ARCHITECTURE ET CONCEPTION D'UN CIRCUIT TURBO-DECODEUR DE CODES PRODUITS

Olivier RAOUL (*), Patrick ADDE (*) et Ramesh PYNDIAH (**)
Département Electronique (*) - Département Signal et Communications (**)

Ecole Nationale Supérieure des Télécommunications de Bretagne
BP 832 - 29285 Brest Cedex - France

RESUME
Cet article aborde l'intégration sur silicium d'un algorithme original de décodage itératif des codes produits à entrée et sorties pondérées. Les performances de cet algorithme en terme de taux d'erreurs binaires sont aussi bonnes que celles des turbo-codes convolutifs. Les solutions proposées, tant pour l'architecture du circuit que pour la mémorisation des résultats et données, permettent de conclure à la faisabilité d'un circuit turbo-décodeur de codes produits. De plus nos simulations montrent que la dégradation des performances du circuit est très faible malgré les contraintes d'intégration (quantification, etc...).

ABSTRACT
This paper deals with the integration of a new iterative decoding algorithm for product codes based on soft decoding and soft decision of the component codes. The performance of this algorithm in terms of Bit Error Rate (BER) is equivalent to that of convolutionnal turbo-codes. The solutions we have proposed for the architecture of the circuit as well as for the memorization of the data indicate the block turbo-decoder circuit is feasible. Moreover, our simulations show that the performance degradation of the circuit due to integration constraints (quantization...) is negligible.

1. INTRODUCTION

Les turbo-codes, inventés par C. Berrou et *al* [1] à Télécom Bretagne, définissent une nouvelle classe de codes correcteurs d'erreurs dont le pouvoir de correction avoisine, dans des conditions de codage idéales, la limite théorique prévue par C. E. Shannon. Ces codes sont obtenus par la concaténation parallèle de deux codes convolutifs systématiques récursifs. Leur décodage utilise un algorithme itératif à entrée et sortie pondérées [2] s'appuyant sur la notion de contre-réaction, rarement employée pour le codage correcteur d'erreurs. La contre-réaction permet de générer une information redondante, dite extrinsèque, utilisée lors du décodage pour en améliorer les performances. Depuis leur invention en 1992, les turbo-codes ont suscité l'intérêt de la communauté scientifique, notamment pour les applications de diffusion numérique de l'information, et font l'objet de nombreuses publications. Une première réalisation sur silicium [3] a été menée à son terme avec succès et a permis de valider le concept de turbo-code. Les excellents résultats obtenus par C. Berrou ont incité R. Pyndiah et *al* [4] à concevoir un équivalent des turbo-codes convolutifs pour les codes en blocs. Leur choix s'est orienté vers une classe particulière de codes en blocs, les codes itérés appelés aussi codes produits. En 1994, R. Pyndiah [5] a proposé pour le décodage des codes produits un algorithme itératif reprenant les concepts de base du turbo-décodage. Les performances de cet algorithme se sont avérées comparables à celles des turbo-codes convolutifs. Dans cet article nous présentons les premiers résultats de l'étude de faisabilité d'un circuit pour le turbo-décodage des codes produits.

2. RAPPELS SUR LES CODES PRODUITS

Les codes produits, inventés par P. Elias en 1954 [6], sont de puissants codes correcteurs d'erreurs (ayant une grande distance minimale de Hamming). Ils sont construits par concaténation de deux ou plusieurs codes en blocs linéaires de faible pouvoir de correction. Considérons deux codes en blocs élémentaires C^1 et C^2 systématiques, de paramètres respectifs (n_1,k_1,d_1) et (n_2,k_2,d_2), où n_i représente la longueur du code C^i, k_i la longueur du message et d_i la distance minimale de Hamming de C^i. Le code produit $C=C^1 \otimes C^2$ se présente alors sous la forme d'une matrice **[C]** de n_1 lignes et n_2 colonnes dans laquelle:
- les symboles binaires d'information définissent une sous matrice **[M]** de k_1 lignes et k_2 colonnes,
- chacune des k_1 lignes de la matrice **[M]** est codée par le code C^2,
-les n_2 colonnes de la matrice ainsi formée sont codées en utilisant le code C^1 pour former **[C]**,
La figure 1 illustre le principe de construction d'un code produit.

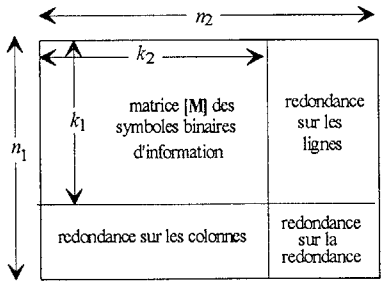

figure 1 : principe du codage d'un code produit

Les paramètres (n,k,d) du code produit C s'obtiennent à partir de ceux des codes C^1 et C^2 par $n=n_1 \times n_2$, $k=k_1 \times k_2$ et $d=d_1 \times d_2$. Le rendement du code produit est égal à $R=R_1 \times R_2$ où R_i est le rendement de C^i. Les paramètres de C s'expriment donc comme le produit des paramètres des codes élémentaires utilisés pour le construire. Une propriété intéressante de ces codes produits, construits à partir de codes en blocs linéaires, vient du fait que d'une part les n_1 lignes de la matrice codée **[C]** sont des mots du

code C^2 et que d'autre part ses n_2 colonnes sont des mots du code C^1. Cette propriété s'avère utile lors du décodage itératif que l'on se propose de mettre en oeuvre puisqu'ainsi toute ligne ou colonne de la matrice **[C]** peut être décodée à chaque itération. Les codes élémentaires retenus par la suite sont, pour des raisons de compromis performances/complexité, des codes BCH étendus de pouvoir de correction t=1 (codes de Hamming) ou t=2, en se plaçant dans la situation où C^1=C^2.

3. PRINCIPE DE L'ALGORITHME DE DECODAGE ITERATIF DES CODES PRODUITS

Depuis leur invention, peu de solutions ont été proposées pour le décodage des codes produits. Celle décrite par S.M. Reddy [7] en 1970 s'avère sous-optimale par rapport au critère du Maximum de Vraisemblance a Posteriori (MVP) et ne permet pas de bénéficier pleinement de la puissance de ces codes. Une autre méthode, itérative, élaborée par J. Lodge et *al.* [8] en 1993 et basée sur l'algorithme de Bahl, conduit à des gains de codage supérieurs. Elle ne s'applique malheureusement en pratique qu'à des codes de longueur faible et n'est donc pas utilisable pour des codes de rendement élevé. L'algorithme itératif développé récemment par R. Pyndiah et *al.*, utilisant le concept de contre-réaction, associe un décodage pseudo-soft des échantillons analogiques présentés en entrée à une méthode originale de mesure de la fiabilité des décisions prises en sortie, basée sur l'estimation du Logarithme du Rapport de Vraisemblance (LRV). Nous allons maintenant présenter cet algorithme qui permet d'atteindre les performances asymptotiques optimales des codes produits.

3.1 Décodage à entrée pondérée

Un décodage à entrée pondérée permet d'exploiter au mieux les capacités de correction d'un code. La solution retenue ici est un décodage de type Chase [9]. Pour le mettre en oeuvre il faut successivement :

1. Repérer la position des p composantes les moins fiables (les plus proches de zéro, seuil de décision binaire) du vecteur de données **R**=(r_1,...,r_n), ligne ou colonne de la matrice codée **[R]** reçue en entrée du décodeur. Ces positions notées $I_1,I_2,...,I_p$ sont telles que: $|r_{I_1}| < |r_l| \ \forall\, l \neq I_1$, $|r_{I_2}| < |r_l| \ \forall\, l \neq I_1, I_2$, ... $|r_{I_p}| < |r_l| \ \forall\, l \neq I_1, I_2, \ldots, I_p$. On génère ainsi p séquences de test élémentaires $\mathbf{T}_e^l$, mots ayant un "1" dans la position I_l et des "0" ailleurs.
2. Combiner linéairement les p $\mathbf{T}_e^l$, pour former q séquences de test $\mathbf{T}^l$.
3. Pour chaque mot de test $\mathbf{T}^l$ on construit le mot binaire $\mathbf{Z}^l$ donné par la relation $\mathbf{Z}^l$=$\mathbf{T}^l$⊕**Y** dans laquelle **Y** est le vecteur binaire obtenu par seuillage de la séquence reçue **R**.
4. Décoder algèbriquement les q mots $\mathbf{Z}^l$, par l'algorithme de Berlekamp (ou équivalent) pour les codes BCH [10], et vérifier quels mots binaires sont bien des mots du code élémentaire utilisé. On les note alors $\mathbf{C}^i$ ($i \leq q$).
5. Pour chaque mot de code $\mathbf{C}^i$, calculer le carré de la distance euclidienne M^i entre $\mathbf{C}^i$ et **R** défini par :

$$M^i = \left\| \mathbf{R} - \mathbf{C}^i \right\|^2 = \sum_{j=1}^{n} \left(r_j - c_j^i \right)^2 \qquad (1)$$

6. Sélectionner le mot de code $\mathbf{C}^d$ à distance minimale de **R**. **D**=$\mathbf{C}^d$ est alors la nouvelle décision binaire.

Le décodage de Chase présenté ci-dessus délivre en sortie, conformément au critère du MVP appliqué aux mots de code $\mathbf{C}^i$ possibles, la séquence binaire la plus probablement émise pour un vecteur d'entrée **R** de données pondérées.

3.2 Algorithme de décodage à sortie pondérée

L'algorithme de décodage doit par ailleurs être à sortie pondérée afin que le processus itératif soit optimal à chaque décodage. Il faut associer une mesure de fiabilité à chaque bit d_j de **D**. Cette pondération peut être déterminée en sortie du décodeur à partir du LRV de la décision d_j :

$$\mathrm{LRV}_j = \ln\left(\frac{\Pr\{d_j = +1 / \mathbf{R}\}}{\Pr\{d_j = -1 / \mathbf{R}\}} \right) \qquad (2)$$

En considérant que les mots de code $\mathbf{C}^i$ sont équiprobables on peut montrer, par application de la règle de Bayes et pour une transmission sur canal de Gauss, que le LRV_j du $j^{ième}$ élément binaire de **D** s'écrit encore :

$$\mathrm{LRV}_j = \ln\left(\frac{\sum_{\mathbf{C}^i \in S_j^{+1}} \exp\left(-\frac{\left| \mathbf{R} - \mathbf{C}^i \right|}{2\sigma^2} \right)}{\sum_{\mathbf{C}^i \in S_j^{-1}} \exp\left(-\frac{\left| \mathbf{R} - \mathbf{C}^i \right|}{2\sigma^2} \right)} \right) \qquad (3)$$

Dans cette expression S_j^s représente l'ensemble des mots de code $\mathbf{C}^i$ ayant un bit c^i_j en position j égal à s, avec s=±1. Pour des codes en blocs de grande dimension, permettant d'atteindre des rendements élevés, cette expression devient rapidement trop complexe à implanter. En ne conservant alors que les deux mots de code $\mathbf{C}^{min(+1)}$ et $\mathbf{C}^{min(-1)}$ à distance euclidienne minimale de **R** et appartenant respectivement à S_j^{+1} et S_j^{-1} , l'expression du LRV_j se simplifie pour donner l'expression approchée :

$$\mathrm{LRV}_j = \frac{1}{2\sigma^2} \left(\left| \mathbf{R} - \mathbf{C}^{min(-1)} \right|^2 - \left| \mathbf{R} - \mathbf{C}^{min(+1)} \right|^2 \right) \qquad (4)$$

En développant suivant les composantes de **R**, on a alors :

$$\mathrm{LRV}_j \approx \frac{2}{\sigma^2} \left(r_j + \sum_{l=1, l \neq j}^{n} r_l c_l^{min(+1)} \rho_l \right) \qquad (5)$$

avec $\rho_l = \begin{cases} 0 \ si \ c_l^{min(+1)} = c_l^{min(-1)} \\ 1 \ si \ c_l^{min(+1)} \neq c_l^{min(-1)} \end{cases}$

où $c_l^{\min(+1)}$ et $c_l^{\min(-1)}$ sont respectivement les symboles en position l des mots $\mathbf{C}^{min(+1)}$ et $\mathbf{C}^{min(-1)}$.

En normalisant le LRV_j par rapport à $2/\sigma^2$ on a enfin :

$$r'_j = \frac{\sigma^2}{2} LRV_j = r_j + w_j \qquad (6)$$

avec $w_j = \sum_{l=1, l \neq j}^{n} r_l c_l^{min(+1)} \rho_l$.

Le LRV_j normalisé r'_j associé à la décision d_j s'exprime donc comme la somme de l'échantillon r_j reçu en entrée du décodeur et d'un terme w_j, représentant l'apport du décodeur, qui est l'équivalent de l'information extrinsèque des turbo-codes convolutifs. r'_j est une estimation de la décision pondérée prise en sortie par le décodeur. Il est du même signe que la décision d_j et sa valeur absolue reflète la fiabilité de cette décision. Calculer le LRV_j normalisé suppose que l'on a bien identifié les deux mots de code $\mathbf{C}^{min(+1)}$ et $\mathbf{C}^{min(-1)}$. En fait, après avoir sélectionné le mot $\mathbf{C}^d$ à distance minimale M^d de **R**, on va rechercher pour chaque bit de $\mathbf{C}^d$ parmi les mots de code $\mathbf{C}^i$, un mot $\mathbf{C}^c$ appelé mot concurrent à distance minimale M^c de **R** et vérifiant $c^c_j \neq c^d_j$.. Si ce mot existe on a :

$$r'_j = \left(\frac{\left(M^c - M^d\right)}{4} \right) c_j^d \qquad (7)$$

Si aucun concurrent n'a pu être trouvé, on utilise la relation

$$r'_j = \beta . c_j^d \qquad (8)$$

où β est une constante positive, fonction de la confiance accordée au décodage effectué et croissante avec les itérations.

3.3 Décodage itératif des codes produits

Avec un code produit on reçoit en entrée du décodeur une matrice de données **[R]** de n_1 lignes et n_2 colonnes. Un premier décodage des colonnes (ou des lignes) de **[R]** par l'algorithme proposé permet de déterminer la matrice **[W]** des informations extrinsèques. On décode de même les lignes (ou les colonnes) de la matrice **[R']**=**[R]**+α**[W]**, où α est un second coefficient de confiance positif, variable avec l'itération, et utilisé pour pondérer la contribution de l'information extrinsèque dont la fiabilité est moindre lors des premières itérations. On obtient ainsi une nouvelle matrice **[W]**. C'est cette procédure que l'on va itérer, une itération comprenant donc un décodage des colonnes suivi d'un décodage des lignes de la matrice de données **[R']**. Une demi-itération peut alors se traduire par le schéma de la figure 2.

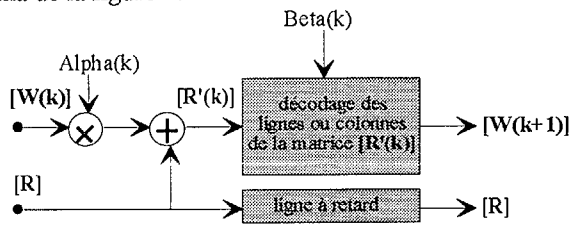

figure 2 : shéma de principe d'un décodeur élémentaire

3.4 Performances

Les performances de cet algorithme, en terme de Taux d'Erreurs Binaires (TEB), obtenues par des simulations de type Monte-Carlo, se sont avérées comparables voire meilleures que celles des turbo-codes convolutifs pour des taux de codage élevés (>0.8). A titre indicatif la figure 3 présente, pour différents codes produits construits à partir de codes BCH étendus, l'évolution du TEB en fonction du rapport signal à bruit E_b/N_0 après quatre itérations, dans le cas d'une transmission par modulation de phase à quatre états (MDP4) sur canal de Gauss.

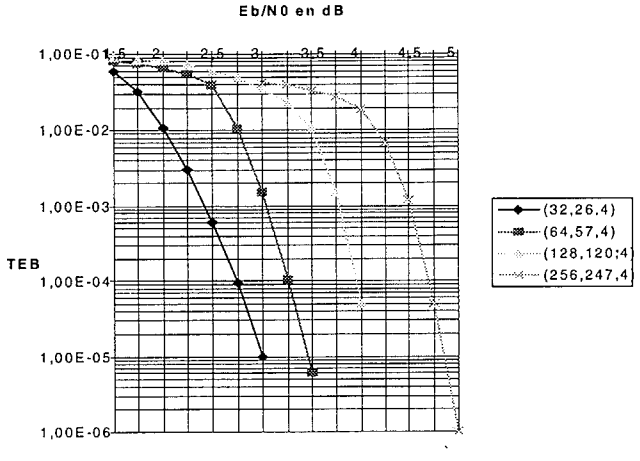

figure 3: performances des codes produits après 4 itérations.

3.5 Choix des séquences de test

Les séquences de test $\mathbf{T}^l$, ici au nombre de seize, ont été déterminées suivant la procédure décrite en **3.1** à partir de la position des cinq échantillons les moins fiables du vecteur reçu. En augmentant ce nombre de bits non fiables, et donc le nombre de séquences test dans l'algorithme de Chase, on accroît la probabilité de trouver un concurrent C^c à chaque bit de C^d. L'amélioration des performances qui en résulte s'effectue toutefois au détriment de la complexité.

3.6 Influence de la quantification

Il est important d'évaluer la dégradation du TEB, en fonction du nombre de niveaux de quantification, quand on passe de valeurs réelles à des valeurs discrètes. Un programme en langage C dont le fonctionnement modélise fidèlement celui du circuit a été écrit pour mesurer précisément la dégradation due aux contraintes d'intégration. Nous nous sommes particulièrement intéressés à l'évolution du TEB avec E_b/N_0 dans le cas du code produit BCH(64,57,4)×BCH(64,57,4) après quatre itérations. Les échantillons ont été codés successivement sur 3, 4 puis 5 bits (dont un bit de signe) avec des coefficients α et β optimisés dans chaque cas. Les conditions de simulation sont restées les mêmes que précédemment. Ces simulations indiquent que le meilleur compromis performances/complexité est obtenu en quantifiant les données sur quatre bits. Dans cette situation la dégradation est minime et reste inférieure à 0.1dB. Ceci s'explique par le fait que les approximations effectuées dans les relations (3) et (4) utilisées pour le calcul des LRV masquent l'effet de la quantification.

4. ARCHITECTURES

Différentes architectures sont possibles pour l'implantation. On peut ainsi imaginer un circuit constitué de modules identiques "pipelinés", chaque décodeur élémentaire (cf figure 2) réalisant alors une demi-itération. Cette solution, d'une grande souplesse d'utilisation, est similaire à celle retenue pour les turbo-codes convolutifs. Chaque module, cascadable, introduit une latence égale au temps de remplissage d'une matrice augmenté du délai nécessaire au décodage. La structure résultante sera dite "à traitement séquentiel" car elle autorise un traitement symbole par symbole des données. Plusieurs étapes du décodage d'une ligne ou d'une colonne de la matrice reçue (comme le calcul de syndrome, la recherche de la position des échantillons non fiables etc...) se traitent naturellement de manière séquentielle et sont donc bien adaptées à cette structure. Suivant la surface silicium occupée par une demi-itération on peut imaginer d'intègrer plusieurs itérations dans le même circuit. Une autre possibilité consiste à intègrer directement plusieurs itérations dans une même puce en utilisant cette fois une seule unité de traitement pour l'ensemble des itérations. Les résultats obtenus par simulation montrent que quatre itérations semblent un bon compromis performances/complexité. Le traitement peut ici s'effectuer symbole par symbole ou vecteur par vecteur. Dans le premier cas, le "traitement séquentiel" impose à la fréquence F_t de fonctionnement interne du circuit de vérifier F_t=2×nombre d'itérations×F_d. La fréquence F_d de réception des données se trouve donc limitée par le nombre d'itérations que l'on prévoit d'intègrer et par la fréquence maximale de fonctionnement liée à la technologie utilisée. La seconde solution, dite "à traitement par bloc", plus complexe à mettre en oeuvre, présente l'avantage de limiter la latence globale du circuit. Cette solution, propre aux codes en blocs, permet de réduire le temps nécessaire au décodage d'un vecteur de données. Elle requiert toutefois une architecture particulière des mémoires de stockage

de **[R]** et **[W]**, afin de conserver une fréquence utilisateur élevée. Contrairement au "traitement séquentiel", le "traitement par bloc" pose certains problèmes d'intégration. Les solutions retenues jusqu'ici pour la recherche des positions des composantes non fiables et le choix des mots de code concurrents, s'avèrent trop complexes pour pouvoir envisager, avec les technologies actuellement disponibles, une intégration avec des codes de longueur supérieure à 128.

5. MEMORISATION DES DONNEES ET DES RESULTATS

Indépendamment de la structure retenue, il est indispensable de mémoriser les matrices **[R]** et **[W]** pour l'itération en cours. Une itération comprenant un décodage des colonnes suivi d'un décodage des lignes, il convient de prévoir une mémoire pour stocker les données qui arrivent pendant que l'autre permet de traiter celles reçues précédemment. Un plan mémoire, pour un code de longueur n et des échantillons codés sur q bits est d'une taille de $n \times n \times q$ bits.

5.1 Structure à traitement séquentiel

L'organisation générale du circuit peut être celle de la figure 4.

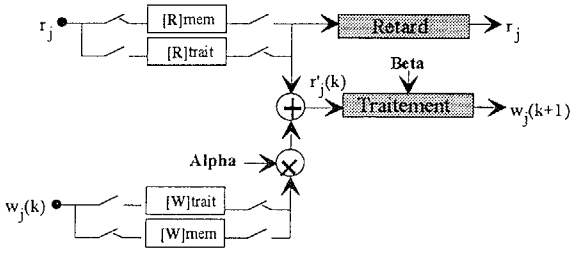

figure 4 : organisation générale de la structure à traitement séquentiel.

Compte tenu de la surface nécessaire, l'implantation des mémoires dans le circuit pour $n \geq 128$ oblige à utiliser une technologie CMOS 0.35µm. On a en effet besoin de quatre plans mémoires par demi-itération, deux pour **[R]** et deux pour **[W]**.

5.2 Structure à traitement par bloc

Le principal avantage de cette architecture est, comme on l'a vu, de conserver une fréquence débit élevée et de limiter la latence globale du circuit en intégrant plusieurs itérations dans le même circuit. Cette solution impose néanmoins de disposer de mémoires pouvant être lues et écrites en ligne comme en colonne. A priori, la surface requise par ces mémoires est supérieure à celle des mémoires classiques mais on passerait, pour quatre itérations, de trente deux plans mémoires avec la structure modulaire à seulement quatre pour le traitement par bloc (deux pour **[R]** et deux pour **[W]**).

Il semble en conclusion qu'il ne soit pas possible d'envisager pour l'instant une intégration avec n=256. Eventuellement on pourrait utiliser des mémoires externes mais, afin d'éviter d'avoir des bus trop importants entre les circuits, seule la solution à traitement séquentiel serait alors à retenir.

6. CONCLUSION

Les premiers résultats de l'étude de faisabilité d'un circuit turbo-décodeur pour codes produits sont très encourageants. En effet nous avons constaté que les performances du circuit turbo-décodeur sont très proches de celles annoncées par R. Pyndiah et *al* [5] malgré les contraintes d'intégration. Ceci permet de donner une légère avance en terme de performances aux turbo-codes en blocs par rapport aux turbo-codes convolutifs d'un point de vue circuit pour des rendements élevés. D'autre part nous avons présenté deux architectures différentes pour l'implantation du turbo-décodeur bloc. La première architecture, "séquentielle", est analogue à celle utilisée par les turbo-décodeurs convolutifs. La seconde architecture, "par bloc", permet de diminuer considérablement le nombre de cycles nécessaires pour décoder un mot. De plus, on peut envisager d'effectuer le décodage de plusieurs lignes (ou colonnes) en parallèle pour réduire le temps nécessaire au décodage de la matrice. On pourra ainsi effectuer plusieurs itérations en utilisant la même mémoire de travail dans le circuit, ce qui réduit énormément la surface mémoire nécessaire. Une première estimation, en utilisant une technologie CMOS 0.8µm et en considérant un débit maximal (40 Mbits/s), donne, pour l'exemple du code produit BCH(64,57,4)×BCH(64,57,4), une surface silicium de 40mm^2 pour une demi-itération avec l'architecture séquentielle modulaire et de 70mm^2 pour quatre itérations avec la structure à traitement par bloc. Ces surfaces, relativement importantes, peuvent être réduites fortement en prenant des codes BCH de longueur n=32 ou 16 et en diminuant le nombre de vecteurs de test (8 ou 4). Le gain le plus significatif concerne la simplification de la méthode de pondération : des premières études le chiffrent à environ 80% pour une perte en performances de 0.1dB.

7. BIBLIOGRAPHIE

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes", Proc IEEE Int. Conf. on Communications (ICC' 93), Geneva, Switzerland, 1993, pp. 1064-1070.

[2] C. Berrou et P. Adde,"Procédé de décodage d'un code convolutif à maximun de vraisemblance et pondération des décisions, et décodeur correspondant", brevet France n°9105279, Europe n°92 460011.7, USA n°07/870,483, France Télécom, TDF, avril 1992.

[3] "CAS5093: Turbo encoder/decoder", Datasheet, COMATLAS, Chateaubourg, France, novembre 1993.

[4] R. Pyndiah, A. Glavieux et C. Berrou, "Décodage itératif de codes produits", brevet France N°93 13858, France Télécom, novembre 1993.

[5] R. Pyndiah, A. Glavieux, A. Picard et S. Jacq, "Near optimum decoding of product codes", IEEE proc. of GLOBECOM'94, San Francisco, Nov. 1994.

[6] P. Elias, "Error free coding", IRE Trans. On Inf. Theory, vol. IT-4, Sept. 1954, pp. 29-37.

[7] S. M. Reddy, "On decoding iterated codes", IEEE Trans. Inform. Theory, vol. IT-16, Sept. 1970, pp. 624-627.

[8] J. Lodge, R. Young, P. Hoeger and J. Hagenauer, "Separable MAP filters for the decoding of product and concatenated codes", Proc. ICC'93, Geneva, Switzerland, May 1993, pp. 1740-1745.

[9] D. Chase, "A class of algorithms for decoding block codes with channel measurement information", IEEE Trans. Inform. Theory, vol. IT-18, Jan. 1972, pp. 170-182.

[10] G. C. Clark, Jr. and J. B. Cain, "Error-correction coding for digital communications", New-York Plenum Press, 1982, pp. 188-214.