Titre: Title:	Décodeur séquentiel haute vitesse : réalisation et tests préliminaires
Auteurs: Authors:	David Haccoun
Date:	1984
Type:	Rapport / Report
Référence: Citation:	Haccoun, D. (1984). Décodeur séquentiel haute vitesse : réalisation et tests préliminaires. (Technical Report n° EP-R-84-19). https://publications.polymtl.ca/9613/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EP-R-84-19
URL officiel: Official URL:	
Mention légale: Legal notice:	

DECODEUR SEQUENTIEL HAUTE VITESSE
REALISATION ET TESTS PRELIMINAIRES*

par

Dr. David Haccoun, ing.
Professeur titulaire
Département de Génie Electrique
Ecole Polytechnique de Montréal

(1984)

* Cette recherche a été subventionnée en partie par le Conseil National de Recherches en Sciences Naturelles et en Génie du Canada, Formation des Chercheurs et Action Concertée, compagnie Marconi Canada.

TABLE DES MATIÈRES

		page
	AVANT-PROPOS	v
1.	INTRODUCTION	1
	1.1 Plan du rapport	2
	1.2 Modèle d'un système de communication numérique	3
2.	CODAGE CONVOLUTIONNEL ET DÉCODAGE PROBABILISTE	6
	2.1 Structure des codes convolutionnels	7
	2.1.1 Arbre et trellis	7
3.	DÉCODAGE SÉQUENTIEL	10
	3.1 Algorithme à pile	11
	3.1.1 Structure des données de l'algorithme	13
	3.1.1.2 Simplification : utilisation de fanion	17
	3.2 Effort de calcul du décodeur séquentiel	20
	3.2.1 Problèmes de débordement	22
4.	RÉALISATION MATÉRIELLE DU DÉCODEUR SÉQUENTIEL	24
	4.1 Historique	24
	4.2 Caractéristiques du décodeur	25
	4.3 Description du décodeur séquentiel	26
	4.3.1 Principes de base	26
	4.3.2 Conception du logiciel de décodage	27
	4.3.3 Conception du matériel	29

5. PROCÉDURES DE TESTS 51

5.1 Généralités 51

5.2 Cueillette des statistiques 51
5.2.1 Métrique accumulée 54
5.2.2 Taille maximale de la file d'attente dans le tampon d'entrée 56
5.2.3 Taille de la file d'attente au tampon de sortie à la fin du bloc 56
5.2.4 Temps de calcul 58
5.2.5 Temps d'attente 60
5.2.6 Nombre de calculs 60
5.2.6.1 Cycles positifs 60
5.2.6.2 Cycles négatifs 62
5.2.6.3 Cycles de sous-piles 62
5.2.7 Nombre d'accès à la pile 64
5.2.8 Nombre d'erreurs 64

6. RÉSULTATS DES TESTS 68

6.1 Simulation des séquences 68

6.2 Paramètres des tests 70

6.3 Résultats 71

7. CONCLUSIONS 82

RÉFÉRENCES 85
ANNEXE 87

LISTE DES FIGURES

		page
1.1	Modèle de base d'un système de communication numérique avec correction d'erreurs	4
1.2	Courbes de performance de plusieurs systèmes de codage	4
2.1	Codeur convolutionnel K = 3, R = ½	8
2.3	Treillis correspondant à l'arbre de la Figure 2.2	8
3.1	Schéma d'un décodeur séquentiel utilisant l'algorithme à pile	12
3.2	Structure des données de la pile	16
3.3	Illustration de la procédure du fanion	19
4.1	Organigramme de l'algorithme à pile implanté dans la réalisation matérielle du décodeur séquentiel	30
4.2	Schéma synoptique de l'appareil	32
4.3	Structure générale du décodeur	34
4.4	Diagramme bloc du matériel de décodage	35
4.5	Structure générale d'une machine de Mealy	37
4.6	Structure du micro-séquenceur	37
4.7	Schéma du micro-séquenceur	38
4.8	Schéma du calculateur des noeuds survivants et mémoire SEQS	41
4.9	Schéma bloc du module SUIV	42
4.10	Schéma bloc du module PILE	42
4.11	Schéma bloc du module SPIL	42
4.12	Schéma du récupérateur de séquence décodée	44
4.13	Schéma bloc d'une pile LIFO du module MESS	45
4.14	Schéma bloc du module MESS	45
4.15	Schéma général de l'organisation du tampon d'entrée	47
4.16	Schéma général de l'organisation du tampon de sortie	50
5.1	Schéma général du système de cueillette des statistiques	53
5.2	Schéma général pour la cueillette de la métrique accumulée finale	55
5.3	Schéma général pour la cueillette de la taille maximale de la file d'attente dans le tampon d'entrée	57
5.4	Schéma général pour la cueillette de la taille de la file d'attente dans le tampon de sortie	59
5.5	Schéma général pour la cueillette du temps de calcul	59
5.6	Schéma général pour la cueillette du temps d'attente	61
5.7	Schéma général pour la cueillette du nombre de cycles positifs par bloc	61
5.8	Schéma général pour la cueillette du nombre de cycles négatifs	63

5.9 Schéma général pour la cueillette du nombre de cycles de sous-pile 63
5.10 Schéma général pour la cueillette du nombre d'accès à la pile 65
5.11 Schéma général du banc d'essai pour la mesure du nombre d'erreurs 67

6.1 Cumulative de la métrique totale en fin de bloc 72
6.2 Cumulative du temps de calcul actif par bloc 74
6.3 Cumulative du nombre de cycles positifs par bloc 75
6.4 Cumulative du nombre de cycles négatifs par bloc 77
6.5 Cumulative du nombre de cycles de sous-piles par bloc 78
6.6 Cumulative de la somme de cycles de sous-piles et de cycles négatifs par bloc 80
6.7 Cumulative du nombre d'accès à la pile 81

LISTE DES TABLEAUX

4.1 Description sommaire des états du décodeur 31

AVANT-PROPOS

La réalisation matérielle du prototype de décodeur séquentiel dont il est question dans ce rapport a impliqué plusieurs personnes au cours des cinq dernières années. Le projet a pris naissance sous la forme d'une recherche de maîtrise, entreprise par M. Christian Morin. Ce projet ambitieux a dépassé largement le niveau et l'étendue des recherches habituelles de la maîtrise. Aussi, les travaux de réalisation se sont-ils poursuivis après la maîtrise de Christian Morin et ont impliqué, en plus de M. Morin, les personnes suivantes : M. Louis-André Poulin, Ing. M.Sc.A., M. Daniel Nielly, Tech.Dipl., M. Serge Ballard, Ing. En particulier, S. Ballard a aussi aidé à la rédaction de ce rapport en ce qui a trait à la description de la machine et des appareillages de cueillette de données. Qu'ils trouvent, chacun d'entre eux, l'expression de notre reconnaissance pour les longues heures de frustations passées à déchiffrer les manifestations bizarres d'une machine qui parfois s'ingéniait à ne pas vouloir fonctionner correctement et à nous induire en erreur.

Le support financier du projet provient de plusieurs sources : le Conseil de recherches en sciences naturelles et en génie du Canada grâce à des subventions individuelles et de Développement de la recherche, le programme de Formation de chercheurs et d'Actions concertées du Québec, la Direction de la recherche de l'École Polytechnique et, enfin, la Compagnie Marconi Canada de Montréal. Grâce à M. David Lee, chef ingénieur et à M. Pierre Desmarteau, Ingénieur de projet, la Compagnie Marconi Canada s'est montrée favorable au projet, et a subventionné dans le cadre de sa politique d'aide aux universités, le parachèvement de la machine et la mise en place de la procédure de tests. Nous leur exprimons notre vive reconnaissance et notre désir de voir continuer ce type de coopération et la relation privilégiée qui as prévalu avec cette compagnie de haute technologie tout au long du projet.

1. INTRODUCTION

L'usage de plus en plus répandu de techniques de transmissions numériques dans les télécommunications terrestres et par satellite conduit à l'utilisation grandissante de procédures de correction automatique des erreurs par codage de canal, qui sont puissantes, pratiques et fiables. Un problème important consiste à développer des techniques de codage et de décodage délivrant de faibles probabilités d'erreur avec des décodeurs de complexité acceptable. Aussi la tendance actuelle est moins axée sur l'analyse et le développement d'algorithmes complexes, mais porte davantage sur les problèmes de réalisations matérielles d'algorithmes puissants et efficaces. Le décodeur dont il est question dans ce rapport est un exemple d'une telle réalisation.

Le principe de base du codage consiste à protéger l'information numérique à transmettre par l'addition de symboles redondants appelés symboles de parité. L'ensemble des symboles d'information et de parité est transmis dans le canal de transmission, et en raison du bruit dans le canal, à la réception certains symboles sont reçus en erreur. L'objet du décodeur est de déterminer la position de ces erreurs et de les corriger, et dans une liaison avec codage, le décodeur est considérablement plus complexe que le codeur.

Le domaine du codage se divise en deux grandes classes appelées codage en bloc et codage convolutionnel. En codage en bloc, l'information est structurée en blocs de données numériques auxquels on ajoute les symboles de parité, alors qu'en codage convolutionnel, l'information se présente sous la forme de séquences de longueurs indéterminées, et typiquement les symboles de parité sont ajoutés à chacun des bits d'information. Ces deux techniques de codage sont nettement distinctes et ont été développées séparément. Ils s'analysent avec des outils mathématiques différents et trouvent des domaines d'appplications distincts. Le codage en bloc utilise

les outils de l'algèbre de polynômes, en particulier la théorie des corps de Galois, alors que le codage convolutionnel utilise une approche probabiliste tirée des principes de la Théorie de l'Information. En principe, le codage en bloc est préféré dans des applications où la structure de l'information à transmettre est structurée naturellement en blocs (systèmes de communications par paquets, systèmes vidéotex, etc.), alors que le codage convolutionnel est utilisé lorsque l'information à transmettre prend la forme de longues séquences ininterrompues (transmission de la parole, des images, des données, etc.), ou de blocs assez longs. Cependant en général, les codes convolutionnels se prêtent à une réalisation matérielle plus simple, et donc sont plus largement utilisés que les codes en blocs, surtout pour les communications numériques satellites.

1.1 Plan du rapport

Après une introduction au modèle d'un système de communication numérique, et aux avantages à utiliser le codage correcteur d'erreur, les notions de codage convolutionnel et de décodage probabiliste sont introduits au Chapitre 2. Le décodage séquentiel et, en particulier l'algorithme à pile, fait l'objet du Chapitre 3. Après un examen des caractéristiques essentielles de l'algorithme, la structure de données de la pile est présentée et les particularités de l'effort de calcul du décodeur séquentiel introduites. La description de la réalisation matérielle du prototype de recherche est fournie au Chapitre 4. Cette description ne fournit que les grandes lignes de la réalisation. Des détails plus complets sur les circuits se trouvent dans le mémoire de thèse de Ch. Morin [14]. L'examen des procédures de test de la machine est décrit en détail dans le Chapitre 5 et, enfin, les résultats de tests de performance effectués avec 250 blocs pour des valeurs de E_b/N_0, variant de 5.0 dB à 7.5 dB par incrément de 0.5 dB, sont fournis au Chapitre 6.

1.2 Modèle d'un système de communication numérique

Dans un système de communication numérique utilisant le codage correcteur d'erreurs ("Forward Error Control" ou FEC) tel que dessiné à la Figure 1.1, la source produit des symboles d'information binaire au rythme de R_s bits/s. Ces symboles d'information sont encodés par l'adjonction de symboles de redondance, et la sortie du codeur est une autre séquence binaire de débit R_c symboles/s. Le taux de codage R exprimé en bits/symbole est donné par le rapport $R = R_s/R_c$. Comme $R < 1$, alors le débit dans le canal R_c est supérieur au débit délivré par la source R_s. Par conséquent, l'introduction du codage correcteur d'erreur conduit à une expansion de largeur de bande de transmission.

Au récepteur, soit P la puissance du signal utile reçu et soit N_o la densité spectrale du bruit qui l'accompagne (en général considéré comme étant blanc et gaussien). Le rapport signal à bruit par bit d'information est donné par $E_b/N_o = P/(N_o R_s)$, et sert de facteur de mérite pour comparer les performances de différents systèmes de modulation et codage. Il est bien évident qu'un système qui peut réduire la valeur de E_b/N_o requise pour une probabilité d'erreur donnée permet un accroissement du débit R_s ou une réduction de la puissance de transmission, ou une combinaison des deux. Le problème de base peut donc se résumer à déterminer le système qui fournira une performance d'erreur donnée avec la plus faible valeur de E_b/N_o.

La Figure 1.2 montre les courbes de performance de plusieurs systèmes de codage utilisant une modulation de type PSK parfaite. Le gain de codage G d'un système de codage est égal à la différence en dB des valeurs de E_b/N_o requises pour une probabilité d'erreur donnée entre ce système de codage et la modulation PSK cohérente sans codage. Par exemple, au taux d'erreur de 10^{-5}, le codage en bloc BCH(128,112) donne un gain de codage de 2 dB, alors que le décodage de Viberbi à quantification pondérée ($K=7$, $R=\frac{1}{2}$) permet un gain de codage égal à 5.0 dB. On peut voir aussi qu'un décodeur séquentiel avec quantification dure permet un gain de codage égal à

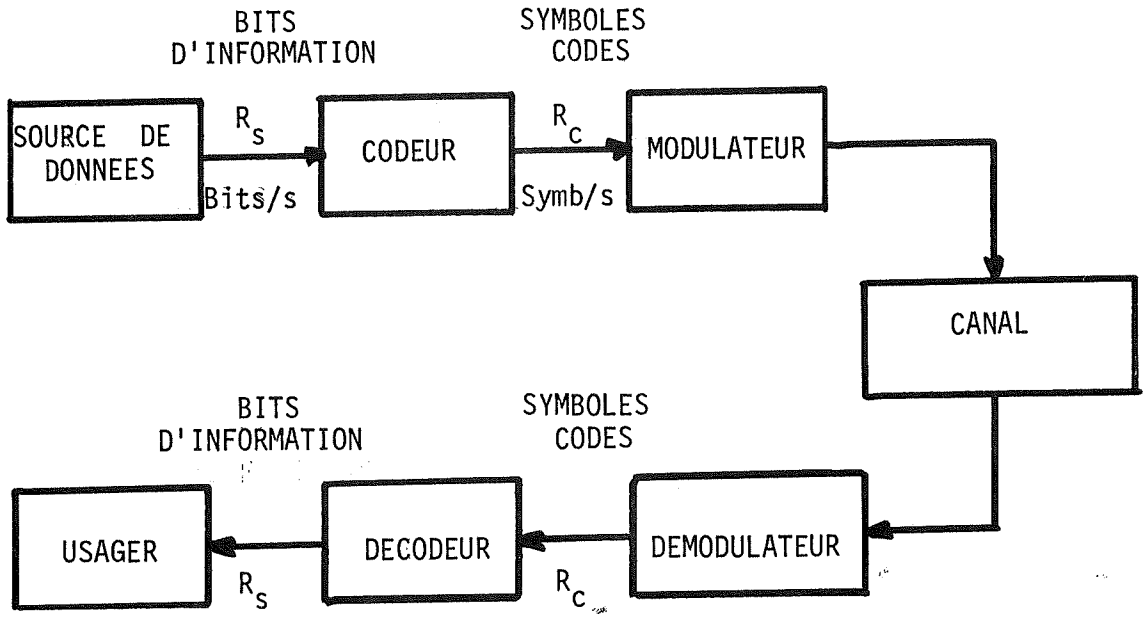


Figure 1.1 Modèle de base d'un système de communication numérique avec correction d'erreurs.

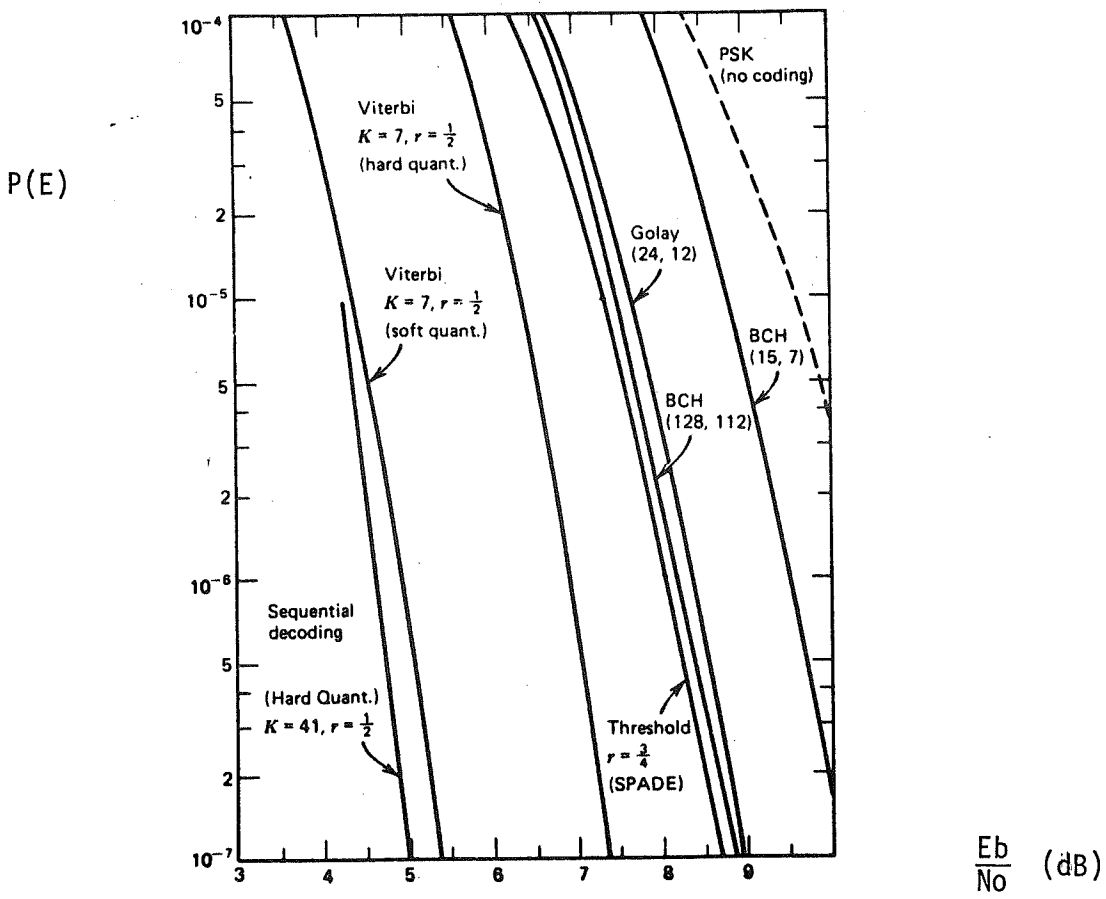


Figure 1.2 Courbes de performance de plusieurs systèmes de codage.

5.2 dB, en quantification pondérée à 8 niveaux ou 3 bits, le gain de codage peut atteindre 7 à 8 dB. D'un point de vue pratique, ce gain de 5.2 dB peut être traduit soit par une réduction de 5.2 dB de la puissance de transmission du système de transmission, soit par une augmentation de la vitesse de transmission des données non codées par un facteur de $10^{0.52} = 3.3$. Dépendant des applications, chacune de ces alternatives peut s'avérer particulièrement intéressante pour améliorer le design d'un système.

Ces notions de base étant établies, les principes du codage convolutionnel sont brièvement introduits au chapitre suivant avant d'examiner en détail le décodage séquentiel.

2. CODAGE CONVOLUTIONNEL ET DÉCODAGE PROBABILISTE

Dans les canaux de communication dits "sans mémoire", c.a.d. où le bruit est essentiellement blanc, les systèmes utilisant le codage convolutionnel avec décodage probabiliste sont parmi les plus intéressants tant du point de vue de leur performance d'erreur que du point de vue de leur réalisation et implantation matérielle : ils peuvent fournir des gains de codage appréciables tout en se prêtant à une réalisation de complexité raisonnable.

Le décodage probabiliste comprend un ensemble de techniques où le message décodé est obtenu par des procédures probabilistes plutôt que par des opérations algébriques fixes, et où les codes utilisés n'ont pas, en principe, à satisfaire à une structure algébrique particulière comme pour les codes en blocs. Les deux principales techniques de décodage probabiliste des codes convolutionnels sont le décodage séquentiel [1] et le décodage de Viterbi [2]. Chacune de ces techniques consiste à trouver un chemin particulier (le message transmis) dans un graphe orienté où on assigne aux branches des "métriques" ou valeurs de vraisemblance entre les données reçues du canal de transmission et les données qui auraient pu être transmises. L'objectif général du décodeur est donc de déterminer le chemin ayant la métrique totale maximum, et ce avec un minimum d'effort et un maximum de fiabilité. Ce chemin trouvé par le décodeur est la séquence décodée. Les techniques de décodage séquentiel et de décodage de Viterbi sont nettement différentes et trouvent des domaines d'application distincts. Etant donné les valeurs importantes de gain de codage permises, ces techniques de décodage s'avèrent particulièrement intéressantes dans les liaisons par satellite où chaque décibel d'énergie de transmission est extrêmement coûteux [3]. Dans ce rapport, nous porterons notre attention principalement sur le décodage séquentiel et sur la réalisation matérielle d'un prototype de recherche d'un décodeur séquentiel rapide. Cependant, avant d'examiner

plus en détail le décodage séquentiel, le codage convolutionnel et la structure des codes convolutionnels sont brièvement décrits ci-dessous. Pour une étude plus approfondie du codage convolutionnel, on pourra considérer par exemple la référence [3].

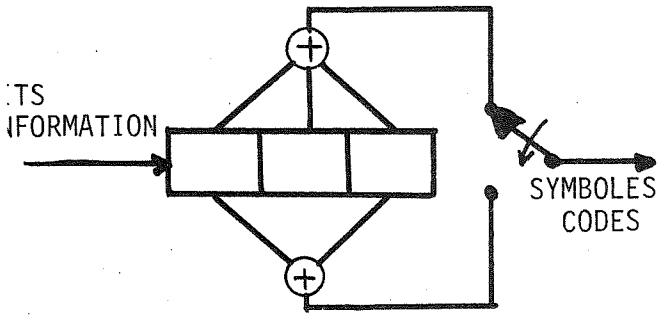
2.1 Structure des codes convolutionnels

Un codeur convolutionnel de taux de codage R = 1/V est représenté par une machine linéaire à états finis composée d'un registre à décalage de K cellules, V additionneurs modulo-2 connectés à certaines cellules du registre à décalage, et d'un commutateur qui balaye les V additionneurs modulo-2. L'ensemble des connexions entre le registre à décalage et les additionneurs modulo-2 spécifie le code. Par exemple, un codeur convolutionnel K = 3, R = ½ est montré à la Figure 2.1.

Un codeur convolutionnel fonctionne comme suit : les bits d'information sont introduits par la gauche, un bit à la fois, et après chaque décalage, les additionneurs modulo-2 sont échantillonnés en séquence par le commutateur, fournissant ainsi V symboles codés qui sont modulés et transmis dans le canal. Le taux de codage est donc R = 1/V. Pour ces codeurs binaires simples, la longueur K du registre à décalage s'appelle la longueur de contrainte du code. Un codeur peut être facilement généralisé, et admettre non 1 mais U bits à la fois dans le codeur, avec U < V. Le taux de codage devient alors R = U/V, et la longueur de contrainte devient un multiple de U, soit K = kU.

2.1.1 Arbre et treillis

Considérant seulement des codes convolutionnels de taux R = 1/V et de longueur de contrainte K, à chaque bit d'information il y correspond 2 branches d'un arbre portant chacune V symboles codés. L'extrémité de chaque branche est un noeud caractérisé par un état du codeur. L'état du codeur est le contenu des (K-1) premières cellules du registre à décalage, et donc le nombre d'états distincts est égal à $2^{(K-1)}$.



jure 2.1 Codeur convolutionnel
K = 3, R = $\frac{1}{2}$

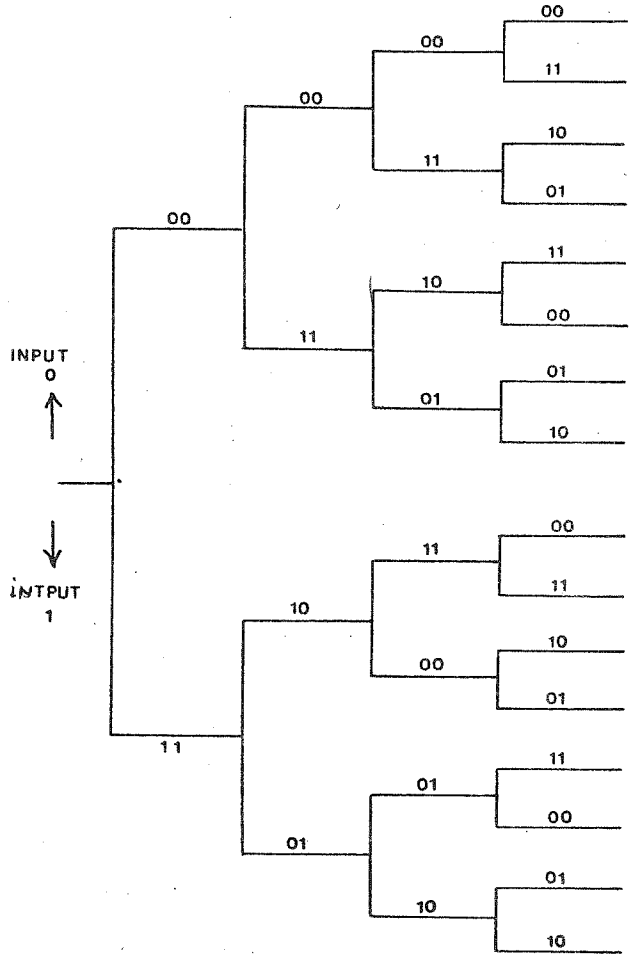


Figure 2.2 Arbre d'encodage du codeur de la Figure 2.1

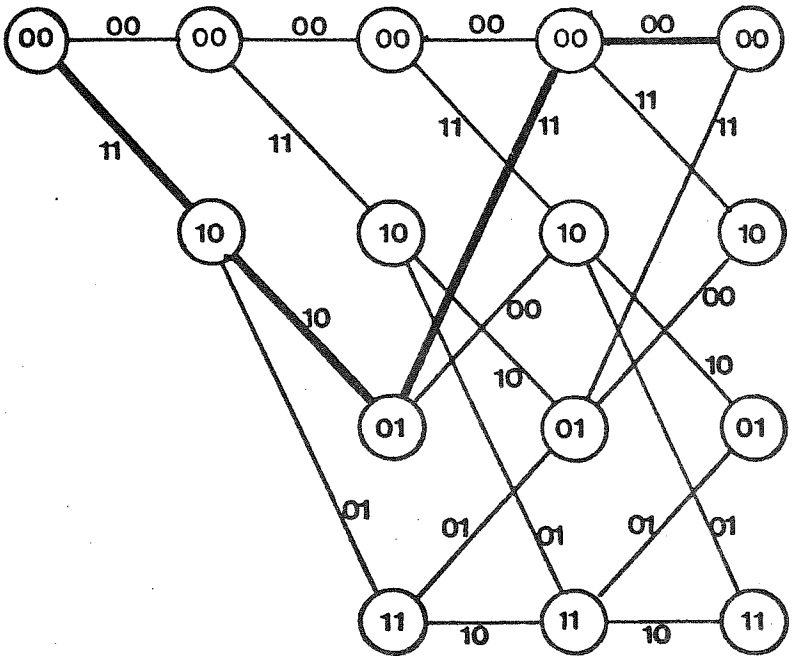


Figure 2.3 Treillis correspondant à l'arbre de la Figure 2.2

Un chemin dans l'arbre est spécifié par la séquence d'information qui est entrée dans le codeur et deux chemins reconvergent (i.e. ont le même état terminal) si leurs (K-1) derniers bits d'information sont identiques [3]. Au-delà d'une profondeur égale à K, l'arbre d'encodage contient donc une énorme redondance qui peut être éliminée en ne gardant qu'un seul chemin au-delà de chaque noeud de reconvergence. L'arbre devient alors un treillis ayant 2^{K-1} états, et pour une séquence d'information de longueur L bits, les chemins dans l'arbre ou le treillis ont donc une longueur maximale égale à L branches. Un exemple d'arbre et de treillis correspondant au codeur de la Figure 2.1 est donné aux Figures 2.2 et 2.3 respectivement.

Les notions de chemin, arbre et treillis sont essentielles à la compréhension du codage et décodage des codes convolutionnels. La séquence d'information étant représentée par un chemin (le chemin correct), la fonction de décodage consiste donc, connaissant la séquence reçue, de trouver le chemin dans l'arbre ou le treillis qui soit le plus "vraisemblable", c.a.d. qui "ressemble" le plus à la séquence reçue. Le décodage séquentiel dont il est question dans le reste du rapport est une des techniques parmi les plus puissantes et les plus efficaces pour trouver ce chemin le plus vraisemblable.

3. DÉCODAGE SÉQUENTIEL

Un décodeur séquentiel utilise la structure en arbre du code et n'explore, un chemin à la fois, que la partie de l'arbre qui paraît être la plus vraisemblable sans explorer l'arbre entier. C'est donc une procédure sous-optimale. Dans un canal sans mémoire, la fonction de vraisemblance utilisée, appelée aussi "métrique", est cumulative le long des branches d'un même chemin. Elle tend à croître le long du chemin correct et tend à décroître le long de tous les chemins incorrects. L'objectif du décodeur est donc d'explorer l'arbre d'encodage le long du chemin ayant la métrique la plus élevée parmi tous les chemins explorés. Par contre, un décodeur de Viterbi utilise la structure en treillis du code et examine tous les chemins qui auraient pu être transmis. En conséquence, l'effort de décodage en nombre de calculs effectués, par bit décodé, est constant mais en général élevé pour le décodeur de Viterbi, alors qu'en décodage séquentiel cet effort est en moyenne très faible mais aussi très variable, avec une distribution de type Pareto, c'est-à-dire :

$$P(C \geq N) \approx \ell N^{-\alpha}, \quad N \geq 1 \qquad (3.1)$$

où ℓ et α sont des constantes qui ne dépendent que du taux de codage et du canal. L'expression (3.1) indique que la distribution du nombre de calculs suit une décroissance algébrique et non pas exponentielle.

Cette variabilité de calcul est l'inconvénient principal du décodage séquentiel alors que le grand nombre de calculs est l'inconvénient du décodeur de Viterbi. La variabilité de l'effort de calcul nécessite l'utilisation d'un tampon à l'entrée du décodeur afin d'y stocker les branches reçues en attente d'être décodées. Le débordement de ce tampon constitue un événement d'erreur catastrophique entraînant un grand nombre de bits en erreur. Il est donc très important de réduire cette variabilité de l'effort de calcul, et certaines procédures ont été élaborées à cette fin [4].

3.1 Algorithme à pile

L'algorithme de décodage séquentiel dont il est question dans cette étude est l'algorithme à pile de ZIGANGIROV et JELINEK (Z-J) [3]. Cet algorithme utilise une pile pour stocker toutes les caractéristiques des chemins explorés, et un tampon d'entrée pour stocker les séquences reçues en attente d'être décodées. Un schéma de principe du décodeur est montré à la Figure 3.1. La pile est une liste ordonnée où sont stockés les chemins explorés par ordre décroissant de leur métrique. Le sommet de la pile contient le chemin ayant la métrique maximum courante; ce chemin est donc celui qui sera prolongé. L'algorithme a donc pour objet de déterminer à chaque étape le sommet de la pile et d'en faire le prolongement. Il se compose des 3 étapes suivantes :

1. Calcul des métriques des deux chemins issus du sommet de la pile et insertion dans la pile de ces deux chemins.
2. Elimination du sommet qui vient d'être prolongé.
3. Détermination du nouveau sommet. Si c'est le noeud terminal, stop. Sinon retour à 1.

Lorsque l'algorithme arrête, le sommet de la pile est le noeud terminal du chemin décodé, qui est alors facilement récupéré.

Bien que très simple, cet algorithme n'est pas pratique car le temps nécessaire à la mise en ordre exacte de la pile est beaucoup trop élevé. Cette difficulté est contournée en effectuant une mise en ordre approximative : les noeuds explorés sont insérés aléatoirement dans des sous-piles de la façon suivante :

un noeud $\underline{U}$ de métrique $\Gamma^{(\underline{U})}$ est inséré au hasard dans la sous-pile Q si

$$QH \leqslant \Gamma^{(\underline{U})} \leqslant (Q + 1)\, H \qquad (3.2)$$

où H est une valeur arbitraire.

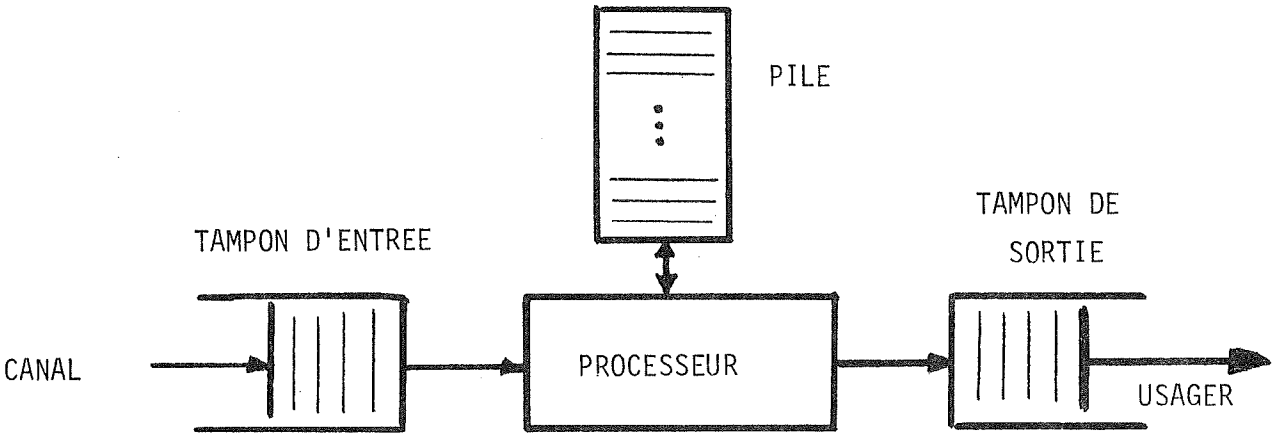


Figure 3.1 Schéma bloc d'un décodeur séquentiel utilisant l'algorithme à pile.

Avec cette modification, la recherche du sommet de la pile est réduite à la recherche de la sous-pile maximum non vide, et le chemin prolongé est choisi au hasard dans cette sous-pile, habituellement selon la procédure dernier-entré-premier-sorti (LIFO). Cette modification de l'algorithme facilite considérablement la procédure de recherche du noeud à prolonger et rend l'algorithme à pile applicable et pratique.

Un décodeur séquentiel pratique tient compte du délai de décodage variable qui découle de la variabilité de l'effort de décodage en utilisant un tampon d'entrée et un tampon de sortie (voir Figure 3.1). Le tampon de sortie régularise le débit de sortie des séquences décodées alors que le tampon d'entrée sert à stocker les données provenant du canal et qui attendent d'être décodées. Aussi un problème important concerne le débordement de ces tampons. On peut montrer que quelle que soit sa taille, il existe une probabilité non nulle pour que le tampon d'entrée déborde entraînant une perte de données et une rupture du lien de communication. L'analyse des débordements et des procédures de redémarrage du système sont parmi les problèmes importants de décodage séquentiel [5][6].

Afin de réduire les conséquences d'un débordement et d'une rupture du lien de communication, les données sont généralement organisées en "blocs" comportant quelque 500 à 2000 branches, chaque bloc se terminant par une séquence connue appelée "queue du message". La queue de longueur égale à (K-1) permet de remettre à zéro le registre à décodage du codeur local du décodeur, et de resynchroniser le système. En cas de débordement, le bloc en question est éliminé, le système est remis à zéro et le décodage peut se poursuivre pour les blocs suivants.

3.1.1 Structure des données de l'algorithme

Dans cette section nous donnons la structure de données utilisée pour simuler sur ordinateur l'algorithme à pile du décodeur séquentiel, en particulier l'organisation de la pile. Cette structure a servi de base à la réalisation matérielle décrite plus loin.

Une "entrée" dans la pile (c.a.d. un noeud) doit comporter trois entités essentielles à l'extension du noeud, à savoir la métrique accumulée du noeud, sa profondeur dans l'arbre, et l'état correspondant du codeur. De plus, on utilise deux pointeurs d'adresses qui servent à la mise en ordre de la pile et à la récupération de la séquence décodée. Toute cette information est stockée dans cinq mots contigus ayant tous la même adresse, et appelés respectivement VALUE, STATE, DEPTH, STAKPT et PATHP*. Un espace-mémoire de plusieurs milliers d'entrées doit être prévu. Enfin, un vecteur auxiliaire (la sous-pile) dénoté AUXPT est aussi utilisé pour trouver facilement la sous-pile maximum et le noeud à être prolongé tel qu'expliqué plus bas. (Dans la réalisation matérielle, AUXPT est dénoté SPIL).

Initialement, tout l'espace-mémoire de la pile et du vecteur auxiliaire AUXPT sont mis à zéro, et on assigne au noeud origine une métrique accumulée de valeur positive et arbitraire afin d'éviter la manipulation de métriques accumulés négatives. Les entrées successives dans la pile se font à des adresses consécutives et, une fois dans la pile, une entrée n'est jamais détruite. Toute entrée ayant une métrique accumulée Γ donnant une même valeur entière de $Q = \Gamma/H$ appartient à la même valeur de sous-pile Q.

D'un point de vue structurel, tous les noeuds de la pile appartenant à une même sous-pile Q_m, sont liés entre eux par leur chaîne de pointeurs STAKPT. Le pointeur STAKPT du premier noeud de la sous-pile Q_m contient la valeur 0, le pointeur STAKPT du second noeud de Q_m contient l'adresse du premier, le pointeur STAKPT du troisième noeud de Q_m contient l'adresse du second, et ainsi de suite jusqu'au dernier noeud de Q_m. L'adresse du dernier noeud de Q_m (c.a.d. le début de la chaîne) est contenue dans le Q_m-ième mot du vecteur AUXPT. En d'autres termes, toutes les entrées de la pile appartenant à une même sous-pile Q sont liées entre elles par les pointeurs STAKPT, et le début de la chaîne se trouve à la Qe adresse du vecteur auxiliaire AUXPT.

---

*Dans la réalisation matérielle du décodeur décrite au Chapitre 4, les éléments de la pile sont dénotés respectivement META, ETAT, PROF, SUIV et PERE.

Soit un noeud de métrique Γ et de sous-pile Q_m à stocker dans la pile à l'adresse courante N. Les valeurs des métriques, état, et profondeur, sont d'abord stockées dans leur registre respectif à l'adresse N. Ensuite, l'adresse contenue dans AUXPT(Q_m) est stockée dans le pointeur STAKPT(N) et la valeur N est stockée dans AUXPT(Q_m), ce qui préserve l'intégrité de la chaîne des entrées de sous-pile Q_m, Pour toute sous-pile vide Z, la valeur correspondante de AUXPT(Z) est zéro.

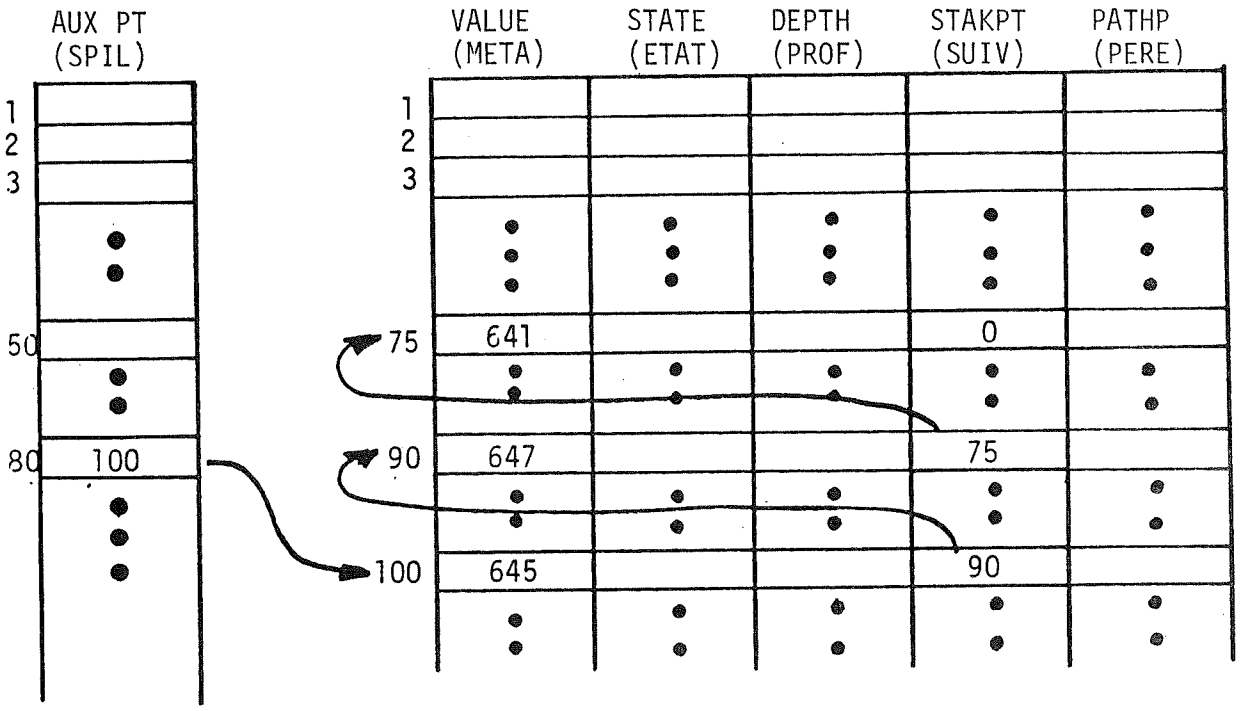
Lorsqu'un noeud doit être prolongé par l'algorithme, son "élimination" consiste simplement à le retirer de la chaîne des pointeurs STAKPT, ce qui a pour effet qu'il ne soit plus jamais prolongé. La sous-pile maximum Q_{top} étant toujours connue par l'algorithme, pour déterminer le noeud à prolonger, il suffit de trouver la première sous-pile non vide, en commençant bien sûr par Q_{top} et, de là, de trouver l'adresse du dernier noeud appartenant à cette sous-pile et inséré dans la pile. Naturellement, la chaîne de pointeurs STAKPT et le vecteur AUXPT doivent être proprement modifiés pour "éliminer" ce noeud prolongé.

A titre d'exemple, considérons les noeuds stockés dans la pile aux adresses 75, 90 et 100, et dont les métriques accumulées sont respectivement 641, 647 et 645. Soit H = 8 de sorte que les sous-piles correspondant à ces métriques sont toutes égales à 641/8 = 80. Ces trois entrées appartenant toutes à la même sous-pile 80, elles sont toutes liées par une même chaîne de pointeurs STAKPT dont le début réside dans AUXPT(80), comme indiqué ci-dessous et par le schéma de la Figure 3.2.

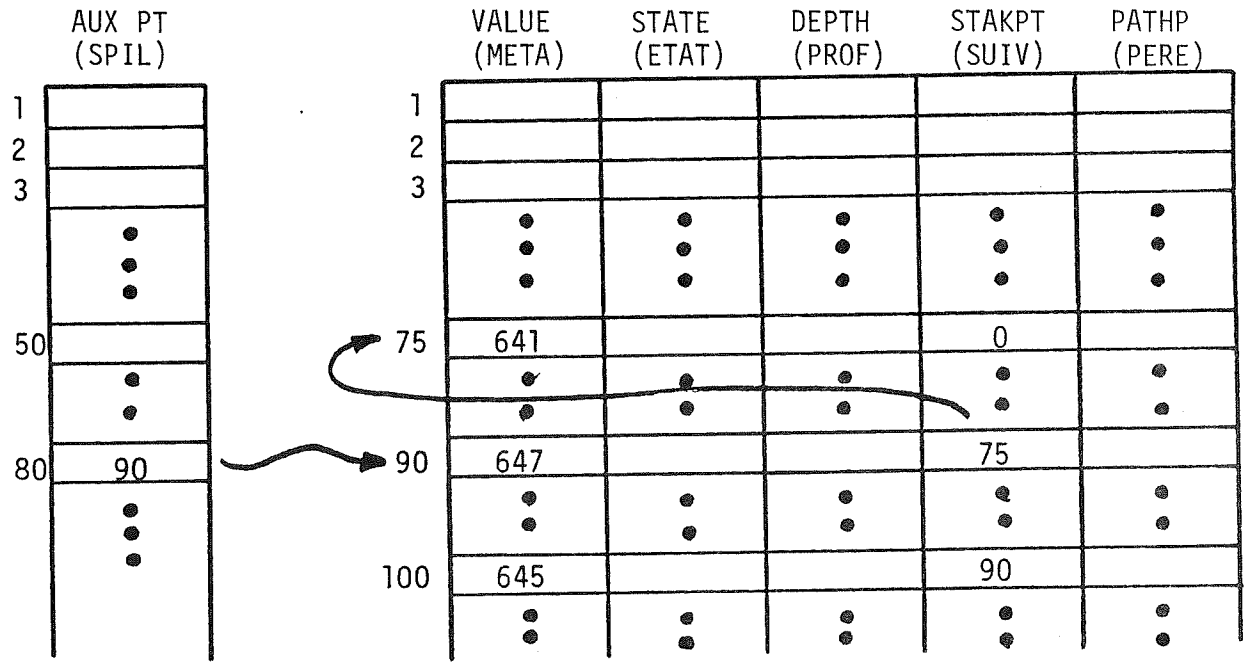
STAKPT(75) = 0
STAKPT(90) = 75
STAKPT(100) = 90

et

AUXPT(80) = 100.



(a)



(b)

Figure 3.2 Structure des données de la pile

(a) Insertion d'un noeud dans la pile
(b) Extention d'un noeud de la pile

Supposons à présent que la sous-pile maximum Q_{top} soit égale à 80. Par conséquent, selon la procédure LIFO décrite plus haut, c'est le dernier noeud de la sous-pile 80 qui doit être prolongé, c.a.d. celui dont l'adresse est contenue dans AUXPT(80), donc le noeud à l'adresse 100. Une fois ce noeud prolongé, on le retire de la chaîne des pointeurs en posant simplement

AUXPT(80) = STAKPT(100) = 90.

La Figure 3.2(b) illustre cette situation.

A la fin du décodage d'un bloc, il faut récupérer la séquence d'information décodée. Ceci est effectué à l'aide de la chaîne de pointeurs PATHP. Le pointeur PATHP de chaque noeud de la pile contient l'adresse du noeud "père" duquel il est issu. Ainsi, partant du dernier noeud décodé, par la chaîne de pointeurs PATHP, on remonte branche par branche au noeud origine de la séquence décodée.

3.1.1.2 Simplification : utilisation de fanion

A l'examen de l'algorithme, il apparaît que pour les codes binaires de taux R = 1/V (1/2, 1/3, ...), chaque extension de chemin implique un accès à AUXPT et deux nouvelles entrées dans la pile. Pour des codes de taux R = U/V, (e.g. 2/3, 3/4, ...), chaque extension implique 2^U entrées dans la pile. Comme une entrée dans la pile nécessite quelque 80 à 100 bits, naturellement on cherche toujours à éliminer toute entrée jugée inutile. Pour les codes de taux R = U/V, une procédure d'élimination de branches basée sur l'utilisation de seuils de rejets a été proposée par l'auteur et a donné d'excellents résultats [7][8].

Dans le cas des codes de taux 1/V, une procédure très simple, appelée procédure du fanion (en anglais "Flag"), permet de réduire considérablement le nombre d'entrées dans la pile. Cette procédure est basée sur

le principe de base que si une des 2 branches issues d'un noeud prolongé a une métrique positive, l'autre branche doit avoir une métrique négative. Par conséquent, si l'une des extensions d'un noeud qui vient d'être prolongé a une métrique positive, cette extension aura nécessairement la métrique accumulée maximum, c.a.d. sera au sommet de la pile et devra donc être prolongée à son tour. Il est donc tout à fait inutile de l'insérer dans la pile pour l'en extraire immédiatement après. Cependant, si cette branche qui n'a pas à été insérée dans la pile faisait partie du chemin correct, alors bien sûr on ne pourrait la récupérer. Donc la seule information utile à préserver pour ce noeud non inséré est le bit d'information qui y correspond. Sachant que ce bit d'information est le complément du bit d'information correspondant à la branche négative issue du même noeud que la branche positive, il suffit donc de l'indiquer par un indicateur binaire, ou fanion. Si l'indicateur est activé (fanion levé), on complémente le bit d'information de la branche, sinon on ne complémente pas. La structure de données de la pile contient donc une entité supplémentaire portant un bit d'information dans laquelle on indique si le fanion est levé ou baissé. La Figure 3.3 donne un exemple de l'utilisation du fanion qui est dénoté COMP.

D'un point de vue pratique, l'usage de ce fanion est-il rentable ? En effet, pour que ce soit rentable d'un point de vue espace-mémoire, il faut que l'économie du nombre d'entrées dans la pile compense l'augmentation de mémoire de la pile entière requise par le fanion. Une analyse théorique originale du comportement de la pile a montré que dépendant du rapport signal à bruit, plus de 70% des extensions du chemin correct donnent lieu à des métriques de branche positive [9], cette proportion augmentant avec le rapport signal à bruit. Sachant qu'une entrée dans la pile implique de 80 à 100 bits d'espace-mémoire, il est clair que l'usage du fanion conduit à une économie de mémoire appréciable. Une analyse détaillée de cette procédure a confirmé cette économie, même dans le cas d'un décodeur séquentiel utilisé en conjonction avec une procédure de retransmission [10] et tous les algorithmes à pile pour les codes de taux 1/V utilisent la procédure du fanion. Cette procédure a bien sûr été implantée dans notre réalisation matérielle du décodeur séquentiel.

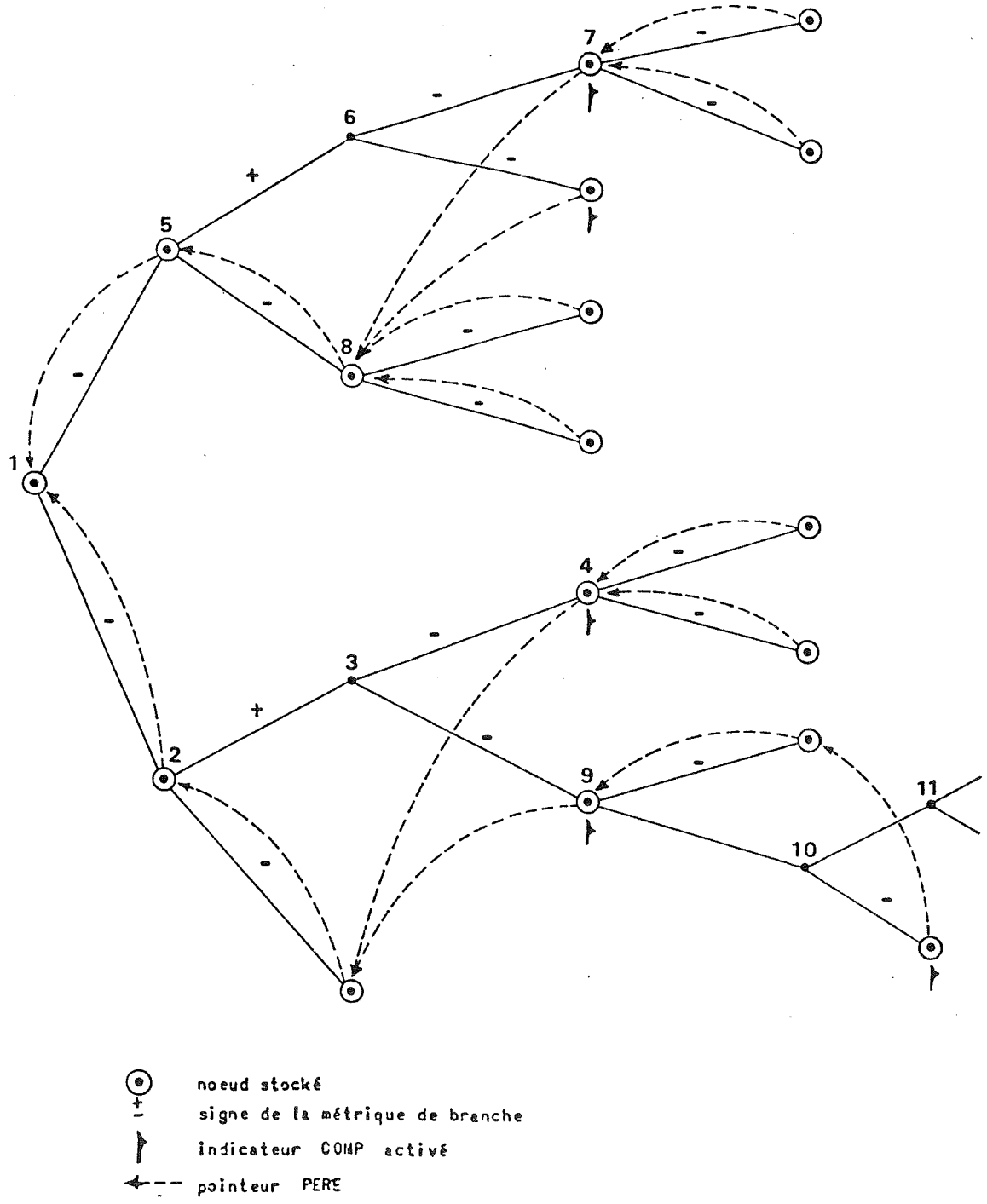


Figure 3.3 Illustration de la procédure du fanion.

3.2 Effort de calcul du décodeur séquentiel

Quel qu'en soit l'algorithme, le décodage séquentiel implique toujours la possibilité pour le décodeur de revenir en arrière dans l'arbre et de changer une décision antérieure, c.a.d. de prendre une autre alternative que celle qui semblait être la meilleure. D'un point de vue théorique et analytique, chaque opération de prolongation d'un chemin est définie comme étant un "calcul". Comme le nombre d'extensions effectué est aléatoire, le nombre de calculs effectué par bloc décodé est donc également aléatoire. Aussi, contrairement aux algorithmes de décodage déterministe, l'analyse du décodage séquentiel implique aussi bien la performance d'erreurs que la distribution de l'effort de calcul. Cette variabilité de l'effort de calcul est l'un des principaux inconvénients du décodage séquentiel et le problème de sa diminution a été l'objet d'une grande activité de recherche [4].

Une analyse théorique relativement complexe a montré que le nombre de calculs C effectué par bit décodé a une fonction de distribution cumulative qui asymptotiquement suit une loi Pareto, c.a.d. donnée par (3.1) et répétée ci-dessous :

$$P\ (C \geqslant N) \leqslant \ell\, N^{-\alpha} \quad , \quad N \geqslant 1 \qquad (3.2)$$

où ℓ et α dépendent du canal et du taux de codage R. L'exposant α est appelé l'exposant Pareto et est un des paramètres clef pour évaluer la performance et la conception de décodeur séquentiel.

La cumulative (3.2) indique que l'effort de calcul suit une décroissance algébrique et non pas exponentielle avec N, représentant ainsi un autre inconvénient du décodeur séquentiel. Par conséquent, il devient impératif de s'assurer que le nombre moyen de calculs par bit soit fini, et également de faire face, en pratique, au retard qui découle de la variabilité de l'effort du décodage.

Tel que mentionné plus haut, le retard de décodage se règle par l'utilisation de tampons d'entrée et de sortie. Quant à borner le nombre moyen de calculs, la réponse réside dans l'analyse théorique de la variabilité de calcul. Une analyse des moments de la distribution [11] a montré que si l'exposant Pareto α de (3.2) est inférieur à 2, la variance de l'effort de décodage diverge, et si $\alpha \leq 1$, la moyenne de cet effort diverge, c.a.d. le nombre moyen de calculs devient infini. Ceci se traduit en pratique par un décodage erratique, avec de très longues recherches de chemins dans l'arbre et des débordements des tampons et de la pile. Le taux de codage qui correspond à la valeur limite $\alpha = 1$ est appelé taux de coupure ("Computational Cut Off Rate") et est dénoté R_{comp}.

Ce taux R_{comp} ne dépend que du canal et se calcule facilement [3], mais d'un point de vue pratique, ce paramètre représente la limite extrême d'utilisation de décodeurs séquentiels. Aussi un important paramètre de design est le rapport R/R_{comp}, que l'on désire aussi près que possible de 1 mais sans l'atteindre. On peut noter en passant que la valeur de E_b/N_o correspondant à R_{comp} peut être calculée pour différents modèles de canaux et de niveaux de quantification, avec en général une amélioration de 2 dB lorsque la quantification du canal passe de 2 niveaux (1 bit) à 8 niveaux (3 bits) [3],[12].

L'analyse théorique des moments de l'effort de calcul en général, et du nombre moyen de calculs C_{AV} en particulier est réputée être difficile, et ne donne que des bornes asymptotiques et relativement peu serrées sur des ensembles de codes. D'un point de vue pratique, ces bornes sont donc très peu utiles pour des fins de design impliquant un code particulier. On doit donc avoir recours à de longues simulations sur ordinateur pour obtenir des valeurs utilisables. Cependant, utilisant une approche théorique totalement différente et basée sur les processus de ramification, une analyse récente de C_{AV} a donné des résultats considérablement plus précis [13]. De plus,

les résultats de cette analyse sont directement applicables car ils utilisent les paramètres du code ainsi que les valeurs particulières de métriques utilisées par le décodeur.

La nature Pareto de la distribution de l'effort de calcul a été confirmée pour toutes sortes de canaux. Un raisonnement simple permet d'expliquer un tel comportement. Lorsque le bruit dans le canal devient assez fort pour provoquer une chute de la métrique du chemin correct, le décodeur entre dans une phase de recherche arrière et prolonge les noeuds des chemins incorrects en conformité avec l'algorithme. Le nombre de ces chemins incorrects croît de façon exponentielle avec la profondeur de la chute de métrique du chemin correct. Cependant, pour des canaux sans mémoire, tout intervalle de bruit qui provoque une chute de métrique apparaît avec une probabilité qui décroît exponentiellement avec la durée de cet intervalle. Le comportement Pareto n'est rien d'autre que l'effet combiné de ces deux comportements exponentiels.

3.2.1 Problèmes de débordement

Tel que mentionné plus haut, la variabilité de l'effort de calcul entraîne l'utilisation d'un tampon d'entrée où les branches reçues du canal sont stockées en attendant d'être décodées. Il est clair qu'un débordement de ce tampon aurait des conséquences catastrophiques et donc il est impératif de prévoir une taille de tampon adéquate pour minimiser, voire éliminer un tel événement.

Cependant on montre que quelle que soit sa taille, il existe toujours une probabilité non nulle pour que le tampon d'entrée déborde. Cette probabilité est assez grande (nettement plus grande que la probabilité d'erreur) et conduit à des effacements (en anglais "erasures"). Ces effacements ne sont pas des erreurs mais sont considérés plutôt comme étant des incertitudes sur la valeur de certains bits.

Pour des séquences de longueur L bits, la probabilité de débordement du tampon d'entrée est approximée par

$$P \text{ (débordement)} \approx L(SB)^{-\alpha} \qquad (3.3)$$

où B est la taille du tampon, α est l'exposant Pareto et S est le gain de vitesse du décodeur. Ce gain de vitesse est le rapport entre le temps d'inter arrivée des bits du canal et le temps d'un calcul par le décodeur. L'expression (3.3) indique encore une distribution Pareto et une probabilité de débordement qui ne varie que lentement avec la taille du tampon, et donc très difficile à combattre. Aussi en pratique, on choisit des tampons d'entrée très grands, et on sectionne les séquences d'information en blocs de longueurs variant de 500 à 2000 ou 3000 bits. De plus, on prend toujours certaines procédures de recouvrement en cas de débordement. Ces procédures peuvent consister en une demande de retransmission des blocs qui sont sur le point de déborder [10], ou en un arrêt de décodage proprement dit et en une estimation de la séquence transmise. Ces procédures de recouvrement, qui peuvent varier selon les applications, sont une partie essentielle du décodage séquentiel et ne peuvent être ignorées dans une réalisation pratique.

4. RÉALISATION MATÉRIELLE DU DÉCODEUR SÉQUENTIEL

4.1 Historique

Une réalisation matérielle de prototype de décodeur séquentiel utilisant l'algorithme Z-J et exploitant les résultats de recherches théoriques et des simulations à l'ordinateur a été amorcée en 1977 dans le cadre du projet de recherche de M.Sc.A. de Christian Morin et poursuivie dans les années suivantes. En plus d'analyser les méthodes de traduire en matériel l'algorithme de décodage, l'objet de cette recherche était d'investiguer et de mieux comprendre certains éléments de la dynamique du décodage. Le prototype devait donc en être un de recherche et non d'utilisation industrielle, c.a.d. devait comporter un grand nombre de points de tests sur les paramètres importants du déroulement des opérations de décodage, en particulier ceux de l'effort de calcul. De plus, afin de mieux étudier les événements de débordement, les espaces mémoires de la pile et des tampons d'entrée et de sortie ont été surdimensionnés.

Une première approche de design basée sur microprocesseurs a été d'abord envisagée. Cependant, étant donné la complexité de l'algorithme et les exigences requises du prototype, cette approche s'est vite avérée inadéquate. Aussi, un design utilisant une structure cablée a donc été élaboré et réalisé. Ce projet qui dépassait largement les exigences d'un mémoire de maîtrise en sciences appliquées a donc été poursuivi au-delà de la maîtrise de Christian Morin pour sa réalisation finale, et une subvention de Marconi Canada Ltée en 1983-1984 a aidé à financer son parachèvement et à l'élaboration des procédures de tests.

Après avoir présenté les caractéristiques de design du décodeur, la description des éléments essentiels du décodeur sera donnée. Les détails précis du matériel ne sont pas fournis dans ce rapport mais sont tous contenus dans le mémoire final de Christian Morin [14].

4.2 Caractéristiques du décodeur

Code

Taux :	$R = \frac{1}{2}$
Longueur de contrainte :	$K \leq 24$

Décodeur (Algorithme Z-J quantifié)

Taille de la pile (nombre d'entrées) :	16 000
Mémoire de stockage d'un noeud :	77 bits
Longueur des blocs (branches) :	$\leq$ 1 024
Cycle de décodage élémentaire	= 375 ns
Vitesse de décodage estimée : calculs/s	5×10^5, 1×10^6
Dynamique des métriques de branches :	-256, +127
Dynamique des métriques totales :	-16384, +16383
Table des métriques :	par branche
Plage des sous-piles :	8

Entrées

Quantification :	2 à 8 niveaux
Tampon d'entrée :	64 000 branches
Débit :	$\leq 1 \times 10^6$ symboles/s
Synchronisation :	externe

Sorties

Débit :	$\leq$ 500 Kbits/s
Tampon de sortie :	64 000 bits
Synchronisation :	externe

Paramètres mesurés

Nombre maximum de bits dans le tampon d'entrée pour chaque bloc
Métrique totale à la fin de chaque bloc
Temps de calcul par bloc
Temps d'attente par bloc
Nombre d'accès à la pile par bloc
Nombre de cycles à métrique positive par bloc
Nombre de cycles à métriques négatives par bloc
Nombre de cycles en mode recherche (sous-piles)
Nombre de bits décodés en erreur (Prob. d'erreur)
Nombre moyen de calculs par bloc.

Performances anticipées

Gain de codage	5 à 8 dB
Probabilité d'erreur	$\leq 10^{-7}$

Ces différentes statistiques sont disponibles sous forme d'histogrammes et sont gérées par un micro-ordinateur AIM-65 incorporé au décodeur. Un programme externe au décodeur convertit ces histogrammes en distributions cumulatives disponibles sous forme de courbes. La collecte des statistiques sera explicitée en détail au Chapitre 5.

4.3 Description du décodeur séquentiel

4.3.1 Principes de base

Le fil directeur dans la conception du décodeur est la réalisation d'une machine permettant d'effectuer environ 1 million de calculs par seconde pour le décodage des codes convolutionnels de taux ½ et de longueurs de contrainte pouvant atteindre 24. Dans ce contexte, un calcul est l'ensemble des opérations requises entre deux extensions consécutives d'un noeud au sommet de la pile.

En général, dans la conception d'une machine numérique, plus une machine sera à "usage général", plus elle sera lente pour exécuter une tâche particulière. Les machines les plus lentes sont les plus polyvalentes et les plus chargées de logiciel, tandis que les machines les plus rapides sont des processeurs spécialisés à structure cablée.

La microprogrammation fait le lien entre le logiciel proprement dit et le matériel. En micro-programmation verticale, une instruction est exécutée par une suite de micro-instructions. Des mémoires rapides pour stocker ces micro-instructions sont requises afin d'accélérer la cadence des opérations. En micro-programmation horizontale, une micro-instruction est subdivisée en un certain nombre de champs qui contrôlent chacun une partie des opérations, permettant ainsi la commande simultanée de plusieurs opérations élémentaires.

La technique de "pipeline" permet d'accroître la vitesse d'exécution des micro-instructions. En effet, la micro-instruction en cours d'exécution est stockée dans un registre de pipelining ("pipeline register") contrôlant le ou les processeurs, et pendant cette exécution, la micro-instruction suivante est appelée (fetched) de la mémoire rapide. Une fois la micro-instruction en cours exécutée, la suivante peut être chargée dans le registre de pipelining rendant ainsi transparent l'accès à la mémoire du micro-programme. C'est cette technique qui a été utilisée dans la conception du prototype dont il est question dans ce rapport.

4.3.2 Conception du logiciel de décodage

Tel que mentionné plus haut, le décodeur utilise l'algorithme Z-J quantifié avec une pile structurée en sous-piles. Ainsi, pour localiser le noeud "sommet" de la pile, il suffit de déterminer la sous-pile non vide de niveau maximum et de là de trouver, si on le désire, le noeud de métrique maximum parmi tous les noeuds de la sous-pile.

Les éléments d'une entrée dans la pile sont dénotés de la façon suivante :

Métrique accumulée = META
Etat du codeur = ETAT
Profondeur dans l'arbre = PROF
Pointeurs d'adresse d'une même sous-pile = SUIV
Pointeur de récupération = PERE.

La mémoire auxiliaire où réside le début de chaque chaîne de pointeurs SUIV est dénotée SPIL.

La procédure du fanion a également été implantée afin d'économiser l'espace mémoire de la pile et de simplifier les opérations de décodage lorsqu'une des branches prolongées a une métrique positive. Tel qu'expliqué à la Section 3.1.1.2, seule la branche de métrique négative est stockée dans la pile et pour cette entrée, le fanion est levé. Ce marquage par fanion prend la forme d'un paramètre supplémentaire dénoté COMP qui constitue le 6e élément d'une entrée dans la pile.

De plus, afin de mieux utiliser la grande vitesse de calcul du décodeur, on exploite les périodes d'attente de nouveaux symboles dans le tampon d'entrée. On rend donc le décodeur "toujours actif" en profitant de la disponibilité temporaire de temps de calcul pour prolonger les chemins de plus grandes métriques après le chemin sommet. Une telle procédure permet de réduire la variabilité de l'effort de calcul au prix d'un nombre de calculs moyens plus grands. Cependant, en pratique cette procédure devrait être suivie avec beaucoup de prudence, car si le débit des symboles d'entrée devenait trop lent, le décodeur remplirait rapidement sa pile causant ainsi un débordement.

Une autre modification à l'algorithme de base et qui découle de la grande vitesse de décodage concerne la pile auxiliaire SPIL et la procédure de récupération des bits décodés. La récupération se fait après que le bloc ait été décodé, et consiste à remonter le chemin décodé à partir du dernier bit jusqu'au premier en suivant les pointeurs de chemins PERE. Afin de ne pas interrompre les opérations de décodage pendant cette récupération, deux ensembles de pointeurs PERE sont utilisés à tour de rôle dans le décodage des blocs successifs. Pendant qu'on récupère les bits sur un ensemble de PERE, l'autre ensemble est utilisé pour le décodage du bloc suivant.

Quant à la pile auxiliaire SPIL, on rappelle qu'elle contient les débuts de la chaîne des pointeurs SUIV qui lient ensemble tous les noeuds d'une même sous-pile. Au début du décodage, toutes les sous-piles sont vides. Par conséquent, toutes les cases de SPIL doivent être remises à zéro

avant le décodage d'un bloc. Cette opération risquant de ralentir la vitesse de décodage, tout comme les pointeurs PERE, on utilise deux ensembles de SPIL. Pendant que le décodeur fonctionne avec l'un, l'autre est remis à zéro par le matériel.

L'algorithme de décodage Z-J a été décomposé en 15 états selon l'organigramme donné à la Figure 4.1 où on reconnaît la partie décodage du sommet de la pile et la partie propre aux opérations de décodeur toujours actif (chemins auxiliaires). Une description sommaire de ces états extraite de [14] est fournie au Tableau 4.1 et le détail de chacune de ces opérations est donnée à l'Annexe A1.

Dans la section suivante, la conception du matériel est décrite de façon sommaire. Une description plus détaillée est fournie dans [14].

4.3.3 Conception du matériel

Le décodeur séquentiel rapide a une structure matérielle qui exploite au maximum le parallélisme des opérations. On peut spécifier des codes convolutionnels jusqu'à une longueur de contrainte de 24, appliqués à des blocs de longueur programmable inférieure ou égale à 1024. Un microprocesseur permet une interaction entre l'utilisateur et l'appareil lors de l'initialisation. Ce dernier sert également à la compilation de diverses statistiques et paramètres de décodage.

On distingue 3 blocs principaux dans la structure matérielle du décodeur (Figure 4.2). Un tampon d'entrée dénoté TENT sert à emmagasiner la séquence sous forme de symboles reçus. Un tampon de sortie filtre le débit des bits décodés et permet une synchronisation externe. Entre ces deux blocs réside le matériel de décodage proprement dit et qui est décrit plus loin.

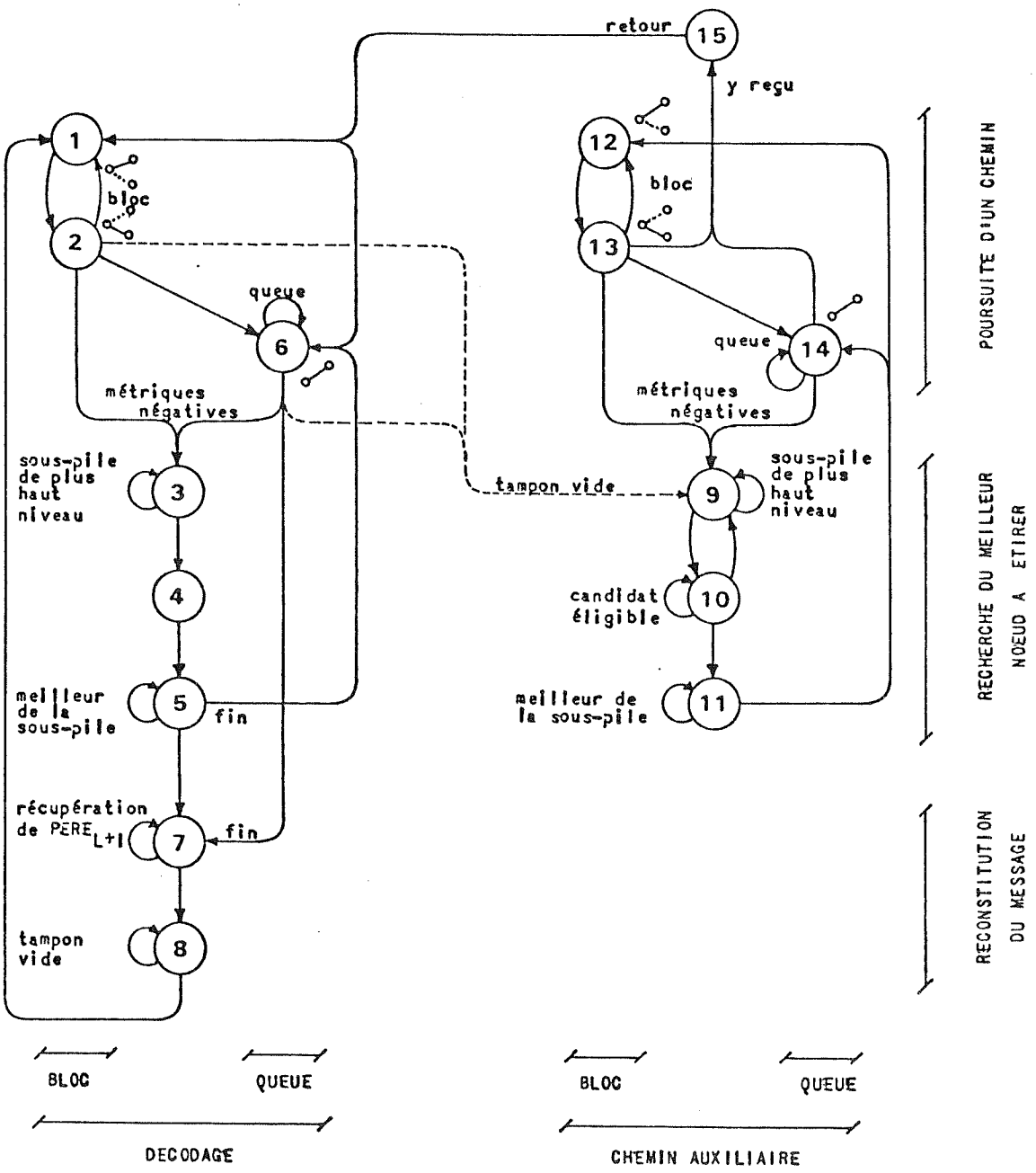


Figure 4.1 Organigramme de l'algorithme à pile implanté dans la réalisation matérielle du décodeur séquentiel.

ÉTAT	DESCRIPTION
1	Extension de la branche "0" dans le bloc
2	Extension de la branche "1" dans le bloc
3	Localisation de la sous-pile de plus haut niveau
4	Accès au noeud suivant dans la même sous-pile
5	Bouclage dans la sous-pile
6	Extension de la branche "0" dans la queue
7	Récupération du noeud terminal dans le bloc
8	Attente d'un symbole
9	Début de l'extension auxiliaire - sauvegarde des registres
10	Recherche d'un candidat à l'extension auxiliaire
11	Recherche du meilleur candidat
12	Extension de la branche "0" dans le bloc (auxiliaire)
13	Extension de la branche "1" dans le bloc (auxiliaire)
14	Extension de la branche "0" dans la queue (auxiliaire)
15	Restitution des informations sauvegardées

TABLEAU 4.1 Description sommaire des états du décodeur

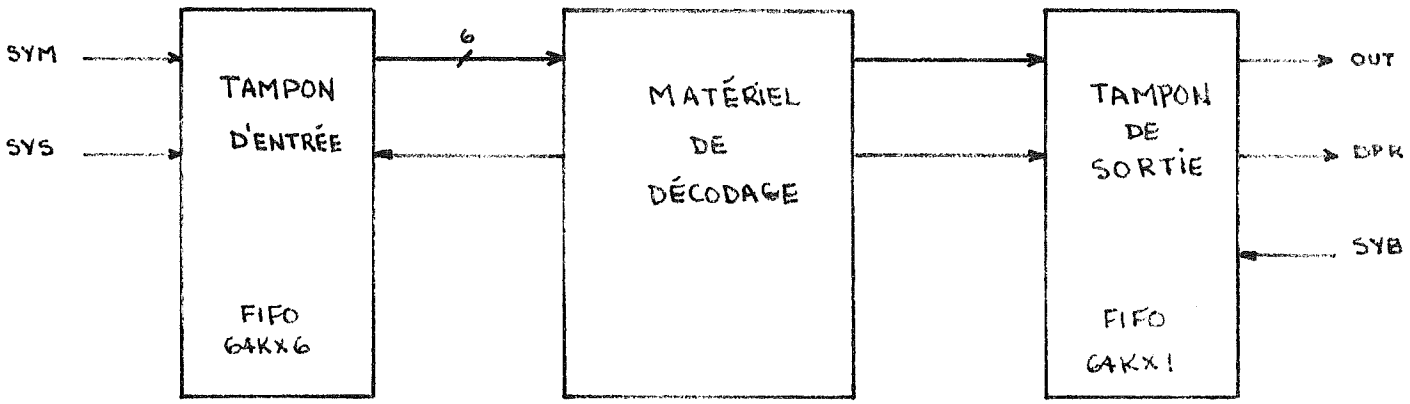


Figure 4.2 Schéma synoptique de l'appareil.

La Figure 4.3 montre sommairement la structure matérielle générale du décodeur. On y voit que les séquences reçues sont d'abord emmagasinées dans le tampon d'entrée (TENT) avant d'être dirigées vers le décodeur et la mémoire SEQS. Cette mémoire sert à conserver les séquences reçues en prévision d'un éventuel retour en arrière dans l'arbre.

Les paramètres du noeud à prolonger (profondeur dans l'arbre, métrique accumulée, etc.) sont stockés dans des registres de données. Ils alimentent une fonction logique combinatoire qui, connaissant la séquence codée correspondante, calcule les paramètres des deux noeuds suivants et les dépose dans la PILE. La mémoire SPIL conserve l'adresse du dernier noeud entré dans chaque sous-pile ce qui, avec l'aide de registres pointeurs SUIV, permet de localiser le meilleur noeud de la pile à prolonger à l'étape de calcul suivante (noeud sommet).

A la fin du traitement d'un bloc, le message est récupéré dans la pile et envoyé dans le tampon de sortie (TSOR) où il devient disponible pour l'usager.

Un microprocesseur gère tout le processus et permet à l'opérateur de modifier les paramètres de décodage (code, longueur des blocs, etc.) et de commander l'arrêt et le départ des opérations. Le microprocesseur agit sur le séquenceur. Celui-ci assure l'enchaînement ordonné des micro-instructions en fonction du résultat des tests effectués simultanément avec les opérations. Il libère en temps opportun les microcommandes à chacun des modules du matériel. La base de temps du séquenceur est fournie par une horloge multiphases qui prend sa référence d'un oscillateur stabilisé par un cristal de quartz.

Le matériel de décodage se divise en 5 blocs distincts. Ceux-ci ont été généralisés afin de mieux percevoir leur rôle. Il s'agit en l'occurrence du micro-séquenceur, du calculateur des survivants d'un noeud (noeuds suivants), de la pile, du récupérateur et du module de remise en ordre (Figure 4.4). Chacun de ces blocs est décrit ci-dessous.

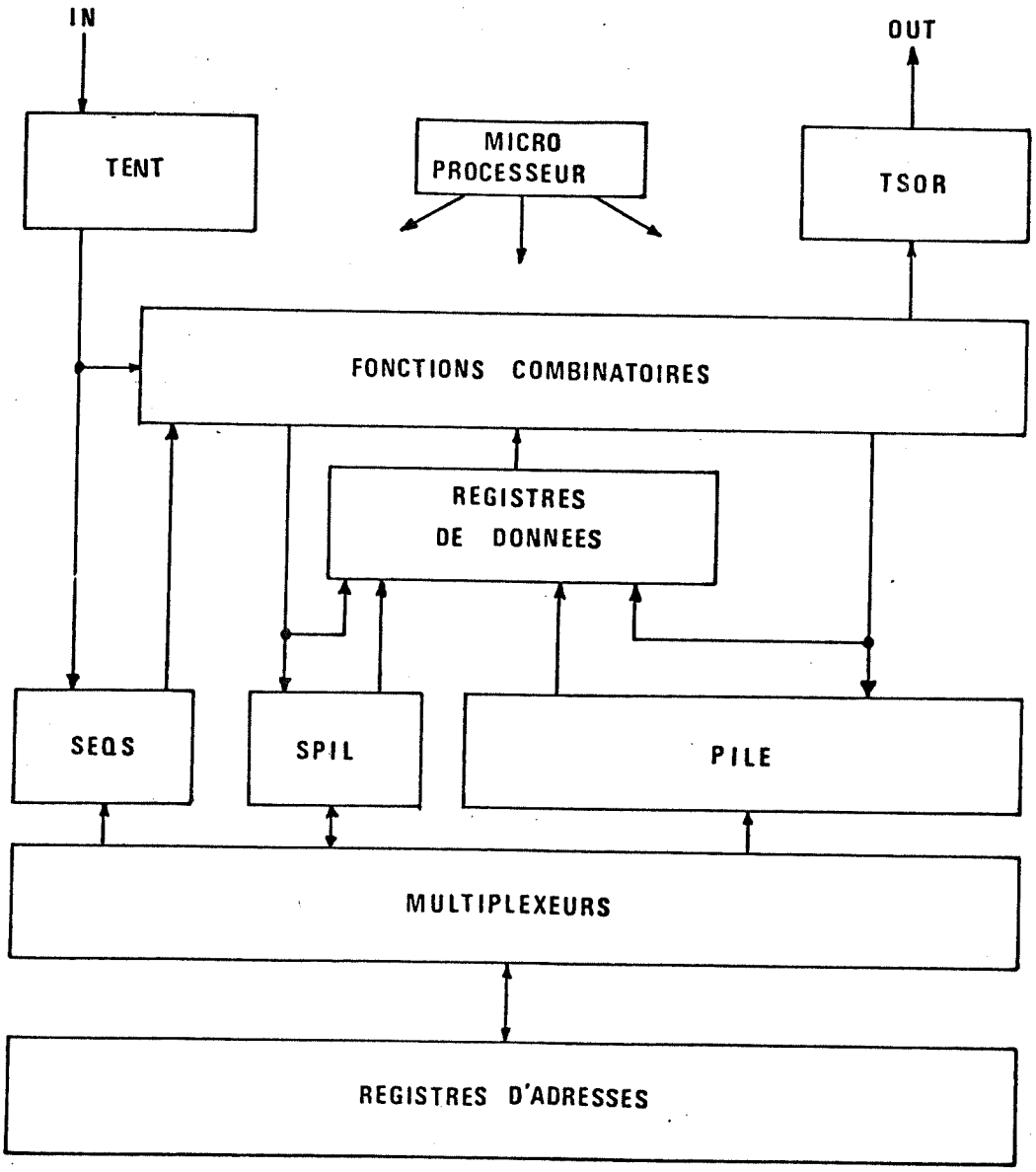


Figure 4.3 Structure générale du décodeur.

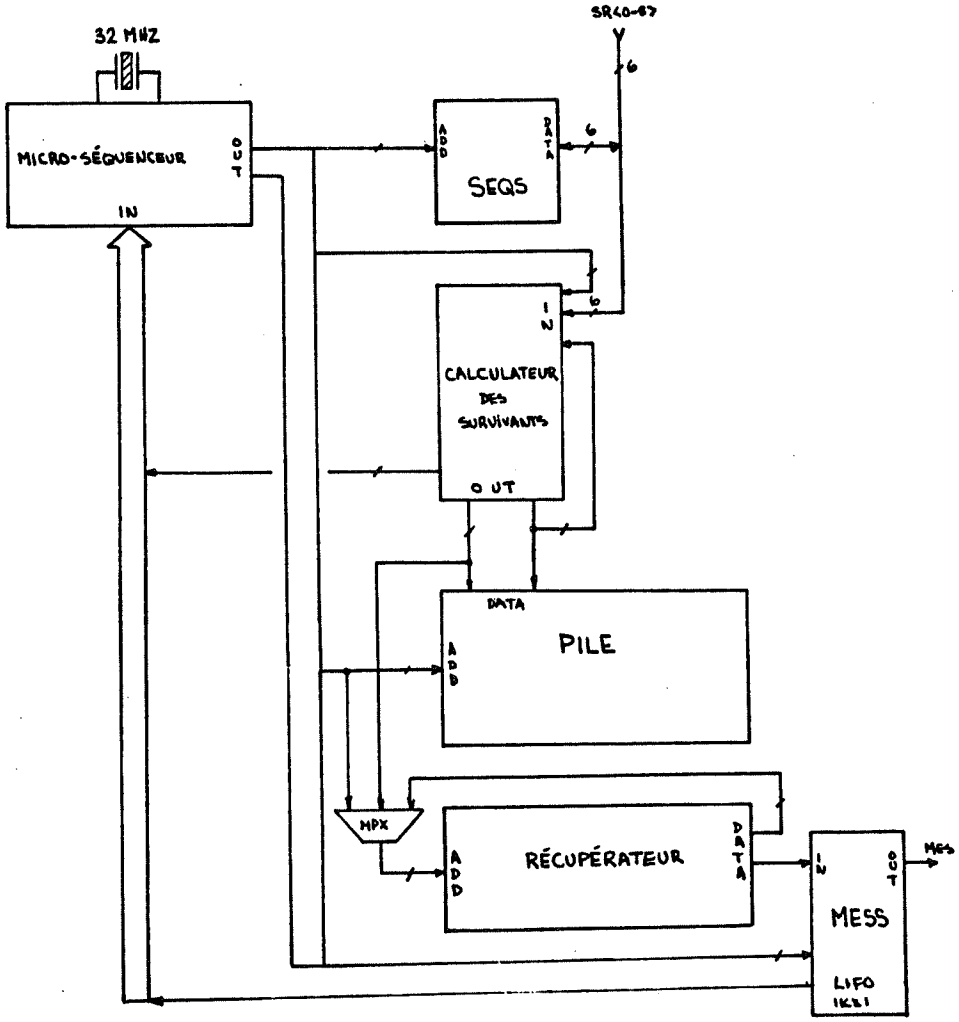


Figure 4.4 Diagramme bloc du matériel de décodage.

LE MICRO-SÉQUENCEUR

Le micro-séquenceur peut être décrit comme une machine de Mealy (Figure 4.5). Il génère un vecteur de microcommande S_1 étant donné un vecteur $\underline{e}$ de résultats de tests et la micro-instruction q_1 qu'il faut exécuter. L'organigramme fourni en annexe définit la fonction G de microcommande et la fonction F de branchement des micro-instructions. De plus, le micro-séquenceur a la responsabilité d'anticiper les adresses d'accès aux mémoires à partir de ses variables internes Y et du vecteur de test $\underline{e}$. On appelle S_2 le vecteur d'adresses anticipées et H la fonction combinatoire qui l'évalue (Figure 4.6). Lors de l'initialisation du décodeur, il faut placer les registres et les mémoires dans un état propice au déclenchement du microprogramme. Des commandes de forçage sont à cette fin ajoutées au micro-séquenceur. Un schéma global est donné à la Figure 4.7. Rappelons qu'un horloge multi-phases est nécessaire au transfert des informations dans les divers modules. L'oscillateur de 32 MHZ stabilisé par un cristal de quartz génère le signal de référence à partir duquel tous les autres seront synthétisés.

CALCULATEUR DES SURVIVANTS D'UN NOEUD

Ce calculateur est la fonction logique combinatoire qui accepte les paramètres d'un noeud comme entrée et fournit les paramètres d'un des noeuds suivants dans l'arbre comme sortie. Ainsi, les variables d'entrée sont fournies par :

- l'état du codeur DE < 0-22 >
- la métrique accumulée DM < 0-14 >
- la profondeur dans l'arbre DP < 0-9 >

D'autres informations essentielles au calcul des survivants sont aussi requises, à savoir :

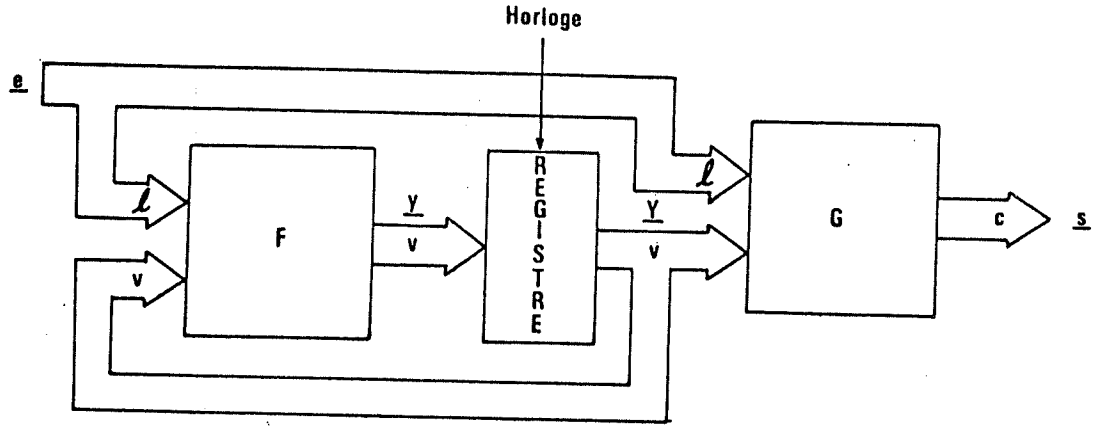


Figure 4.5 Structure générale d'une machine de Mealy.

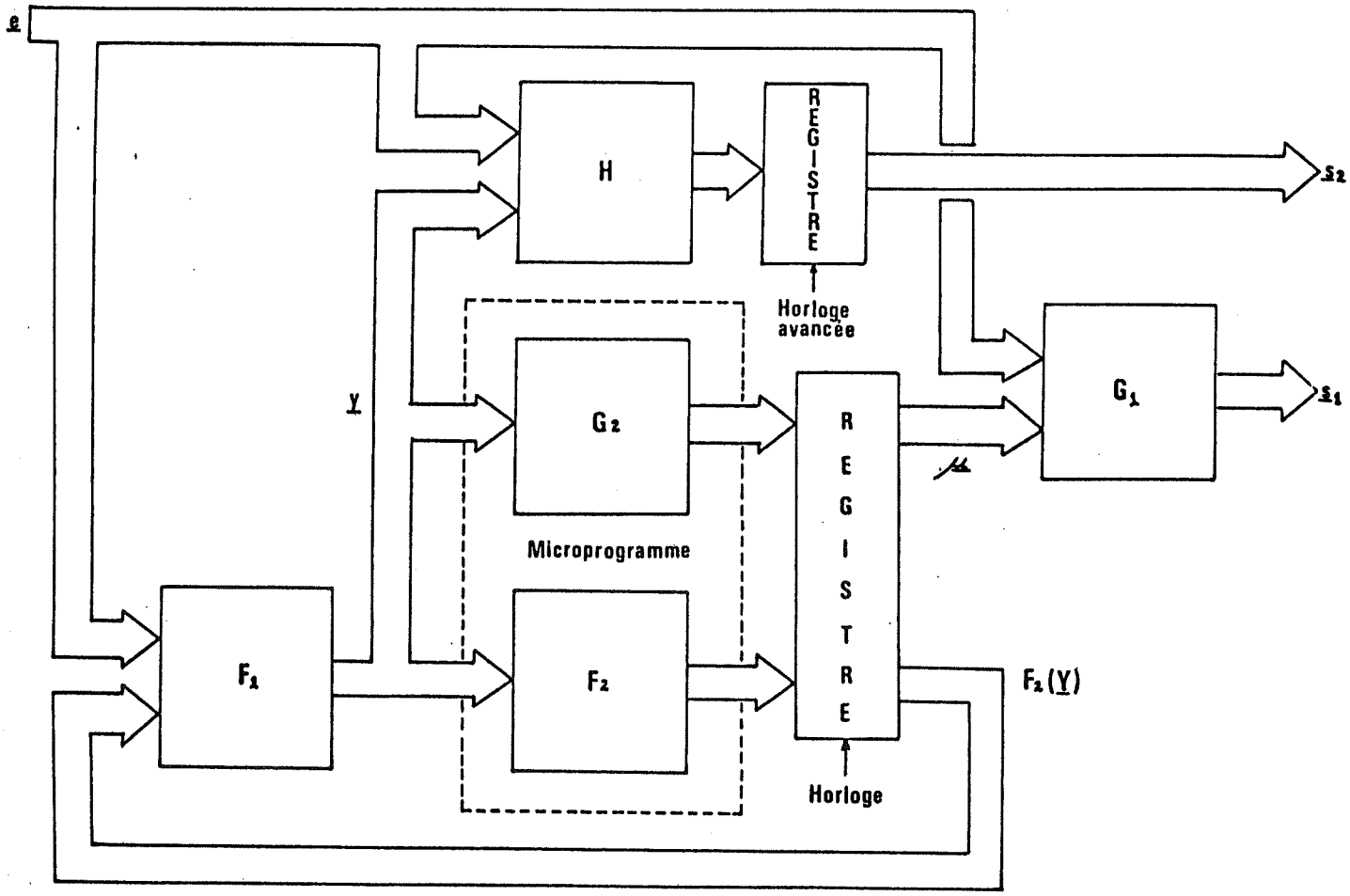


Figure 4.6 Structure du micro-séquenceur.

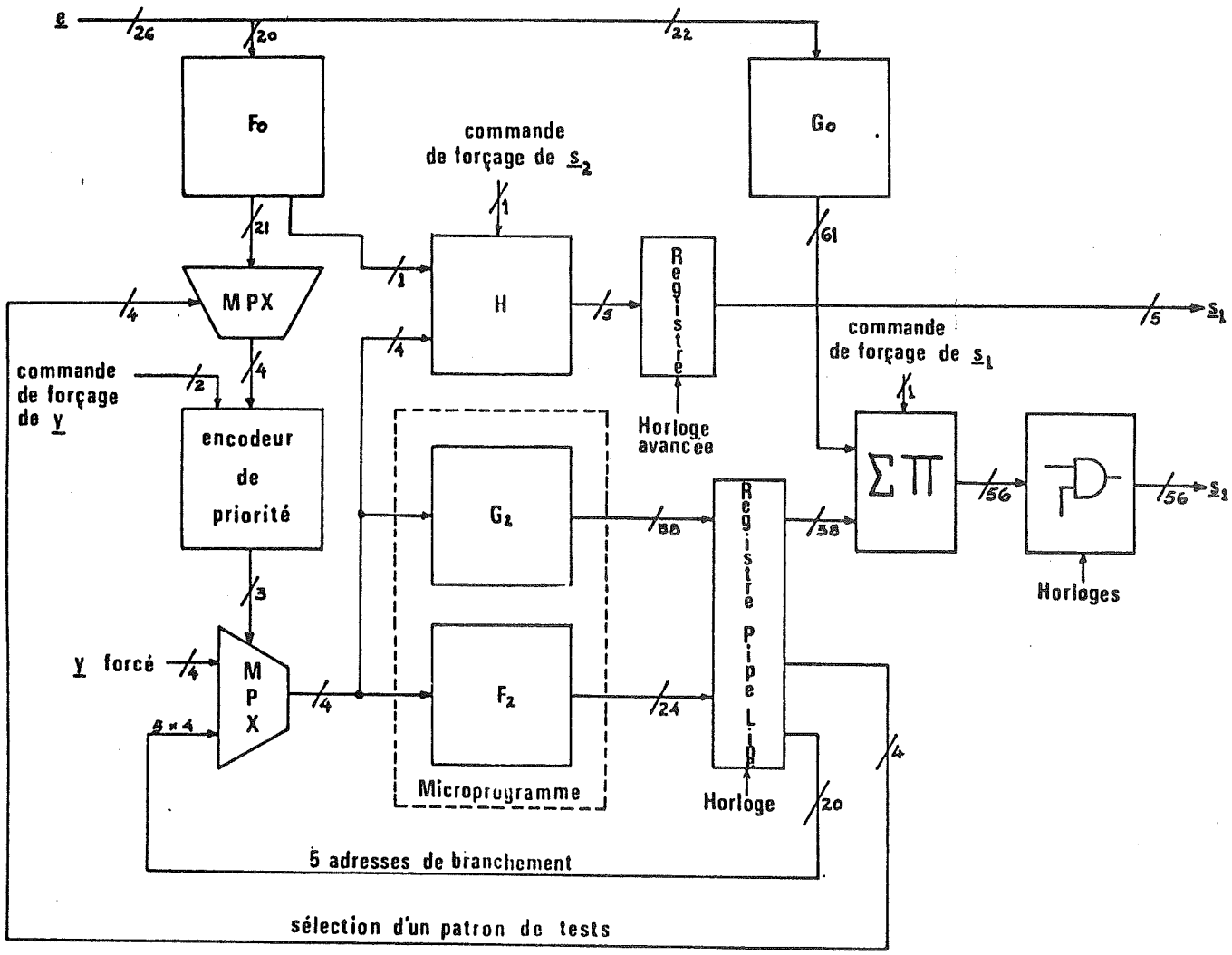


Figure 4.7 Schéma du micro-séquenceur.

- la séquence reçue SR ⩽ 0-5 ⩾
- les codes GA ⩽ 0-23 ⩾ et GB ⩽ 0-23 ⩾, représentant les vecteur de connexion du codeur
- la branche à étendre BR
- l'expression de la métrique de branche FMC ⩽ 0-9 ⩾

Le premier élément est le codeur programmable qui est une copie conforme du codeur utilisé à l'émetteur. Ce codeur est entièrement réalisé à l'aide de portes logiques NON-ET. Des LATCHES conservent l'état des codes GA et GB afin de faciliter la tâche au codeur. A partir de l'état d'un noeud, on peut regénérer les deux séquences correspondant aux branches qui y émergent et qui représentent les deux hypothèses d'information 0 ou 1 à transmettre.

Une mémoire, chargée par le micro-processeur lors de l'initialisation, contient toutes les valeurs de métriques correspondant aux combinaisons des variables d'entrée. Ces valeurs stockées dans METR sont comprimées et doivent être corrigées par la fonction FMC afin de rétablir la dynamique de la métrique. Une fois la métrique de branche corrigée, elle est ajoutée à la métrique accumulée du noeud précédent pour donner celle du nouveau noeud (module FCM). Le case de la sous-pile SPIL correspondante est obtenue en divisant FCS, la valeur de la nouvelle métrique accumulée, par 8. Cette division consiste en un décalage à droite de trois positions. FCS est annulé lorsque FCM est négatif afin que tous les noeuds de métrique accumulée négative soient dirigés vers la sous-pile de plus bas niveau. Le nouvel état du codeur FCE est formé de l'ancien état DE décalé vers la gauche de 1 position, auquel on ajoute la valeur du bit décodé BR pour la branche correspondante. De plus, la nouvelle profondeur FCP est égale à l'ancienne DP augmentée de 1.

Les paramètres FCE, FCM et FCP qui représentent respectivement l'état du codeur, la métrique accumulée et la profondeur font également partie de la pile et sont identifiés par ETAT ⩽ 0-22 ⩾, META ⩽ 0-14 ⩾ et

PROF ⩽ 0-9 ⩾. Le registre ETAT ⩽0⩾ qui contient le bit décodé BR, fait également partie du récupérateur. La mémoire SEQS est dimensionné à 1K x 6 de façon à contenir entièrement la séquence reçue correspondant au bloc en cours de décodage quelle que soit le nombre de niveaux de quantification du canal. Ainsi, tout recul dans l'arbre se fait dans SEQS. Ces séquences reçues sont stockées dans la mémoire SEQS dès leur retrait du tampon d'entrée. La Figure 4.8 montre l'interaction des divers modules décrits ci-dessus.

PILE

Les paramètres META, ETAT et PROF qui décrivent respectivement les métriques accumulées, état du codeur et profondeur d'un noeud, sont tous stockés à une même adresse de la pile. Toutes les entrées dans la pile se font séquentiellement et l'information emmagasinée n'est jamais détruite. L'adresse du dernier noeud N entré est stockée à l'adresse de la mémoire auxiliaire SPIL qui correspond à la sous-pile du noeud. La présence d'un 0 dans une case de SPIL indique que la sous-pile correspondante est vide. Le paramètre SUIV est ajouté à la pile pour représenter l'adresse de la pile où se trouve le noeud qui le suit immédiatement dans sa sous-pile. D'un point de vue pratique, celui-ci est disjoint de la pile afin de permettre un accès simultané à PILE et à SUIV à des adresses différentes. Ces modules mémoire sont tous basés sur des mémoires RAM dynamiques 4116 et exigent donc un rafraîchissement. La Figure 4.9 montre les différents blocs entourant la mémoire SUIV, la Figure 4.10 concerne la PILE et enfin la Figure 4.11 montre la structure de SPIL. Cette dernière n'utilise pas les mêmes composants de base. Les circuits mémoires utilisés sont des RAM statiques et la matrice est organisée en 4096 adresses de 14 bits.

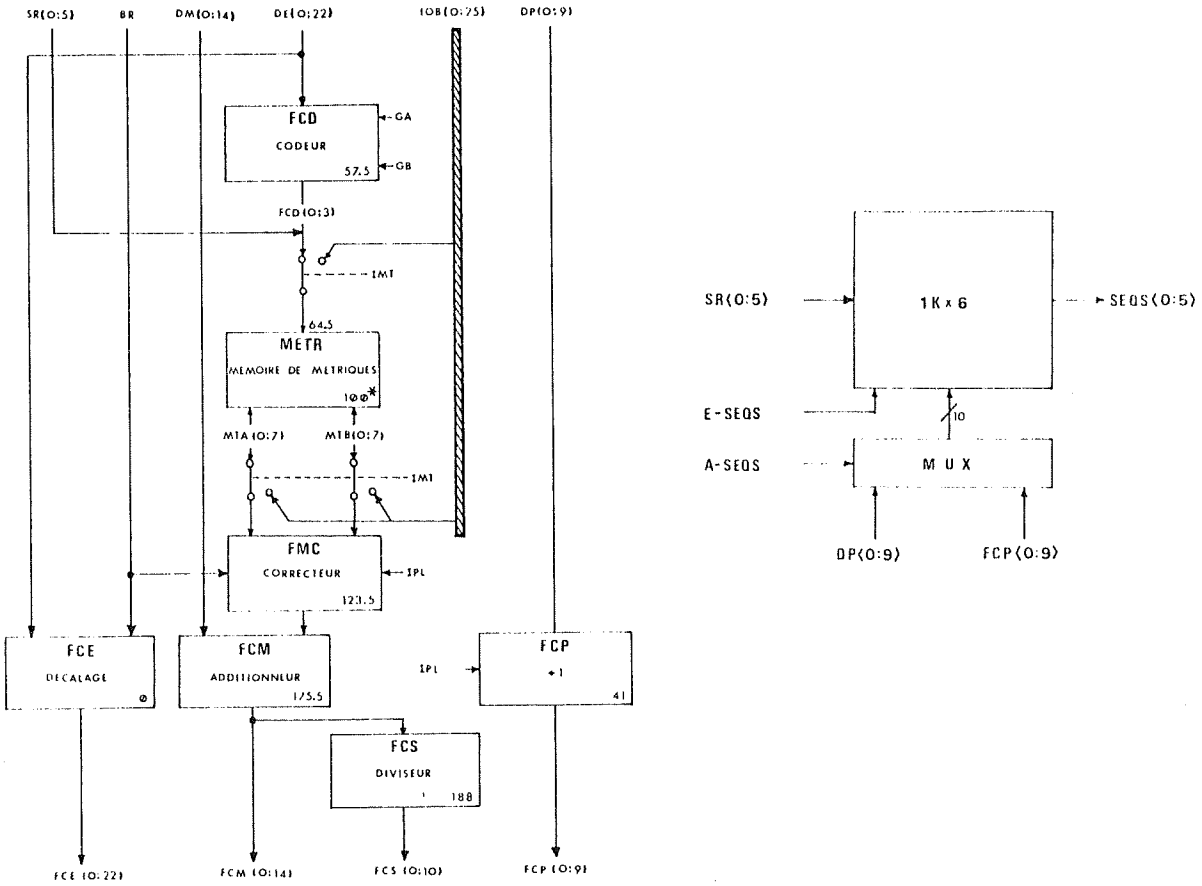


Figure 4.8 Schéma du calculateur des noeuds survivants et mémoire SEQS.

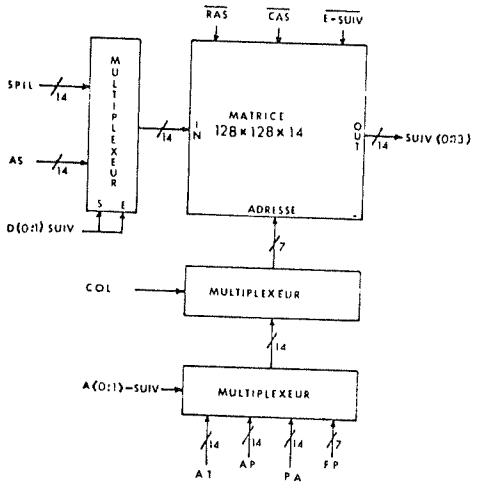


Figure 4.9

Schéma bloc du module SUIV

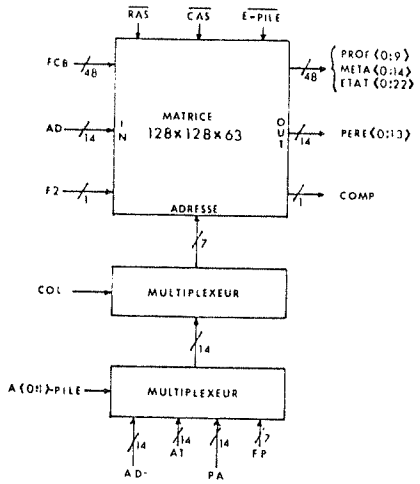


Figure 4.10

Schéma bloc du module PILE

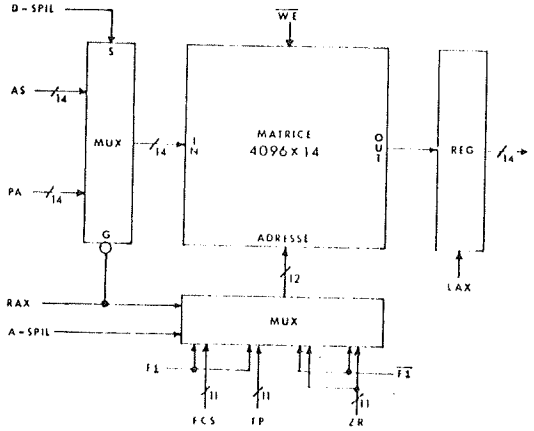


Figure 4.11

Schéma bloc du module SPIL

RÉCUPÉRATEUR

Le récupérateur est formé de 2 modules mémoires identiques, utilisés à tour de rôle par la PILE. Les paramètres faisant partie du récupérateur sont : PERE ⩽ 0-13 ⩾, COMP et ETAT⩽ 0 ⩾. Pendant que le décodeur utilise l'un de ces modules, le récupérateur extrait de l'autre les données pertinentes à la reconstitution du message décodé du bloc précédent. Le bit de message récupéré est la somme MODULO-2 du paramètre ETAT⩽ 0 ⩾ provenant de l'adresse indiquée par PERE et du paramètre COMP de cette même adresse. Cette adresse donne l'adresse du noeud prédécesseur et ainsi de suite jusqu'à ce que tous les bits aient été récupérés. La Figure 4.12 présente le schéma des deux modules du récupérateur.

MODULE DE REMISE EN ORDRE (MESS)

Le récupérateur, partant du noeud terminal de l'arbre, parcourt à rebours le chemin décodé. Chaque bit récupéré est stocké dans une pile appelée MESS, le premier bit récupéré correspondant au dernier bit décodé. L'adresse 0 de cette pile contient donc le dernier bit du message décodé, tandis que la dernière adresse accédée dans cette pile contient le premier bit du message. Il suffit alors de lire le contenu de MESS pour la dernière adresse d'écriture et de remonter dans la pile par décrémentation de l'adresseur. Cette pile adopte donc une structure LIFO. Pour un message de longueur BL bits, la récupération requiert donc BL cycles alors que le transfert de MESS dans le tampon de sortie peut en demander davantage. Il est alors nécessaire d'avoir deux piles similaires. Pendant qu'une sera écrite par le récupérateur, l'autre transférera son contenu au tampon de sortie. A la Figure 4.13, les divers composants d'une des piles LIFO sont illustrés, et l'ensemble du module MESS est donné à la Figure 4.14.

TAMPON D'ENTRÉE (TENT)

La séquence reçue au décodeur est formée de 2 symboles codés et ces symboles peuvent être quantifiés ou pas. La fréquence de réception maximale des symboles est limitée à 1.0 MHZ, soit une période minimale T_r de 1 micro-seconde par symbole. La période de décodage d'une séquence est

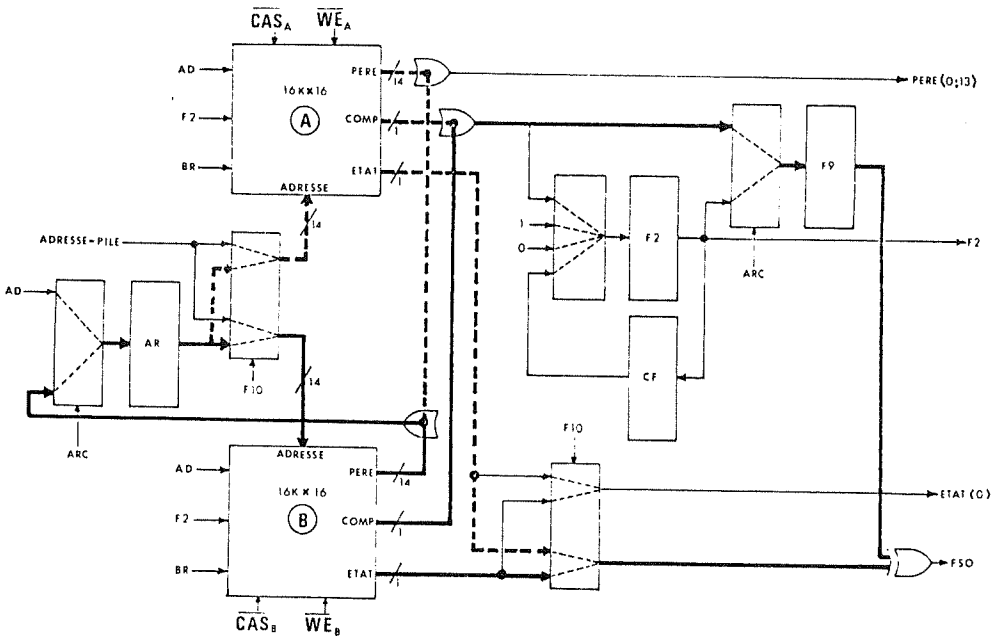


Figure 4.12 Schéma du récupérateur de séquence décodée. Les connexions en trait gras sont celles utilisées par le récupérateur quand le module B lui est assigné.

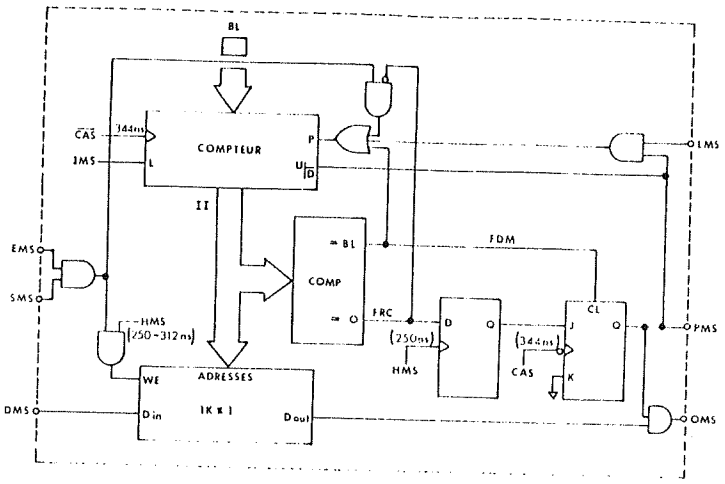


Figure 4.13 Schéma bloc d'une pile LIFO du module MESS.

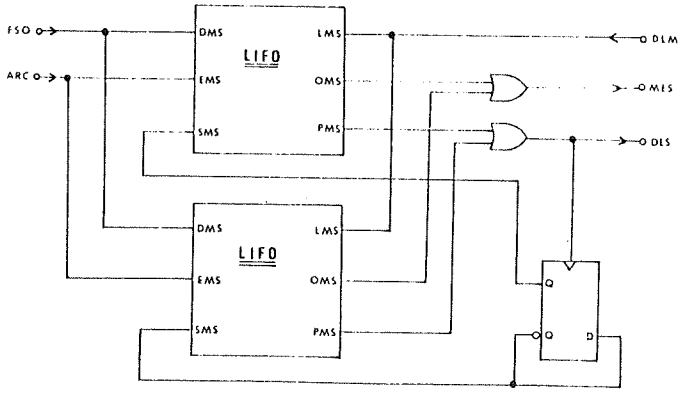


Figure 4.14 Schéma bloc du module MESS.

minimale, lorsqu'aucun des symboles reçus n'est en erreur. On a alors un temps de décodage élémentaire T_d de 375 nano-secondes par symbole. Cette situation correspond à des métriques de branche positives. Si certains symboles étaient en erreur, la métrique de branche n'est plus positive pour l'extension de ces noeuds particuliers, obligeant certains retours arrières. La période de décodage T_d dépend alors du nombre de chemins explorés et de ce fait devient une variable aléatoire. Il est donc possible que la période de décodage soit supérieure à la période de réception ($T_a \geqslant T_r$). Il faut donc doter l'appareil d'un tampon d'entrée pour stocker les symboles requis et qui attendent d'être décodés.

En quantification douce, chaque symbole est représenté par 3 bits. Les séquences formées des symboles SYM_A et SYM_B sont emmagasinées dans une pile FIFO ayant une taille de 64K x 6.

Quatre blocs distincts forment le tampon d'entrée, à savoir : la mémoire TENT, organisée en quatre pages de 16K x 6 bits, le module d'adressage, le module FIFO placé à l'entrée de TENT, et le séquenceur du tampon d'entrée. La Figure 4.15 donne un schéma du tampon d'entrée.

La mémoire TENT est formée de 24 modules 4116 de taille 16K x 1. Ces mémoires sont dynamiques et nécessitent un rafraîchissement, et leur organisation a été choisie de telle sorte que l'on puisse accéder à une séquence de deux symboles en un seul cycle de base. Il est possible de sélectionner les pages individuellement (lecture, écriture) ou simultanément (rafraîchissement). Les données d'entrée SQ peuvent ainsi être distribuées sur tous les circuits. De même, les sorties peuvent être reliées quatre à quatre puisqu'une seule de ces quatre sera activée à la fois, les autres demeurant en haute impédance. Le registre SE ⩽ 0-5 ⩾ conserve la prochaine séquence à être sollicitée par le décodeur. Les cellules des mémoires 4116 sont organisées en 128 lignes de 128 colonnes. Il faut donc multiplexer les lignes d'adresses. Dans un premier temps, l'adresse de ligne est validée

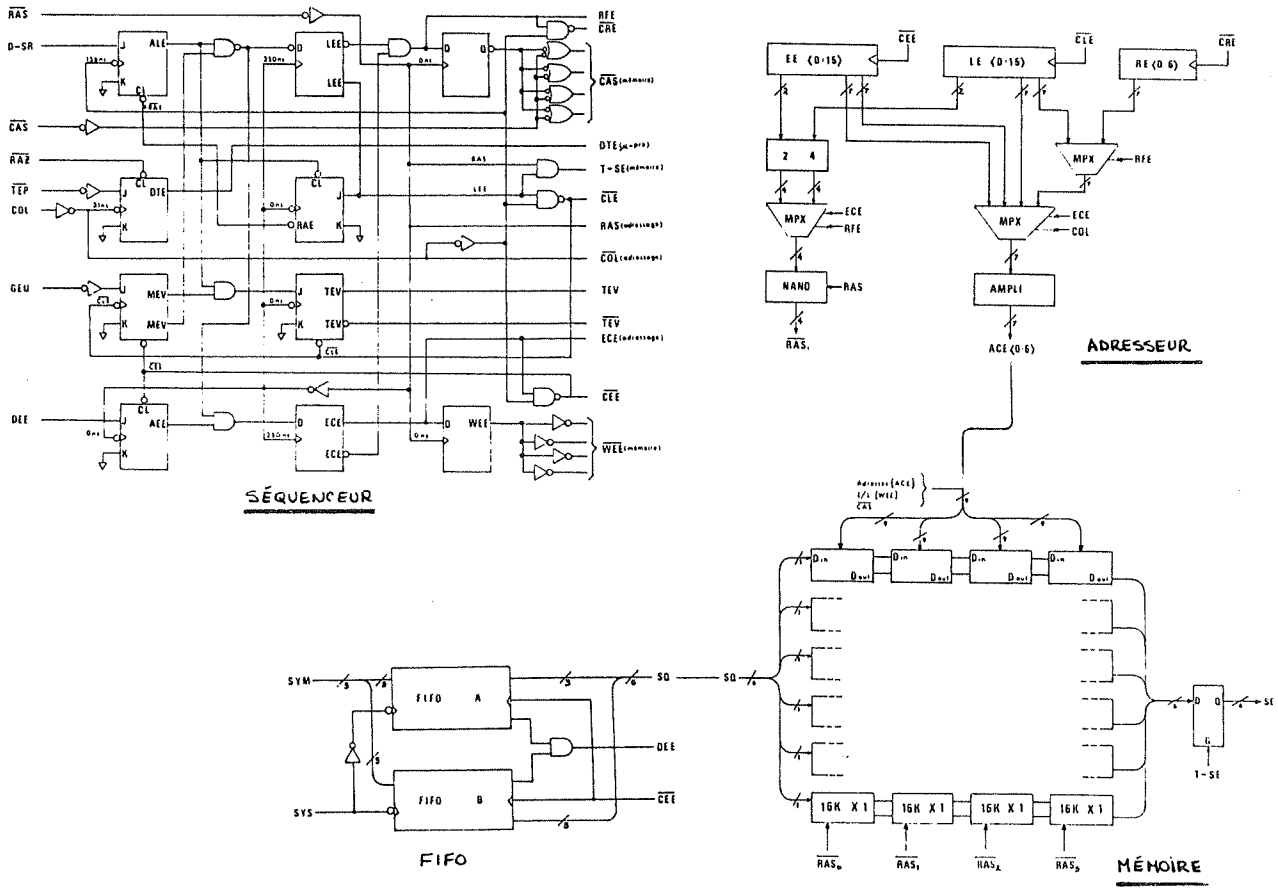


Figure 4.15 Schéma général de l'organisation du tampon d'entrée.

par un front descendant de $\overline{RAS}$ et dans un second temps, l'adresse de colonne est validée par $\overline{CAS}$. Le signal $\overline{WEE}$ indique en tout temps si l'on se trouve en lecture ou en écriture.

Le module d'adressage est constitué de trois registres d'adresse. Le registre d'écriture EE ⩽ 0-15 ⩾ est incrémenté par l'horloge $\overline{CEE}$ émise par le séquenceur lors des demandes d'écriture. Les 2 lignes de plus haut niveau de ce registre servent à la sélection de la page de mémoire 16K x 6. Les lignes d'adresses EE ⩽ 0-6 ⩾ et EE ⩽ 7-13 ⩾ sont multiplexées afin de distinguer les adresses de lignes des adresses de colonnes des mémoires. Le registre de lecture LE ⩽ 0-15 ⩾ est similaire en tous points au registre d'écriture. Par contre le registre de rafraîchissement RE ⩽ 0-6 ⩾ agit sur toutes les pages simultanément. La mise à zéro des registres d'adressage se fait lors de la mise sous tension de l'appareil. Les blocs seront donc stockés l'un à la suite de l'autre sans qu'il y ait remise à zéro des registres d'adresses.

Durant la période de décodage de la queue, la lecture des séquences dans le tampon peut occuper la totalité des cycles d'accès, n'en laissant aucun pour écrire les séquences reçues du démodulateur. Afin d'éviter les pertes sporadiques de séquences, la solution adoptée a consisté à employer une seconde pile FIFO pour stocker les séquences reçues qui ne peuvent être écrites immédiatement dans la mémoire TENT. La pile FIFO est formée de 2 piles indépendantes, la première contenant SYM_A. l'autre contenant SYM_B. Elles sont chargées sur ordre d'un signal de synchronisation des symboles SYS, l'une sur le front montant et l'autre sur le front descendant. Lorsque les 2 piles sont non vides, une demande d'écriture dans TENT est formulée au séquenceur. La taille du FIFO a été établie à 16 mots en réponse au débit d'entrée des symboles et du temps de réponse aux demandes d'écriture.

Le séquenceur répond aux demandes d'écriture formulées par le FIFO, aux demandes de lectures provenant du décodeur et tient compte des contraintes de rafraîchissement. C'est aussi ce module qui détermine à quel moment les signaux de contrôle du module d'adressage doivent être prêts.

TAMPON DE SORTIE

L'insertion d'un tampon de sortie entre le matériel de décodage proprement dit et l'utilisateur permet de régulariser le débit des bits décodés. Il permet également de délivrer les bits décodés à la demande de l'utilisateur, qui se voit par conséquent exempté de l'obligation de se synchroniser avec le décodeur. Le tampon de sortie a une taille de 64K x 1, mémoire disposée en quatre pages de 16 K, constituée encore de mémoires 4116. Un bit lu du tampon de sortie est d'abord stocké dans un registre TS avant d'apparaître en sortie. Ceci évite au demandeur d'attendre que le bit sollicité soit lu de la mémoire et ainsi réduit au minimum le temps d'accès apparent au tampon de sortie. Le transfert d'un bit lu dans TS se fait sous la commande du séquenceur par le biais du signal LES. Pour que la mémoire apparaisse comme une pile FIFO où le premier bit entré est le premier sorti, deux registres pointeurs sont utilisés tout comme pour le tampon d'entrée. L'adresse de lecture sera conservée par LS ⩽ 0-15 ⩾ alors que ES ⩽ 0-15 ⩾ pointe vers l'adresse d'écriture. Ici aussi un registre RS ⩽ 0-6 ⩾ tient à jour l'adresse de rafraîchissement de la mémoire.

Le séquenceur traite les demandes d'écriture et de lecture, puis fournit au matériel de décodage l'état du tampon de sortie. Un front descendant du signal SYB commande de copier le contenu de TS dans OUT et impose au séquenceur à lire le bit décodé suivant pour mettre TS à jour. Le signal DPR avertit l'utilisateur que des données sont prêtes à être lues dans le tampon de sortie. La Figure 4.16 donne une vue d'ensemble des constituants du tampon de sortie.

Nous avons décrit dans ce chapitre les concepts fondamentaux utilisés dans la conception du décodeur, ainsi que la structure matérielle telle qu'implantée. Naturellement, un grand nombre de détails n'ayant pas leur place dans le cadre de ce rapport ont été omis. Ces détails sont cependant disponibles dans la référence [14].

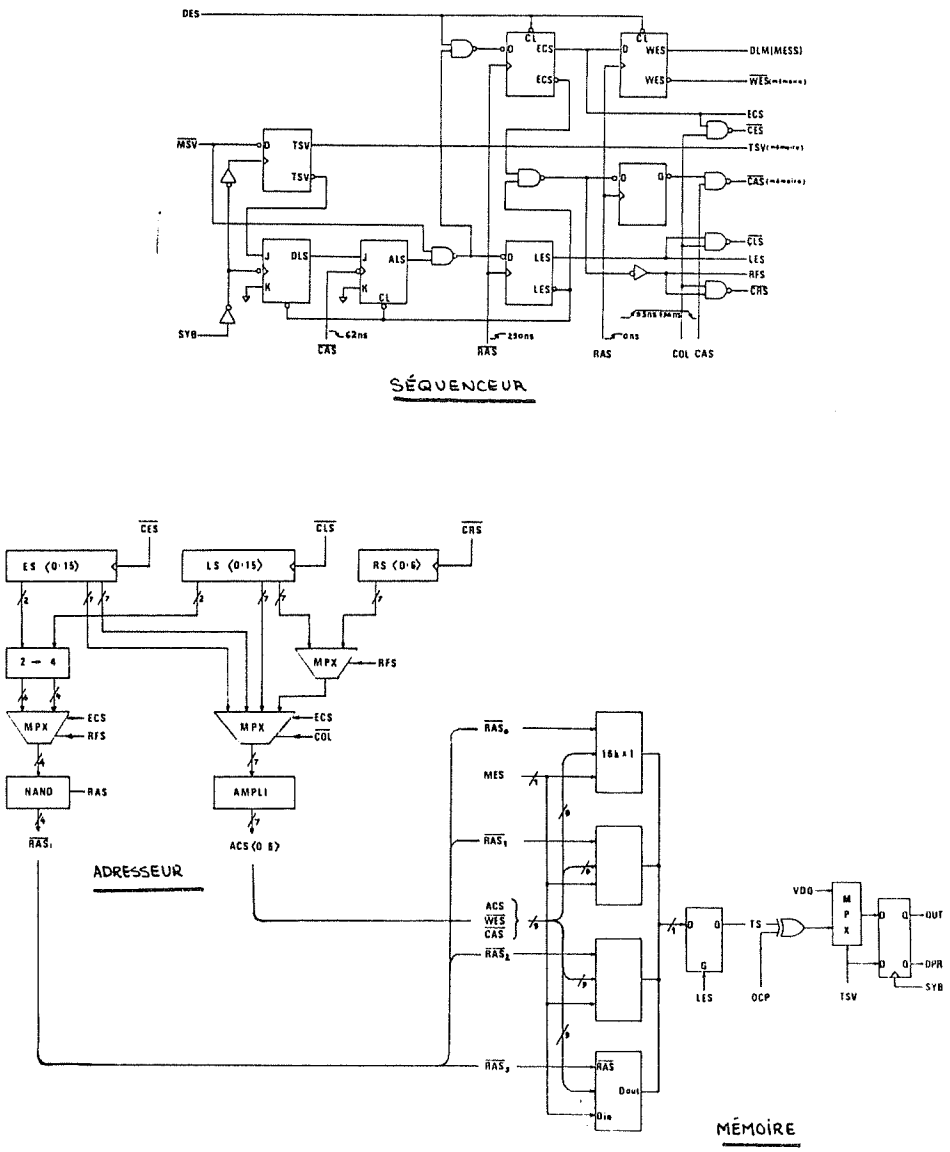


Figure 4.16 Schéma général de l'organisation du tampon de sortie.

5. PROCÉDURES DE TESTS

5.1 Généralités

Tel que mentionné plus haut, un grand nombre de tests décrivant la dynamique de décodage a été envisagé pour le prototype. Les résultats des tests devraient servir à mieux comprendre certains éléments fins de la procédure du décodage et à faire ressortir les interactions entre les différentes étapes de l'algorithme en plus de fournir les résultats classiques des performances d'erreur du décodeur. A cet effet, une étude préalable sur la méthodologie à employer pour la cueillette et la présentation de ces données et statistiques a dû être entreprise.

Etant donné la vitesse de décodage de la machine (supérieure à 500 Kbits/s), des statistiques sur plusieurs heures de fonctionnement auraient très vite débordé les différents compteurs envisagés pour la cueillette de données. De plus, et encore à cause de la vitesse de décodage, il a été décidé que les différentes informations seront recueillies par bloc et non pas par bit, et que ces informations seront emmagasinées sous forme d'histogrammes. Dans le but d'uniformiser leur présentation, tous les histogrammes comportent 256 cellules quel que soit le paramètre considéré. Quant au nombre d'entrées dans chaque cellule d'un histogramme quelconque, il a été fixé à 999,999, ce qui revient à dire qu'une série de tests ne devrait pas impliquer plus de 999,999 blocs.

5.2 Cueillette des statistiques

La cueillette des statistiques s'effectue sous le contrôle du micro-ordinateur AIM-65 incorporé au décodeur. Des modules fonctionnant au rythme du décodage enregistrent chaque information pertinente au cours du décodage de chaque bloc. A la fin du décodage d'un bloc, les résultats sont stockés dans des registres tampons et les modules sont reconditionnés.

Pendant le décodage du bloc suivant, le micro-ordinateur AIM-65 interroge ces registres tampons les uns après les autres et met à jour les tableaux correspondant aux différents histogrammes. Pour chacun des paramètres sélectionnés, un tableau comprenant 256 cases différentes est prévu et chaque case peut emmagasiner 999,999 entrées. Par conséquent, quel que soit le paramètre considéré, la cumulative ne peut avoir plus de 256 points. Ce nombre de points de mesures a été jugé suffisant pour donner toute la précision désirée.

Tel que mentionné plus haut, la cueillette des statistiques s'effectue sous le contrôle du microprocesseur AIM-65. Une unité périphérique d'entrée de sortie (PIA) établit le lien entre le micro-ordinateur et les divers modules de statistiques.

Deux ports sont utilisés : l'un sert au contrôle de la cueillette des statistiques (dénoté "STAT.CONTROL BUS"), et l'autre sert au transfert des données (dénoté "STAT.DATA BUS"). La structure de ces ports est fournie ci-dessous, et un schéma bloc de l'opération est donné à la Figure 5.1.

STAT.DATA BUS ⩽ 0-7 ⩾ : Adresse relative d'un élément du tableau de la statistique sélectionnée par "STAT.CONTROL BUS".

STAT.CONTROL BUS ⩽ 0 ⩾ : (STC) sélection du temps de calcul
⩽ 1 ⩾ : (STA) sélection du temps d'attente
⩽ 2 ⩾ : (SAP) sélection du nombre d'accès à la pile
⩽ 3 ⩾ : (SPO) sélection du nombre de cycles positifs
⩽ 4 ⩾ : (SSP) sélection du nombre de cycles sous-pile
⩽ 5 ⩾ : (SNE) sélection du nombre de cycles négatifs
⩽ 6 ⩾ : (STS) sélection de la taille du tampon de sortie
⩽ 7 ⩾ : (STE) sélection de la taille du tampon d'entrée
⩽ 8 ⩾ : (SMT) sélection de la métrique accumulée totale.

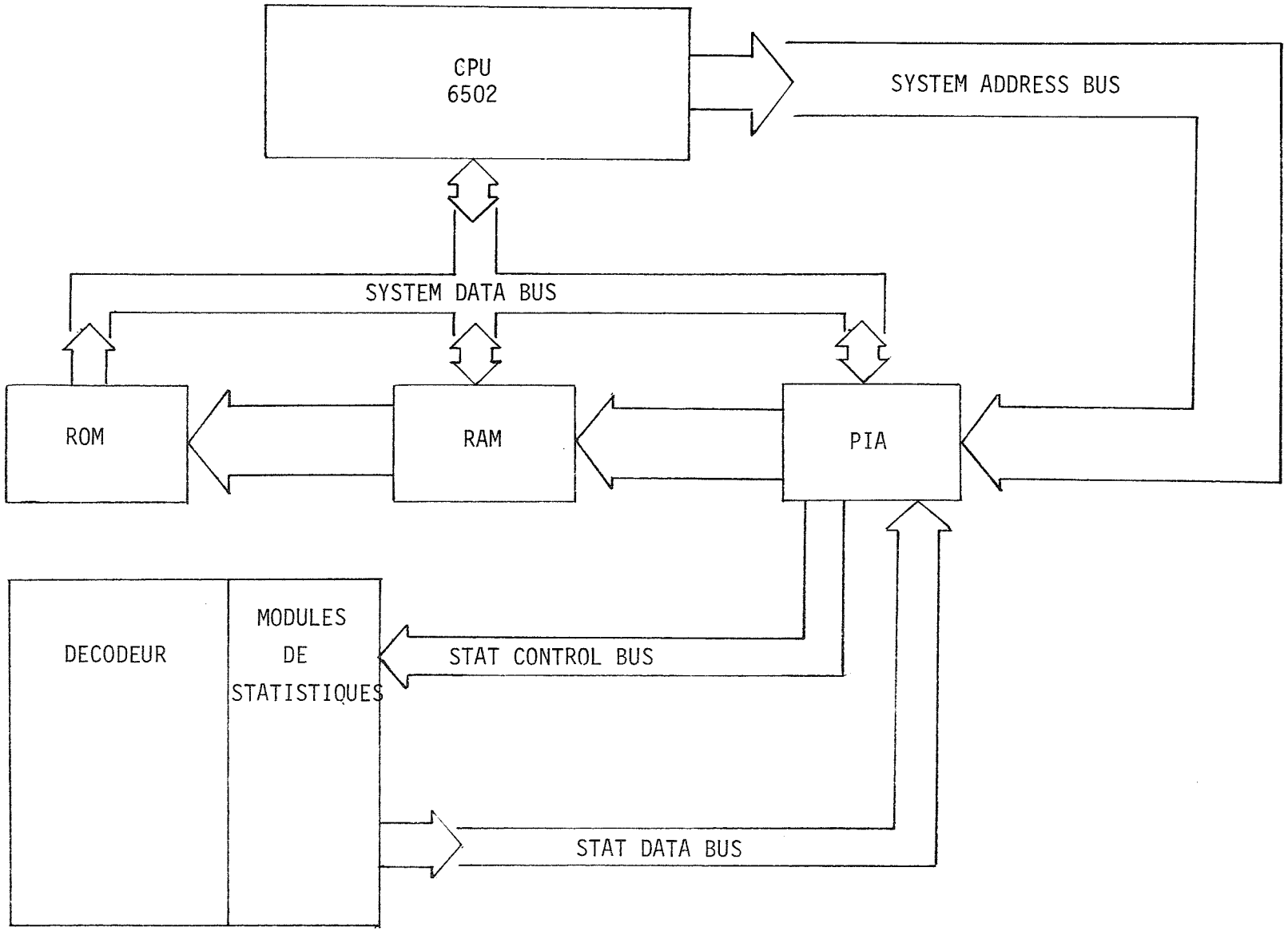


Figure 5.1 Schéma général du système de cueillette des statistiques.

La cueillette proprement dite est effectuée de la façon suivante : à la fin du décodage d'un bloc, une interruption est générée. Le microprocesseur répond à cette interruption en procédant à la cueillette des statistiques et au marquage (i.e. incrémentation des registres), des tableaux correspondant aux différents paramètres testés. Ceci étant effectué, le micro-ordinateur retombe en attente d'une nouvelle interruption lui indiquant qu'un nouveau bloc a été décodé. Cette procédure se répète aussi longtemps que le décodage se poursuit.

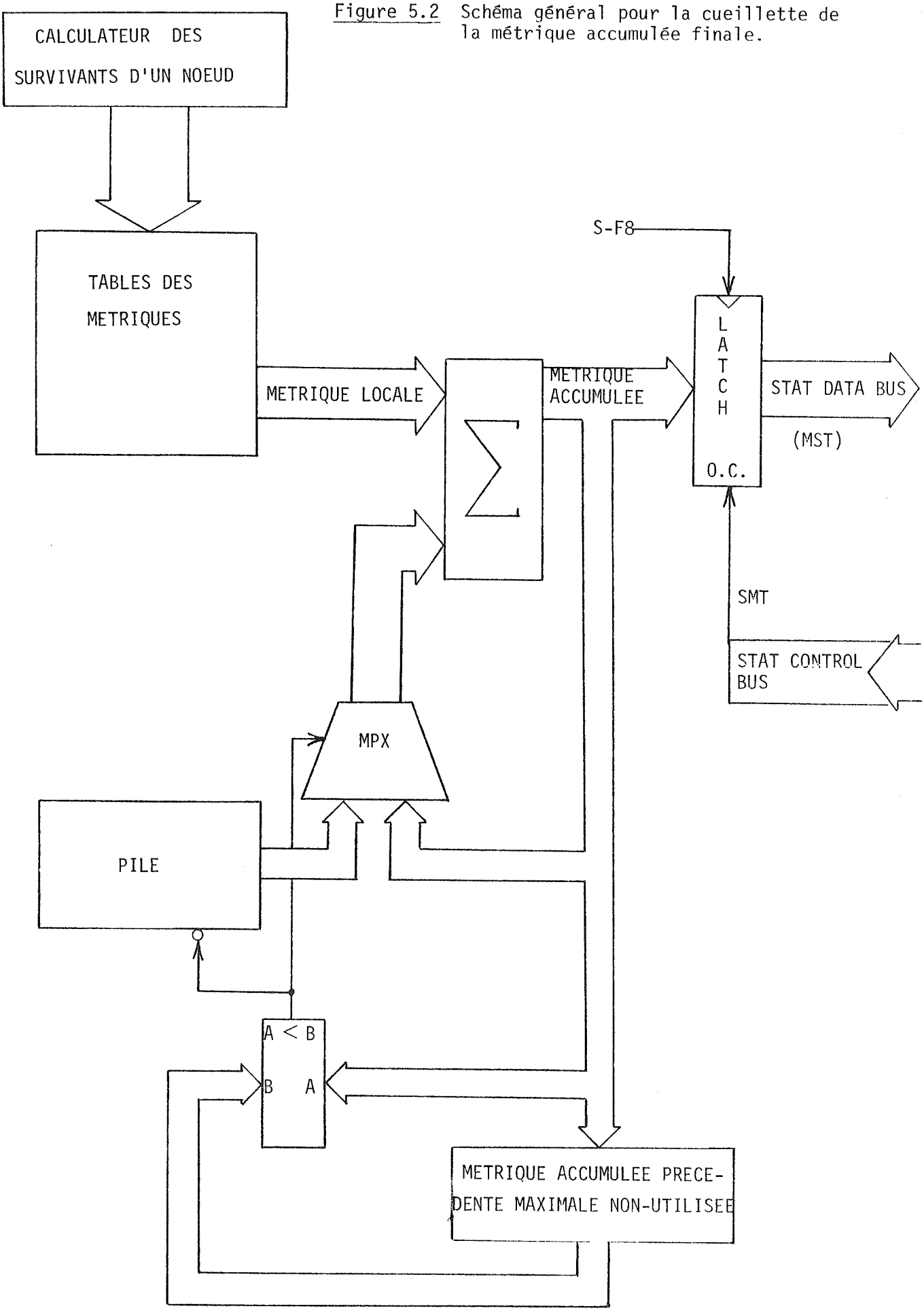
Une fois tous les blocs décodés, le microprocesseur prépare l'impression sur papier du contenu de chacun des tableaux. Comme indiqué plus haut, chacun de ces tableaux comporte 256 entrées. A partir de ces tableaux, les histogrammes et cumulatives sont construits et tracés à l'ordinateur VAX 11-750 utilisant un programme conçu spécialement à cette fin. L'automatisation du tracé des courbes permet une présentation très souple des résultats et leur obtention très rapidement après la fin des tests.

Les principes de la cueillette étant établis, la méthode utilisée pour "mesurer" chacun des paramètres est décrite dans les sections suivantes.

5.2.1 Métrique accumulée

La métrique accumulée du noeud prolongé se trouve en tout temps à la sortie du sommateur. Il s'agit donc de mémoriser la valeur de la métrique accumulée du noeud terminal, c.a.d. lorsque tout le bloc a été décodé. Un signal particulier (signal S-F8) indique que la métrique accumulée totale se retrouve sur le bus [14]. Ce signal est donc utilisé pour conditionner la mémorisation. Pendant le décodage d'un nouveau bloc, cette information de métrique totale accumulée dénotée MST ⩽ 0-14 ⩾ est présente sur le STAT.DATA BUS. MST ⩽ 14 ⩾ étant le bit de signe, et supposant celui-ci toujours positif, seuls les bits MST ⩽ 6-13 ⩾ seront stockés dans le tableau des cumulatives. Par incréments $\Delta = 64$, la plage couverte est donc [+64, +16 383]. La Figure 5.2 illustre la procédure utilisée.

Figure 5.2 Schéma général pour la cueillette de la métrique accumulée finale.



5.2.2 Taille maximale de la file d'attente dans le tampon d'entrée

La taille de la file d'entrée est obtenue en tout temps en considérant la différence des adresseurs du tampon d'entrée, sous la forme

$$\text{ADD } \Delta = \text{ADD IN} - \text{ADD OUT} = \text{ADD IN} + \overline{\text{ADD OUT}} + 1.$$

Un comparateur et un registre tampon permettent de conserver la plus grande différence pour un bloc en cours de décodage. A la fin du décodage, le signal d'activation T-AA force une mémorisation de cet écart maximum, dénoté EM. Comme pour les autres statistiques, au cours du décodage du bloc suivant cette information est insérée à sa place dans le tableau de ce paramètre. Par incrément $\Delta = 256$, la plage couverte est [256,65535]. Toute métrique supérieure à la dernière case du tableau , c.a.d. supérieure à 65535 est insérée dans cette dernière case. Ainsi, bien que EM soit EM $\leqslant$ 0-15 $\geqslant$, seuls les bits EM $\leqslant$ 8-15 $\geqslant$ sont conservés. La Figure 5.3 illustre la procédure utilisée.

5.2.3 Taille de la file d'attente au tampon de sortie à la fin du bloc

La taille de la file d'attente au tampon de sortie varie en fonction de contraintes et caractéristiques d'utilisation extérieures au décodeur proprement dit. Aussi pour nos besoins, seule la taille de la file d'attente présente dans le tampon de sortie à la fin du décodage d'un bloc présente un intérêt. D'un point de vue physique, l'espace disponible dans cette partie du écodeur s'est avéré beaucoup trop restreint pour permettre l'addition des circuits nécessaires à la détermination de la taille maximum de la file d'attente.

La procédure utilisée pour la cueillette de ce paramètre est semblable à celle utilisée dans le cas du tampon d'entrée et consiste encore en la différence des adresseurs de lecture et de l'écriture par la méthode du complément à 2.

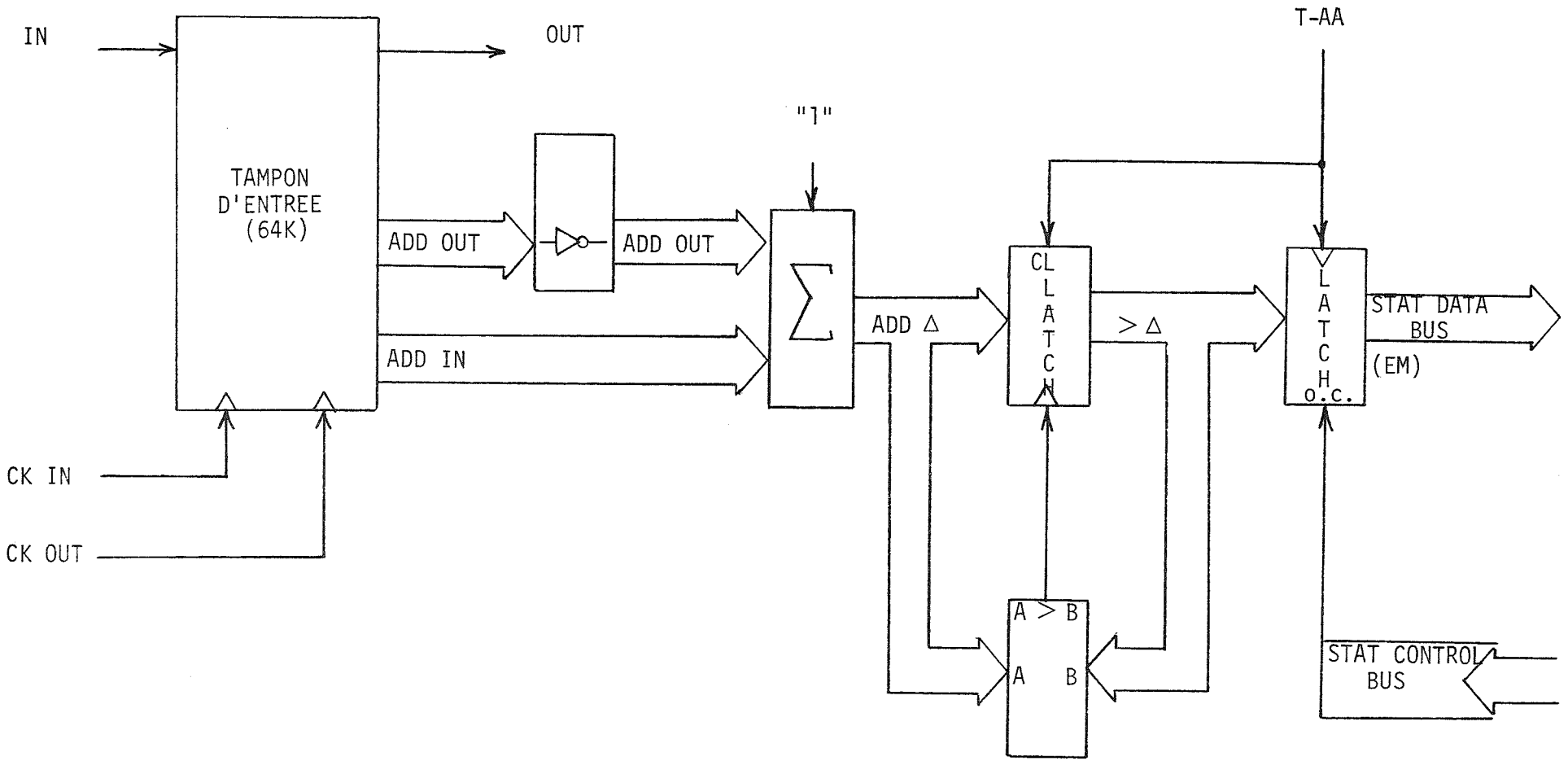


Figure 5.3 Schéma général pour la cueillette de la taille maximale de la file d'attente dans le tampon d'entrée.

Tel qu'indiqué à la Figure 5.4, le signal T-AA indiquant la fin du décodage d'un bloc active la mémorisation de cette différence représentée par le signal GTS. Là encore, cette valeur est tabulée au cours du décodage du bloc suivant. Les signaux GTS utilisent 16 bits, i.e. GTS ⩽ 0-15 ⩾, mais pour l'évaluation des cumulatives, seuls les bits GTS ⩽ 8-15 ⩾ sont considérés. Par incréments de 256, la plage de valeurs couvertes est donc [256,65535+], et toute valeur supérieure à 65535 entre dans cette dernière case.

5.2.4 Temps de calcul

Pour chaque cycle d'horloge, le décodeur effectue un calcul à condition qu'il ne soit pas en situation d'attente (micro-instruction 8) ou qu'il n'est pas en mode toujours actif, c.a.d. qu'il n'effectue pas d'extensions auxiliaires (micro-instruction 9). Ici, on compte donc le nombre de cycles d'horloge pour lesquels le décodeur n'est pas dans un de ces deux états. Comme ce sont les nombres de cycles d'horloge qui sont cumulés, il faut les convertir en temps en utilisant 375 ns par cycle.

Le signal "LAX" est choisi pour indiquer le début d'un cycle. Combiné à la non apparition des états 8 et 9, on obtient un signal apte à incrémenter un compteur d'événements 0-32K. Le décompte de ces événements, dénoté TC est mémorisé à la fin du décodage par le signal de fin de décodage de bloc T-AA, et quelques nanosecondes plus tard le compteur est remis à zéro et redevient disponible pour le bloc suivant. La mise à jour du tableau cumulatif correspondant au paramètre temps de calcul d'un bloc donné est encore effectuée pendant le décodage du bloc suivant.

Tel qu'indiqué plus haut, le compteur peut compter jusqu'à 32767 événements avec indication de débordement. Avec des incréments de 128, la plage couverte est donc [128, 32767+], où les débordements sont indiqués par +. La Figure 5.5 schématise la procédure.

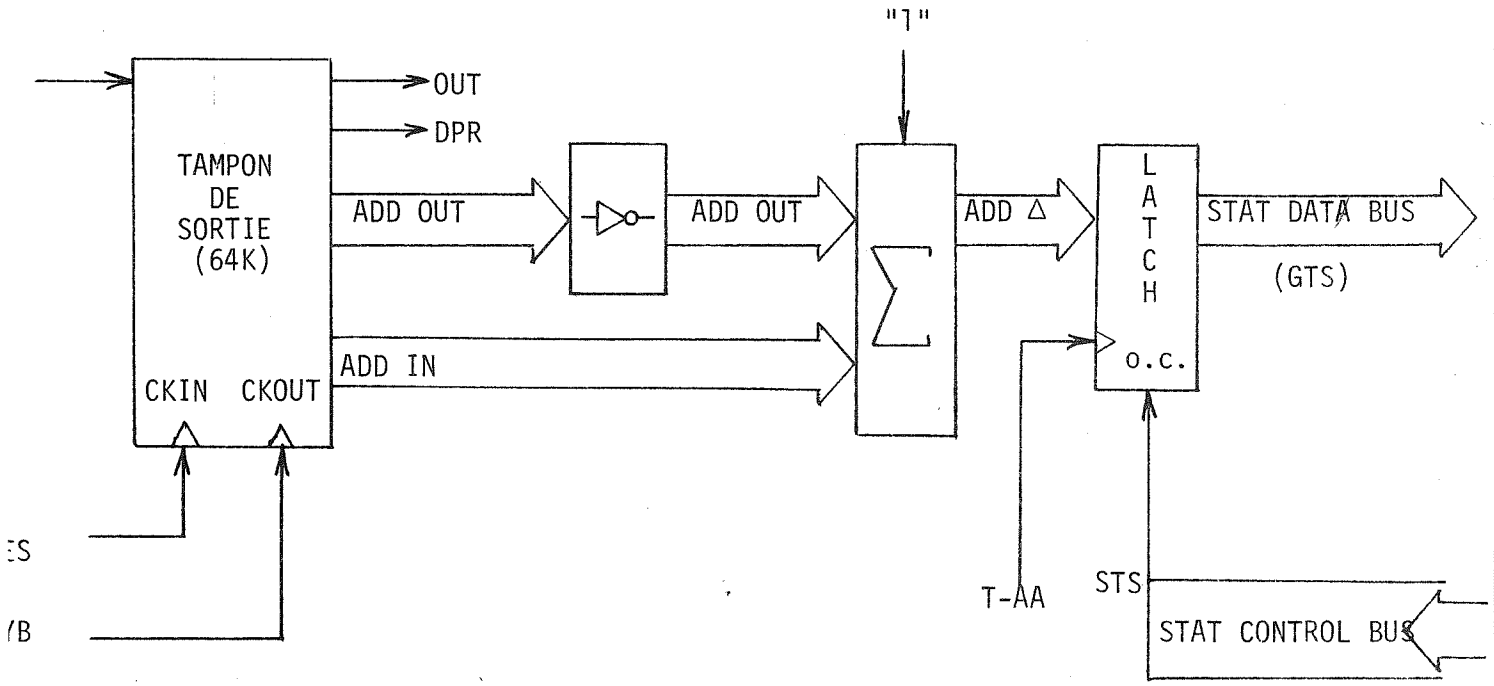


Figure 5.4 Schéma général pour la cueillette de la taille de la file d'attente dans le tampon de sortie.

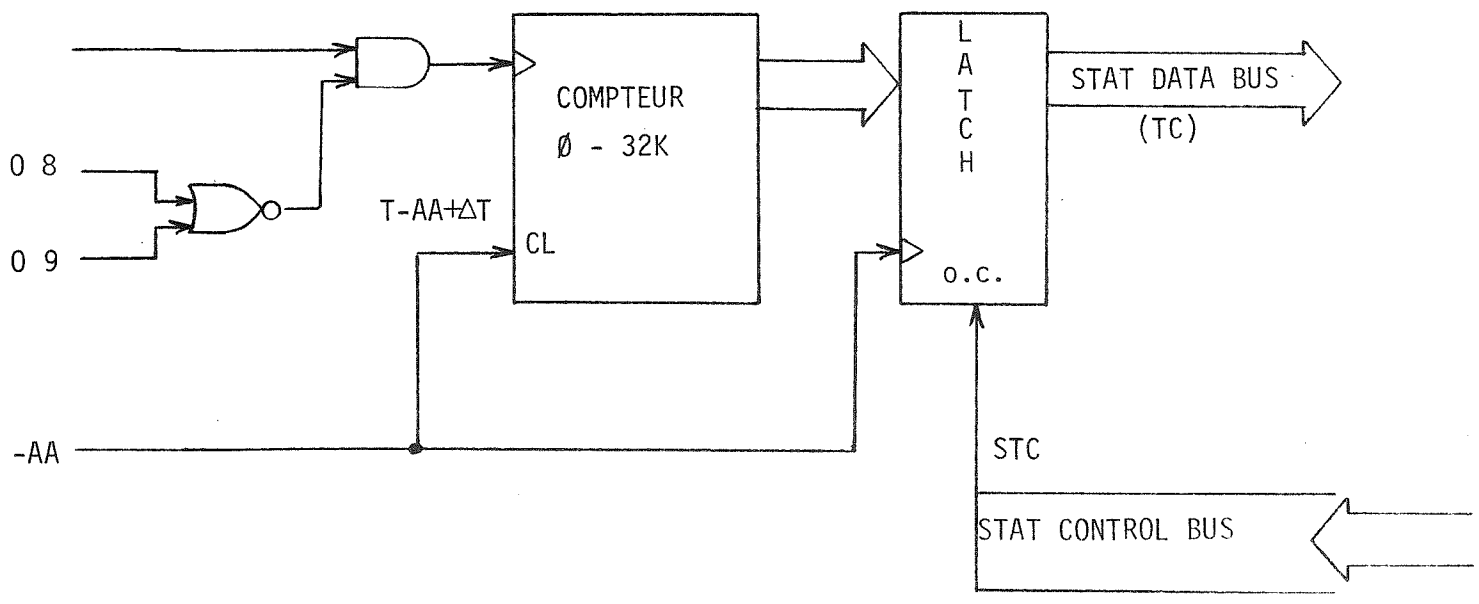


Figure 5.5 Schéma général pour la cueillette du temps de calcul.

5.2.5 Temps d'attente

Le même principe employé pour le temps de calcul est utilisé pour établir les statistiques du temps d'attente (voir Fig. 5.6). Ici cependant, il s'agit de compter le nombre de cycles durant lesquels le décodeur est soit dans l'état 8 (attente), soit dans l'état 9 (extensions auxiliaires). Ce paramètre dénoté TA est donc le complément du temps de calcul, et la somme de ces deux paramètres permet donc de calculer le temps total (actif et attente) de décodage.

Par incréments de 128, la plage couverte est encore [128,32767+] avec encore indication de débordement.

5.2.6 Nombre de calculs

Un calcul pouvant comporter des cycles d'extension de branche positive, de cycles d'extension de deux branches négatives et des cycles de sous-pile concomittants, on distingue les cycles positifs, les cycles négatifs et les cycles de sous-pile.

5.2.6.1 Cycles positifs

Trois états possibles du décodeur correspondent à des cycles dits positifs, à savoir :

- Extension de la branche 0
- Extension de la branche 1
- Insertion d'un noeud dans la pile.

Il suffit donc de compter le nombre de fois où le décodeur se trouve dans l'un de ces états, et qui se représente par la combinaison des signaux suivants :

$$[(\text{MICRO 1} + \text{MICRO 6}) \cdot \text{MES} \cdot \overline{\text{TSF}}] + [\text{MICRO 2} \cdot \text{MES} \cdot \overline{\text{TSG}}]$$

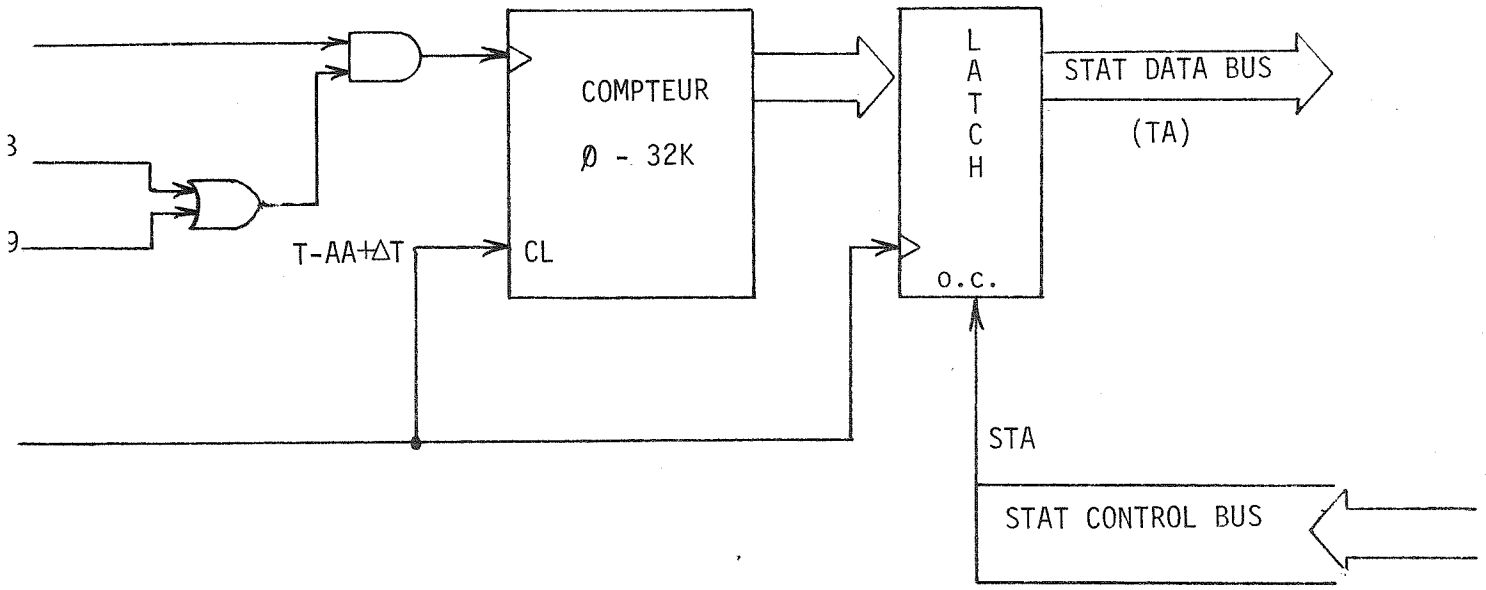


Figure 5.6 Schéma général pour la cueillette du temps d'attente.

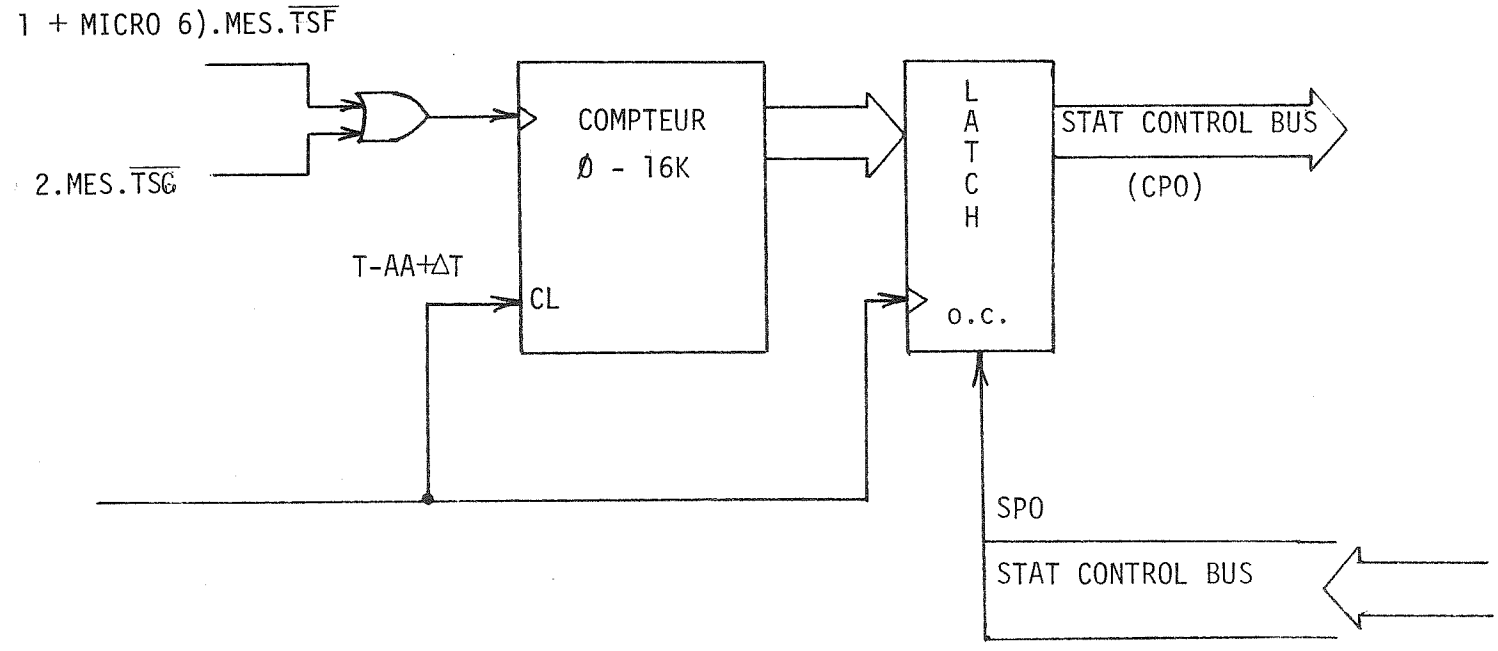


Figure 5.7 Schéma général pour la cueillette du nombre de cycles positifs par bloc.

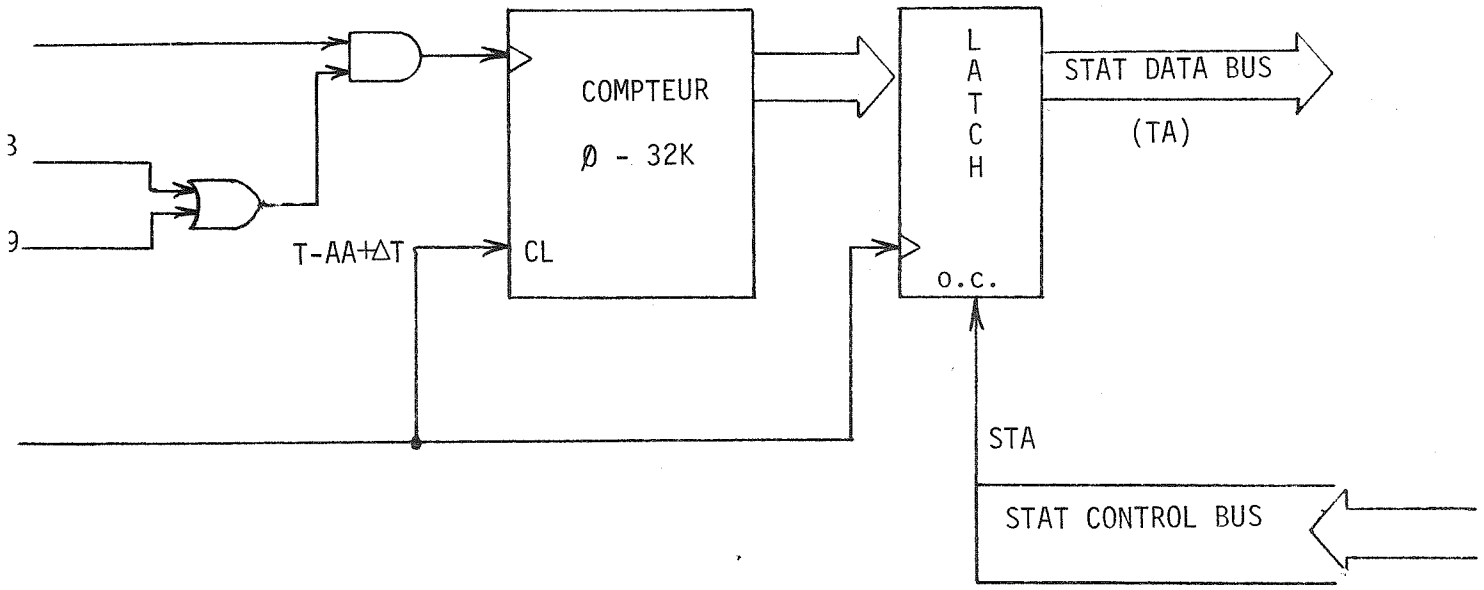


Figure 5.6 Schéma général pour la cueillette du temps d'attente.

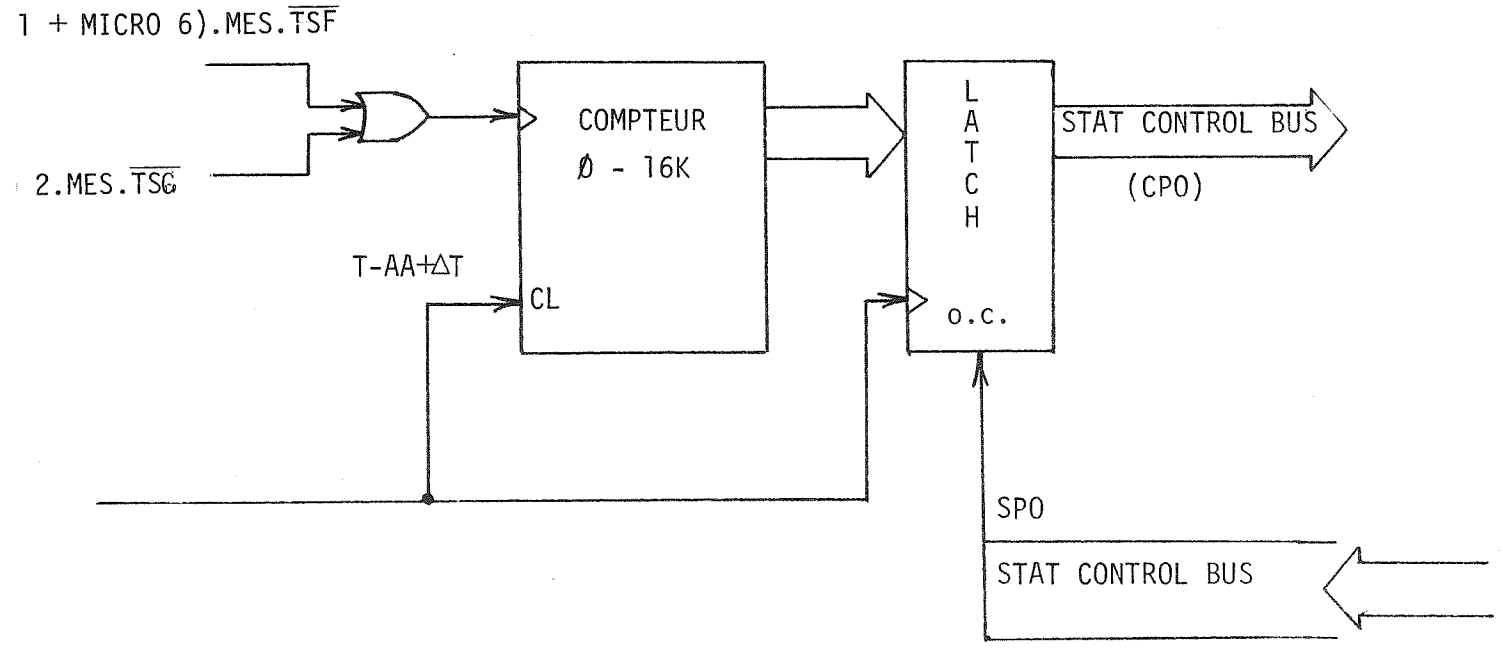


Figure 5.7 Schéma général pour la cueillette du nombre de cycles positifs par bloc.

Tel que montré à la Figure 5.7, ce nombre de cycles positifs tel qu'il apparaît dans le STAT • DATA BUS est dénoté CPO.

Un simple calcul du maximum possible pour un bloc de 1024 bits a fixé la limite supérieure du compteur à 16383. Avec des incréments de 64, la plage couverte est donc [64,16383].

Suite à la capture du compte final pour chaque bloc décodé, le compteur est remis à zéro pour être prêt pour le décodage d'un nouveau bloc. L'insertion de chaque nouvelle valeur dans le tableau cumulatif est encore effectuée pendant le décodage du bloc suivant.

5.2.6.2 Cycles négatifs

Un cycle négatif apparaît en conjonction avec un débit de cycle de sous-pile et correspond à un accès d'un noeud dans une sous-pile. On utilise un compteur 0-16K pour cumuler l'apparition de ces événements dans un bloc et la valeur finale dénotée CNE est transférée à un registre tampon à la fin du décodage du bloc. Une remise à zéro du compteur est ensuite effectuée. Par incréments de 64, la plage couverte est [64,16383]. La Figure 5.8 illustre la méthode.

5.2.6.3 Cycles de sous-piles

Un cycle de sous-piles correspond au début de la recherche du sommet de la pile, et correspond à l'exécution de la micro-instruction 3. Tel que montré à la Figure 5.9, le signal qui identifie le nombre de sous-pile à la fin d'un bloc est CSP. Par incréments de 64, la plage couverte est [64,16383].

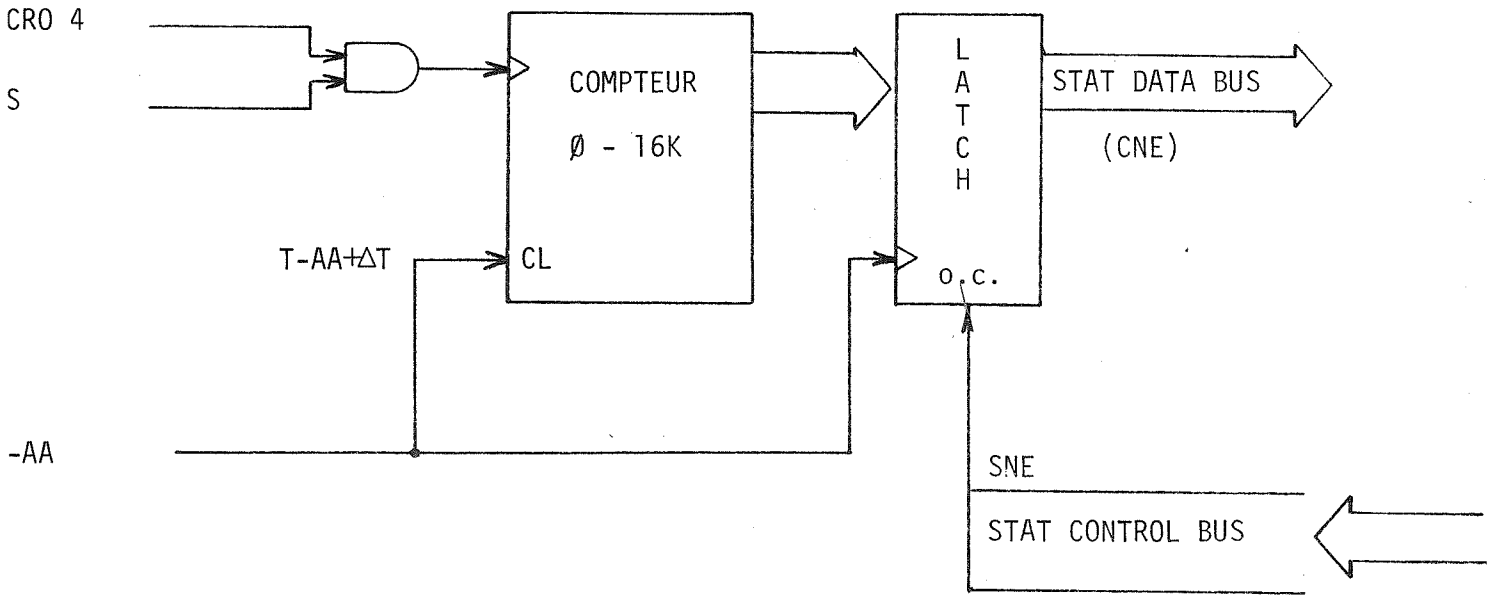


Figure 5.8 Schéma général pour la cueillette du nombre de cycles négatifs.

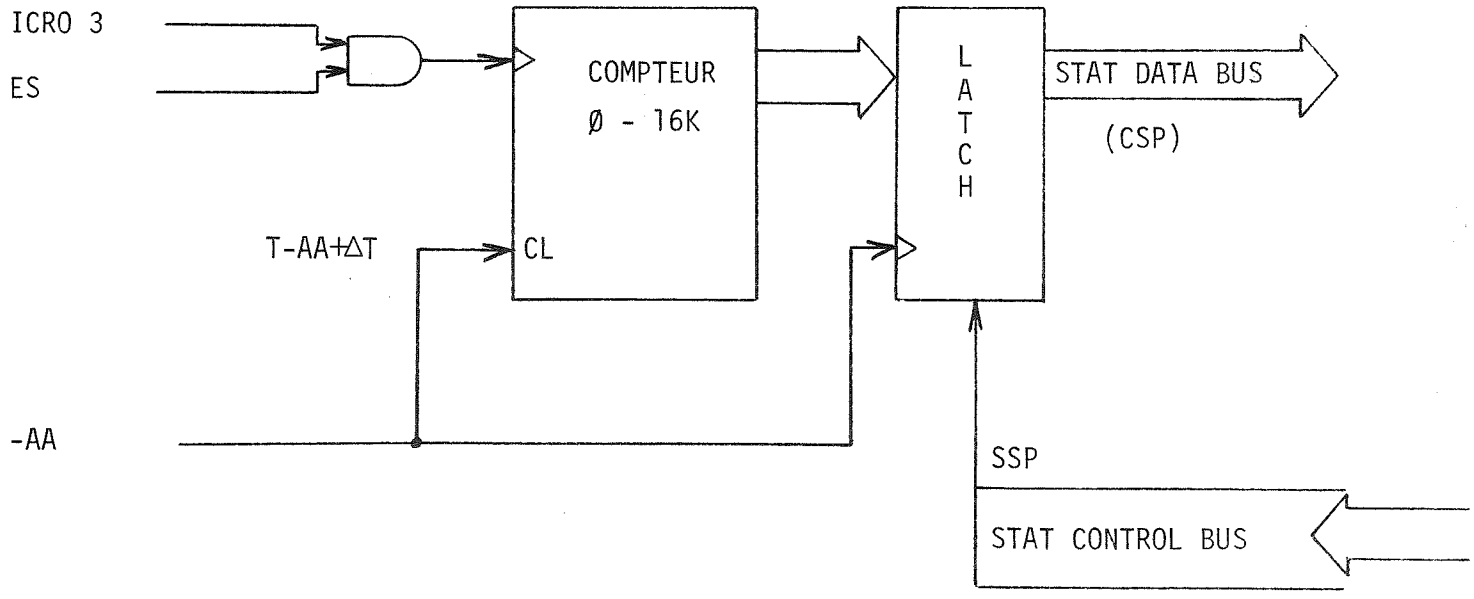


Figure 5.9 Schéma général pour la cueillette du nombre de cycles de sous-pile.

5.2.7 Nombre d'accès à la pile

On rappelle que toutes les entrées dans la pile se font séquentiellement et l'information stockée dans la pile n'est jamais détruite. Le contenu du compteur SUIV du décodeur représente l'adresse de la pile où se trouve le noeud qui le précède dans la même sous-pile. Quand un noeud est entré dans la pile, son adresse est copiée dans la case correspondante de sa sous-pile. Ainsi, deux adresseurs sont utilisés pour accéder à la pile. L'un sert à l'entrée d'un noeud dans la pile et l'autre sert à la récupération des noeuds.

L'adresseur utilisé pour connaître le nombre d'entrées dans la pile est un compteur 0-32K qui est remis à zéro à chaque nouveau bloc. Il suffit donc de conserver dans un registre l'adresse maximum atteinte par le compteur au cours du décodage. Tel que montré à la Figure 5.10, cette adresse est identifiée dans le STAT•DATA BUS par le signal NAP. Avec des incréments de 128 et une prévision de débordements, la plage couverte est $\lfloor 128, 32767+ \rfloor$.

5.2.8 Nombre d'erreurs

La performance d'erreur du décodeur est fournie sous la forme d'une répartition du nombre d'erreurs par blocs, à partir de laquelle une probabilité d'erreur par bit décodé ou par bloc est facilement déduite.

Deux mises en oeuvre de statistiques d'erreur ont été effectuées. La première, entièrement en logiciel, consiste simplement en une comparaison des séquences d'information (préalablement stockées) et des séquences décodées. Tout comme pour les autres statistiques, un tableau donnant la répartition des erreurs par bloc est construit, avec ici un incrément égal à 16 erreurs.

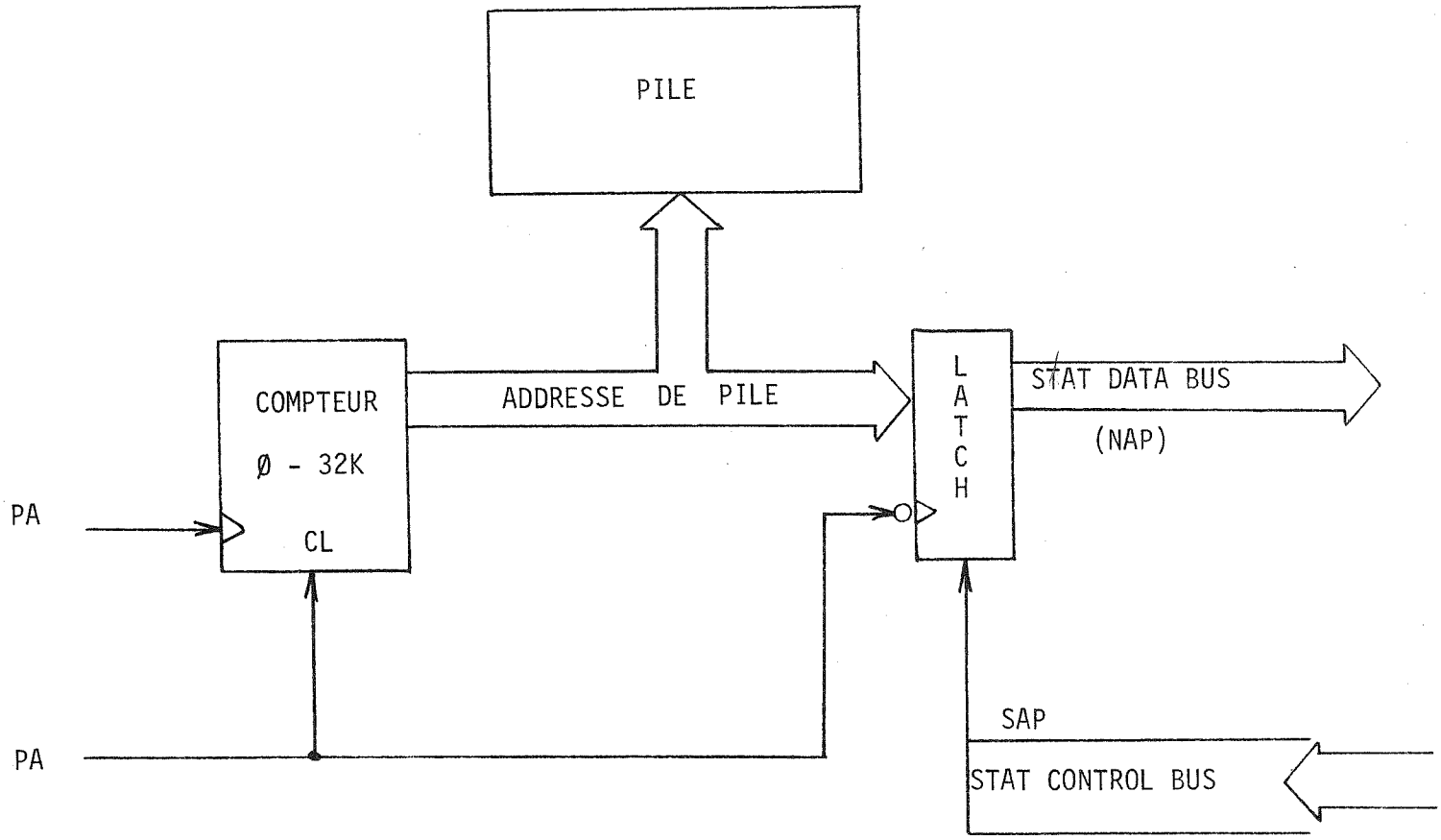


Figure 5.10 Schéma général pour la cueillette du nombre d'accès à la pile.

La deuxième mise en oeuvre est effectuée en matériel et est considérablement plus complexe. En fait, elle constitue un véritable banc d'essai pour le décodeur. Tel que montré à la Figure 5.11, le banc d'essai comprend la génération et le codage des séquences d'information, la génération et la mesure du bruit additif ainsi que la détection de séquences codées bruitées.

La procédure de comptage des erreurs se compose d'une mémoire FIFO recevant les séquences d'information, et d'un comparateur alimenté par la FIFO et la sortie du décodeur. Notons en passant que l'élaboration de ce banc d'essai est en quelque sorte extérieure au projet du décodeur proprement dit. Etant réalisé à plus de 90%, son parachèvement nécessite encore quelques mois d'effort. Aussi, les tests présentés dans ce rapport ont utilisé la mise en oeuvre en logiciel. Ces résultats font l'objet du chapitre suivant.

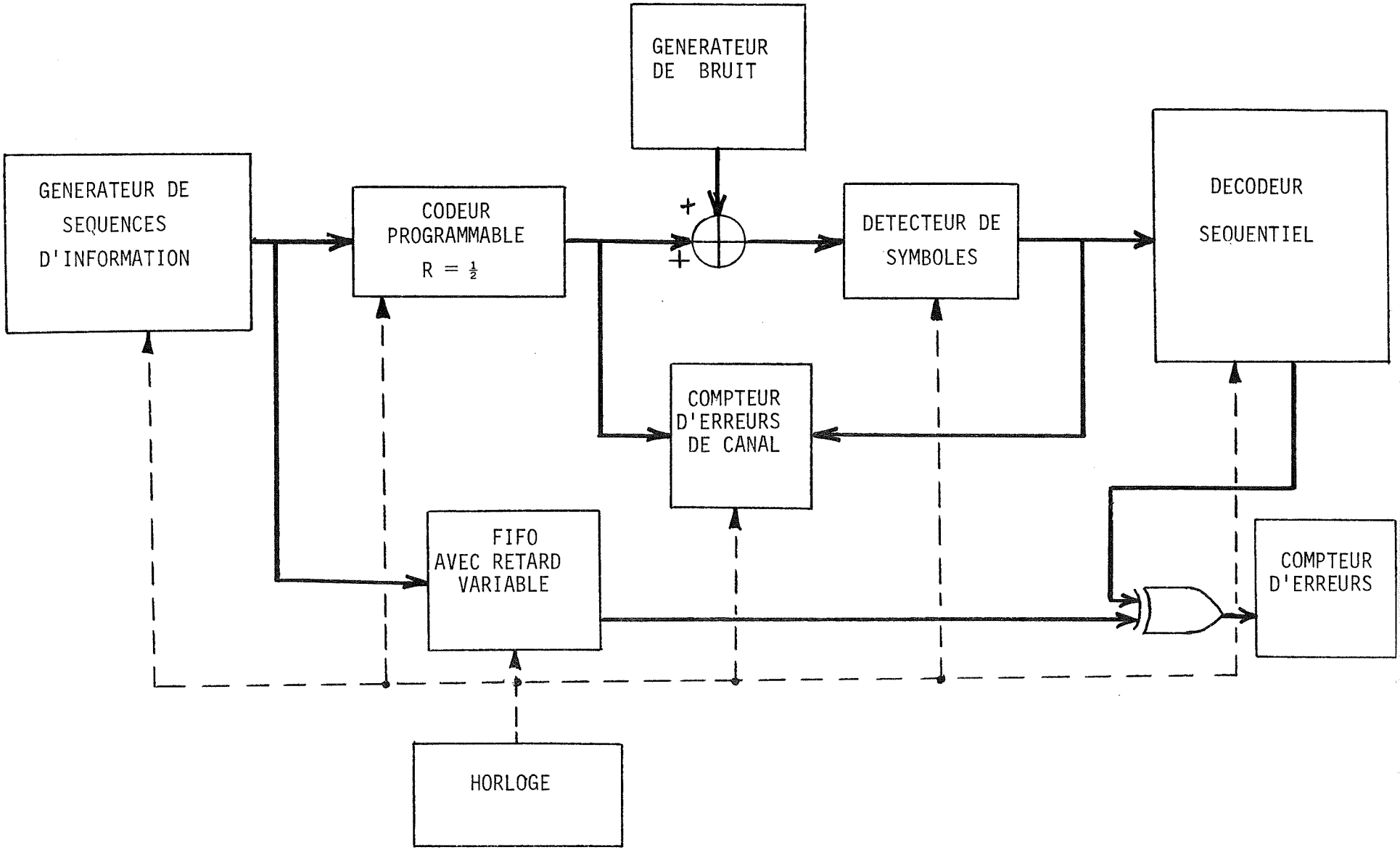


Figure 5.11 Schéma général du banc d'essai pour la mesure du nombre d'erreurs.

6. RÉSULTATS DES TESTS

Les différents modules servant à la cueillette des statistiques ont tous été montés et vérifiés, et tout le logiciel nécessaire à la mise en forme et à la gestion des données a été instauré et vérifié. Quant à la conduite des tests proprement dite, tel que mentionné à la section 5.2.8, elle nécessite au préalable la mise en place du banc d'essai. La réalisation du banc d'essai en matériel du décodeur s'est avérée être considérablement plus complexe que prévue initialement. En particulier, la simulation en matériel d'un canal bruité requiert une source de bruit à très haut débit et proprement étalonnée afin de contrôler avec précision les rapports signaux à bruit E_b/N_0. Ce banc d'essai en matériel n'ayant pu être parachevé à temps pour le montage d'une longue série de tests exhaustifs, des tests ont donc été effectués en utilisant une procédure entièrement en logiciel sous le contrôle du micro-ordinateur AIM-65. Rappelons encore que l'élaboration du banc d'essai en matériel est extérieure au projet du décodeur et conduit à la réalisation d'une entité physiquement distincte du décodeur. Le décodeur lui-même peut parfaitement fonctionner sans son banc d'essai; il suffit de l'alimenter par des séquences codées convolutionnellement et provenant d'un canal réel qui ajoute du bruit gaussien blanc. L'adjonction d'un banc d'essai en matériel est motivée par le désir d'avoir un montage complet auto-suffisant et parfaitement contrôlable pour analyser les performances et la dynamique du décodage.

6.1 Simulation des séquences

Tel que mentionné plus haut, les tests ont été conduits en utilisant une procédure entièrement logicielle sous le contrôle du microprocesseur AIM-65. A cet effet, en principe, il suffit d'alimenter le décodeur par des séquences codées et bruitées. En pratique et à cause de la

grande vitesse de décodage de la machine, cette procédure s'est avérée difficile à réaliser surtout au niveau de la génération du bruit numérique correspondant aux différentes valeurs de E_b/N_o. En effet, bien que disposant de générateurs de bruit pseudo-aléatoire logiciel, les séquences de bruit ne peuvent être engendrées au débit de quelque 1.5 Mbits/s correspondant à la vitesse de décodage du décodeur. Par conséquent, il a été décidé de ne pas utiliser une entrée directe des séquences provenant du canal dans le décodeur, mais plutôt de stocker ces séquences dans toute la mémoire disponible au AIM-65 (32 K octets), de les formatter en blocs et, sous le contrôle du AIM-65, de présenter ces blocs au décodeur à intervalles réguliers. Le décodeur fonctionnant considérablement plus rapidement que le AIM-65, les temps d'attente entre les blocs seront donc beaucoup plus longs que le temps de décodage, avec un facteur d'utilisation ("duty factor") de l'ordre de 1/1000.

Les séquences reçues du canal se composent du bruit et des séquences d'information codées. Deux alternatives se présentent pour les séquences d'information. La première consiste à ne transmettre que des "0", et la seconde consiste à transmettre une séquence d'information aléatoire, par exemple du texte. L'avantage de la séquence de "0" est qu'elle ne nécessite aucun codage puisque le codage d'une suite de L zéros donne une suite de 2L zéros. De plus, le décompte des erreurs est particulièrement simple : toute sortie non nulle correspond à une erreur. Avec cette méthode, il suffit donc de stocker dans la mémoire du AIM la séquence de bruit pseudo-aléatoire.

Pour la deuxième méthode, la séquence d'information étant aléatoire, il faut d'abord la coder et ensuite l'ajouter numériquement (modulo-2) au bruit. Un codeur logiciel a donc été écrit et a reçu comme séquence d'information quelques lignes de texte converti en code ASCII. La séquence codée résultante a été ajoutée modulo-2 aux séquences de bruit correspondant à différentes valeurs de E_b/N_o et le tout stocké dans la mémoire du AIM.

Quant au décompte des erreurs, il est effectué par comparaison entre la séquence décodée (en bits) et la séquence originale. Bien que d'un point de vue théorique les performances du décodeur ne sont pas dépendantes des séquences d'information particulière, seuls les résultats obtenus avec les séquences aléatoires sont donnés dans ce rapport.

6.2 Paramètres des tests

Les tests ont été conduits avec les paramètres suivants :

Code :

Taux de codage	R = 1/2
Longueur de contrainte	K = 24
Générateurs (non systématiques)	GA = 15 746 319
	GB = 11 552 015

Canal :

Canal binaire symétrique (quantification dure)

Rapport signaux à bruit :

E_b/N_o	R/R_{comp}
5.0 (dB)	0.934
5.5 (dB)	0.864
6.0 (dB)	0.803
6.5 (dB)	0.504
7.0 (dB)	0.704
7.5 (dB)	0.664

Séquences :

Longueur des blocs : 488 + 23 = 511 bits

Nombre de blocs : 250

Statistiques recueillies par bloc :

- Cumulative de la métrique accumulée
- Cumulative du temps de calcul
- Cumulative du nombre de cycles positifs
- Cumulative du nombre de cycles négatifs
- Cumulative du nombre de cycles de sous-pile
- Cumulative du nombre d'accès à la pile
- Cumulative du nombre de cycles négatifs + nombre de cycles des sous-piles
- Cumulative du nombre d'erreurs.

6.3 Résultats

A l'exception de la cumulative du nombre d'erreurs, les résultats correspondant aux statistiques ci-dessus sont donnés aux Figures 6.1 à 6.7. Le nombre d'erreurs ayant été constamment égal à zéro pour tous les tests, la cumulative de cette statistique n'est pas fournie. Quant aux statistiques non recueillies mais définies au Chapitre 5 (i.e. Taille maximale de la file d'attente dans le tampon d'entrée, et dans le tampon de sortie; temps d'attente) elles deviennent ici irrelevantes en raison de la procédure logicielle utilisée.

La cumulative de la métrique accumulée par bloc est montrée à la Figure 6.1. On voit que la valeur finale ainsi que la variation de la métrique du chemin décodé est d'autant plus faible que le rapport signal à bruit augmente. La valeur maximum atteinte est inférieure à 3 900 et ne nécessite donc que 12 bits, alors que 14 bits ont été prévues à cette fin. Pour la valeur E_b/N_o = 5 dB, la métrique varie entre 2450 et 3200, indiquant ainsi une grande variabilité due au bruit. Naturellement, cette variabilité devrait encore augmenter si E_b/N_o diminuait encore jusqu'à 4.63 dB, valeur qui correspond à R/R_{comp} = 1.

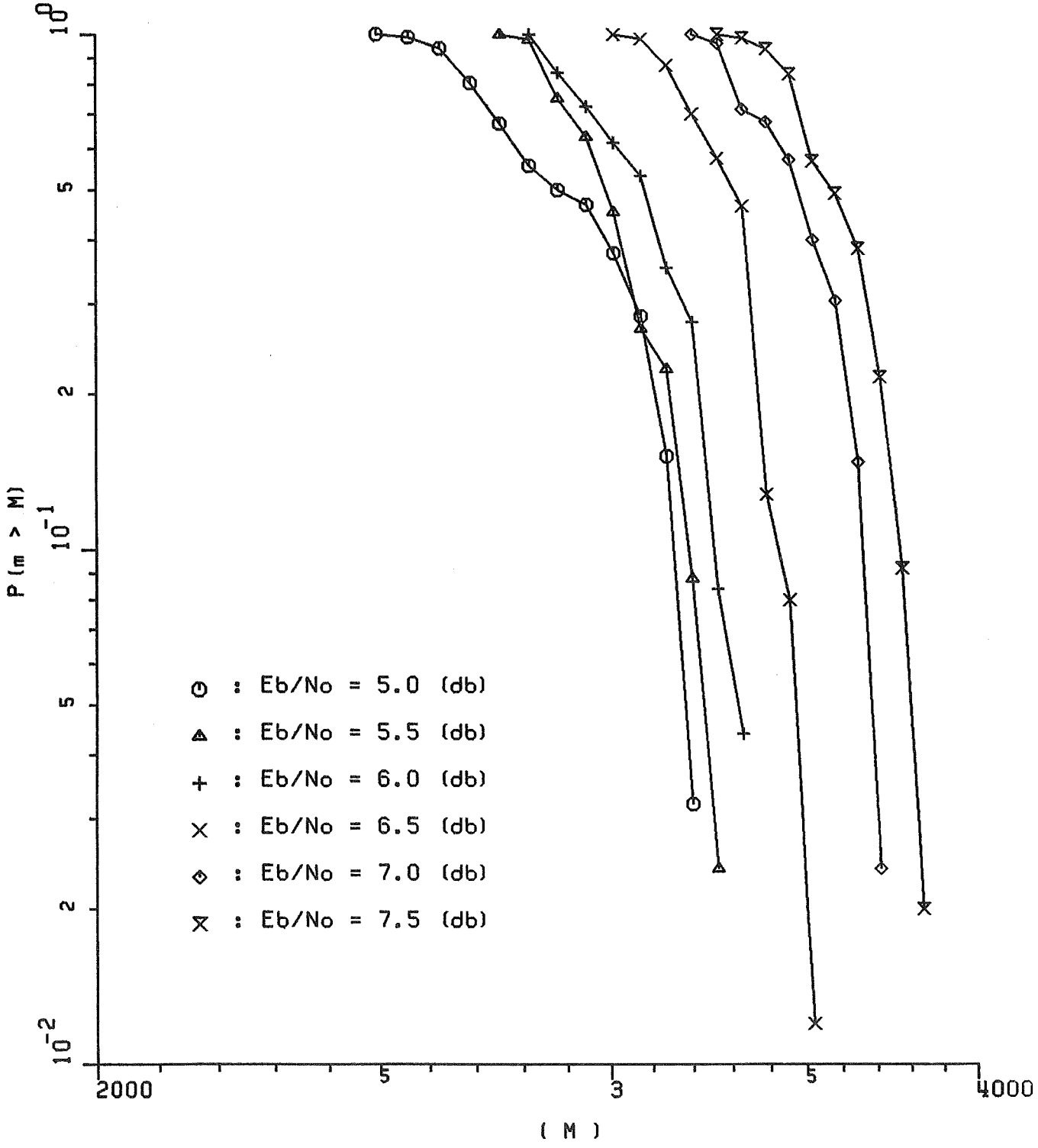


Figure 6.1 Cumulative de la métrique totale en fin de bloc.

La distribution du temps de calcul actif est donnée à la Figure 6.2. On rappelle que cette statistique correspond à l'activité de décodage proprement dite, sans attente ni extensions de chemins auxiliaires. La Figure 6.2 indique qu'à l'exception de E_b/N_o = 5.0 dB, il n'y a que très peu de variation du temps de calcul indiquant un décodage de bloc assez stable, c.a.d. sans grands retours en arrière. Pour E_b/N_o = 7 dB et 7.5 dB, le temps de calcul minimum est 530 microsecondes, alors que pour les valeurs de E_b/N_o inférieures à 7 dB ce temps minimum est égal à 580 microsecondes indiquant un plus grand nombre de retours en arrière, et indiquant également la présence de blocs ayant nécessités exactement le même effort minimal pour E_b/N_o variant de 5.0 dB à 6.5 dB. Comme on aurait pu s'y attendre à E_b/N_o = 5 dB, le décodeur a été constamment actif pendant beaucoup plus longtemps, avec un temps de calcul dépassant quelque 900 microsecondes. Une telle variabilité est bien sûr une indication de recherches arrières prolongées avec une accumulation concomittante dans le tampon d'entrée. La répercussion de cette variabilité de l'effort de calcul actif se fera donc sentir aussi bien dans le tampon d'entrée que dans le tampon de sortie.

Les cumulatives concernant les différents aspects de l'effort de calcul du décodeur sont fournies aux Figures 6.3 à 6.6 inclusivement. La distribution du nombre de cycles positifs est montrée à la Figure 6.3. Pour toutes les valeurs de E_b/N_o, au moins 448 cycles positifs ont été effectués par bloc. Avec une probabilité égale à 0.5 et 0.02, au moins 512 cycles ont été effectués pour les valeurs de E_b/N_o égales à 5.0 dB et 6.0 dB respectivement. Cette inconsistence apparente peut s'expliquer en partie par le nombre de retours en arrière beaucoup plus fréquents à 5 dB plutôt qu'à 6 dB, et par le fait que durant ces recherches arrières, les chemins prolongés peuvent aussi avoir une métrique de branche positive. Cependant, un examen plus précis de ce phénomène requiert des tests plus exhaustifs avec un plus grand nombre de blocs que 250.

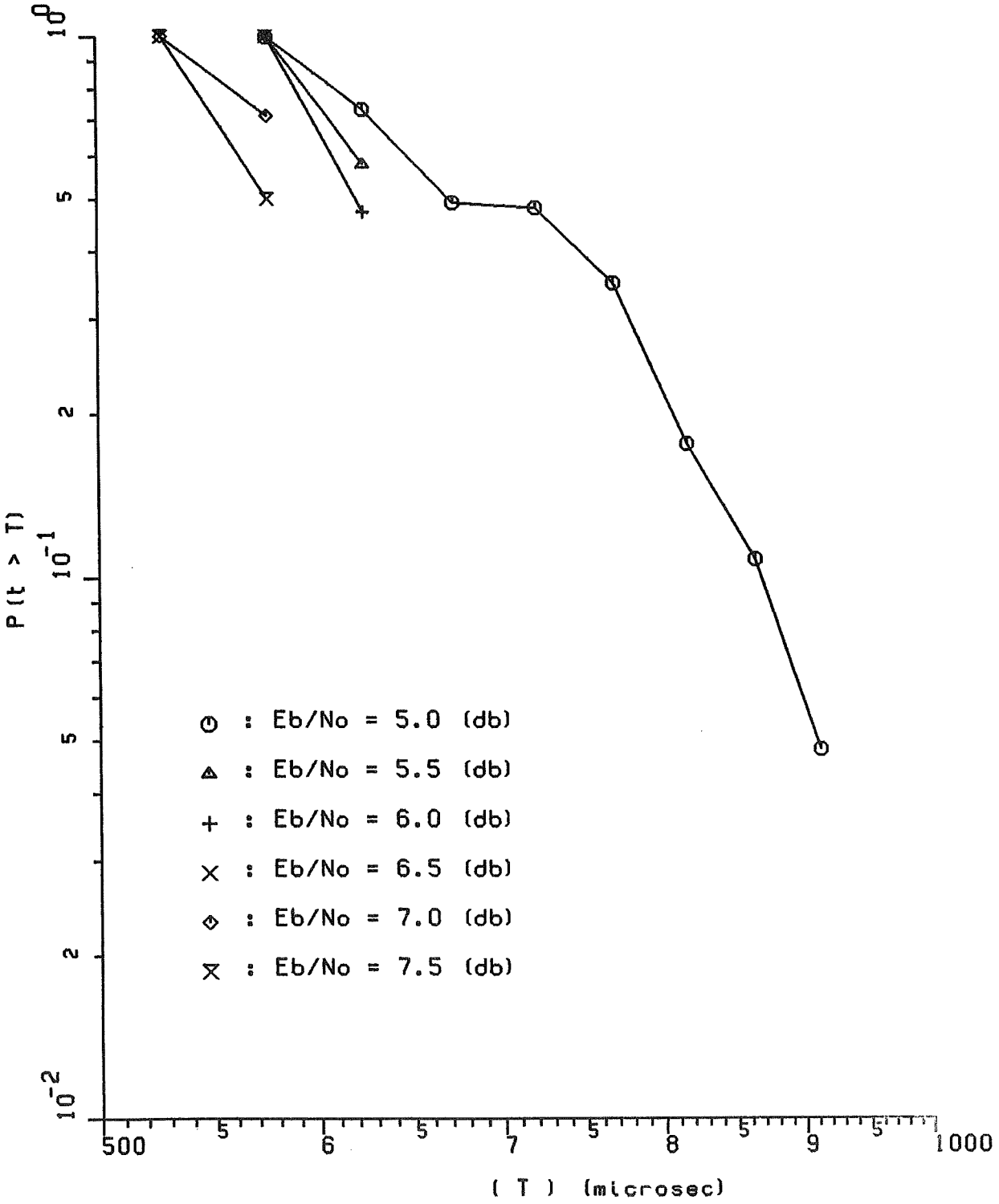


Figure 6.2 Cumulative du temps de calcul actif par bloc.

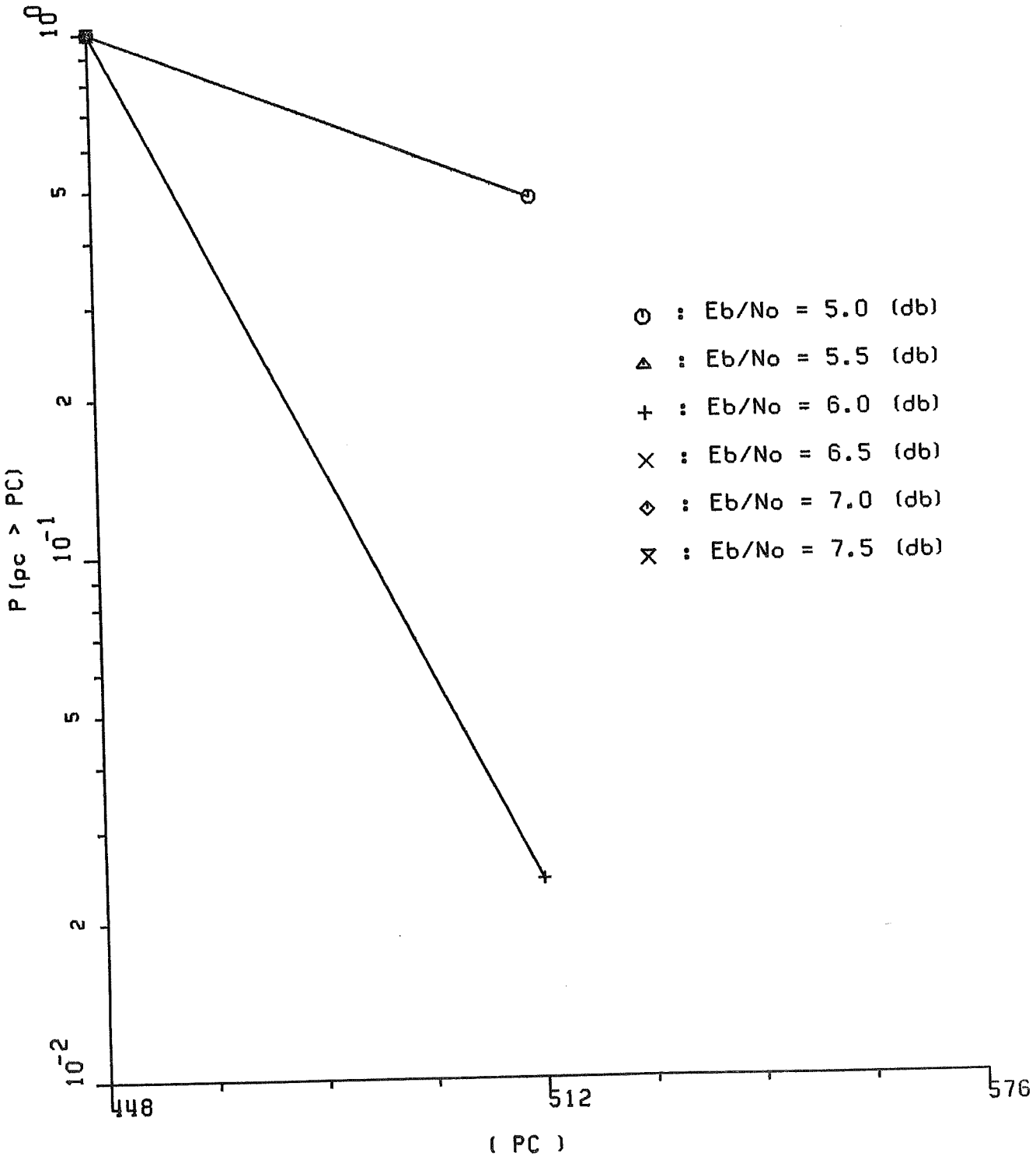


Figure 6.3 Cumulative du nombre de cycles positifs par bloc.

La cumulative du nombre de cycles négatifs est donnée à la Figure 6.4 et montre que ce nombre varie en sens inverse du rapport E_b/N_o, tout comme il fallait s'y attendre. Cependant, la figure indique une démarcation nette entre la valeur E_b/N_o = 5 dB et toutes les autres, indiquant un comportement hautement non linéaire, voire même un décrochage du nombre de cycles négatifs à mesure que R/R_{comp} se rapproche de 1. Des tests sur un plus grand nombre de blocs sont requis pour distinguer les figures entre elles et de plus un examen de la Figure 6.4 indique que l'incrément $\Delta = 64$ choisi pour cette statistique est probablement trop grand.

La cumulative du nombre de cycles de sous-piles par bloc de code est donnée à la Figure 6.5 et complémente en quelque sorte celle du nombre de cycles négatifs. Ici toutes les courbes se distinguent les unes des autres et montrent une assez grande variabilité de cette statistique en fonction de E_b/N_o. Certaines particularités et inconsistences locales semblent exister. En effet, on pourrait en première approximation penser que le nombre de cycles de sous-piles devrait toujours décroître avec E_b/N_o croissant. En fait, le comportement du nombre de cycles de sous-piles est étroitement lié à la dynamique des métriques de branche, et à la procédure de recherche de la sous-pile maximum non vide. Ici, la recherche de cette sous-pile maximum non vide s'effectue toujours à partir de la plus grande sous-pile atteinte au cours du décodage du bloc. A partir de cette valeur maximum extrême, l'algorithme balaye les sous-piles inférieures par intervalles de métriques égales à 8. Par conséquent, plus les chutes de métriques du chemin correct seront grandes, plus il faudra de cycles de sous-piles pour atteindre la sous-pile où réside le noeud à prolonger. Comme la valeur de la chute de métrique de branche augmente avec E_b/N_o, on doit donc s'attendre à toujours avoir de nombreux cycles de sous-pile, même pour de grandes valeurs de E_b/N_o. Quant à la non régularité et inconsistences locales des courbes de la Figure 6.5 entre elles, elles sont fort probablement dues au trop petit nombre de blocs testés.

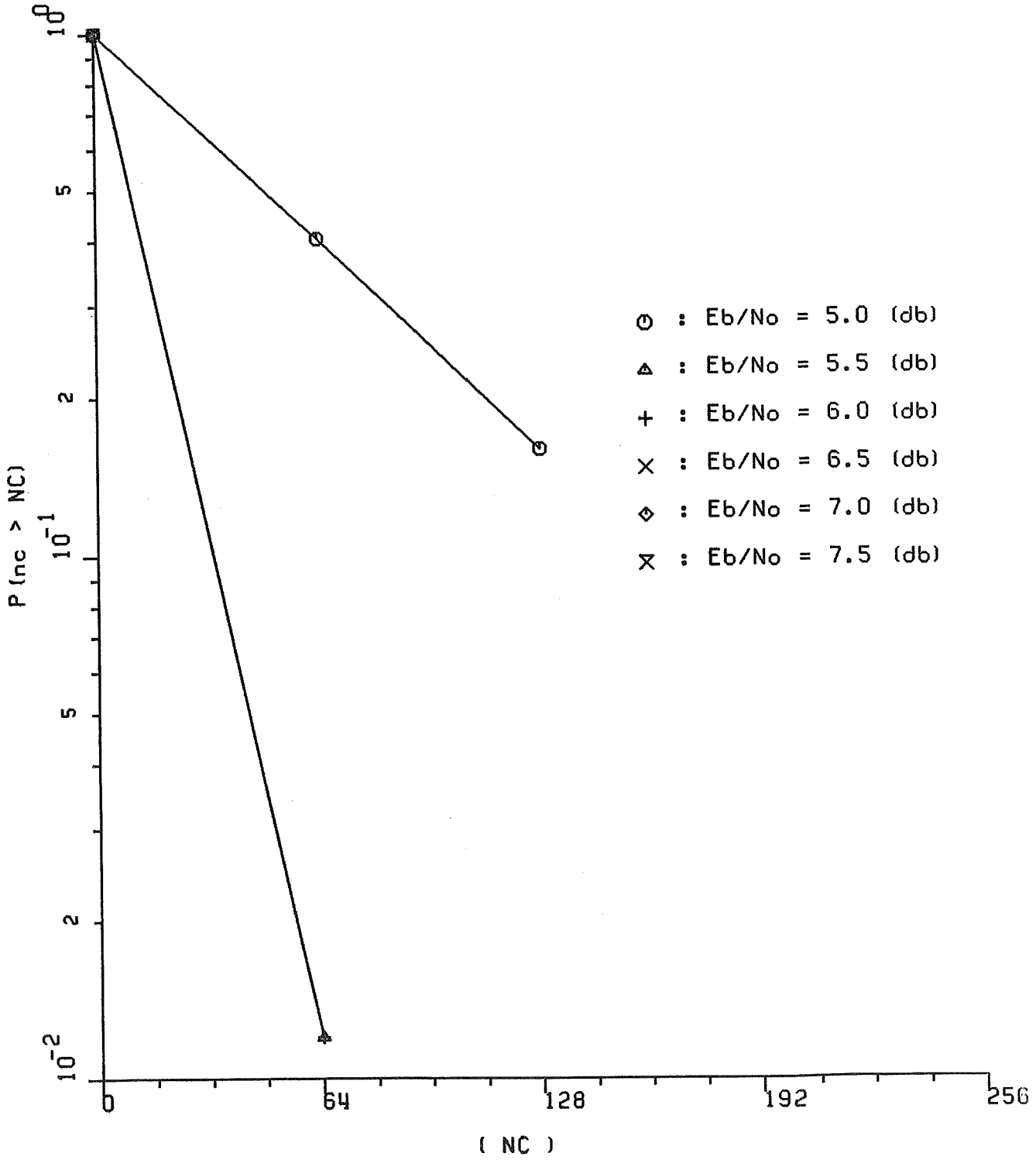


Figure 6.4 Cumulative du nombre de cycles négatifs par bloc.

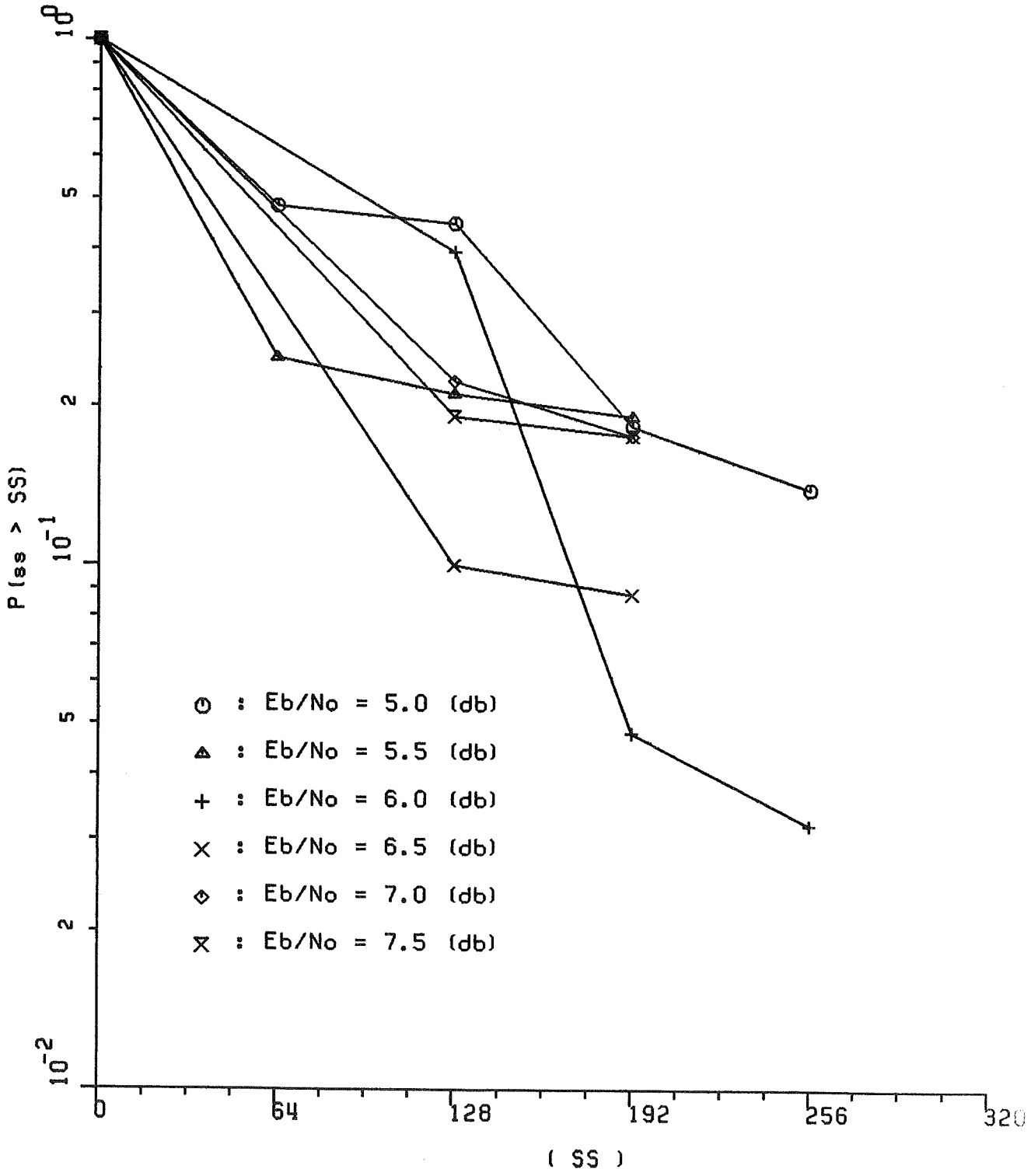


Figure 6.5 Cumulative du nombre de cycles de sous-piles par bloc.

Comme chaque recherche arrière implique des cycles de sous-pile et des cycles négatifs, un comportement global de la distribution des calculs durant les recherches arrières est donné à la Figure 6.6 où la cumulative de ces deux évènements est fournie.

La cumulative du nombre d'accès à la pile est fournie à la Figure 6.7. Cette distribution donne la cumulative de la taille de la pile requise pour le décodage d'un bloc de 511 bits et donne donc de précieux renseignements sur l'espace mémoire à prévoir pour la pile. Tout d'abord, la Figure 6.7 indique que le taille de la pile croît à mesure que E_b/N_o décroît, comme on pouvait le prévoir. Cependant, la taille maximum de la pile demeure modeste, ne dépassant 896 entrées qu'avec une probabilité d'environ 1%. Une entrée dans la pile correspondant à 77 bits, l'espace mémoire est donc 896 x 77 = 68992 bits ou 8624 octets. Il est bon de rappeler que la procédure du fanion a été utilisée afin de ne stocker qu'une seule entrée dans la pile dans le cas de cycles positifs. Etant donné la distribution du nombre de cycles positifs (voir Figure 6.3), on peut donc apprécier l'énorme économie de mémoire réalisée par cette procédure. Notre décodeur comportant une pile de 16 000 entrées, l'étendue du surdimensionnement ne pourra être évaluée qu'après avoir effectué des tests sur des blocs de longueur 1024 bits.

Tel que mentionné plus haut, les statistiques ont aussi comporté le décompte du nombre d'erreurs par bloc. Aucune erreur n'a été observée durant le décodage des quelque 250 x 480 = 122 000 bits effectué pour chacune des 6 valeurs de E_b/N_o, soit 732 000 bits, témoignant ainsi de la puissance de correction d'erreur du décodeur séquentiel. Cependant, une évaluation plus complète de la machine ne peut être effectuée que par des tests sur des centaines de millions de bits. Pour procéder à de tels tests, seule la méthode de test en matériel est adéquate. Le parachèvement préalable du banc d'essai matériel est donc indispensable à la conduite de ces tests.

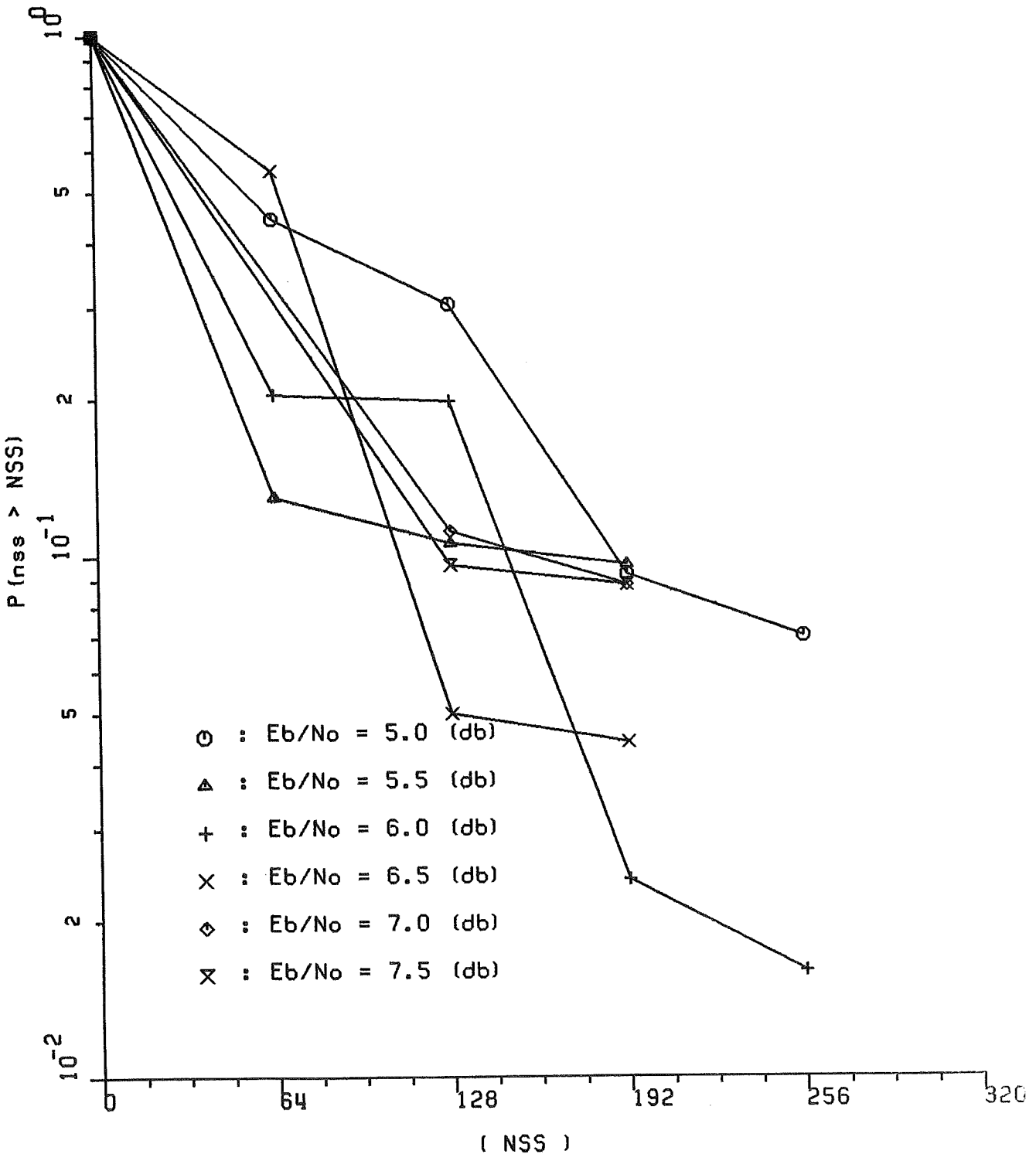


Figure 6.6 Cumulative de la somme de cycles de sous-piles et de cycles négatifs par bloc.

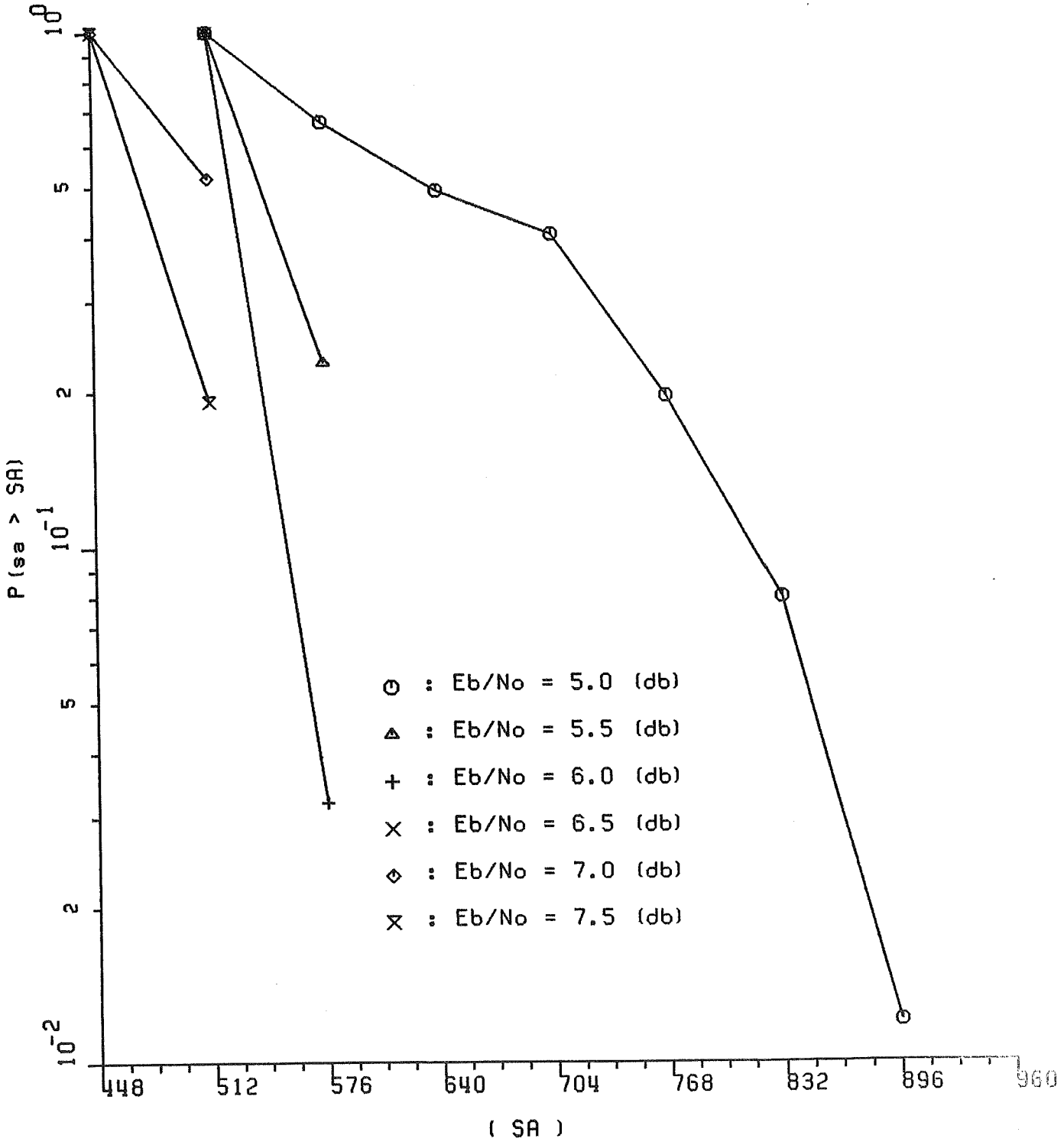


Figure 6.7 Cumulative du nombre d'accès à la pile.

7. CONCLUSIONS

Le prototype de décodage séquentiel utilisant l'algorithme à pile est pour l'essentiel parachevé et tous les appareillages nécessaires à la mesure de ses tests de performance réalisés et vérifiés. Tel que mentionné au Chapitre 6, ces appareillages de collecte de statistiques concernent essentiellement les paramètres de la dynamique de l'effort du décodage. Les tests de performance d'erreur nécessitent la mise en place d'un banc d'essai étalonné qui, n'étant pas unique au test du décodeur séquentiel, est en quelque sorte extérieur au projet.

Tout comme dans la plupart des projets de recherche, un certain nombre d'imprévus est apparu tout au long de la réalisation du projet. Tout d'abord, bien que disposant d'un prototype ayant fonctionné pendant de nombreuses heures, et étant donc en état de fonctionnement au début du présent projet (septembre 1983), une multitude de pannes du décodeur est apparue entre novembre 1983 et mars 1984. Ces pannes, dues à la défaillance des circuits et mémoires (fort probablement reliées à la "mortalité infantile" des circuits) exigeaient chacune de très gros efforts pour être réparées et ont donc causé des retards considérables au bon déroulement du projet. En fait, pendant plusieurs mois, l'effort a porté exclusivement sur le dépannage du décodeur, déplaçant d'autant les travaux sur la procédure de tests proprement dite.

Les résultats des tests préliminaires décrits au Chapitre 6 illustrent la variance qui existe entre l'analyse théorique, la simulation à l'ordinateur de l'algorithme et sa réalisation matérielle. En effet, dans l'analyse théorique de l'effort de calcul, un "calcul" est défini comme l'extension d'un noeud et l'insertion dans la pile de ses descendants. Aucune distinction n'est portée sur le type d'extension : progression en avant, recherche arrière, ... La simulation à l'ordinateur permet, bien

qu'imparfaitement de faire cette distinction. Cependant dans la réalisation matérielle, les différents aspects d'un "calcul" prennent toute leur importance par les notions de cycles positifs, négatifs et de sous-piles. Ainsi, il apparaît que les recherches arrières peuvent être considérablement plus coûteuses par leur nombre de cycles de sous-pile que toute progression en avant. Un design efficace de décodeur séquentiel devra donc réduire au maximum le nombre de cycles de sous-pile. Une façon simple d'y arriver est de ne pas faire commencer les cycles de sous-pile à partir de la sous-pile maximum atteinte, mais à partir de la valeur de la dernière sous-pile utilisée au cours de l'extension précédente.

Bien qu'importantes et intéressantes en soi, pour une réalisation matérielle, les notions de "calcul" sont probablement moins importantes que celles de temps de calcul actif. D'un point de vue pratique, le pourcentage du temps pendant lequel le décodeur est vraiment actif est des plus importants et a une influence directe sur la taille de la file d'attente dans le tampon d'entrée. Etant donné la grande vitesse de décodage de notre prototype, les temps d'attente sont mis à profit dans l'option de décodeur "toujours actif". Cependant, la mise en application de cette option doit être faite avec grande prudence, car si les temps d'attente étaient trop longs et trop fréquents, un décodeur toujours actif ferait très vite déborder la pile.

Concernant les tailles de la pile et des tampons d'entrée et de sortie, les résultats indiquent que les espaces mémoires prévus ont été surdimensionnés. Rappelant que les tests effectués ne sont pas exhaustifs, toute conclusion ferme sur ces espaces mémoires est prématurée.

Enfin, le décodeur a montré un aperçu de sa puissance de décodage en corrigeant toutes les erreurs qui lui ont été présentées. D'autres tests plus longs et plus exhaustifs permettront de tracer sa véritable courbe de performance.

En résumé, le projet a été mené à bon terme, et nous disposons à présent d'un prototype de recherche original de décodage séquentiel utilisant l'algorithme à pile. Au mieux de nos connaissances, ce prototype est à l'avant-garde de toute autre réalisation de décodage séquentiel utilisant l'algorithme à pile. Un certain nombre de travaux et de mises au point doivent encore être effectués avant d'élaborer une campagne de tests exhaustifs. En particulier, la réalisation du banc d'essai étalonné permettra la conduite de tests de performance complets, et de là ouvrir la voie à des conceptions de décodeurs séquentiels pratiques, efficaces et économiques.

RÉFÉRENCES

[1] F. JELINEK, "A Fast Sequential Decoding Algorithm Using a Stack", IBM Journal of Research and Development, Vol. 13, Nov. 1969.

[2] A.J. VITERBI, "Convolutional Codes and Their Performance in Communication Systems", IEEE Trans. on Com. Tech., Vol. COM-19, Oct. 1971.

[3] V. BHARGAVA, D. HACCOUN, R. MATYAS, P. NUSPL, "Digital Communications by Satellite", John Wiley, New York, Oct. 1981.

[4] D. HACCOUN, M. FERGUSON, "Generalized Stack Algorithm for the Decoding of Convolutional Codes", IEEE Trans. on Inf. Theory, Vol. IT.21, Nov. 1975, pp. 638-651.

[5] D. HACCOUN, M. DUFOUR, "Stack an Input Buffers Overflows of Stack Decoding Algorithms", Book of Abstracts, IEEE International Symposium on Information Theory, Santa Monica, California, Feb. 1981.

[6] D. HACCOUN, "Problèmes de débordement de décodeurs séquentiels à pile", 8e Colloque GRETSI sur le traitement du signal et ses applications, Nice, France, juin 1981, pp. 919-923.

[7] D. HACCOUN, CHEN NAIYUN, "Empirical Investigation of High Rate Sequential Decoding", Book of Abstracts, IEEE Trans. on Information Theory, Les Arcs, France, June 1982, p. 110.

[8] D. HACCOUN, CHEN NAIYUN, "Variants of the Stack Algorithm for the Decoding of High Rate Codes by Sequential Decoding", 1st Canadian and International Satellite Communications Conference, Ottawa, June 1983, pp. 21.4.1-21.4.5.

[9] D. HACCOUN, "A Markov Chain Approach to the Sequential Decoding Metric", IEEE Trans. on Information Theory, Vol. IT-26, Jan. 1980, pp. 109-113.

[10] P.Y. PAU, D. HACCOUN, "Sequential Decoding with ARQ", Book of Abstracts, IEEE International Symposium on Information Theory, St.Jovite, Quebec, Sept. 1983, pp. 40-41.

[11] I.M. JACOBS et E.R. BERLEKAMP, "A Lower Bound to the Distribution of Computation for Sequential Decoding", IEEE Trans. on Information Theory, Vol. IT-13, April 1967, pp. 167-174.

[12] J.M. WOZENCRAFT and I.M. JACOBS, "Principles of Communications Engineering", J. Wiley, New York, 1965.

[13] D. HACCOUN, "A Branching Process Analysis of the Average Number of Computations of the Stack Algorithm", IEEE Trans. on Information Theory, Vol. IT-30, No. 3, pp. 497-508, May 1984.

[14] C. MORIN, "Conception d'un décodeur séquentiel rapide", Mémoire de maîtrise en Sciences appliquées, Département de génie électrique, Ecole Polytechnique de Montréal, Nov. 1980, 286 p.

ANNEXE

Dans cette annexe, extraite de [14] les 15 états du décodeur sont décrits en détail avec leurs branchements aux autres états de la machine et les conditions qui régissent ces branchements.

ETAT	DESCRIPTION	BRANCHEMENT	CONDITION
1	Extension de la branche "0" . Stocker les symboles reçus dans SEQS . Stocker le noeud dans la pile si la métrique de branche est négative	2	Inconditionnel
2	Extension de la branche "1" . Conserver le noeud dont la métrique de branche est positive (s'il en est un) . Stocker le noeud dans la pile si la métrique de branche est négative . Accéder au prochains symboles dans SEQS ou le tampon d'entrée	3 9 6 1	Chercher le sommet de la pile si les deux métriques de branche sont négatives Sinon, et si la séquence sollicitée au tampon d'entrée n'est pas disponible, pousser un chemin auxiliaire Sinon, et si l'on est dans la queue autrement
3	Début de la recherche du sommet de la pile . Si la sous-pile de plus haut niveau déjà examinée est vide, regarder la précédente	3 4	Si tel est le cas, réitérer autrement, progresser
4	Accès au suivant du noeud dans sa sous-pile . Copie des paramètres du dernier noeud entré dans la sous-pile de plus haut niveau dans le registre TM . Initialisation des variables de calcul pour 5	5	Inconditionnel

ETAT	DESCRIPTION	BRANCHEMENT	CONDITION
5	Bouclage dans la sous-pile . Si le noeud examiné n'est chaîné à aucun autre ou que l'algorithme quantifié est employé: . aller chercher l'adresse du suivant de son suivant . conserver ce noeud si sa métrique est supérieure à celle du noeud TM (meilleure évaluation à date) . Sinon réorganiser le chaînage pour escamoter le noeud choisi et . initialiser le décodeur si on est en fin de queue . accéder à la prochaine séquence reçue si on ne l'est pas	5 7 9 1 6	Si le noeud examiné est chaîné à un autre et que l'algorithme quantifié n'est pas sollicité Sinon et si on en a terminé avec la queue Sinon et si on est en recherche avan. et que le tampon d'entrée est vide Sinon et si on est dans le bloc Autrement (on est dans la queue)
6	Traitement d'un noeud dans la queue . Stocker les symboles reçus dans SEQS . Si le noeud à étirer est terminal: . initialiser le décodeur . Sinon: . conserver le noeud calculé si la métrique de branche est positive . accéder aux prochains symboles dans SEQS ou le tampon d'entrée . stocker le noeud dans la pile si la métrique de branche est négative	7 3 9 6	Si le noeud à étirer est terminal Sinon et si la métrique de branche est négative Sinon et si on est en recherche avant et que le tampon d'entrée est vide et que le noeud étiré n'est pas terminal Sinon on réitère avec le noeud étiré

ETAT	DESCRIPTION	BRANCHEMENT	CONDITION
7	Récupération du noeud terminal dans le bloc . Si le noeud examiné est dans la queue: . aller chercher son précédesseur dans la pile . Sinon: . initialiser les registres . lancer le récupérateur automatique du message décodé	7 8	Si l'on se trouve toujours dans la queue autrement
8	Attente	8 1	Si le tampon d'entrée est vide Sinon
9	Début de l'extension auxiliaire . Conserver l'état des registres de façon à pouvoir reprendre les opérations là où on les a laissées . Chercher la sous-pile non vide de plus haut niveau	6 1 9 10	Si le tampon d'entrée est non vide et que l'on est dans la queue Si le tampon d'entrée est non vide et que l'on est dans le bloc Sinon et si la sous-pile visitée est vide autrement
10	Recherche d'un candidat à l'extension auxiliaire . Eliminer la candidature des noeuds à une profondeur telle que les symboles reçus ne sont pas disponibles . Procéder de suivant en suivant dans la sous-pile	11 10 9	Si le noeud est éligible Sinon examiner le noeud auquel il est chaîné dans la sous-pile Si aucun noeud de la sous-pile est éligible, examiner la sous-pile de niveau inférieur
11	Recherche du meilleur candidat . Equivalent de 5		Voir 5
12	Extension de la branche "0" . Equivalent de 1		Voir 1

ETAT	DESCRIPTION	BRANCHEMENT	CONDITION
13	Extension de la branche "1" . Equivalent de 2	Voir 2	Si le tampon d'entrée est non vide durant l'exécution de 13 ou 14, on branche à 15
14	Extension de la branche "0" dans la queue . Equivalent de 6		Voir 6
15	Restitution des informations de décodage normal . Récupérer l'information stockée en 9 . Remettre tout en place (équivalent à un retour de sous-routine)	1 6	Si l'on était dans le bloc autrement