Architecture de turbo-décodeur en blocs entièrement parallèle pour la transmission de données au-delà du Gbit/s

Christophe Jego, Patrick Adde, Camille Leroux

Architecture de turbo décodeur en blocs entièrement parallèle pour la transmission de données au-delà du Gbit/s

Christophe JEGO*, Patrick ADDE*, Camille LEROUX*

Résumé

Cet article présente une nouvelle architecture atteignant de très haut débit pour le turbo décodage de codes produits. Ce type d'architecture est capable de décoder des codes produits reposant sur des codes binaires de type BCH ou des codes m_aire de type Reed Solomon. Son principal atout est qu'elle permet l'élimination des plans mémoires associés aux matrices générées par un code produit entre les différentes demi-itérations pour le turbo décodage. En fait, la solution architecturale que nous détaillons dans ce papier offre de nouvelles opportunités pour l'application des turbocodes dans des systèmes nécessitant des débits supérieurs au Gbit/s comme les systèmes de transmission sur fibre optique.

Mots clés : Code correcteur d'erreurs, Turbocodes en blocs, Décodage entièrement parallèle, Architecture à très haut débit.

ABOVE GBIT/S DATA TRANSMISSION USING FULL-PARALLEL BLOCK TURBO DECODER ARCHITECTURE

Abstract

This paper presents a new circuit architecture for turbo decoding, which achieves ultra high data rates when using product codes as error correcting codes. This architecture is able to decode product codes using binary BCH or m-ary Reed Solomon component codes. The major advantage of our full-parallel architecture is that it enables the memory block between each half-iteration to be removed. In fact, the proposed architecture opens the way to numerous applications such as optical transmission. In particular, our block turbo decoding architecture can support optical transmission at data rates above Gbit/s.

Key words : Forward error correction, Block turbo codes, Full-parallel decoding, Ultra high-speed architecture.

* GET/ENST Bretagne, CNRS TAMCIC UMR 2872, Technopôle de Brest-Iroise, 29285 Brest cedex, France ; firstname.lastname@enst-bretagne.fr

Sommaire

I. INTRODUCTION

Les codes correcteurs d'erreurs (codage canal) sont une des solutions permettant d'améliorer la qualité des communications numériques. Le principe du codage canal est d'introduire de la redondance dans la séquence d'information binaire afin de corriger les erreurs de transmission durant la réception de l'information. Deux grandes classes de code correcteur d'erreurs existent : les codes convolutifs et les codes en blocs. Au début des années 90, une nouvelle famille de code correcteur d'erreurs a été découverte par C. Berrou [1] : les turbocodes. Cette famille de codes correcteurs d'erreurs est construite par concaténation de codes élémentaires. Les turbocodes sont le résultat de deux innovations majeures : la concaténation de deux codes pour le codage et le décodage itératif. Le décodage itératif est appliqué à des décodeurs élémentaires à entrées et sorties pondérées (EPSP). Ce type de décodage est une solution qui fournit de bonnes performances tout en nécessitant un niveau de complexité raisonnable. Le concept général de décodage itératif appliqué à des décodeurs élémentaires EPSP a ensuite été appliqué aux codes produits [2] et aux codes LDPC [3]. De plus, le principe turbo a été étendu à l'ensemble des fonctionnalités constituant le récepteur d'un système de communications numériques telles que la turbo-détection dans les domaines des canaux sélectifs en fréquence ou des canaux MIMO, la turbo-égalisation ou la turbo-modulation.

Depuis le début des années 2000, les turbocodes ont été adoptés par de nombreuses applications. Ce type de codes est par exemple particulièrement intéressant pour les systèmes de communications cellulaires. Les turbocodes font ainsi partie intégrante des normes UMTS, CDMA2000, DVB-RCS, IEEE802.16 et HyperMAN. Ils favorisent l'augmentation des débits de transmission et l'amélioration de la qualité de service. Par ailleurs, des études sont actuellement en cours pour introduire des turbocodes dans des systèmes de stockage de l'information tels que les disques durs ou les DVD ou dans des systèmes de transmission sur fibre optique. Dans un contexte d'évolution vers le haut débit, la transmission sur fibre optique est une technologie de pointe qui favorise l'évolution de l'infrastructure de desserte des contenus. Les codes correcteurs d'erreurs sont utilisés dans les systèmes de communication optiques pour corriger la dégradation en terme de taux d'erreur binaire due aux phénomènes de dispersion et de non-linéarité de la fibre optique. Les premiers codes correcteurs d'erreurs utilisés dans les communications optiques [4] sont les codes Reed Solomon. Par exemple, la norme ITU-T G.709 [5] qui spécifie les interfaçages et les rendements pour les réseaux d'accès optique, contient un code Reed Solomon comme code correcteur d'erreurs. Le code retenu est le code Reed Solomon (255,239) comprenant des symboles sous forme d'octets dont la

capacité de correction maximale est de 8 octets et la capacité de détection est de 16 octets. Dans ce cas, le gain de codage est alors d'environ 6 dB. Ces dernières années d'autres codes correcteurs d'erreurs plus puissants ont été proposés pour améliorer le gain de codage. Parmi eux, les turbocodes en blocs ont un gain de codage potentiel autour de 10 dB [6, 7]. Typiquement, les turbocodes en blocs ont des performances se situant environ à 1 dB de la limite de Shannon.

Les nouvelles générations de réseaux d'accès optique qui se caractérisent par une montée en débit en ligne impliquent l'implémentation d'architectures de codes correcteurs d'erreurs atteignant des débits très élevés. Des solutions architecturales de décodeur Reed Solomon (255, 239) ont été proposées pour des débits de 40 Gbit/s [8] et de 80 Gbit/s [9]. En 2002, une nouvelle architecture pour le turbo décodage de codes produits a été présentée [10]. Cette solution architecturale permet d'atteindre des débits autour de 6 Gbit/s. En 2005, Mitsubishi Electric a annoncé le développement d'un circuit de turbo décodage de codes produits pour des transmissions optiques à 10 Gbit/s [11]. Dans ce contexte, nous présentons dans cet article, un nouveau type d'architecture pour le turbo décodage de codes produits. Ce type d'architecture élimine les plans mémoires associés aux matrices générées par le code produit. En effet dans les architectures traditionnelles, des plans mémoires sont nécessaires pour sauvegarder les informations échangées au cours des itérations du processus de turbo décodage. Ainsi, les architectures obtenues traitent des données à très haut débit pouvant dépasser le 10 Gbit/s avec une latence d'exécution bien inférieure aux architectures proposées jusqu'alors. Enfin, il est à noter que ce type d'architecture peut s'appliquer pour le turbo décodage de codes produits utilisant des codes binaires de type BCH ou des codes m_aire de type Reed Solomon [12].

Dans une première partie, nous allons rappeler le principe de turbo décodage des codes produits. Puis, nous présentons un état de l'art sur les architectures de turbo décodeurs dédiées aux codes produits. Nous détaillons ensuite notre solution architecturale reposant sur un turbo décodage entièrement parallèle. De plus, le lecteur peut consulter la référence [13] qui décrit brièvement le principe sur lequel repose notre approche. Enfin, nous proposons une architecture de turbo décodeur de codes produits pour le très haut débit (>10 Gbit/s). Cette solution architecturale repose sur un traitement entièrement parallèle que nous allons détailler mais également sur les précédentes innovations architecturales proposées par l'ENST Bretagne à savoir le traitement pipeline au sein du turbo décodeur [14] et le traitement simultané de plusieurs symboles d'une ligne ou d'une colonne au sein du décodeur élémentaire [10].

II. LE TURBO DÉCODAGE DES CODES PRODUITS

Un code produit est un code performant obtenu à partir de deux codes en blocs simples ayant de faibles distances de Hamming minimales δi. En effet, sa distance de Hamming minimale est alors égale au produit des distances de Hamming minimales des codes élémentaires utilisés, et son rendement au produit des rendements élémentaires.

Si on considère deux codes en blocs élémentaires $C_1(n_1, k_1)$ et $C_2(n_2, k_2)$, le code produit se présente sous forme de matrice C à n_1 lignes et n_2 colonnes où :

– les données binaires d'informations sont représentées par une sous-matrice M à k_1 lignes et k_2 colonnes,
– chacune des k_1 lignes de la matrice M est codée par le code C_2,
– chacune des n_2 colonnes de la matrice C est codée par le code C_1.

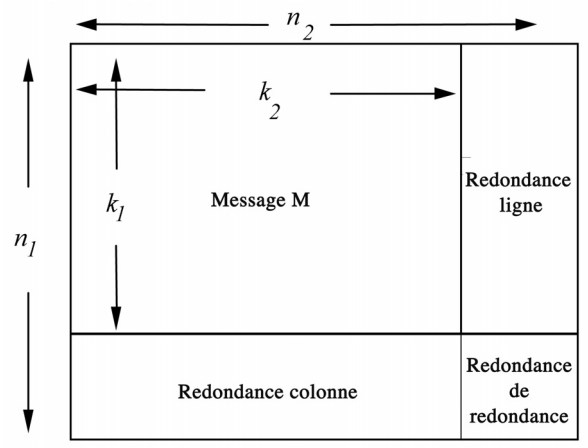


FIG 1. – Principe des codes produits.
Principle of product codes.

Si le code C_1 est linéaire, les $(n_1 - k_1)$ lignes construites par C_1 sont des mots de code de C_2 et peuvent donc être décodés comme les k_1 premières lignes. Un code produit se caractérise par n_1 mots de code de C_2 suivant les lignes, et par n_2 mots de code de C_1 suivant les colonnes. Les codes C_1 et C_2 peuvent être obtenus à partir de codes élémentaires convolutifs ou de codes en blocs linéaires.

Le turbo décodage d'un tel code consiste à faire un décodage à entrées et sorties pondérées (EPSP) de toutes les lignes puis de toutes les colonnes de la matrice C.

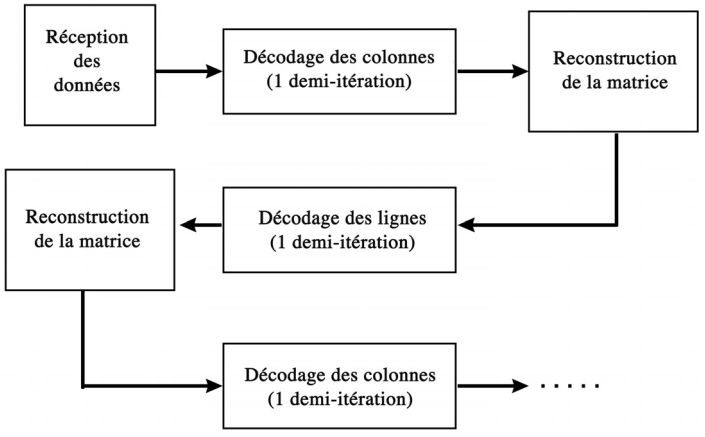


FIG 2. – Le processus itératif de décodage.
The decoding iterative process.

De manière générale, les informations échangées d'une demi-itération à une autre sont définies par le schéma suivant :

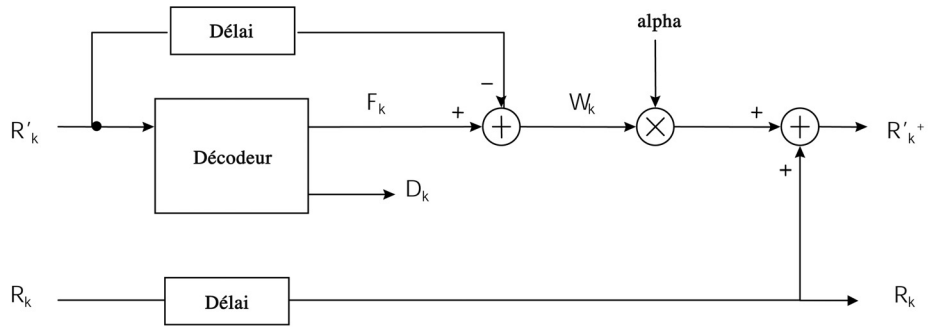


FIG 3. – Schéma-bloc d'une demi-itération du décodeur élémentaire EPSP.
Bloc diagram of a half-iteration elementary SISO decoder.

R_k correspond à l'information reçue du canal, R'_k à l'information qui vient de la demi-itération précédente et R'^{+}_k à l'information envoyée à la demi-itération suivante. La sortie de chaque demi-itération est donc égale à R_k plus l'information extrinsèque, W_k multipliée par *alpha*. Cette information extrinsèque correspond à l'apport du décodeur. Elle est obtenue par différence entre la sortie pondérée F_k du décodeur et l'entrée pondérée de ce même décodeur.

Nous considérons par la suite le décodeur à entrées et sorties pondérées comme un bloc ayant R_k et R'_k (échantillonnés sur q bits) comme entrées, délivrant R'^{+}_k et R_k (échantillonnés sur q bits) à la sortie avec une certaine latence L (retard nécessaire pour mettre en œuvre l'algorithme de décodage). Il prend le nom d'Unité de Traitement (UT). La structure interne du décodeur à entrées et sorties pondérées n'influence pas directement la structure du turbo décodeur. C'est pourquoi, nous ne présentons pas dans ce papier l'algorithme de Chase-Pyndiah et l'architecture associée que nous utilisons. Cependant, le lecteur peut le cas échéant consulter les références [2, 14, 15] qui traitent de ces aspects.

En considérant une autre découpe de ce schéma bloc, R'_k peut être remplacé par W_k qui devient entrée-sortie du bloc : R'_k est alors variable interne.

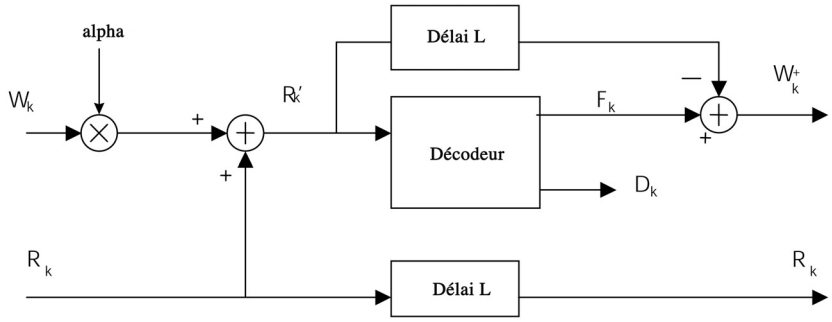


FIG 4. – Autre schéma-bloc pour le décodeur élémentaire EPSP.
Another block diagram for the elementary SISO decoder.

III. ÉTAT DE L'ART SUR LES ARCHITECTURES DE TURBO DÉCODEURS DE CODE PRODUIT

L'intégration d'un turbo décodeur peut être effectuée selon deux techniques :

- Technique séquentielle : structure où le décodeur élémentaire et le plan mémoire sont utilisés pour la totalité des demi-itérations ;
- Technique pipeline : structure où un décodeur élémentaire et un plan mémoire sont utilisés pour chaque demi-itération.

Dans la suite de ce document, nous faisons des comparaisons entre les différentes architectures. Une des caractéristiques retenues pour les comparer est la latence introduite par l'architecture. Classiquement, la latence se définie comme le retard temporel induit par l'architecture entre ses entrées et ses sorties. Dans notre contexte, cette définition ne permet pas une comparaison entre les techniques séquentielle et pipeline. En effet, la latence doit être inférieure à une valeur pour la technique séquentielle. Cette valeur est indépendante de l'architecture de décodage comme nous allons le voir par la suite. Il s'agit alors plutôt d'une contrainte temporelle pour l'architecture de décodage. Par contre, l'expression de la latence pour la technique pipeline est directement dépendante de l'architecture de décodage. En fait, la comparaison possible entre les deux techniques concerne le nombre symboles considérés au cours du traitement complet d'un symbole. C'est la raison pour laquelle, nous avons choisi de définir la latence comme étant le nombre de symboles traités par le turbo décodeur avant qu'un nouveau symbole présent en entrée de l'architecture soit à son tour complètement traité.

III.1. Architecture séquentielle pour le turbo décodage

Dans ce type d'architecture, le circuit réalise l'ensemble des demi-itérations à partir d'un décodeur élémentaire et d'un plan mémoire. Un bouclage sur la structure de base est nécessaire comme indiqué sur la figure 5.

L'intérêt principal de cette architecture est le faible encombrement du turbo décodeur. Le circuit complet comporte un plan mémoire composé de quatre mémoires de taille qn_1n_2 bits indépendamment du nombre d'itérations effectuées. Deux des quatre mémoires fonctionnent

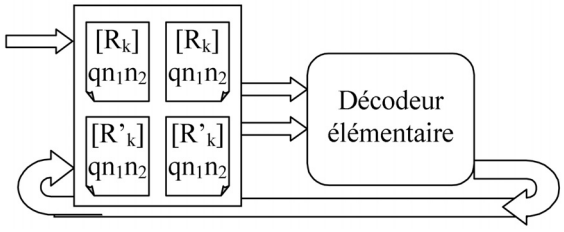


FIG 5. – Architecture séquentielle de turbo décodeur.
Turbo decoder sequential architecture.

en mode lecture, les deux autres fonctionnent en mode écriture. Il y a une inversion des modes de fonctionnement (lecture/écriture) des mémoires R'_k entre chaque demi-itération. Pour les mémoires R_k, l'inversion du mode de fonctionnement intervient à la réception d'une nouvelle matrice d'information. Les mémoires sont des RAM classiques accessibles par adressage suivant les lignes et les colonnes.

Le décodage à partir de cette structure consiste à faire un décodage à entrées et à sorties pondérées de toutes les lignes puis de toutes les colonnes de la matrice *C*. Ce processus est réitéré autant de fois que nécessaire (fonction du nombre de demi-itérations). La latence globale (telle que nous l'avons définie) de l'architecture séquentielle est au maximum de :

$$L_{atence} = 2*n_1 n_2 \quad (1)$$

Les $n_1 n_2$ premiers symboles sont considérés lors du remplissage d'une matrice de données et les $n_1 n_2$ symboles suivants sont considérés lors du traitement itératif de cette matrice. Dans ce cas, la latence est une contrainte qui fixe en particulier le nombre maximum d'itérations possibles dans la configuration retenue pour l'architecture.

L'inconvénient majeur de l'architecture séquentielle pour le turbo décodage de code produit est le débit de traitement des données. En effet, le débit doit tenir compte de l'utilisation d'une structure de base (un décodeur élémentaire et un plan mémoire) pour toutes les demi-itérations. Le débit maximal de traitement des données pour le turbo décodeur est donc au plus égal au débit de traitement d'un décodeur élémentaire divisé par le nombre de demi-itérations. C'est pourquoi l'architecture séquentielle permet un débit de traitement des données limité. L'utilisation de cette architecture séquentielle dans un contexte haut débit est par conséquent impossible en l'état.

III.2. Architecture pipeline pour le turbo décodage

La technique pipeline pour l'intégration d'un turbo décodeur repose sur une architecture modulaire. Dans ce type d'architecture, un décodeur élémentaire et un plan mémoire sont associés. Cette structure de base est alors mise en cascade comme indiqué sur la figure 6.

L'architecture finale est donc constituée d'autant de structures de base (association décodeur élémentaire et plan mémoire) que de demi-itérations. Chaque plan mémoire est composé de quatre mémoires de taille $qn_1 n_2$ bits. Le fonctionnement des différentes mémoires est similaire au fonctionnement d'une architecture séquentielle (cf. III.1.). En fait, l'inconvénient majeur de l'architecture pipeline pour le turbo décodage de code produit est l'encombrement du turbo décodeur. En effet, la mise en cascade des structures de base peut rendre cette solution architecturale très coûteuse lorsque le nombre d'itérations augmente, suivant la longueur du code produit retenu et/ou suivant le pouvoir de correction.

Le décodage à partir d'une structure pipeline consiste à faire un décodage à entrées et à sorties pondérées de toutes les lignes ou de toutes les colonnes d'une matrice pour chacune des demi-itérations. Par exemple pour *it* itérations, l'architecture du turbo décodeur contient $2it$ décodeurs élémentaires et $8it$ mémoires de taille $qn_1 n_2$ bits. Dans ce cas, l'encombrement du circuit provient en grande partie des blocs mémoires. En effet, il est indispensable de

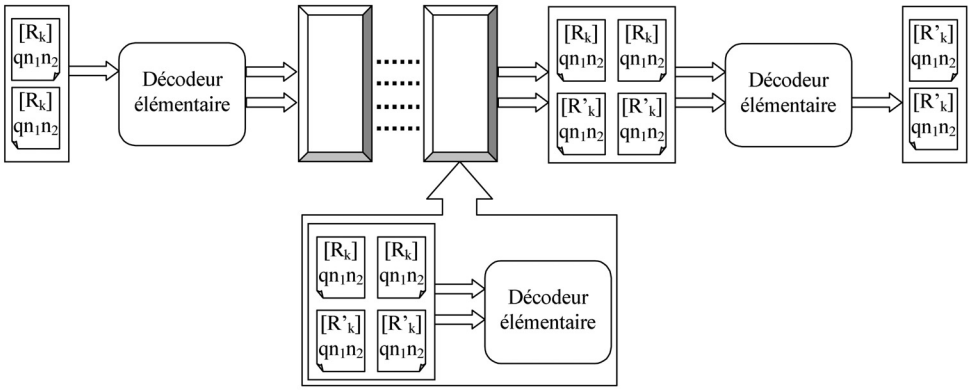


FIG 6. – Architecture pipeline de turbo décodeur.
Turbo decoder pipeline architecture.

mémoriser les matrices R_k (information reçue du canal) et R'_k (information qui vient de la demi-itération antérieure). Pour ce type d'architecture, la latence est là encore définie comme le nombre de symboles traités par le turbo décodeur avant qu'un nouveau symbole présent en entrée du circuit soit à son tour complètement traité. Dès lors, la latence introduite par l'architecture pipeline est $n_1n_2 + L_i$ symboles par demi-itération. Dans cette expression, les n_1n_2 premiers symboles sont considérés lors du remplissage d'une matrice de données et les L_i symboles suivants sont considérés lors du décodage proprement dit d'un symbole d'une ligne ($L_1 = xn_2$) ou d'un symbole d'une colonne ($L_2 = xn_1$) de cette matrice. La latence de décodage L_i d'un symbole au sein d'un décodeur élémentaire dépend directement de sa structure interne. En fait, il s'agit d'un multiple x du nombre n_2 de symboles d'une ligne ou du nombre n_1 de symboles d'une colonne. La latence globale introduite par l'architecture pipeline est donc de :

$$L_{atence} = \{(n_1n_2*2it) + it*(L_1 + L_2)\} \text{ pour it itérations} \quad (2)$$

L'intérêt de l'architecture pipeline est le débit de traitement des données. En effet, le débit obtenu est le débit de traitement d'un décodeur élémentaire. Cependant, elle entraîne un accroissement de l'encombrement du circuit de turbo décodage.

III.3. Architecture haut débit pour le turbo décodage

Les limitations en terme de débit de traitement des données se situent dans les décodeurs élémentaires pour les architectures de turbo décodage de code produit. Une solution pour augmenter le débit est de permettre au décodeur de traiter plusieurs symboles d'une ligne ou d'une colonne simultanément. Une solution architecturale reposant sur ce constat a été proposée dans le brevet de P. Adde et R. Pyndiah [16]. Nous décrivons dans la suite de ce paragraphe cette solution architecturale.

L'idée principale est d'augmenter le débit de traitement des données tout en conservant une même vitesse de fonctionnement pour le plan mémoire. La solution consiste alors à mémoriser plusieurs données à la même adresse. Il faut, cependant, pouvoir utiliser ces données aussi bien suivant les lignes que les colonnes. C'est pourquoi une organisation particulière de l'accès à la mémoire est nécessaire. Considérons, par exemple, deux lignes adjacentes et deux colonnes adjacentes dans la matrice initiale.

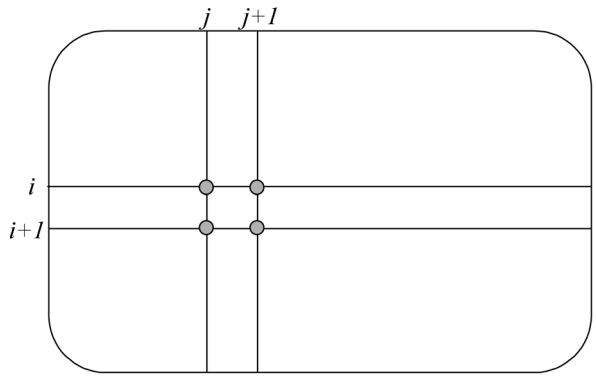


FIG 7. – Symboles adjacents dans la matrice initiale.
Adjacent symbols in the initial matrix.

Les quatre symboles (i, j), $(i, j+1)$, $(i+1, j)$ et $(i+1, j+1)$ constituent alors une donnée de la nouvelle matrice qui a quatre fois moins d'adresses (I, J) mais des données quatre fois plus grandes. Pour le décodage ligne, les échantillons (i, j), $(i, j+1)$ sont affectés à un décodeur élémentaire DEC1, $(i+1, j)$ et $(i+1, j+1)$ à un décodeur élémentaire DEC2. Pour le décodage colonne, il faut prendre (i, j), $(i+1, j)$ pour DEC1 et $(i, j+1)$, $(i+1, j+1)$ pour DEC2. Si les décodeurs élémentaires savent traiter en entrée (lecture de la RAM) et en sortie (écriture de la RAM) ces couples de symboles simultanément, alors le débit de traitement des données de la matrice sera quatre fois plus rapide que pour le décodeur classique. Le coût matériel de cette solution architecturale implique l'utilisation de deux décodeurs élémentaires traitant simultanément un couple de symboles d'une donnée de la matrice dans l'exemple considéré.

En généralisant, si une donnée de la nouvelle matrice contient m symboles d'une ligne et l symboles d'une colonne, le temps de traitement de la matrice est $m.l$ fois plus rapide avec seulement m décodeurs élémentaires pour le traitement des lignes lors d'une demi-itération et l décodeurs élémentaires pour le traitement des colonnes lors d'une demi-itération. Si les codes C_1 et C_2 sont identiques, les « décodeurs lignes » et les « décodeurs colonnes » le sont aussi : alors, $m = l$ et m décodeurs élémentaires sont nécessaires. Cette organisation des matrices de données ne requiert pas d'architectures de mémoires particulières, ni une horloge plus rapide. En fait, seul l'adressage de la mémoire est différent. La complexité matérielle de la solution triviale, consistant à décoder m^2 bits en parallèle, est de m^2 décodeurs élémentaires. Dans la solution architecturale proposée, la complexité matérielle du nouveau décodeur élémentaire est d'environ $m/2$ fois celle du décodeur élémentaire précédent [10]. Ainsi, une complexité de $m^2/2$ décodeurs élémentaires permet l'obtention d'une vitesse m^2 fois plus élevée. Enfin pour une taille identique, la mémoire comporte m^2 fois moins de données que la matrice initiale C. A technologie équivalente, son temps d'accès sera donc moindre.

IV. ARCHITECTURES PARALLÈLES DE TURBO DÉCODEUR

Dans cette partie, nous présentons un nouveau type d'architecture pour les turbo décodeurs de code produit. Ce type d'architecture élimine les plans mémoires associés aux données du code produit entre les différentes demi-itérations. Il permet aussi de traiter des données à des débits bien supérieurs à ceux des architectures précédentes (cf partie III). Enfin, la latence d'exécution de ce type d'architecture est fortement réduite.

IV.1. Principe du décodage parallèle d'un code produit

Un code produit est un code performant obtenu à partir de deux codes en blocs simples ayant de faibles distances de Hamming minimales δi. En effet, sa distance de Hamming minimale est alors égale au produit des distances de Hamming minimales des codes élémentaires utilisés, et son rendement au produit des rendements élémentaires.

Si on considère deux codes en blocs élémentaires $C_1(n_1, k_1)$ et $C_2(n_2, k_2)$, le code produit se présente sous forme de matrice C à n_1 lignes et n_2 colonnes.

Le décodage d'un tel code consiste à faire un décodage à entrées et à sorties pondérées de toutes les lignes puis de toutes les colonnes de la matrice C. Le décodage des n_2 mots de code suivant les colonnes est réalisé après un décodage de l'ensemble des n_1 mots de code suivant les lignes dans les solutions architecturales existantes. Cette approche nécessite l'utilisation d'un plan mémoire entre le décodage ligne et le décodage colonne. La latence de l'architecture séquentielle est au maximum de $2*n_1n_2$ (cf paragraphe III.1) et celle d'une architecture pipeline est de $\{(n_1n_2*2t) + it*(L_1+L_2)\}$ pour it itérations (cf paragraphe III.2).

Or, les n_1 lignes et les n_2 colonnes correspondent respectivement à n_1 et n_2 mots de code indépendants. C'est pourquoi le décodage des n_1 mots de code peut être effectué en parallèle si des ressources matérielles (décodeurs élémentaires) sont disponibles. De même, il est possible d'effectuer le décodage des n_2 mots de code en parallèle. Enfin, il est à noter que le traitement des symboles constituant un mot de code n'a pas d'ordre particulier. Il est juste nécessaire de traiter l'ensemble des symboles en repérant le symbole de parité lors du décodage.

En tenant compte de l'ensemble de ces propriétés, nous proposons un traitement parallèle de la matrice C. Pour chaque traitement de la matrice suivant les lignes ou les colonnes, tous les mots de code sont décodés en parallèle. Pour cela, nous supposons que nous disposons du nombre adéquat de décodeurs élémentaires.

De plus, si nous disposons des ressources matérielles pour décoder l'ensemble des lignes et des colonnes d'une matrice C, alors il devient possible de commencer le décodage des colonnes dès que des symboles composant les mots de code des lignes ont été traités. L'objectif est alors d'éliminer les plans mémoires. Or un traitement particulier de la matrice permet cette élimination. Nous allons dans la suite de ce paragraphe expliciter ce traitement en considérant tout d'abord une matrice carrée C à n lignes et n colonnes. Une généralisation de l'approche sera ensuite fournie pour une matrice C à n_1 lignes et n_2 colonnes.

IV.1.1. Décodage parallèle d'une matrice carrée *C* à *n* lignes et *n* colonnes

Dans ce paragraphe, nous présentons dans un premier temps l'exemple de la matrice carrée *8*8*. Dans un second temps, une généralisation est proposée pour le décodage des matrices carrées.

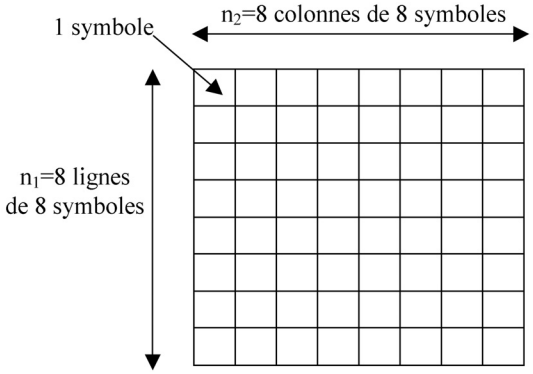
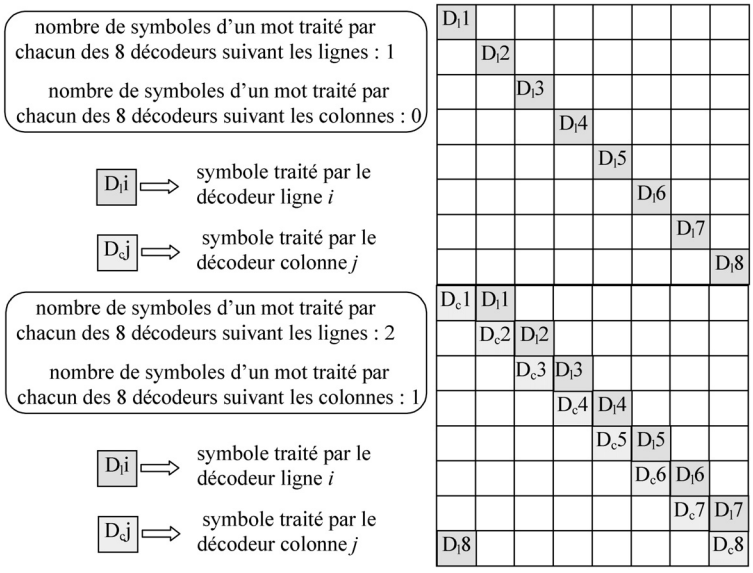
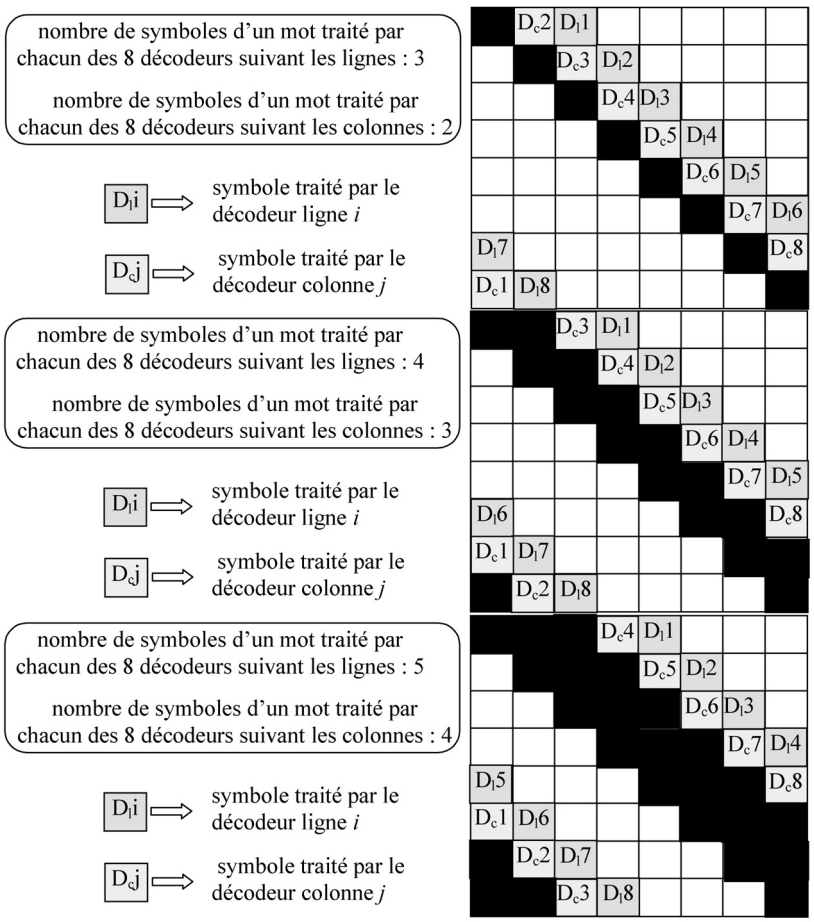
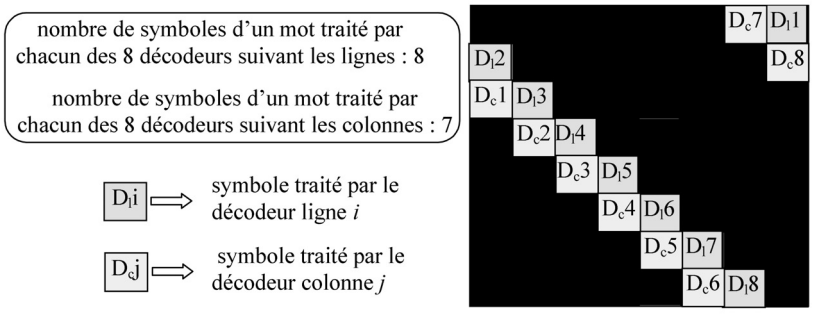


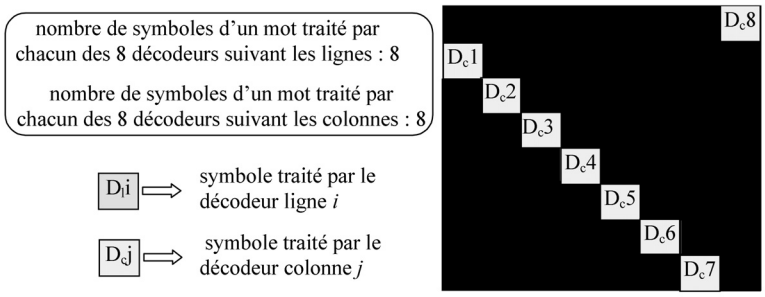
FIG 8. – Les 64 symboles de la matrice *8*8*.
*64 symbols of the matrix 8*8.*

Nous utilisons 16 décodeurs (8 pour les lignes et 8 pour les colonnes). Les 8 décodeurs lignes ($D_l i$ avec $i \in [1,8]$) traitent 8 mots de code de 8 symboles. Les 8 décodeurs colonnes ($D_c j$ avec j [1,8]) traitent également 8 mots de code de 8 symboles. La matrice est traitée en commençant par les symboles se trouvant sur une diagonale. Pour un traitement suivant les lignes, l'indice indiquant le passage d'un symbole à l'autre dans un mot de code est incrémenté de 1 modulo 8 (nombre de décodeurs lignes). Par contre, pour un traitement suivant les colonnes, l'indice indiquant le passage d'un symbole à l'autre dans un mot de code est décrémenté de 1 modulo 8 (nombre de décodeurs colonnes).









Nous définissons la latence d'un décodeur élémentaire comme le nombre de symboles traités par le décodeur avant qu'un nouveau symbole présent en entrée du circuit soit à son tour complètement traité. Cette latence dépend de la structure du décodeur élémentaire et en particulier du nombre x de mots de code traités en pipeline. Dans le cas de la matrice *8*8*, la latence des décodages lignes ou colonnes est de $8x$ symboles. Enfin, la latence entre le décodage des lignes et le décodage des colonnes est nulle (absence de plan mémoire).

Généralisation :

Si nous considérons deux codes en blocs élémentaires identiques C(n, k, δ), le code produit se présente sous forme d'une matrice C à n lignes et n colonnes.

Si nous disposons de 2n décodeurs, il est possible de décoder n lignes et n colonnes en parallèle.

Pour ce faire, si i est l'indice compris entre 1 et n des décodeurs lignes, alors un décodeur i commence le traitement d'un mot de code par le $i^{ème}$ symbole de ce mot de code. De plus, chaque décodeur ligne i traite les symboles d'un mot de code en incrémentant l'indice (i modulo n) associé aux symboles. La latence de décodage de la matrice C est alors de $L = xn$ symboles.

De même, si j est l'indice compris entre 1 et n des décodeurs colonnes, alors un décodeur j commence le traitement d'un mot de code par le $j^{ème}$ symbole de ce mot de code. Par contre, chaque décodeur colonne j traite les symboles d'un mot de code en décrémentant l'indice (j modulo n) associé aux symboles. La latence de décodage de la matrice C est alors de $L=xn$ symboles.

Enfin, si nous considérons une itération complète, la latence de décodage de la matrice C suivant les lignes puis les colonnes est de 2L symboles.

IV.1.2. Décodage parallèle d'une matrice *C* à n_1 lignes et n_2 colonnes

L'un des objectifs du décodage parallèle d'une matrice *C* est d'éliminer les plans mémoires. Cela implique d'avoir autant de décodeurs pour les lignes que pour les colonnes. Si les nombres de lignes et de colonnes d'une matrice *C* sont différents, alors il suffit de prendre $2n_{min}$ décodeurs élémentaires tel que :

(3) $$n_{min} = min(n_1, n_2)$$

Parmi eux, n_{min} sont utilisés pour le décodage des lignes et n_{min} sont utilisés pour le décodage des colonnes. Par conséquent, l'architecture est capable de décoder en parallèle n_{min} lignes et n_{min} colonnes de la matrice *C*.

Dans ce paragraphe, nous présentons dans un premier temps l'exemple de la matrice *8*16*. Dans un second temps, une généralisation est proposée pour le décodage des matrices à n_1 lignes et n_2 colonnes.

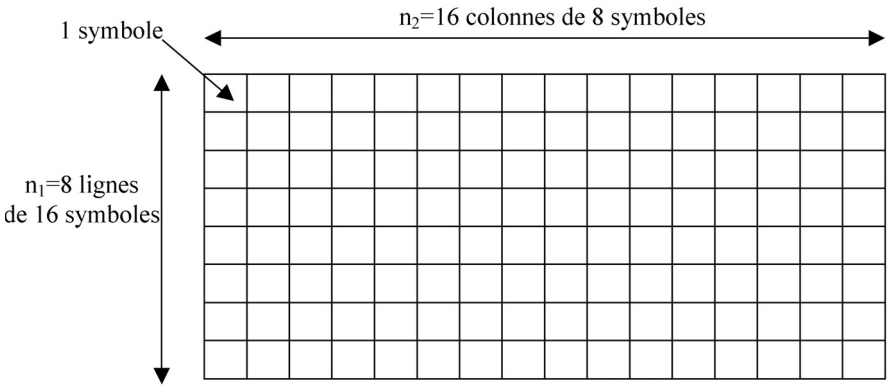
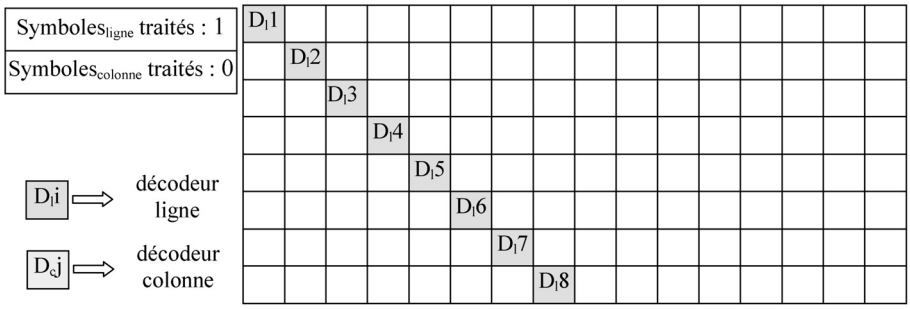
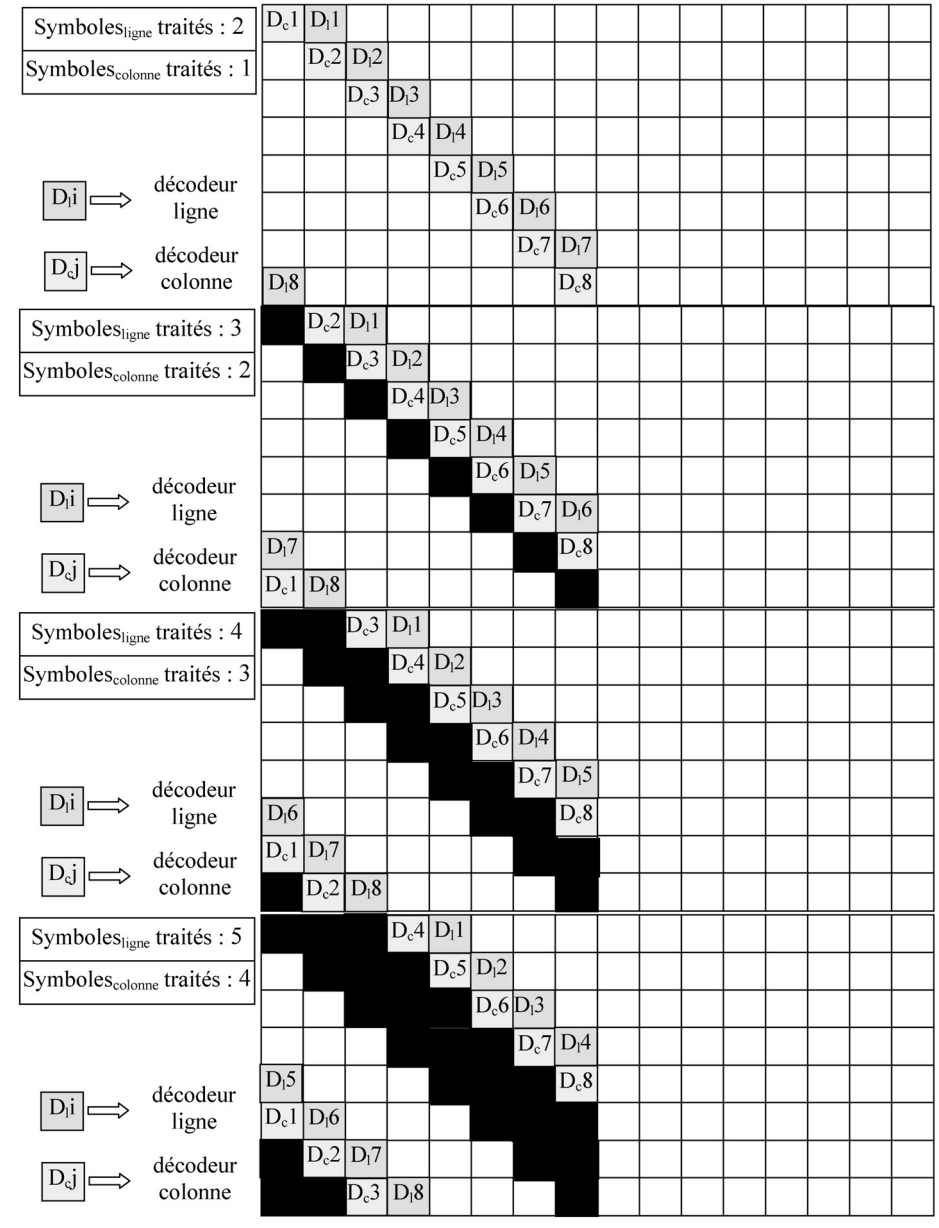
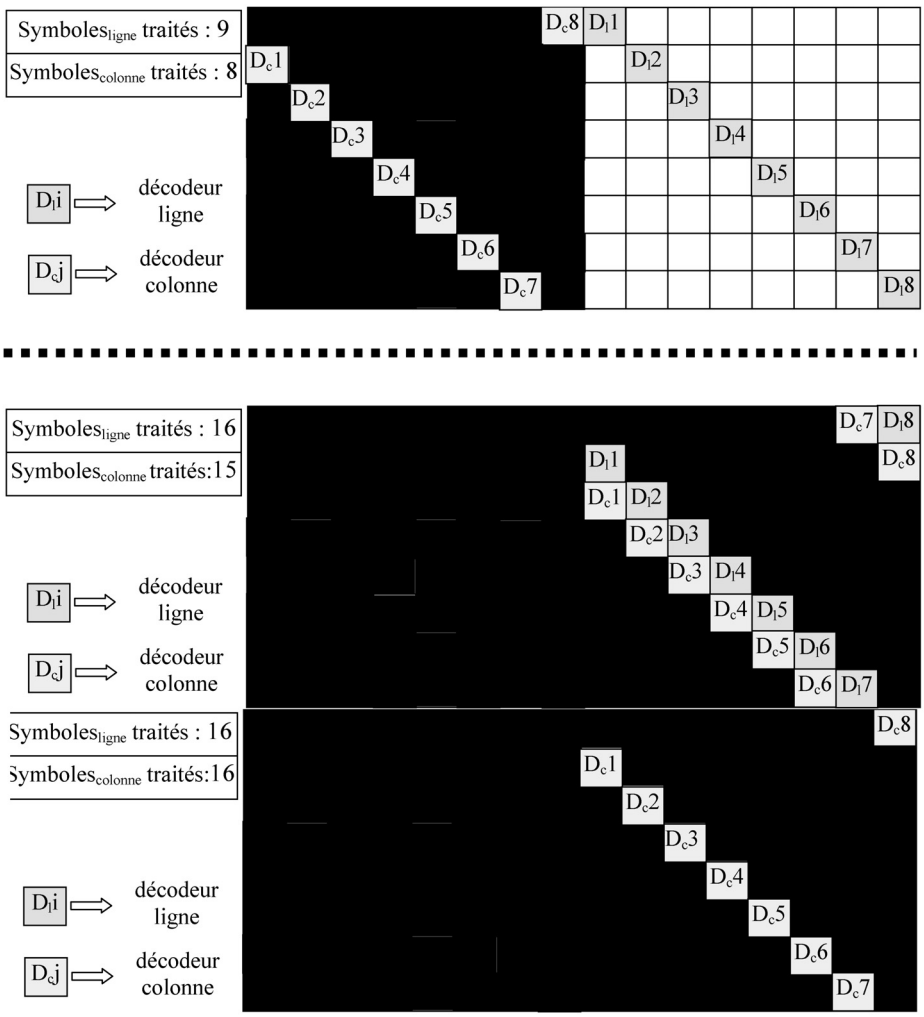


FIG 9. – Les 128 symboles de la matrice *8*16*.
*128 symbols of the matrix 8*16.*

Nous utilisons d'après (1), 16 décodeurs (8 pour les lignes et 8 pour les colonnes). Les 8 décodeurs lignes ($D_l i$ avec $i \in [1,8]$) traitent 8 mots de code de 16 symboles. Les 8 décodeurs colonnes ($D_c j$ avec $j \in [1,8]$) traitent deux fois 8 mots de code de 8 symboles. La matrice est traitée en commençant par les symboles se trouvant sur une diagonale. Pour un traitement suivant les lignes, l'indice indiquant le passage d'un symbole à l'autre dans un mot de code est incrémenté de 1 modulo 8 (nombre de décodeurs lignes). Par contre, pour un traitement suivant les colonnes, l'indice indiquant le passage d'un symbole à l'autre dans un mot de code est décrémenté de 1 modulo 8 (nombre de décodeurs colonnes).







Nous définissons la latence d'un décodeur élémentaire comme le nombre de symboles traités par le décodeur avant qu'un nouveau symbole présent en entrée du circuit soit à son tour complètement traité. Cette latence dépend de la structure du décodeur élémentaire et en particulier du nombre x de mots de code traités en pipeline. Pour la matrice *8*16*, la latence des décodages lignes et colonnes sont respectivement de *16x* et de *8x* symboles. Enfin, la latence entre le décodage des lignes et le décodage des colonnes est nulle (absence de plan mémoire).

Généralisation :

Si nous considérons deux codes en blocs élémentaires $C_1(n_1, k_1, \delta_1)$ *et* $C_2(n_2, k_2, \delta_2)$, *le code produit se présente sous forme d'une matrice C à* n_1 *lignes et* n_2 *colonnes.*

Si nous disposons de $2n_{min}$ décodeurs tel que $n_{min}= min(n_1, n_2)$, il est possible de décoder n_{min} lignes et n_{min} colonnes en parallèle.

Pour ce faire, si i est l'indice compris entre 1 et n_{min} des décodeurs lignes, alors un décodeur i commence le traitement d'un mot de code par le $i^{ème}$ symbole de ce mot de code. De plus, chaque décodeur ligne i traite les symboles d'un mot de code en incrémentant l'indice (i modulo n_{min}) associé aux symboles. La latence de décodage de la matrice C est alors de $L_1=xn_2$ symboles.

De même, si j est l'indice compris entre 1 et n_{min} des décodeurs colonnes, alors un décodeur j commence le traitement d'un mot de code par le $j^{ème}$ symbole de ce mot de code. Par contre, chaque décodeur colonne j traite les symboles d'un mot de code en décrémentant l'indice (j modulo n_{min}) associé aux symboles. La latence de décodage de la matrice C est alors de $L_2=xn_1$ symboles.

Enfin, si nous considérons une itération complète, la latence de décodage de la matrice C suivant les lignes puis les colonnes est de $L_1 + L_2$ symboles.

IV.2. Application pour le turbo décodage des codes produits

Nous avons décrit dans le paragraphe précédent notre technique de décodage parallèle des codes produits. Cette technique peut être appliquée comme nous allons le voir à toutes les architectures de turbo décodeurs de code produit.

IV.2.1. Gestion des données de la matrice C entre les itérations

L'un des avantages de la technique de décodage parallèle des codes produits que nous proposons, est d'éliminer les plans mémoires associés aux matrices générées par un code produit entre les différentes demi-itérations. Une gestion de la communication des symboles des différents mots de code de la matrice entre le décodage des lignes et des colonnes est alors nécessaire. Une solution triviale consiste à placer un réseau d'interconnexion entre les décodeurs des lignes et des colonnes comme indiqué sur la figure 10.

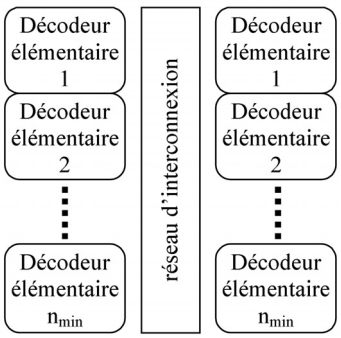


FIG 10. – Communication entre les décodeurs des lignes et des colonnes.
Communication between row and column decoders.

Ce réseau d'interconnexion peut être matérialisé par de simples liaisons point à point ou par un réseau de type *Crossbar.* Cependant, nous proposons d'insérer un réseau d'interconnexion dynamique multi-étages de type *Omega* [17]. Un réseau dynamique est un réseau dont la topologie de connexion varie au cours du temps. Ce type de réseau répond à notre problématique car les symboles traités par les décodeurs élémentaires sont ensuite répartis sur l'ensemble des décodeurs de la demi-itération suivante selon un profil de communication de type permutation circulaire. De plus, le réseau d'interconnexion de notre architecture doit traiter n_{min} communications simultanément. Cette caractéristique est un cas de figure typique d'un réseau dynamique. Le réseau dynamique que nous avons retenu est de type multi-étages. L'intérêt de cette structure par rapport à une solution à base de liaisons point à point ou de type *Crossbar* est la limitation des connexions et des commutateurs. En effet, dans les réseaux multi-étages le nombre de liaisons et le nombre de commutateurs évoluent de manière logarithmique :

(4) $$\textbf{\textit{nombre de liaisons : }}(n_{min} * Log_2 n_{min})$$

(5) $$\textbf{\textit{nombre de commutateurs : }}((n_{min} * Log_2 n_{min})/2)$$

Les commutateurs contiennent généralement deux entrées et deux sorties. Ils sont assemblés sous forme de tableau rectangulaire de dimension n_{min} lignes et $Log_2 n_{min}$ colonnes. Le nombre d'étages est donc de $Log_2 n_{min}$ dans ce type de réseau d'interconnexion.

Le réseau d'interconnexion dynamique multi-étages de type *Omega* est basé sur le principe de permutation circulaire. L'algorithme de liaison consiste à décaler circulairement le passage de l'information entre les sources et les destinations. Un exemple de communication

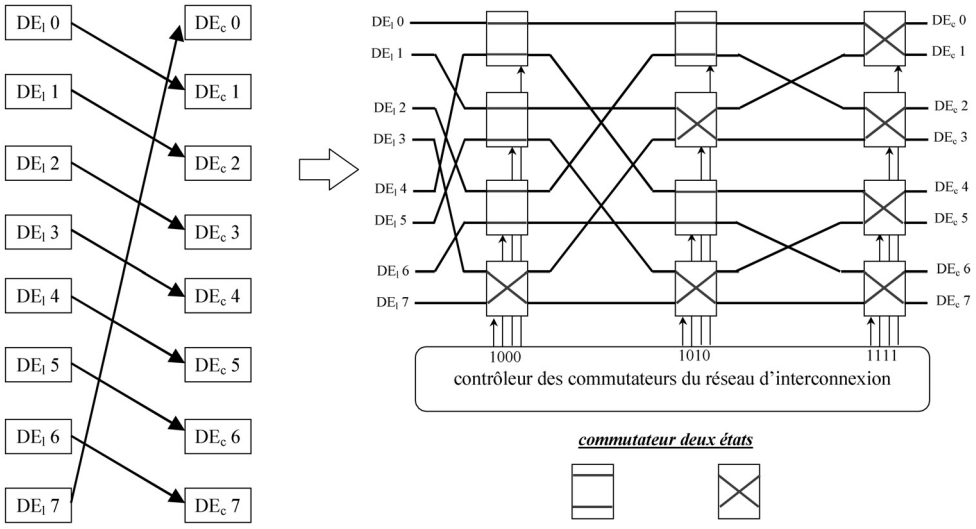


FIG 11. – Exemple de communication entre les décodeurs lignes et colonnes avec un réseau d'interconnexion dynamique multi-étages de type *Omega.*

Communication example between row and column decoders with a multi-stages dynamical connection network Omega.

entre les décodeurs élémentaires pour le traitement d'une matrice *8*8* suivant les lignes puis suivant les colonnes est illustré par la figure 11. Les commutateurs sont des circuits très simples et peu coûteux à réaliser à l'aide d'interrupteurs en technologie CMOS. Un commutateur deux positions correspond à quatre interrupteurs et un inverseur. La complexité en équivalent portes logiques est donc de 2,5 portes. Le réseau d'interconnexion de l'exemple de communication de la figure 11 a une complexité matérielle de $30q$ portes logiques (q : nombre de bits de quantification des symboles de la matrice). Le réseau nécessite également un contrôleur pour le positionnement des commutateurs suivant les connexions désirées.

IV.2.2. Architecture séquentielle parallèle

Dans ce type d'architecture, le circuit réalise l'ensemble des demi-itérations à partir d'une structure élémentaire et d'un plan mémoire. La structure élémentaire se compose de n_{min} décodeurs élémentaires. Un bouclage sur la structure de base est nécessaire comme indiqué sur la figure 12.

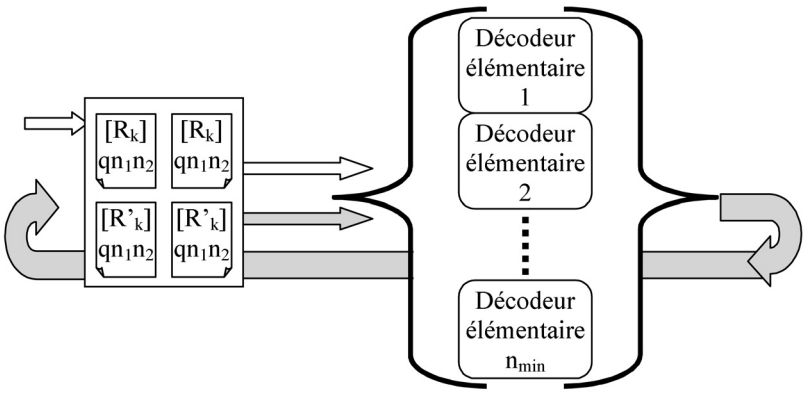


FIG 12. – Architecture parallèle séquentielle au niveau demi-itération d'un turbo décodeur.
Sequential parallel architecture for a half-iteration of turbo decoding.

L'inconvénient majeur de l'architecture séquentielle classique pour le turbo décodage de code produit est le débit de traitement des données. L'architecture séquentielle parallèle permet de résoudre en partie cette limitation. De plus, il est possible d'augmenter les débits de traitement des données en intégrant un circuit qui réalise une itération sur une structure de base comme indiquée sur la figure 13. La latence pour le traitement d'une itération de la matrice C est de $(L_1 + L_2)$ symboles. La latence globale introduite alors par l'architecture séquentielle parallèle doit donc être au maximum de $2*n_1n_2$ (n_1n_2 premiers symboles sont considérés lors du remplissage d'une matrice de données et les n_1n_2 symboles suivants sont considérés lors du traitement itératif de cette matrice). Dans ce cas, la latence globale est une contrainte qui fixe en particulier le nombre maximum d'itérations possibles. C'est pourquoi, il faut s'assurer que le nombre d'itérations *it* permet le respect du maximum de latence autorisé, c'est-à-dire :

(6) $$L_{atence} = (L_1+L_2)*it < n_1 n_2$$

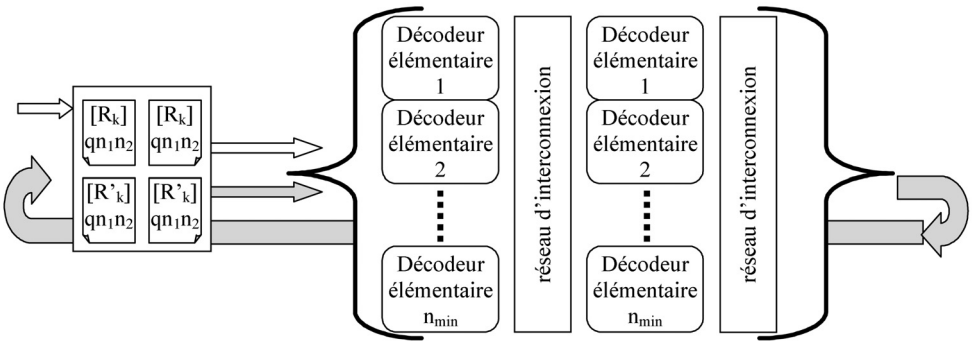


FIG 13. – Architecture parallèle séquentielle au niveau itération d'un turbo décodeur.
Sequential parallel architecture for an iteration of turbo decoding.

Le débit de l'architecture parallèle séquentielle est intéressant. En effet, il est n_{min} fois plus élevé que celui de l'architecture séquentielle classique. De plus, le débit peut encore être multiplié par deux si l'architecture est séquentielle au niveau itération (cf figure 13). Néanmoins, l'utilisation de l'architecture séquentielle parallèle dans un contexte très haut débit (> 10 Gbit/s) est impossible en l'état.

IV.2.3. Architecture pipeline parallèle

La technique pipeline pour l'intégration d'un turbo décodeur repose sur une architecture modulaire. Dans une architecture pipeline parallèle, $2n_{min}$ décodeurs élémentaires et deux réseaux d'interconnexion sont associés. La structure permet le décodage de la matrice C suivant les lignes puis les colonnes. Cette structure de base est alors mise en cascade comme indiqué sur la figure 14.

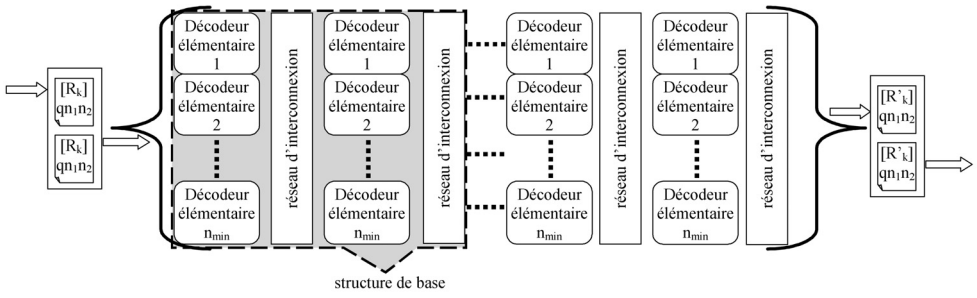


FIG 14. – Architecture pipeline parallèle d'un turbo décodeur.
Pipeline parallel architecture for turbo decoding.

L'architecture finale est donc constituée d'autant de structures de base (association de $2n_{min}$ décodeurs élémentaires et deux réseaux d'interconnexion) que d'itérations. Cette solution architecturale ne nécessite alors au maximum que quatre mémoires de taille qn_1n_2 bits. De plus, si le module de réception associé au turbo décodeur a des bus de transferts de données appropriés en entrée et/ou en sortie de l'architecture, alors certaines mémoires peuvent être supprimées. Au final, cette architecture pipeline parallèle a un encombrement qui dépend de la taille de la matrice C. En effet, $2n_{min}$ décodeurs élémentaires sont nécessaires pour chaque itération. C'est pourquoi l'encombrement du circuit final peut s'avérer coûteux si la taille de la matrice à décoder est importante.

Le décodage à partir d'une structure pipeline parallèle consiste à faire un décodage à entrées et à sorties pondérées de toutes les lignes ou de toutes les colonnes d'une matrice pour chacune des demi-itérations selon le principe décrit dans le paragraphe IV.1. Dans ce cas, l'encombrement du circuit concerne essentiellement les décodeurs élémentaires. La complexité due aux plans mémoires est bien moindre que pour la solution architecturale pipeline classique. De plus, comme la mémorisation des matrices R_k (information reçue du canal), R'_k (information qui vient de la demi-itération antérieure) et R'^{+}_k (information envoyée à la demi-itération suivante) n'est plus nécessaire entre les demi-itérations, la latence globale est également bien moins importante. Nous définissons la latence globale comme le nombre de symboles traités par le turbo décodeur avant qu'un nouveau symbole présent en entrée du circuit soit à son tour complètement traité. La latence d'une itération complète est de (L_1+L_2) symboles (cf paragraphe IV.1). La latence de traitement de la matrice C pour *it* itérations de ce type d'architecture est donc de :

$$L_{atence} = (L_1+L_2)*it \text{ pour } it \text{ itérations} \tag{7}$$

L'avantage principal de l'architecture pipeline parallèle est le débit de traitement des données qui peut être atteint. En effet, le débit obtenu est le débit de traitement d'une structure de base. Ce débit est n_{min} supérieur à celui d'une architecture pipeline classique. De plus, il est possible d'augmenter encore ce débit en utilisant le principe du brevet décrit dans le paragraphe III.3.

V. PROPOSITION D'UNE ARCHITECTURE PARALLÈLE DE TURBO DÉCODEUR DE CODE PRODUIT POUR LE TRÈS HAUT DÉBIT

L'un des apports du brevet déposé par P. Adde et R. Pyndiah est d'augmenter le débit de traitement des données tout en conservant une fréquence constante pour le plan mémoire et le décodeur. Le décodeur élémentaire peut alors traiter plusieurs symboles d'un mot de code de la matrice simultanément. Dans la suite du document, nous utiliserons la notation *k-UT* pour indiquer que le décodeur élémentaire traite k données d'une même ligne (ou d'une même colonne) simultanément.

Dès lors, des architectures de turbo décodeur de code produit de faible latence pour le très haut débit sont possibles. Pour ce faire, l'architecture du turbo décodeur doit permettre un traitement parallèle des différents mots de code de la matrice selon le principe que nous avons proposé. De plus, les décodeurs élémentaires doivent être capables de traiter plusieurs

symboles d'un mot de code simultanément. Une architecture ayant ces propriétés est présentée dans la figure 15.

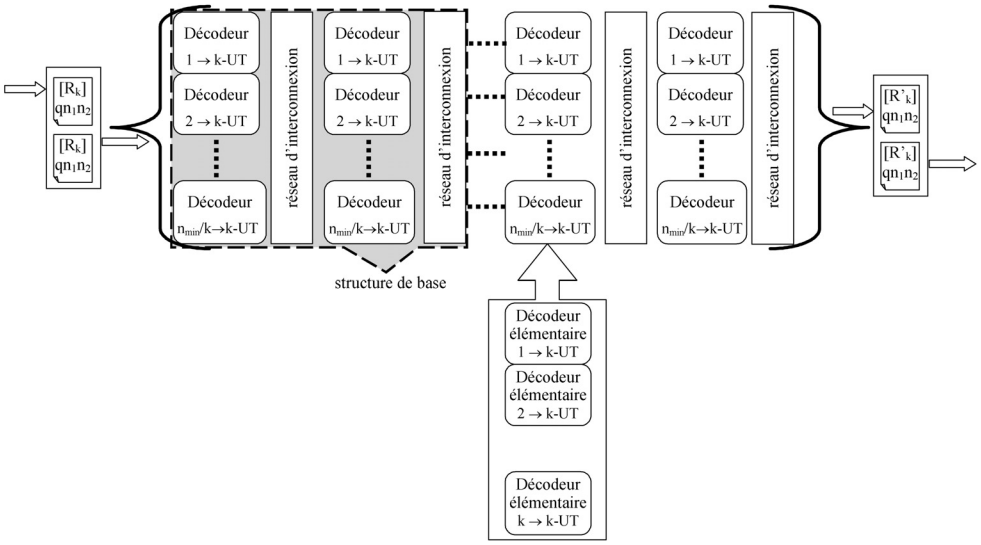


FIG 15. – Architecture d'un turbo décodeur pour le très haut débit.
Turbo decoder architecture for very-high-rate.

L'architecture proposée comprend autant de structures de base (association de $2n_{min}/k$ décodeurs et de deux réseaux d'interconnexion) que d'itérations. Chaque décodeur *k-UT* se décompose en *k* décodeurs élémentaires capables de traiter simultanément *k* symboles d'un même mot de code. Au final, la complexité en terme de nombre de décodeurs élémentaires est similaire à celle d'une structure pipeline parallèle pipeline classique, c'est-à-dire $2n_{min}$ décodeurs élémentaires par itération. Cependant, la complexité du décodeur élémentaire *k-UT* est environ *k*/2 fois supérieure à celle d'un décodeur élémentaire classique *1-UT*. Cette solution architecturale ne nécessite alors au maximum que quatre mémoires de taille qn_1n_2 bits. L'encombrement du circuit final est essentiellement dû à l'encombrement des décodeurs élémentaires. Cet encombrement peut s'avérer là encore coûteux suivant la taille de la matrice.

La latence d'une itération complète est de $(L_1/k+L_2/k)$ symboles. Elle est inférieure à celle d'une architecture pipeline parallèle classique car le décodeur élémentaire traite *k* symboles d'un mot de code simultanément. La latence de traitement de la matrice *C* pour *it* itérations de ce type d'architecture est donc de :

$$L_{atence} = (L_1/k + L_2/k)*it \text{ pour it itérations} \quad (8)$$

Cette latence est très faible si nous la comparons à celles obtenues dans les architectures classiques.

L'atout indéniable de l'architecture que nous proposons est le débit pouvant être atteint. Il est désormais possible d'intégrer des circuits de turbo décodeur de code produit ayant des débits supérieurs à 10 Gbit/s. Ainsi, le gain en débit est d'un facteur $(n_{min}/k)*k^2$ par rapport à une architecture pipeline classique. De plus, ce gain en débit demeure élevé (n_{min}/k) par rapport à la solution architecture pipeline proposée dans le brevet de P. Adde et R. Pyndiah. C'est pourquoi nous affirmons proposer une architecture de turbo décodeur de code produit pour les très hauts débit.

Pour illustrer l'apport de notre proposition d'architecture parallèle, nous allons comparer les performances pour un turbo décodeur étudié dans la thèse de J. Cuevas [18]. Il s'agit du turbo décodeur de code produit utilisant deux codes BCH étendus (32,26,4). Le tableau I donne les performances en terme de débit et de latence de traitement du décodage de la matrice *32*32* pour une demi-itération. La complexité en nombre de portes logiques est également fournie. Il est à noter que pour les décodeurs haut débit et très haut débit, les décodeurs élémentaires traitent quatre symboles simultanément (*4-UT*). La technologie d'intégration est la CMOS 0,09 µm de chez STMicroelectronics.

TABLEAU I. – Principales caractéristiques des différentes architectures pour une demi-itération.

Features of different architectures for a half-interation.

décodage *32*32*	k-UT	Débit (Mbit/s)	Latence (nbre de symboles)	Complexité décodeur (nbre de portes)	Capacité mémoire (nbre de bits)	Complexité réseau connexion (nbre de portes)
architecture référence	1-UT	100	64	5 500	20480	0
architecture haut débit [10, 18]	4-UT	1 600	16	44 000	20480	0
architecture parallèle très haut débit (IV.3)	4-UT	12 800	16	352 000	0	1 000

Nous constatons sur cet exemple que le débit de notre solution architecturale (12,8 Gbit/s) est 128 fois plus important que le débit de référence. De plus, ce débit est 8 fois plus élevé que celui obtenu par l'architecture de la thèse de J. Cuevas. La latence quant à elle, est divisée par quatre. Le coût matériel induit par notre proposition au niveau des décodeurs élémentaires est 64 fois plus élevé que l'architecture de référence et 8 fois plus élevé que l'architecture de la thèse de J. Cuevas. Il faut néanmoins se souvenir que notre proposition élimine les plans mémoires entre chaque demi-itération. Cette simplification est à mettre en regard à la complexité matérielle introduite au niveau des décodeurs élémentaires. De plus, si nous considérons le ratio nombre de portes du décodeur par Mbit/s, alors nous constatons qu'il est d'environ 55 pour l'architecture de référence et d'environ 27,5 pour les deux autres architectures. Le coût matériel du Mbit/s est donc divisé par deux.

VI. BILAN RÉCAPITULATIF DES DIFFÉRENTES ARCHITECTURES POSSIBLES

Dans ce dernier paragraphe, nous dressons un bilan récapitulatif des architectures de turbo décodeur de code produit. Nous comparons les performances en terme de latence, de débit et de complexité des différentes solutions. Nous considérons deux familles d'architecture : les architectures classiques et les architectures parallèles. La famille des architectures classiques correspond à l'état de l'art sur les architectures de turbo décodeur de code produit. Elle se compose des architectures séquentielles et pipelines. Pour les structures pipelines, nous différencions les architectures composées de décodeurs élémentaires *1-UT* ou *k-UT*. La famille des architectures parallèles correspond quant à elle aux architectures obtenues en appliquant la technique novatrice de décodage parallèle que nous avons décrit dans ce document. La définition des performances des différentes architectures dépend d'un certain nombre de grandeurs caractéristiques telles que :

– n_1 : nombre de lignes de la matrice C,
– n_2 : nombre de colonnes de la matrice C,
– n_{min} : minimum entre les nombres n_1 et n_2,
– it : nombre d'itérations dans le processus turbo,
– q : nombre de bits de quantification des symboles de la matrice C,
– k : nombre de symboles d'une ligne ou d'une colonne traitée simultanément,
– x : nombre de mots de code traités en parallèle dans l'architecture pipeline,
– $L_1 = xn_2$: latence de décodage d'une ligne de la matrice C,
– $L_2 = xn_1$: latence de décodage d'une colonne de la matrice C.

Le tableau II est un tableau récapitulatif qui met en évidence les performances de la famille des architectures parallèles en terme de latence et de débit. De plus, cette famille élimine les plans mémoires associés aux matrices de données d'un code produit entre les demi-itérations pour les structures pipelines. Cette élimination des plans mémoires implique l'utilisation d'un réseau d'interconnexion entre les demi-itérations. Cependant, nous avons proposé une solution triviale et peu coûteuse en terme de complexité pour le réseau d'interconnexion. Naturellement, ces résultats ont un coût qui est la complexité au niveau des décodeurs élémentaires. Néanmoins, le coût matériel du Mbit/s pour les architectures parallèles est identique à celui des architectures classiques équivalentes.

TABLEAU II. – Récapitulatif des différentes architectures possibles.

Summary of all architectures.

	Architecture classique			Architecture parallèle		
	Séquentielle	Pipeline		Séquentielle	Pipeline	
	1-UT	1-UT	k-UT	1-UT	1-UT	k-UT
latence (nbre symboles)	$< 2*n_1n_2$	$2it*n_1n_2 + it*(L_1+L_2)$	$2it*n_1n_2 + it*((L_1+L_2)/k)$	$n_1n_2+(it*(L_1+L_2)) < 2*n_1n_2$	$it*(L_1+L_2)$	$it*((L_1+L_2)/k)$
débit (Mbits/s)	D_{ref}	$D_{ref}*2it$	$D_{ref}*2it*k^2$	$D_{ref}*n_{min}$	$D_{ref}*2it*n_{min}$	$D_{ref}*2ti*(n_{min}/k)*k^2$
nbre de décodeurs él.	1	$2it$	$2it*k*(k/2)$	n_{min}	$n_{min}*2it$	$(n_{min}/k)*2it*k*(k/2)$
capacité mémoire (en bit)	$4qn_1n_2$	$4qn_1n_2*2it$	$4qn_1n_2*2it$	$4qn_1n_2$	0	0
nbre de réseaux d'interconnexion	0	0	0	0	$2it$-1	$2it$-1

Remarque : comme un décodeur élémentaire k-UT a une complexité environ *k/2* fois plus importante qu'un décodeur élémentaire 1-UT d'après [18], nous ajoutons un facteur *k/2* dans le nombre de décodeurs élémentaires quand les décodeurs élémentaires k-UT sont utilisés.

VII. CONCLUSION

Nous avons décrit dans cet article un nouveau type d'architecture de décodage des codes produits à faible latence, pouvant fonctionner à très haut débit. Ces architectures reposent sur une technique de traitement parallèle suivant les lignes et les colonnes de la matrice du code produit. Le principe de traitement parallèle de la matrice du code produit constitue l'innovation majeure de notre proposition. Pour le turbo décodage, nous avons montré comment cette proposition peut s'appliquer aux solutions architecturales séquentielles et pipelines. Ainsi, le turbo décodage parallèle permet d'augmenter le débit de traitement des données tout en diminuant la latence globale du circuit de turbo décodage. De plus, cette approche élimine les plans mémoires entre les demi-itérations pour les architectures pipelines. Enfin, si nous appliquons notre technique de turbo décodage parallèle aux architectures décrites dans le brevet déposé par P. Adde et R. Pyndiah, les circuits de turbo décodeurs résultants peuvent atteindre des débits supérieurs à 10 Gbit/s. Un premier circuit de turbo décodage a été intégré sur une carte de prototypage contenant un FPGA Virtex II-Pro. Le turbo décodeur en question traite le code produit BCH$(16,11)^2$. Il a été validé dans un environnement fonctionnant à 2,4 Gbit/s pour le débit en entrée et avec une fréquence de fonctionnement de 37,5 MHz pour le turbo décodeur. Cette première expérimentation a permis de valider le principe de traitement parallèle de la matrice du code produit. De plus, une étude visant à proposer un circuit de turbo décodage sur une cible FPGA pour la couche physique des réseaux d'accès optique ayant un débit ligne de 10 Gbit/s est en cours.

Manuscrit reçu le 26 juin 2006
Accepté le 25 octobre 2006

BIBLIOGRAPHIE

[1] BERROU (C.), GLAVIEUX (A.), THITIMAJSHIMA (P.), Near Shannon limit error correcting coding and decoding: Turbo Codes, *ICC93 IEEE International Conference on Communication*, **2/3**, 1993.

[2] PYNDIAH (R.), GLAVIEUX (A.), PICART (A.), JACQ (S.), Near optimum decoding of product codes, *GLOBECOM94*, 1994.

[3] GALLAGER (R. G.), Low Density Parity Check Codes, *IRE Trans. Inform. Theory*, 1962.

[4] AZADET (K.), HARATSCH (E. F.), KIM (H.), SAIBI (F.), SAUNDERS (J. H.), SHAFFER (M.), SONG (L.), YU (M.-L.), Equalization and FEC techniques for optical transceivers, *Solid-State Circuits, IEEE Journal of*, **37**, issue 3, 2002.

[5] ITU-T G.709, Interface ofr the Optical Transport Networks, *Telecommunication Standardization Section, International Telecommunication Union*, 2003.

[6] AIT SAB (O.), LEMAIRE (O. V.), Block turbo code performances for long-haul DWDM optical transmission systems*, Optical Fiber Communication Conference*, vol. 3, 2000.

[7] MIZUOCHI (T.), Recent Progress in Forward Error Correction for Optical Communication Systems**,** *IEICE Transactions on Communications*, **E88-B**, nb. 5, 2005.

[8] SONG (L.), YU (M.-L.), SHAFFER (M. S.), 10- and 40-Gb/s forward error correction devices for optical communications, *Solid-State Circuits, IEEE Journal of ,* **37**, issue 11, 2002.

[9] LEE (H.), A high-speed low-complexity Reed-Solomon decoder for optical communications**,** *Circuits and Systems II : Express Briefs, IEEE Transactions on*, **52**, Issue 8, 2005.

[10] CUEVAS (J.), ADDE (P.), KEROUEDAN (S.), PYNDIAH (R.), New architecture for high data rate turbo decoding of product codes, *GLOBECOM 2002*, **2**, 2002.

[11] TAGAMI (H.), KOBAYASHI (T.), MIYATA (Y.), OUCHI (K.), SAWADA (K.), KUBO (K.), KUNO (K.), YOSHIDA (H.), SHIMIZU (K.), MIZUOCHI (H.), MOTOSHIMA (K.), A 3-bit soft-decision IC for powerful forward error correction in 10-Gb/s optical communication systems**,** *Solid-State Circuits, IEEE Journal of,* **40**, issue 8, 2005.

[12] PIRIOU (E.), JEGO (C.), ADDE (P.), LE BIDAN (R.), JÉZÉQUEL (M.), Efficient Architecture For Reed Solomon Block Turbo Code, *ISCAS 2006, IEEE International Symposium on Circuits and Systems*, 2006.

[13] JEGO (C.), ADDE (P.), Full-parallel architecture for turbo decoding of product codes, *Electronics letters,* vol. 42 nb.18, 2006.

[14] KEROUEDAN (S.), ADDE (P.), PYNDIAH (R.), How we implemented Block Turbo Codes, *annals of telecommunication,* **56**, N° 7-8, 2001.

[15] PYNDIAH (R.), ADDE (P.), ZHOU (R.), Block Turbo Codes : Ten years later, *IEE Seminar on Sparse Graph Codes*, 2004.

[16] ADDE (P.), PYNDIAH (R.), Module, dispositif et procédé de décodage à haut débit, d'un code concaténé, *brevet français n° 00/14521*, *brevet international PCT FR01/03509.*

[17] LAWRIE (D. H.), Access and alignment of data in an array processor, *IEEE Trans. Comput***., C-24**, n° 10, 1975.

[18] CUEVAS (J.)*,* Turbo Décodage de Code Produit Haut Débit, Thèse de Doctorat à l'Université de Bretagne Sud, 2004.