pas une séquence aléatoire mais respectent un ordre établi. Il est possible de biaiser la fonction de tri de façon à favoriser les noeuds se trouvant dans les M meilleurs emplacements.

Une étude plus approfondie des configurations de réseaux de tri est présentée en annexe A. Le problème de la sélection optimale n'est pas encore résolu.

### 6.2.3 Extenseurs

Les extenseurs sont les éléments de calcul responsables de la génération des noeuds à partir de leurs pères. Chaque extenseur calcule les deux fils d'un noeud. Cette opération est constituée du calcul de la parité de l'état et de l'addition des métriques de branches appropriées (une pour chaque fils) à la métrique accumulée (celle du père).

Chaque extenseur reçoit comme information:

1- L'état du noeud à étendre
2- La métrique accumulée
3- Le code
4- L'ensemble des possibilités de métriques de branche.

Liste 6.2 Informations requises par chaque extenseur.

Etant donné que tous les extenseurs se trouvent à la même profondeur, ils utilisent tous les mêmes symboles. Un pré-traitement peut être effectué pour réduire le nombre de possibilités de métriques de branche. C'est ce sous-ensemble, composé de 4 possibilités, qui est communiqué à tous les extenseurs. Le code est également le même pour tous les extenseurs.

L'état de chaque noeud ainsi que la métrique accumulée sont circulés dans le système. Cette information est fournie à l'extenseur illustré à la figure 6.4 par son entrée 'Père'. Chaque extenseur développe la branche 0 et la branche 1 de son père.

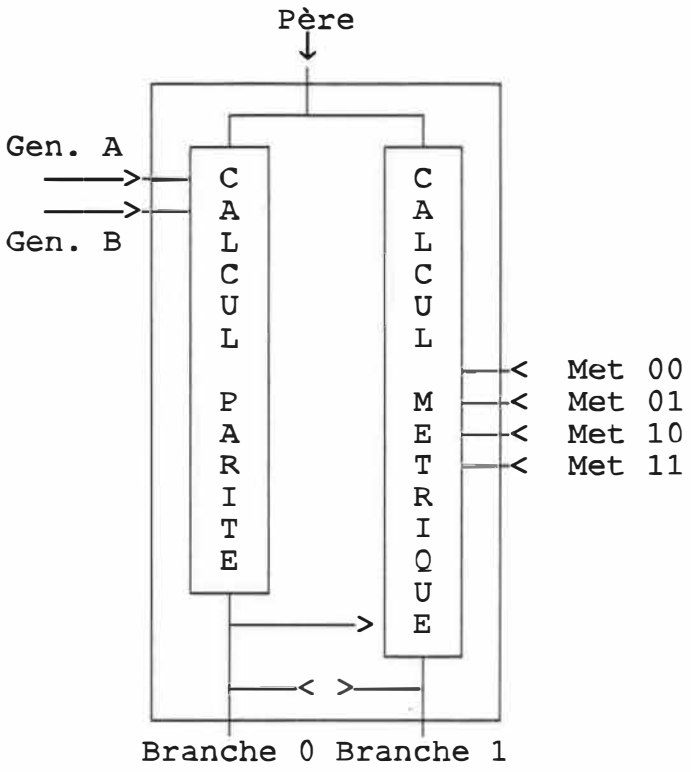


Figure 6.4 Représentation de l'extenseur.

Lorsque les communications sont effectuées de façon sérielle, le calcul de la parité peut être effectué par une porte OU-EXCLUSIF (XOR) avec une entrée en rétro-action et une porte ET (AND) pour tenir compte du code. Le calcul de la métrique est effectué par un additionneur d'un bit avec la retenue en rétro-action.

Les communications sérielles font cependant apparaître deux problèmes. Ces problèmes concernent l'ordre des bits dans les communications. A la sous-section précédente, les bits de la métrique devaient précéder ceux de l'état. De plus ils devaient être présentés MSB en premier. Sans ces conditions, il est impossible d'effectuer un tri de façon correcte et efficace.

Cependant, l'opération d'addition demande les bits de la métrique dans l'ordre inverse soit LSB (Least Significant Bit) en premier. De plus les extenseurs basent la

sélection de la métrique de branche à additionner sur le résultat de l'opération de parité effectuée avec l'état. L'extenseur doit donc recevoir les bits de l'état d'abord puis ceux de la métrique dans l'ordre LSB. Rappelons que le réseau de tri base sa décision sur la métrique (ordre MSB) et fait suivre l'état.

Même en admettant que les bits de la métrique soient dans le bon ordre, il est impossible de commencer à additionner les métriques de branches aux métriques accumulées dès que ces dernières sortent du réseaux de tri, car les métriques de branches ne sont définies qu'après l'opération parité sur l'état alors que ce dernier n'est pas encore passé dans le réseau de tri. Il est aussi impossible de passer les états en premier dans le réseaux de tri car les positions des aiguilleurs sont définies par les valeurs des métriques.

Ces contradictions empêchent les deux blocs (extenseur et RTA) de former un grand pipeline. Chaque bloc doit donc terminer ses opérations avant que l'autre puisse commencer son traitement. Un temps mort doit être placé dans le système. Ce temps mort est un phénomène normal dans les systèmes pipelines, où l'on doit vider partiellement le pipeline. Ce temps mort ne peut être éliminé.

Bien qu'il soit impossible de connaître la réponse de l'addition des métriques de branche avant que le calcul de l'état ne soit fini, on peut cependant terminer le calcul des réponses possibles avant de connaître la parité de l'état. On calcul toutes les valeurs possibles de la métrique accumulée, 4 possibilités, puis une fois le résultat de l'opération parité connu, les bonnes métriques sont choisies. Cette technique, surtout utilisée dans les additionneurs rapides, est inspirée du "carry select" et elle minimise la durée du temps mort inséré dans le système.

Afin de présenter la métrique dans le bon ordre aux additionneurs, des tampons de style dernier arrivé, premier sorti (LIFO) sont utilisés. Le premier de ces

tampons inverse la métrique devant les additionneurs alors que les autres effectuent la même opération après ces derniers.

Ces nouveaux raffinements sont illustrés à la figure 6.5. Ces solutions ont deux implications. Ainsi conçu, l'extenseur utilise beaucoup plus de transistors qu'une version où il n'y aurait pas de problème d'ordre des bits. Deuxièmement, la dégradation de performance due à la latence de chargement du réseau de tri est le rapport entre le nombre d'étages du réseau de tri et le nombre total de bits ayant à circuler dans la pipe. Pour état = 20, met = 15, M = 128 et un tri imparfait nécessitant lg2M étages, la réduction de performance est lg(256)/(20+15) soit légèrement plus de 22%. Le mouvement détaillé de l'information est décrit en annexe A.

On peut voir à la figure 6.5 les quatre additionneurs (ADD) recevant chacun une des quatre valeurs possibles des métriques de branche ainsi que l'unité de sélection 4 à 2 requise par l'approche "carry select". De façon stricte, seulement deux tampons LIFO sont requis après les additionneurs, ils devraient alors être placés après l'unité de sélection 4 à 2 et ainsi augmenter le temps mort. Le tampon premier arrivé premier sorti (FIFO, First In, First Out) additionnel est requis pour préserver l'état. Les multiplexeurs (MUX) en sortie sont utilisés pour envoyer à tour de rôle la métrique et l'état dans le réseau de tri.

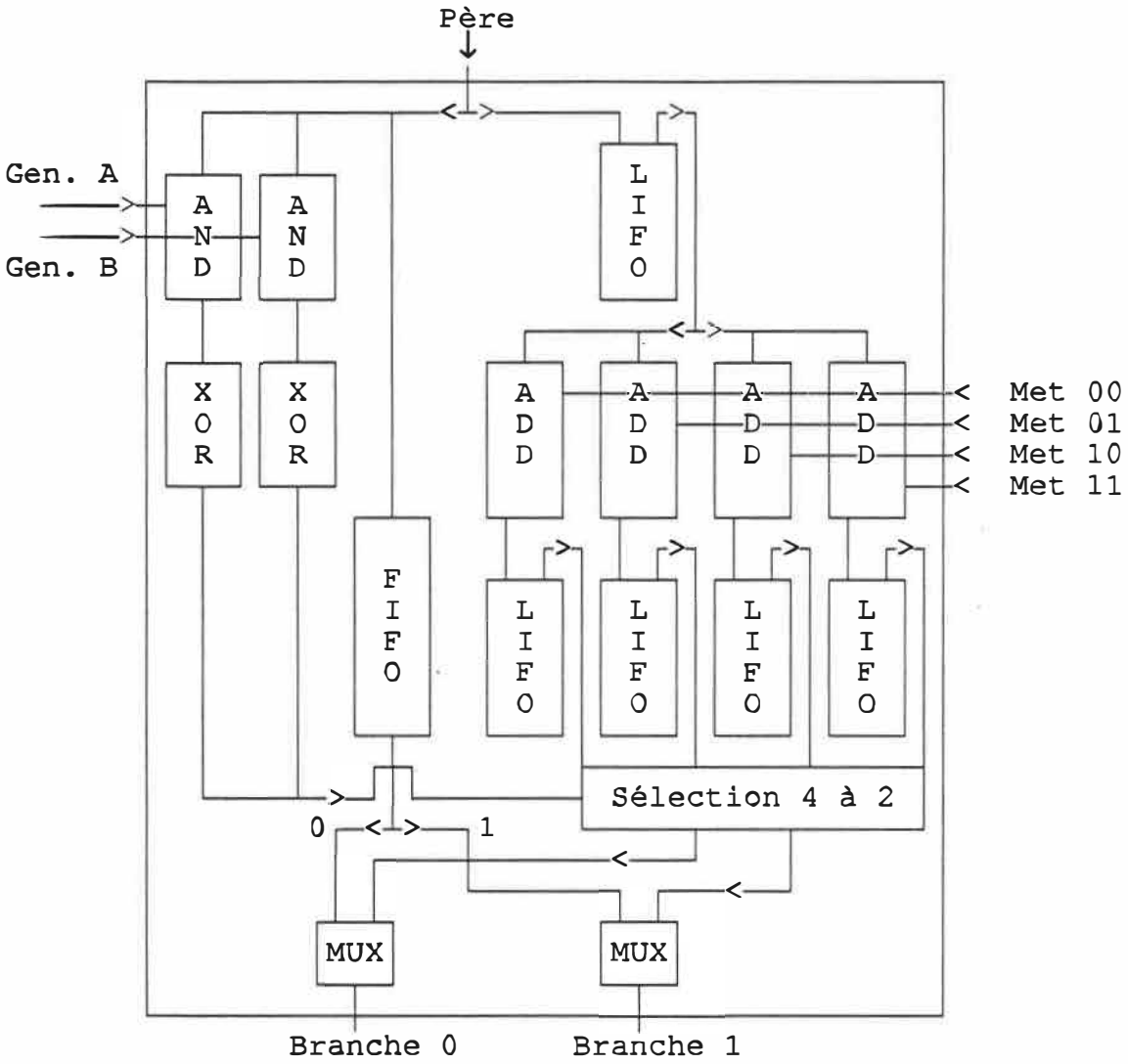


Figure 6.5 Représentation détaillée de l'extenseur.

Malgré l'emploi de communications sérielles, la cellule extenseur demeure assez complexe. Celle-ci n'est répétée que M fois mais chaque copie de cette dernière nécessite jusqu'à 115 cellules de mémoire sous forme de registres à décalage, 4 additionneurs, et une dizaine de portes élémentaires. Chaque extenseur devrait donc utiliser près de mille transistors.

### 6.2.4 Autres Considérations

Une puce contenant les éléments de calcul mentionnés précédemment n'aurait besoin que de 7 broches, 2 pour la masse et l'alimentation, une horloge, une chaîne de chargement des données, une pour l'entrée des symboles, une pour l'information décodée et la dernière comme "reset". En réalité il est préférable que toutes les sorties des M chemins soient disponibles à l'extérieur. De plus, une puce dans laquelle tous les chemins seraient disponibles à l'extérieur pourrait être utilisée dans d'autres variantes de l'algorithme de décodage M-chemins, par exemple les algorithmes bidirectionnels.

Un système versatile utiliserait donc les broches de cette façon:

| Fonction | entrée/sortie | # broches |
|---|---|---|
| M chemins | sortie | M |
| Horloge système | sortie | 2 |
| Horloge haute vitesse | entrée | 4 |
| Symboles du canal | entrée | 6 |
| Masse et alimentation | entrée | 2M/7 |
| Chaîne de chargement | entrée | 1 |
| Test | entrée | 1 |
| Init/Fonctionne | entrée | 1 |
| Sélection de la puce | entrée | 1 |
| Total | | 16+9M/7 |

Tableau 6.1 Répartition des broches pour un décodeur M-Chemins

Les problèmes majeurs de ce circuit sont la propagation et la génération de l'horloge extrêmement rapide requise. Si l'opération la plus lente n'exige que 5 ns, il est alors possible d'utiliser une horloge à deux phases interne de 100 Mhz avec période active (duty cycle) de 50% (5 ns). Un cycle de décodage requiert 20+15+lg2M cycles d'horloge interne. Un système propageant 128 chemins décode alors à plus de 2.3 Mbits/sec. Un système utilisant une période active (duty cycle) plus appropriée à une architecture à registre à décalage serait capable d'une fréquence d'horloge plus élevée et par conséquence d'une performance supérieure.

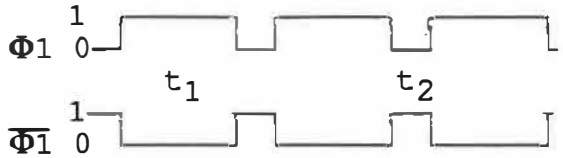


Figure 6.6 Représentation d'une horloge à deux phases utilisant une phase active de 75% (150 MHz, $t_1$ = 5 ns = opération, $t_2$ = 1.7 ns = transfert).

Bien que quasiment impossible à générer de l'extérieur à cause de la bande passante limitée des broches, il est possible d'utiliser deux séquences à demi-vitesse et de les combiner au moyen d'une porte XOR. Cette technique peut être cascadée autant de fois que nécessaire (en supposant que le temps d'arrivé des transitions est bien défini).

Le problème de la propagation interne des horloges peut être amoindri en réduisant la charge de chacune des lignes à un minimum. Une horloge à deux phases, combinée avec l'utilisation de demi-portes de transmission contribuerait sensiblement à ce but, surtout si on considère que cette puce est principalement composée de registres à décalage. Bien que l'inconvénient de cette technique soit la perte de vitesse, les inverseurs suivant les portes de transmission peuvent être spécialement équilibrés afin de réduire le problème. Encore une fois, les deux phases des horloges peuvent être générées sur la puce afin d'éviter l'inductance des broches.

Si les horloges sont difficiles à propager, les informations globales comme le code et le sous-ensemble de métriques de branche ne le sont pas moins. Ces données doivent elles aussi arriver à haute vitesse dans les extenseurs. Une dérive dans la propagation de ces données serait également désastreuse.

Le bloc symboles de branche de la figure 6.1 contient les vecteurs générateurs du code, une table de métriques de branches, quatre additionneurs pour calculer les quatre possibilités transmises aux extenseurs et une logique de sélection des métriques de branche dérivées des symboles reçus du canal. Une quantification de 3 bits est possible

pour les symboles reçus du canal. Une représentation détaillée de ce bloc est illustrée à la figure 6.7.

La logique utilisée pour calculer les sous-ensembles de métriques de branche n'est requise qu'une seule fois pour les M extenseurs et elle est la même que celle utilisée dans une puce extenseur parallèle réalisée à l'Ecole Polytechnique de Montréal [42]. De légères modifications doivent être apportées étant donné la nature sérielle des communications versus l'approche parallèle de la puce extenseur mais la logique de contrôle demeure essentiellement la même.

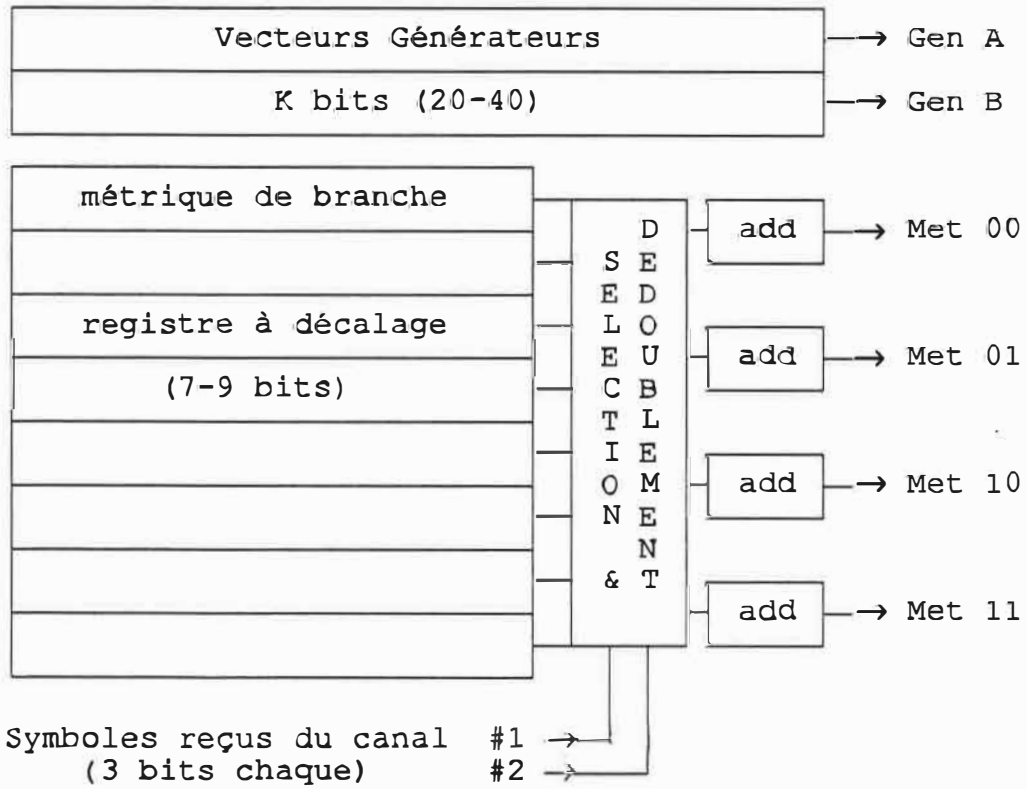


Figure 6.7 Représentation détaillée des éléments de calcul communs à tous les extenseurs. Calcul des métriques de branche et des vecteurs générateurs.

Le nombre de transistors requis pour les fonctions du bloc 'symboles de branches' n'est pas tellement élevé: soit environ 152 bits de mémoire sous forme de

registres à décalage, 4 additionneurs d'un bit et environ une trentaine de portes simples. Le total ne devrait pas dépasser 2000 transistors, ce qui est relativement négligeable.

Une unité de perforation peut être ajoutée afin de permettre le décodage de codes de taux élevés et ainsi augmenter la flexibilité de la puce. L'unité de perforation n'affectant que les valeurs des métriques de branches, sa réalisation est facile et ne requiert pas beaucoup de transistors (environ 500 [42]).

Le tableau 6.2 estime le nombre de transistors requis pour la fabrication de la puce. Ces estimés ne tiennent compte que des cellules de base. Les transistors utilisés pour les horloges et les autres ajouts ne sont pas comptés. Cependant nous estimons qu'il serait possible de réaliser ces puces sur des surfaces de 0,5 et de 1,0 $cm^2$ dans une technologie de 1,2 µm en admettant que les problèmes associés à la propagation des horloges aient été résolus.

| Fonction | Nombre de répétitions | Nombre de transistors | M = 64 | 128 |
|---|---|---|---|---|
| Symboles | 1 | 2000 | 2000 | 2000 |
| Ext | M | 1000 | 64000 | 128000 |
| RTA | M(1+lg(2M)) | 35 | 18000 | 40500 |
| Total | | | 84000 | 170500 |

Tableau 6.2 Nombre de transistors requis pour un décodeur M-chemins (approximations).

## 6.3 Système Bidirectionnel

La réalisation d'un décodeur bidirectionnel (chapitre 5) implique deux complications additionnelles par rapport au système unidirectionnel décrit à la section 6.2. Premièrement, les décodeurs avant et arrière doivent gérer conjointement un algorithme de jonction des chemins. Comme démontré au chapitre 5, la jonction des

chemins permet la détection d'erreurs et un algorithme 'intelligent' permet un léger gain de codage.

En deuxième lieu, si l'algorithme bidirectionnel utilise des longueurs variables, le décodeur doit décider quelle direction est la moins susceptible d'avoir perdu le chemin correct. Un décodeur bidirectionnel variable offre un gain de codage substantiel par rapport à n'importe quel système et l'alternative doit être considérée.

### 6.3.1 Configurations Bidirectionnelles

Les architectures à longueurs constantes seront d'abord étudiées.

Utilisant les blocs définis précédemment (section 6.2), trois structures sont possibles pour la réalisation d'un décodeur bidirectionnel. Premièrement, une puce peut conserver deux ensembles de chemins et les passer à tour de rôle dans les modules extenseur et tri. Un ensemble est composé des M chemins et de leurs métriques et états, c'est-à-dire un bloc de M bits de large par 35 bits de profond.

Les coûts pour réaliser une telle approche sont un léger accroissement de la complexité de la structure de contrôle et un investissement en mémoire assez considérable (minimum de 35M). Par contre le matériel requis par la technique "carry select" de même que le temps mort peuvent être éliminés au prix d'un léger allongement du FIFO de l'état (figure 6.5). Les extensions dans une direction n'étant pas successives, il est possible d'utiliser le temps d'extension de l'ensemble avant pour réaliser l'inversion des bits de l'ensemble arrière requis par les extenseurs. En faisant précéder la métrique par l'état dans les extenseurs, le matériel requis par la technique "carry select" ainsi que le temps mort nécessaire pour charger le réseau de tri sont éliminés. Ce système demeure régulier et il est légèrement plus rapide que la version unidirectionnelle.

La deuxième approche consiste à dédoubler tous les composants à l'exception de la logique de contrôle général. Cette approche permet un autre gain de vitesse mais les

temps morts sont encore présents. Cette approche demande également le double du nombre de broches et n'est probablement pas la meilleure solution.

La dernière façon de réaliser un système bidirectionnel consiste à placer deux puces unidirectionnelles côte-à-côte. Cette solution implique évidement une duplication de tous les composants mais elle permet de réduire la complexité des puces ou d'augmenter le nombre de chemins traités, ce qui améliore la qualité de décodage.

### 6.3.2 Jonction des Chemins

Une puce capable de décoder de façon bidirectionnelle se doit de posséder un algorithme de jonction des chemins qui, comme son nom l'indique, sert à fusionner les chemins avant et arrière lorsque les décodeurs se rencontrent. Tel que traité à la section 5.2, l'algorithme de jonction choisi n'est utilisé qu'une seule fois par trame et doit comparer tous les états des deux décodeurs. Si une paire correspond parfaitement alors ces deux chemins sont choisis, autrement le chemin de métrique supérieure de chaque direction est choisi. La direction offrant le meilleur écart entre ses métriques maximale et minimale devrait être utilisée pour fournir les bits appartenant à la longueur de contrainte commune. Les chemins respectifs sont ensuite retracés.

Etant donné le manque de régularité de l'algorithme de jonction, de ses nombreuses variantes et l'application de celui-ci seulement une fois par trame, une approche hors puce est suggérée. L'approche hors puce permet d'augmenter le parallélisme et réduit la complexité du contrôle de la puce. Une approche extérieure ne nécessite pas une réalisation matérielle dédiée. Un microprocesseur pourrait accomplir la tâche. Etant donné qu'on n'opère la jonction qu'une seule fois par trame et il est possible de maintenir le même débit de traitement en l'effectuant en pipeline avec le décodage proprement dit.

La comparaison des états peut être facilitée par le réseau de tri. Dans une procédure de terminaison, la puce passerait l'état en premier dans le réseau de tri. Les deux groupes d'états, avant et arrière, étant ainsi ordonnés (partiellement ordonnés), trouver deux états identiques devient plus facile. Cet aspect ne sera pas développé d'avantage. Le retracement des chemins quant à lui est fonction de la mémoire du décodeur et est indépendant de la méthode de jonction.

En plus d'avoir l'avantage de permettre l'étude d'autres algorithmes de jonction, le microprocesseur peut servir à d'autres tâches comme la communication avec le système hôte, le test et le diagnostic du système, l'assemblage des trames décodées ainsi qu'à l'initialisation des puces. Comme on le verra à la sous-section suivante, il est également possible que le microprocesseur aide à la décision de la direction à poursuivre dans un système à longueurs variables.

De toute façon, l'utilisation d'une approche hors puce est presque inévitable étant donné les choix de design de la puce décrite précédemment. En effet, il est plus facile de placer deux puces unidirectionnelles dans un système bidirectionnel et d'utiliser un microprocesseur pour faire la jonction que de combiner les deux décodeurs sur un même morceau de silicium et d'incorporer en plus un algorithme de jonction. Le seul avantage du système à une puce serait justement une réalisation sur une seule puce, donc un système moins complexe.

### 6.3.3 Sélection de la Direction de Décodage

Si des longueurs variables sont utilisées lors du décodage bidirectionnel, alors le choix du décodeur devant attendre et de celui poursuivant l'exploration doit être fait à chaque cycle. Cette importante opération est insérée dans le chemin critique. Nous rappelons ici qu'au chapitre 5 le critère de décision utilisé consistait à comparer la

différence des valeurs de métrique du chemin maximum et du chemin minimum. Le plus grand écart de métrique déterminant donc la direction à poursuivre.

Ce critère de sélection est relativement facile à réaliser, car le réseau de tri donne exactement la position du meilleur élément (tableau 6.1) et par symétrie la position du pire élément. Le calcul du critère de décision implique seûlement la soustraction des métriques et la comparaison avec le résultat de l'autre direction.

Le système unidirectionnel décrit à la section 6.2 commence l'inversion et le calcul des métriques accumulées dès que ces dernières sortent du réseau de tri; ce ne peut être le cas si l'on insère une décision supplémentaire. Le système doit attendre de savoir de quel ensemble l'extension doit provenir avant de pouvoir commencer celle-ci.

Bien que le critère de décision n'implique en réalité qu'une comparaison, le temps requis pour effectuer cette dernière est inséré dans le chemin critique. Sous peine d'introduire un autre délai dans le cycle de décodage, on doit retirer ce temps du chemin critique.

La technique de "carry select" peut encore être utilisée pour cacher le temps de calcul requis par la comparaison. Toutefois, cette approche signifie un dédoublement de presque toutes les fonctions de la puce. Bien que les unités de calculs soient des unités sérielles, un tel dédoublement entraîne un gaspillage énorme. De plus, le rendement et la complexité de la puce se détériorent.

Une alternative envisagée est l'emprunt d'une technique utilisée dans les processeurs rapides, soit une décision retardée "delayed branching". Le processeur effectue toujours l'opération suivant le branchement conditionnel que le branchement soit pris ou non. Cette technique suppose qu'une instruction utile aux deux alternatives sera placée derrière le branchement. Dans le cas présent, une extension peut être considérée comme une instruction et, dans la majorité des cas, l'extension de la branche suivante est

utile. La méthode est applicable sans modifications et parvient à retirer la comparaison du chemin critique.

Une réalisation de cet algorithme permet d'éliminer le dédoublement de l'unité de calcul réduisant ainsi le nombre de transistors au prix d'une légère augmentation de la complexité dans le contrôle.

Les versions à une seule puce décrites plus haut contiennent déjà tout le matériel nécessaire à la réalisation de cet algorithme. Il faut noter que le temps mort doit être réinséré dans la première version de la puce car la même direction peut avoir besoin de deux extensions de suite. Il s'agit donc de réorganiser les cellules de mémoires et de remettre la logique de "carry select" des additionneurs.

La version à deux puces du décodeur demande un nombre suffisant de sorties additionnelles, pour permettre aux données nécessaires à la prise de décision d'être transmises à l'extérieur. Une unité de comparaison choisirait la meilleure direction et générerait un signal 'arrêt/continue' pour chaque puce. Comme chaque cycle est environ 40 à 45 fois plus long que la fréquence d'horloge utilisée, le comparateur extérieur n'a pas besoin d'être extrêmement rapide et le signal de contrôle a tout le temps voulu pour arriver avant le début du cycle suivant.

## 6.4 Conclusions

Il à été démontré dans ce chapitre comment réaliser à partir d'une puce de base un système de décodage multi-chemins. La puce est suffisamment générale pour être utilisée dans un système bidirectionnel à longueur variable.

Les avantages et désavantages du décodage M-chemins ont d'abord été expliqués, puis une architecture utilisant des processeurs simples d'un bit allié avec un système de communications sérielles à été décrite. Cette architecture permet une bonne

vitesse de décodage et l'utilisation d'un parallélisme massif pour l'extension des chemins ainsi que pour la sélection des M meilleurs chemins.

Les blocs de base ainsi que les cellules composant ces blocs ont été décrits avec suffisamment de détail à la section 6.2 pour permettre une réalisation facile du système. Un schéma au niveau transistor ainsi que son chronogramme (timing diagram) pour la cellule de tri sont fournis en annexe A. On y trouve également une analyse de quelques configurations des réseaux de tri.

La dernière section de ce chapitre a couvert l'utilisation de la puce décrite auparavant dans un système bidirectionnel et des modifications possibles selon différents choix de design. Cependant, il est évident qu'avec les technologies d'intégration actuelles, la réalisation de ce système demande un investissement important de ressources humaines et il n'est probablement pas réaliste d'espérer qu'un prototype à grande vitesse soit réalisé dans le cadre d'un projet de maîtrise. Une version à échelle réduite utilisant moins de chemins (M = 4 à 16) et une horloge moins rapide demeure néanmoins réalisable avec des moyens modestes.

# CHAPITRE 7
# CONCLUSIONS

Le but principal de cette recherche visait à développer des algorithmes et architectures permettant la réalisation de puissants décodeurs pour codes convolutionnels. Ces décodeurs devaient être de complexité réduite mais sans pour autant sacrifier la puissance correctrice du code.

Pour ce faire, deux approches furent explorées. La première consiste en une réalisation matérielle d'un décodeur séquentiel à pile. La deuxième implique le développement d'un algorithme de recherche en largeur combiné avec une exploration bidirectionnelle de l'arbre d'encodage.

Une pile systolique fut réalisée [32]. A L'aide de celle-ci deux nouveaux algorithmes de tri, furent développés: soient l'algorithme systolique utilisant des tranches ordonnées et l'algorithme semi-systolique par insertions. Ces architectures furent comparées selon des critères de complexité, de temps d'exécution et de qualité de tri. Cette étude démontre que le gain d'efficacité des nouveaux algorithmes par rapport aux autres algorithmes originaux peut être moins substantiel que prévu et que l'application doit dicter le choix d'une architecture ou d'une autre.

La deuxième approche attaqua directement la variabilité de l'effort de calcul du décodeur séquentiel. Afin d'obtenir une réduction de la variabilité de l'effort de calcul du décodage séquentiel, un algorithme bidirectionnel allié à un algorithme sous-optimal de recherche en largeur fut developpé. Ce nouvel algorithme fut simulé et les résultats démontrent clairement qu'un gain de 1 dB est possible sur l'algorithme de Viterbi lorsque comparé à complexité équivalente pour une probabilité d'erreur de $10^{-5}$. Ce gain de

codage est obtenu par l'utilisation d'un code beaucoup plus puissant, malgré que la technique de décodage elle-même soit sous-optimale.

Il fut également démontré qu'une réalisation ITGE, Intégration à très grande échelle, est possible. Une machine parallèle équipée de processeurs binaires rapides fut décrite. Chaque unité de traitement sériel est responsable de l'extension d'un noeud produisant ainsi ses deux noeuds fils. Le réseau de tri adaptatif est, lui aussi, réalisé de façon sérielle. Une analyse d'espace et du nombre de transistors requis, basée sur des cellules prototypes, démontre qu'il est possible de réaliser un tel projet en utilisant les technologies disponibles aujourd'hui.

## 7.1 Recherches Futurs

Le décodage bidirectionnel ouvre une autre porte dans la recherche de codes optimaux. Un code optimal pour le décodage bidirectionnel doit posséder la meilleure distance libre $d_{free}$ possible ainsi que des fonctions de distance des colonnes semblables pour le code et son contraire.

Une recherche théorique est encore à faire quant à la métrique à utiliser pour ces décodeurs. En effet la métrique utilisée influence grandement la qualité du décodage. L'utilisation de la métrique de Viterbi donne une moins bonne probabilité d'erreur que l'utilisation de la métrique de Fano à cause de la longueur variable dans la jonction des chemins.

Liées à la métrique, des analyses visant à trouver le nombre de chemins nécessaires pour une certaine performance restent encore à effectuer. Ces analyses pourraient corroborer les résultats obtenus et être utilisées afin de trouver un optimum en terme de rapport coût/performance.

La présence de plusieurs salves d'erreurs par trame pose également certains problèmes. Il est alors possible que les décodeurs avant et arrière perdent simultanément le chemin correct. Une période de décodage érronée est alors inévitable.

L'influence de plusieurs autres variables sur les performances est encore à démontrer ainsi que l'élaboration de techniques alternatives en cas de non-aboutement. Dans le décodage bidirectionnel variable, bien que plusieurs solutions au problème du choix du décodeur à avancer furent proposées, la recherche d'une solution optimale à ce problème constitue un défi de taille.

# REFERENCES


[1] Bhargava, V.K., Haccoun, D., Matyas, R., et Nuspl, P., *Digital communications by satellite*, Wiley, New York, 1981.

[2] Viterbi, A.J. and Omura, J.K., *Principles of Digital Communication and Coding*, McGraw-Hill, New York, 1979.

[3] Lin, S., et Costello, D.J. Jr., *Error Control Coding*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[4] Wu, W.W., Haccoun, D., Peile, R., and Hirata, Y., "Coding for Satellite Communications," *IEEE Journal on Selected Areas in Communiations*, vol. SAC-5, no 4, pp-724-748, May 1987.

[5] Heller, J.A. and Jacobs, I.M. "Viterbi Decoding for Satellite and Space Communications," *IEEE Transactions on Communication Technologies*, vol. COM-19, no 5, pp-751-772, Oct 1971.

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

[7] Chevillat, P.R. and Costello, D.J., "Distance and Computation in Sequential Decoding," *IEEE Transactions on Communications*, vol. COM-24, pp.440-447, April 1977.

[8] Johanesson, R., and Paaske, E., "Further Results on Binary Convolutional Codes with an Optimum Distance Profile," *IEEE Transactions on Information Theory*, vol. IT-24, No. 2, pp.264-268 March 1978.

[9] Massey, J.L., "Threshold Decoding", Cambridge, Mass., M.I.T. Press, 1963.

[10] Wozencraft, J.M., "Sequential decoding for reliable communications," *1957 IRE Nat. Conv. Record*, vol. 5, pt. 2, pp.11-25.

[11] Haccoun, D., Ferguson, M., "Generalized Stack Algorithm for Decoding Convolutional Codes," *IEEE Transaction on Information Theory*, vol. IT-21, no. 6, pp.638-651, November 1975.

[12] Viterbi, A.J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transaction on Information Theory*, vol. IT-13, no. 2, pp.260-269, April 1967.

[13] Forney, G.D.Jr, "The Viterbi Algorithm," *Proceedings of the IEEE*, vol. 61, no 3, pp.268-278, March 1973.

[14] Forney, G.D.Jr, "Coding System design for advanced solar missions," Submitted to NASA Ames Res. Ctr. by Codex Corp., Watertown, Mass., Final Rep., Contract NAS2-3637, December 1967.

[15] Omura, J.K. "On the Viterbi decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-15, pp.177-179, January 1969.

[16] Viterbi, A.J., "Convolutional codes and their performance on communication systems," *IEEE Transaction on Communication Technology*, vol. COM-19, no. 5, pp.751-772, October 1971.

[17] Odenwalder, J.P., "Optimal Decoding of Convolutional Codes," Ph.D. Dissertation, Dept. of Elect. Eng., U.C.L.A., January 1970.

[18] Fano, R.M. "A Heuristic Discussion of Probabilistic Decoding," *IEEE Transaction on Information Theory*, Vol. IT-19, pp.64-73, April 1963.

[19] Zigangirov, K., "Some Sequential Decoding Procedures," *Probl. Peredach Inform.*, vol. 2, no. 4, pp.13-15, 1966.

[20] Jelinek, F., "Fast Sequential Decoding Algorithm Using a Stack," *IBM Res. Develop.*, vol. 13, pp.675-685, November 1969.

[21] Geist, J.M., "Search properties of some sequential decoding algorithms," *IEEE Transaction on Information Theory*, vol. IT-19, pp.519-526, July 1973.

[22] Jacobs, I.M., and Berlekamp, E.R., "A Lower Bound to the Distribition of Computation for Sequential Decoding," *IEEE Transactions on Information Theory*, vol. IT-21, pp.638-651, November 1975.

[23] Gallager, R.G.G., *Information Theory and Reliable Information*, Wiley, NY, 1968.

[24] Haccoun, D., "Variabilité de calculs et débordements de décodeurs séquentiels à pile," *Traitement du Signal*, vol. 3, no. 3, pp.127-143, 1986.

[25] Chevillat, P.R. and Costello, D.J., "A Multiple Stack Algorithm for Erasurefree Decoding of Convolutional Codes," *IEEE Transactions on Communications*, vol. COM-25, no 12, pp.1460-1470, December 1977.

[26] Jelinek, F., and Anderson, J.B., "Instumental Tree Encoding of Information Sources," *IEEE Transactions on Information Theory*, vol. IT-17, pp.118-119, January 1971.

[27] Anderson, J.B., et Ho, C.-W.P., "Architecture and Construction of a Hardware Sequential Encoder for Speech," *IEEE Transactions on Communications*, vol. 25, no 7, pp.703-707, July 1977.

[28] Simons, S.J. and Wittke, P.H., "Low complexity decoders for constant envelope digital modulations," *IEEE Transactions on Communications*, vol. COM-31, pp.1273-1279, December 1983.

[29] Pottie, G.J. and Taylor, D.P. "A comparaison of reduced Complexity decoding algorithms for trellis codes," *Proc. 14th Biennial Symposium on Communications*, Queen's University, Kingston, Ont., Canada, pp.A.1.5-A.1.8, May 29-June 1, 1988.

[30] Lin, C.F., "A Trucated Viterbi Algorithm Approach to Trellis Decoding, Report TR86-3, Rennsalaer Polytechnique Institut, Troy, N.Y., September 1986.

[31] Knuth, D.E., *Sorting And Searching*, Addisson-Wesley, Reading, Mass., 1973.

[32] Lavoie, P., Belzile, J., Toulgoat, M., Haccoun, D., and Savaria, Y., "VLSI Design of a Systolic Priority Queue Chip for Stack Sequential Decoders," *Proc. Canadian Conference on VLSI* , Halifax, NS., Canada, Oct 13-15, 1988.

[33] Mead, C., et Conway, L., *Introduction to VLSI Systems*, Addison-Wesley, Reading, Mass., 1980.

[34] Kung, H.T., "Why Systolic Architectures?," *Computer*, pp.37-46, January 1982

[35] Kung, H.T., and Leiserson, C.E., "Systolic Arrays (for VLSI)," *Proc. Symp. Sparse Matrix Computations and Their Applications*, pp.256-282, Nov. 1978.

[36] Leiserson, C.E., "Systolic Priority Queue," *Proc. of Caltech conf. on VLSI*, 1979.

[37] Chang, C.Y. and Yao, K., "Systolic Array Processing of the Stack Algorithm," submitted *IEEE Transactions on Information Theory*, 1987.

[38] Lavoie, P., "Algorithme de décodage Séquentiel à Pile Tronquée: Analyse Markovienne et Architecture Systolique", Thèse de Doctorat, Dept. Génie Elect., Ecole Polytechnique, Montréal, Canada, Septembre 1990.

[39] Mohan, S., and Sood, A.K., "A Multiprocessor Architecture for the (M,L)-Algorithm Suitable for VLSI Implementation," *IEEE Transactions on Communications*, Vol. 34, No. 12, pp.1218-1224, December 1986.

[40] Simons, S.J., "A Nonsorting VLSI Structure for Implementing the (M,L) Algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 3, pp.538-546, April 1988.

[41] Weste, N., et Eshraghian, K., *Principles of CMOS VLSI Design*, Addison-Wesley, Reading, Mass., 1985.

[42] Bélanger, N., "Architectures Multiprocesseure de Décodeurs Séquentiels à Pile", Mémoire de Maîtrise es Science Appliquées, Dept. Génie Elect., Ecole Polytechnique, Montréal, Canada, Décembre 1989.

[43] Anderson, J.B. and Lin, C.F., "M-Algorithm decoding of channel convolutional codes," *Conf. Proc., 20th Annu. Conf. Inform. Sci. Syst.*, Princton Univ., Princeton, NJ, March 19-21, 1986.

# ANNEXE "A"
# ETUDE SUPPORTANT LA FAISABILITE D'UNE REALISATION ITGE

Cette annexe contient l'information détaillée utile à une réalisation en technologie ITGE d'un décodeur multi-chemins (chapitre 6). Elle présente une cellule de tri en détail avec son chronograme. Une analyse des réseaux de tri ainsi que des bornes sur leur qualité sont développées à la section suivante et finalement le mouvement des données dans la machine est expliqué à la dernière section.

## A.1 Cellule de Tri

Une cellule de tri doit comparer ses deux entrées de façon sérielle, à l'instant où les entrées ne sont plus équivalentes, choisir l'entrée de valeur supérieure, fixer le routage de façon appropriée et demeurer dans cet état jusqu'à la fin du mot. Une fois le mot terminé, la cellule doit revenir dans l'état original. Il est important de remarquer que si les entrées sont identiques, la position du routeur n'est pas importante.

Une cellule de tri composée de 35 transistors est illustrée à la figure A.1. Cette cellule évite les courses possibles et son chronogramme (timing diagram) est donné à la figure A.2.

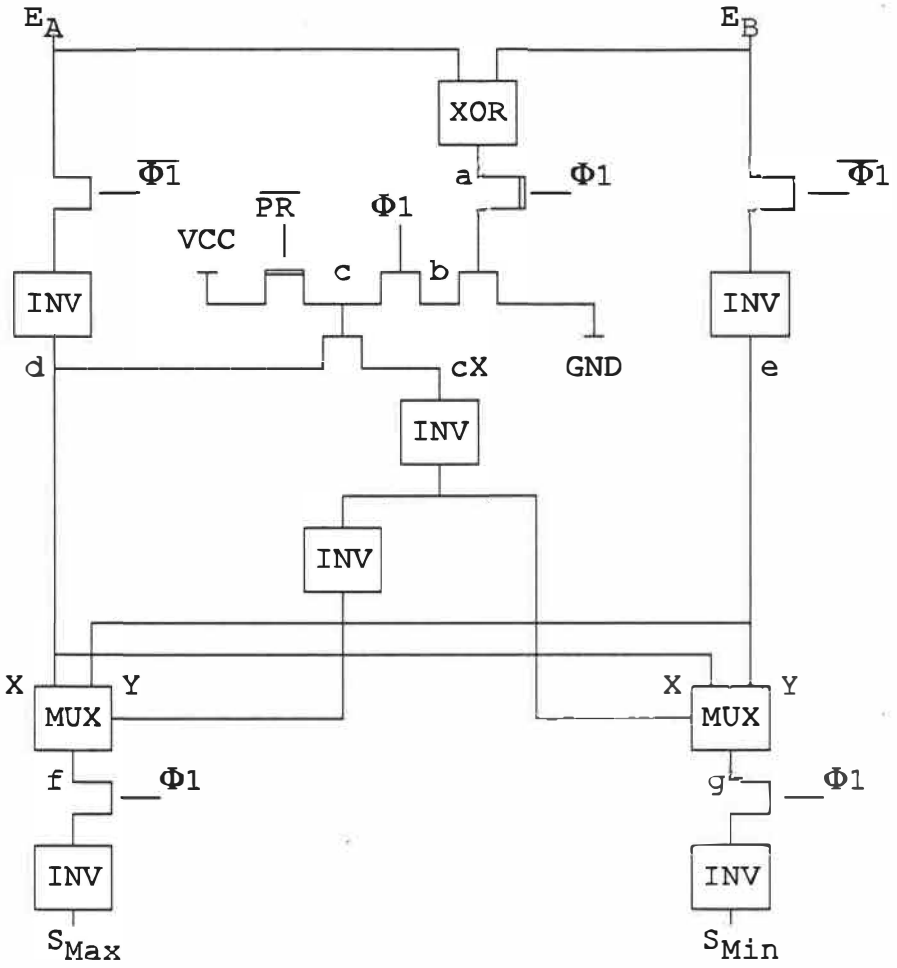


Figure A.1 Schéma logique de la cellule de tri. Les transistors N n'ont qu'une seule ligne sur la grille alors que les P en possèdent deux.

La cellule de tri de la figure A.1 est optimisée pour le nombre de transistors et pour le délai. Etant donné les hautes vitesses désirées, des noeuds dynamiques peuvent être utilisés pour retenir l'information sans avoir de problème de perte de tension. L'utilisation de demi-portes de transmission simplifie le routage des lignes d'horloge et réduit la charge capacitive sur ces dernières.

Les autres éléments de la cellule sont élémentaires. Les multiplexeurs (MUX) sont des cellules à 4 transistors. Les inverseurs (INV) qui suivent les demi-portes de transmission doivent être balancés pour récupérer les niveaux logiques. Finalement, le

OU-EXCLUSIF (XOR) est une cellule à 6 transistors. Bien que cette version du XOR produise des aléas, la demi-porte de transmission (a) garantit que le signal sera stable lorsqu'il sera utilisé. Une version à 3 transistors est possible si on permet une consommation DC.

Le signal actif haut cX contrôle les multiplexeurs et signifie 'choisit l'entrée X'. Le signal PR' est utilisé pour réinitialiser le réseau de tri avant l'envoi des nouvelles données et doit être effectif durant Φ1 pour éviter des problèmes de partage de charges entre les noeuds b et c.

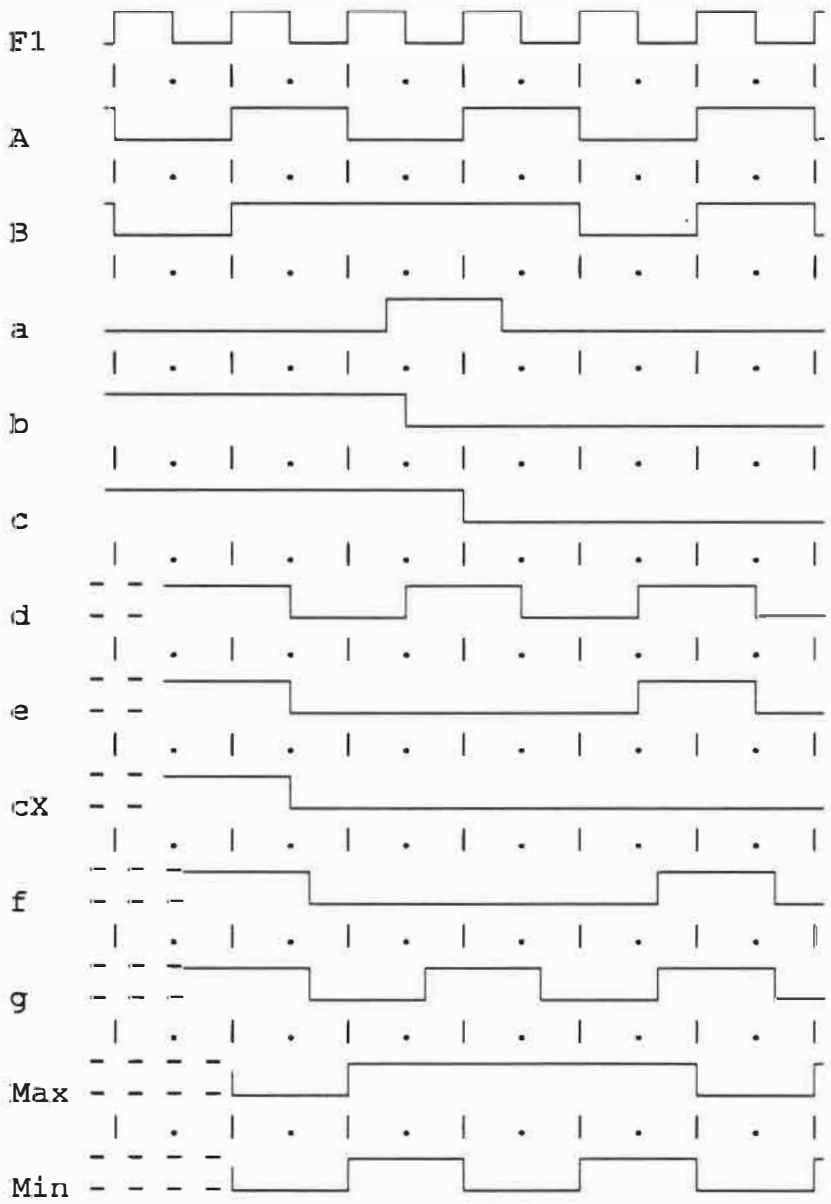


Figure A.2 Chronogramme de la cellule de tri

## A.2 Réseaux de Tri

Les réseaux de tri et de permutations ont été et continuent d'être fort étudiés; le problème de la sélection quant à lui semble étrangement ignoré. Dans le cas du tri,

certaines configurations optimales en terme de délai ou de comparaisons furent obtenues pour les différentes valeurs de N, N étant le nombre d'items original [31]. Les meilleurs résultats de délais et de comparaisons connus pour différentes valeurs de N sont inclus dans le tableau A.1. Seuls les résultats pour $N \leq 8$ sont exacts. Les bornes du chapitre 6 sont utilisées pour le problème de la sélection lorsque $N > 8$.

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D(N) | 0 | 1 | 3 | 3 | 5 | 5 | 6 | 6 | 7 | 7 | 8 | 8 | 9 | 9 | 9 | 9 |
| C(N) | 0 | 1 | 3 | 5 | 9 | 12 | 16 | 19 | 25 | 29 | 35 | 39 | 46 | 51 | 56 | 60 |

Tableau A.1 Délai et Nombre Minimum de Comparateurs pour plusieurs valeurs de N.

Ces réseaux de tri complet de même que les réseaux de sélection complets ne pouvant atteindre le délai minimum désiré lg2M, des réseaux de sélection partielle sont examinés. La classe de réseaux étudiés doit choisir les M plus grands éléments parmi 2M. L'ordre des éléments choisis n'est pas important mais le délai doit être de l'ordre de O(lg2M).

## A.2.1 Evaluation des Performances

Etant donné le réseau de sélection de la figure A.3, le problème est d'évaluer et de prévoir la qualité de la solution obtenue. Une méthode permettant l'analyse des performances, ainsi que la qualité de la sélection effectuée doit être développée.

Les définitions suivantes seront utilisées pour établir les bornes sur la qualité de la sélection. Elles sont basées sur la figure A.3. Les *items* ou *éléments* traversent la figure de gauche à droite sur les *lignes*. Une ligne est définie comme un trait horizontal (fil ou bus). La *position* d'une ligne est son emplacement physique dans l'image en commencant à compter à partir du bas. La *valeur* d'une ligne est la valeur de l'item à comparer passant sur ce fil. Chaque ligne comporte un certain nombre de *comparateurs*.

Les comparateurs sont représentés par des traits verticaux. Tous les comparateurs de même distance du début du réseau sont dit de même *rang* ou *étage*. Ainsi la figure A.3 illustre un réseau de 2 rangs ou étages, de 4 comparateurs, et de 4 lignes.

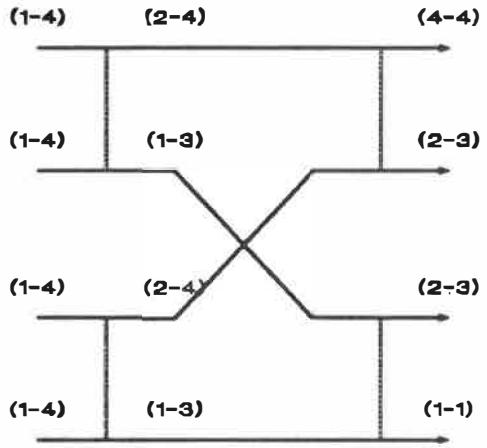


Figure A.3 Réseau de sélection à 4 entrées possédant 4 comparateurs.

Dans notre méthode, une borne inférieure et une borne supérieure sur les valeurs possibles des items triés sont attachées à chaque ligne du réseau de tri. La borne inférieure représente la plus petite valeur possible sur cette ligne alors que la borne supérieure indique la plus grande valeur possible sur cette même ligne.

Les règles régissant les bornes après un comparateur sont définies en fonction des bornes avant ce comparateur. Les règles utilisées ici supposent que toutes les lignes sont indépendantes, c'est-à-dire que l'intersection des ensembles des lignes originales couvertes par les entrées $E_A$ et $E_B$ est nul. Les réseaux de sélection partielle qui nous intéressent utilisent des comparaisons binaires et lg2M étages, par conséquent des lignes indépendantes sont possibles et préférables.

Les bornes sont données en fonction des valeurs des items à trier. Pour simplifier l'analyse on suppose que les items à trier ont des valeurs entières, uniques et comprisent entre 1 et N = 2M. La sortie $S_{Max}$ de chaque comparateur représente la

valeur d'entrée la plus près de N alors que la sortie $S_{Min}$ représente la valeur d'entrée la plus petite (près de 1).

Les bornes $S_{Max}(Sup,Inf)$ et $S_{Min}(Sup,Inf)$ sont facilement calculables. La borne supérieure de la sortie $S_{Max}$, $S_{Max}(Sup)$, représente la plus grande valeur possible permise par cette sortie. Il est donc évident qu'elle doit être égale à la valeur maximale des bornes supérieures des entrées $E_A$ et $E_B$ soit $MAX(E_A(Sup),E_B(Sup))$.

La borne inférieure de la sortie, $S_{Max}$, $S_{Max}(Inf)$, représente la valeur minimale permise par cette sortie. Chacune des entrées $E_A$ et $E_B$ possède respectivement Inf-1 lignes de valeur inférieure, si ces lignes sont indépendantes, $S_{Max}(Inf)$ sera la somme du nombre de lignes inférieures des entrées. Les équations pour $S_{Max}$ sont alors:

$$\begin{aligned} S_{Max}(Sup) &= MAX(E_A(Sup),E_B(Sup)) \\ S_{Max}(Inf) &= E_A(Inf) + E_B(Inf) \end{aligned} \qquad (A.1)$$

Par un raisonnement similaire les équations suivantes sont dérivées pour la sortie $S_{Min}$:

$$\begin{aligned} S_{Min}(Sup) &= N - (E_A(Sup) + E_B(Sup)) - 1 \\ S_{Min}(Inf) &= MIN(E_A(Inf),E_B(Inf)) \end{aligned} \qquad (A.2)$$

Les équations A.1 et A.2 sont utilisées pour analyser les performances d'un réseau dont les entrées des comparateurs sont indépendantes. Il est possible de dériver des équations pour prévoir la qualité de tels réseaux.

Il suffit de déterminer les bornes possibles à la sortie du réseau de tri ainsi que le facteur de répétition de ces bornes. Pour un réseau de N éléments et lgN étages dont les entrées des comparateurs sont indépendantes, les bornes de sortie d'une ligne dépendent uniquement du chemin suivi par la sortie. Une sortie qui aurait été minimale min fois et maximal lgN-min fois aurait comme bornes $(2^{lgN-min},N+1-2^{min})$. Si min = lgN, nous avons la plus petite entrée et la borne indique (1,1). Par contre si min = 0 alors nous avons la plus grande et la borne est (N,N).

Le nombre de sorties ayant une borne spécifique, même *classe*, est donné par C(Min,lgN) où C(a,b) est la fonction combinaison définie comme:

$$C(a,b) = \frac{b!}{a!(b-a)!}.$$

La qualité du tri est donc définie par:

$$C(min, lgN) \quad (2^{lgN-min}, N+1-2^{min}) \qquad (A.3)$$

Le problème est la sélection des M plus grandes valeurs parmi 2M. La fonction C(a,b) étant symétrique, il devient évident que deux cas existent: selon que lgN soit paire ou impaire. Si lgN est paire cela implique une valeur centrale dans la distribution du nombre de lignes ayant une borne spécifique. La moitié de ces lignes devront être choisies et l'autre moitié rejetées. Par contre une valeur de lgN impaire implique une distribution du nombre de lignes ayant une borne spécifique à deux valeurs centrales distinctes, il est donc aisé de choisir laquelle on doit éliminer.

Dans les deux cas, les

$$2^{\lceil 0.5 lgN \rceil} - 1 \qquad (A.4)$$

éléments maximaux sont toujours présents et par symétrie ce même nombre d'éléments minimaux sont absents. On remarque que la fonction $\lceil\ \rceil$ favorise les valeurs impaires de lgN. La proportion d'élément maximaux toujours présent est donné par:

$$\frac{2^{\lceil 0.5 lgN \rceil} - 1}{N/2} \qquad (A.5)$$

On remarque également que cette proportion diminue avec une augmentation de N. Ces bornes sont des bornes inférieurs, on peut toujours construire un cas pathologique pour lequel il n'y aurait que les items garantis qui soient "bons" parmi les M choisis. Cependant, en moyenne le nombre de "bons" items est plus élevé.

Le nombre de valeurs possibles dans une classe S est donné par:

$$S = Sup - Inf = \frac{(2^{min} - 1)(N - 2^{min})}{2^{min}} + 1 \qquad (A.6)$$

dans cette équation min est constant car tous les éléments d'une même classe on été minimums le même nombre de fois. Par contre le nombre de "bons" (supérieur en valeur à N/2) éléments b dans cette classe est égal à:

$$b = Sup - N/2 = N/2 + 1 - 2^{min}. \qquad (A.7)$$

Si les bons éléments sont répartis également entre toutes les sorties possibles, alors la moyenne de "bons" items dans une catégorie $E_C[n]$ est:

$$E_C[n] = \frac{C(min, lgN)\,MIN(b, S)}{S} \qquad (A.8)$$

en sommant pour toutes les valeurs de min des catégories choisies on obtient:

$$E[n] = \sum_{min=0}^{\lceil 0.5lgN \rceil} E_C[n] \quad , lgN \text{ impaire}$$
$$E[n] = \sum_{Min=0}^{\lceil 0.5lgN \rceil} E_C[n] + 0.5*E_C[n]\Big|_{min=\lceil 0.5lgN \rceil+1} \qquad (A.9)$$

On remarque l'apparition d'un terme supplémentaire lorsque lgN est impaire. On suppose encore une fois que les "bons" éléments sont répartis de façon uniforme sur l'ensemble des sorties possibles de cette classe.

L'ajout d'autres étages de comparateurs peut améliorer la fonction de sélection, cependant les lignes couvertes par les comparaisons des étages additionnels ne sont plus indépendantes. De façon plus générale, les bornes sont calculées en comptant le nombre de lignes originales couvertes par chaque entrée et en ne comptant les entrées communes qu'une seule fois.

$$S_{Max}(Sup) = MAX(E_A(Sup), E_B(Sup)) \qquad (A.10)$$
$$S_{Max}(Inf) = E_A(Inf) \cup E_B(Inf)$$

$$S_{Min}(Sup) = N - (E_A(Sup) \cup E_B(Sup)) - 1 \qquad (A.11)$$
$$S_{Min}(Inf) = MIN(E_A(Inf), E_B(Inf))$$

où $\cup$ représente la fonction union.

## A.2.2 Configurations de Réseaux

On voit à la figure A.3 un réseau avec N=4 items, 2 étages de comparateurs. A chaque comparateur, l'item possédant la plus petite valeur est toujours placé sur la ligne inférieure. Il en est de même pour tous les graphes subséquents.

Ainsi, à la figure A.3, la position supérieure (4-4) ne peut contenir que la valeur supérieure et la position inférieure (1-1) ne peut contenir que la plus petite valeur. Les deux positions mitoyennes contiennent toujours le deuxième et le troisième item.

Comme le problème se résume par la sélection de M éléments parmi 2M, si on choisit les M positions supérieurs alors le réseau garantit la présence de la plus grande valeur, l'absence de la plus petite valeur et une sélection parfaite dans 50% des cas .

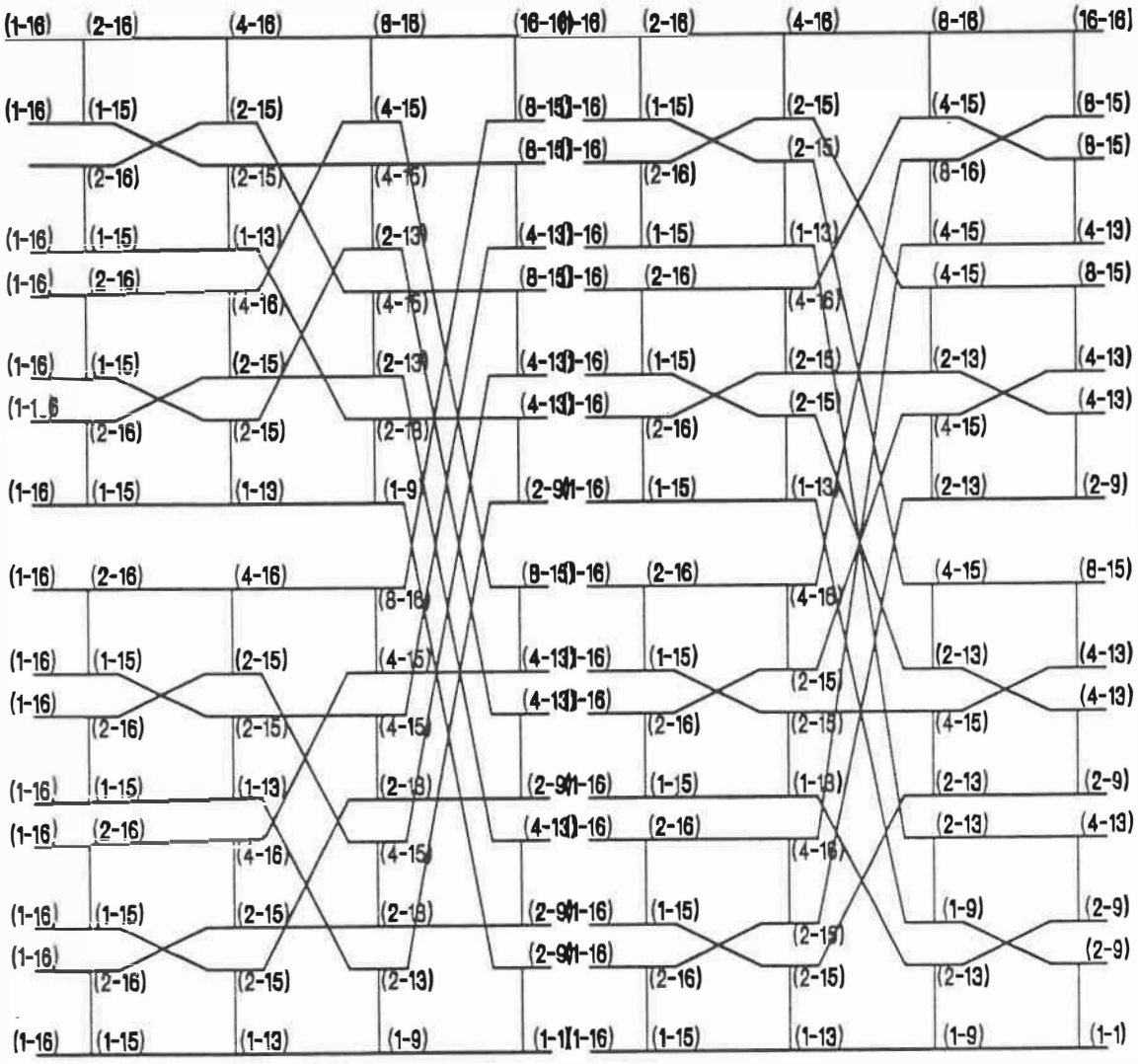

Figure A.4 Réseaux de tri à 16 entrées comprenant 32 comparateurs.

Les deux réseaux de la figure A.4 sont équivalents à l'exception du routage moins régulier mais en moyenne plus court dans la version de droite. Ces réseaux sont basés sur le réseau Banyan et toutes les comparaisons sont indépendantes.

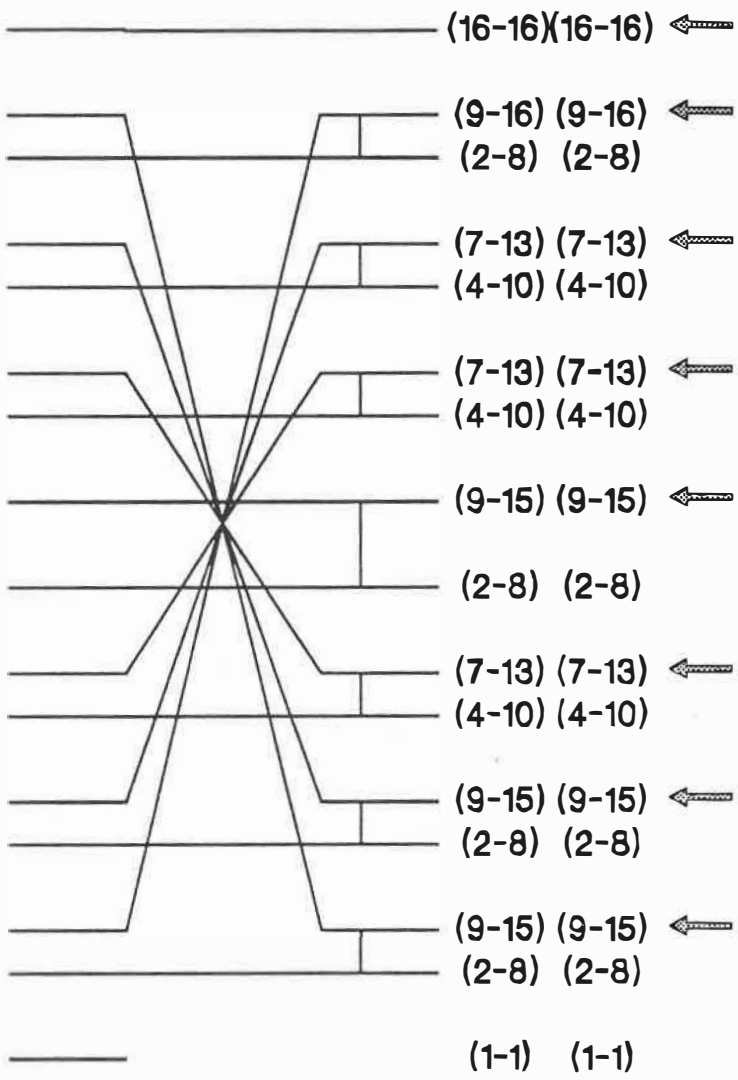


Figure A.5 Papillon de comparateurs à 16 entrées.

Les réseaux de la figure A.4 n'utilisent pas un grand nombre d'étages mais n'obtiennent pas non plus une qualité de sélection extraordinaire. La position du plus grand est connue et les éléments 16-14 sont garantis alors que les 1-3 ne se retrouvent jamais dans la sélection.

L'ajout d'un autre étage de comparateurs (figure A.5) basé sur la technique d'Alekseyev donne un gain substantiel. Cet étage est le même pour les deux configurations de la figure A.4.

Les chiffres de gauche correspondent aux bornes pour le réseau de gauche alors que ceux de droite correspondent au réseau de droite. On remarque que bien que l'ordre ne soit pas le même, les lignes choisies (marquées par des flèches) demeurent inchangées. Le nouvel étage ajoute un gain substantiel à la qualité de la sélection, ainsi les items de valeur 11-16 sont garantis alors que les items de valeur 1-6 ne seront jamais présents. On doit également remarquer que les items de valeur 9 et 10 ont plus de chance d'être présents que d'être absents.

Les lignes n'étant plus indépendantes dans le dernier étage, les formules A.10 et A.11 doivent être utilisées pour calculer les bornes.

Une répartition uniforme des sorties est obtenue en inversant les comparateurs de façon à ce que la plus grande valeur sorte sur la ligne inférieure. Cette opération ne coûte rien et prépare les sorties du réseau de tri pour les entrées des extenseurs.

Bien que n'étant qu'une des configurations possibles, il serait surprenant qu'un autre réseau puisse atteindre une meilleure sélection que le réseau Banyan car toutes les comparaisons sont effectuées sur des ensembles indépendants. Il existe cependant un recoupement des ensembles au dernier étage.

Le problème de la sélection optimale utilisant un réseau de tri n'est pas encore résolu. Il est cependant improbable que la qualité de sélection du Banyan puisse être améliorée par un autre algorithme agissant sous les mêmes conditions.

Une façon d'améliorer la qualité du tri par rapport à la technique Banyan est d'utiliser la connaissance du comportement du décodeur. Les meilleurs chemins ayant tendance à donner naissance à des bons chemins, il est possible de biaiser l'algorithme.

Cette approche ne fut pas étudiée ici. On note que le tri de la technique biaisée ne sera préférable à celui du réseau Banyan que dans ce cas précis et qu'il est fort probable qu'il soit de qualité inférieure pour le problème général.

## A.3 Mouvement des Données

Cette section illustre le mouvement des données dans l'extenseur. Au premier cycle, les registres à décalage des extenseurs initialisent les données aux bonnes valeurs c'est-à-dire les états = 0 et les métriques = $-\infty$, sauf l'extenseur 1 pour lequel la métrique initiale est 0. Les métriques sont initialisées dans les LIFOs de sortie. Donc au premier cycle, $M_{14}$ (MSB) passe dans le réseau de tri et le contenu du FIFO ne change pas.

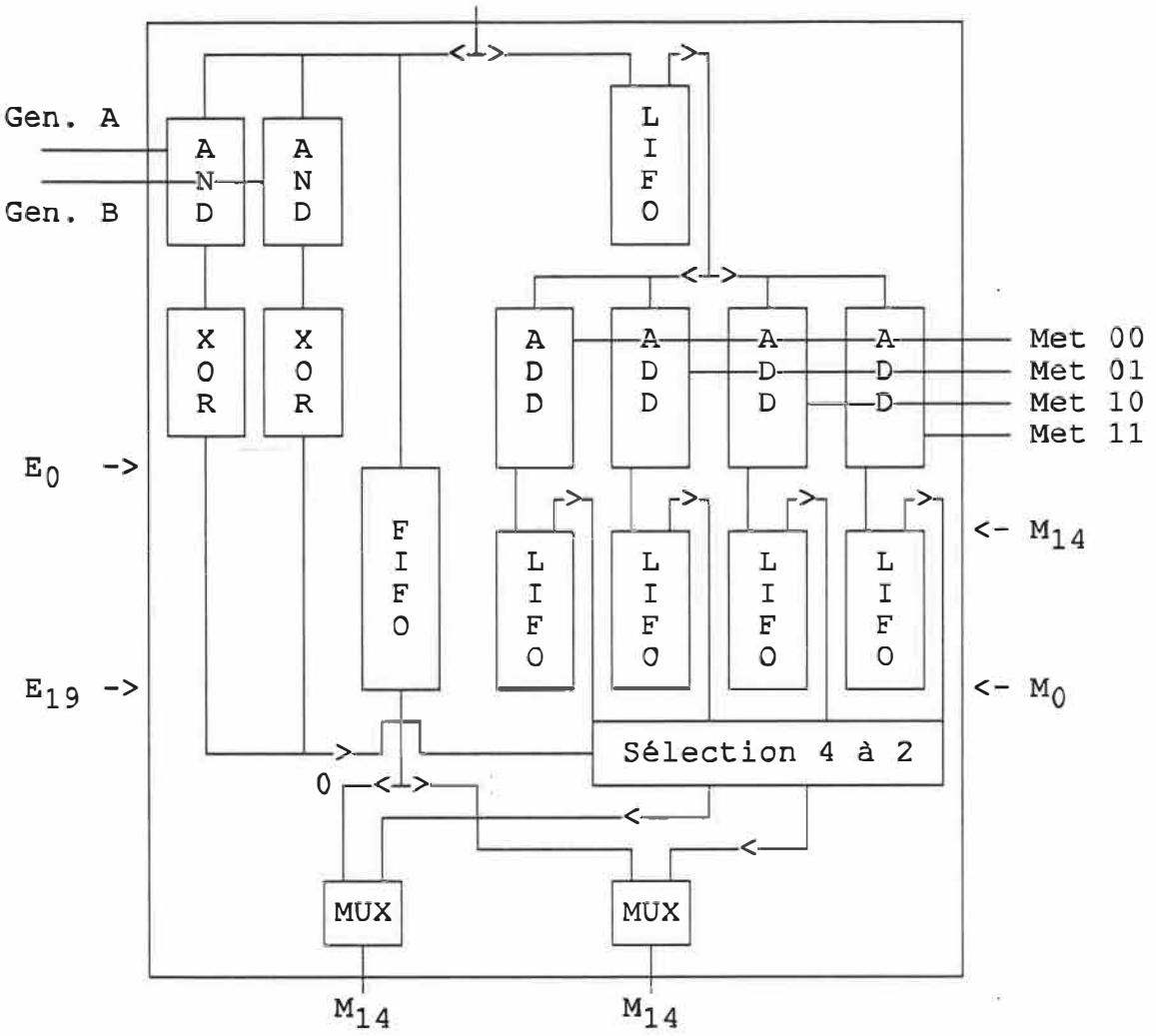


Figure A.6 Données dans l'extenseur à l'instant 1.

Après lg2M (8 pour M=128) cycles le premier bit de métrique finit de passer à travers le réseau de tri et entre dans le LIFO supérieur. Les bits de la métrique continuent de sortir du bas de l'extenseur. On est rendu à la 14-lg2M = 6 bit. L'état demeure dans le FIFO.

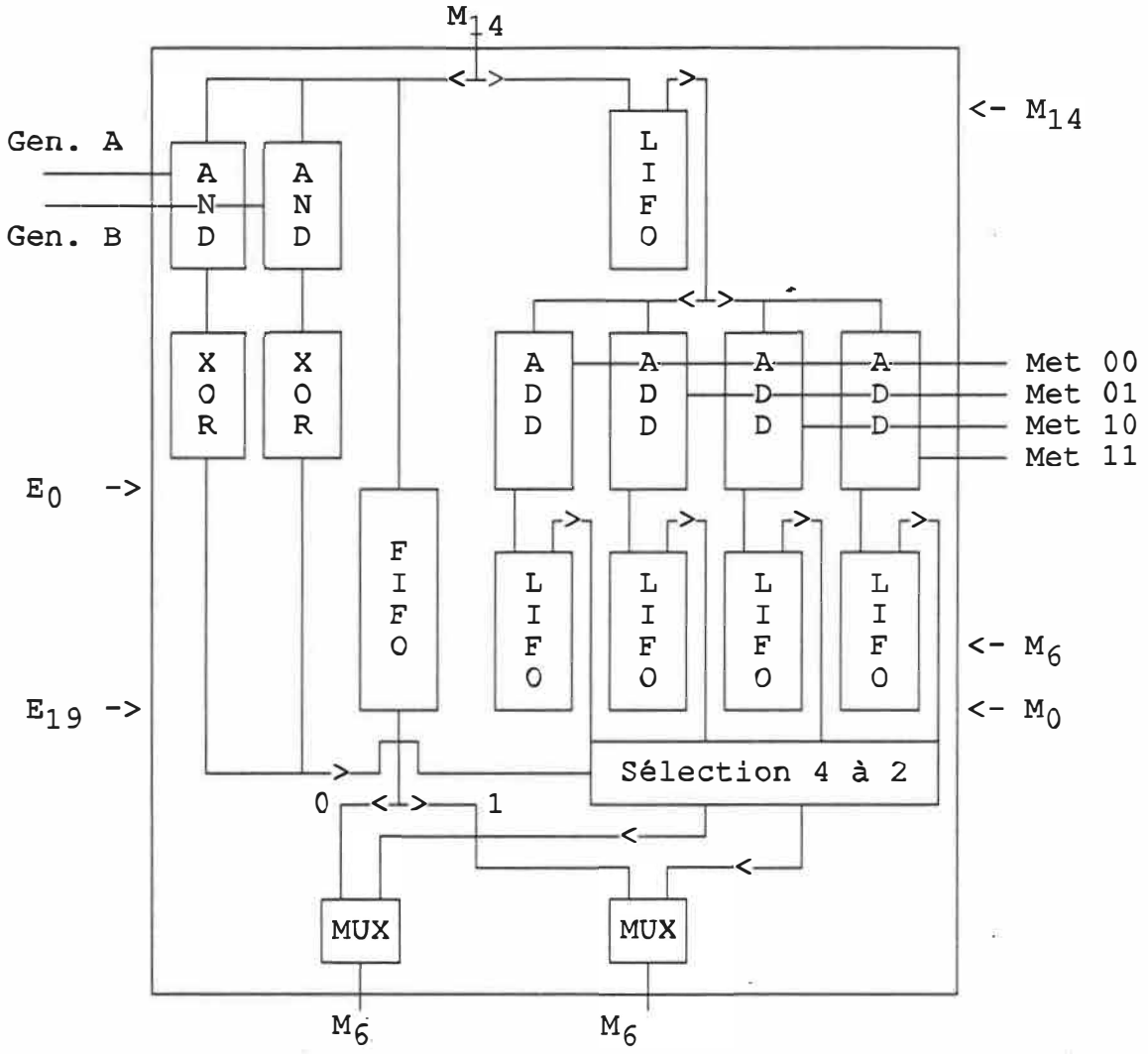


Figure A.7 Données dans l'extenseur à l'instant lg2M.

Après 15 cycles le dernier bit de métrique, $M_0$, entre dans le réseau de tri. Au cycle suivant le premier bit d'état, $E_{19}$, suivra dans le réseau de tri. Au haut de l'extenseur, le bit de métrique $M_7$ entre dans le LIFO.

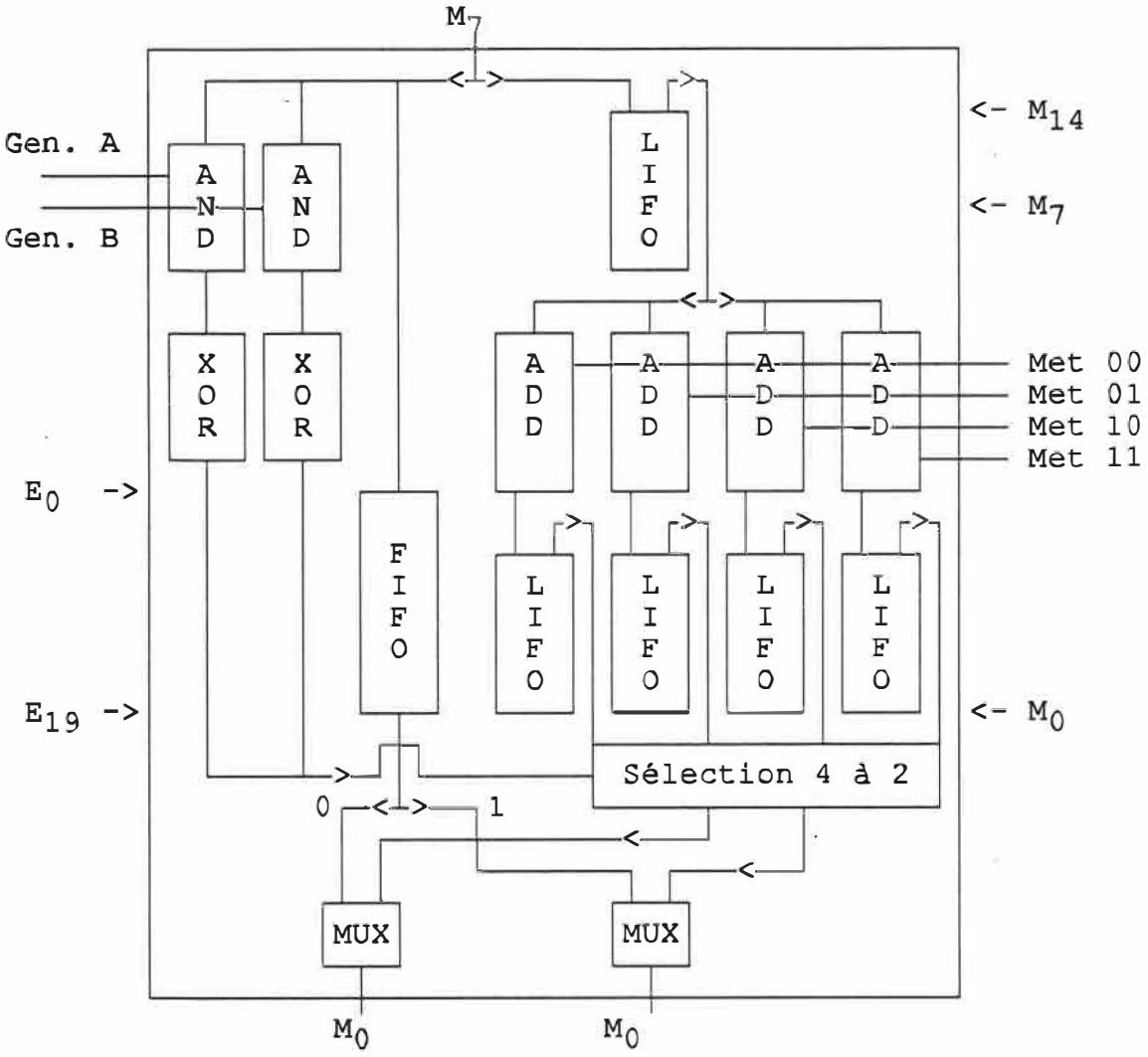


Figure A.8 Données dans l'extenseur à l'instant 15.

Après 23 cycles, le premier bit de l'état arrive en haut de l'extenseur. La métrique est complètement entrée dans le LIFO et on commence à additionner les métriques de branches à cette dernière. On commence à inverser la nouvelle métrique accumulée dans les LIFO de sortie. Les bits de l'état suivent progressivement.

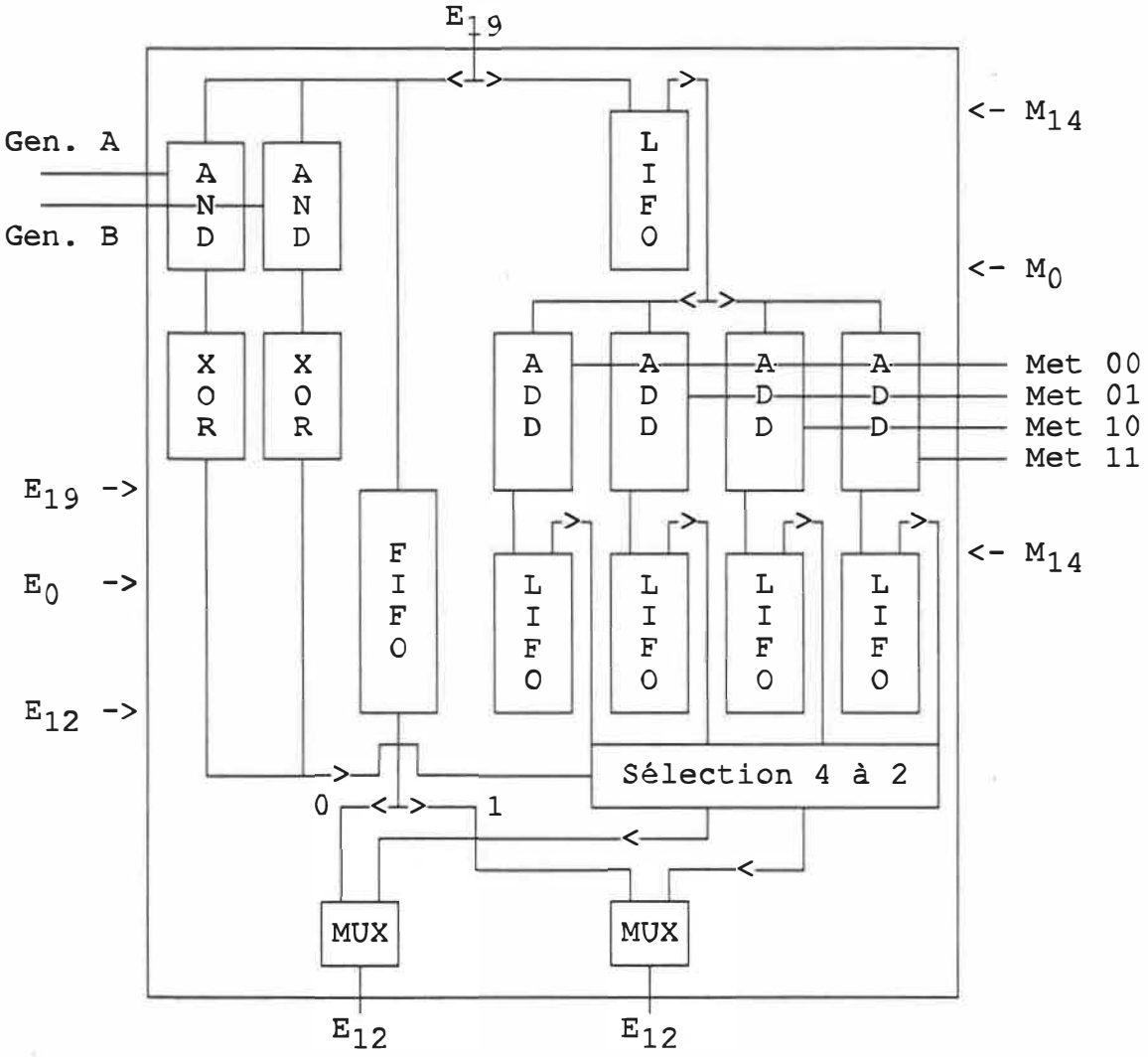


Figure A.9 Données dans l'extenseur à l'instant 23.

Après 35 cycles le dernier bit de l'état sort de l'extenseur. Les nouvelles métriques accumulées sont presque calculées; elles finiront au cycle 38 et attendront l'arrivée du dernier bit d'état jusqu'au cycle 43 pour commencer à sortir de l'extenseur.

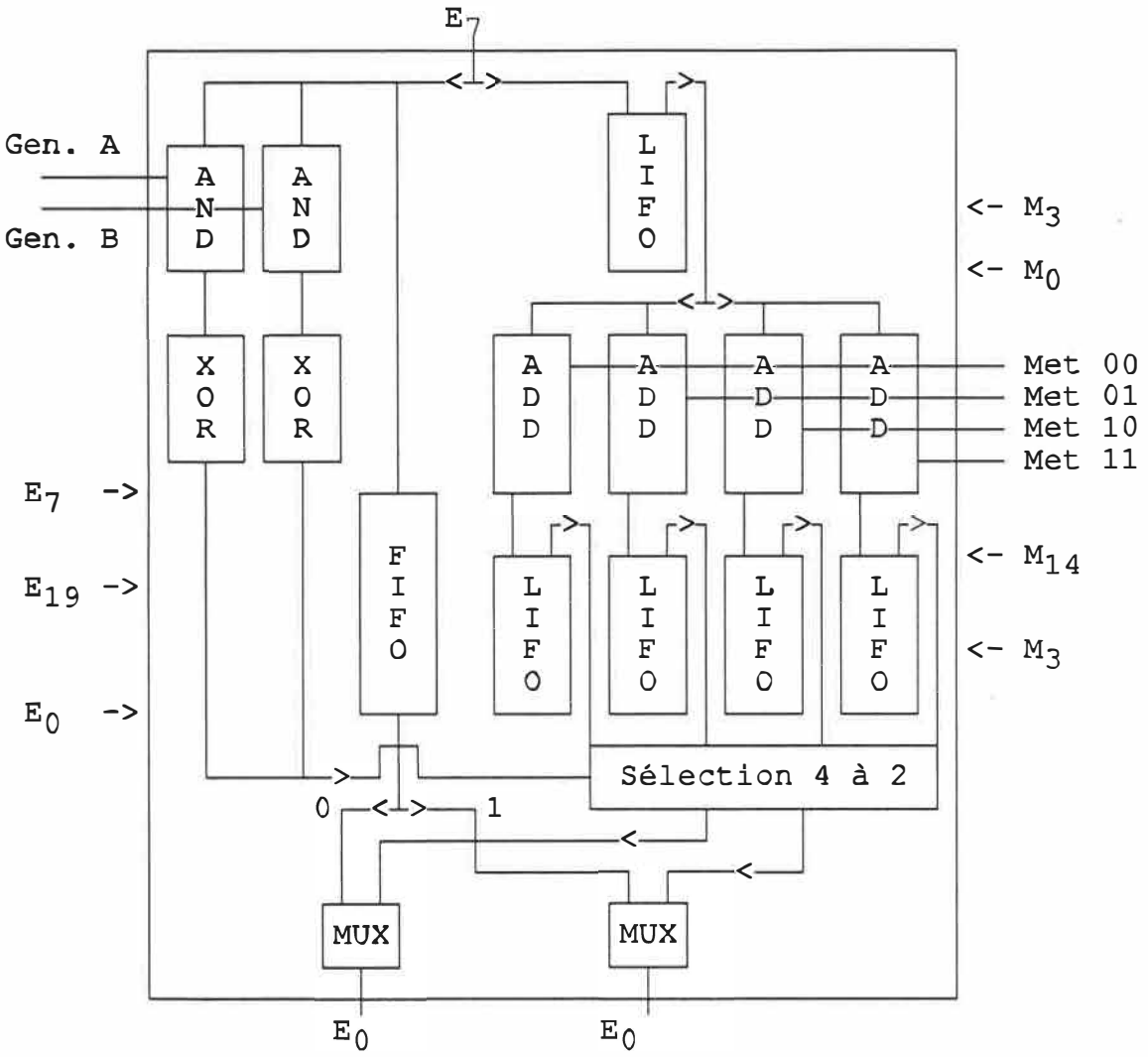


Figure A.10 Données dans l'extenseur à l'instant 35.

Au cycle 43 le dernier bit de l'état arrive dans l'extenseur, les métriques appropriées commencent à sortir et on se retrouve comme au premier cycle.

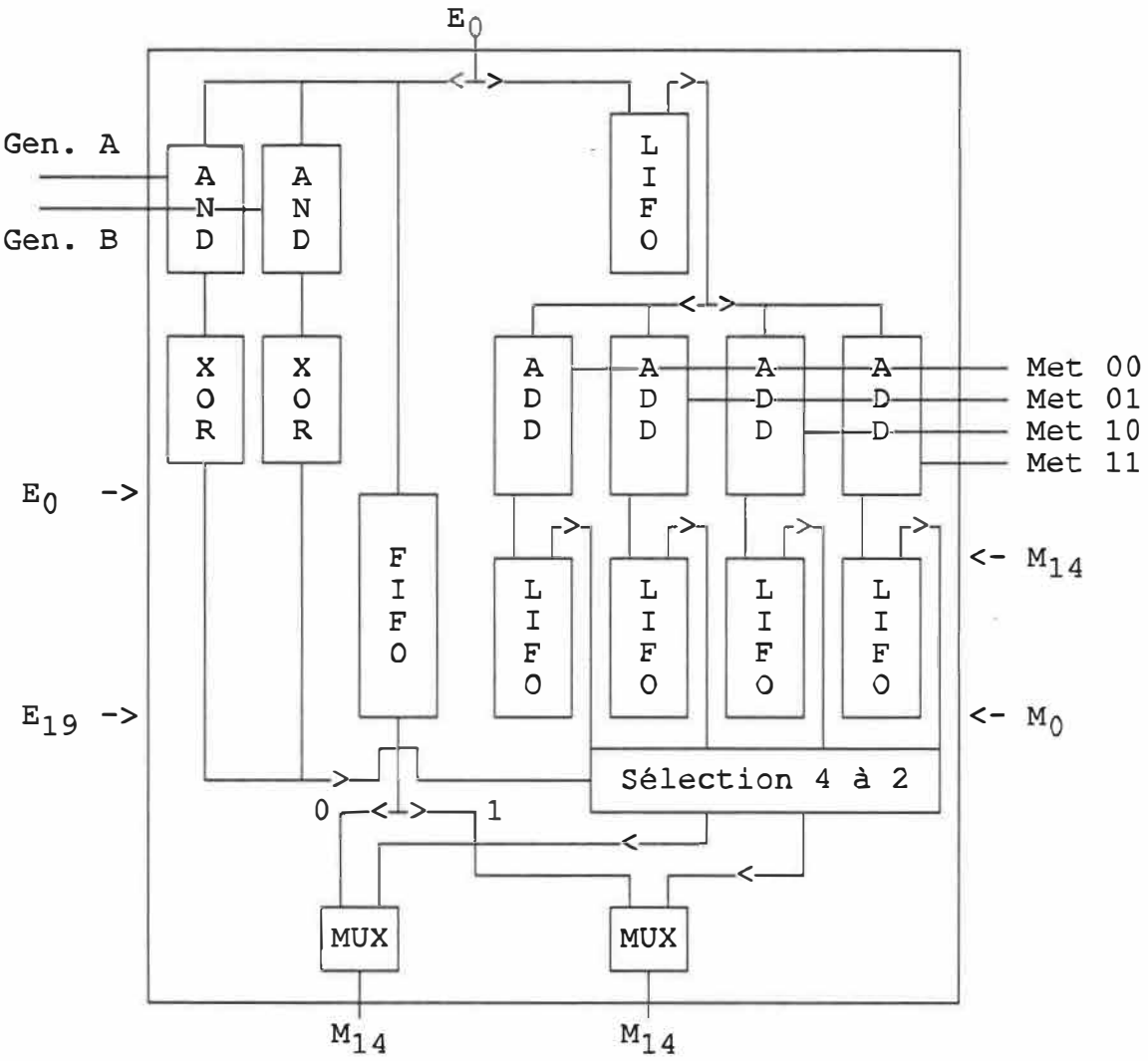


Figure A.11 Données dans l'extenseur à l'instant 43.

On constate donc que 43 cycles sont requis pour avancer d'un pas dans l'arbre de décodage. Le nombre de cycles requis provient de la somme du nombre de bits dans l'état, la métrique et le nombre d'étages dans le réseau de tri.

# ANNEXE "B"
# SIMULATION

## B.1 Méthode de simulations

La simulation d'une architecture massivement parallèle, un processeur par chemin, sur un ordinateur uniprocesseur requiert beaucoup de temps de calcul. Dans le but de réduire le temps nécessaire à la confection d'une courbe, l'algorithme de simulation suivant fut utilisé.

1-Initialiser le nombre de chemins à la plus petite valeur désirée (normalement 16).
2-Initialiser le rapport signal à bruit à une valeur de $E_b/N_0$ (normalement 4.5 dB).
3-Initialiser le fichier de semence en erreur au fichier de semence en erreur du même rapport signal à bruit mais du nombre de chemin précédent. Si ce dernier est absent (première simulation), l'ordinateur choisira lui-même les semence à simuler.
4-Simuler les trames ayant obtenu des erreurs. Si une trame cause des erreurs noter sa semence dans un fichier.
5-Augmenter le rapport signal à bruit et retourner à l'étape 4 jusqu'à ce que qu'il n'y ait plus aucune erreur.
6-Augmenter le nombre de chemins jusqu'à la valeur désirée ou jusqu'à ce qu'il n'y ait plus aucune erreur et retourner à l'étape 2.

Algorithme B.1 Algorithme de simulation.

Cet algorithme réduit le temps requis pour une simulation de deux façons. Même avec un petit nombre de chemins et un rapport signal à bruit de l'ordre de $E_b/N_0$ = 4.5 dB, plus de 80% des trames sont sans erreur. Si ces trames ne produisent aucune erreur dans ces conditions, elles ne produiront aucune erreur dans de meilleures conditions. Après chaque incrément du rapport signal à bruit un nombre substantiel de trames n'ont pas besoin d'être resimulé. Le même argument tient en ce qui a trait au nombre de chemins. Si une trame peut être décodée correctement avec M chemins, elle le sera probablement avec 2M.

Le nombre total de trames simulées pour arriver à une série de courbes de performances d'erreur selon le rapport signal à bruit et selon le nombre de chemins peut ainsi être réduit considérablement comme le démontre les résultats du tableau B.1.

| $E_b/N_0$ (dB) | M = 32 | 64 | 128 | 256 |
|---|---|---|---|---|
| 4.5 | 50000 | 5265 | 2594 | 1244 |
| 4.7 | 5265 | 2594 | 1224 | 637 |
| 4.9 | 3627 | 1647 | 750 | 359 |
| 5.1 | 2295 | 936 | 404 | 190 |
| 5.3 | 1423 | 520 | 226 | 106 |
| 5.5 | 967 | 323 | 132 | 66 |
| 5.7 | 645 | 209 | 82 | 34 |
| 5.9 | 409 | 120 | 53 | 17 |
| 6.1 | 262 | 75 | 31 | 12 |
| 6.3 | 173 | 48 | 23 | 9 |
| 6.5 | 116 | 36 | 17 | 8 |
| 6.7 | 74 | 18 | 10 | 6 |
| 6.9 | 49 | 13 | 9 | 5 |
| 7.1 | 33 | 11 | 8 | 4 |
| 7.3 | 22 | 9 | 7 | 4 |
| 7.5 | 15 | 7 | 5 | 4 |
| 7.7 | 13 | 5 | 3 | 2 |
| 7.9 | 13 | 4 | 2 | 1 |

Tableau B.1 Progression du nombre de trames simulées pour K = 20, bidirectionnel à longueurs constantes.

Bien que ne simulant que très peu de bits pour les rapports signal à bruit élevés et un grand nombre de chemins, les résultats sont valides sur l'ensemble des trames de départ. En effet, les résultats furent validés à l'aide d'une série complète de simulations. C'est-à-dire que l'on recommence tous les bits à chaque rapport signal à bruit et pour chaque nombre de chemins.

Les simulations furent également effectuées avec seulement des zéro comme bits d'information. Bien que cette façon d'effectuer les simulations puisse biaiser les

résultats, il fut démontré avec des simulations utilisant des bits d'information aléatoires que le biais n'est pas significatif.

L'ensemble des procédures utilisées pour raccourcir le temps de simulation n'a pas affecté les résultats de plus que de 0.1 dB. Cette valeur étant insignifiante par rapport au temps de calcul économisé, les procédures furent gardées tout au long des simulations.

## B.2 Logiciel de simulations

Les simulations furent effectuées sur des ordinateurs de type SUN3, SUN4 et SPARC. Le code écrit en langage C est compilé pour chaque architecture et il est portable sans modifications.

Le simulateur comprend environs 7500 lignes de code alors que son environnement en comprend 4000. Ils furent écrit sur une période d'un an en utilisant 2 année-hommes.

Le simulateur lui-même est capable de simuler plusieurs architectures de décodeurs séquentiel et multi-chemins selon plusieurs longueurs de contraintes, patrons de perforations, taux de codage et rapport signal à bruit. Les statistiques à évaluer sont également variables et dépendantes de l'architecture utilisée.

L'environnement est une interface graphique facile à utiliser qui fonctionne par fenêtres, menus, curseurs ou boutons de sélection. L'environnement s'occupe de fabriquer les fichiers de données requis par le simulateur et de recompiler ce dernier lorsqu'il se doit. Il permet également l'inclusion de procédures de déverminage ainsi que la spécification d'horaires de simulations.

Des logiciels d'extraction et d'analyse des données existent aussi. Les fichiers de sorties sont tous sous un format ASCII standard et peuvent aussi bien être passés d'un logiciel à un autre ou bien être directement imprimés.

La facilité d'utilisation n'a cependant pas affecté les performances du simulateur. Le simulateur prend en moyenne 33.3ms par bit-chemin sur une station Sparc. Ainsi une trame de 500 bits d'information plus 19 bits de queue prend 1.1 secondes avec l'algorithme de décodage bidirectionnel à 64 chemins.

# ANNEXE "C"
# NOTATION

| Symbole | Signification |
|---|---|
| $\oplus$ | Addition modulo-2. |
| $\lceil A \rceil$ | Arrondit A au prochain entier. |
| $\approx$ | Approximativement. |
| $d_H$ | Distance de Hamming. |
| lgN | logarithme base 2: $\log_2 N$. |
| lnN | logarithme néperien: $\log_e N$. |
| logN | logarithme base 10: $\log_{10} N$. |
| N! | N Factoriel: N * N-1 * N-2 * ... * 2 * 1. |
| FDC(n) | Fonction de distance de colonne de l'ordre n. |
| $\lim_{k \to A} f(k)$ | Limite de f(k) lorsque k tends vers A. |
| K | Longueur de contrainte. |
| $L_C$ | Longueur de convergence des chemins explorés. |
| L | Longueur d'une trame. |
| v | Mémoire du codeur (K-1). |
| $\underline{U}$ | Message à transmettre. |
| $\underline{X(U)}$ | Message codé. |
| $\underline{U'}$ | Message décodé. |
| $\underline{Y}$ | Message reçu. |
| V | Nombre d'additionneurs modulo-2. |
| M | Nombre de chemins explorés. |
| P[A] | Probabilité de A. |
| P[A \| B] | Probabilité conditionnelle de A étant donné B. |
| $P_B$ | Probabilité d'erreur par bit. |
| $P_E$ | Probabilité d'erreur par bloc. |
| $\prod_{k=A}^{B} f(k)$ | Produit de la fonction f(k) pour les valeurs de k entre A et B. |
| PDC | Profil de distance de colonne d'un code. |
| $\sqrt{N}$ | Racine carrée de N. |
| $\sum_{k=A}^{B} f(k)$ | Somme de la fonction f(k) pour les valeurs de k entre A et B. |
| R | Taux de codage (1/V). |
| $\cup$ | Union. |
| $\underline{X}$ | Vecteur X. |