Décodage des codes LDPC par l'algorithme $\lambda-$Min

Frédéric GUILLOUD[1], Emmanuel BOUTILLON[2], Jean-Luc DANGER[1]

[1]LTCI Telecom Paris
46, rue Barrault, 75634 Paris Cedex 13, France

[2]LESTER, Université de Bretagne Sud
BP 92116, 56321 LORIENT Cedex, France

frederic.guilloud@enst.fr, emmanuel.boutillon@univ-ubs.fr, jean-luc.danger@enst.fr

Résumé – Le décodage optimal des codes LDPC met en oeuvre un algorithme itératif qui propage les informations extrinsèques et les informations a priori dans le graphe bipartite reliant les variables aux équations de parité. Nous proposons dans cet article tout d'abord une simplification du calcul de parité, qui facilite le décodage et son implementation architecturale, sans dégradation significatives des performances du décodage. Nous proposons aussi une architecture adaptée à cet algorithme qui permet décoder une grande classe de codes réguliers ou irréguliers.

Abstract – Optimal LDPC codes decoding is performed using an iterative algorithm : it propagates extrinsic and a priori information through the bipartite graph which is the link between variable nodes and parity check nodes. In this article, a simplification of the parity check processing is proposed, which makes the decoding and its architectural design easier, without any significant loss of performance. A new decoding architecture is also proposed which is well suited to the $\lambda-$min algorithm. Both regular and irregular codes can be decoded by this architecture.

Introduction

Les codes LDPC (low density parity check) sont des codes en blocs dont la matrice de parité comporte un faible nombre de 1 [6] ; ils ont été bien étudiés d'un point de vu théorique depuis leur redécouverte en 1995 [7], et les problèmes posés par leur intégration matérielle commencent à être abordés. Le choix d'un code LDPC pour la norme DVB-S2 rend le sujet de très grande actualité. Dans cet article, nous proposons une simplification du calcul des parités par un algorithme nouveau que nous avons appelé algorithme $\lambda-$min. Cet algorithme permet de réduire significativement la taille de la mémoire requise par le décodeur LDPC, et ce, sans dégradation significative de la performance. De plus, nous décrivons un mode de réalisation série du traitement des parités très bien adapté avec l'algorithme $\lambda-$min et permettant naturellement le décodage de codes LDPC irréguliers.

Après avoir rappelé le principe du décodage itératif optimal, nous présenterons l'algorithme $\lambda-$min et ses performances. Enfin, nous présenterons un processeur de parité réalisant l'algorithme $\lambda-$min selon un mode série. Les gains de mémorisation seront aussi donnés.

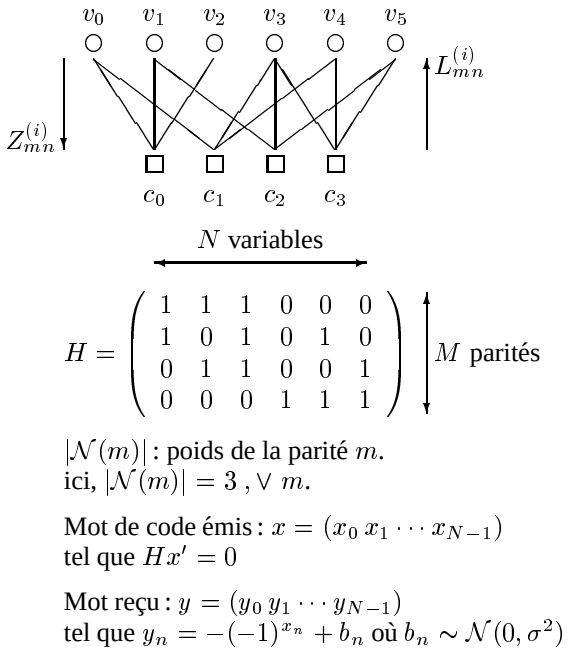

FIG. 1: *Exemple d'un graphe bipartite et de sa matrice de parité*

1 Décodage itératif optimal et notations

Les codes LDPC peuvent être modélisés par des graphes bipartites (figure 1) : il s'agit de graphes dont on peut séparer les noeuds en deux groupes : pour les codes LDPC, il y a d'une part N bits (variables) v_n et d'autre part M parités c_m (checks). Lorsqu'un bit apparaît dans une parité, les deux noeuds correspondants sont reliés.

En utilisant les notations de Fossorier et al. [5], [3], on pose $\mathcal{N}(m) = \{v_n : \ H_{mn} = 1\}$ l'ensemble des bits impliqués dans la parité c_m et soit $\mathcal{N}(m) \backslash n$ ce même ensemble sans le bit v_n.

De même, soit $\mathcal{M}(n) = \{c_m : \ H_{mn} = 1\}$ l'ensemble des parités dans lesquelles le bit v_n est impliqué, et soit $\mathcal{M}(n) \backslash m$ ce même ensemble sans la parité c_m. La cardinalité d'un ensemble $\mathcal{A}$ est noté $|\mathcal{A}|$.

Le décodage itératif des codes LDPC est basé sur la propagation de l'information des variables v_n à travers les parités c_m. Soit $Z_{mn}^{(i)}$ le logarithme du rapport de vraissemblance (LLR) du message allant du bit v_n à la parité c_m lors de la $i^{\text{ème}}$ itération (information a priori). Soit de même $L_{mn}^{(i)}$ le LLR du message allant de la parité c_m au bit v_n (information extrinsèque).

· **Initialisation :** Le décodage est initialisé par le LLR de chaque bit n :

$$L_n^{(0)} = \frac{2y_n}{\sigma^2},\ \sigma^2 = N_o/2. \tag{1}$$

· **Itérations :** Les itérations i comportent 2 étapes :

1. la mise à jour des informations a priori : pour tout v_n et pour tout $c_m \in \mathcal{M}(n)$:
$$Z_{mn}^{(i)} = Z_n^{(i)} - L_{mn}^{(i-1)} \tag{2}$$
où $Z_n^{(i)} = L_n^{(0)} + \sum_{m\in\mathcal{M}(n)} L_{mn}^{(i)}$ représente la valeur souple du bit v_n. Le mot de code reçu est $\hat{x}_i = \left\{ s\left(Z_n^{(i)}\right)\right\}_{1\leq n\leq N}$, où $s(a)$ représente le signe de a. On calcule aussi le syndrome $\xi_i(\hat{x}_i) = H\hat{x}_i'$.
2. la mise à jour des infomations extrinsèques : pour tout c_m et pour tout $v_n \in \mathcal{M}(n)$:
$$L_{mn}^{(i)} = S_{mn}^{(i)} \times M_{mn}^{(i)} \tag{3}$$
avec
$$S_{mn}^{(i)} = s\left(Z_{mn}^{(i)}\right) \times p_m^{(i)} \tag{4}$$
où $p_m^{(i)} = \prod_{n\in\mathcal{N}(m)} s\left(-Z_{mn}^{(i)}\right)$ désigne la parité de l'équation c_m, et
$$M_{mn}^{(i)} = -\bigoplus_{n'\in\mathcal{N}(m)\backslash n} \left(-\left|Z_{mn'}^{(i)}\right|\right) \tag{5}$$
où $\oplus$ est la fonction commutative et associative telle que [1] :
$$I_0 \oplus I_1 = \ln\frac{\exp(I_0)+\exp(I_1)}{1+\exp(I_0+I_1)} \tag{6}$$

L'algorithme itératif est terminé dès que le syndrôme calculé à la première étape est nul, ou bien si le nombre maximum prédéfini d'itérations est atteint.

Le calcul de la parité est complexe cat il nécessite des LUT (look up table) pour les fonctions non-linéaires. L'algorithme *BP-Based* proposé par M.P.C Fossorier et al. [5] simplifie l'équation (5) par :

$$M_{mn}^{(i)} = \min_{n'\in\mathcal{N}(m)\backslash n} \left|Z_{mn'}^{(i)}\right|. \tag{7}$$

Pour compenser la perte importante de performance introduite par cette simplification, une amélioration est apportée par l'ajout d'un coefficient correctif ([3] puis [4]). Mais la perte de performance introduite reste importante pour les codes plus proches de la limite de Shannon (codes irréguliers et de grandes tailles).

2 L'algorithme $\lambda-$min

Nous proposons dans cette partie l'algorithme $\lambda-$min. Cet algorithme permet différents compromis performances-complexités entre l'algorithme optimal et l'algorithme BP.

2.1 Présentation

L'équation (6) est modifiée afin de simplifier sa quantification :

$$I_0 \oplus I_1 = -\mathrm{s}(I_0)\mathrm{s}(I_1)\min\left(|I_0|,|I_1|\right) + \mathrm{f}(I_0-I_1) - \mathrm{f}(I_0+I_1) \tag{8}$$

On peut noter que la fonction f: $x \mapsto \log(1+\exp(-|x|))$ est exponentiellement décroissante. Ainsi, la valeur de $\left|L_{mn}^{(i)}\right|$ est fortement déterminée par les $\left|Z_{mn'}^{(i)}\right|$ les plus faibles. D'où l'idée de simplifier le calcul des parités en ne prenant en compte que les $\lambda\,(\lambda>1)$ informations a priori de plus faibles modules.

Soit $\mathcal{N}_\lambda^{(i)}(m) = \{n_0,\cdots,n_{\lambda-1}\}$ le sous-ensemble de $\mathcal{N}(m)$ qui contient les λ indices des bits de la parité c_m ayant les plus faibles informations a priori (en module) à l'itération i. On note aussi $\Omega_\lambda^{(i)}(m) = \{|Z_{mn}^{(i)}|,\ n\in\mathcal{N}_\lambda^{(i)}(m)\}$ l'ensemble des λ valeurs absolues minimales des informations a priori relatives à la parité m lors de l'itération i. On note de même $S^{(i)}(m) = \{s(L_{mn}^{(i)}),\ n\in\mathcal{N}(m)\}$ l'ensemble des signes de ces mêmes informations. Alors on approxime (5) par :

$$M_{mn}^{(i)} = -\bigoplus_{n'\in\mathcal{N}_\lambda(m)\backslash n} -\left|Z_{mn'}^{(i)}\right| \tag{9}$$

Notons que l'équation (9) génére seulement $\lambda+1$ [1] valeurs distinctes, contre $|\mathcal{N}(m)|$ pour l'équation (5). En effet, tous les bits n'appartenant pas à $\mathcal{N}_\lambda(m)$, reçoivent la même valeur extrinsèque calculée par (9). On note $E_\lambda^{(i)}(m) = \{e_j,\ j\in(0,1,\cdots,\lambda)\}$ l'ensemble des $\lambda+1$ informations extrinsèques calculées lors de la mise à jour des parités.

2.2 Résultats

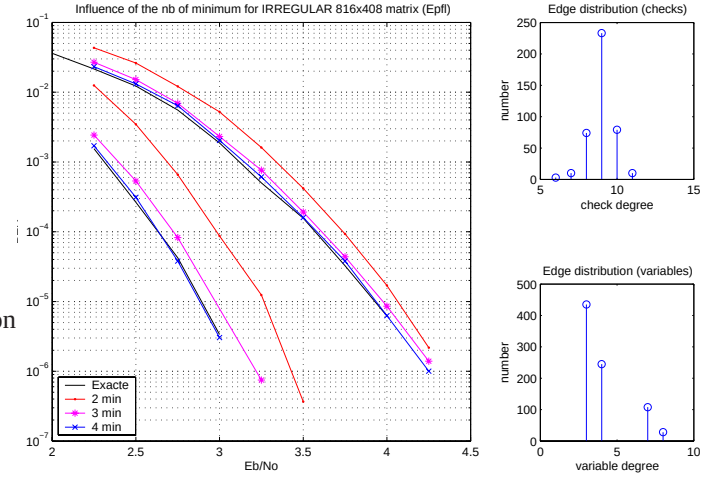

FIG. 2: *Comparaison des taux d'erreurs par mot et par bit selon le nombre de minima pris en compte, dans le cas d'une matrice irrégulière construite à l'aide des degrés fournis par le programme* `LdpcOpt` *[8]*

La figure 2 montre l'influence de λ sur les performances, pour une matrice irrégulière dont les coefficients ont été obtenu grâce au programme `LdpcOpt` [8]. La proportion du nombre de bit impliqués dans une parité (check distribution) ainsi que la proportion du nombre de parités impliquant un bit (variable distribution) sont représentées à droite de la figure. A un taux d'erreurs binaire de 10^{-4}, l'algorithme $2-$min apporte une perte de 0.17 dB et de 0.36 dB pour respectivement 5 et 50 itérations.

1. On peut remarquer que le cas $\lambda = 2$ différe de l'algorithme BP-based pour les bits $n \notin \mathcal{N}_2(m) = \{n_0, n_1\}$: le premier renvoie $Z_{mn_0}^{(i)} \oplus Z_{mn_1}^{(i)}$ et le second simplement $Z_{mn_0}^{(i)}$.

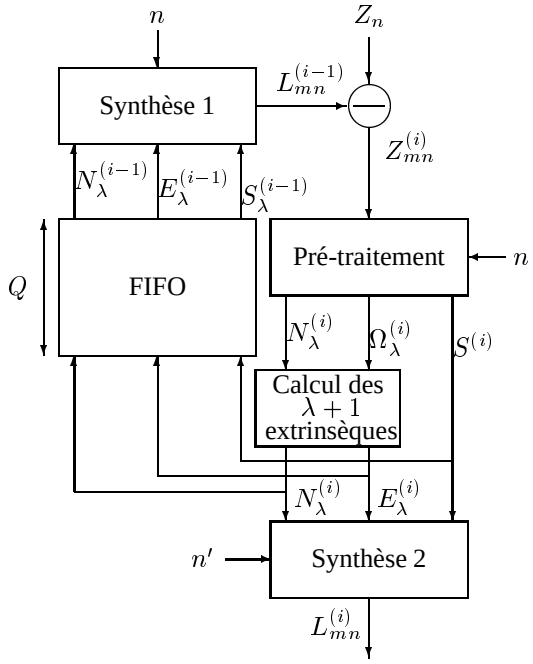

FIG. 3: *Description d'un processeur de parité*

Pour $\lambda \geq 4$, aucune perte n'est mesurable entre l'algorithme BP et le $\lambda-$min.

3 Architecture

Seule l'architecture de processeur de parité (PCP, pour Parity Check Processor) est décrite dans cet article. De nombreuses architectures déjà proposées peuvent être modifiées pour utiliser avantageusement le PCP proposé. On peut citer en particulier les architectures décrites dans [2] et [9]. Nous ferons juste l'hypothèse que P PCP travaillent en parallèle. Ainsi, le calcul des M parités d'une itération demande $Q = M/P$ macrocycles. Autrement dit, un PCP traitera Q parités à chaque itération.

3.1 Description du PCP

La simplification que nous avons apportée au calcul de la parité permet de réduire la complexité architecturale du PCP. Son synoptique est représenté sur la figure 3 et se divise en 5 étapes, exécutées en pipe-line :

1. Soustracteur : Génération des $Z_{mn}^{(i)}$ selon l'équation (2). Les $L_{mn}^{(i-1)}$ sont générés comme à l'étape 4 grâce aux informations sauvegardées lors de l'étape 5.
2. Pré-Processing : tri en série des λ plus petits modules $|Z_{mn}^{(i)}|$ et calcul du signe.
3. Calcul des $(\lambda+1)$ informations extrinsèques [2] de l'équation (9).
4. Synthèse : elle permet de générer les informations extrinsèques $L_{mn}^{(i)}$ à partir des données $\mathcal{N}_\lambda^{(i)}(m)$, $E_\lambda^{(i)}(m)$ et $S^{(i)}(m)$.

2. pour $\lambda > 2$; si $\lambda = 2$ il n'y a qu'un seul calcul à faire.

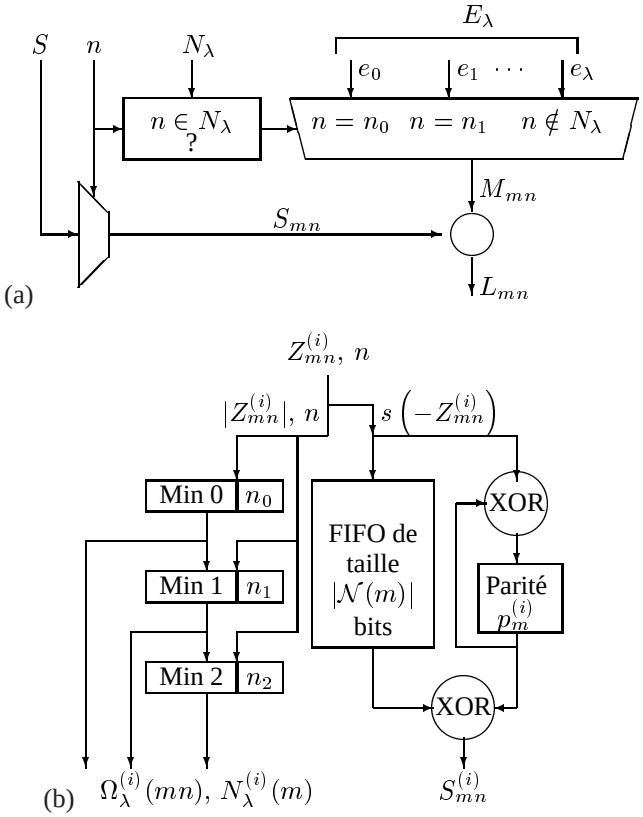

FIG. 4: *(a) : Le bloc synthèse. (b) : Le bloc Pré-traitement pour l'algorithme* $3-$*min.*

5. FIFO : mémorisation des données $\mathcal{N}_\lambda^{(i)}(m)$, $E_\lambda^{(i)}(m)$ et $S^{(i)}(m)$ dans la pile de taille Q pour pouvoir regénérer les $L_{mn}^{(i)}$ à l'itération $i+1$.

La figure 4-(a) décrit le bloc synthèse qui permet de générer les informations extrinsèques de tous les bits. Les index des bits sont comparés aux éléments de l'ensemble $\mathcal{N}_\lambda^{(i)}(m)$. Selon le résultat de la comparaison, le module de l'information extrinsèque ad hoc est choisi parmi les $\lambda+1$ valeurs de l'ensemble $E_\lambda^{(i)}(m)$, puis multiplié par le signe correpondant de l'ensemble $S^{(i)}(m)$.

Le bloc pré-traitement est représenté sur la figure 4-(b) (cas $\lambda = 3$). Il permet de faire un tri par insertion des nouvelles informations dans une succession de λ cellules "comparateur - mémoire" : à chaque coup d'horloge, la valeur courante de $|Z_{mn}^{(i)}|$ est comparée aux λ précédentes valeurs stockées ; selon le résultat de la comparaison, elle est insérée dans l'ordre décroissant et la valeur la plus grande est perdue. Ce bloc permet aussi simultanément de mémoriser les signes des informations dans une pile FIFO et de calculer la parité $p_m^{(i)}$.

Le chronogramme de la figure 5 décrit l'enchaînement des étapes du PCP lors du traitement du $q^{\text{ème}}$ macrocycle. Toutes ces étapes durent $|\mathcal{N}(m)|$ cycles, sauf celle du calcul qui peut se faire en $T_c = \lambda + 1$ cycles. La contrainte du fonctionnement en pipe-line est que le temps de calcul des informations extrinsèque ne doit pas dépasser celui des autres étapes, soit $|\mathcal{N}(m)|$ cycles. Pour décoder des codes irréguliers, il suffit de traiter les parités dans l'ordre croissant de leur poids. Si ce n'était pas le cas, il se pourrait que l'étape k d'un macrocycle q ne soit pas terminée au moment d'exécuter l'étape k du macrocycle $q+1$. Le pipe-line devrit alors être modifié.

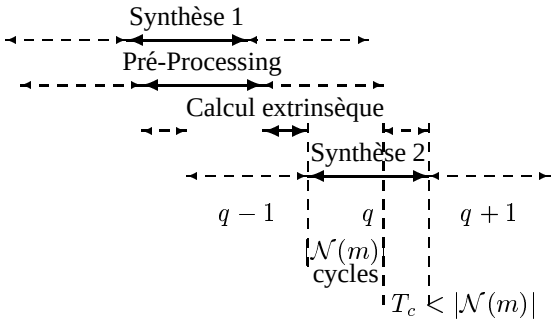

FIG. 5: *Séquencement des opérations d'un PCP*

3.2 Impact sur la mémoire

Pour cette partie, on note $N_b + 1$ le nombre de bits permettant de coder les données (N_b pour le module et 1 bit de signe). Lors du calcul exact de chaque parité, il est nécessaire de stocker $|\mathcal{N}(m)|$ informations extrinsèques $Z_{mn}^{(i)}$ qui sont codées sur $N_b + 1$ bits. Cela donne au total $(N_b + 1)|\mathcal{N}(m)|$ bits à sauvegarder.

La simplification du calcul de parité apportée par l'algorithme $\lambda-$min permet de réduire la quantité de mémoire interne des processeurs de parité et c'est le bloc synthèse de la figure 3 qui se charge de la décompression. Avec cette simplification , il sera nécessaire de stocker :

- $\lambda + 1$ résultats de l'équation (9) soit $(\lambda + 1)N_b$ bits.
- λ pointeurs sur λ éléments de l'ensemble $\mathcal{N}_\lambda(m)$ soit : $\lambda \log_2(|\mathcal{N}_\lambda(m)|)$ bits
- $|\mathcal{N}(m)|$ signes ainsi que le produit des signes (parité $p_m^{(i)}$), soit $|\mathcal{N}(m)| + 1$ bits.

Le rapport entre la quantité de mémoire nécessaire pour stocker les informations relatives aux informations extrinsèques dans le cas de l' algorithme $\lambda-$min et dans le cas de l'algorithme classique, pour une parité c_m est donc de :

$$\frac{(\lambda+1)N_b + \lambda \log_2(|\mathcal{N}_\lambda(m)|) + |\mathcal{N}(m)| + 1}{(N_b+1)|\mathcal{N}(m)|}. \qquad (10)$$

La figure 6 illustre ce rapport en fonction de λ et de $|\mathcal{N}(m)|$. On constate par exemple que pour une parité c_m de poids $|\mathcal{N}(m)| = 20$ et dans le cas de l'algorithme $2-$min, la mémoire nécessaire au stockage des données relatives aux information extrinsèques n'occupe que 30% de la mémoire qui serait nécessaire si l'on utilisait l'algorithme classique.

Conclusion

Dans cet article, un nouvel algorithme sous-optimal, l'algorithme $\lambda-$min, a été présenté. Il permet la simplification de la mise à jour des parités lors du décodage itératif des codes LDPC, sans dégradation significative des performances de décodage.

Une architecture de processeur de parité utilisant cet algorithme a été proposée. Elle permet une compression importante de la mémoire nécessaire au stockage des informations extrinsèques associée à chaque entrée non nulle de la matrice de parité du code.

L'utilisation d'un tel processeur de parité dans une architecture utilisant des bancs mémoires permet un parallèlisme plus

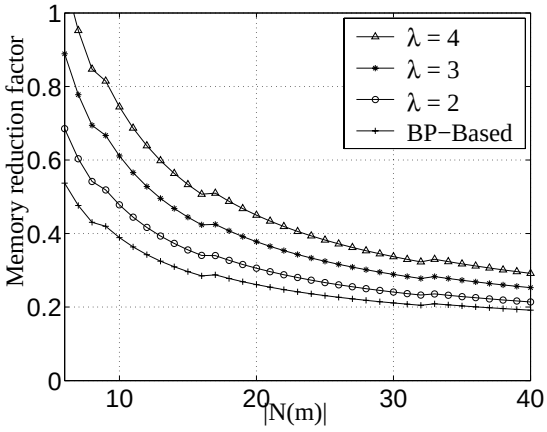

FIG. 6: *Gain mémoire obtenu en passant de l'algorithme BP à l'algorithme $\lambda-$min, avec $N_b = 5$.*

important qu'avec un processeur de parité classique ; c'est-à-dire que pour un même nombre de blocs mémoires, le nombre de parités traitées simultanément est plus important. Cette association permet également un décodage des codes réguliers et irréguliers, sans trop contraindre leur matrice de parité. Une telle architecture de décodeur est en cours de développement dans notre laboratoire.

Références

[1] G. Battail and A. H. M. El-Sherbini. Coding for radio channels. *Annales des Télécommunications*, 37:75–96, 1982.

[2] E. Boutillon, J. Castura, and F.R. Kschichang. Decoder-first code design. In *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, pages 459–462, Brest, France, 2000.

[3] J. Chen and M.P.C. Fossorier. Near optimum universal belief propagation based decoding of low-density parity-check codes. *Transactions on Communications*, 50:406–414, March 2002.

[4] J. Chen and M.P.C. Fossorier. Density evolution for two improved bp-based decoding algorithms of ldpc codes. *IEEE Communication Letters*, 6:208–210, May 2002.

[5] M.P.C. Fossorier, M. Mihaljević, and I. Imai. Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation. *Transactions on Communications*, 47:673–680, May 1999.

[6] R.G. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, MIT, 1963.

[7] D.J.C MacKay and R.M. Neal. Near shannon limit performance of low-density parity-check codes. *Electronic Letter*, 32:1645–1646, 1996.

[8] R. Urbanke. Ldpcopt. Available at http://lthcwww.epfl.ch/research/ldpcopt/.

[9] T. Zhang and K.K. Parhi. Vlsi implementation-oriented (3,k)-regular low-density parity-check codes. In *Worshop on Signal Processing Systems 2001, SIPS*, Sept. 2001.