Implémentation sur FPGA d'un turbo codeur-décodeur en blocs à haut-débit avec une faible complexité.

T. TA[1], P. LERAY[1], A. LE GLAUNEC[1]

[1] SUPELEC - Campus de Rennes, équipe ETSN.

Avenue de la Boulaie, BP 81127, 35511 Cesson-Sévigné Cedex, France. Téléphone : 33 [0]2.99.84.45.00.

Email : thomas.ta@supelec.fr, pierre.leray@supelec.fr, Annick.leglaunec@supelec.fr.

Résumé – Ce papier présente une implémentation sur FPGA (Field Programmable Gate Array) d'un turbo codeur-décodeur en blocs de faible complexité pour des applications à haut débit (*i.e.* > 25Mbps). Le code retenu pour l'implémentation est le code produit BCH étendu (32, 26, 4)² (résultant de la concaténation de deux codes BCH étendus (32,26,4)). Les simulations en langage C et la synthèse en VHDL ont permis de montrer que l'utilisation de la structure itérative à traitement par blocs pour l'implémentation du turbo codeur-décodeur peut atteindre un débit de 50 Mbits/s tout en ayant une faible complexité (*i.e.* < 4500 éléments logiques).

Abstract *– In this paper, we present an implementation on FPGA (Field Programmable Gate Array) of a turbo encoder-decoder of product code which is able to achieve high data rate (i.e. > 25 Mbits/s) and has a low complexity. For this implementation, we consider a product code BCH(32,26,4)⊗BCH(32,26,4). In order to reach a high data rate, we use the reiterated structure with blocks processing that has nevertheless a high complexity. We propose afterwards to apply a pipeline structure to reduce the complexity of the turbo decoder. The simulation in the C language and the synthesis with VHDL language shows that our implementation of a turbo encoder-decoder of product code can reach a data rate of 50 Mbits/s and have a low complexity (less than 4500 logic elements).*

1. Introduction

Le développement des services multimédia, des terminaux portables et des besoins associés en terme de capacité des réseaux et de haut débit avec un spectre de plus en plus encombré, impliquent une contrainte forte sur la simplicité des récepteurs. Il faut d'une part limiter la consommation des terminaux et donc augmenter leurs autonomies et d'autre part diminuer leurs coûts. Cette contrainte sur la complexité, même pondérée par le progrès de la technologie, est contradictoire avec l'augmentation des débits. Dans les systèmes numériques actuels, le codage de canal, et en particulier le décodage, sont des opérations coûteuses en temps de traitement et en complexité. Aujourd'hui, les codes correcteurs d'erreurs les plus utilisés sont les codes convolutifs, parfois associés à un code Reed-Solomon. C'est le cas du GSM (Global System for Mobile), du DAB (Digital Audio Broadcasting), du DVB (Digital Video Broadcasting) terrestre et satellite. En 1994, R. Pyndiah et *al*. [1] ont proposé une nouvelle méthode de décodage des codes produits, les turbo codes en blocs. Ces codes correcteurs d'erreurs présentent l'avantage d'avoir des performances proches de la limite théorique de Shannon pour des rendements de codage élevés [1], et satisfont alors la contrainte de grande efficacité spectrale.

Après avoir rappelé le principe des codes produits, ce papier présente l'implémentation sur FPGA d'un turbo codeur-décodeur du code produit BCH étendu (32,26,4)² à haut débit. L'implémentation est réalisée selon la structure itérative à traitement par blocs. Une méthode d'optimisation du rapport complexité/performances et une organisation des mémoires adaptée au traitement par blocs sont ensuite proposées.

2. Codeur des codes produits

2.1 Les codes produits

Les codes produits, introduits par P. Elias en 1954 [2], sont des codes de fort pouvoir de correction, obtenus par concaténation série de deux ou plusieurs codes en blocs linéaires de faible pouvoir de correction. Considérons deux codes en blocs élémentaires $\mathbf{C}1$ et $\mathbf{C}2$ de paramètres respectifs (n_1,k_1,d_1) et (n_2,k_2,d_2), où n_i représente la longueur, k_i la dimension et d_i la distance minimale de Hamming du code $\mathbf{C}i$. Les paramètres (n,k,d) du code produit $\mathbf{C} = \mathbf{C}1 \otimes \mathbf{C}2$ sont alors égaux à $n = n_1 \times n_2$, $k = k_1 \times k_2$ et $d = d_1 \times d_2$. Son rendement est aussi égal au produit des rendements des codes $\mathbf{C}1$ et $\mathbf{C}2$. Pour une application aux réseaux à haut débit, tels que les réseaux locaux sans fil HIPERLAN/2 (High PERformance Local Area Network) [3], les codes retenus pour construire les codes produits sont des codes BCH étendus (32,26,4) et $\mathbf{C}1 = \mathbf{C}2$. Dans ce cas, le code produit BCH étendu (32,26,4)² se présente alors sous forme d'une matrice carrée $[\mathbf{C_e}]$ de n_e lignes et n_e colonnes (avec $n_e = n+1$), dans laquelle :

- les $k \times k = 26 \times 26$ bits d'information forment la matrice $[\mathbf{M}]$,
- chacune des k lignes de la matrice $[\mathbf{M}]$ est codée en utilisant le code BCH(31,26,3),
- chacune des n colonnes de la matrice $[\mathbf{C}]$ est codée par le code BCH(31,26,3),
- la dernière ligne et la dernière colonne de la matrice $[\mathbf{C_e}]$ sont respectivement constituées des bits de parités sur les lignes, des bits de parité sur les colonnes et du bit de parité sur les bits de parité.

La figure 1 présente un code produit BCH étendu (n_e, k, d_e)², avec $d_e = d + 1$.

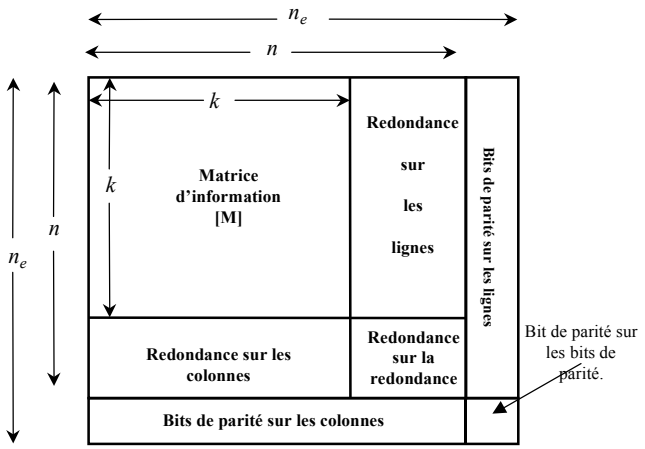

Figure 1 : principe du codage d'un code produit étendu [**Ce**].

2.2 Architecture du codeur des codes produits

Nous supposons que les bits des matrices [**M**] et [$\mathbf{C_e}$] sont respectivement transmis à l'entrée du codeur et disponibles à sa sortie, bit par bit, dans l'ordre de gauche à droite et de haut en bas. Afin de respecter cet ordre de transmission et d'après le concept du code produit BCH étendu (32,26,4)², nous déduisons le schéma bloc du circuit de codage présenté sur la figure 2 et constitué de :

- un circuit "codeur ligne" qui sert à calculer la redondance sur les lignes,
- un circuit "codeur colonne" qui calcule la redondance sur les colonnes et la redondance sur la redondance,
- deux circuits de calcul des bits de parité sur les lignes et sur les colonnes,
- un multiplexeur pour sélectionner les bits à transmettre dans l'ordre prédéfini,
- une mémoire RAM de capacité 32x6 pour sauvegarder la redondance sur les colonnes, la redondance sur la redondance et les bits de parités sur les colonnes,
- un séquenceur qui coordonne tous les circuits du codeur en générant les adresses de la mémoire RAM, les signaux de commande « demande de donnée », la sélection du multiplexeur, etc..

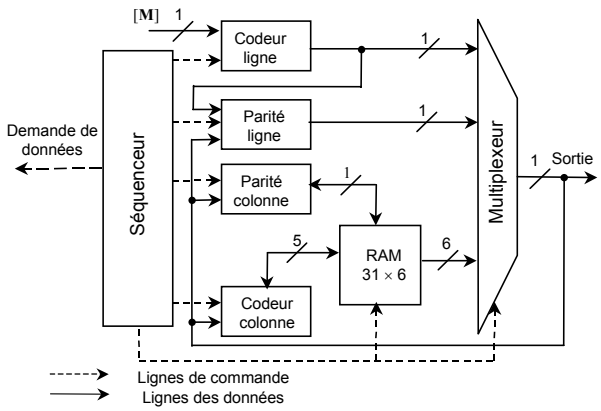

Figure 2 : schéma bloc du codeur du code produit BCH étendu (32,26,4)².

3. Turbo-décodeur en blocs

3.1 Algorithme de turbo-décodage des codes produits

L'algorithme de turbo décodage est basé sur le concept de contre-réaction et associe l'algorithme de décodage à entrées pondérées de Chase [4] à un calcul de pondération en sortie du décodeur selon l'algorithme de Pyndiah [1].

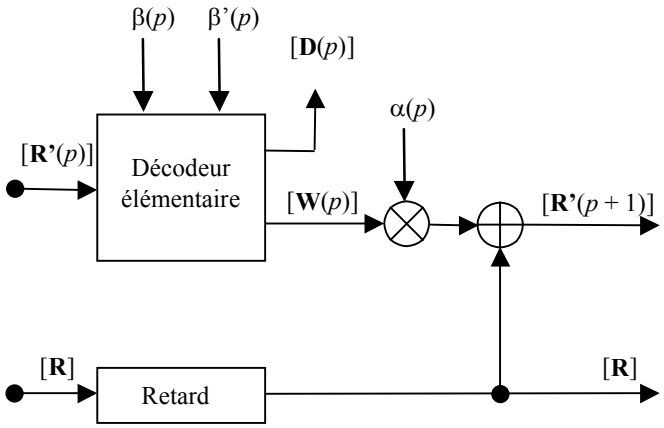

Figure 3 : schémas de principe d'un décodeur élémentaire.

Cet algorithme utilise des décodeurs élémentaires (*cf.* figure 3) mis en cascade dont les différents paramètres et données sont :

- [**R**] la matrice reçue,
- [**W**(p – 1)] la matrice d'information extrinsèque obtenue par le décodage précédent,
- [**R'**(p)] = [**R**] + α(p) [**W**(p – 1)],
- [**D**(p)] la matrice décidée,
- α(p) et β(p) des constantes, fonctions de p.

À la $p^{\text{ème}}$ demi-itération, le décodage de l'$i^{\text{ème}}$ ligne (ou colonne) $\mathbf{R}_i$'(p) de la matrice [**R**'(p)] comporte les étapes suivantes :

a) Décodage de type Chase pour déterminer un mot de code décidé **D** à distance euclidienne minimum du mot reçu $\mathbf{R}_i$'(p). Ce décodage consiste à :
- sélectionner et mémoriser la position et le module des m composantes les moins fiables (notées MF1, MF2,…, MFm) du mot reçu $\mathbf{R}_i$'(p),
- générer les t vecteurs de test $\mathbf{Y}^j$, $j \in [1,t]$, par inversion d'une ou de plusieurs positions des composantes les moins fiables dans le mot binaire $\mathbf{Y}^0$ obtenu par seuillage du mot reçu $\mathbf{R}_i$'(p),
- décoder $\mathbf{Y}^0$ et les t vecteurs de test $\mathbf{Y}^j$ par algorithme de décodage binaire pour obtenir les (t + 1) mots décodés $\mathbf{C}^k$ avec $k \in [0,t]$,
- calculer la distance euclidienne entre $\mathbf{C}^k$ et le mot reçu $\mathbf{R}_i$'(p),
- sélectionner parmi les mots de code $\mathbf{C}^k$, le mot de code décidé **D** dont la distance euclidienne M^d est minimale,

b) Calcul de pondération. Ce calcul est basé sur l'estimation du Logarithme du Rapport de Vraisemblance de chaque bit d_j du mot décidé **D**(p) et nécessite de :

- rechercher parmi les mots de code $\mathbf{C}^k$, un mot de code $\mathbf{C}^c$, appelé le concurrent, dont la distance euclidienne M^c est minimale et tel que $c_i^c \neq d_i$,
- calculer l'information extrinsèque $w_{ij} = (M^d - M^c) \times d_j - r'_{ij}$ si le mot $\mathbf{C}^c$ est trouvé, sinon $w_{ij} = \beta(p) \times d_j - r'_{ij}$ (w_{ij} et r'_{ij} étant respectivement les $j^{ème}$ éléments du vecteur $\mathbf{W}_i(p)$ et du mot reçu $\mathbf{R}_i'(p)$).

Dans l'objectif d'obtenir un haut débit (*i.e.* > 25 Mbits/s) avec une faible complexité, nous avons retenu la version simplifiée de l'algorithme de Chase-Pyndiah proposée dans [5], [6] qui utilise 8 vecteurs de test. Cette version simplifiée consiste à prendre, parmi les mots décodés $\mathbf{C}^k$, le deuxième mot décodé à distance euclidienne minimale du mot reçu comme l'unique concurrent. Elle permet ainsi de diminuer la complexité du calcul de pondération par 10 sans dégrader significativement les performances du décodeur [5]. La figure 4 représente une comparaison des performances entre l'algorithme Chase-Pyndiah optimal et simplifié après 4 itérations de turbo décodage du code BCH(32,26,4)².

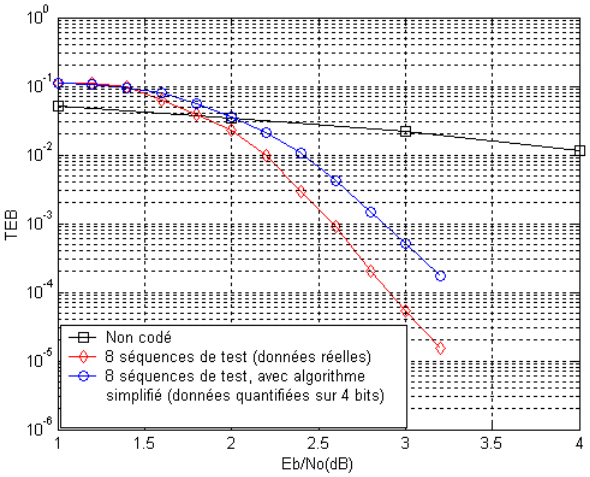

Figure 4 : performances de turbo-décodage des codes produits BCH(32,26,4)² après la 4ème itération.

Le décodage complet met en œuvre 4 itérations. Ainsi le temps alloué pour le traitement d'un vecteur est de 4 périodes d'horloge. Les symboles reçus r_{ij} sont quantifiés sur 4 bits (1 bit pour le signe et 3 bits pour le module).

3.2 Architectures du turbo-décodeur

Plusieurs architectures sont possibles pour implémenter l'algorithme Chase-Pyndiah décrit précédemment. Parmi ces architectures, la structure itérative à traitement par blocs [5] est la plus adaptée pour une application aux réseaux de haut débit tout en ayant une complexité raisonnable. En effet, en utilisant un seul décodeur élémentaire pour effectuer plusieurs itérations de décodage, elle réduit non seulement la complexité et la latence globale du circuit (2 fois le temps de remplissage d'une matrice), mais elle permet également d'avoir un débit élevé. Pour implémenter le décodeur élémentaire, nous avons tout d'abord transcrit en VHDL les étapes de l'algorithme Chase-Pyndiah décrit au paragraphe précédent en plusieurs blocs avant de les assembler comme le montre la figure 5.

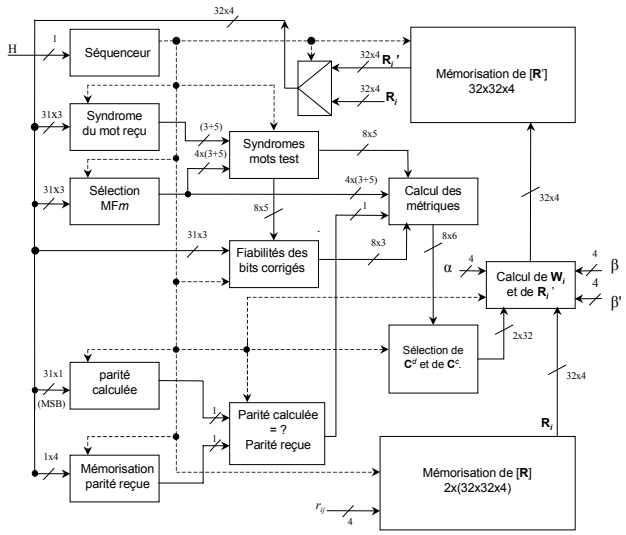

Figure 5 : schéma bloc du décodeur élémentaire.

Le blocs « mémorisation de [**R**] » et « mémorisation de [**R'**] » se composent chacun de deux RAM travaillant alternativement en mode d'écriture et de lecture. Tous les blocs du décodeur sont coordonnés par le bloc « séquenceur ».

3.3 Mémorisation des matrices [R] et [R']

Pour le traitement et la mémorisation des matrices [**R**] et [**R'**], la structure itérative à traitement par blocs requiert des mémoires à écriture et à lecture en ligne comme en colonne permettant d'accéder à un vecteur entier (de dimension n_e) en une seule période d'horloge. Ce double mode d'accès ligne/colonne peut être réalisé en organisant la mémoire en n_e plans parallèles et en insérant un système de décalage sur les entrées et les sorties des données. La figure 6 montre le concept de ces mémoires à double accès ligne/colonne de capacité $n \times n \times q$ bits.

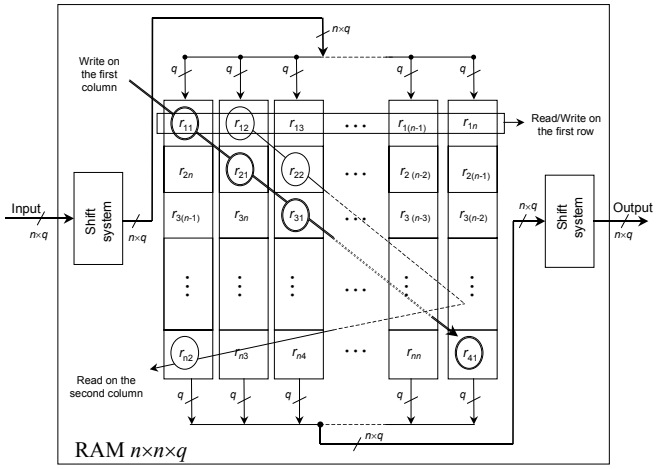

Figure 6: concept des mémoires de capacité $n \times n \times q$ bits pouvant être lues et écrites par vecteur en ligne et en colonne.

Le concept de ces mémoires est basé sur le principe que les bits d'un même vecteur ne doivent pas être sauvegardés sur le même plan afin d'accéder à un vecteur entier en une

seule période d'horloge. Pour cela, lors de l'écriture, les données doivent être décalées, à l'aide des systèmes de décalage, de manière rotative dans un sens et puis, lors de la lecture, dans le sens inverse (sauf la première ligne et la première colonne).

3.4 Optimisation du décodeur

En vue d'optimiser l'implémentation du décodeur, nous proposons une modification de l'architecture améliorant la cadence de traitement tout en réduisant la complexité. En effet, avec la structure itérative à traitement par blocs, l'intégration de l'étape de sélection des composantes les moins fiables (*cf.* §3.1) est très encombrante (27% de la taille totale du décodeur). De plus, son temps de calcul est long et pénalise le débit du décodeur. Afin de résoudre ces deux problèmes, nous avons tout d'abord partitionné le fonctionnement de la sélection des composantes les moins fiables de façon à mettre en place une structure pipeline pour diminuer le temps critique. Ensuite, grâce à ce partitionnement, les blocs « Sélection MF*m* », « syndromes mots test », « fiabilité des bits corrigés » et « Calcul des métriques » (*cf.* figure 4) peuvent être réalisés de manière itérative, ce qui permet de réduire la complexité totale du codeur de plus de 50% et d'augmenter débit maximal jusqu'à 50 Mbits/s (*cf.* tableau 1).

Tableau 1 : encombrement du turbo codeur-décodeur du code produit BCH étendu $(32,26,4)^2$.

	Débit maximal Mbits/s	Éléments logiques	RAM (bits)
Codeur	64	140	192
Décodeur	50	4251	18032

4. Conclusion

Ce papier a montré que l'implémentation d'un turbo décodeur du code produit BCH étendu $(32,26,4)^2$ selon la structure itérative traitement par blocs peut satisfaire les contraintes de haut débit (*i.e.* > 25 Mbits/s) et de faible complexité (moins de 4500 éléments logiques). Nous avons également exposé la structure du codeur du code produit BCH étendu $(32,26,4)^2$. Grâce à sa faible complexité, il peut être implémenté sur le même composant FPGA que le décodeur.

Remerciements

Cette étude a été financée par le laboratoire Information Telecom Europe (ITE) de Mitsubishi Electric, Rennes.

Références

[1] R. PYNDIAH, A.GLAVIEUX, A. PICART, S. JACQ, "Near optimum decoding of product codes", *Globecom'94*, *San Fransisco*, 1994.

[2] P. ELIAS, "Error free coding", *IRE Trans. On Inf. Theory*, vol. IT-4, pp. 29-37, September 1954.

[3] N. CHAPALAIN-LE FLOC'H, "Application des Turbo Codes en Blocs pour les réseaux locaux sans fil à haut débit", *thèse de l'Université de Bretagne Occidentale*, 2002.

[4] D. CHASE, "A class of algorithms for decoding block codes with channel measurement information", *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 170-182, no.1, January 1972.

[5] P. ADDE, R. PYNDIAH, "Recent simplifications and improvements in Block Turbo Codes", *Proceedings of the 2nd International Symposium on Turbo Codes*, pp. 133-136, France, 2000.

[6] O. RAOUL, "Conception et performances d'un circuit intégré turbo décodeur de codes produits", *thèse de l'Université de Bretagne Occidentale*, novembre 1997.