Modélisation de la Reconfiguration Dynamique appliquée à un décodeur LDPC Non Binaire

LAURA CONDE-CANENCIA[1], JEAN-CHRISTOPHE.PREVOTET[2], YASET OLIVA[2], YVAN EUSTACHE[1]

[1] Université Européenne de Bretagne – UBS, Lab-STICC, CNRS UMR 3192
Centre de Recherche - BP 92116, F-56321 Lorient Cedex - France
[2] Laboratoire IETR, Rennes, France

[1]laura.conde-canencia@univ-ubs.fr, yvan,.eustache@univ-ubs.fr
[2] jean-christophe.prevotet@insa-rennes.fr, yaset.oliva@insa-rennes.fr

Résumé – Ce document constitue le rapport final du projet Jeune Chercheur financé par le GdR-ISIS, ayant permis d'établir une collaboration entre les laboratoires Lab-STICC UBS et IETR. L'objectif du projet a été de modéliser les principes de la reconfiguration dynamique dans un récepteur adapté au standard WiMAX IEEE802.16m (réseau sans fil métropolitain). Le récepteur suit les principes de techniques ACM et inclut un décodeur LDPC non-binaire. L'étude réalisée a permis de s'adapter à des scénarios réalistes qui respectent le standard. Les contributions par rapport au document intermédiaire sont principalement dans la section 6. Le bilan financier du projet est donné à la fin du document (section 9).

1 Introduction

Les travaux du LabSTICC au tour de décodeurs LDPC ont permis de développer (d'abord en logiciel, puis sur FPGA) un décodeur LDPC non-binaire qui implémente une version simplifiée de l'algorithme de décodage sub-optimal Extended Min-Sum, nommée Bubble-Check [BC2010]. Ce décodeur est adapté a une famille de codes LDPC non-binaires dans le corps de Galois GF(q), avec q=64 [Po2008]. Dans cette famille, chaque code est défini par une matrice *ultra-sparse* qui est déterminée à partir des paramètres spécifiques du code : la taille de bloc (N) et le rendement du code (R). D'autre part, la sub-optimalité du décodeur est déterminée par le paramètre n_m (taille tronquée des messages échangés entre les nœuds de parité et les nœuds de variable, $n_m << q$). La version finale du décodeur est de type ad-hoc, c'est-à-dire qu'il sera valable pour un jeu de valeurs (N, R, n_m), et généralisable pour différentes valeurs exigées par une application concrète.

Or, dans les systèmes réels actuels, où les techniques de type ACM (Adaptive Coding and Modulation) sont de plus en plus utilisées, il devient très souhaitable que le récepteur puisse s'adapter selon les besoins du système aux différents scénarios et cela en temps réel. Sur des systèmes de référence tels que le LTE (4G) ou le WiMAX, la possibilité d'avoir un décodeur flexible pouvant alterner entre différents rendements et tailles de blocs constitue donc un enjeu majeur ouvrant de nouvelles perspectives.

L'objectif du projet est donc d'assurer la flexibilité du récepteur en étudiant la possibilité de reconfigurer dynamiquement certaines parties matérielles en fonction de l'état du canal. Pour ce faire, une étude amont doit être réalisée afin d'étudier la faisabilité d'une telle approche. La cible matérielle visée est un SoPC (System on Programmable Chip) faisant intervenir un processeur destiné au contrôle du décodeur LDPC et le décodeur lui-même réalisé à l'aide d'éléments logiques sur une zone reconfigurable. Un système d'exploitation est également prévu de manière à gérer la reconfiguration des éléments du décodeur au cours du temps en fonction de l'état du canal (plusieurs configurations du décodeur seront alors mises en œuvre en fonction du contexte).

L'approche proposée consiste à établir un modèle de l'ensemble de la plateforme (OS + application + architecture) dans un but de simulation et de validation de l'approche.

2 Principe et choix d'un scénario

Le principe de l'ACM consiste à choisir l'association modulation/code (MCS) permettant de garantir le débit plus élevé avec un niveau d'erreur acceptable. Concrètement, nous considérons 12 MCS possibles (Tableau 1) et un Taux d'Erreur Frame (TEF) seuil de 10e-2. L'efficacité spectrale (η) est donnée par $\eta=R.\log_2 M$, avec M le nombre de signaux de la modulation. Ainsi le MCS7 a $\eta = 3$ bit/s/Hz.

Tab 1 : Schémas de modulation/codage considérés

MCS	Modulation	Rendement de codage
{1, 2, 3, 4}	QPSK	{1/2, 2/3, ¾, 5/6}
{5, 6, 7, 8}	16-QAM	{1/2, 2/3, ¾, 5/6}
{9, 10, 11, 12}	64-QAM	{1/2, 2/3, ¾, 5/6}

Le système cible retenu a été celui du standard WiMAX IEEE802.16m (réseau sans fil métropolitain). En [D2.3.2], les auteurs présentent des résultats de simulation niveau système pour ce standard, obtenus avec une plateforme de simulation système (SLS) qui suit les directives des documents [TGm-EMD] et [TGm-SDD]. La Figure 1 montre l'efficacité spectrale maximale que l'on peut obtenir avec les différents MCS dans le cas du canal ITU-R piéton B à 3km/h. La Figure 2 donne les pourcentages d'utilisation des différents

MCS en fonction de la distance à la station de base dans le réseau. Ces informations vont permettre au récepteur de s'adapter en fonction de sa vitesse et de l'état du canal, comme suit : si la vitesse correspond à celle du canal ITU-R piéton B, une estimation du SNR du canal permettra de choisir le MCS le mieux adapté (Figure 1). Si la vitesse est plus importante, le système baserait son choix de MCS en fonction des informations de la Figure 2, faisant une estimation de distance à la station de base (principalement en fonction de sa vitesse et de sa trajectoire).

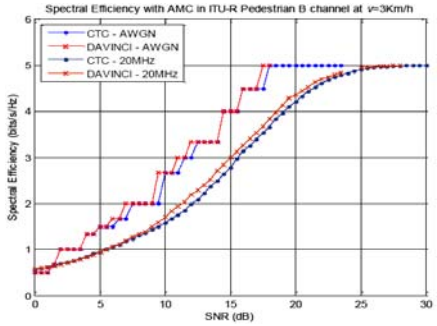

Figure 1 : efficacité spectrale maximale obtenue avec la technique ACM avec les MCS considérés

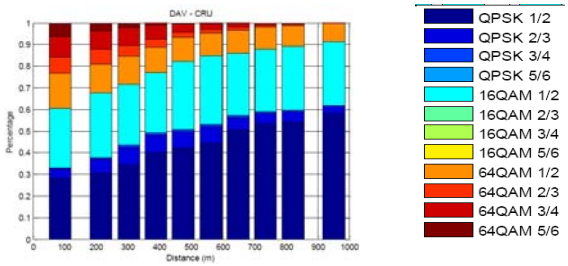

Figure 2 : pourcentage d'utilisation de chaque MCS en fonction de la distance à la station de base

3 Modélisation de la plateforme

L'ensemble de la plateforme a été modélisée en suivant la méthodologie définie dans OveRSoC. Cette méthodologie propose de réaliser un flot d'exploration aidant un concepteur de systèmes à explorer différents choix architecturaux lors du portage d'une application complète sur une architecture reconfigurable. OveRSoC fournit un modèle possédant un haut niveau d'abstraction et permet de simuler rapidement et efficacement des systèmes complexes. Le modèle consiste à spécifier la plateforme en renseignant un certain nombre d'attributs spécifiques. Ces attributs renseignent le modèle plus ou moins précisément et peuvent être raffinés au fur et à mesure du flot d'exploration.

La première étape du flot consiste à décrire l'application sous la forme d'un graphe de tâches au format .xml. La plateforme matérielle est ensuite spécifiée en précisant quels sont les supports d'exécution des tâches (processeurs, zones reconfigurables, mémoires, etc.). La seconde étape consiste à simuler en SystemC les deux précédents modèles afin de générer un ensemble de métriques destinées à aider le concepteur. Un environnement graphique basé sur JAVA permet de faciliter l'analyse de ces métriques afin de rendre son utilisation plus conviviale

4 Description de la plateforme

4.1 Modèle de l'application

L'application est décrite sous la forme d'un graphe de tâches précisant clairement leurs différentes connexions ainsi que leur précédence. A un premier niveau de description, les tâches sont spécifiées par des attributs simples tels que leur nature (logicielle ou matérielle), leur temps d'exécution sur un support précisé et le nombre de ressources logiques requises dans le cas où elles sont exécutées dans une zone reconfigurable. Un temps de configuration est alors spécifié dans un autre attribut. Une tâche peut également être mixte laissant la possibilité de l'exécuter aussi bien en logiciel qu'en matériel de manière à offrir plus de flexibilité. C'est alors à l'OS de décider de la nature de l'implantation en fonction des contraintes d'exécution.

Chaque tâche possède également d'autres attributs comme un nom, une priorité ainsi que deux listes précisant ses successeurs et prédécesseurs.

4.2 Modélisation de l'OS

Dans le modèle proposé, l'OS est relativement simple. Son rôle consiste à gérer et ordonnancer les différentes tâches de l'application. Actuellement, trois services principaux sont envisagés et modélisés.

1. Le service de gestion de tâches : Ce service permet au concepteur de créer ou détruire des tâches dynamiquement. Une primitive *create_task()* a été modélisée et se base sur la primitive *sc_spawn()* de SystemC. Cette primitive consiste à élaborer une structure C++ contenant toutes les données relatives à la tâche, ses descripteurs, nom, priorité. Une seconde primitive *kill_task()* permet de libérer toutes les ressources allouées à une tâche spécifique.
2. Le service d'ordonnancement : Ce service a pour objectif de séquencer les tâches en temps réel. Il consiste à choisir la future tâche à être exécutée en fonction d'une politique spécifique fournie par le concepteur. Cette décision est prise périodiquement lors d'une interruption de l'horloge système. A chaque interruption, le service choisit une tâche dans une liste de tâches qui sont dans l'état PRET. La préemption est également modélisée.
3. Le service de décision : Ce service consiste à décider si une tâche mixte doit être implantée en logiciel ou en matériel en fonction des contraintes spécifiques compatible avec un fonctionnement temps réel. Afin de prendre une décision efficace, ce service connaît le nombre de ressources logiques disponibles dans la zone reconfigurable. Dans un premier temps, ce service est modélisé très simplement et consiste à placer une tâche

matérielle avant une tâche logicielle si la place dans la zone reconfigurable est suffisante. Néanmoins, l'utilisateur peut également prévoir d'autres types de décision.

4.3 Modélisation de l'architecture

Le modèle d'architecture consiste à modéliser les éléments constitutifs de la plateforme à haut niveau d'abstraction. Pour le processeur, seule sa fréquence de fonctionnement est nécessaire de manière à évaluer le temps d'un cycle de traitement. D'autre part, la zone reconfigurable est vue comme un ensemble de N ressources logiques disponibles.

5 Cas d'étude

La méthodologie présentée a été appliquée à l'application présentée en 2). L'idée générale consiste à modéliser un récepteur mobile capable de recevoir des informations et de s'adapter aux changements du canal de manière à choisir la configuration avec l'efficacité spectrale plus élevée garantissant une qualité de transmission en termes de taux d'erreurs (TEF seuil de 10^{-2}). Aussi, le récepteur est capable de se reconfigurer en fonction de l'état du canal et de la distance à la station de base. A l'heure actuelle, les deux objets configurables sont le décodeur et le démodulateur.

Le scénario d'utilisation est le suivant : une tâche *capteur_vitesse* consiste à analyser périodiquement la vitesse du mobile. Cette tâche est exécutée en logiciel et envoie des informations à une seconde tâche *scrute_canal* identifiant les propriétés du canal. Cette dernière tâche est fondamentale dans la mesure où elle impose le fonctionnement du récepteur dans une configuration donnée. A titre d'exemple, si le canal est fortement bruité (faible SNR), le système devra sacrifier en efficacité spectrale pour garantir le TEF seuil, en choisissant le MCS adapté (à savoir QPSK R=1/2, η =1 bit/s/Hz, dans le pire cas)

Pour des raisons de performances, le décodeur et le démodulateur sont réalisés en matériel dans une zone reconfigurable de type FPGA. A chaque configuration, il convient donc de charger le bitstream dans le FPGA ce qui requiert un temps non négligeable. Le FPGA est dimensionné de telle sorte que deux décodeurs et deux démodulateurs puissent être implantés en parallèle dans le même circuit. Cependant, deux configurations ne sont jamais actives en même temps. De manière à s'affranchir du temps de configuration, dès lors qu'une configuration est active, l'autre a la possibilité de se charger indépendamment, dans une autre partie du circuit.

Lorsque la tâche *scrute_canal* est à nouveau exécutée, une nouvelle configuration est chargée dans le FPGA tandis que les informations reçues continuent à être traitées dans la configuration précédente.

Dans le scénario proposé, les informations reçues proviennent d'une image de 320x240 pixels codés sur 8 bits. A l'émetteur, cette image est découpée en trames de 192 symboles de 6 bits puis envoyée au récepteur qui devra réaliser des traitements pour effectuer un suivi de cible.

Pour des raisons de performances, les tâches relatives au traitement d'images sont toutes implantées en matériel à l'exception du calcul de centre de gravité, de l'estimation de mouvement et de l'affichage qui sont exécutées sur le processeur. Ces tâches ne sont pas configurables et sont uniquement vues comme des accélérateurs matériels.

La cible architecturale visée est un FPGA VIRTEX5 comportant un processeur de type Microblaze cadencé à 50 MHz. Le processeur exécutera le noyau mais également les fonctions logicielles.

5.1 Modélisation de l'application

Afin d'alimenter le modèle, toutes les tâches logicielles ou matérielles ont été testées puis les différents temps d'exécution ont été évalués sur la cible considérée. D'autre part, les temps de configuration du décodeur et du démodulateur ont également été déterminés.

Finalement un fichier de description de l'application en .xml a été décrit prenant en compte la synchronisation des tâches et la communication entre ces dernières. Le tableau 2 résume les informations relatives aux tâches.

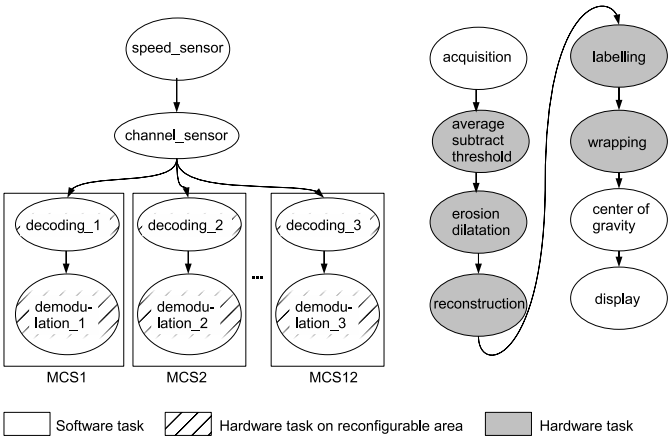

Figure 1. : Graphe des tâches de l'application

Tableau 1

Nom de la tâche	Type	Temps d'exécution (ms)
Capteur_vitesse	Logiciel	0.2
Scrute_canal	Logiciel	0.3
Decodeur	Matériel	0,172 à 0,355
Démodulateur	Matériel	0.003 à 0.012
Acquisition	Logiciel	0,5
Moyenne, soustraction, seuillage	Matériel	4,74
Erosion, dilatation	Matériel	0,421
Reconstruction	Matériel	9,53
Etiquetage	Matériel	7,52
Enveloppe	Matériel	8,48
Centre de gravité	logiciel	0,02
Affichage	logiciel	40

5.2 Modélisation du noyau

En partant des besoins de l'application, un simple noyau basé sur MicroC-OSII a été modélisé. Le noyau fournit les mécanismes de bases nécessaires à la gestion des tâches (création, destruction, etc.) ainsi que les mécanismes de synchronisation (mutex, semaphores, etc). Finalement, des files de messages sont également nécessaires afin de faire transiter les données d'une tâche à une autre. Dans un premier temps, le contenu des tâches est purement virtuel et consiste en une boucle infinie faisant appel à la primitive *wait()* de SystemC. Cette primitive permet de modéliser le temps d'exécution des tâches logicielles et matérielles. D'autre part, les temps d'exécution de chacun des services du noyau ont été mesurés sur la cible et renseignés dans le modèle.

5.3 Modélisation de la plate-forme matérielle

Le modèle de plateforme consiste en un processeur généraliste et une zone reconfigurable dynamiquement de type FPGA. Au niveau de description considéré, le processeur consiste simplement en une boîte noire tandis que la zone reconfigurable possède uniquement un nombre de cellules logiques capables d'accueillir les différents blocs matériels. Les attributs de cette plateforme ont été déterminés préalablement (fréquence du processeur, ressources mémoire, dimension des blocs en termes de ressources logiques).

6 Résultats obtenus

Afin d'illustrer l'utilisation de la méthodologie d'exploration, nous proposons de suivre deux scénarios de simulation pour l'exécution de la plate-forme. Le premier scénario consiste à spécifier les attributs qui conduisent à un mauvais fonctionnement du système complet. Après avoir analysé les mesures obtenues, l'utilisateur pourra modifier les attributs de la plateforme afin d'obtenir un résultat satisfaisant issu du simulateur. Dans le premier scénario (scénario 1), la taille de la zone reconfigurable est fixée de telle manière qu'elle correspond à la taille des blocs de décodage-démodulation. Il est également supposé que les conditions de canal sont connues et que le système commence avec une configuration par défaut : *decode1_HW* avec la démodulation *demod1_HW* (cf. Figure 2).

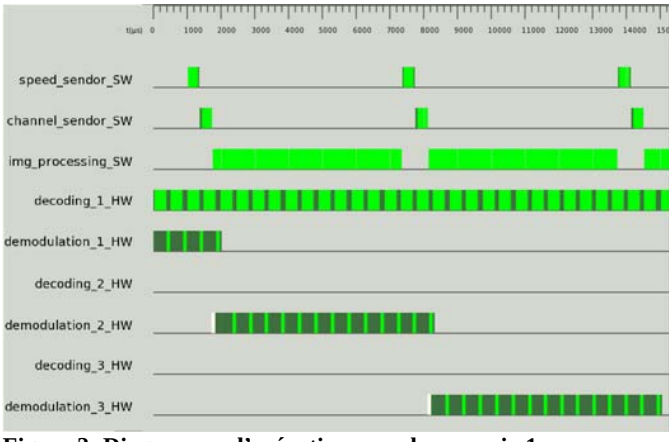

Figure 2: Diagramme d'exécution pour le scenario 1

Dans cette simulation, le noyau démarre en exécutant la tâche logicielle *speed_sensor*; l'exécution de tâche *channel_sensor* démarre et les nouvelles conditions du canal sont détectées. Avant de poursuivre le traitement d'image à l'aide la tâche *img_processing_SW*, le signal de la nouvelle configuration est validé. En suivant les indications du noyau, le service de reconfiguration tente de mettre en place les blocs de la nouvelle configuration avant d'arrêter les anciens, comme spécifié dans les conditions de l'application. A ce point, il est important de noter que le bloc de démodulation est le seul bloc qui est configuré (en raison d'un plus petit nombre de cellules logiques par rapport au bloc de décodage). Par conséquent, nous pouvons voir que decoding_1_HW continue de s'exécuter tandis que la démodulation est mise en œuvre par la tâche demodulation_2_HW ce qui conduit à un dysfonctionnement du système complet. Un problème identique est observé à 8000 ns.

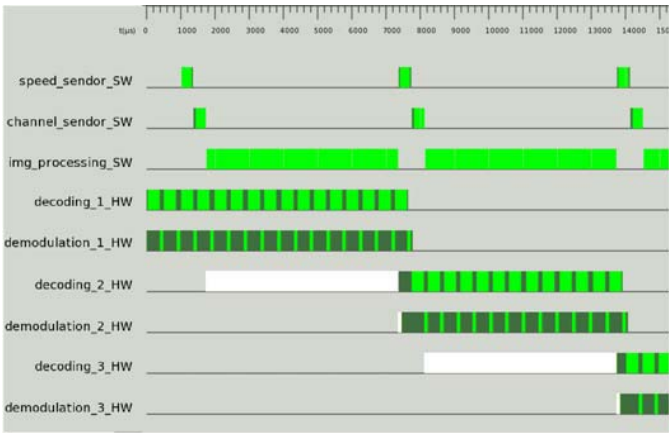

Figure 3: Diagramme d'exécution pour le scenario 2

En suivant la méthodologie, le concepteur peut maintenant modifier des attributs spécifiques et lancer plusieurs simulations successives jusqu'à ce qu'un résultat correct sortent du simulateur. Après plusieurs itérations, il s'est avéré que modifier la taille de la zone reconfigurable de telle sorte que deux instances de démodulateurs et deux instances de décodeurs soit implantées en même temps, peut suffire pour garantir un fonctionnement correct du système.

La Figure 3 décrit les résultats de simulation avec une taille de zone reconfigurable optimisée (scenario 2). Dans ce scénario, les blocs de la seconde configuration peuvent être configurés dans le FPGA alors que les blocs de la configuration 1 continuent à fonctionner. Par conséquent, les données entrantes sont traitées par les tâches adéquates ce qui signifie que le système est capable de permuter entre deux configurations sans perte d'informations, tout en respectant les échéances. Au temps t=8000ns, le service de configuration tire bénéfice de l'inactivité des blocs de configuration1 afin de les remplacer par les blocs de configuration 3. On peut également noter que dans le cas où la configuration 1 serait demandée (à la place de la configuration 3), il ne serait pas nécessaire de procéder au chargement

d'une nouvelle configuration ce qui se traduirait par un temps de configuration nul.

7 Conclusion

Dans ce rapport, nous avons présenté les concepts d'une méthodologie haut niveau destinée à modéliser une plateforme reconfigurable gérée par un système d'exploitation. Cette méthodologie permet au concepteur d'explorer très rapidement des choix architecturaux permettant de porter une application temps réel qui correspond à un récepteur qui implémente une technique de type ACM avec un décodeur LDPC non-binaire. La faisabilité de son implantation a ainsi été démontrée. Les perspectives de ce travail consistent à implanter l'application sur la plateforme ciblée et de tester l'efficacité du modèle.

8 Références

[BC2010] E. Boutillon and L. Conde-Canencia, "Bubble-check: a simplified algorithm for elementary check node processing in extended min-sum non-binary ldpc decoders," *Electronics Letters*, vol. 46, pp. 633–634, april 2010.

[D2.3.2] DAVINCI report, "System Level evaluation, Issue 2", available at www.ict-davinci-codes.eu/project/deliverables/D232.pdf

[Po2008] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,dc)-ldpc codes over gf(q) using their binary images," *IEEE Trans. Commun.*, vol. 56, pp. 1626–1635, Oct. 2008.

[TGm-EMD] IEEE 802.16m, "Evaluation Methodology Document (EMD)," IEEE 802.16m-08/004r5, 15 January 2009.

[TGm-SDD] IEEE 802.16m, "System Description Document (SDD)," IEEE 802.16m08/003r8, 10 April 2009.

9 Bilan financier

A présent, le budget de 5000€ a été seulement partiellement dépensé, comme suit :

- Conférence européenne : 757,87 €
- Participation au GRETSI 2011 : ~ 675 €
- Participation à l'AG du GDR-ISIS : ~ 200 €
- Réunion avancement : 142,25 €

Il nous reste alors ~ 3225 €, que nous envisageons d'utiliser comme suit :

- Participation à ICC'2012, à Ottawa, Canada en Juin 2012 : ~ 2000€
- Participation à EUSIPCO'2012 à Bucharest, Rumania, en Août 2012 : ~ 1000€.

Nous demandons ainsi de pouvoir étendre la période d'utilisation du financement jusqu'au Août 2012.

Les autres réunions, autres que le réunion téléphoniques, qui ont été financées par ailleurs ont eu lieu aux dates suivantes :

- Kick-off meeting (Lorient) : 13/7/2010
- Réunion avancement (Rennes) : 24/11/2010
- Réunion avancement (Rennes) : 30/3/2011
- Réunion finale (Lorient) : 13/9/2011