Architectures auto-reconfigurables pour le contrôle dynamique du compromis QoS/SNR/débit de turbo décodeurs de code produit

Christophe JEGO* et Jean-Philippe DIGUET**

*GET / ENST Bretagne, CNRS TAMCIC UMR 2872, Brest, France
christophe.jego@enst-bretagne.fr

** LESTER / Université de Bretagne Sud, CNRS FRE 2734, Lorient, France
jean-philippe.diguet@univ-ubs.fr

Résumé : Les systèmes sur silicium reconfigurables constituent une voie prometteuse pour atteindre le compromis recherché dans le domaine des systèmes embarqués entre une nécessaire flexibilité et une impérieuse efficacité. La flexibilité est requise d'une part pour des raisons de coût de conception en permettant de modifier une architecture en fonction de l'évolution ou de l'hétérogénéité des standards. D'autre part, la reconfiguration permet d'ajuster dynamiquement l'adéquation entre une architecture et les conditions de fonctionnement afin d'optimiser l'efficacité calculatoire et énergétique. Cet article traite de la conception d'architectures adaptatives pour le turbo décodage de code produit dont le comportement est par nature dépendant des données. Le concept est développé sur un FPGA Xilinx offrant une réalisation accessible de la reconfiguration dynamique. Il permet de décider de la configuration matérielle adaptée à l'état du canal de transmission afin de favoriser un compromis Performance / QoS/ Consommation.

I- Introduction sur les technologies reconfigurables

Différentes solutions ont été imaginées pour introduire de la reconfiguration au sein de systèmes sur silicium (SOC). Ceux-ci peuvent être classés en trois catégories. La première et la plus flexible est l'utilisation de processeurs embarqués (ARM, Tensilica, …) qui sont adaptés à des applications orientées contrôle mais dont le surcoût en contrôle est prohibitif d'un point de vue énergétique. La seconde catégorie est celle constituée par les différentes familles de FPGA à grain fin (Altera, Xilinx, Lattice, Atmel). Ce type de composant est particulièrement efficace du point de vue des performances pour l'implantation de fonctions travaillant au niveau bit (machines d'états, cryptage, …), par contre le contrôle de reconfiguration (mémoire, routage) induit une consommation, notamment statique, importante. Enfin, la troisième catégorie est celle des architectures configurables dites à gros grain disposant d'un degré de reconfiguration au niveau mot. Les architectures telles que XPP de PACT [1], Moustique [2], DRP de NEC disposent de chemins de données et d'unités de traitements configurables adaptées au traitement du signal et de l'image. Afin de disposer d'un compromis entre ces différentes caractéristiques, des architectures hétérogènes ont été conçues notamment par des fabricants de FPGAs tels que Xilinx et Altera qui proposent différentes familles de circuits intégrant plus ou moins de blocs spécialisés (DSP, mémoire, transceiver, processeurs). Par ailleurs d'autres approches sont apparues, celles-ci reposent principalement sur des processeurs auxquels sont associés des architectures VLIW programmables telles que ISEF [3] Avispa-IM2 [2] ou des chemins de données configurables reposant sur des unités de traitement élémentaires Pico [4], Critical Blue [5], Chameleon [6] permettant d'obtenir des gains d'efficacité énergétiques réellement significatifs notamment dans les domaines des télécommunications et du multimédia.

Traditionnellement l'architecture de systèmes embarqués est décidée à l'aide de techniques de co-conception logicielle/matérielle utilisées notamment pour déterminer le meilleur compromis entre performances, surface et consommation. Prendre une décision hors ligne impose une stratégie au pire cas, cette approche se traduit par une surestimation importante des ressources lorsque les applications traitées possèdent une charge qui varie sensiblement

en fonction des données et de l'environnement du système. Ce cas de figure correspond à celui des mobiles multimédia. Une alternative à cet écueil est la reconfiguration dynamique qui permet de modifier une architecture en fonction des besoins de l'application. L'architecture de FPGA Virtex2Pro de Xilinx a rendu accessible (non sans effort) les techniques et outils permettant de mettre en œuvre la reconfiguration dynamique. Des preuves de concept ont été ainsi apportées dans des domaines d'applications nécessitant du parallélisme tels que le cryptage [7], la vidéo[8], les processeurs en bande de base [9] et les applications graphiques. Cependant les solutions existantes ne se focalisent pas sur la décision alors que celle-ci est décisive. Dans domaine du codage canal qui nous intéresse ici, nous notons que dans [9] la décision de reconfiguration est prise de manière grossière et implique une reconfiguration complète du FPGA ce qui n'est pas acceptable pour les traitements en bande de base. Par ailleurs les architectures hétérogènes et à gros grain sont a priori plus efficaces d'un point de vue énergétique. Elles permettent également une reconfiguration plus rapide mais elles sont utilisées dans une optique de type multi-modes où l'objectif est d'adapter l'architecture au standard choisi (par exemple WCDMA dans [10]). Notre approche, appliquée aux turbo décodeurs de code produit, vise à traiter de manière globale la question de la décision et l'auto-reconfiguration en s'appuyant sur des systèmes configurables Xilinx pour lesquels les techniques sont à présent connues. Bien qu'encore peu performants en consommation, ces systèmes présagent de nouvelles architectures reconfigurables auxquels est consacré par exemple le projet européen IST FET AETHER [11] qui implique en France le CEA/LIST et le CNRS (LESTER/I3S).

II- Etat de l'art sur les architectures de turbo décodeurs de code produit

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 [12] : 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 une solution de décodage qui fournit de bonnes performances tout en nécessitant un faible niveau de complexité. Actuellement, ces codes sont considérés comme étant parmi les codes les plus efficaces pour le codage canal (gain de 2 à 4 dB par rapport aux codes correcteurs d'erreurs classiques). En 1994, R. Pyndiah [13] propose les turbo codes en blocs (TCB) comme une solution alternative aux turbo codes convolutifs (TCC).

Le schéma de codage repose sur une structure de code produit, défini par Elias [14], comme étant une concaténation de deux ou plusieurs codes en blocs linéaires. Ainsi, les codes produits sont une combinaison des propriétés des codes qui les composent. Si nous considérons par exemple deux codes BCH (n, k, δ) identiques, alors les paramètres du code produit résultant sont donnés par: n^2 (longueur de code), k^2 (nombre de symboles d'information), d^2 (distance minimale de Hamming). En 1972, Chase [15] a proposé un algorithme permettant d'approcher la séquence optimale décodée d'un code produit. Cet algorithme a une complexité raisonnable et introduit une faible dégradation des performances. En 1994, Pyndiah [13] a complété cet algorithme pour permettre l'estimation de décisions pondérées associées aux symboles de la séquence décodée. En effet, l'algorithme de décodage itératif, qui réalise le turbo décodage de code produit, est une mise en cascade de décodeurs élémentaires EPSP (Entrée Pondérée Sortie Pondérée) traitant successivement les lignes et les colonnes. De plus, une reconstruction de la matrice des symboles est nécessaire entre les décodages lignes et les décodages colonnes. De nombreuses études et réalisations ont été effectuées à l'ENST de Bretagne : elles concernent des codes produits construits à partir de codes BCH corrigeant 1, 2, 3 et 5 erreurs [16] [17].

L'intégration d'un turbo décodeur de code produit peut être effectuée selon deux approches, la première est une technique séquentielle (structure où le décodeur élémentaire et le plan mémoire sont utilisés pour la totalité des demi-itérations) et la seconde technique est pipeline (structure où un décodeur élémentaire et un plan mémoire sont utilisés pour chaque demi-itération). Dans les architectures séquentielles de turbo décodeur, le circuit réalise l'ensemble des demi-itérations à partir d'un décodeur élémentaire et d'un plan mémoire. 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 n^2 symboles indépendamment du nombre d'itérations effectuées. Deux des quatre mémoires fonctionnent en mode lecture, les deux autres fonctionnent en mode écriture. Il y a une inversion des

modes de fonctionnement (lecture/écriture) de deux mémoires entre chaque demi-itération. Pour les deux dernières mémoires, 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 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. 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 n^2 symboles. Le fonctionnement des différentes mémoires est similaire au fonctionnement d'une architecture séquentielle. 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. En effet, 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 2*it* décodeurs élémentaires et *8it* mémoires.

III- Proposition d'une architecture reconfigurable dynamiquement pour les turbo décodage de code produit

L'émergence des technologies reconfigurables offrant à la fois les performances d'un ASIC et une souplesse d'utilisation, permet d'envisager de nouvelles solutions architecturales. Dans ce cadre, nous proposons une architecture reconfigurable de type pipeline pour l'intégration d'un turbo décodeur. Cette architecture repose sur une structure modulaire évolutive en fonction du contexte d'utilisation. En effet, le décodage itératif de code produit est une approche où une succession de décodage est appliquée à un même bloc de manière à converger vers un résultat probant. Au cours des itérations, une information dite extrinsèque qui correspond à l'apport de chaque décodage est échangée. Or, suivant le contexte où se trouve le turbo décodeur à un instant donné, le nombre d'itérations nécessaire à son bon fonctionnement évolue. Par exemple, le taux d'erreur binaire évolue en fonction du nombre d'itérations pour un rapport signal à bruit fixé comme le montre la Figure 1. Dans cet exemple, le turbo décodeur est dédié au code produit $(32,26,4)^2$.

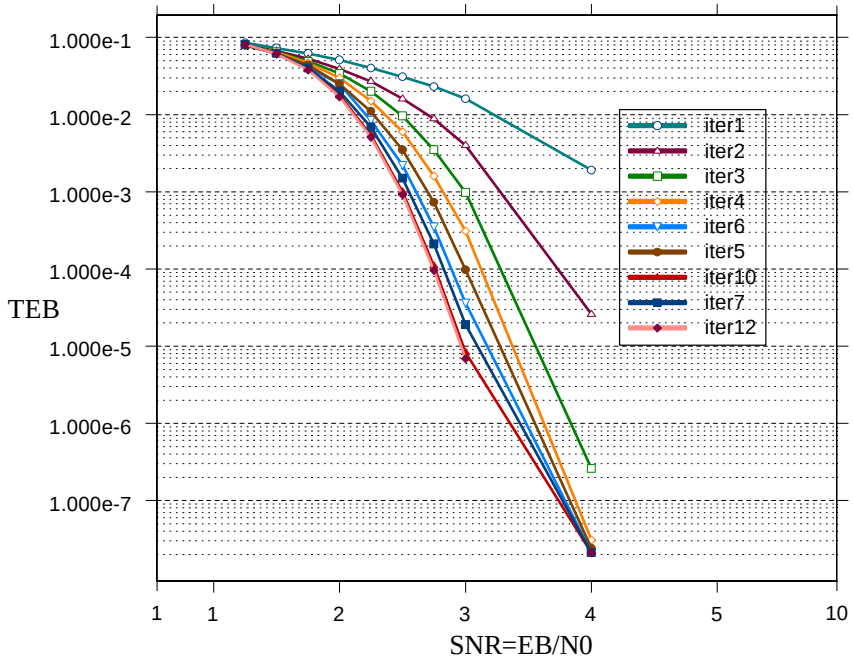

Figure 1: évolution des performances en terme de TEB pour un SNR fixé au cours des itérations

La complexité de la structure pipeline est directement fonction du nombre d'itérations à effectuer comme nous l'avons indiqué précédemment. C'est pourquoi, nous proposons une architecture reconfigurable qui adapte le nombre de blocs de base au nombre d'itérations comme présentée sur la Figure 2

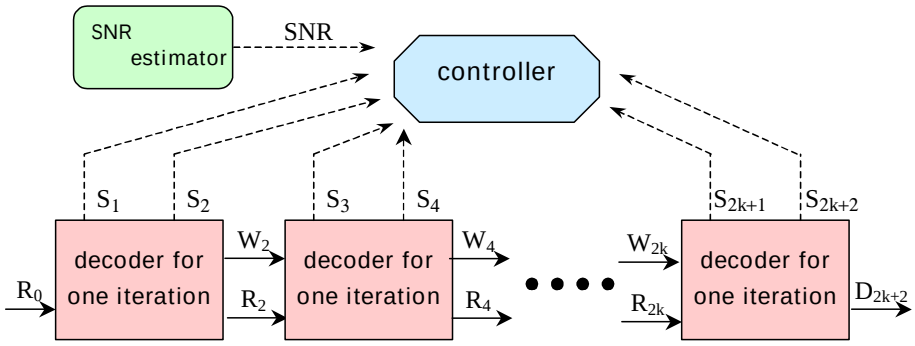

Figure 2 : structure pipeline modulaire du turbo décodeur

Un module de contrôle a alors pour fonction de déterminer pour chaque matrice d'information réceptionnée, le nombre d'itérations adéquat en fonction du rapport signal à bruit et du taux d'erreur cible. L'objectif de l'architecture étant de travailler à un débit fixe qui est celui du bloc de base. Cette approche implique donc d'associer au turbo décodeur un bloc d'estimation de bruit dont la fonction est d'évaluer le rapport signal à bruit de la matrice d'information au cours de sa réception [18]. La solution architecturale comprend donc autant de configurations que le nombre d'itérations maximum. Le module de base quant à lui est dédié à une itération, il est constitué de deux plans mémoires et deux décodeurs élémentaires à entrées et à sorties pondérées (Figure 3). La matrice est donc décodée successivement suivant les lignes puis suivant les colonnes. De plus, une sortie a été ajoutée au décodeur élémentaire. Elle indique la valeur du syndrome pour chaque ligne ou chaque colonne de la matrice. Cette information est envoyée vers le contrôleur et sert au critère d'arrêt que nous avons ajouté à notre architecture pipeline modulaire de turbo décodage. Le critère d'arrêt peut être énoncé de la manière suivante : *« si les syndromes pour chacune des colonnes après décodage suivant les lignes et les syndromes pour chacune des lignes après un décodage suivant les colonnes sont nuls au cours d'une même itération alors la matrice est composée de mots de code sur ses deux dimensions »*. Ainsi un schéma à deux niveaux est implanté pour la décision de configuration, il est le suivant : pour un débit et un SNR donnés, un nombre d'itérations et donc une configuration peuvent être décidés. Ensuite si le critère d'arrêt indique que la convergence a été atteinte avant le nombre d'itérations prévu alors le calcul est arrêté.

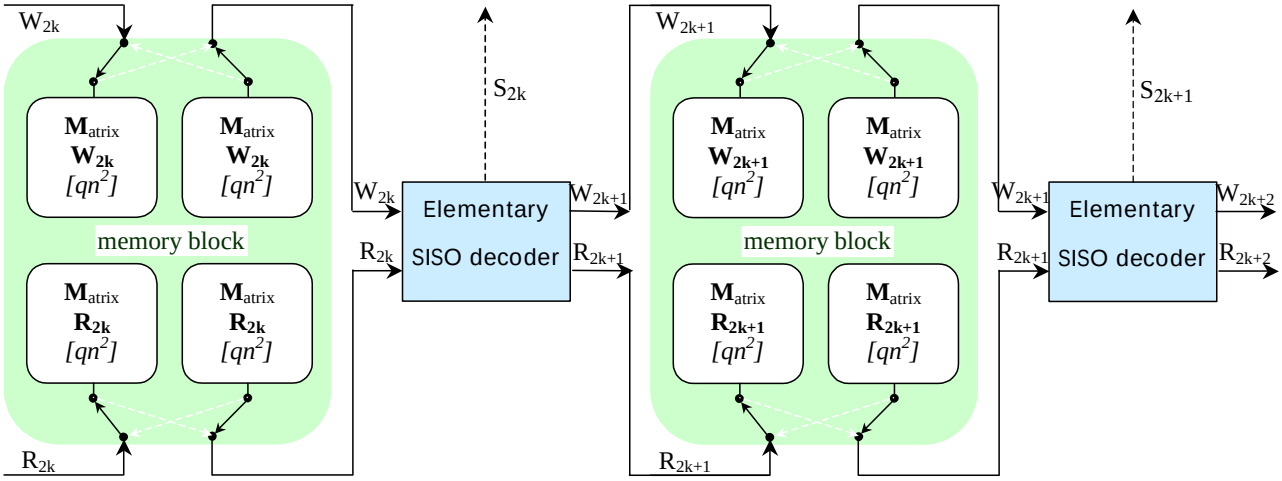

Figure 3 : module de base dédiée à une itération

IV- Vers l'intégration de la solution architecturale reconfigurable dynamiquement dédié à un turbo décodeur de code produit $(32,26,4)^2$

Le turbo décodeur est obtenu par une mise en cascade de module de base comme indiqué précédemment. Dans cette étude, nous avons retenu un décodeur élémentaire BCH (32,26,4) corrigeant au plus une erreur dans un mot de 32 symboles. Le principe du processus itératif (turbo décodage) du code produit consiste à traiter les 32 lignes de la matrice 32*32, à reconstituer la matrice et à décoder les 32 colonnes. Une itération de décodage est donc la somme de deux demi-itérations (lignes et colonnes). Les résultats de simulation de la Figure 1 montrent que le gain de codage est obtenu lors des huit premières itérations du processus. C'est pourquoi, nous ne considérons au maximum que huit itérations. De plus, l'architecture proposée nécessite un bloc qui estime le SNR de la trame au cours de sa réception. Dès lors, le contrôleur détermine en temps réel le nombre de modules de base nécessaires à partir d'une table de correspondance entre le SNR et le nombre d'itérations. Cette table est issue des résultats préalables de simulation. Elle est détaillée dans le Tableau 1. Le contrôleur élémentaire pilotant la reconfiguration dynamique est intégré dans un composant ICAP d'un circuit Virtex2Pro de Xilinx

SNR	Nombre d'itérations	Taux d'erreurs binaire
< 1 dB	1	$9.5\ 10^{-2}$
1.25 dB	2	$8\ 10^{-2}$
1.5 dB	5	$6.2\ 10^{-2}$
1.75 dB	6	$4.1\ 10^{-2}$
2 dB	7	$2\ 10^{-2}$
[2.25 dB, 3.25dB]	8	$[6.2\ 10^{-3}, 9.4\ 10^{-7}]$
3.5 dB	7	$2\ 10^{-7}$
3.75 dB	6	$4.8\ 10^{-8}$
4 dB	5	NS
4.25 dB	2	NS
4.5 dB <	1	NS

Tableau 1 : table de correspondance entre SNR et BER pour un turbo décodeur $(32,26,4)^2$

Les différentes versions plus ou moins complexes en nombre de module de base du turbo décodeur ont été réalisées et testées au sein du laboratoire TAMCIC. La complexité du module de base sur une cible Virtex2Pro est de 260 slices et de 4 blocs de 128 bits pour la partie traitement. Il faut ajouter la complexité pour les deux plans mémoires qui est de 8 blocs de 4096 bits. La version actuelle ne comprend pas l'estimateur de bruit. Nous considérons que cette information est disponible au niveau du système de réception. En effet, l'estimation du rapport signal à bruit est une fonction présente dans de nombreux récepteurs de communications numériques. Par exemple, elle intervient dans les modulations adaptatives, le contrôle de puissance et l'adaptation des rendements de code [19]. Il est néanmoins possible de développer un module d'estimation propre au turbo décodeur. Cette étude va faire l'objet de travaux au laboratoire TAMCIC.

La reconfiguration dynamique du FPGA fera appel à des techniques connues de contrôle des FPGA Xilinx Virtex2Pro déjà éprouvées au sein du laboratoire LESTER où différentes expériences ont été menées autour de la reconfiguration dynamique d'un modulateur GSM/GPRS [20], d'un architecture reconfigurable dynamiquement pour le cryptage AES/RC6 suivant différents compromis débit/énergie [21] et l'auto-configuration d'un Virtex2Pro [22] pilotée par le PowerPC intégré. En réalité la principale difficulté du travail restant à réaliser relève moins de la conception architecturale qui est achevée que de l'implantation de celle-ci à l'aide d'outils de CAO encore peu stables et peu documentés. Concrètement la reconfiguration est simplifiée par la structure en pipeline de l'architecture. Ajouter un étage n consiste à reconfigurer un module supplémentaire pendant que le système continue de fonctionner avec n-1 étages, lorsque la configuration est achevée la sortie de l'étage n-1 est commutée vers l'entrée de l'étage n. Inversement lorsqu'un étage est supprimé, le résultat est issu de la mémoire de sortie de l'étage n-1 et l'étage n devenu inutile est effacé.

Conclusion

Nous avons présenté dans cet article une solution permettant de tirer profit des architectures dynamiquement reconfigurables pour adapter en ligne l'implantation d'un turbo décodeur de code produit. À l'issue d'une étude théorique qui a montré comment le nombre d'itérations nécessaire, pour atteindre une contrainte de BER donné, varie en fonction du SNR nous avons implanté une méthode peu coûteuse pour décider la convergence et donc de la configuration adéquate. La solution architecturale retenue étant pipeline, celle-ci est parfaitement adaptée à une implantation efficace au sein d'une architecture reconfigurable. Le choix du FPGA est motivé par le prototypage rapide d'une architecture reconfigurable et à pour objectif de démontrer la faisabilité d'un système adaptatif tenant compte en temps réel des caractéristiques mouvantes d'une application embarquée, en l'occurrence il s'agit d'un turbo-décodeur traitant de données par essence variablement bruitées.

Références

[1] XPP PACT, http://www.pactcorp.com
[2] HIVE, http://www.siliconhive.com
[3] Stretch, http://www.stretchinc.com
[4] PICOChip, http://www.picochip.com
[5] CriticalBlue, http://www.criticalblue.com
[6] Paul M. Heysters, *"Coarse-Grained Reconfigurable Processors– Flexibility meets Efficiency"*, PhD thesis, University of Twente, the Netherlands, September 2004.
[7] H. Walder and M. Platzner, "Reconfigurable Hardware Operating Systems: From Design Concepts to Realizations", ERSA, Las Vegas, USA, 2003.
[8] J-Y.Mignolet and al. "Infrastructure for Design and Management of Relocatable Tasks in a Heterogeneous Reconfigurable System-on-Chip", DATE 2003, Munich, Germany, 2003.
[9] J. Liang, R. Tessier, D. Goeckel, "*A dynamically reconfigurable, power-efficient turbo decoder*", in the Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, April 2004.
[10] G. K. Rauwerda and G. J. M. Smith, "*Implementation of a Flexible RAKE Receiver in Heterogeneous Reconfigurable Hardware*" IEEE Int. Conf. on Field-Programmable Technology, Australia, Dec. 2004.
[11] AETHER, http://www.aether-ist.org/
[12] C Berrou, A. Glavieux, P. Thitimajshima, "*Near Shannon limit error correcting coding and decoding : Turbo Codes*", IEEE International Conference on Communication ICC93, vol. 2/3, May 1993.
[13] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq, "*Near optimum decoding of product codes*", GLOBECOM94, November 1994.
[14] P. Elias, "*Error-free coding*", IRE Trans. on Inf. Theory, vol. IT-4, pp. 29-37, Sept. 1954.
[15] D. Chase, "*A class of algorithms for decoding block codes with channel measurement information*", IEEE Trans. Inform. Theory, vol IT-18, pp 170-182, January 1972.
[16] S. Kerouedan, P. Adde, R. Pyndiah, "*How we implemented Block Turbo Codes*", annals of telecommunication, Vol. 56, N° 7-8, pp. 447-454, July-August 2001.
[17] J. Cuevas, P. Adde, S. Kerouédan, R. Pyndiah, "*New architecture for high data rate turbo decoding of product codes*", GLOBECOM 2002, Taipei, Taiwan, pp. 139-143, November 2002.
[18] Todd A; Summers, Stephen G. Wilson, "*SNR Mismatch and Online Estimation in Turbo Decoding*", IEEE Transactions on Communications, Vol. 46, N°4, April 1998.
[19] Chen Y; N.C Beaulieu, "*Maximum likelihood SNR estimators for digital receivers*, communications, computers and signal processing", 2005. PACRIM. 2005 IEEE Pacific Rim Conference on, pp 637-640, August 2005.
[20] J-Ph.Delahaye, G.Gogniat, C.Roland and P.Bomel, "*Software Radio and Dynamic Reconfiguration on a DSP/FPGA platform*", Frequenze, Journal of Telecommunications, pages 152-159, N°58, 5-6/2004.
[21] D.Dupé, J-Ph.Diguet, "*Architecture Adaptative pour le cryptage selon un compromis énergie/débit*", LESTER, juin 2005.
[22] B.Lecunff, E.Truffart, J-Ph. Diguet, G.Gogniat, "*Auto-Reconfiguration 2D d'un Virtex2Pro par le power PC*", LESTER, mars 2006.