FAST SOFT DECISION LDPC DECODER IMPLEMENTATION IN C

Type de contenu : Images animées

Titre(s) : FAST SOFT DECISION LDPC DECODER IMPLEMENTATION IN C ; BEINSCHOB, Patric ; LEMASSON, Jérôme ; SLT CAMUS, Antoine

Autre(s) responsabilité(s) : BEINSCHOB, Patric (Directeur de thèse)
LEMASSON, Jérôme (Directeur de thèse)
SLT CAMUS, Antoine (Secrétaire)

Editeur, producteur : Ecoles Militaires de Saint-Cyr Coëtquidan

Note de thèses et écrits académiques : Filière Scientifique - Option Electronique Promotion Chef d'Escadron Francoville Date de soutenance : 01/01/2011

Résumé ou extrait : Etude : PRESENTATION : La demande croissante de rapidité et de haut débit dans le domaine de la communication a poussé les recherches à se rapprocher au maximum de la limite de Shannon. Ainsi ont été redécouverts au milieu des années 1990 un nouveau type de codes correcteurs d'erreurs : les codes LDPC. Ceux-ci proposent des résultats très satisfaisant en termes de qualité de transmission d'information à travers des canaux relativement bruyants. Néanmoins, les résultats sont d'autant plus satisfaisants que les éléments utilisés sont complexes. Or si la complexité des calculs augmente, le temps de calcul augmente aussi et rend donc le système inexploitable. On peut distinguer deux types de décodages : le décodage à décision dure va utiliser la valeur des bits reçu avec les équations de parité pour retrouver les bits. Le décodage à décision douce va utiliser les probabilités des valeurs des bits (grâce au logarithme de rapport de vraisemblance) pour retrouver la valeur des bits. Le travail à effectuer est donc d'implémenter un décodeur en langage C qui réponde à des critères de qualité et de rapidité comparables à ceux du décodeur construit dans MATLAB. CONTRAINTES : La première des contraintes est de travailler sur C, il faut donc apprendre les notions de base du langage C. La deuxième contrainte est de créer un décodeur utilisant la décision douce, c'est-à-dire les logarithmes de rapport de vraisemblance. Enfin, la dernière contrainte concerne le temps d'exécution du décodeur. Le but est de construire un décodeur capable de décoder assez rapidement afin de ne pas gêner le débit du canal. On considèrera dans notre cas que le canal est le Canal à Bruit Gaussien Blanc Additif. Le décodeur doit recevoir en entrée : la matrice de parité du code, la valeur intrinsèque du Logarithme du Rapport de Vraisemblance (LRV), et le nombre maximum d'itération souhaité. Il doit retourner : la valeur a posteriori du LRV, le nombre d'itérations effectuées, si le mot décodé est correcte ou non. Plus tard, on voudra utiliser ce décodeur pour décoder des mots de codes par paquet. DEMARCHE : Premièrement, il s'agit de choisir l'algorithme de décodage que l'on va programmer. Plusieurs algorithmes de décodage ont été proposés faisant varier complexité et rapidité. Ces algorithmes se basent sur le principe d'itération. Le but ici est de trouver comment obtenir des performances correctes avec un temps d'exécution acceptable. Dans un second temps, l'algorithme choisit va être programmé sur MATLAB. Le but de cette opération est de tester la qualité de l'algorithme. En effet, sous MATLAB, on va pouvoir facilement comparer nos résultats avec ceux du décodeur de MATLAB (qui est une source fermée). En troisième partie, si l'algorithme programmé sur MATLAB convient, il s'agit désormais de transcrire l'implémentation en

langage C. L'interface MEX de MATLAB permet d'exécuter une implémentation en langage C sur MATLAB. Ainsi, on va pouvoir vérifier si des erreurs ont été commises lors de la traduction du code de MATLAB au langage C. Enfin, en évaluant le temps effectué par chaque opération, la dernière étape consiste à optimiser l'implémentation en termes de rapidité. RESULTATS OBTENUS : En ce qui concerne le choix de l'algorithme, nous avons choisi l'algorithme à propagation de croyance qui propose une qualité de décodage quasi optimale mais qui reste relativement complexe. L'implémentation de l'algorithme sous MATLAB a montré des résultats égaux à ceux que propose la source fermée. On peut donc affirmer que l'algorithme convient. Cependant, le temps effectué pour décoder un mot est resté trop important. L'étape suivante était de traduire le code en langage C. Une fois cela effectué, grâce au logiciel callgrind, nous avons pu obtenir le temps relatif mis pour chaque opération durant le décodage. Le but étant d'économiser du temps pour l'exécution du décodage, nous avons réduit au maximum les calculs. L'implémentation finale montre combien nous avons prog

Sujet(s) : MATLAB
algorithme
codage de données
communication de l'information
langage C