Décodeur Radix-16 à entrées et sorties pondérées pour un turbo-décodage à haut débit

Oscar SÁNCHEZ[1], Christophe JÉGO[2], Michel JÉZÉQUEL[1]

[1]Institut MINES-TELECOM TELECOM Bretagne UMR, CNRS 6285 Lab-STICC
Technopôle Brest Iroise CS 83818, 29238 Brest Cedex 3

[2]IPB / ENSEIRB-MATMECA CNRS IMS, UMR 5218 351
Cours de la Libération 33405 Talence
Université de Bordeaux, France

oscar.sanchez-gonzalez@telecom-bretagne.eu, christophe.jego@enseirb-matmeca.fr
michel.jezequel@telecom-bretagne.eu

Résumé – Actuellement, le principal défi pour la mise en œuvre de turbo-décodeurs est d'atteindre des débits de données élevés comme exigés par les standards de communication actuels et futurs. Pour relever ce défi, une faible complexité du décodeur radix-16 à Entrées Pondérées et Sorties Pondérées (EPSP) pour l'algorithme Max-Log-MAP est proposé dans ce papier. Basé sur l'élimination des chemins parallèles dans le diagramme d'un treillis radix-16, des solutions architecturales pour réduire la complexité des différents blocs du décodeur EPSP sont proposées. Aussi, deux techniques complémentaires sont introduites afin de surmonter la dégradation qui apparaît lorsque des turbo décodeurs basés sur le décodeur EPSP proposé sont considérées. Ainsi, une pénalité inférieure à 0,05 dB est observée pour des turbocodes à 8 états par rapport à un turbo décodeur radix-2 traditionnel, pour 6 itérations.

Abstract – At present, the main challenge for hardware implementation of turbo decoders is to achieve the high data rates required by current and future digital communication system standards. In order to address this challenge, a low complexity radix-16 Soft-Input Soft-Output (SISO) decoder for the Max-Log- MAP algorithm is proposed in this paper. Based on the elimination of parallel paths in the radix-16 trellis diagram, architectural solutions to reduce the hardware complexity of the different blocks of a SISO decoder are detailed. Moreover, two complementary techniques are introduced in order to overcome BER/FER performance degradation when turbo decoders based on the proposed SISO decoder are considered. Thus, a penalty lower than 0.05dB is observed for a 8 state binary turbo code with respect to a traditional radix-2 turbo decoder for 6 decoding iterations.

1 Introduction

L'utilisation des turbocodes [1] dans les standards de communication sans fil, tels que 3GPP-LTE et WiMAX, impose de concevoir des architectures de turbo-décodage haut débit. Alors que le standard LTE-Advanced propose des débits aux environs de 1 Gbit/s, il n'est pas imaginable de concevoir des architectures où la complexité n'est pas maîtrisée. Le défi est donc de proposer une architecture de décodage capable de supporter l'augmentation du débit tout en maîtrisant la complexité calculatoire sans dégrader de manière significative les performances de correction.

Le décodage de turbocodes est réalisé par un processus itératif construit autour de deux décodeurs EPSP qui échangent une information appelée information "extrinsèque". Les algorithmes utilisés dans les décodeurs EPSP sont intrinsèquement récursifs, et par conséquent difficilement parallélisables. Pour s'affranchir de cette contrainte, une technique appelée "radix-2^{N_T}" a été proposée [2–4] afin de calculer N_T transitions dans le treillis associées aux codes convolutifs durant une période d'horloge.

La plupart des travaux dans la littérature qui proposent des architectures radix-2^{N_T} abordent principalement l'augmentation du débit, et seulement quelques-uns proposent des techniques pour réduire la complexité associée. Dans [3], une architecture radix-16 pour l'unité ACS (Add Compare Select) est proposée, où l'opération de comparaison entre 16 branches est simplifiée en deux niveaux. Ainsi, le surcoût matériel résultant du radix élevé est réduit. Dans [4] une architecture radix-16 pour l'algorithme Log-MAP exécuté en deux étapes est proposée. Dans notre travail, la principale contribution est la conception d'une architecture EPSP radix-16 qui implémente l'algorithme Max-Log-MAP basé sur une unité ACS radix-8. Nous pouvons ainsi réduire le chemin critique du décodeur EPSP, qui se trouve dans l'unité ACS, par rapport aux travaux précédents [3, 4]. La démarche prend aussi en compte la complexité matérielle de l'ensemble des unités composant le décodeur EPSP.

Ce document est organisé comme suit. Dans la section 2 l'architecture proposée pour le décodeur EPSP est présentée. Une comparaison de la performance entre différentes versions du turbo-décodeur est montrée dans la section 3. Dans la section 4, les résultats de l'implémentation matérielle sont données. Finalement, la section 5 conclut l'article.

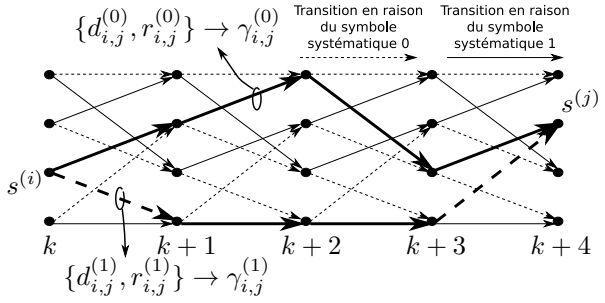

FIGURE 1 – Diagramme de transition pour un treillis associé à une unité ACS Radix-16.

2 Architecture du décodeur EPSP

Un décodeur EPSP reçoit l'information du canal - LLR systématique $L(d_k)$ et redondant $L(r_k)$ - et l'information *a priori* (L_k^a). Il est composé de trois blocs principaux de calcul : l'unité BMU (Branch Metric Unit), l'unité ACS et l'unité SOU (Soft Output Unit). L'unité BMU calcule la métrique de branche notée par γ, qui est utilisée par l'unité ACS pour effectuer des opérations récursives. L'unité SOU calcule l'information extrinsèque et prend les décisions de décodage.

Nous prenons ici comme cadre applicatif le turbocode utilisé dans le standard LTE (binaire 8 états). Soit $S = \{s^{(0)}, \ldots, s^{(7)}\}$ l'ensemble des états que peut prendre le codeur. Ces états sont représentés sous forme algébrique : $s^{(i)} \equiv \sum_{m=0}^{2} a_m^{(i)} \cdot 2^m$, avec $a_m^{(i)} \in \{0, 1\}$. La Figure 1 illustre la transition d'états $s^{(i)} \rightarrow s^{(j)}$ dans le treillis du code pour une unité ACS radix-16 ($N_T = 4$). Pour cette valeur de radix, deux chemins (chemins parallèles) existent à partir de n'importe quel état à l'instant k jusqu'à tout état au temps $k + 4$. Soit ces deux chemins correspondent à la séquence de bits systématiques $d_{i,j}^{(b)} = (d_k^{(i,j,b)}, d_{k+1}^{(i,j,b)}, d_{k+2}^{(i,j,b)}, d_{k+3}^{(i,j,b)})$ qui génère la séquence de bits redondants $r_{i,j}^{(b)} = (r_k^{(i,j,b)}, r_{k+1}^{(i,j,b)}, r_{k+2}^{(i,j,b)}, r_{k+3}^{(i,j,b)})$, avec $b \in \{0, 1\}$. Soit $\gamma_{i,j}^{(b)}$ la métrique de branche calculée par l'unité BMU radix-16 classique pour le chemin b. Comme indiqué dans [5], les chemins parallèles peuvent être éliminés avant d'être fournis à l'ACS. Ainsi, il est possible de remplacer une unité ACS radix-16 par une unité ACS radix-8 qui est moins complexe et présente un chemin critique plus court. Dans ce cas, chaque paire d'états $(s^{(i)}, s^{(j)})$, au temps k et $k + 4$ respectivement, est reliée seulement par un chemin. La métrique de branche pour ce chemin est donc $\gamma_{i,j}^{max} = \max(\gamma_{i,j}^{(0)}, \gamma_{i,j}^{(1)})$.

Nous pouvons exprimer les bits systématiques et redondants des deux chemins parallèles associés à la transition $s^{(i)} \rightarrow s^{(j)}$ comme donné dans (1). Il est à noter que pour chaque paire de chemins parallèles $d_{k+1}^{(i,j,0)} = d_{k+1}^{(i,j,1)}$ et $r_{k+2}^{(i,j,0)} = r_{k+2}^{(i,j,1)}$ puisqu'ils sont indépendants de b. Ainsi, les valeurs provenant du canal de transmission correspondant aux bits d_{k+1} et r_{k+2} n'affectent pas le choix de $\gamma_{i,j}^{max}$ entre $\gamma_{i,j}^{(0)}$ ou $\gamma_{i,j}^{(1)}$. Sur la base de ces observations, nous proposons des architectures pour les unités BMU, ACS et SOU. Les principales caractéristiques de ces unités sont décrites ci-dessous.

– **BMU** : En tenant compte uniquement des bits systématiques et redondants qui influencent le choix de γ^{max} (bits différents de d_{k+1} et r_{k+2}), l'architecture proposée effectue un calcul partiel qui permet de choisir le chemin qui a la métrique de branche la plus grande. Puis, γ^{max} est calculé en ajoutant les LLRs de d_{k+1} et r_{k+2}. L'architecture résultante (figure 2) ne coûte que 53% des ressources matérielles d'une mise en oeuvre classique.

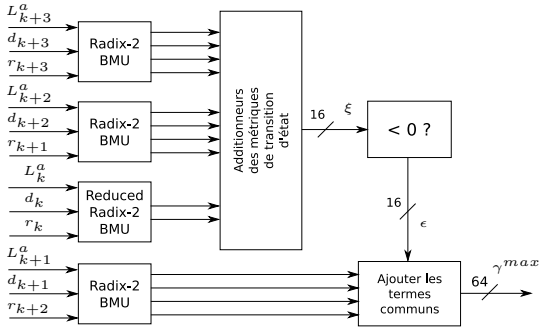

FIGURE 2 – Architecture de l'unité BMU radix-16.

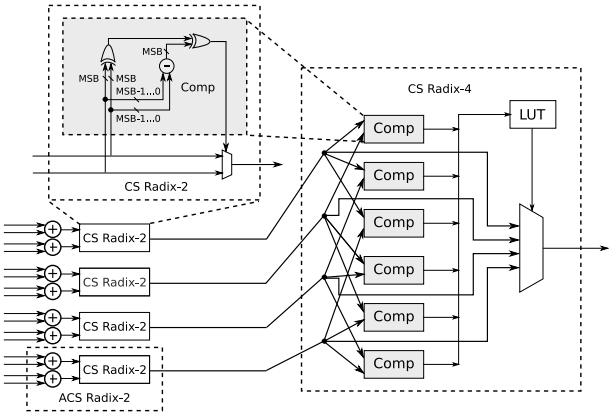

FIGURE 3 – Architecture de l'unite ACS radix-8 proposée.

– **ACS** : L'unité ACS proposée, illustré dans la figure 3, correspond à une architecture radix-8. Elle est composée de quatre unités ACS radix-2, et d'une unité Compare-Select (CS) radix-4. Par rapport à une architecture radix-2, le chemin critique est seulement augmenté du chemin critique de l'unité CS radix-4. Mais quatre transitions dans le treillis du code sont traitées à chaque cycle d'horloge (notons qu'une seule transition est traitée dans une architecture radix-2).

– **SOU** : Normalement l'unité SOU est conçue comme un arbre de comparaisons suivi de soustracteurs. Dans notre architecture, nous ne pouvons pas utiliser un arbre de comparaison statique car il n'est pas possible de savoir a priori quels sont les chemins qui seront éliminés dans chaque transition d'état dans le treillis du code. C'est pourquoi nous proposons l'arbre de comparaison montré dans la figure 4, où des multiplexeurs sont utilisés pour adapter les comparaisons en fonction des chemins retenus. Les entrées de cet arbre sont organisées en prenant en compte les propriétés déduites des équations (1).

$$
\begin{aligned}
d_{i,j}^{(b)} &= (d_k^{(i,j,b)}, d_{k+1}^{(i,j,b)}, d_{k+2}^{(i,j,b)}, d_{k+3}^{(i,j,b)}) = (a_0^{(i)} + a_1^{(i)} + b, a_1^{(i)} + a_2^{(i)} + a_0^{(j)}, a_2^{(i)} + a_1^{(j)} + b, a_0^{(j)} + a_2^{(j)} + b) \\
r_{i,j}^{(b)} &= (r_k^{(i,j,b)}, r_{k+1}^{(i,j,b)}, r_{k+2}^{(i,j,b)}, r_{k+3}^{(i,j,b)}) = (a_0^{(i)} + a_2^{(i)} + b, a_1^{(i)} + a_0^{(j)} + b, a_2^{(i)} + a_0^{(j)} + a_1^{(j)}, a_1^{(j)} + a_2^{(j)} + b)
\end{aligned} \tag{1}
$$

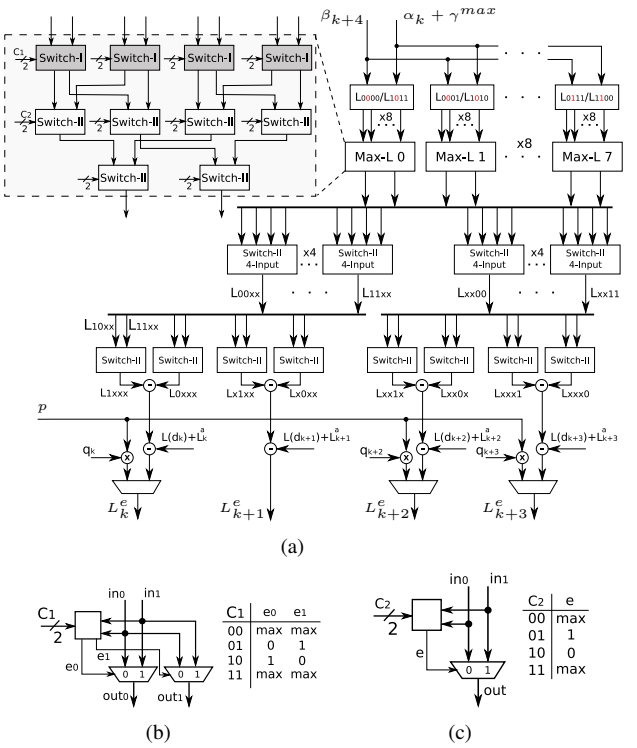

FIGURE 4 – (a) Unité SOU radix-16 proposée. (b) *Switch-I* circuit. (c) *Switch-II* circuit.

3 Performance du turbo-décodeur

Si le décodeur EPSP que nous proposons est utilisé dans un turbo-décodeur, les performances en termes de BER et FER sont significativement affectées. Nous avons observé une dégradation autour de 0.2dB, pour des rendements de codage R = 1/2 et 1/3, à un FER de 10^{-6}. De plus, un plancher d'erreurs élevé apparaît. Comme l'unité SOU ne tient pas compte de tous les chemins dans le treillis du code, les valeurs extrinsèques sont calculées approximativement. Ainsi, au cours du processus itératif les effets négatifs de ces approximations sont amplifiés. Pour surmonter ce problème, nous proposons deux techniques : 1) une valeur heuristique (p dans la figure 4(a)) qui est utilisée quand il n'est pas possible de calculer une valeur extrinsèque, à savoir quand tous les chemins qui sont nécessaires ont été éliminés, et 2) une technique visant à diminuer la corré-lation qui apparait entre les valeurs extrinsèques produites dans une transition radix-16 du treillis du code. Ces deux techniques sont décrites dans le reste de cette section.

3.1 Valeur du paramètre p

Nous avons établi une expression pour p suivant une approche similaire à celle présentée dans [6] pour le turbocodes produits. Dans [6], quand il n'y a pas un mot de code concurrent pour

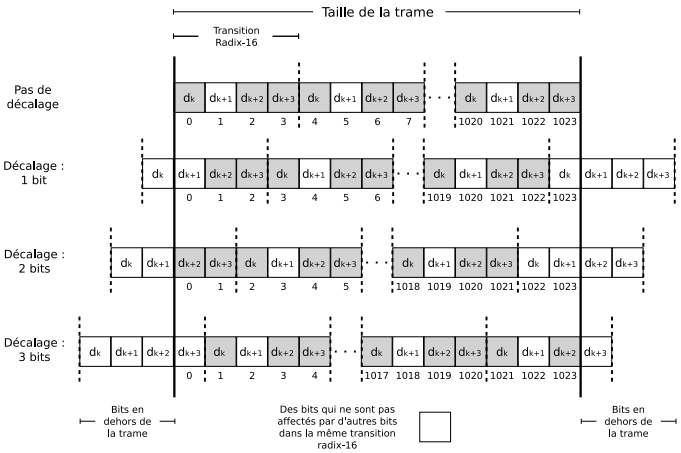

FIGURE 5 – Décalage de la trame reçue par le décodeur EPSP afin de réduire les interférences dans une transition radix-16.

calculer la fiabilité d'un bit, la sortie souple est calculée comme la somme de la magnitude d'un ensemble de valeurs LLR du canal (entrée du décodeur). Dans notre cas, comme il n'y a que $N_T = 4$ bits systématiques pour chaque transition radix-16, nous avons remplacé la somme par une opération minimale. Ainsi, nous évitons des valeurs extrinsèques trop optimistes. La valeur p est donc donnée dans (2). Il correspond à la valeur de fiabilité minimale à l'entrée du décodeur. Des simulations Monte-Carlo ont montré la pertinence de cette expression.

$$
p = \min_{i=0,1,2,3} \left(\left| L_{k+i}^{a} + L(d_{k+i}) \right| \right) \tag{2}
$$

3.2 Corrélation entre les valeurs extrinsèques dans une transition radix-16

Grâce à la structure du code que nous considérons, le deuxième bit de la transition radix-16 (d_{k+1}) n'est pas affecté, *i. e,* le bit est protégé, lors de l'élimination des chemins parallèles. En profitant de cette observation, nous proposons une technique de décalage sur la trame traitée par le décodeur EPSP, comme illustré dans la figure 5 pour une taille de trame de 1024 bits. Dans ce cas, quand il n'y a pas de décalage, les bits $1, 5, 9, \cdots, 1021$ sont protégés. Avec un décalage de un bit, les bits $0, 4, 8, \cdots, 1023$ sont protégés. Avec un décalage de deux ou trois bits, les bits protégés sont différents. Ainsi, si nous changeons la valeur de décalage dans des itérations consécutives du turbo-décodeur, les effets de notre architecture simplifiée peuvent être significativement réduits. Il est à noter que la technique de décalage nécessite une initialisation appropriée des métriques d'état au début et à la fin de la trame décalée (figure 5).

Les performances pour le turbocode du standard LTE avec 6 itérations et une taille de trame de 1024 bits sont données dans la figure 6. Deux architectures ont été considérées : une basée sur un décodeur EPSP radix-2, et l'autre basée sur notre décodeur

Unité	Radix-2	Radix-4	Radix-16 (unité ACS radix-16)	Radix-16 (proposé)
BMU	635	3,2k	28,6k	15,3k
ACS (Surface pour les 8 états)	3,4k ([7], 10 bits)	8,2k ([2], 11 bits)	34,1k ([3], 12 bits)	16,6k (12 bits)
SOU	2,2k	5,3k	23,5k	18,3k
Décodeur EPSP	10,5k	28,1k	148,9k	82,1k

TABLE 1 – Complexité matérielle exprimée en équivalent porte logique pour différents décodeurs EPSP à une fréquence de 200MHz. (Buffers d'entrée et β non considérés.)

EPSP radix-16. Deux rendements de codage $R = 1/2, 1/3$ ont été pris en compte. 6 et 9 bits ont été choisis pour représenter l'information du canal et l'information extrinsèque respectivement. Pour l'architecture radix-2, 10 bits sont nécessaires pour les métriques d'état. Pour l'architecture radix-16 12 bits sont utilisés. Nous pouvons observer comment, grâce aux deux techniques, il est possible de limiter la dégradation à 0,05dB par rapport au turbo-décodeur radix-2, *i. e*, un turbo-décodeur qui présente une performance idéale pour l'algorithme Max-Log-MAP en virgule fixe.

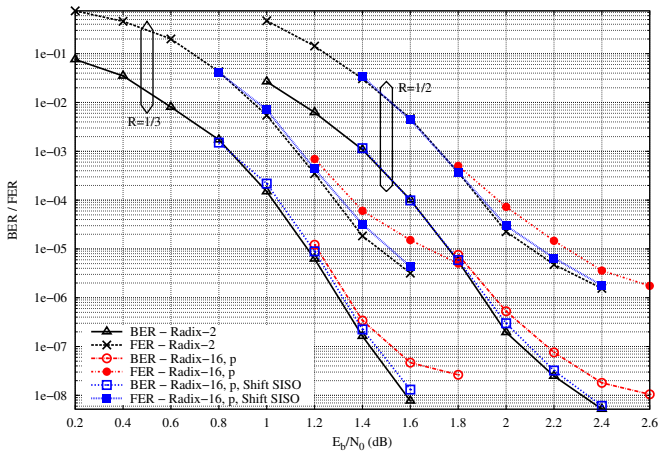

FIGURE 6 – Simulation du turbo-décodeur en virgule fixe (1024 bits par trame). Architectures radix-2 et radix-16.

4 Résultats d'implémentation

Le tableau 1 récapitule les résultats de synthèse logique obtenus pour une cible ASIC en technologie 90nm de chez STMicroelectronics. Des architectures radix-2, radix-4 et radix-16 de la littérature sont aussi considérées. Une fréquence de 200MHz a été ciblée. Dans ce tableau la complexité des architectures EPSP est exprimée en équivalent porte logique. L'architecture d'un décodeur EPSP comprend 2 unités BRU, 2 unités ACS et 1 unité SOU.

Nous pouvons observer que le décodeur EPSP proposé est 0,55 fois moins complexe que le radix-16 classique. Par rapport à une architecture radix-2, il est donc possible, pour une fréquence d'horloge donnée, de multiplier le débit utile par 4 au prix d'une augmentation de la complexité dans un rapport légèrement inférieur à 8. Ce résultat pourrait paraître décevant mais il faut le replacer dans le contexte du turbo-décodage où l'utilisation d'un radix-16 permet d'éviter une duplication de la totalité du turbo-décodeur.

Ainsi, dans le cas d'un turbo-décodeur utilisant 32 décodeurs EPSP et pour une taille de trame de 1024 bits, la complexité totale est multipliée par un facteur 3,1 alors que le débit est multiplié par 4.

5 Conclusion

Nous avons proposé une architecture permettant de simplifier un EPSP à base de radix-16. Cette nouvelle architecture génère des dégradations du taux d'erreurs. Nous avons proposé deux techniques permettant de les compenser. L'architecture proposée a démontrée être compétitive par rapport aux architectures dans l'état de l'art.

Références

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding : turbo-codes," in *IEEE ICC '93, Geneva*, 1993, pp. 1064 – 1070.

[2] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 8 –17, jan. 2011.

[3] C.-H. Tang, C.-C. Wong, C.-L. Chen, C.-C. Lin, and H.-C. Chang, "A 952MS/s Max-Log MAP decoder chip using radix-4 x 4 ACS architecture," in *Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian*, nov. 2006, pp. 79 –82.

[4] K.-T. Shr, Y.-C. Chang, C.-Y. Lin, and Y.-H. Huang, "A 6.6pj/bit/iter radix-16 modified log-MAP decoder using two-stage ACS architecture," in *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian*, nov. 2011, pp. 313 – 316.

[5] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation : breaking the ACS-bottleneck," *Communications, IEEE Transactions on*, vol. 37, no. 8, pp. 785 –790, aug 1989.

[6] P. Adde and R. Pyndiah, "Recent simplifications and improvements in Block Turbo Codes," in *2nd International Symposium on Turbo Codes & Related Topics, September 4-7, Brest, France*, 2000, pp. 133 – 136.

[7] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 98 –106, jan. 2009.