Conception d'architectures embarquées : des décodeurs LDPC aux systèmes sur puce reconfigurables

François Verdier

UNIVERSITÉ
de Cergy-Pontoise

École doctorale Sciences et Ingénierie **UFR Sciences et Techniques**

Conception d'architectures embarquées : des décodeurs LDPC aux systèmes sur puce reconfigurables

Habilitation à diriger les recherches
Synthèse des travaux

présentée et soutenue publiquement le 5 Décembre 2006

par

François Verdier

Composition du jury

Rapporteurs :	Michel Auguin	Directeur de recherches CNRS Laboratoire I3S
	Emmanuel Boutillon	Professeur à l'Université de Bretagne-Sud Directeur du LESTER
	Jean-Didier Legat	Professeur à l'Université Catholique de Louvain Doyen de la Faculté des Sciences Appliquées
Examinateurs :	Inbar Fijalkow	Professeur à l'ENSEA Directrice du laboratoire ETIS
	Didier Demigny	Professeur à l'Université de Rennes-1
	Michel Robert	Professeur à l'Université de Montpellier II Directeur du laboratoire LIRMM
Invité :	Bernard Candaele	Thalès-Communication

ETIS – Équipes de Traitement des Images et du Signal – UMR 8051

À Clémentine qui, malgré son jeune âge,
a tenu à assister à ma soutenance de doctorat
et à Iris qui, depuis le 9 septembre,
insiste beaucoup pour faire comme sa sœur.

Remerciements

Mes remerciements s'adressent en premier lieu à Didier Demigny qui, en 1995, a accepté de bien vouloir faire partie de mon jury de doctorat et qui m'a éclairé de ses remarques constructives. Je tiens à le remercier également de m'avoir accueilli au sein de son laboratoire, en 2000, après une trop longue période durant laquelle mes perspectives scientifiques s'étaient un peu obscurcies. Cette date a marqué le début d'une nouvelle aventure, qu'il a su favoriser et à laquelle Didier a su pleinement participer.

En second lieu, je tiens à remercier Inbar Fijalkow, qui a pris la succession de Didier à la direction d'ETIS, qui m'a toujours assuré de son soutien et qui a cru en moi.

Je remercie bien sûr les membres de mon jury de bien vouloir me permettre de présenter mes travaux devant eux. Parmi eux, je remercie particulièrement Michel Auguin qui aura eu l'occasion de suivre ma carrière au travers des deux manuscrits que je lui ai fait parvenir et qui, il l'ignore probablement d'ailleurs, a su faire naître en moi l'envie de travailler sur les OS embarqués.

Je remercie également tous les membres de l'équipe « Architecture » du laboratoire ETIS qui ont fait, et qui font encore, que cette équipe n'est pas tout à fait comme les autres...

J'ai une pensée toute particulière et sincère pour mes collègues enseignants-chercheurs de l'IUT qui se sont lancés dans la même aventure que moi au sein de l'équipe « Architecture » et qui ont permis que tout ceci arrive.

Enfin, j'adresse mes remerciements les plus profonds à Stéphanie qui partage avec moi depuis plusieurs années un grand intérêt pour la recherche scientifique, qui connaît bien les difficultés que cela représente, qui me soutient inconditionnellement et qui, en plus, partage ma vie.

Avant-Propos

Ce document présente la synthèse de mes travaux de recherche depuis ma nomination aux fonctions de maître de conférences (en section 61) en 1996 à l'IUT de Cergy, département Génie Électrique et Informatique Industrielle. En Septembre 2000, j'ai intégré le laboratoire ETIS (Équipes de Traitement des Images et du Signal, UMR CNRS 8051). Le laboratoire possède la triple tutelle du CNRS, de l'Université de Cergy-Pontoise (UCP) et de l'École Nationale Supérieure de l'Électronique et de ses Applications (ENSEA).

Les membres du laboratoire ETIS développent leurs activités autour de quatre thèmes et sont organisés en quatre équipes de recherche :

1. L'équipe « Traitement d'images et masses de données »
2. L'équipe « Traitement du signal et modélisation en imagerie »
3. L'équipe « Neurocybernétique »
4. L'équipe « Architecture »

L'équipe « Architecture » du laboratoire ETIS développe ses activités autour de la conception d'architectures électroniques dédiées au traitement optimal des images et du signal. Jusqu'en 2000, les pôles d'excellence essentiels de l'équipe « Architecture » étaient l'étude et la conception d'**Architectures à Reconfiguration Dynamique** et la problématique de l'**Adéquation Algorithme Architecture**. Le premier axe de recherche s'est illustré principalement au travers du projet **ARDOISE** (Architecture Reconfigurable Dynamiquement Orientée Image et Signal Embarquable) qui a été mené par un regroupement de laboratoires (dont ETIS était le principal maître d'œuvre) au sein d'une action incitative. Le deuxième axe étudie l'intégration sur silicium d'algorithmes non triviaux se prêtant *a priori* difficilement à des réalisation matérielles efficaces. J'anime, depuis 2001, le projet principal de ce deuxième axe sur l'étude de l'implantation matérielle de décodeurs LDPC. L'évolution naturelle des activités de l'équipe vers la conception d'architectures de plus en plus complexes a conduit à la création, en 2003, d'un troisième pôle d'excellence sur les **Systèmes sur Puces Reconfigurables** dans lequel s'insère mon deuxième axe d'activité sur les méthodologies de conception d'architectures reconfigurables, notamment dans le cadre d'un projet ANR depuis 2005.

J'ai accompagné cette évolution des activités de l'équipe « Architecture » depuis 2003, date à laquelle j'ai pris la responsabilité de l'animation scientifique de cette équipe. J'ai ainsi aidé activement à l'accroissement de la taille de notre l'équipe en participant notamment aux différentes commissions de recrutement de collègues enseignants-chercheurs ou en motivant certains collègues à participer à nos travaux. J'ai particulièrement apporté ma contribution en négociant plusieurs collaborations académiques et contrats industriels et

en obtenant plusieurs financements de doctorat. J'encadre ou je participe à l'encadrement scientifique de quatre des cinq nouvelles thèses de l'équipe depuis 2003.

Contexte historique

J'exerce les fonctions d'enseignant-chercheur depuis 1995, date de l'obtention de mon doctorat, au sein du département GEII (Génie Électrique et Informatique Industrielle) de l'IUT de Cergy-Pontoise. J'ai été recruté sur un poste de maître de conférences en Septembre 1996. Jusqu'en Septembre 2000, date de mon intégration au laboratoire ETIS, mes activités d'enseignement et mes responsabilités collectives au département GEII ont pris le pas sur mes activités de recherche. Les conditions difficiles de l'exercice d'une activité de recherche au sein de l'IUT (absence d'équipe de recherche sur place, charge importante d'enseignement) expliquent la "pause scientifique" qui apparaît sur mon CV (page 1).

J'ai décidé de reprendre une activité de recherche soutenue depuis Septembre 2000. Ce document est la synthèse de mes travaux depuis cette date.

Activités de recherche

Mes activités de recherche au laboratoire s'articulent autour de deux axes principaux :

- L'étude de l'implantation matérielle efficace de décodeurs LDPC pour la correction des erreurs de transmission dans les systèmes de communication.
- La définition de nouvelles méthodologies de conception d'architectures reconfigurables fortement intégrées, notamment en tenant compte des systèmes d'exploitation embarqués.

Les systèmes modernes de communication (radio, satellitaire, diffusion de programmes multimédia) impliquent en effet des débits d'information de plus en plus importants (jusqu'à quelques Gbits/s dans certains cas) et des qualités de transmission sans défauts. Pour répondre à ce compromis, on cherche à transmettre de l'information au plus près de la limite des canaux, cette limite étant la *capacité* du canal définie par Claude Shannon. Les codes correcteurs d'erreurs permettent d'augmenter la fiabilité des informations transmises et, par conséquent, d'augmenter les débits d'information. Les codes LDPC ont été découverts au début des années 1960 et sont, depuis 1999, revisités en raison de la technologie qui les rends aujourd'hui *faisables*. En effet, le caractère **pseudo aléatoire** des codes, la très forte **irrégularité** des accès aux données, la très importante quantité de calcul et leur **parallélisation difficile** rendent leurs implantations matérielles non triviales. Le développement de décodeurs pour les codes LDPC est une activité très largement soutenue dans la communauté académique mais aussi industrielle en raison des marchés toujours croissants de systèmes de communication. Je participe au développement de ces décodeurs LDPC au laboratoire depuis 2001.

Les systèmes intégrés auxquels les architectes de silicium sont aujourd'hui confrontés (y compris d'ailleurs dans le cas des décodeurs LDPC) exhibent une complexité qui dépasse la productivité des équipes de conception. La loi de Moore – et par un certain opportunisme technologique – les besoins exprimés par les applications, imposent la conception de systèmes d'encombrement minimal (idéalement sur une seule puce) et atteignant des

performances de l'ordre de celles des supercalculateurs d'hier. Les méthodologies et les outils de conception traditionnels sont aujourd'hui mis en échec face à la complexité des systèmes : mélange de logiciel et de matériel, *embarquabilité* de plus en plus forte, très faible consommation, flexibilité des architectures, flot de déploiement d'application sans rupture, etc. L'intégration de structures **reconfigurables** (de type FPGA par exemple) dans les circuits, la prise en compte de contraintes **système** et le développement de flots de modélisation/validation à **très haut niveau d'abstraction** sont aujourd'hui nécessaires. C'est dans ce domaine que j'exerce le deuxième axe de mes activités depuis 2003.

Structure du document

Ce document est organisé en quatre chapitres. Le chapitre 1 est une présentation générale de mon parcours personnel où je présente, de manière synthétique, les faits marquants de mon bilan scientifique (page 4) en termes d'animation, de relations industrielles et de rayonnement. La liste exhaustive de mes encadrements est donnée page 6. Un résumé de mes activités de recherche se trouve page 9. Je donne la liste exhaustive de mes publications en page 15.

Le chapitre 2 détaille mes activités autour de la conception d'architectures efficaces pour le décodage des codes LDPC. Les aspects d'adéquation entre ces algorithmes si particuliers et les architectures y sont présentés ainsi que des applications au cas des codes pour la norme DVB-S2 de diffusion radio de programmes multimédia et au cas des codes travaillant dans des corps de Galois d'ordre élevés.

Le chapitre 3 présente mes activités dans le domaine des méthodologies de conception des architectures reconfigurables embarquées. En particulier, la problématique de la modélisation, de l'exploration et de la conception de systèmes d'exploitation temps-réels embarqués y est présentée.

Les chapitres 2 et 3 présentant des activités relativement distinctes (bien que toutes les deux à connotation très architecturale), j'ai préféré les conclure par deux listes bibliographiques séparées (pages 59 et 95).

Le chapitre 4 clos ce document en dressant le bilan scientifique de mes activités et en dessinant les perspectives à long terme de mes travaux.

Table des matières

1

Parcours

1.1 Curriculum Vitæ

1.1.1 Situation

Né le 9 Mai 1966 à Champigny-sur-Marne (94)
Nationalité : Française
Marié, 2 enfants
Adresse : 8, avenue Charles de Gaulle, Esc. E, 78230 Le Pecq
Téléphone : 08 72 75 22 75

Maître de Conférence à l'Université de Cergy-Pontoise (UCP) depuis 1996
IUT de Cergy – Département GEII
Laboratoire d'accueil : Équipes de Traitement des Images et du Signal (ETIS)
ENSEA/UCP/CNRS-UMR 8051
Responsable de l'équipe « Architecture » du laboratoire ETIS
6, Avenue du Ponceau 95014 Cergy-Pontoise Cedex
Tél : 01 30 73 62 82
Fax : 01 30 73 66 27
Email : verdier@ensea.fr
Web : http ://www-etis.ensea.fr/
Titulaire d'une délégation CNRS au sein du laboratoire ETIS en 2004-2005 et 2005-2006

1.1.2 Parcours académique

2004-2006 : Délégation CNRS au sein du laboratoire ETIS, UMR CNRS 8051
depuis 1996 : Maître de Conférences à l'Université de Cergy-Pontoise, IUT de Cergy, Département GEII
1994-1996 : ATER à l'Université de Cergy-Pontoise (IUT GEII)
1995 : Doctorat d'Électronique de l'Université de Paris-XI, sous la direction du professeur Bertrand Zavidovique, soutenu le 25 Janvier 1995 à l'Institut d'Électronique Fondamentale sur le thème : *Conception de logiciels de synthèse automatique d'architectures VLSI pour le traitement d'images : Le problème du contrôle.* Obtenu avec

la mention Très Honorable et les félicitations du jury composé de :
- Jean Gallice, président,
- Michel Auguin et Didier Demigny, rapporteurs
- Magdy Bayoumi, François Durbin et Georges Quénot, examinateurs

1990 : DEA d'Électronique, option « Architecture des Machines et Traitement du Signal » de l'Université de Paris-XI, obtenu avec les honneurs (major de promotion). Mémoire sur le thème : *Les contrôleurs de chemins de données.*

1.1.3 Parcours scientifique

Mes activités de recherche ont commencé en 1990 lorsque, à l'issue de ma formation académique en EEA (Électronique, Électrotechnique et Automatique), j'ai effectué mon stage de DEA au sein du Laboratoire « Systèmes de Perception » de l'Établissement Technique Central de l'Armement sous la direction du Pr. Bertrand Zavidovique. Au cours de mon stage de DEA, puis au cours de ma thèse, je me suis intéressé à la problématique de la synthèse automatique d'architectures numériques. La problématique au cœur de cette première partie de mes recherches est celle de la génération automatique, à partir d'une description *ad hoc*, d'**opérateurs** VLSI pour le traitement de l'information (en l'occurrence, pendant ma thèse, pour le traitement d'image). Les techniques que j'ai mises en œuvre durant ces travaux sont principalement l'allocation d'opérateurs, l'ordonnancement d'opérations et le partage de registres. De plus, ce problème ayant été démontré NP-complet, une méthode d'optimisation stochastique particulière (le recuit-simulé) a été utilisée comme technique d'optimisation globale. J'ai développé à partir de ces travaux un environnement complet de synthèse automatique d'opérateurs de traitement d'image qui a été démontré par la génération du code RTL VHDL de trois circuits VLSI en technologie CMOS $1\mu m$. J'ai soutenu ma thèse de doctorat en Janvier 1995 [1]. Ces travaux ont donné lieu à plusieurs communications dans des congrès internationaux ([7], [8], [9], [22]) et plusieurs articles dans des revues ou ouvrages : [2] (ouvrage collectif), [3] (revue internationale) et [6] (l'article de synthèse dans la revue Traitement du Signal).

Mes travaux post-doctoraux se sont déroulés au cours de deux périodes distinctes. La première est celle qui a suivi immédiatement l'obtention de mon doctorat et au cours de laquelle mon activité s'est portée principalement sur la fin de mes travaux de thèse. J'ai participé, au sein de mon ancien laboratoire, au démarrage de l'activité d'un doctorant (Ivan Kraljič) sur une thématique proche de mes propres travaux : la conception d'automates de vision à partir de l'émulation sur une machine à programmation fonctionnelle. Plusieurs communications dans des congrès internationaux ont été faites à cette occasion ([10], [11], [23]).

J'ai également, au cours de la même période, démarré une activité sur le site de Cergy (dans les locaux de l'IUT GEII) en collaboration avec l'Université de Paris-XII et sous la responsabilité du professeur Patrick Siarry (alors en poste à l'IUT). Le thème de cette activité était, là encore, orienté vers la conception d'architectures de traitement d'images. J'ai eu la responsabilité du co-encadrement ponctuel d'un doctorant de l'Université de Paris-XII (Nabil Mesbah) dont le travail consistait à développer un modèle d'architecture massivement parallèle pour la restauration d'images par des méthodes stochastiques. Cette activité a conduit à une communication dans une conférence francophone [24].

De 1997 à 2000, mes charges d'enseignement (mise en place de nouveaux cours d'informatique industrielle, de travaux pratiques) aussi bien que mes responsabilités pédagogiques (suivi des stagiaires IUT, administration informatique) m'ont conduit à n'avoir qu'une activité de recherche très restreinte. Seule une collaboration ponctuelle avec le laboratoire AXIS (IEF, Paris-XI Orsay) sous la responsabilité des Pr. B. Zavidovique et A. Mérigot m'a permis de maintenir une activité scientifique. Ces travaux ont porté sur le portage d'un algorithme de mise en correspondance (algorithme des mariages stables) sur un modèle d'architecture massivement parallèle asynchrone pour le traitement d'images (la maille associative d'Orsay) [12]).

Ma deuxième période de recherche post-doctorale a démarré à partir de mon arrivée au sein du laboratoire ETIS (UMR CNRS 8051) en Septembre 2000, date à laquelle j'ai intégré l'équipe « Architecture » du laboratoire. J'anime, depuis 2001, l'un des principaux projets de l'axe « Adéquation Algorithme Architecture et IP » de l'équipe, en collaboration avec David Declercq (équipe « Signal pour les télécommunications » du laboratoire). Ce thème porte sur l'étude et la conception de nouvelles architectures matérielles capables de réaliser la correction des erreurs de transmission (codes LDPC) dans les chaînes de communication numérique. De 2001 à 2003, cette activité a été soutenue par le CNRS au travers d'une action JEMSTIC (Jeunes Équipes, chercheurs et Mobilité) et a conduit à la définition d'une nouvelle classe de codes correcteurs d'erreurs LDPC (les codes HC-LDPC) ainsi qu'à un modèle architectural de décodeurs permettant une parallélisation efficace du décodage. Ce travail qui a, très récemment, été publié dans une revue majeure du domaine [4] se poursuit aujourd'hui dans le cadre d'une collaboration avec la société ST-MICROELECTRONICS au travers de deux thèses de doctorat que je co-encadre.

Au cours de mes travaux, et en considérant la complexité croissante des algorithmes de traitement que nous cherchons à intégrer, j'ai été amené à envisager des implantations sur silicium de plus en plus hétérogènes qui mettent en œuvre des unités de traitement à la fois câblées (matérielles) et programmées (logicielles). Les technologies d'intégration que nous ciblons autorisant aujourd'hui plus d'un milliard de transistors par puces, l'aspect méthodologique de la conception des circuits est réapparu au centre de mes préoccupations. C'est pourquoi j'ai démarré, en 2003, un deuxième axe de recherche orienté vers le développement de nouvelles méthodologies d'intégration **système** des architectures embarquées. J'ai participé à plusieurs groupes de réflexion au niveau national sur ce thème (par l'animation d'actions dans l'EPML POMARD et le GdR ISIS) et j'ai été à l'initiative de la création du thème « Systèmes sur Puces Reconfigurables » de l'équipe « Architecture » que j'anime depuis cette période. Mes activités dans ce domaine s'orientent vers la prise en compte des contraintes d'intégration des systèmes d'exploitation embarqués et vers les méthodologies de conception, de modélisation et d'exploration des architectures complexes. J'encadre scientifiquement deux doctorants sur ces thématiques.

Le résumé de ces activités de recherche est donné page 9.

1.2 Faits marquants

1.2.1 Animation Scientifique

Depuis Avril 2003, je suis responsable de l'équipe de recherche « Architecture » du laboratoire ETIS. L'équipe est aujourd'hui composée de 6 maîtres de conférences (dont 1 HDR), un ingénieur d'étude CNRS à mi-temps (partagé avec une autre équipe du laboratoire) et un ATER de l'UCP. L'équipe accueille actuellement 5 doctorants. Je suis également membre du conseil scientifique du laboratoire.

J'ai eu également l'occasion d'exercer des responsabilités d'animation scientifique à l'échelle nationale :

- Animateur du thème « OS » de l'Équipe Projet Multi-Laboratoires CNRS POMARD (Projet Outils, Méthodes et Architectures pour la Reconfiguration Dynamique) de 2003 à 2004,
- Animateur de l'action « OS pour les architectures de TDSI » du thème C (Adéquation Algorithme Architecture) du GdR ISIS depuis Septembre 2005. Dans ce cadre, j'ai organisé ou participé à l'organisation de plusieurs réunions du GdR ISIS sur le thème « Outils, méthodes et OS pour les architectures de TDSI »,
- Participation, dans le cadre du GdR ISIS, au groupe de réflexion sur le projet d'extension au niveau européen des colloques français JFAAA (Journées Francophones sur l'Adéquation Algorithmes Architectures) et READ (Rétines Électroniques, Asic-fpga et Dsp).

Je participe également à l'animation scientifique des formations doctorales sur le site de Cergy :

- Participation active au montage du dossier du master SIGE (Sciences Informatiques et Génie Électrique) de l'UCP
- Responsable du parcours « Architecture » de la spécialité recherche ESA (Électronique des Systèmes Autonomes) du master SIGE de l'UCP (ouverture de la spécialité en Octobre 2005)
- Co-responsable du parcours « Architecture et Systèmes » de la spécialité recherche SIC (Systèmes Intelligents et Communicants) du master SIGE

1.2.2 Relations académiques et industrielles

Dans le cadre de mes recherches, j'ai été amené à négocier ou à participer à plusieurs collaborations avec des partenaires académiques et industriels. Dans la liste ci-dessous, j'indique les projets sur lesquels j'ai été à l'initiative de la collaboration ou sur lesquels j'ai un rôle majeur (porteur), les projets sur lesquels j'ai un rôle d'interlocuteur (partenaire) et les projets que j'ai aidé à démarrer et sur lesquels je n'ai pas d'activité scientifique importante autre que celle de responsable d'équipe (implication).

Contrats industriels

- Porteur (avec David Declercq, du laboratoire ETIS) d'une collaboration entre les équipes « Architecture » et « Signal pour les télécommunications » du laboratoire et l'entreprise STmicroElectronics (site de Grenoble) sur le thème des architectures de décodeurs LDPC (codes correcteurs d'erreurs) depuis Octobre 2003. Deux financements de thèse BDI ont été obtenus dans ce cadre (Arthur Segard en 2003 puis Adrian Voicila en 2004). J'assure le co-encadrement scientifique de ces deux doctorants.
- Porteur d'une collaboration avec la société Thalès-Communications (site de Colombes) sur le thème de l'intégration sur matériel reconfigurable des concepts de la radio-logicielle. Une thèse CIFRE (Grégory Gailliard) dont j'assure le co-encadrement scientifique est en cours depuis Novembre 2005.
- Partenaire d'une collaboration avec le laboratoire SATIE (UMR 8029, antenne de Cergy), le groupement d'entreprises SAFRAN (Hispano-Suiza, Sagem) et la société Actel (fabricant de circuits FPGA) sur le thème de l'intégration sur SoC de l'électronique de commande des actionneurs pour l'avionique. Le financement d'un post-doctorant est prévu en 2006 et celui d'une thèse de doctorat en 2007.
- Partenaire sur le montage d'un projet de collaboration dans le cadre du pôle de compétitivité System@tic porté par la société Thalès-Trt. Le projet Ter@ops vise à développer les nouveaux concepts architecturaux nécessaires à la réalisation d'un calculateur embarqué massivement parallèle (1 Téra-opérations par seconde dans 1 litre). Le projet a été labellisé par le comité de pilotage du pôle et démarrera le 1er Décembre 2006.
- Porteur d'une collaboration avec la société Thalès-Trt sur le thème de la modélisation des systèmes d'exploitation temps-réels embarqués (thèse BDI de Emmanuel Huck à partir du 1er Novembre 2006).

Collaborations académiques

- Implication dans le montage d'une collaboration entre l'équipe « Architecture » du laboratoire ETIS et le projet académique européen « HESS-2 » (double tutelle du département STIC et de l'IN2P3 du CNRS) pour le développement de la deuxième tranche du projet « HESS »[1] d'astronomie particulaire à haute énergie. Cette collaboration implique l'équipe dans la réalisation de l'architecture du *trigger* (déclencheur d'évènements) de niveau 2 pour la détection et la classification des particules. Une thèse BDI a été obtenue en Novembre 2005 (Sonia Khatchadourian) avec un financement STIC/IN2P3.
- Porteur d'une collaboration avec le groupe R2D2 de l'équipe LASTI de l'IRISA à Lannion sur l'étude des architectures reconfigurables dynamiquement. Cette collaboration a conduit au dépôt d'une Action de Recherche Amont (ARA) qui a été labéllisée par l'Agence Nationale de la Recherche (ANR) en Décembre 2005 et qui implique trois équipes de recherche académiques (R2D2/IRISA Université de

[1] Ce projet implique plusieurs dizaines de laboratoire européens et fait partie des « Grands Équipements » du CNRS.

Rennes-1, SYEL/LISIF Université de Paris-6 et l'équipe « Architecture » du laboratoire ETIS) et une dizaine d'enseignants-chercheurs.

En tant que responsable de l'équipe « Architecture » du laboratoire depuis 2003, je suis également l'interlocuteur privilégié des instances locales (laboratoire ETIS et Conseil Scientifique de l'Université), nationales (CNRS, principalement) et des partenaires industriels pour les recherches de financement et toutes les questions administratives relatives aux bourses doctorales de l'équipe (5 nouvelles thèses depuis 2003, dont 4 BDI et 1 CIFRE).

1.2.3 Encadrements

Depuis ma nomination aux fonctions de maître de conférences, j'ai eu l'occasion d'encadrer ou de co-encadrer plusieurs doctorants. J'ai également encadré plusieurs stages ingénieurs et de master recherche. Voici la liste exhaustive, dans l'ordre chronologique, de mes activités d'encadrement :

Encadrements de thèse

1. Co-encadrement ponctuel, en 1997 et durant moins d'un an, de la thèse de **Nabil MESBAH** sur le thème *Élaboration d'un processeur numérique massivement parallèle dédié au traitement par recuit simulé d'images brouillées.* Thèse de Doctorat de l'Université de Paris 12 sous la direction principale des professeurs K. Madani et P. Siarry, soutenue en juin 1999.
2. Co-encadrement depuis le 1er Décembre 2003 de la thèse de **Arthur Segard**, à hauteur de 60%, avec Didier Demigny (directeur, 30%) et David Declercq (co-encadrant, 10%), sur le thème *Conception d'une architecture reconfigurable pour le décodage LDPC et son intégration dans un environnement intégré de type SoC sous contraintes de temps-réel.* Thèse de l'Université de Cergy-Pontoise avec une bourse BDI co-financée par la société ST-MICROELECTRONICS (Crolles). La soutenance de cette thèse est prévue en Janvier 2007.
3. Co-encadrement depuis le 1er Novembre 2004 de la thèse de **Adrian Voicila**, à hauteur de 30%, avec David Declercq (directeur, 70%), sur le thème *Etude théorique des décodeurs LDPC non binaires, proposition d'architecture et adéquation technologique.* Thèse de l'Université de Cergy-Pontoise avec également une bourse BDI co-financée par la société ST-MICROELECTRONICS (Crolles). La soutenance est prévue fin 2007.
4. Encadrement scientifique principal, depuis le 1er Novembre 2005, de la thèse de **Grégory Gailliard**, (directeur de thèse : Lounis Kessal), sur le thème *Extension de la démarche SCA au niveau plate-forme SoC.* Thèse de l'Université de Cergy-Pontoise avec un financement CIFRE de la société THALÈS-COMMUNICATION (Colombes).
5. Encadrement scientifique principal, à partir du 1er Novembre 2006, de la thèse de **Emmanuel Huck**, (directeur de thèse : Lounis Kessal), sur le thème *OS temps-réel hétérogènes pour les systèmes sur puces reconfigurables : architecture du service de*

gestion des tâches. Thèse de l'Université de Cergy-Pontoise avec une bourse BDI co-financée par la société THALÈS RESEARCH TECHNOLOGY (Palaiseau).

Encadrements de stages

1. Stage de Master recherche de **Jean-Marc Philippe** : *Architecture pour les codes de Gallager*, en 2002. Jean-Marc Philippe a soutenu sa thèse dans l'équipe R2D2 de l'IRISA (antenne de Lannion) en novembre 2005 et est aujourd'hui chercheur au CEA à Saclay.
2. Stage de Master recherche de **Arthur Segard** : *Communications entre tâches logicielles et matérielles dans un contexte d'OS temps réel sur une architecture FPGA reconfigurable*, en 2003.
3. Stage de PFE ENSEA de **Fabien Tulars** : *Architecture SoC pour les codes de Gallager*, en 2003.
4. Stage de PFE ENSEA de **Claire Billaud** : *Mise en œuvre d'un décodeur MPEG-2 sur architecture mixte processeur + FPGA*, en 2004.
5. Stage de PFE ENSEA de **Alexsandro Bonatto** : *Portage d'un système d'exploitation temps réel multiprocesseur sur plate-forme reconfigurable*, en 2005.

1.2.4 Rayonnement

Depuis 2000, j'ai été sollicité en tant que relecteur par différents comités de sélection pour des revues et conférences scientifiques :

- revue *IEEE Transactions on Very Large Scale Integration Systems*
- revue *EURASIP Journal on Applied signal Processing*, numéro spécial *Machine vision on a chip*
- revue *Traitement du Signal*
- conférence *The international conference on Field-Programmable Logic and its Applications (FPL)* en 2004
- conférence *The International Conference on Image Analysis and Processing (ICIAP)* en 2005
- conférence *IEEE Global Telecommunications Conference (Globecom)* en 2005
- membre du comité de programme et animateur d'une session du workshop *Reconfigurable Communication-centric SoCs (ReCoSoC)* en 2006
- colloque francophone GRETSI (Groupe de Recherche et d'Étude en Traitement des Images et du Signal) en 2007
- conférence *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* en 2007

Au-delà de mes activités de recherche, je participe également à différentes instances et commissions :

- Membre élu (collège MCF) du conseil scientifique de l'UCP (deuxième mandat)
- Membre invité du bureau du conseil scientifique de l'UCP depuis Mai 2004

- Membre élu de la commission de spécialistes 61ème section de l'UCP (deux mandats dont un en tant que vice-président)
- Membre de la CSE 61 de l'Université de Paris-XI (deux mandats en tant que suppléant extérieur)
- Membre de la CSE 60-61-63 de l'Université de Paris-XII (jusqu'en 2003)

1.2.5 Enseignement

En dehors de quelques interventions ponctuelles extérieures que j'ai pu effectuer au cours de ma carrière (principalement dans des formations de troisième cycle à l'Université d'Orsay et de Paris-VI), mes enseignements ont eu lieu au sein de l'Université de Cergy-Pontoise, en premier et troisième cycle. J'effectue depuis 1996 l'intégralité de mon service statutaire (192 heures) au département GEII de l'IUT de Cergy. Depuis cette date, mon service réel est d'environ 240 heures TD chaque année (en dehors de ma période de délégation bien entendu).

- **Master SIC** (Systèmes Intelligents et Communicants, ex-DEA TIS) de l'UCP
  - Cours de tronc commun (5 heures) sur les technologies reconfigurables depuis 2001
  - Co-responsable du parcours « Systèmes et Architectures » depuis 2005
- **Master ESA** (Électronique des systèmes Autonomes) de l'UCP (ouverture du master en octobre 2005)
  - Responsable du domaine de compétence « Architecture »
  - Responsable du module « Architectures de traitement pour les systèmes embarqués » de tronc commun (20 heures)
  - Responsable du module « Nouvelles technologies, nouvelles architectures » du parcours de spécialité (20 heures)
- **Master-pro S2IC** (Systèmes Informatiques, Intelligents et Communicants, ex-DESS SMC) de l'UCP
  - Cours sur l'architecture et le fonctionnement des processeurs DSP
  - Cours sur la technologie des circuits reconfigurables
- **Département GEII** de l'IUT de Cergy-Pontoise
  - Responsable du cours, des TD et des TP d'automatique linéaire et échantillonnée depuis 1998. Création de TD, rédaction du support de TD et TP, création de nouveaux TP, montage d'une maquette pédagogique sur la commande des moteurs à courant continu.
  - Cours, TD et TP d'informatique Industrielle (circuits FPGA, microprocesseurs, programmation assembleur). Création de nombreux supports de cours, TD, TP et TR (mini-projets). Montage de plusieurs maquettes pédagogiques pour l'enseignement des circuits FPGA Xilinx et des microprocesseurs.
  - Séminaires en 1996, 1998 et 2001 pour des groupes de 14 à 30 étudiants de la CEE sur les technologies FPGA [29], [30], [31]
  - Responsable de la création du réseau informatique pédagogique et administratif de 1997 à 2004 : création des comptes étudiants, sauvegarde, mise en réseau des ordinateurs et administration des serveurs.
  - Participation aux jurys et suivi de stages.

1.3 Résumé de mes activités de recherche

Depuis mon arrivée au laboratoire ETIS en 2000, mes activités de recherche s'articulent, dans une certaine continuité avec mes travaux de doctorat, autour de deux problématiques architecturales principales : 1) la réalisation d'opérateurs matériels optimaux pour la correction des erreurs de transmission (codes LDPC) dans le domaine des communications numériques et 2) l'étude des méthodologies de conception pour les architectures reconfigurables fortement intégrées (les RSoC : *Reconfigurable System-on-Chip*). Ces deux activités sont résumées ci-dessous :

1.3.1 Architectures pour le décodage LDPC

Ce thème de recherche est né au début de l'année 2001, après mon arrivée au laboratoire, par la mise en commun des compétences locales en traitement de signal (avec David Declercq) et en conception d'architectures intégrées. L'objectif de ce thème est de concevoir de nouvelles architectures numériques capables d'implanter les algorithmes utilisés dans les systèmes de communication développés au sein d'ETIS.

Dans les systèmes de communication numérique modernes (objets communicants, assistants personnels, radiotéléphones) une large part de l'efficacité des transmissions dépend des algorithmes de codage et de décodage de canal. Ces algorithmes sont responsables de la détection comme de la correction des erreurs de transmission et deviennent absolument nécessaires dès lors qu'il s'agit de transmettre des données autres que la voix (données informatiques, images, trames TCP/IP, etc). Une attention toute particulière a été portée sur des algorithmes itératifs originaux de correction d'erreur depuis une dizaine d'années (les Turbo-Codes). D'autres types d'algorithmes possédant des comportements asymptotiques d'excellente qualité ont été développés (par Gallager, en 1963) et redécouverts récemment (par MacKay, en 1999). Ils sont aujourd'hui étudiés avec beaucoup d'attention. C'est donc naturellement vers ces types d'algorithmes que s'est orienté ce thème de recherche pluridisciplinaire. Cette action a été soutenue activement par le laboratoire ETIS et a été financée sur la période 2001-2002 puis 2002-2003 par le CNRS au travers d'une action JEMSTIC (Jeunes Equipes, Chercheurs et Mobilité) pour un montant total de 26000 euros.

Les codes HC-LDPC et les architectures de décodeurs

Les codes LDPC (*Low Density Parity Check*) sont basés sur des algorithmes itératifs de propagation de croyance sur des graphes factoriels. La nature très irrégulière des transferts de données au sein de ces algorithmes ainsi que la quantité importante de données à traiter rendent les implantations matérielles non triviales. Les deux premières années d'activité de ce thème ont permis de développer conjointement un modèle architectural et une classe de codes LDPC (les codes HC-LDPC pour *Hardware-Constrained LDPC*) rendant faisable et efficace l'intégration de ces codes dans les systèmes de communication numériques mobiles. En particulier, nous avons formalisé une méthode de construction parallèle de matrices LDPC qui simplifie l'implantation matérielle en réduisant significativement la quantité de mémoire nécessaire pour le codage de la matrice. De plus, le modèle architectural des

décodeurs associés permet une réalisation parallèle efficace, résolvant le problème difficile de l'accès pseudo-aléatoire aux données sans dégrader les performances de décodage.

J'ai encadré en 2002 un stage de DEA sur ce thème (Jean-Marc Philippe) et un stage de fin d'études ENSEA en 2003 (Fabien Tulars). Cette activité a donné lieu à plusieurs communications dans des congrès nationaux et internationaux ([13], [14], [25]) et, très récemment, à la publication d'un article dans une revue majeure [4]. Plusieurs collaborations nationales ont également été menées dans le cadre de ces travaux :

- Participation à l'Action Spécifique « Radio Logicielle » (AS 27),
- Dépôt d'un dossier de projet RNRT (PRALINE) en 2003 avec le laboratoire LESTER de l'Université de Bretagne-Sud, l'ENST de Bretagne et la société Turboconcept (financement non obtenu),
- Communication dans le cadre d'une journée du GDR ISIS [32].

Ces travaux ont suscité l'intérêt d'un acteur industriel majeur (la société ST MICROELECTRONICS) qui a décidé de participer à ces recherche sous la forme du co-financement de deux thèses de doctorat.

Les décodeurs LDPC pour la norme DVB-S2

La première thèse concerne le développement d'une architecture matérielle pour la réalisation du décodeur LDPC utilisé dans la norme de radiodiffusion vidéo DVB-S2 (Arthur Segard). Dans la norme DVB-S2, la taille des codes (jusqu'à 64800 bits), le débit (110 Mbits par seconde en QPSK) et la performance à atteindre (taux d'erreur de paquets de 10^{-7}) imposent la réalisation d'architectures de décodeurs particulièrement performantes.

L'approche d'Adéquation Algorithme-Architecture qui a été menée dans les travaux de Arthur Segard a consisté à pousser plus loin l'étude des compromis respectifs sur la parallélisation, des équations de décodage et de l'ordonnancement de ces équations. Une architecture originale, mélant l'algorithme dit *min-sum* à un très fort taux de parallélisation (jusqu'à 360 processeurs élémentaires) et à une technique de résolution des conflits d'accès des processeurs aux mémoires a été développée. Cette architecture exhibe des performances supérieures aux solutions concurrentes et permet d'atteindre des débits importants grâce à une accélération de la convergence [18].

Le décodage des codes LDPC non-binaires

La deuxième thèse porte sur l'implémentation des algorithmes de décodage LDPC dans les corps de Galois d'ordre élevé (par rapport aux implémentations binaires traditionnelles). Les travaux de Adrian Voicila ont porté en premier lieu sur l'évaluation et la réduction de la complexité de ces algorithmes. En effet, il est montré que des implémentations dans GF(256) – où les quantités élémentaires ne sont plus binaires mais des vecteurs à 256 valeurs – permettent d'obtenir des performances très supérieures aux codes binaires. En contrepartie, la complexité matérielle de tels décodeurs, en termes aussi bien de quantité de mémoire nécessaire comme de temps de calcul, devient prohibitive. L'algorithme EMS (*Extended Min-Sum*) associé à une implémentation de type *forward-backward* des opérations élémentaires et à un paramétrage du décodage a permis de réduire signi-

ficativement la complexité d'un décodeur LDPC dans $GF(q)$ ([27], [20]). Une réalisation matérielle devient alors envisageable.

1.3.2 Intégration système des SoC

Le deuxième axe de mes recherches consiste à étudier les méthodologies de développement et d'intégration système des architectures de type RSoC (*Reconfigurable System On Chip* : systèmes sur puce reconfigurables). Ce type d'architecture de circuit consiste à intégrer sur une seule puce de silicium l'ensemble des ressources de calcul et de mémorisation nécessaires à un système de traitement de l'information. L'utilisation de ces architectures permet de répondre aux problèmes critiques d'embarquabilité, de flexibilité et de densité d'intégration. L'intégration des architectures reconfigurables (de type FPGA par exemple) au sein de ces SoC pose de nombreux problèmes méthodologiques aussi bien qu'architecturaux qui ont été identifiés comme des verrous scientifiques par la communauté des architectes de silicium.

Au sein du laboratoire ETIS, j'ai démarré une activité visant à étudier en premier lieu les problèmes de communication entre les différents dispositifs internes (matériels et logiciels) d'un SoC afin de proposer une méthodologie efficace de développement d'applications sur de telles architectures. L'existence d'un système d'exploitation embarqué dans l'architecture a été montré comme étant une solution possible pour répondre au problème de l'hétérogénéité de la plate-forme [15]. Ces premiers résultats ont guidé ce nouvel axe de mes recherches sur la problématique d'intégration système et les OS embarqués pour les architectures reconfigurables. Dans ce cadre, j'ai participé activement à l'Équipe Projet Multi-Laboratoires (EPML) CNRS « POMARD » en tant qu'animateur du thème « Intégration Système - OS » en 2003 et 2004. J'anime également cette thématique au sein du GdR ISIS dans le thème C (AAA) au travers de l'action « OS pour les architectures de TDSI ».

La modélisation des systèmes d'exploitation

À partir des conclusions qui ont été tirées des activités de ce groupe de travail ([33]), j'ai initié une thématique « OS pour le reconfigurable » au sein du laboratoire ETIS à laquelle participent aujourd'hui plusieurs jeunes maîtres de conférences. L'équipe s'est associée à deux laboratoires français pour démarrer en 2005 un Projet de Recherche Amont (projet ANR OvERSoC) sur le développement d'une méthodologie d'exploration et de validation des architectures de RTOS embarqués pour les plates-formes reconfigurables. Les résultats préliminaires de ces travaux ont été publiés dans des conférences francophones et internationales ([16], [21], [26]).

Ma participation personnelle dans ce projet est le développement d'une stratégie de modélisation des OS embarqués dans les flots de conception des plate-forme RSoC. J'ai développé en particulier un modèle d'OS temps-réel en SystemC autorisant la simulation à plusieurs niveaux d'abstraction successifs de la couche exécutive d'une telle plate-forme. Le modèle d'OS qui a été développé utilise les mécanismes spécifiques de la version 2.1 du noyau de simulation SystemC et autorise un comportement totalement dynamique du

logiciel embarqué. Ces travaux font actuellement l'objet d'une soumission d'article dans une revue internationale [5].

Ces travaux constituent les première pistes d'une thèse de doctorat, avec une bourse BDI du CNRS et co-financée par la société THALÈS RESEARCH TECHNOLOGY, sur la modélisation et le raffinement des OS embarqués. Cette thèse, qui a démarré le 1er Novembre 2006, se déroulera à l'interface des activités du projet OVERSOC et de l'un des projets de coopération du pôle de compétitivité SYSTEM@TIC dans lequel l'équipe « Architecture » du laboratoire participe (le projet TER@OPS). L'objectif du projet TER@OPS est de développer une nouvelle architecture de calculateur embarqué à très hautes performances (1 Téra-opérations par secondes) au sein duquel un système d'exploitation temps-réel embarqué pourrait participer à la surveillance de la qualité de service et assurer la fiabilité et la tolérance aux pannes de l'architecture.

La conception d'architectures pour la radio-logicielle

Le problème de la définition de nouvelles méthodologies de conception d'architectures complexes est également au centre d'une collaboration avec la société THALÈS-COMMUNICATION au travers d'une thèse CIFRE qui a démarré en Novembre 2005 et que j'encadre scientifiquement (thèse de Grégory Gailliard). L'axe qui est développé dans cette thèse étudie la déclinaison matérielle des mécanismes à mettre en œuvre pour le déploiement sur architectures reconfigurables (FPGA par exemple) d'applications réparties. Le choix d'une spécification à très haut niveau des formes d'ondes est guidé par la méthodologie SCA (*Software Communication Architecture*) imposée pour les nouveaux systèmes de radiocommunication militaires.

Les premiers travaux ont mis en évidence la nécessité d'adopter une méthodologie de conception autorisant réellement la réutilisation de blocs matériels [17] et le besoin de définir des méthodologies adaptées au cas particulier des plates-formes de radio-logicielle [19]. Ils se poursuivent actuellement par l'étude de l'intégration matérielle de mécanismes de type *conteneurs CORBA* pour résoudre les problèmes liés à la portabilité et l'interopérabilité des plates-formes et formes d'ondes.

1.4 Perspectives de recherche

Les perspectives de ce travail s'articulent selon deux directions correspondant aux deux axes de mes recherches en cours : **la conception d'architectures génériques et programmables pour le décodage LDPC** et **le développement d'une méthodologie d'exploration des architectures reconfigurables embarquées**. Ces travaux s'intègrent plus globalement et de manière naturelle dans une perspective unique visant à proposer des outils de développement, de validation et d'exploration des nouvelles architectures embarquées pour le traitement de l'information.

L'étude des nouveaux modèles architecturaux d'applications non triviales

L'exemple du décodage de canal et des algorithmes de correction des erreurs de transmission dans les communications numériques est représentatif d'une nouvelle catégorie de problèmes auxquels sont confrontés aujourd'hui les concepteurs d'architectures électroniques. La quantité de calcul à effectuer, à des vitesses toujours plus importantes et sous des contraintes de flexibilité et d'*embarquabilité* toujours plus sévères conduit à proposer des modèles de calcul en rupture avec les schémas classiques. Il s'agit de proposer des modèles architecturaux à parallélisme massif tout en respectant les contraintes des systèmes embarqués (consommation réduite d'énergie, taille de silicium restreinte). De plus, pour faire face à l'évolution rapide des standards et des conditions d'utilisation, ces systèmes doivent posséder des caractéristiques de flexibilité importante. L'utilisation de certaines propriétés de *reconfiguration dynamique* des ressources doit permettre d'atteindre ces objectifs.

Ces modèles architecturaux sont en rupture avec les approches classiques dans le sens où les sous-systèmes en question ne sont plus conçus indépendamment de l'environnement dans lequel il s'intègrent (la chaîne de communication par exemple). La prise en compte du contexte extérieur ou de contraintes de plus haut niveau s'avère nécessaire. Il faut dès lors proposer des méthodologies de conception *ad hoc* pour ces modèles architecturaux novateurs.

L'exploration et la validation des mécanismes de gestion dynamique des plates-formes

Ici, ce sont les contraintes économiques et industrielles qui motivent l'étude de nouvelles techniques de conception. Le *time-to-market* guide aujourd'hui les industriels à remettre en cause leurs méthodes de conception pour fournir sur le marché des circuits dont la durée de conception et le coût de développement doivent être minimaux. Là encore, des plates-formes génériques, flexibles et reprogrammables sont nécessaires. Des flots de conception complets – de l'expression du besoin « client » à la réalisation sur silicium – restent à définir. La *reconfigurabilité* des plates-formes est aujourd'hui l'approche la plus prometteuse et doit être prise en compte dans les méthodologies de conception.

Parallèlement, l'existence de ces nouvelles architectures ainsi que les flots de conception correspondant nous autorise à revisiter un certain nombre de problèmes qui ont pu être, jusque là, écartés faute de solutions matérielles réalistes. Le domaine de la vision

embarquée pour la robotique autonome est un exemple caractéristique d'application où la forte flexibilité d'une architecture embarquée peut apporter des solutions innovantes et ouvrir de nouvelles voies de recherche.

Vers les architectures auto-adaptables ?

Plus globalement, on cherche aujourd'hui à concevoir des systèmes capables de s'adapter aussi bien à des environnements non connus *a priori* (la robotique autonome en est certainement le meilleur exemple) qu'à des conditions de fonctionnement en évolution permanente (c'est le cas des réseaux de communication *ad hoc*). Tolérance aux pannes ou aux fautes, sûreté de fonctionnement, adaptation automatique, flexibilité et généricité sont les caractéristiques nécessaires des systèmes de traitement de l'information qu'il nous est demandé de concevoir. *Adaptive Computing*, architectures bio-inspirées, modèles génétiques sont les mots-clés des futurs programmes de recherche en conception d'architectures embarquées dans lesquels pourront s'intégrer mes travaux.

1.5 Bibliographie personnelle

Thèse de doctorat

[1] François Verdier. *Conception de logiciels de synthèse automatique d'architectures VLSI pour le traitement d'images : le problème du contrôle.* PhD thesis, Université de Paris-XI, Institut d'Électronique Fondamentale, Janvier 1995.

Chapitre d'ouvrage

[2] B. Zavidovique, C. Fortunel, G. Quénot, A. Safir, J. Sérot, and F. Verdier. *VLSI Design Methodologies for Digital Signal Processing Architectures*, chapter Automatic synthesis of vision automata, pages 261–318. Kluwer Academic Publisher, 1994.

Articles dans des revues internationales

[3] F. Verdier and B. Zavidovique. A High Level Synthesis System for VLSI Image Processing Applications. *International Journal of VLSI Design*, 7(4) :111–121, 1998.

[4] F. Verdier and D. Declercq. A Low Cost Parallel Scalable FPGA Architecture for Regular and Irregular LDPC Decoding. *IEEE Transactions on Communications*, 54(7) :1215–1223, July 2006.

Articles de revue en cours d'expertise

[5] François Verdier, Mickaël Maillard, and Benoît Miramond. Designing a custom SoC for robotic vision : the problem of modelling the embedded real-time operating system. In *EURASIP Journal on Applied Signal Processing*, Soumis en septembre 2006.

Articles dans des revues nationales

[6] F. Verdier and B. Zavidovique. Des architectures intégrées pour la vision : synthèse automatique en trois exemples. *GRETSI/CNRS Traitement du signal*, 14(2) :227–249, 1997.

Communications dans des conférences internationales avec actes

[7] F. Verdier, A. Safir, and B. Zavidovique. A High Level Synthesis Algorithm Including Control Constraints. In *Proceedings of the 18th EUROMICRO Conference - Paris, France*, September 1992.

[8] F. Verdier and B. Zavidovique. A Complete Environment for Global Architecture Synthesis. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'1993) - New Orleans, LA, USA*, pages 77–81, December 1993.

[9] F. Verdier and B. Zavidovique. High Level Synthesis of a Defect Detector. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'1995) - Como, Italia*, pages 411–415, September 1995.

[10] I. Kraljic, F. Verdier, G. Quénot, and B. Zavidovique. Systematic Design of Image Processing ASICs through Real-Time Emulation. In *11th International Conference on Systems Engineering, (ICSE'1996) - Las Vegas, USA*, July 1996.

[11] I. Kraljic, F. Verdier, G. Quénot, and B. Zavidovique. Investigating Real-Time Validation of Real-Time Image Processing ASICs. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'1997) - Cambridge, MA, USA*, pages 116–125, October 1997.

[12] F. Verdier, A. Mérigot, and B. Zavidovique. Fast Stable Matching Algorithm Using Asynchronous Parallel Programming Model. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'2000) - Padova, Italy*, pages 131–135, September 2000.

[13] F. Verdier and D. Declercq. A LDPC Parity Check Matrix Construction for Parallel Hardware Decoding. In *Third International Symposium on Turbo Codes and Related Topics*, pages 235–238, Brest, France, Sept. 2003.

[14] D. Declercq and F. Verdier. Optimization of LDPC Finite Precision Belief Propagation Decoding with Discrete Density Evolution. In *Third International Symposium on Turbo Codes and Related Topics*, pages 479–482, Brest, France, Sept. 2003.

[15] A. Segard and F. Verdier. SOC and RTOS : Managing IPs and Tasks Communications. In *International Conference on Field-Programmable Logic and its Applications (FPL)*, pages 710–718, Antwerp, Belgium, August 2004.

[16] I. Benkhermi, A. Benkhelifa, D. Chillet, S. Pillement, J-C. Prévotet, and F. Verdier. System-Level Modelling for Reconfigurable SoCs. In *20th Conference on Design of Circuits and Integrated Systems (DCIS'05)*, Lisboa, Portugal, November 2005.

[17] G. Gailliard, B. Mercier, M. Sarlotte, B. Candaele, and F. Verdier. Towards a SystemC TLM based Methodology for Platform Design and IP Reuse : Application to Software Defined Radio. In *Second European Workshop on Reconfigurable Communication-centric SoCs (ReCoSoC'06)*, Montpellier, France, July 2006.

[18] Arthur Segard, François Verdier, David Declercq, and Pascal Urard. A DVB-S2 compliant LDPC decoder integrating the Horizontal Shuffle Scheduling. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Tottori, Japan, December 2006.

[19] Gregory Gailliard, Eric Nicollet, Michel Sarlotte, and François Verdier. Transaction Level Modelling of SCA compliant Software Defined Radio Waveforms and Platforms PIM/PSM. In *International Conference on Design and Test in Europe (DATE 2007)*, Nice, France, April 2007.

[20] Adrian Voicila, David Declercq, François Verdier, and Marc Fossorier. Low complexity, Low memory EMS algorithm for non-binary LDPC codes. In *IEEE International Conference on Communications (ICC 2007)*, Glasgow, Scotland, June 2007.

Communications invitées

[21] F. Verdier, J-C. Prévotet, A. Benkhelifa, D. Chillet, and S. Pillement. Exploring RTOS issues with a high-level model of a reconfigurable SoC platform. In *European Workshop on Reconfigurable Communication-centric SoC (ReCoSoC 2005)*, Montpellier, France, June 2005.

Communications dans des conférences nationales avec actes

[22] F. Verdier and B. Zavidovique. Vers une automatisation complète de la conception d'automates de vision intégrés. In *RFIA'1996 : Congrès AFCET sur la Reconnaissance des Formes et l'Intelligence Artificielle - Rennes, France*, January 1996.

[23] I. Kraljic, F. Verdier, G. Quénot, and B. Zavidovique. Conception systématique de circuits vlsi de vision par émulation fonctionnelle. In *Actes du Colloque READ'97 - Rétines Electroniques, ASIC-FPGA et DSP pour la vision et le traitement d'images en temps réel - INT, Paris*, Mai 1997.

[24] N. Mesbah, K. Madani, P. Siarry, and F. Verdier. Codage des paramètres du recuit-simulé pour la conception d'une architecture numérique massivement parallèle dédiée au traitement d'images. In *Seconde Conférence Internationale sur les Métaheuristiques - Sophia-Antipolis, France*, Juillet 1997.

[25] F. Verdier, D. Declercq, and J-M. Philippe. Parallélisation et Implantation FPGA d'un Décodeur LDPC. In *Proceedings of the JFAAA*, pages 28–31, Monastir, Tunisie, Dec. 2002.

[26] I. Benkhermi, A. Benkhelifa, D. Chillet, S. Pillement, J-C. Prévotet, and F. Verdier. Modélisation niveau système de SoC reconfigurables. In *8ème Symposium en Architectures nouvelles de machines (SympA'05)*, Le Croisic, France, April 2005.

[27] A. Voicila, D. Declercq, M. Fossorier, and F. Verdier. Algorithmes simplifiés pour le décodage de codes LDPC non-binaires. In *20ème Colloque sur le Traitement du Signal et des Images (GRETSI'05)*, Louvain-La-Neuve, Belgium, September 2005.

Communications diverses

[28] François Verdier. Expérience pédagogique de l'utilisation d'une solution intégrée pour la conception de systèmes numériques complexes et le prototypage. Séminaire "OrCAD – Pspice V9 : l'offre enseignement", Université de Jussieu, Mars 1999.

[29] François Verdier. XILINX FPGA Technology : From schematic capture to device programming. Séminaire IUT GEII pour un public de 14 étudiants universitaires Finlandais, Avril 1996.

[30] François Verdier. XILINX FPGA Technology : From schematic capture to device programming. Séminaire IUT GEII pour un public de 30 étudiants universitaires CEE (Finlande, Allemagne, Irlande), Février 1998.

[31] François Verdier. XILINX FPGA Technology : From schematic capture to device programming. Séminaire IUT GEII pour un public de 20 étudiants universitaires CEE (Finlande, Allemagne, Irlande, Angleterre), Avril 2001.

[32] François Verdier and David Declercq. Quelques aspects de l'implantation matérielle des codes LDPC. Communication GDR ISIS/LDPC, Décembre 2002.

[33] François Verdier. Bilan du thème OS de l'EPML POMARD : Problématique d'intégration OS dans les plate-formes à reconfiguration dynamique. Workshop du RTP SOC, Mai 2004.

2

Conception et Optimisation d'Architectures de Décodeurs LDPC

Sommaire

2.1 Introduction

Dans le domaine des systèmes de communication sur canal bruyant, une intense activité de recherche s'est développée depuis une cinquantaine d'années. Ces recherche se sont en particulier concentrées sur les techniques permettant de protéger les informations transmises vis-à-vis du bruit du canal et de corriger les erreurs de transmission. Deux grandes familles de codes correcteurs d'erreurs sont étudiées massivement depuis environ dix ans : les turbo-codes [35] et les codes LDPC (*Low Density Parity Check*) [52]. Les codes LDPC sont des codes à vérification de parité construits à partir de matrices pseudo-aléatoires. Les caractères creux et pseudo-aléatoire de ces matrices influencent d'ailleurs fortement les performances de ces codes.

Dans la communauté du codage, de très nombreux travaux ayant conduit à une production scientifique importante ont été menés sur la construction de ces codes correcteurs d'erreurs. Il est hors du propos de ce manuscrit de décrire les travaux dans ce sens mais nous verrons dans la suite que l'approche « architecturale » des codes LDPC, initiée par les travaux de V. Sorokine *et. al* dans [61] et [62], a permis de développer certaines techniques de construction de matrices ayant de bonnes propriétés matérielles ([36], [37], [69], [73]), en particulier du point de vue de la parallélisation ([49], [65], [59]) tout en obtenant également de bonnes performances de décodage. Des implémentations sur technologies reconfigurables ont également été proposées ([50], [71], [47], [39], [46]).

Le problème particulier auquel nous nous intéressons ici est celui de concevoir des architectures de décodeurs. L'opération de codage est, par nature, relativement simple en appliquant des opérateurs de calcul de parité (OU exclusif en binaire). Pour le décodage en revanche, les codes LDPC requièrent une quantité de calcul importante, qui dépend d'ailleurs beaucoup de la taille du code, sur des données généralement stockées dans des mémoires temporaires dont la taille est critique elle aussi. La performance de décodage d'un code est calculée généralement en taux d'erreur binaire (BER : *Bit Error Rate*) ou en taux d'erreur de trame (FER : *Frame Error Rate*) puisque ce sont des codes en bloc. Nous avons montré que de nombreux paramètres interviennent dans la performance de décodage :

- l'algorithme de décodage (l'algorithme général de propagation de croyance ou ses nombreux dérivés),
- les équations spécifiquement utilisées pour le calcul des données extrinsèques et leurs implantations,
- l'ordonnancement des équations de décodage,
- la représentation des données (virgule flottante, virgule fixe) et les erreurs d'arrondis éventuelles qu'elle entraîne,
- le degré de parallélisation du décodeur.

Nous montrerons dans la suite de ce chapitre que l'approche que nous avons adoptée dans les différents cas qui nous ont intéressés consiste à examiner la manière dont tous ces paramètres interagissent pour définir, selon les cas, soit de nouvelles familles de codes (par la manière dont les matrices sont construites), soit de nouveaux algorithmes de décodage

(par l'implémentation particulière qui en est faite) et à définir les modèles architecturaux de décodeurs adaptés. Pour toute cette partie, nous nous sommes limités aux canaux dits « classiques » à bruit blanc additif gaussien (AWGN).

Nous donnons les éléments clés des principes et des notations relatifs aux codes LDPC et à leur représentation dans la suite de cette introduction. La partie 2.2 détaille les premiers résultats que nous avons obtenus avec les codes dits HC-LDPC et l'architecture des décodeurs associés utilisant l'algorithme *log-BP*. La partie 2.3 décrit ensuite l'étude menée sur le cas particulier des codes qui ont été adoptés pour la norme DVB-S2. Est présentée en particulier la remise en cause de l'algorithme de décodage au profit d'une version plus adaptée au fort parallélisme nécessaire. La partie 2.4 est plus exploratoire dans la mesure où nous étudions la complexité de codes travaillant dans des corps de Galois d'ordre élevé et qui n'ont, à ce jour, pas encore d'implantation matérielle faisable.

2.1.1 Les codes LDPC : notations

Un code LDPC régulier ou irrégulier est défini par un bloc de N symboles (le mot de code transmis) et un ensemble de M équations de vérification de parité. Chaque équation de parité s'applique sur un sous-ensemble des symboles du mot de code. Dans le cas particulier d'un code LDPC régulier, tous les symboles participent exactement au même nombre de fonctions de parité et toutes les fonctions de parité s'appliquent sur le même nombre de symboles. Dans le cas général d'un code irrégulier, ces quantités sont exprimées par deux polynômes $\lambda(x)$ et $\rho(x)$ détaillés plus loin.

Le mot de code $\underline{C}$ de longueur N contient $K = N - M$ symboles dits d'information et M symboles de redondance (servant à protéger les symboles d'information). Le rendement de ce code est au maximum $R \leq \frac{K}{N}$.

$$H = \begin{Bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{Bmatrix}$$

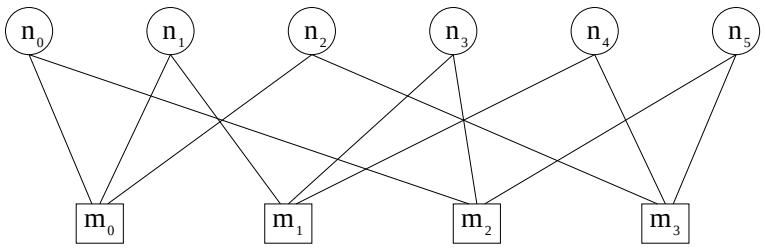


FIG. 2.1 – Matrice H d'un code LDPC et représentation sous forme de graphe factoriel. Les valeurs non nulles de la matrice correspondent à des branches du graphe entre les nœuds de variable (n_i) et les nœuds de parité (m_j). Dans cette figure le code est binaire et régulier.

Un code LDPC peut être représenté (figure 2.1) par sa matrice de vérification de parité H, de taille $M \times N$ et de densité $N_e/(M \times N)$ (N_e est le nombre de valeurs non nulles dans la matrice). Pour tout mot de code $\underline{C}$, la matrice H vérifie l'équation de parité $H \cdot \underline{C} = 0$. Dans le cas d'un code binaire, une valeur non nulle $h_{mn} = 1$ dans la matrice indique que le n^{ieme} symbole est utilisé dans la m^{ieme} équation de parité. Un code LDPC binaire régulier est donc défini par une matrice contenant exactement le même nombre de 1 dans chaque colonne et le même nombre de 1 dans chaque ligne. Dans le cas d'un code non-binaire (comme ceux qui sont utilisés dans la partie 2.4), les quantités h_{mn} sont des valeurs q-aires définies dans $GF(q)$.

La matrice de décodage peut également être représentée de manière graphique par un graphe bipartite possédant N nœuds associés aux symboles du canal (les nœuds de données) et M nœuds de vérification de parité (les nœuds de parité) comme celui de la figure 2.1. En utilisant les notations de [51], les connectivités de ces nœuds sont données par les polynômes $\lambda(x)$ et $\rho(x)$ où :

$$\lambda(x) = \sum_{i=2}^{d_{v_{max}}} \lambda_i x^{i-1} \tag{2.1}$$

est utilisée pour indiquer la proportion des branches du graphe (λ_i) qui sont connectées à des nœuds de données de degré i, et :

$$\rho(x) = \sum_{j=2}^{d_{c_{max}}} \rho_j x^{j-1} \tag{2.2}$$

qui donne la proportion des branches (ρ_j) connectées à des nœuds de parité de degré j.

Un nœud de donnée de degré i (respectivement un nœud de parité de degré j) est un nœud connecté à exactement i (respectivement j) branches du graphe. Les quantités $d_{v_{max}}$ et $d_{c_{max}}$ sont les bornes maximales de connectivités des nœuds.

La matrice H est typiquement construite de manière aléatoire ce qui conduit à une topologie aléatoire et irrégulière du graphe.

2.1.2 Le décodage des codes LDPC

Le décodage d'un code LDPC consiste à appliquer les équations de décodage sur l'ensemble des nœuds du graphe jusqu'à convergence de l'algorithme. Étant donnée la structure intrinsèquement parallèle de la représentation graphique du code, il existe un grand nombre de manières d'ordonnancer dans le temps ces équations de décodage.

Plusieurs approches de décodage des codes LDPC existent, qui sont toutes basées sur l'algorithme dit de propagation de croyance (*BP : Belief Propagation*) [38]. Toutes les études architecturales présentées dans ce manuscrit n'utilisent pas cet algorithme de propagation de croyance mais en sont inspirées. C'est pourquoi nous utiliserons pour la suite de cette introduction l'algorithme BP et, plus précisément, sa réalisation dans le domaine logarithmique (*log-BP*) pour des raisons évidentes de simplification architecturale (implantations de sommes et soustractions au lieu de multiplications et divisions).

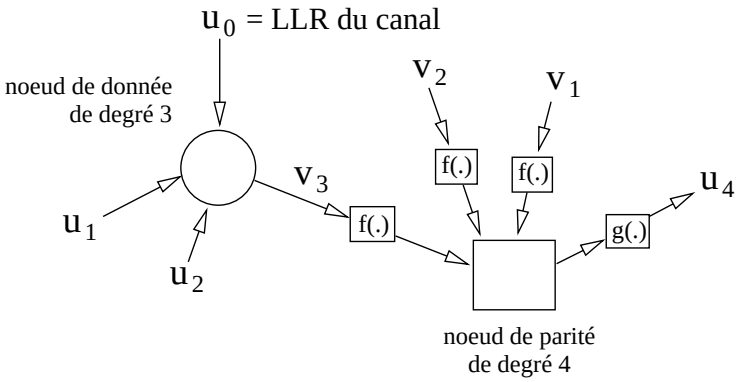


FIG. 2.2 – Représentation et notations des nœuds et des messages, les flèches représentent le flot des messages.

Décoder un code LDPC avec l'algorithme log-BP consiste à propager des *messages* le long des branches du graphe. On notera v_k les messages circulant des nœuds de données vers les nœuds de parité et u_k les messages circulant des nœuds de parité vers les nœuds de données. Les messages représentent des probabilités (ou des croyances) des symboles vers les équation de parité et des équations sur les symboles respectivement. Une dernière quantité manipulée par l'algorithme est le logarithme du rapport de vraisemblance (*LLR : Log-Likelyhood Ratio*) qui est associée à chacun des nœuds de données. Ce rapport est utilisé comme entrée de l'algorithme et provient du canal.

Les notations suivantes sont utilisées pour la description de l'algorithme log-BP :

- Le mot binaire codé à transmettre sur le canal est : $\underline{C} = [c_0 ... c_{N-1}]^T$
- La sortie du modulateur (du coté émetteur) : $\underline{X} = Mod[\underline{C}]$
- La fonction de transfert du canal est : $\underline{Y} = \underline{X} + \underline{W}$ où $\underline{W}$ est un bruit blanc additif gaussien.
- L'algorithme est initialisé avec le logarithme du rapport de vraisemblance : $u_0[n] = \log\left(\frac{p(c_n=0|y_n)}{p(c_n=1|y_n)}\right)$

L'algorithme de décodage consiste en trois étapes principales qui sont exécutées de manière itérative (cf. figure 2.2 pour les notations) :

1. Le processus des nœuds de données (*data update*) qui calcule les messages v_k à partir des messages u_k et des LLR. Pour chaque nœud de donnée (i étant le degré du nœud et u_0 le logarithme du rapport de vraisemblance associé à ce nœud) :

$$v_k = u_0 + \sum_{l=1, l\neq k}^{i} u_l \tag{2.3}$$

2. Le processus des nœuds de parité (*check update*) qui calcule les messages u_k. Pour chaque nœud de parité (j étant le degré du nœud) :

$$u_k = g\left(\sum_{l=1, l\neq k}^{j} f(v_l)\right) \times \prod_{l=1, l\neq k}^{j} \mathrm{sgn}(v_l) \tag{2.4}$$

3. Le processus de décision « dure » qui effectue l'estimation de la valeur des symboles du code. Pour chaque nœud de donnée :

$$\hat{c} = \mathrm{sgn}(u_0 + \sum_{l=1}^{i} u_l) \tag{2.5}$$

On peut noter que les étapes 1 et 3 peuvent facilement être effectuées simultanément (les mêmes messages interviennent dans les calculs). L'étape 3 nécessite une somme complète des messages entrants, ainsi qu'un calcul des signes, tandis que l'étape 1 implique le calcul de soustractions supplémentaires. Deux fonctions mathématiques sont nécessaires dans l'étape 2 pour transposer le calcul de la convolution initiale dans le domaine logarithmique de Fourier. Ces fonctions sont définies de la manière suivante :

$$f(x) = \log(\tanh(\frac{|x|}{2})) \tag{2.6}$$

$$g(x) = 2 \times \tanh^{-1}(\exp(x)) \tag{2.7}$$

Toutes ces étapes constituent la description de l'algorithme log-BP et sont décrites de manière plus complète dans [38].

2.2 Les architectures de décodeurs HC-LDPC

Contexte
Cette partie porte sur les travaux qui ont été effectués dans le cadre d'une action JEMSTIC, en collaboration avec David Declercq (ETIS, équipe signal), qui a été proposée par le laboratoire et financée par le CNRS en 2001 et 2002.

Contrairement à des solutions orientées vers la technologie ASIC, où la topologie du graphe LDPC est réalisée par un réseau figé et complexe de routage entre des éléments de calculs ([36] par exemple), nous nous sommes fixés comme contrainte de fournir une certaine flexibilité dans la structure du décodeur. Le choix du rendement du code, de son profil d'irrégularité, de sa taille et d'une manière générale de la matrice H est un degré de liberté que nous voulons offrir dans notre architecture. C'est pourquoi nous avons opté pour un modèle architectural dans lequel tous les messages véhiculés le long des branches du graphe sont stockés dans des mémoires et sont traités par un ensemble de processeurs élémentaires. Cette organisation architecturale est connue pour être relativement consommatrice de ressources (mémoire principalement) mais, par des optimisations particulières, autorise la construction d'une architecture flexible permettant de décoder une large classe de codes. C'est cette classe de code ainsi que l'architecture du décodeur adaptée qui sont présentées maintenant.

2.2.1 Le problème du caractère aléatoire des matrices

Soit un code LDPC représenté par son graphe bipartite comme illustré figure 2.3. Dans cette représentation, chaque connexion entre un nœud de donnée et un nœud de parité peut être représentée par un identifiant unique A_e qui est l'adresse, dans les mémoires correspondantes, des messages u_k et v_k propagés le long de cette connexion. Dans ce modèle, les branches du graphe sont numérotées dans l'ordre naturel de $A_e = 0$ à $A_e = N_e - 1$ (vues du coté des nœuds de données). De manière équivalente, les valeurs non nulles de la matrice de décodage H sont numérotées du haut vers le bas, dans l'ordre des colonnes d'abord.

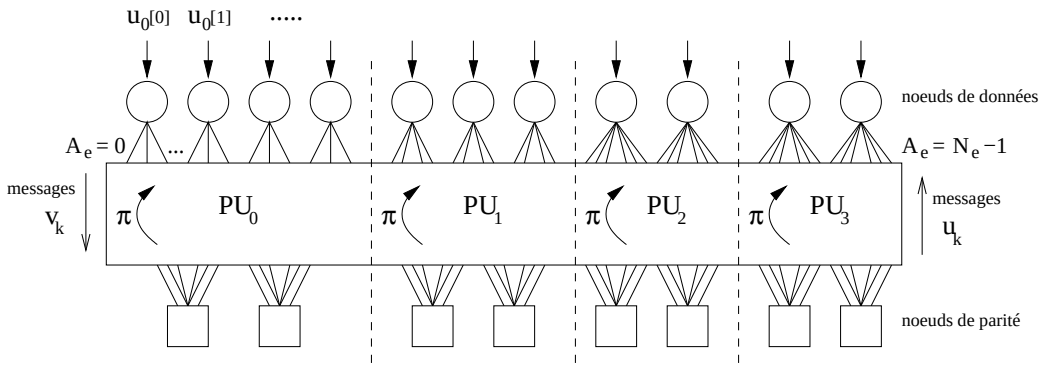


FIG. 2.3 – Partitionnement de la représentation d'un code LDPC régulier ou irrégulier sous forme de graphe. Chaque partition est associée à une une unité de calcul.

Décoder ce code LDPC revient donc à itérer les deux étapes suivantes :

1. le processus descendant (*data update*) qui calcule les messages v_k des nœuds de données vers les nœuds de parité en utilisant l'équation (2.3) et
2. le processus montant (*check update*) qui calcule les messages u_k en utilisant l'équation (2.4).

Puisque les deux types de messages sont contenus dans des mémoires, les deux étapes de décodage sont équivalentes à des lectures et écritures à des adresses particulières dans les deux types de mémoires. Plus précisément, grâce à la numérotation des branches du graphe, la première étape consiste à accéder aux mémoires dans l'ordre naturel (de 0 à N_e-1), tandis que la deuxième étape accède aux mémoires dans un ordre pseudo-aléatoire. Dès lors, n'importe quelle topologie de graphe (et n'importe quelle matrice LDPC) peut être obtenue par un unique ensemble d'entrelacement $\pi = \{\pi_n\}$ de taille N_e où π_n est la n^{ieme} connexion des nœuds de parité (la n^{ieme} place mémoire) utilisée par le processus de vérification de parité (*check update*).

Un code LDPC particulier, régulier ou irrégulier, est alors complètement défini par son ensemble de paramètres : $\{N, M, \lambda(x), \rho(x), \pi\}$. Les deux profils d'irrégularité $\lambda(x)$ et $\rho(x)$ se ramenant à deux quantités fixes $d_{v_{max}}$ et $d_{c_{max}}$ dans le cas d'un code régulier.

L'entrelaceur peut être réalisé sous la forme d'une mémoire contenant toutes les valeurs π_n. Pour les codes de très grande taille cependant, cette solution peut rapidement devenir très coûteuse en terme d'occupation mémoire. Optimiser la complexité de cet entrelaceur est alors un facteur déterminant de la qualité de la réalisation matérielle d'un décodeur pour des codes de grandes tailles. La parallélisation de tels codes est une solution possible pour réduire à la fois la durée des calculs et pour réduire l'occupation mémoire à la condition que la taille de l'entrelaceur soit constante vis-à-vis du degré de parallélisme. C'est sous cette contrainte que nous avons développé une sous-classe particulière de codes LDPC, les codes HC-LDPC (*Hardware-Constrained LDPC*), bien adaptés à la parallélisation de leur décodeurs.

La suite présente les bases de la construction de ces codes HC-LDPC ainsi que le modèle architectural du décodeur associé. Ces travaux ont été présentés dans [64] et [65].

2.2.2 Construction des codes HC-LDPC réguliers ou irréguliers

La construction d'un code HC-LDPC parallélisé en N_p partitions débute avec une matrice très fortement stucturée et diagonale par blocs $H_d = diag[H_\pi^0, H_\pi^1, \dots H_\pi^{N_p-1}]$ comme illustré figure 2.4.

La matrice H_d, de taille $(M \times N)$, doit satisfaire les conditions suivantes :

- chaque sous-matrice est strictement régulière (au sens des colonnes comme des lignes) pour la simplification de la réalisation des entrelaceurs,
- le nombre total de valeurs non nulles N_e dans la matrice H_d est uniformément réparti sur les N_p sous-matrices (toutes les matrices H_π^p possèdent le même nombre de valeurs non nulles N_e/N_p) afin d'assurer que le décodage de toutes les partitions peut se faire en un temps de calcul identique,
- la structure de chaque sous-matrice reflète le même entrelaceur $\pi = \{\pi_n\}$ afin de réduire l'occupation mémoire nécessaire au stockage de l'entrelaceur total.

Dans le cas particulier d'un code régulier, toutes les sous-matrices sont identiques. Pour les codes irréguliers, les sous-matrices H_π^p sont de différentes tailles, ont des degrés de colonnes ou de lignes différents, ces quantités dépendant des polynômes $\lambda(x)$ et $\rho(x)$.

On peut noter que dans cette méthode de construction, on suppose des matrices de décodage irrégulières en degrés des nœuds de variable mais régulières en degrés des nœuds de parité. Cette particularité n'impose toutefois pas une contrainte trop sévère puisqu'il est montré que de tels codes possèdent de bonnes performances. Cependant, la construction de codes irréguliers est aujourd'hui toujours assez difficile et fait toujours l'objet de nombreuses recherche et la troisième condition ci-dessus est quelquefois plutôt difficile à remplir. Puisque l'entrelaceur est construit de manière aléatoire mais contrainte (il doit être identique pour toutes les sous-matrices), notre technique de construction est souvent mise en échec à cause de conflits de branches. Cette difficulté doit être relativisée par le fait que la troisième condition est limitée au nombre de degrés différents dans les sous-matrices. Dans les simulations qui ont été faites, nous nous sommes limités à quatres degrés différents (voir table 2.1) et la construction de l'entrelaceur s'est faite avec succès dans 40% des cas.

Le nombre de partitions N_p défini donc le nombre maximum de degrés différents des nœuds de variable et de parité dans les codes HC-LDPC. À ce point, le code parallèle est la concaténation de N_p partitions indépendantes (N_p graphes disjoints) qui peuvent être allouées sur un ensemble de N_p unités de calcul (pour les calculs des messages) et N_p mémoires (pour le stockage des messages).

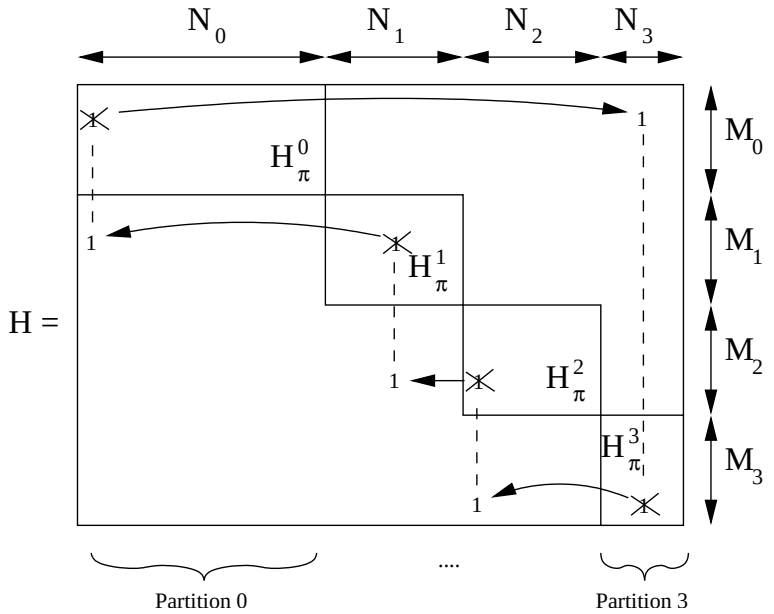


FIG. 2.4 – Exemple de la construction en deux étapes d'un code HC-LDPC parallèle : construction diagonale et déplacement des valeurs non nulles. Ici, le code est construit avec $N_p = 4$ partitions. Le code résultant est irrégulier au sens des colonnes uniquement dans cet exemple.

L'étape suivante consiste à connecter les graphes indépendants avec des branches *inter-partitions*. Ces connexions sont obtenues en déplaçant un sous-ensemble de valeurs non

Tab. 2.1 – Les meilleurs codes de rendement $R \approx 0.5$ avec $\rho(x) = x^7$ et $N_p = 8$ partitions. δ est le seuil du code calculé par évolution de densité et C est la capacité du canal.

Code	$\underline{\lambda}$	Rendement	$(\delta - C)$
Code40	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{8}x^4 + \frac{3}{8}x^{10}$	0.533	0.1754
Code41	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{8}x^4 + \frac{3}{8}x^{11}$	0.528	0.1533
Code42	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{4}x^5 + \frac{1}{4}x^{13}$	0.533	0.1638
Code43	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{4}x^5 + \frac{1}{4}x^{14}$	0.531	0.1478
Code44	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{4}x^5 + \frac{1}{4}x^{15}$	0.529	0.1535
Code45	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{4}x^5 + \frac{1}{4}x^{16}$	0.528	0.1682
Code46	$\lambda(x) = \frac{1}{4}x + \frac{1}{4}x^2 + \frac{1}{4}x^5 + \frac{1}{4}x^{17}$	0.526	0.1832

nulles depuis les matrices diagonales H_π^p vers les autres partitions. Afin de ne pas modifier les degrés des lignes et colonnes (les degrés des nœuds de donnée et de parité), les déplacements des branches sont des déplacements circulaires obtenus en associant à un sous-ensemble des valeurs de l'entrelaceur un décalage de partition circulaire s. Pour une valeur particulière de l'entrelaceur π_n, dans la partition p et son décalage associé s_n, cette branche est déplacée dans la partition $(p + s_n) mod N_p$. La figure 2.4 donne quelques exemples de tels déplacements. Par construction, la matrice résultante possède exactement les mêmes profils d'irrégularités $\lambda(x)$ et $\rho(x)$. La performance du code ainsi obtenu dépend bien entendu du nombre de branches qui ont été déplacées. Cette proportion a été fixée à $\frac{N_p - 1}{N_p}$ pour chacune des partitions de telle sorte que la distribution totale des branches dans la matrice H soit aussi uniforme que possible.

Cette méthode de construction s'applique bien entendu à des matrices partitionnées en un nombre quelconque N_p de partitions. De plus, des codes réguliers ou irréguliers peuvent être construits de cette façon. Dans le cas des codes irréguliers cependant, il faut noter que n'importe quel déplacement de branche n'est pas toujours autorisé. En effet, il n'existe pas systématiquement un entrelaceur unique *générant* toutes les sous-matrices H_π^p avec des déplacements quelconques. D'autre part, les profils d'irrégularités sont limités au nombre de partitions, N_p fixant le nombre maximum de degrés différents des nœuds de données. Dans ce travail, nous nous sommes volontairement limités à des matrices irrégulières dans les degrés des colonnes uniquement. La table 2.1 montre toutefois que ces codes, de rendement proche de 1/2, peuvent obtenir de très bonnes performances. L'étude des performances des codes de la table 2.1 a été menée en estimant les seuils de ces codes par la méthode de l'évolution de densité, en précision infinie, et pour des tailles de codes infinies. Une étude par évolution de densité en précision finie a également été menée au cours de cette étude [43].

2.2.3 La parallélisation des matrices HC-LDPC

La solution architecturale que nous avons proposée consiste à traiter par des processeurs élémentaires tous les messages du graphe qui sont stockés dans des mémoires

distinctes. Ce modèle architectural réalise un ordonnancement séquentiel de toutes les opérations sur les nœuds et est adaptée à n'importe quel algorithme de passage de messages itératif. L'ordonnancement séquentiel des opérations est connu pour être consommateur de mémoire mais conduit en revanche à des chemins de données simples et facilement optimisables par l'insertion de multiples niveaux de pipeline. La topologie pseudo aléatoire des matrices est réalisée grâce à des unités *ad hoc* de calcul des adresses mémoire et à des réseaux de routage entre les mémoires et des processeurs élémentaires. La figure 2.5 illustre ce modèle avec un seul processeur.

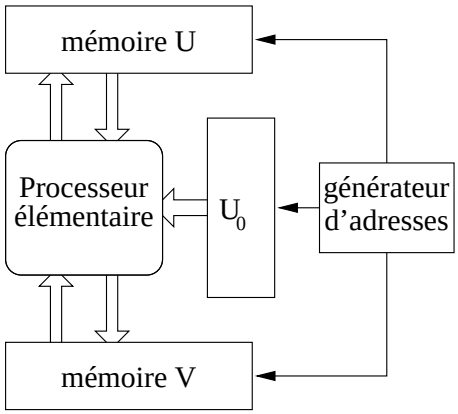


FIG. 2.5 – Des mémoires distinctes contiennent les messages u et v. Un processeur élémentaire calcule alternativement les messages auxquels il accède en mémoire grâce à une unité d'adressage. Une mémoire supplémentaire est nécessaire pour initialiser l'algorithme avec les rapports de vraisemblance.

Dans ce modèle architectural, la complexité matérielle (en quantité de mémoire) et la durée du décodage sont principalement proportionnelles au nombre de valeurs non nulles (N_e) dans la matrice de décodage. Pour décoder des codes de grande taille, nous proposons une architecture parallèle basée sur le même ordonnancement séquentiel des calculs et sur un découpage du code en plusieurs partitions. Cette parallélisation repose sur les deux principes suivants :

- Sous certaines contraintes de synchronisation qu'il faudra assurer, la durée totale de décodage est divisée par le nombre de processeurs élémentaires (le nombre de partitions) c'est à dire par le taux de parallélisme,
- Le modèle accepte les codes irréguliers en partitionnant le graphe en plusieurs partitions régulières contenant chacune des nœuds de même degrés.

À partir de la représentation du code sous sa forme de graphe telle que représentée figure 2.3, l'architecture que nous avons proposée permet une réalisation efficace d'un décodeur matériel réalisable notamment sur technologie FPGA. Les caractéristiques générales du modèle sont les suivantes :

- Le décodeur utilise une quantité de mémoire minimale pour stocker les messages et l'entrelaceur. Cette contrainte est critique lorsque nous ciblons les technologies FPGA. L'organisation mémoire ainsi que le codage des grandeurs en précision fixe ont été optimisés.

- Le modèle permet une réalisation parallèle et réduit la durée du décodage par un facteur N_p (le taux de parallélisation) sans surcoût de mémoire.
- L'ensemble de l'architecture est pipelinée de manière à réduire la latence du décodage. Une itération complète de décodage d'un graphe à N_e branches nécessite $T_p + 2 \times N_e$ cycles d'horloge (T_p est le nombre de cycles nécessaires au remplissage du pipeline).
- Une seule séquence d'entrelacement de taille minimale distribue les adresses mémoire à toutes les partitions. En conséquence, la taille mémoire utilisée pour coder la topologie du graphe est elle aussi réduite d'un facteur N_p.
- La réalisation parallèle du décodeur est applicable au codes réguliers comme aux codes irréguliers.

Cette organisation du décodeur peut de plus être utilisée avec n'importe quelle variante de l'algorithme BP. Plus précisément, ce modèle est indépendant de l'implémentation particulière des équations de décodage, celles-ci n'affectant que la structure des unités de calcul. En revanche, seul l'ordonnancement séquentiel des calculs (ordonnancement de type *flooding*) n'est possible. Pour des réalisation plus complexes du décodage comme, par exemple, l'ordonnancement proposé par Marc Fossorier dans [70], nous avons modifié en profondeur le modèle architectural. Ce nouveau modèle est à la base des travaux que nous avons effectués sur les matrices DVB-S2 et est présenté dans la partie 2.3, page 37.

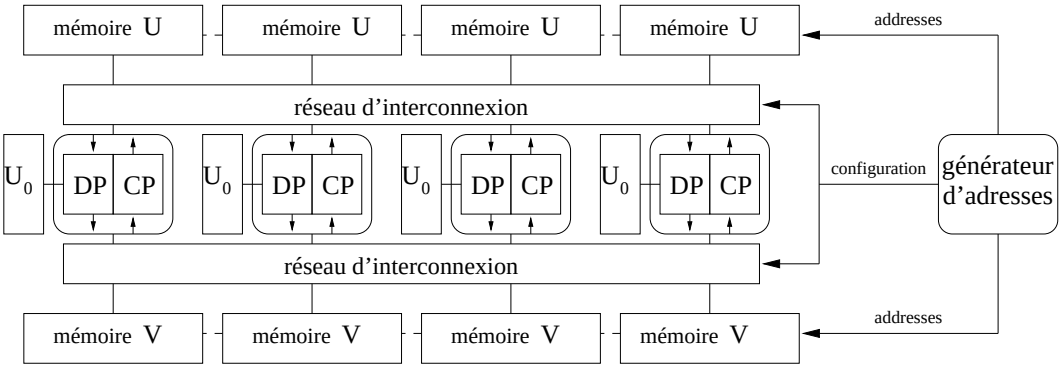


Fig. 2.6 – Modèle architectural parallèle du décodeur HC-LDPC. Le code est répartit en plusieurs partitions associées chacunes à un processeur élémentaire et à deux mémoires de messages (U et V). Les accès mémoire sont réalisés au travers de deux réseaux de routage configurés par un générateur d'adresses (la séquence d'entrelacement).

La figure 2.6 illustre le modèle parallélisé ici sur $N_p = 4$ partitions. Les processeurs élémentaires sont globalement identiques et composés de deux chemins de données indépendants, le processeur des nœuds de données (DP) et le processeur des nœuds de parité (CP). Les deux processeurs travaillent alternativement et calculent séquentiellement les équations 2.3 et 2.4. Les branches inter-partitions sont créées par les réseaux d'interconnexion qui sont contrôlés par les décalages associés à chaque branche. Un générateur d'adresse unique délivre les adresses dans les mémoires U et V, les configurations des réseaux et les signaux de contrôle nécessaires. Dans le cas des codes irréguliers, les processeurs DP travaillent sur des partitions associées à des nœuds de données à degrés variables. Des mémoires FIFO sont donc insérées dans les chemins de données afin d'har-

moniser globalement les latences de lecture et d'écriture sur l'ensemble de l'architecture. Si le code est régulier dans sa distribution des degrés des nœuds de parité, les structures des processeurs CP sont toutes identiques. La figure 2.7 décrit les structures des deux chemins de données.

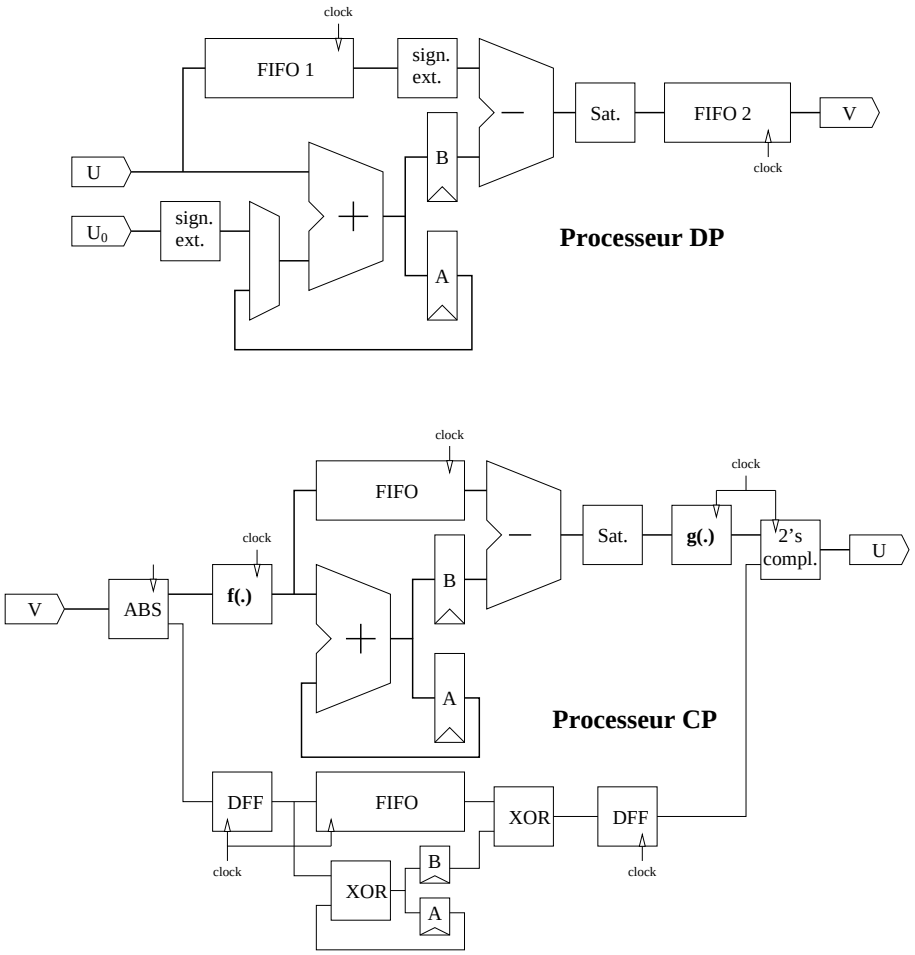


FIG. 2.7 – Architectures des deux chemins de données du processeur élémentaire : nœuds de données (en haut) et nœuds de parité (en bas). Dans la structure du processeur DP, la mémoire FIFO 2 permet d'équilibrer toutes les latences des processeurs de degrés différents.

Le processus complet de décodage s'effectue de la manière suivante :

1. une somme complète des messages entrants est effectuée dans l'accumulateur A : $A = u_0 + \sum_{k=1}^{i} u_k$ (processeur DP) et $A = \sum_{k=1}^{j} f(|v_k|)$ (processeur CP). La dernière accumulation est faite dans le registre B puis :
2. les messages sortants sont calculés avec des soustractions partielles avec le registre B : $v_l = B - u_l$ (processeur DP) et $|u_l| = g(B - f(|v_l|))$ (processeur CP).

Les deux chemins de données sont pipelinés et conduisent à des lectures et écritures en mémoire à chaque cycle d'horloge. Les latences d'écriture d'un processeur DP de degré d_v et d'un processeur CP de degré d_c sont respectivement $L_{DP} = d_v + 1$ et $L_{CP} = d_c + 5$.

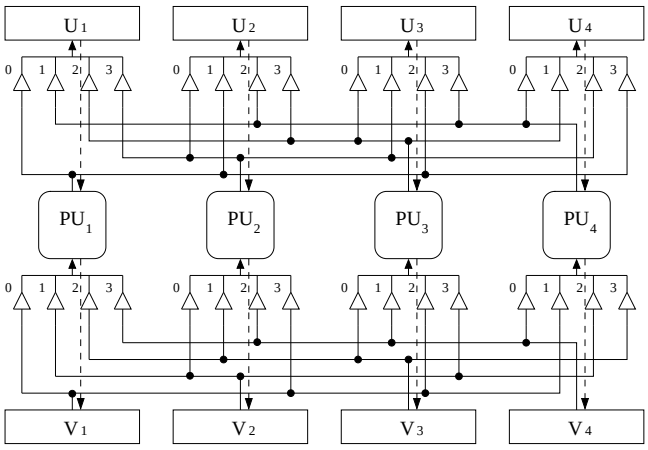


Fig. 2.8 – Exemple de réalisation des réseaux d'interconnexion (ici à 4 processeurs) à partir de buffers à haute impédance. Les deux réseaux identiques mais symétriques implantent deux registres à décalage circulaire. Les flèches en pointillés représentent les liens directs utilisés dans la phase de calcul des nœuds de données.

Toutes les écritures des processeurs DP peuvent être synchronisées par l'insertion des mémoires FIFO de profondeur $(d_{v_{max}} - d_v)$ où $d_{v_{max}}$ est le degré maximum des nœuds de données dans le graphe. De cette manière, la condition de synchronisation des calculs est assurée, les processeurs travaillent simultanément et la durée de décodage totale est divisée par le taux de parallélisme.

Réseaux d'interconnexion

Dans la construction des codes HC-LDPC, le déplacement d'une branche depuis une partition locale vers une connexion inter-partition est équivalente à trouver un chemin depuis les mémoires v vers les processeurs et des processeurs vers les mémoires u. Ces chemins sont réalisés par deux réseaux d'interconnexion identiques configurés par la partie déplacement du générateur d'adresses. Les branches non déplacées sont considérées comme des accès "locaux" entre les processeurs et leurs mémoires u et v locales. Lorsqu'une branche déplacée est traitée, les réseaux sont configurés par le générateur d'adresses et tous les accès des processeurs vers les mémoires se font de manière identique. Les réseaux sont utilisés uniquement pendant la phase de calcul des nœuds de parité (mémoire v vers mémoire u) et fonctionnent comme des registres à décalage circulaires. Si les mémoires u et v possèdent deux ports d'accès (mémoires double port), les deux réseaux peuvent facilement être réalisés grâce à des buffers à haute impédance (figure 2.8). Ce type de réalisation est d'ailleurs bien adapté aux cibles FPGA (tant pour la disponibilité de mémoire double port que de buffers *tri-state* d'ailleurs).

Le calcul des adresses est fait par une machine à états associée à une mémoire contenant la séquence d'entrelacement et les déplacements des branches. Pendant la première phase de décodage (celle du calcul des nœuds de données) les processeurs accèdent aux données dans l'ordre naturel et les adresses sont fournies par un simple comptage binaire. Les adresses ainsi que les signaux de contrôle des réseaux d'interconnexion sont

obtenus par lecture dans la table d'entrelacement lors de la deuxième phase uniquement (calcul des nœuds de parité). La taille de la mémoire du générateur d'adresse ne dépend que du nombre de branches dans le code et du nombre de partitions. Si un code HC-LDPC contient N_e branches découpées en N_p partitions, alors chaque partition contient $\frac{N_e}{N_p}$ branches dont les adresses sont codées sur $log_2(N_p)$ bits. La mémoire d'entrelacement contient $\frac{N_e}{N_p}$ mots de $log_2(\frac{N_e}{N_p})$ bits pour les adresses et $log_2(N_p)$ bits pour les déplacements. La taille mémoire totale est donc :

$$\frac{N_e}{N_p} \times (log_2(\frac{N_e}{N_p}) + log_2(N_p)) \simeq \frac{N_e}{N_p} \times (log_2(N_e))$$

Soit exactement divisée par le nombre de partitions lorsque N_e et N_p sont des puissances de 2 (la taille mémoire se calcule évidemment en entier).

Calculs en précision finie

Un autre aspect de la réduction des ressources nécessaires à la réalisation du décodeur matériel consiste à étudier la représentation des données ([54], [72]). Il a été effectivement montré que la précision des calculs peut affecter sérieusement les performances d'un décodeur. Nous avons déterminé en particulier que, en utilisant l'algorithme de décodage BP, les courbes de décodage exhibent un palier d'erreur (*error floor*) qui dépend du nombre de bits qui sont utilisés pour le codage des parties entières et fractionnaires des messages [43]. Le nombre de bits utilisés pour la représentation entière des messages (la dynamique) a une incidence directe sur le niveau de ce palier d'erreur et le nombre de bits de la partie fractionnaire (la précision) sur le seuil du code.

Le même comportement a pu être constaté aussi bien sur les codes réguliers que sur les codes irréguliers. L'apparition de ce palier d'erreur a notamment pu être lié aux effets de quantification dans le cas d'autres codes [38]. Nous avons également étudié par une version discrète de la méthode d'évolution de densité les impacts de la réduction de précision sur les deux fonctions mathématiques $f(x)$ et $g(x)$. De manière intéressante, une modification de la base logarithmique permet de compenser en partie la perte de performance due à la réduction de précision. Par évolution de densité, nous avons déterminé les bases logarithmiques idéales pour différentes précisions, en choisissant celles qui donnent les seuils de décodage les plus bas [43]. L'effet du changement des bases de calcul des deux fonctions $f(x)$ et $g(x)$ semble se rapprocher des techniques de corrections des codes, dans notre cas par un facteur multiplicatif. Ces bases optimales dépendent malheureusement de la structure du code (le profil d'irrégularité) ainsi que de la précision de décodage, limitant ainsi la généralisation de ces conclusions. Les résultats de simulation qui sont donnés plus loin ont été obtenus avec les bases optimales et un codage en précision finie $Q4.4$ (4 bits pour la partie entière et 4 bits pour la partie fractionnaire).

2.2.4 Résultats

Les résultats de simulation qui sont donnés ici ont été obtenus sur deux codes différents en utilisant l'algorithme de décodage log-BP. Le premier est un code régulier (3,6) avec deux tailles différentes ($N = 1056$ et $N = 4128$) implanté sur 1, 2, 4, 8 et 16 processeurs

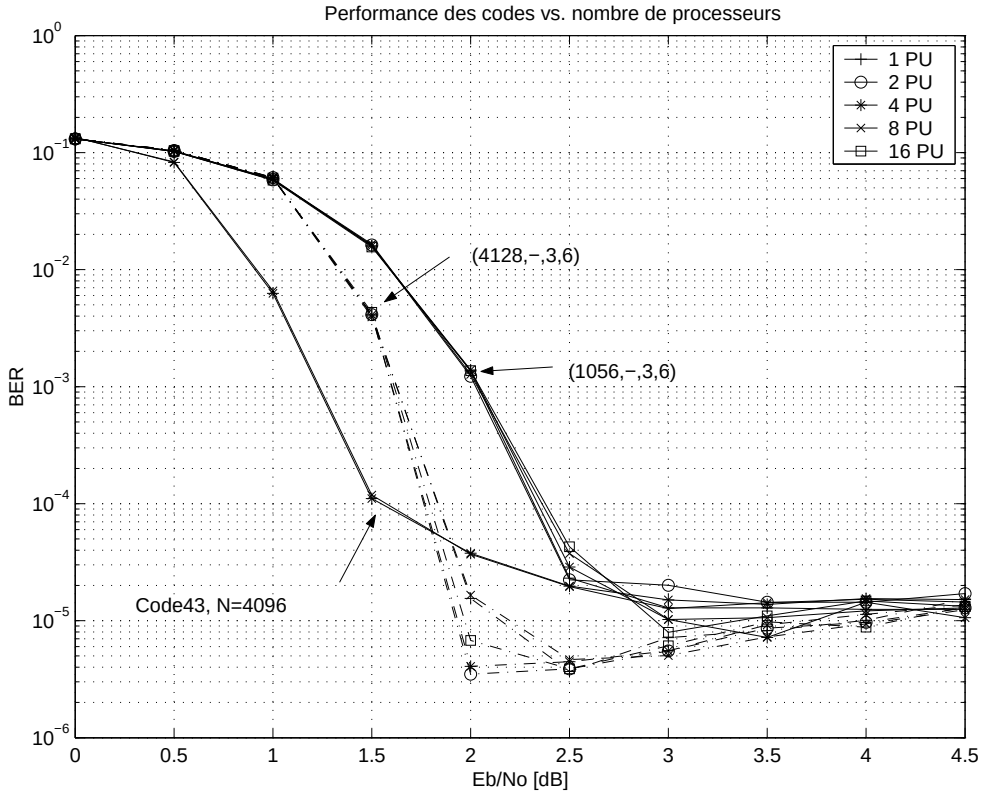


Fig. 2.9 – Performances de décodage comparées de deux codes réguliers et d'un code irrégulier en fonction du taux de parallélisation. La convergence des performances est supposée être atteinte ici à 50 itérations.

élémentaires et une base logarithmique de 1.27. Le deuxième code est un code irrégulier de taille $N = 4096$ et de rendement $R = 0.531$. Le profil d'irrégularité (en nœuds de données uniquement) de ce code ("Code43") est donné sur la table 2.1. Le profil de ce code fait apparaître 4 degrés différents, celui-ci ne peut donc être implanté que sur 4 partitions ou sur n'importe quel nombre de partition multiple de 4. La base du logarithme utilisée ici est 1.28.

La figure 2.9 montre que le nombre de partitions n'a pas d'incidence notable sur les performances de ces codes. Nous avons également comparé ces résultats avec ceux obtenus sur une matrice complètement aléatoire [64]. Les comparaisons semblent indiquer que les codes HC-LDPC se comportent de manière similaire aux codes aléatoires jusqu'à un taux de parallélisme de 24. Les paliers d'erreur qui apparaissent sur les courbes de la figure 2.9 semblent également être indépendants du nombre de processeurs. La diminution du seuil de décodage avec l'augmentation de la taille du code ainsi que les meilleures performances des codes irréguliers face aux codes réguliers sont les deux comportements attendus sur les HC-LDPC qui montrent, là encore, que l'effet de notre modèle de parallélisation n'affecte pas significativement le fonctionnement de notre décodeur.

En utilisant le fait que notre modèle de parallélisation divise le temps total de décodage par le nombre de processeurs utilisés, on peut constater un gain de performance significatif avec l'augmentation des ressources allouées (figure 2.10). En particulier, une amélioration

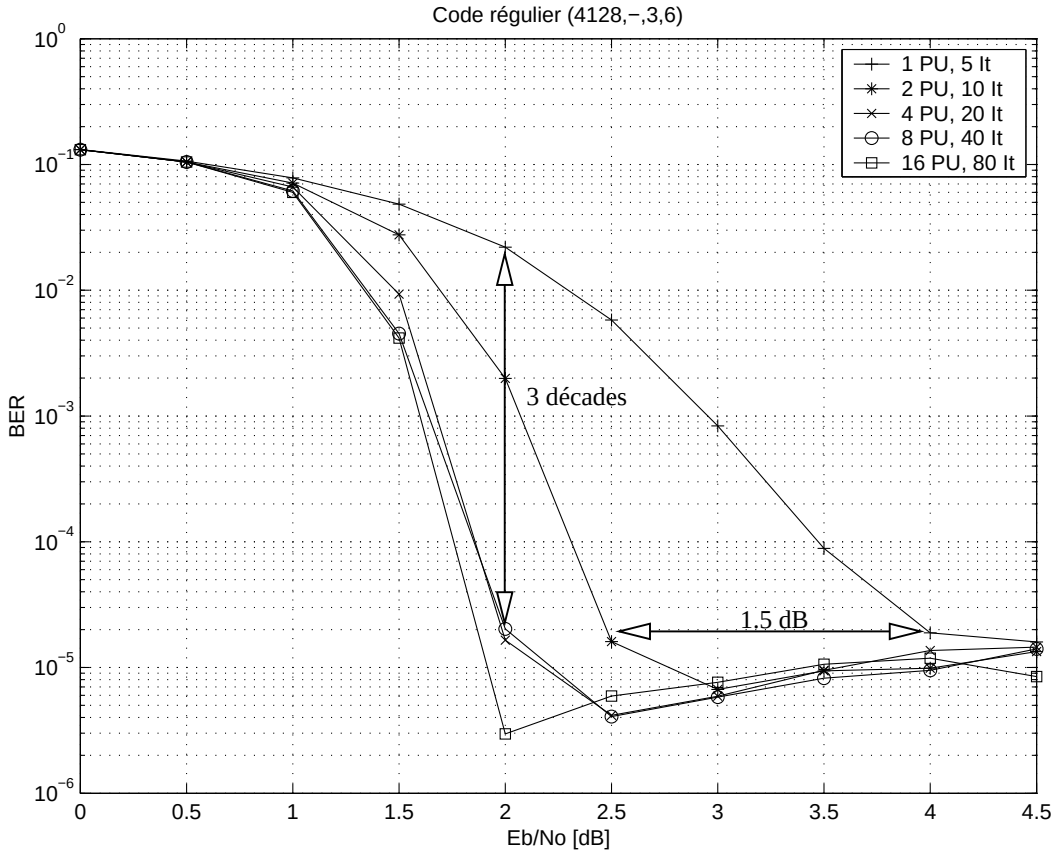


FIG. 2.10 – Performances de décodage comparées d'un code régulier (4128,-,3,6) à temps de décodage constant.

de 1.5 dB est obtenue à un taux d'erreur de 2.10^{-5} lorsque l'on passe de 1 à 2 processeurs. De le même manière, l'efficacité de décodage à 2 dB s'améliore d'environ trois décades en passant de 1 à 4 processeurs. En revanche, on constate un effet de saturation du décodeur (à partir de l'itération 40 environ), ce qui peut évidemment constituer une limite à l'amélioration des performances par l'augmentation du parallélisme. C'est pourquoi nous présentons les résultats comparés à durée de décodage identique (figure 2.11). La durée de décodage est calculée par le nombre d'itérations divisé par le nombre de processeurs élémentaires utilisés. Bien que le décodeur exhibe cette saturation des performances à 2.0 dB, la limite est atteinte plus rapidement lorsque le nombre de processeurs augmente. En particulier, 5 itérations suffisent avec 16 processeurs tandis que 15 itérations sont nécessaires avec 2 processeurs seulement.

Résultats d'implantation sur FPGA

À partir de ce modèle architectural, plusieurs décodeurs HC-LDPC ont été conçus sur une architecture cible FPGA. Le circuit qui a été utilisé est un modèle APEX 20Ke ALTERA (plate-forme Excalibur-ARM). L'ensemble de l'architecture a été développé en VHDL avec comme conséquence un certain degré de flexibilité : un faible nombre

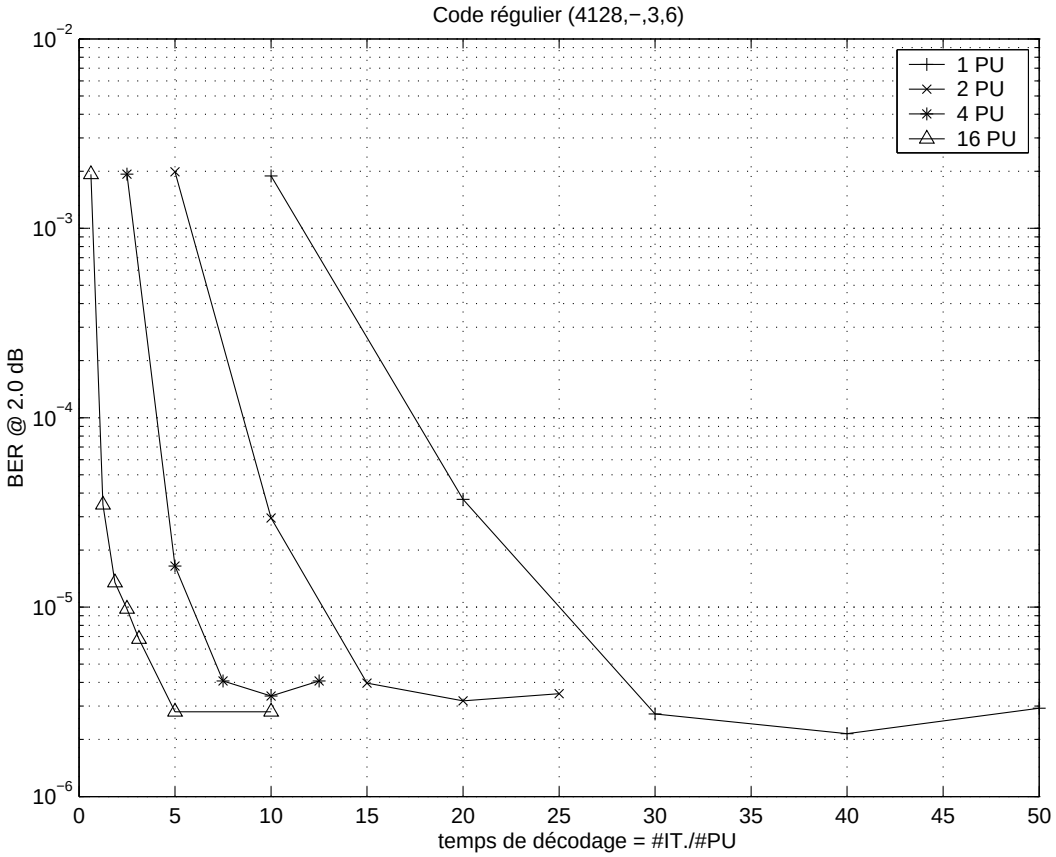


Fig. 2.11 – Taux d'erreur binaire à un rapport signal à bruit de 2.0 dB en fonction du temps de décodage. L'axe horizontal représente le temps de décodage (le nombre d'itérations divisé par le nombre de processeurs).

seulement de modifications du code source ont été nécessaires pour explorer différents codes, de différentes tailles et rendements et pour différents taux de parallélisation.

Les résultats du placement et routage d'un code irrégulier de taille $N = 2048$ et de rendement $R = 0.531$ réalisé sur 4 processeurs élémentaires exécutant l'algorithme log-BP sont les suivants : 978 cellules logiques (LC) sont nécessaires à la réalisation des 4 processeurs (2.5% d'occupation), 71 LC pour les deux réseaux d'interconnexion (8 bits), 542 LC pour la logique de contrôle et un total de 164224 bits de mémoire pour le stockage des LLR, des messages et de l'entrelaceur. L'occupation très restreinte des ressources d'interconnexion ainsi que la faible occupation mémoire de l'entrelaceur (24960 bits) permet d'imaginer la réalisation de décodeurs avec des taux de parallélisme élevés sur ce type de plate-forme.

Une itération complète de décodage nécessite $2 \times \frac{N_c}{N_p} + N_{LAT}$ cycles d'horloge où N_{LAT} représente les latences cumulées des deux chemins de données (CP et DP). La fréquence maximale de fonctionnement de ce décodeur a été calculée à 37.89 Mhz et autorise donc un débit de canal de 1Mb/s avec seulement 4 processeurs.

2.3 Cas des codes pour la norme DVB-S2

Contexte
Cette partie porte sur les travaux qui ont été effectués dans le cadre de la thèse de Arthur Segard [57] en collaboration avec David Declercq. Ces travaux ont été supportés par la société ST-microElectronics *sous la forme d'un co-financement BDI.*

Les travaux précédents ont été réutilisés et étendus pour une application aux décodeurs LDPC nécessaires dans la récente adoption de la norme DVB-S2 de radiodiffusion de programmes multimédia par satellite [45], [53]. Les caractéristiques particulières de cette chaîne de communication, notamment de la voie de réception, imposent un certain nombre de contraintes sur la réalisation matérielle des dispositifs. Les points principaux de cette deuxième génération de norme sont d'une part un objectif de performance élevé et l'ajout d'un certain degré de flexibilité dans la chaîne de communication (choix des rendements, des tailles de trames, des types de modulation). Il est évident que ces contraintes augmentent significativement la complexité des récepteurs.

La suite décrit sommairement la norme DVB-S2 et précise les contraintes de performance et de flexibilité imposées en particulier à l'étape de (dé)codage de canal LDPC (§2.3.1). L'étude architecturale d'un décodeur parallèle implantant un ordonnancement performant est ensuite décrite (§2.3.2). Le décodeur est décrit plus précisément dans la partie §2.3.3. J'aborde ensuite dans la partie §2.3.4 l'adaptation de ce décodeur particulier au matrices de la norme DVB-S2. Les résultats de simulation de ce décodeur ainsi qu'une estimation du coût technologique qui a été effectuée par ST-microElectronics sont également présentés. Je dresse enfin dans la partie §2.3.5 les pistes potentielles de ce travail.

2.3.1 Les spécificités de la norme DVB-S2

La chaîne globale de communication telle que préconisée dans la norme DVB-S2 est décrite, pour la partie émission uniquement, sur la figure 2.12.

L'innovation majeure qui a été introduite dans cette évolution de norme de radiodiffusion consiste à prévoir la possibilité de modifier les caractéristiques des parties émission et réception en fonction de critères qui dépendent à la fois de l'application et de la qualité de la transmission. Le type de modulation (QPSK, 8PSK, xAPSK) ainsi que le rendement de codage (1/4, 1/3...5/6, 8/9, 9/10) peuvent être modifiés selon le mode de transmission : diffusion multiple (*broadcast*) ou transmission en mode point-à-point. De plus, la norme prévoit une voie de retour par transmission terrestre qui permet, dans une certaine mesure, une estimation de la qualité de la transmission et une adaptation de la modulation et des rendements en fonction des caractéristiques du canal atmosphérique (ACM : *Adaptive Coding Modulation*). Enfin, la norme DVB-S2 prévoit la possibilité de modifier les paramètres de la transmission en fonction également du type des données transmises (vidéo MPEG, voix, paquets TCP/IP, paquets ATM...).

Les matrice de (dé)codage LDPC sont imposées par la norme et correspondent à deux tailles différentes de trames : 64800 et 16200 bits.

La qualité de la transmission qui est visée correspond à mode quasiment sans erreurs

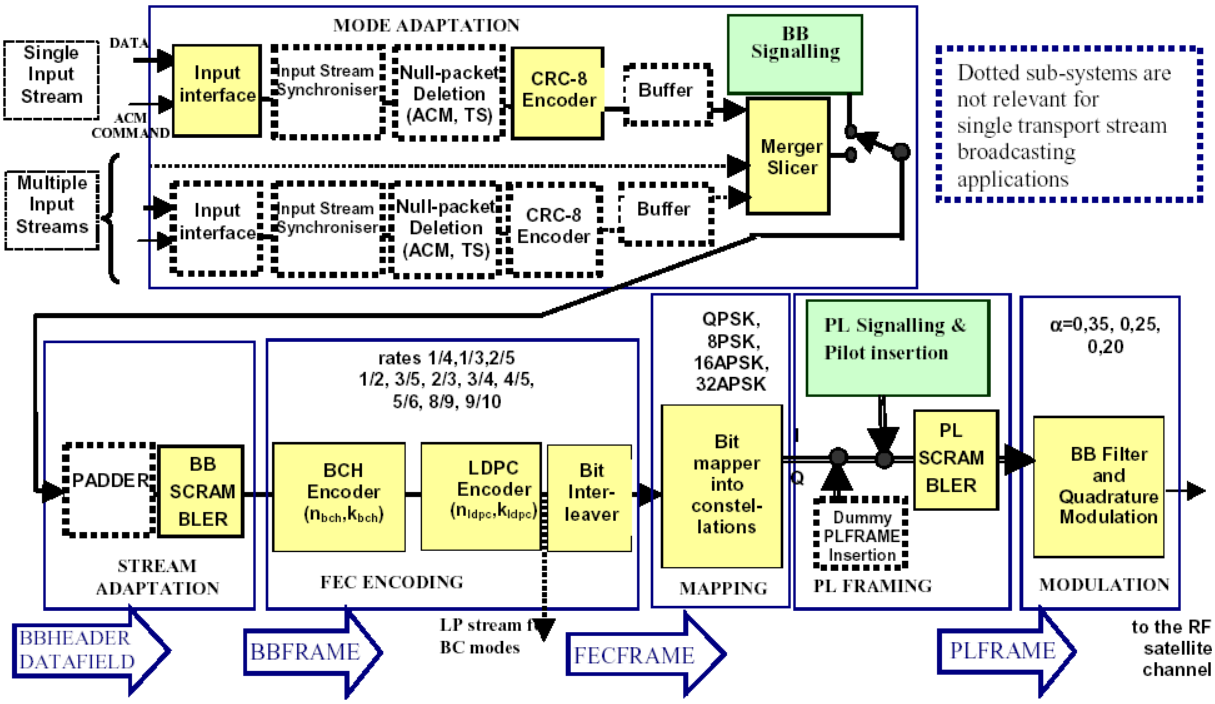


Fig. 2.12 – Chaîne d'émission DVB-S2

(QEF : *Quasi Error Free*) avec un taux d'erreurs de paquets de 10^{-7} (pour un paquet de 144 octets). Cette qualité correspond à un seul paquet de données erroné durant la transmission d'une séquence vidéo à 5 Mb/s pendant une heure.

La traduction de ces contraintes sur l'architecture du décodeur LDPC impose donc à la fois une grande performance et une grande flexibilité.

2.3.2 Modèle architectural du décodeur HSS

Pour la réalisation des décodeurs adaptés à ces codes de grande taille, nous avons choisi comme hypothèse de départ un modèle architectural *générique* simplifié à l'extrême : une architecture de type SIMD avec un ensemble de mémoires contenant les valeurs qui seront traitées au cours du décodage, un ensemble de processeurs élémentaires et un réseau d'interconnexion permettant le transport des données entre les processeurs et les mémoires.

Les taux de parallélisation que nous visons sont élevés (plusieurs centaines) et la structure du réseau d'interconnexion ne pourra pas être quelconque. En effet, réaliser sur silicium un réseau entièrement connecté (*crossbar*) de taille 400×400 sur des bus de données de plusieurs bits (typiquement 4 à 6 comme dans les décodeurs du marché) est totalement irréalisable du point de vue technologique. La surface de routage nécessaire serait de plusieurs ordres de grandeur supérieure à la surface des unités de traitement et de mémorisation. De plus, les délais imposés par ce type de réseau limiteraient fortement la performance de l'architecture. C'est pourquoi nous avons choisi d'envisager la conception de cette architecture à partir d'un réseau de type *barrel shifter* unidirectionnel. La figure

2.13 illustre le modèle architectural minimal que nous nous sommes imposé.

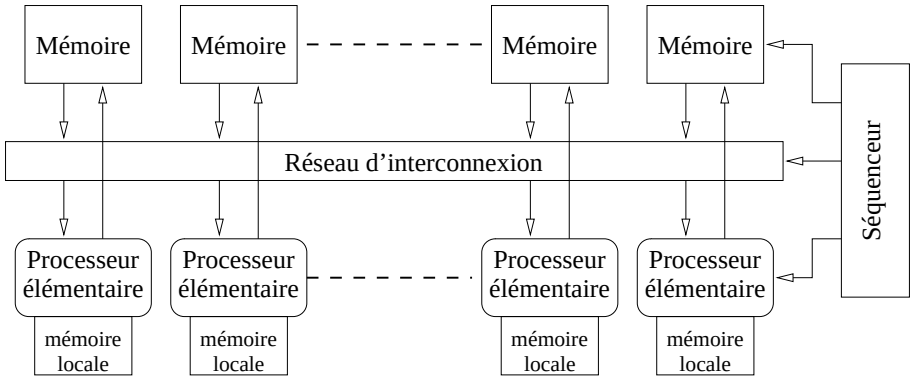


FIG. 2.13 – Modèle architectural SIMD parallèle du décodeur. Le code est découpé en plusieurs partitions. Chaque partition est prise en charge par un processeur élémentaire, doté de sa propre mémoire locale. Les processeurs accèdent, en lecture uniquement, aux mémoires distribuées contenant les messages au travers d'un réseau unique d'interconnexion.

Les travaux que nous avons menés ont permis de déployer sur cette architecture minimale une version parallèle d'un algorithme de décodage efficace utilisant l'ordonnancement brouillé des équations de décodage. Les contraintes architecturales que nous nous sommes fixées ont imposé de nombreuses optimisations et compromis qui sont décrits dans la suite de cette partie.

L'ordonnancement *Horizontal Shuffled*

L'un des facteurs d'amélioration des performances d'un décodeur LDPC consiste à optimiser l'ordre dans lequel sont effectuées les opérations de décodage sur les messages du graphe. L'algorithme utilisant l'ordonnancement brouillé (*Shuffled*) [70] permet une accélération de la convergence du décodage de manière significative. Cet ordonnancement consiste, d'une certaine manière, à augmenter la fréquence de mise à jour des messages en exécutant simultanément les équations associées aux nœuds de données et de parité. Cet ordonnancement est illustré sur la figure 2.14.

En organisant les calculs de cette façon, la fréquence de mise à jour des messages extrinsèques v_i est multipliée par le degré de connexion des nœuds de données ausquels ils sont liés (exactement par $d_v - 1$). Les informations se propagent plus rapidement dans le graphe et la convergence du décodage est atteinte plus rapidement. En effet, si on note l l'itération au cours de laquelle un message est calculé, on constate que dans l'ordonnancement classique, tous les messages sont mis à jour à partir des informations de l'itération $(l-1)$. Au contraire, avec l'ordonnancement brouillé, un message u_i^l peut être recalculé à partir d'informations datant de l'itération l.

Réduction de la complexité L'augmentation de la fréquence de mise à jour des messages s'accompagne naturellement d'une augmentation, dans les mêmes proportions, du nombre de calcul à effectuer. Si on note C_c (respectivement C_v) la complexité (en nombre

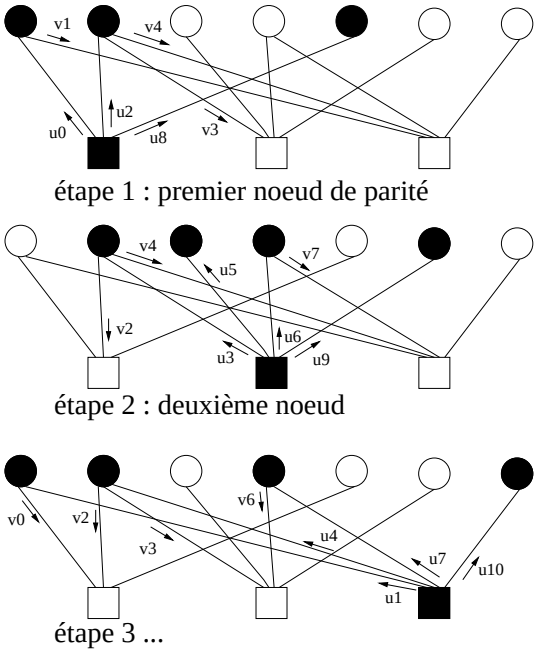


Fig. 2.14 – Principe de mise à jour des messages dans l'ordonnancement brouillé.

d'opérations) des calculs nécessaires à la mise à jour d'un nœud de parité (respectivement d'un nœud de donnée), alors la complexité équivalente d'une itération de l'algorithme de décodage "classique" est :

$$C_{FS} = M \times C_c + N \times C_v \tag{2.8}$$

Dans le cas d'un ordonnancement brouillé :

$$C_{HSS} = M \times (C_c + d_c \times C_v) = M \times C_c + N \times C_v + (d_v - 1) \times N \times C_v \tag{2.9}$$

On a donc une augmentation significative de la complexité, d'une quantité de l'ordre de Nd_vC_v, ce qui peut devenir très rapidement prohibitif, particulièrement dans le cas des codes de grande taille. Cette augmentation du nombre de calculs s'explique, dans l'équation (2.17) par exemple, par le nombre de sommes nécessaires à chaque mise à jour d'un message $u - i$.

Les calculs ont donc été réorganisés de manière à éviter ces termes additionnels. En particulier, l'étape de mise à jour des nœuds de données est ré-exprimée :

$$S_n = LLR_n + \sum_{k=1, k \neq i}^{d_v(n)} u_k(n) \tag{2.10}$$

Où $d_v(n)$ est le degré du nœud de donnée n et $u_k(n)$ les messages entrants dans ce nœud. En conséquence :

$$v_i = S_n - u_i \tag{2.11}$$

D'où, en introduisant la notion d'itération (l), les calculs deviennent :

$$v_i^{(l)} = S_n^* - u_i^{(l-1)} \tag{2.12}$$

$$u_i^{(l)} = \bigoplus_{k=1,k\neq i}^{d_c} v_k^{(l)} \tag{2.13}$$

$$S_n^{**} = u_i^{(l)} + v_i^{(l)} \tag{2.14}$$

Dans les équations ci-dessus, les notations S_n^* et S_n^{**} représentent les valeurs respectives de la somme S_n avant (S_n^*) et après la mise à jour (S_n^{**}) suite au calcul, par le processeur concerné, des branches considérées. Il est important de noter que les mises à jour de cette quantité s'effectuent à une fréquence plus grande – très exactement $(d_v - 1)$ fois plus souvent – que celle des itérations (l).

Cette transformation des équations de décodage possède l'avantage de diminuer le nombre de calculs à effectuer. En revanche, cette amélioration s'accompagne de plusieurs conséquences sur l'architecture du processeur élémentaire de calcul, sur la taille des bus de données qui véhiculent les messages, le réseau d'interconnexion et la synchronisation de l'ensemble. Ces points sont discutés plus loin.

Impact du degré de parallélisation sur la convergence Il a été montré que l'ordonnancement brouillé permet une accélération de la convergence du décodage en anticipant la prise en compte des mises à jour de messages dans les calculs. Cette analyse mérite cependant d'être pondérée par deux aspects : les caractéristiques du canal et l'effet de la parallélisation. En premier lieu, il faut tenir compte du fait que lorsque les messages traités sont très fortement bruités, l'interdépendance des calculs peut entraîner la propagation d'erreurs dans le graphe et conduire à une convergence vers un mauvais mot de code.

D'autre part, lorsque les calculs sont effectués en parallèle, certaines équations des nœuds de parité seront calculées indépendamment (par des processeurs différents fonctionnant simultanément) même s'il existe une dépendance fonctionnelle par l'un des nœuds de données. Le degré de parallélisation des calculs pourrait donc conduire à limiter l'intérêt de ce type d'ordonnancement et ce, de manière de plus en plus forte avec l'augmentation du taux de parallélisme. Nous n'avons pas eu l'occasion d'étudier de façon précise l'impact de la parallélisation sur la convergence de l'ordonnancement HSS. Une étude dans ce sens constituerait une des perspectives de notre travail. Les résultats que nous avons obtenus (page 45) montrent cependant que même avec un taux de parallélisme élevé (360 sur les matrices DVB-S2) nous accélérons la convergence du décodage d'un facteur 2 environ.

Organisation des données dans l'architecture La présence d'un réseau d'interconnexion unique et unidirectionnel dans l'architecture pose un problème évident du placement des données dans les mémoires de l'architecture. En effet, si les processeurs peuvent lire dans toutes les mémoires distribuées, ils ne peuvent en revanche écrire que dans la mémoire qui leur est associée (celle de leur propre partition). Les données vont donc subir, au cours du décodage, des déplacements au gré des commandes envoyées au réseau d'interconnexion.

Ces déplacements de données n'ont pas de conséquence majeure lorsque l'algorithme de décodage peut se découper en deux phases (l'algorithme "par vague" utilisé dans la partie §2.2 fonctionne selon ce principe). Dans le cas d'un ordonnancement brouillé, qui ne se réalise qu'en une seule passe, il sera impossible d'assurer que les données écrites (mises à jour) sont stockées aux mêmes emplacements que les données lues (à mettre à jour). La solution qui a été mise en œuvre consiste à assurer qu'à la fin d'une itération[2] les données seront disponibles dans les mêmes mémoires qu'au début de l'itération. C'est la représentation parallèle de la matrice et, plus particulièrement, l'organisation des commandes du réseau d'interconnexion qui assurera ce placement pseudo-statique des données. En cous d'itération en revanche, les valeurs S_i seront déplacées de partition en partition.

2.3.3 Architecture du décodeur

L'architecture complète d'un décodeur LDPC parallèle utilisant l'ordonnancement HSS et l'algorithme *MinSum* a été développée au cours de ce travail. Cette partie décrit les points essentiels de l'architecture. La figure 2.15 illustre l'architecture globale. La structure du processeur élémentaire est illustrée figure 2.16.

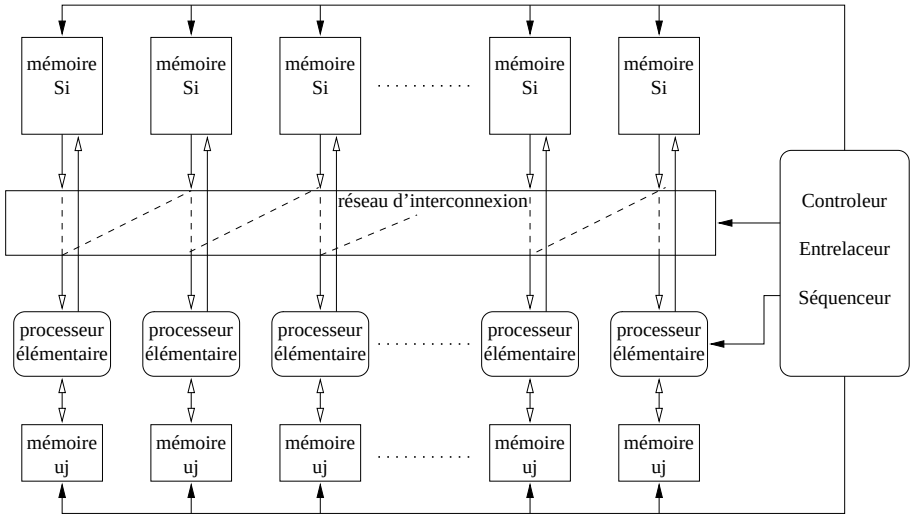


Fig. 2.15 – Structure de l'architecture globale du décodeur LDPC HSS.

L'algorithme *MinSum* et la structure du processeur

Une des variantes de l'algorithme de décodage à propagation de croyance est la version dite *MinSum* où les fonctions hyperboliques sont approchées par des calculs simples de minimum.

Dans l'algorithme *MinSum*, les opérations se déroulent selon les équations suivantes :

[2]Une itération de l'algorithme *HSS* est terminée lorsque tous les processeurs élémentaires de l'architecture ont terminé les calculs sur les nœuds de parité qu'ils ont en charge.

– Étape 1 (*check update*), mise à jour des messages u_i, pour un nœud de parité de degré d_c :

$$u_i = \bigoplus_{k=1,k\neq i}^{d_c} v_k \tag{2.15}$$

avec l'opérateur $\oplus$ défini de la manière suivante :

$$\begin{aligned} |v_i| \oplus |v_j| &= \min(|v_i|, |v_j|) + \text{offset} \\ \text{sgn}(v_i \oplus v_j) &= \text{sgn}(v_i) \times \text{sgn}(v_j) \end{aligned} \tag{2.16}$$

– Étape 2 (*data update*), mise à jour des messages v_i, pour un nœud de donnée de degré d_v :

$$v_i = LLR_i + \sum_{k=1,k\neq i}^{d_v} u_k \tag{2.17}$$

– Étape de décision, calculée sur le signe de l'information *a posteriori* :

$$S_i = LLR_i + \sum_{k=1}^{d_v} u_k \tag{2.18}$$

L'utilisation de ces équations de décodage permet une intéressante optimisation de la structure du processeur de calcul élémentaire. Nous avons exploité l'équation 2.16 pour réduire de manière significative la complexité du chemin de données du processeur. En effet, le calcul de cette équation ne produit que deux quantités $|u_i|$ distinctes quelque soit le degré d_c du nœud de parité considéré. Ces valeurs étant mémorisées localement par le processeur élémentaire, nous avons réduit fortement la quantité de mémoire nécessaire. L'ensemble des d_c signes doit toutefois être calculé et mémorisé.

Le flot des données traversant le processeur élémentaire est constitué des sommes S_i attachées à chaque nœud de données. Ces sommes étant calculées sur $d_v + 1$ messages (dans le cas d'un code régulier par exemple et en tenant compte des quantités LLR), la taille du bus transportant ces valeurs est significativement plus importante que dans le cas d'un algorithme BP classique : elle est bornée par $(k + log_2(d_v + 1))$ si k est le nombre de bits utilisés pour coder les messages u. Cette augmentation de la largeur des bus a été compensée en effectuant les accès aux mémoires distribuées en deux cycles (poids faible puis poids fort). Nous utilisons donc de cette manière un réseau d'interconnexion à bus "standard" (6 bits typiquement).

L'augmentation du nombre de cycle nécessaires aux accès mémoire n'est pas pénalisante dans notre cas car une seule itération HSS est suffisante pour exécuter l'ensemble des calculs, par opposition aux deux demi-itérations de l'ordonnancement classique "par vague". De plus, le partage des transferts mémoire en deux cycles permet de résoudre le problème des accès de lecture/écriture en un cycle (dûs au pipeline du processeur). Toutes les mémoires distribuées sont organisées en deux bancs (poids fort et poids faible) fonctionnant en mode "ping-pong" et résolvent les lectures et écritures simultanées.

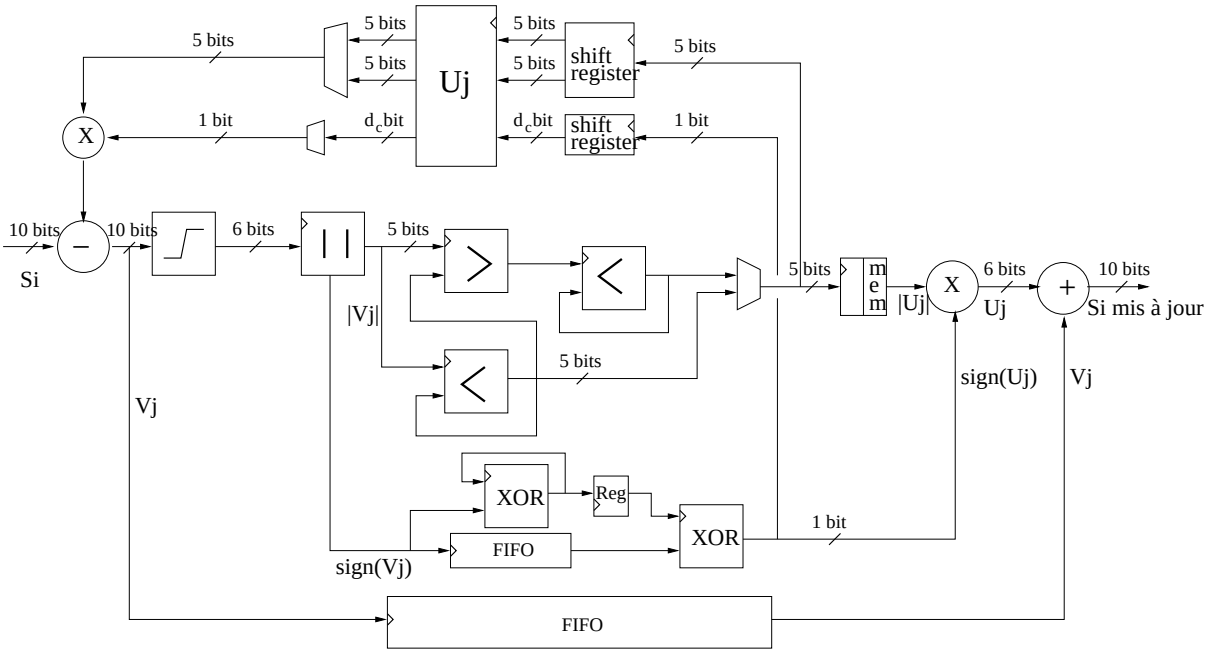


Fig. 2.16 – Architecture interne du processeur élémentaire réalisant le décodage *MinSum*.

2.3.4 Adaptation au cas des matrices DVB-S2

Adapter ce modèle architectural au cas des matrices DVB-S2 impose un taux de parallélisme de 360. De plus, pour atteindre un débit de canal élevé, chaque processeur élémentaire du décodeur effectue, à chaque cycle d'horloge, une lecture et une écriture en mémoire (la somme S_i). Une phase de calcul dans l'ordonnancement brouillé implique que chaque somme S_i est lue en mémoire, mise à jour, et ré-écrite en mémoire $d_v - 1$ fois par chaque processeur.

Étant donné le fort taux de parallélisme et le caractère pseudo-aléatoire des matrices, il est par conséquent difficile d'éviter des conflits d'accès aux mémoires. Ces conflits interviennnent lorsqu'une valeur particulière S_i est nécessaire au calcul par un nouveau processeur avant d'avoir été mise à jour en mémoire par un autre processeur lors d'un calcul précédent. Puisque les matrices de la norme DVB-S2 n'ont pas été conçues pour ce type d'ordonnancement, ce cas particulier apparaît sur la totalité des matrices.

Les deux sources de conflits d'accès aux mémoires ont été identifiées (figure 2.17). Le premier type de conflit (type ①) apparaît lorsque le processeur P_B nécessite la lecture de la valeur S_i alors que le processeur P_A n'a pas encore eu le temps de la mettre à jour en mémoire à cause de sa propre latence (qui est de l'ordre de d_c). La solution proposée dans [44] pour résoudre ce problème consiste à suspendre tous les cycles de lecture des processeurs en attendant le cycle d'écriture qui mettra à jour S_i. L'inconvénient de cette méthode réside dans le ralentissement évident du décodage (perte de cycles à chaque itération). Notre solution consiste à effectuer des permutations de l'ordre dans lequel les deux processeurs vont accéder à la valeur S_i (de manière équivalente, permuter localement sur les deux nœuds de parité concernés l'ordre des branches du graphe).

Le deuxième type de conflit (type ②) apparaît lorsque deux processeurs différents

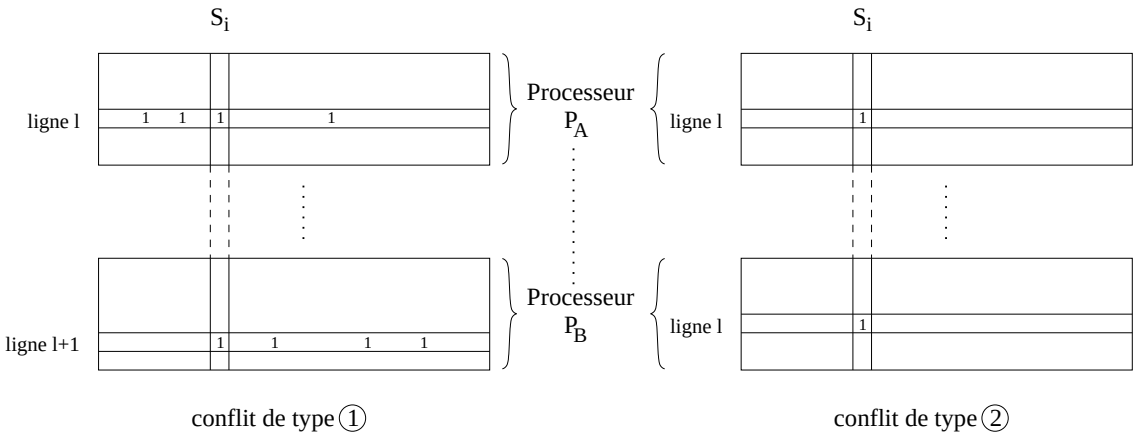


FIG. 2.17 – Les deux types de conflits d'accès aux mémoires dans le cas de l'ordonnancement HSS sur les matrices DVB-S2.

tentent de lire (et d'écrire) la même valeur S_i. Nous avons déterminé que la fréquence d'apparition de ces conflits augmente avec le taux de parallélisation. En particulier, sur les matrices de rendement 1/2, ce cas apparaît 720 fois avec un parallélisme de 40 et 2880 fois avec un parallélisme de 360. Il n'existe à notre connaissance, dans toutes les architectures proposées, pas de solution à ce problème. Dans ce cas, une organisation particulière de quantités S_i intermédiaires en mémoire a été développée et permet de résoudre ce problème sans augmentation drastique du nombre de cycles nécessaires au décodage (cette solution est présentée en détail dans [58]).

Résultats de décodage des matrices DVB-S2

Les résultats de simulation qui sont donnés figures 2.18 et 2.19 ont été obtenus avec un taux de parallélisme maximal (360 processeurs), un réseau d'interconnexion de type *barrel shifter* 360×360 et des bus de données de 6 bits. Pour une complexité des calculs équivalente, on constate l'amélioration significative de la convergence : 25 itérations seulement pour atteindre une performance équivalente à l'itération 50 de [63].

Évaluation technologique

Une estimation technologique a également été faite sur cette architecture de décodeur par la société ST-MICROELECTRONICS et permet de prédire une occupation silicium de $11mm^2$ (technologie ST). À titre de comparaison, la précédente solution DVB-S2 [63] nécessite $15,8mm^2$. La réduction de surface s'explique par la structure plus simple du processeur de décodage élémentaire et des améliorations dans la technologie des mémoires. Le décodeur proposé dans [44] nécessite beaucoup moins de surface silicium ($4,1mm^2$) mais implante un taux de parallélisme beaucoup plus faible (40 au lieu de 360).

En utilisant cette estimation technologique, avec une fréquence d'horloge de 200Mhz, le débit canal estimé est de 394 Mb/s (591 Mb/s à 300Mhz). D'autre part, pour atteindre le débit imposé par la norme DVB-S2 (135 Mb/s), une fréquence d'horloge de 69Mhz serait suffisante, ce qui permet d'envisager une réduction importante de la puissance consommée.

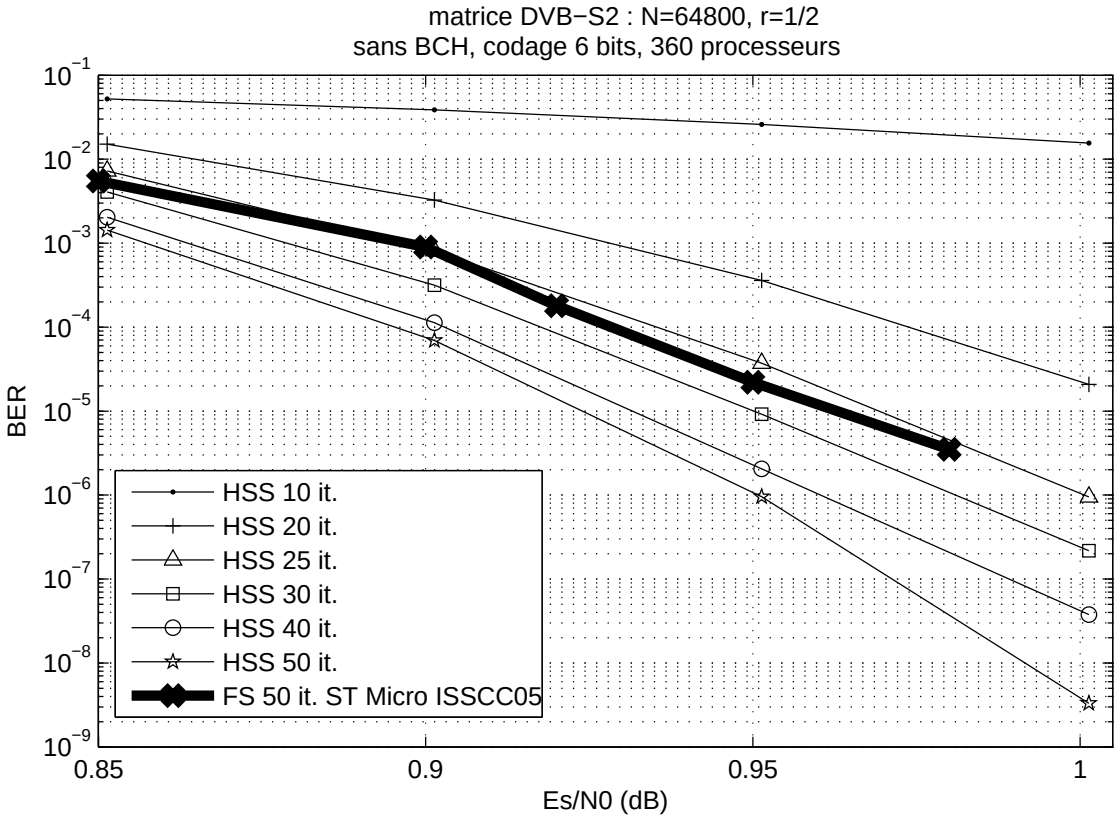


Fig. 2.18 – Comparaison des performances du décodeur DVB-S2/HSS vis-à-vis du décodeur proposé par ST-micro en 2005. Les performances tracées ici ne tiennent pas compte du décodeur BCH.

Les résultats que nous avons obtenus sur les matrices DVB-S2 font l'objet d'une communication internationale [58]. L'ensemble de cette étude fait l'objet de la thèse de Arthur Segard [57].

2.3.5 Conclusions et perspectives

La première conclusion que l'on tirer de ces travaux sur les décodeurs LDPC est que la réalisation efficace d'une architecture repose sur un certain nombre de compromis à trouver entre la simplicité des traitements et leur efficacité finale. C'est d'ailleurs sur un tel compromis (complexité/performance) que nous avons démarré nos recherches sur les décodeurs non binaires présentés un peu plus loin.

Dans le cas des décodeurs binaires en effet, l'adoption d'un ordonnancement non trivial (HSS) a permis d'atteindre des performances jusque là non atteintes sur les codes DVB-S2. En revanche, cet ordonnancement n'a pas été immédiat à implanter et son impact sur d'autres caractéristiques de l'architecture (les accès aux données en mémoire, la largeur des bus, le codage et la représentation de la matrice) a nécessité de profondes optimisations.

Le deuxième enseignement à tirer de ces expériences est qu'en introduisant un certain degré d'irrégularité dans les traitements, de nouvelles perspectives de performances

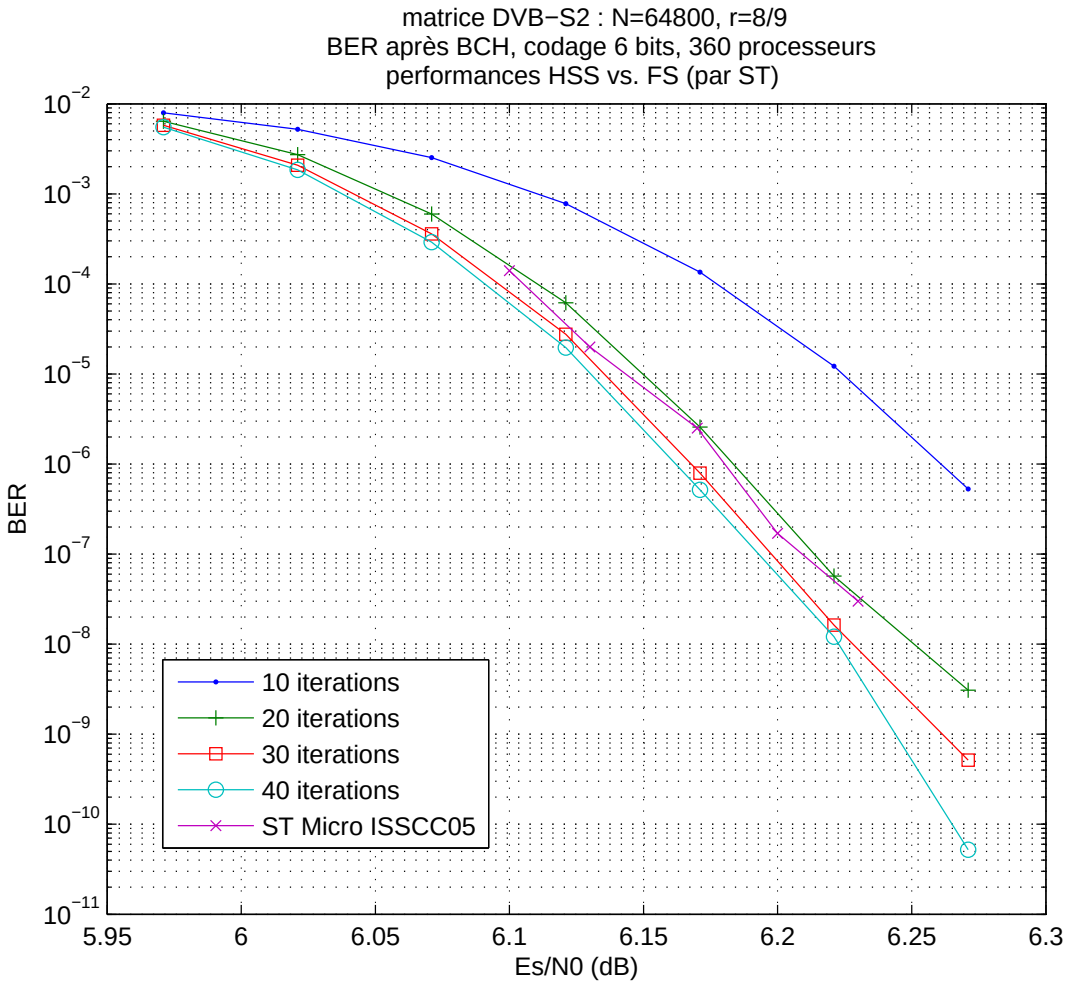


FIG. 2.19 – Comparaison des performances HSS vs. FS avec BCH tel que imposé par la norme DVB-S2.

peuvent être atteintes. L'ordonnancement HSS n'est effectivement pas celui qui apparaît de manière immédiate sur les graphes LDPC. C'est la raison pour laquelle il nous semble intéressant de développer ces travaux dans ce sens : étudier la faisabilité d'une organisation des calculs de décodage LDPC de plus en plus irrégulière, flexible et adaptative. En particulier, des ordonnancements adaptatifs méritent d'être envisagés comme le *Statistical Shuffled Scheduling* où les calculs des nœuds de parité sont déclenchés selon des règles de priorité adaptatives liées aux caractéristiques des messages : les messages les plus « sûrs » possèdent des priorités plus grandes et sont calculés en priorité. De cette manière, on peut espérer encore une accélération de la vitesse de convergence des algorithmes de décodage et, par effet de conséquence, réduire le nombre d'itérations et optimiser le débit de l'architecture.

Adopter un tel ordonnancement des calculs revient à se donner un degré de liberté supplémentaire dans l'organisation des calculs et implique donc de doter le décodeur d'une possibilité d'auto-adaptation dynamique. Les modèles architecturaux classiques de paral-

lélisme et de synchronisation doivent donc être remis en cause, voire même abandonnés. La solution que nous avons commencé à imaginer consisterait à proposer une architecture de type « ferme de processeurs » où un contrôleur global serait chargé de distribuer les calculs à un grand nombre d'unités de traitement. Chaque unité de calcul serait chargée de renvoyer au contrôle global une sorte de mesure de confiance tirée des messages calculés localement et qui servirait, lors de la prochaine itération, à réorganiser les priorités de calcul. Une telle architecture devient profondément non statique et non synchrone (dans les transferts de données) et sa conception repose probablement sur de nouveaux outils et de nouvelles méthodologies de conception.

2.4 Cas des codes LDPC dans $GF(q)$

Contexte
Cette partie porte sur les travaux qui ont été effectués et sont toujours en cours dans le cadre de la thèse de Adrian Voicila [66] en collaboration avec David Declercq. Ces travaux ont été supportés par la société ST-microElectronics *sous la forme d'un co-financement BDI.*

Dans cette partie de nos travaux, nous nous intéressons au décodage de mots de code dont les symboles ne sont plus binaires mais exprimés dans des alphabets d'ordre plus élevés. On s'intéresse ici à des corps de Galois $GF(q)$ pour q allant jusqu'à 256.

Ces codes particuliers exhibent des performances théoriques supérieures à celles obtenues dans le cas binaire. Les codes LDPC binaires sont de plus en plus envisagés dans un certain nombre de standards de correction d'erreurs comme pour le stockage magnétique, DVB-S2 (on l'a vu dans le §2.3) et dans les évolutions de la norme WiFi (IEEE 802.1x comme WIMAX en particulier). Les codes LDPC non binaires seront donc de très bons candidats des futurs standards de correction d'erreurs dès lors qu'il deviendra faisable de les réaliser matériellement. Ces codes possèdent notamment deux propriétés très intéressantes. D'une part, les valeurs h_{mn} dans les matrices H ne sont plus binaires mais q-aires, ce qui donne un degré de liberté supplémentaire dans la construction du code. D'autre part, il a été montré que ces codes sont particulièrement efficaces sur des mots de tailles plus faibles qu'en binaire (typiquement < 5000 bits) et pour des degrés de nœuds réguliers ([34], [40], [48], [56]). Ces deux dernières propriétés rendent *a priori* les décodeurs plus simples à réaliser matériellement. On a vu en effet dans la partie 2.2 que le très fort taux de parallélisme nécessaire aux codes de grande taille peut nuire aux performances des décodeurs en rendant l'accès aux données particulièrement complexe.

En revanche, les symboles du code non binaire étant considérés comme des variables aléatoires dans $GF(q)$, les messages circulant sur les branches du graphe sont des vecteurs de q valeurs, q etant l'ordre du corps (cf. équations (2.20) et (2.21) page suivante). Dès lors, une implémentation matérielle directe des algorithmes de décodage – à l'heure actuelle, principalement l'algorithme BP – conduirait à une complexité évoluant en $\mathcal{O}(q^2)$. C'est pourquoi il n'existe à ce jour pas de réalisation matérielle raisonnable d'un tel décodeur.

Nous avons donc démarré en 2005 un projet de recherche consistant à étudier les pistes possibles d'une réduction drastique de la complexité des algorithmes de décodage non binaires avant de pouvoir en envisager d'éventuelles réalisations matérielles. Ce travail est inspiré de travaux récents [41], [42] qui proposent une version de l'algorithme de décodage utilisant une représentation logarithmique des messages. Ce type de représentation des messages possède l'avantage de supprimer les opérations complexes (multiplications et divisions) ainsi qu'une plus forte robustesse aux erreurs de quantification introduits par la représentation en précision finie [55], [67]. De plus, l'algorithme proposé dans [42] – baptisé EMS pour *Extended Min-Sum* – est une version à complexité restreinte par rapport à l'algorithme Min-Sum général. La solution proposée consiste à séparer les vecteurs en deux ensembles de valeurs, le premier à n_m valeurs sur lesquelles les équations classiques du Min-Sum s'appliquent normalement, et le deuxième ensemble à $q - n_m$ valeurs pour lesquelles les équations sont très fortement simplifiées donc moins complexes mais plus

sous-optimales encore.

Lorsque $n_m \ll q$, la complexité équivalente des fonctions de mise à jour des nœuds de parité est fortement réduite (de l'odre de $\mathcal{O}(n_m \times q)$) sans dégradation importante des performances, y compris pour des codes dans des corps jusqu'à $q = 256$. Cependant, la complexité de l'algorithme EMS exprimée à la fois en nombre d'opérations et en quantité de mémoire nécessaire au stockage des messages reste toujours dépendante de l'ordre du corps et constitue toujours un obstacle à une réalisation matérielle raisonnable.

La suite de cette partie donne les éléments clés des travaux qui ont été menés à partir de l'algorithme EMS et qui ont conduit à la réduction importante de cette complexité. Nous présentons d'abord les notations utilisées pour les décodeurs non-binaires (§2.4.1) puis donnons les premiers résultats intéressants (§2.4.2).

2.4.1 Notations utilisées

Les principes généraux de fonctionnement des algorithmes de décodage LDPC dans les corps de Galois GF(q) s'appliquent de la même manière que dans le cas binaire, nous donnons ici uniquement les spécificités dans le cas non-binaire.

L'équation de vérification de parité d'un nœud de degré d_c (impliquant d_c nœuds de données) sur un symbole c_t du code est donnée par :

$$\sum_{t=1}^{d_c} h_t c_t = 0 \quad \text{dans GF(q)} \tag{2.19}$$

où h_t est la valeur non nulle correspondante de la matrice H.

Un vecteur LDR associé à une variable aléatoire $c \in GF(q)$ est :

$$\mathbf{L}(c) = [L[0] \dots L[q-1]]^T \tag{2.20}$$

avec :

$$L[i] = \log \frac{P(c = \alpha_i)}{P(c = \alpha_0)} \tag{2.21}$$

avec $P(c = \alpha_i)$ la probabilité que la variable aléatoire c prenne la valeur $\alpha_i \in GF(q)$ (la notation log est utilisée ici pour représenter le logarithme naturel). Avec ces définitions, $L[0] = 0, \quad L[i] \in \mathbb{R}$. Par exemple, le logarithme du rapport de vraisemblance (LLR) des messages en sortie du canal est noté $\mathbf{L}_{ch} = \{L_{ch}[k]_{k=0\dots q-1}\}$, qui contient q termes du type (2.21), les valeurs des probabilités $P(c = \alpha_i)$ dépendant des caractéristiques du canal.

Le comportement général d'un algorithme de décodage dans GF(q) est similaire à celui du décodage binaire et fait intervenir deux étapes principales qui sont la mise à jour des nœuds de données et la mise à jour des nœuds de parité. Une étape supplémentaire est nécessaire pour la permutation des messages (multiplication dans GF(q)) par les valeurs non nulles de la matrice H. Nous nous intéressons particulièrement à la mise à jour des nœuds de parité qui font apparaître les calculs les plus coûteux.

La figure 2.20 illustre les notations utilisées dans cette partie. Soit $\{\mathbf{V}_{p_i v}\}_{i=1..d_v}$ l'ensemble des messages entrants dans un nœud de donnée de degré d_v, et $\{\mathbf{U}_{vp_i}\}_{i=1..d_v}$ le message de sortie de ce nœud. L'index 'pv' indique que le message vient d'un nœud de

permutation à destination d'un nœud de donnée (*variable*), et '*vp*' pour la direction inverse. De manière similaire, $\{\mathbf{U}_{p_i c}\}_{i=1..d_c}$ (respectivement $\{\mathbf{V}_{cp_i}\}_{i=1..d_c}$) sont les messages à l'entrée (respectivement à la sortie) d'un nœud de parité de degré d_c. Les vecteurs $\mathbf{U}$ et $\mathbf{V}$ sont tous de taille $n_m < q$

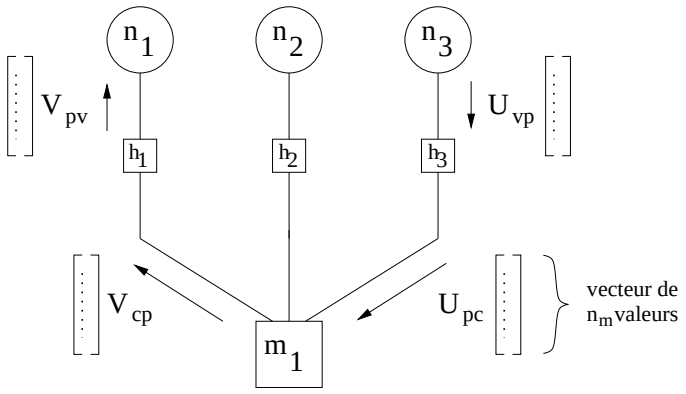


FIG. 2.20 – Notations utilisées dans le cas d'un graphe factoriel dans $GF(q)$. Des nœuds de permutation (h_i) apparaissent pour représenter la multiplication des messages dans $GF(q)$. Tous les vecteurs représentés contiennent n_m valeurs.

2.4.2 Réduction de la complexité du décodage dans $GF(q)$

L'algorithme EMS proposé dans [42] permet de réduire la complexité des équations de vérification de parité en n'utilisant que les n_m valeurs les plus significatives dans les équations de mise à jour des messages $\mathbf{V}_{cp_i}$. En revanche, les messages de sorties sont toujours composés de q valeurs et tous les messages du graphe sont stockés en mémoire sous la forme de vecteurs de q valeurs. La conséquence est que la complexité globale de cet algorithme évolue en $\mathcal{O}(n_m \times q)$. Une réduction supplémentaire de complexité est encore nécessaire, de manière à ne conserver que les n_m valeurs les plus significatives pour l'ensemble des messages. La manière exacte dont cette réduction de la taille des messages est effectuée est hors du propos de cette partie du manuscrit. Elle est décrite précisément dans [20] et exploite l'utilisation des équation Min-Sum pour approximer les $q - n_m$ messages ignorés par une valeur scalaire γ qui a été estimée par la méthode d'évolution de densité et par minimisation du seuil de décodage de l'algorithme.

L'étape de mise à jour des messages des nœuds de données se réduit à une série d'additions termes à termes de vecteurs (restreints aux n_m valeurs les plus grandes) et à une somme avec un scalaire pour les $q - n_m$ autres valeurs. L'étape de mise à jour des nœuds de parité reste le goulet d'étranglement de l'algorithme. La première technique de réduction de complexité a consisté à implémenter cette partie de manière récursive combinée à un calcul de type *forward-backward* (figure 2.21). La décomposition des calculs d'un nœud de parité de degré $d_c \geq 4$ repose sur l'existence de messages intermédiaires ($\mathbf{I}$) qui sont également codés sur n_m valeurs. Le message de sortie $\mathbf{V}_{cp_i}$ est la plupart du temps calculé à partir d'une combinaison d'un message $\mathbf{U}$ et d'un message intermédiaire $\mathbf{I}$ (détail sur la figure 2.21b).

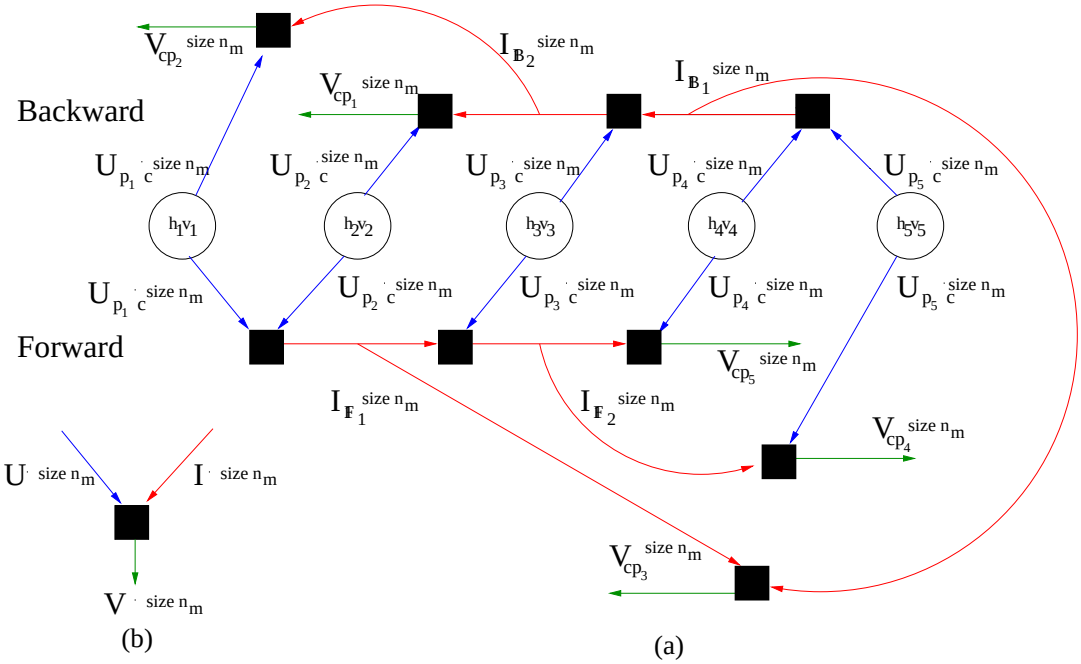


FIG. 2.21 – Structure récursive de type Forward/Backward d'un nœud de parité de degré $d_c = 5$ (a). Détail de l'étape élémentaire (b).

La principale réduction de complexité repose donc sur l'étape élémentaire $\mathbf{V}_{cp_i} = f(\mathbf{U}, \mathbf{I})$. Les hypothèse qui sont faites considèrent que tous les vecteurs $\mathbf{U}$, $\mathbf{I}$ et $\mathbf{V}$ sont mémorisés de telle manière que leurs valeurs sont en permanence triées dans un ordre décroissant. On dispose également des vecteurs index (contenant des valeurs dans GF(q)) β_U, β_I et β_V. On définit également la quantité $S(\beta_V[i])$ comme l'ensemble des combinaisons possibles qui satisfont la vérification de parité $\beta_V[i] \oplus \beta_U[j] \oplus \beta_I[p] = 0$. Avec ces notations, l'équation élémentaire devient :

$$V[i] = \max_{S(\beta_V[i])} (U[j] + I[p]) \qquad i = 0 \dots n_m - 1 \tag{2.22}$$

Lorsque l'un des index mis en jeu dans l'équation (2.22) n'est pas présent dans les vecteurs, le calcul est compensé par le scalaire γ. De cette façon, une réalisation directe de l'équation (2.22) implique une complexité de l'ordre de $O(n_m^2)$.

Nous avons été plus loin dans la simplification de l'algorithme en tentant de réduire l'équation à une complexité indépendante de l'ordre du corps. Pour celà, nous avons proposé une structure de calcul qui est composée essentiellement d'un trieur de n_m valeurs associé à une logique d'exploration intelligente des n_m^2 combinaisons. Cette structure de calcul, illustrée figure 2.22, explore les combinaisons d'une matrice virtuelle $M[i,p] = U[j] + I[p]$ contenant les n_m^2 valeurs candidates potentielles du vecteur $\mathbf{V}$. En particulier, nous nous basons sur le fait que, les vecteurs $\mathbf{U}$ et $\mathbf{I}$ étant triés par ordre décroissant, les n_m valeurs les plus grandes de M sont localisées dans la première partie antidiagonale. Enfin, l'algorithme étant séquentiel, nous faisons l'hypothèse que le trieur nécessaire à l'algorithme fonctionne comme une unité d'insertion d'une nouvelle valeur dans une liste

déjà triée, ce qui peut être réalisable matériellement en temps constant par exemple.

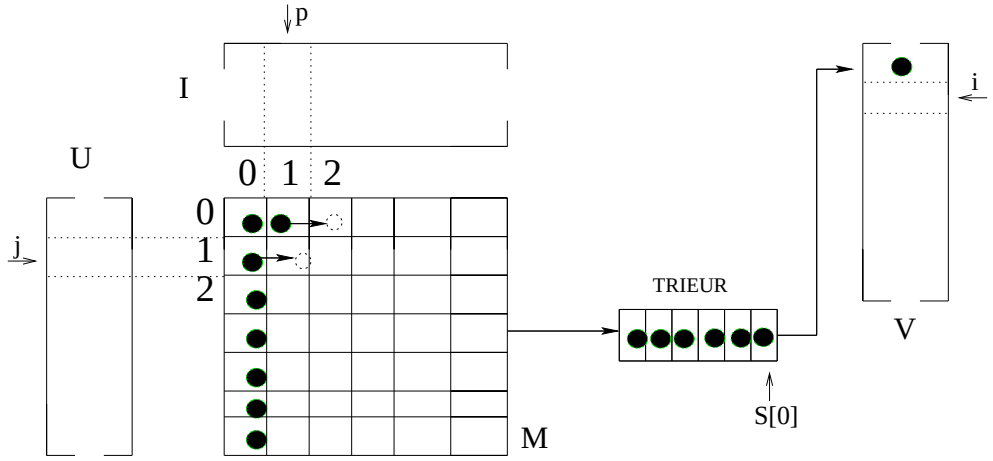


FIG. 2.22 – Structure de calcul à faible complexité pour la réalisation d'une étape élémentaire. La matrice M est purement virtuelle puisque son stockage (n_m^2 valeurs) n'est pas nécéssaire. Les quantités $M[i,p]$ sont calculées progressivement au cours de l'algorithme.

Les étapes principales de l'algorithme de calcul élémentaire deviennent donc :

1. Initialisation : les valeurs de la première colonne de M sont introduites une à une dans le trieur.
2. La valeur la plus grande est extraite du trieur.
3. Test : est-ce que l'index GF(q) de la valeur extraite du trieur existe déjà dans le vecteur **V** ?
   Oui : pas d'action,
   Non : cette valeur est insérée dans le vecteur **V**
4. Exploration : le voisin de droite de la valeur extraite du trieur – par rapport à sa position dans la matrice M – est inséré dans le trieur
5. Test d'une condition d'arrêt ou retour à l'étape 2

Cet algorithme comporte une limitation qui réside dans le fait qu'il n'est pas possible d'assurer le remplissage complet du vecteur **V** en un nombre prédictible d'itérations. En effet, à cause de la présence de configurations multiples $U[j] + I[p]$ conduisant au même index GF(q) de **V**, il n'est pas possible de stopper l'algorithme après n_m itérations seulement. Nous avons donc défini K comme étant le nombre d'itérations nécessaires pour remplir **V**. La complexité, calculée en nombre d'opérations, devient donc proportionnelle à K et non plus à q. On peut noter d'ailleurs que $K \in [n_m, \frac{n_m^2}{2}]$, ce qui correspond à notre objectif de réduction de complexité. La valeur exacte de K dépend bien entendu des vecteurs **U** et **I** et devrait être évaluée dans le pire cas si l'on veut dimensionner une éventuelle future architecture (particulièrement si on veut estimer le débit maximum possible sur le canal). Une étude statistique a cependant permis de montrer que K conserve dans la plupart des cas une valeur relativement faible. À titre d'exemple, la figure 2.23 donne la densité de probabilité de K calculée sur un code LDPC dans GF(256) avec $n_m = 32$.

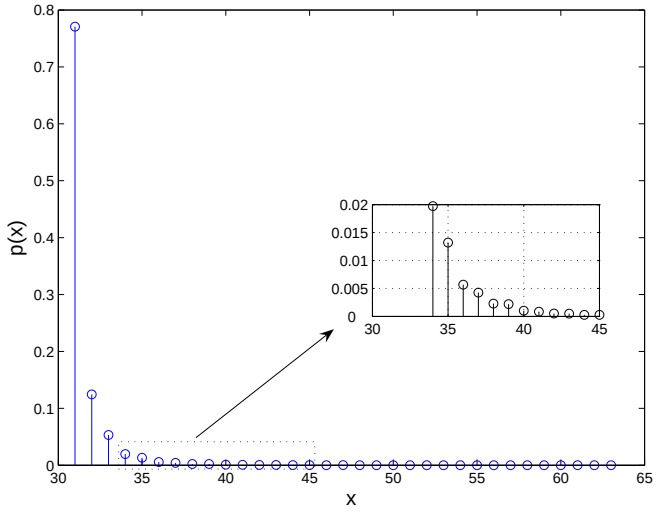


Fig. 2.23 – Densité de probabilité discrète du nombre d'itérations (K) nécessaires pour le remplissage du vecteur $\mathbf{V}$ pour un code dans GF(256), $n_m = 32$. Le point de fonctionnement du code a été pris dans la région du *waterfall*.

Plus précisément, la densité de propabilité de K possède une évolution exponentielle et décroit très rapidement. Ce qui conduit à penser que les pires cas sont suffisamment rares pour ne pas affecter significativement les performances du décodage. Cette hypothèse a été vérifié, là encore, par la méthode de l'évolution de densité et nous avons déterminé qu'une valeur $K_{\max} = 2 \times n_m$ n'affecte pas le seuil de décodage pour une large gamme de paramètres du code. Pour une valeur de $K_{\max} = 2 \times n_m$ et lorsque toutes les valeurs de $\mathbf{V}$ n'ont pas été calculées, nous avons décidé de compléter le reste du vecteur par des valeurs constantes γ. L'estimation du pire cas de la complexité de notre algorithme élémentaire se ramène donc à :

$$\mathcal{O}(K_{\max} \log_2 n_m) \Rightarrow \mathcal{O}(n_m \log_2 n_m) \tag{2.23}$$

Ce qui correspond au nombre d'opérations `max` nécessaires pour insérer $K_{\max}$ élements dans une liste triée de taille n_m.

2.4.3 Proposition d'algorithme à complexité minimale

La complexité des calculs pour la mise à jour des messages des nœuds de données et de parités sont résumées dans la table 2.2 en fonction des degrés respectifs des nœuds (d_v et d_c). Ces complexités s'appliquent aussi bien sur les codes réguliers que sur les codes irréguliers. Cependant, dans ce dernier cas, les valeurs des degrés de connectivités suivent les profils d'irrégularités. La réduction de la taille des messages de q à n_m s'applique aussi bien sur les vecteurs du canal $\mathbf{L}_{ch}$ que sur les messages extrinsèques. Les valeurs mentionnées dans la table 2.2 sont les pires cas ($K_{\max}$) et sont, la plupart du temps, des bornes maximales.

La complexité de calcul des messages $\mathbf{U}$ en sortie des nœuds de données est calculée sur une implémentation récursive, valable pour $d_v \geq 3$.

	Complexité d'un nœud de parité
Nb. max	$3(d_c-2)K_{max}\log_2 n_m$
Nb. add réelles	$3(d_c-2)(K_{max}+n_m)$
Nb. add dans GF(q)	$3(d_c-2)(K_{max}+n_m)$

	Complexité d'un nœud de données
Nb. max	$3(d_v-2)n_m\log_2(2n_m)$
Nb. add réelles	$3(d_v-2)2n_m$

TAB. 2.2 – Complexités de calcul des messages avec l'algorithme EMS et des messages de taille n_m.

On peut constater que, globalement, la complexité de cet algorithme est dominée par $\mathcal{O}(n_m \log_2(n_m))$ (si $K_{max} = 2n_m$). De plus, les complexités des deux types de nœuds sont du même ordre de grandeur, ce qui constitue une propriété intéressante en vue de la conception d'un seul et même processeur de calcul en charge des deux processus (les calculs étant en effet du même type : additions, comparaisons et tri). Enfin, les équations de complexités sont indépendantes de l'ordre du corps, ce qui était l'objectif de ce travail, et sont largement inférieures à toutes les propositions existantes à ce jour ([42], [60], [67], [68]).

L'occupation mémoire prévisible de cet algorithme est calculée pour le stockage des valeurs du canal $\mathbf{L}_{ch}$ et des valeurs des messages extrinsèques $\mathbf{U}$ et $\mathbf{I}$. Tous les messages sont des vecteurs de quantités réelles associés aux vecteurs des index β. Si les quantités réelles sont représentées en précision finie avec un codage sur N_b bits, la quantité totale de mémoire nécessaire est $n_m * N * d_v * (N_b + \log_2 q)$ qui est linéaire en n_m.

2.4.4 Paramétrage de l'algorithme et résultats expérimentaux

Puisque n_m est le paramètre principal de ce nouvel algorithme, qui fixe la complexité des calculs et la quantité de mémoire nécessaire, nous avons fait une analyse asymptotique de l'impact de ce paramètre sur le seuil de décodage (figure 2.24). Cette étude a permis notamment d'évaluer dans quelle mesure la réduction de n_m affecte les performances du decodage.

La figure 2.24 donne les seuils de décodage (exprimés en $(E_b/N_0)_{dB}$) d'un code LDPC non binaire régulier de rendement $R = 0.5$ avec les paramètres $d_c = 2$ et $d_v = 4$ pour différentes valeurs de n_m et pour les corps GF(64) et GF(256). Une telle étude, qui doit être étendue à une plus large gamme des paramètres, est une première piste pour une estimation *a priori* des paramètres d'un décodeur en fonction du compromis performances désirées vis-à-vis de la complexité.

La figure 2.25 donne les résultats expérimentaux comparatifs (après convergence) de plusieurs versions d'algorithmes de décodage d'un code régulier ($d_c = 2$ et $d_v = 4$) de rendement $R = 0.5$ et de taille équivalente $N = 848$ bits. On note $\mathrm{EMS}^{n_m}_{\mathrm{GF(q)}}$ le décodeur EMS présenté dans cette partie et fonctionnant dans le corps GF(q) avec comme paramètre n_m. Ces courbes démontrent l'intérêt d'une estimation précoce des paramètres du décodeur puisque l'on constate qu'un décodeur $\mathrm{EMS}^{32}_{\mathrm{GF(64)}}$ offre de meilleures performances, quelque

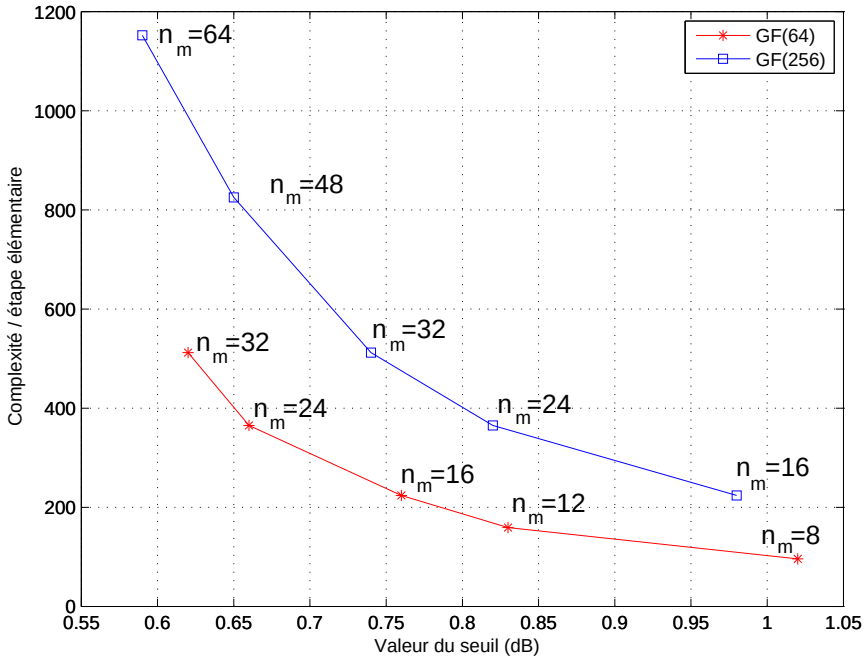


Fig. 2.24 – Paramétrage de l'algorithme : choix de la complexité de calcul de l'algorithme EMS en fonction de l'estimation du seuil de décodage.

soit le rapport signal à bruit, qu'un décodeur $\mathrm{EMS}^{32}_{\mathrm{GF}(256)}$, bien que ces deux décodeurs exhibent la même complexité (cf. figure 2.24).

La deuxième constatation porte sur la zone du palier d'erreurs où le décodeur $\mathrm{EMS}^{32}_{\mathrm{GF}(64)}$ dépasse les performances d'un décodeur BP pour le même ordre de corps. Ceci peut s'expliquer par la meilleure insensibilité aux pseudo-mots de code de l'algorithme EMS corrigé par rapport à l'algorithme BP.

2.4.5 Conclusion et perspectives

Les travaux que nous avons mené dans le cadre de la thèse de Adrian Voicila permettent, dès aujourd'hui, de dresser les premières pistes de la définition d'une architecture de décodeur LDPC non binaire. En effet, nous nous sommes intéressés à la réduction de la complexité probable de ce type de décodeur, étape préliminaire incontournable avant une étude architecturale plus poussée. Nos résultats ont conduit à une version d'algorithme à complexité minimale, qui est de plus paramétrable en fonction du compromis désiré (performance / complexité).

La construction effective d'une telle architecture constitue une perspective à court et moyen terme puisque les premiers jalons ont été posés dans le cadre du travail déjà effectué.

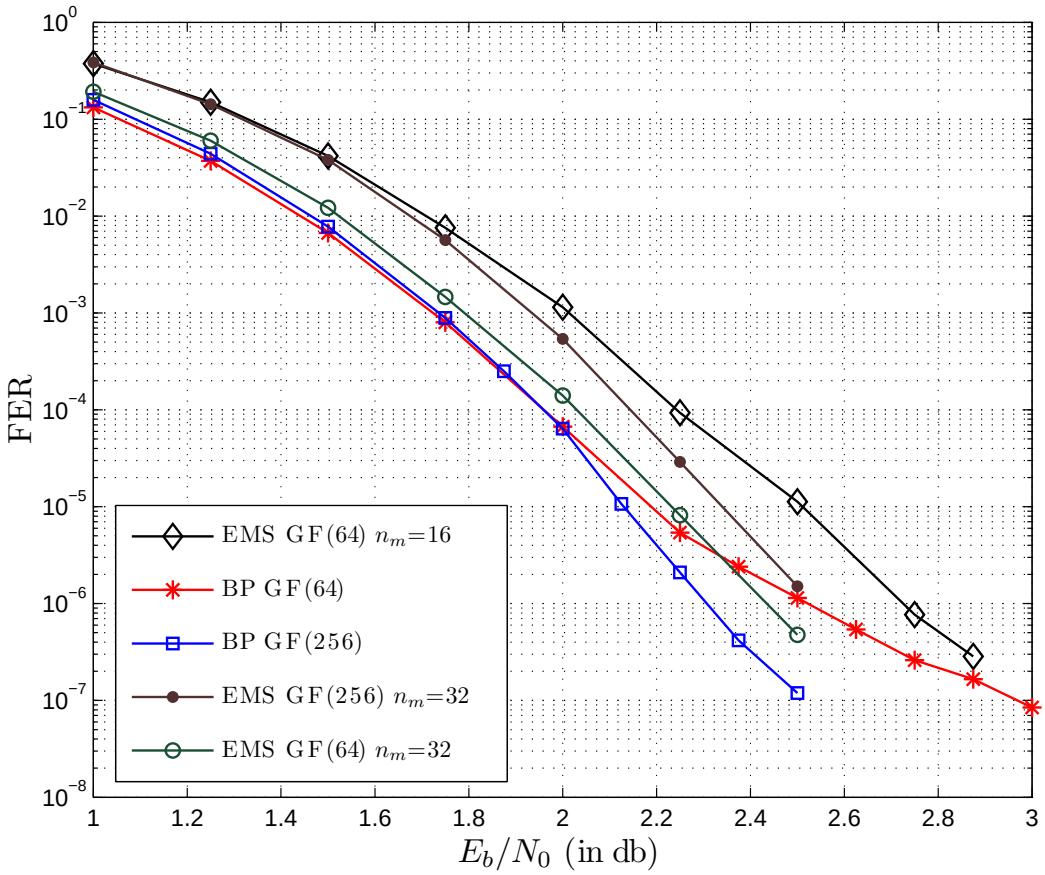


FIG. 2.25 – Comparaisons entre les version BP et EMS des algorithmes de décodage non-binaire, en implémentation virgule flottante. Les codes sont de taille 848 bits.

Références Bibliographiques

[34] A. Bennatan and David Burshtein. Design and Analysis of Nonbinary LDPC Codes for Arbitrary Discrete-Memoryless Channels. *IEEE Trans. on Information Theory*, 52(2) :549–583, February 2006.

[35] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shanon limit error-correcting codes : Turbo-codes. In *Proceedings of the ICC, Geneva, Switzerland*, 1993.

[36] Andrew J. Blanksby and Chris J. Howland. A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder. *IEEE J. Solid-State Circuits*, 37(3) :404–412, Mar. 2002.

[37] E. Boutillon, J. Castura, and F.R. Kschischang. Decoder-First Code Design. In *Proceedings of Turbo Codes*, pages 459–462, Brest, France, Sept 2000.

[38] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X-Y. Hu. Reduced-Complexity Decoding of LDPC Codes. *IEEE Trans. on Comm.*, 53(8) :1288–1299, August 2005.

[39] Yanni Chen and Dale Hocevar. A FPGA and ASIC Implementation of Rate 1/2 8088-b Irregular Low Density Parity Check Decoder. In *Proceedings of the IEEE Global Communications Conference (Globecom)*, pages 113–117, San Francisco, CA, USA, December 2003.

[40] M. Davey and D. MacKay. Low Density Parity Check Codes over GF(q). *IEEE Communication Letters*, 2 :165–167, June 1998.

[41] D. Declercq and M. Fossorier. Extended MinSum Algorithm for Decoding LDPC Codes over gf(q). In *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Adelaide, Australia, September 2005.

[42] D. Declercq and M. Fossorier. Decoding Algorithms for Nonbinary LDPC Codes over GF(q). *To appear in IEEE Trans. on Communications*, 2007.

[43] D. Declercq and F. Verdier. Optimization fo LDPC Finite Precision Belief Propagation Decoding with Discrete Density Evolution. In *Third International Symposium on Turbo Codes and Related Topics*, pages 479–482, Brest, France, Sept. 2003.

[44] John Dielissen, Andries Hekstra, and Vincent Berg. Low cost LDPC decoder for DVB-S2. In *Design, Automation and Test in Europe Conference (DATE)*, pages 130–135, Munich, Germany, March 2006.

[45] ETSI. Digital Video Broadcasting (DVB) ; Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News

Gathering and other broadband satellite applications (DVB-S2), 2005. ETSI document number EN 302 307.

[46] Frédéric Guilloud. *Architecture générique de décodeur de code LDPC.* PhD thesis, ENST, Département Électronique et Communications, Juillet 2004.

[47] Dale E. Hocevar. LDPC Code Construction with Flexible Hardware Implementation. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 2708–2712, Anchorage, Alaska, May 2003.

[48] X.-Y. Hu and E. Eleftheriou. Binary Representation of Cycle Tanner-Graph GF(2^q) Codes. In *Proc. of the IEEE International conference on Communication (ICC)*, pages 528–532, Paris, France, June 2004.

[49] S. Kim, G.E. Sobelman, and J. Moon. Parallel VLSI Architectures for a class of LDPC Codes. In *Proceedings of the IEEE International Symposium on Circuits and systems (ISCAS)*, volume II, pages 93–96, 2002.

[50] Benjamin Levine, R. Reed Taylor, and Herman Schmit. Implementation of Near Shannon Limit Error-Correting Codes Using Reconfigurable Hardware. In *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, 2000.

[51] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman. Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation. Technical Report TR-97-044, Digital Equipment Corporation System Research Center, Berkeley, CA, 1997.

[52] D. MacKay. Good Error-Correcting Codes based on Very Sparse Matrices. *IEEE Trans. Inf. Theo.*, 45(2) :399–431, March 1999.

[53] A. Morello and V. Mignone. DVB-S2 : The Second Generation Standard for Satellite Broad-band Services. *Proceedings of the IEEE*, 94(1) :210–227, January 2006.

[54] L. Ping and W.K. Leung. Decoding Low Density Parity Check Codes with Finite Quantization Bits. *IEEE Commun. Lett.*, 4(2) :62–64, Feb. 2000.

[55] L. Ping and W.K. Leung. Decoding low density parity check codes with finite quantization bits. *IEEE Commun. Lett.*, 4(2) :62–64, February 2000.

[56] C. Poulliat, M. Fossorier, and D. Declercq. Design of non binary LDPC codes using their binary image : algebraic properties. In *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, Seattle, USA, July 2006.

[57] Arthur Segard. Conception d'une architecture reconfigurable pour le décodage LDPC et son intégration dans un environnement intégré de type SOC sous contraintes temps-réel. Thèse BDI au Laboratoire ETIS - ENSEA/UCP - UMR CNRS 8051, depuis le 1er Décembre 2003.

[58] Arthur Segard, François Verdier, David Declercq, and Pascal Urard. A DVB-S2 compliant LDPC decoder integrating the Horizontal Shuffle Scheduling. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Tottori, Japan, December 2006.

[59] Anand Selvarathinam, Gwan Choi, Krishna Narayanan, Abhiram Prabhakar, and Euncheol Kim. A Massively Scaleable Decoder Architecture for Low-Density Parity-Check Codes. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume II, pages 61–64, Bangkok, Thailand, May 2003.

[60] H. Song and J.R. Cruz. Reduced-Complexity Decoding of Q-ary LDPC Codes for Magnetic Recording. *IEEE Trans. Magn.*, 39 :1081–1087, March 2003.

[61] V. Sorokine, F. Kschischang, and S. Pasupathy. Gallager Codes for CDMA Applications - Part I : Generalizations, Constructions, and Performance Bounds. *IEEE Trans. Commun.*, 48(10) :1660–1668, Oct. 2000.

[62] V. Sorokine, F. Kschischang, and S. Pasupathy. Gallager Codes for CDMA Applications - Part II : Implementations, Complexity, and System Capacity. *IEEE Trans. Commun.*, 48(11) :1818–1828, Nov. 2000.

[63] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta. A 135 Mbp/s DVB-S2 Compliant codec Based on 64800b LDPC and BCH Codes. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, February 2005.

[64] F. Verdier and D. Declercq. A LDPC Parity Check Matrix Construction for Parallel Hardware Decoding. In *Third International Symposium on Turbo Codes and Related Topics*, pages 235–238, Brest, France, Sept. 2003.

[65] F. Verdier and D. Declercq. A Low Cost Parallel Scalable FPGA Architecture for Regular and Irregular LDPC Decoding. *IEEE Transactions on Communications*, 54(7) :1215–1223, July 2006.

[66] Adrian Voicila. Etude théorique des décodeurs LDPC non binaires, proposition d'architecture et adéquation technologique. Thèse BDI au Laboratoire ETIS - ENSEA/UCP - UMR CNRS 8051, depuis le 1er Octobre 2004.

[67] H. Wymeersch, H. Steedam, and M Moeneclaey. Computational complexity and quantization effects of decoding algorithms of LDPC over GF(q). In *Proc. of ieee International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages 669–672, Montreal, Canada, May 2004.

[68] H. Wymeersch, H. Steedam, and M Moeneclaey. Log-Domain Decoding of LDPC Codes over gf(q). In *Proc. of IEEE International Conference on Communication (ICC)*, pages 772–776, Paris, France, June 2004.

[69] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam. High Thoughput Low-density Parity Check Decoder Architectures. In *Proceedings of Globecom*, pages 3019–3024, Nov. 2001.

[70] J. Zhang and M. Fossorier. Shuffled Belief Propagation Decoding. In *Proceedings of ASILOMAR'02*, pages 8–15, Pacific Grove (USA), November 2002.

[71] T. Zhang and K. Parhi. An FPGA Implementation of (3,6)-Regular Low-Density Parity-Check Code Decoder. *EURASIP Journal on Applied Signal Processing*, 2003(6), May 2003.

[72] T. Zhang, Z. Wang, and K. Parhi. On Finite Precision Implementation of Low Density Parity Check Codes Decoder. In *Proceedings of the IEEE International Conference on Circuits and Systems (ISCAS)*, Sydney, Australia, May. 2001.

[73] Hao Zhong and Tong Zhang. Design of VLSI Implementation-Oriented LDPC Codes. In *Proceedings of the IEEE Vehicular Technology Conference (VTC)*, Orlando, Florida, October 2003.

3

Les Méthodologies de Conception des Systèmes Embarqués

Sommaire

3.1 Introduction

Le deuxième axe, plus récent, de mes travaux en conception d'architectures porte sur la prise en compte de contraintes plus globales d'intégration des opérateurs de calcul au sein de chaînes complètes de traitement de l'information. L'augmentation très significative des puissances de traitement qui sont demandées aujourd'hui aux applications ainsi que l'intégration technologique toujours plus forte des architectures (sous forme mono-puce par exemple) conduit en effet à imposer une plus grande flexibilité et hétérogénéité aux systèmes. Cette flexibilité peut s'obtenir par la réalisation de tout ou partie des opérateurs

de calcul sous forme logicielle. Cette solution conduit à une forme d'hétérogénéité des architectures qui est complexe à gérer. Unifier les mécanismes de communication entre les tâches matérielles et les tâches logicielles est un domaine de recherche particulièrement actif.

La flexibilité des architectures peut bien entendu aussi s'envisager par l'utilisation des technologies matérielles reconfigurables (FPGA notamment). On a montré que dans ce dernier cas, la gestion des ressources reconfigurables – surtout dynamiquement – implique là aussi de faire appel à des couches logicielles comme des systèmes d'exploitation embarqués. Les bénéfices mais aussi les verrous technologiques et scientifiques dûs à la présence d'OS dans les architectures reconfigurables a fait d'ailleurs l'objet d'une étude poussée dans le cadre du démarrage de notre projet OveRSoC.

Une autre manière d'obtenir une grande flexibilité dans les architectures de traitement consiste à employer les techniques de spécification, de description et de déploiement des applications jusqu'alors utilisées dans le domaine du génie logiciel. Les plates-formes de radio-logicielle, spécifiquement celles visées par la standardisation SCA, appliquent justement les concepts d'applications réparties, de méthodologie objet et les intergiciels CORBA pour atteindre ces objectifs de flexibilité.

Ce chapitre décrit donc les travaux que j'ai effectués selon deux axes particuliers. Le premier porte sur la présence de systèmes d'exploitation (temps-réels ou non) embarqués dans les architectures mono-puces, dotées éventuellement de capacités de reconfiguration dynamique (architecture RSoC :*Reconfigurable System-on-Chip*). Plus généralement, cette problématique aborde le déploiement d'applications fortement hétérogènes au sens de leur modèle de calcul puisqu'elles mettent en jeu des traitement cablés (matériels) et programmés (logiciels). Ces applications sont plus complexes à spécifier, à modéliser, à simuler et finalement à concevoir ou à synthétiser. Pour aborder ce problème, je me suis intéressé en premier lieu aux méthodes à employer pour modéliser et simuler, dans un flot de conception complet, la présence d'un OS dans les architectures. C'est particulièrement le comportement dynamique des applications et, par conséquent, celui d'un OS embarqué qui nous a intéressé dans un premier temps. Ces travaux sont présentés dans la partie §3.2, page 65.

Le deuxième axe de mes travaux qui est décrit dans ce chapitre porte sur le déploiement d'applications très complexes, dont la spécification est fortement influencée par les techniques du génie logiciel (diagrammes de classes UML, couches d'abstraction par API, CORBA, etc). Ces applications sont destinées à être portées sur des architectures fortement hétérogènes et distribuées. Nous avons abordé le problème de la conception de plates-formes de radio-logicielle sous deux angles. Le premier a porté sur l'utilisation des nouvelles techniques de modélisation des architectures (SystemC TLM) pour améliorer la prise en compte précoce, dans les flots de conception, des contraintes spécifiques à ce domaine applicatif. Le deuxième angle concerne la déclinaison en matériel (avec comme cible technologique les circuits FPGA) des mécanismes de base de la distribution d'applications orientées objet. La couche d'intergiciel distribué CORBA est au centre de notre problématique. Ces travaux sont décrits dans la partie §3.3, page 83.

3.2 La problématique d'intégration des OS dans les architectures embarquées

J'aborde dans cette partie l'ensemble des travaux que j'ai menés ou que j'ai encadrés sur le thème des applications mixtes (logicielles et matérielles) et leur implémentation sur des plates-formes réelles. En premier lieu, je présente les travaux qui ont été menés par Arthur Segard en 2003 sur un modèle d'unification des communications et des appels systèmes au sein d'une architecture dont une partie de l'application s'exécute sur des composants matériels et l'autre sur un (ou des) processeurs généralistes (§3.2.1). Ce modèle d'unification a été également proposé pour résoudre une partie des problèmes rencontrés dans les architectures reconfigurables dynamiquement.

Je présente ensuite (§3.2.2) les conclusions des discussions qui ont eu lieu, dans le cadre des actions nationales que j'ai initiées (GT « intégration système des SoCs » de l'EPML POMARD puis action « OS pour les architectures de TDSI » dans le thème C du GdR ISIS). Ces conclusions identifient les verrous liés à la conception de plates-formes mono-puces reconfigurables et, en particlulier, les besoins de modélisation et d'exploration des systèmes d'exploitation temps-réels nécessaires à leur gestion.

Je détaille ensuite (§3.2.3) la partie de mes travaux personnels sur la définition des mécanismes de base nécessaires à la modélisation d'un OS temps-réels en SystemC. Un modèle d'OS construit à partir de ces mécanismes a été développé et est présenté rapidement. Je donne également les premiers éléments d'une future méthodologie d'exploration d'OS embarqués, notamment pour la conception d'architectures fortement dynamiques (§3.2.4).

3.2.1 Les systèmes d'exploitation embarqués comme solutions à la gestion de l'hétérogénéité

Contexte
Cette partie des travaux a été réalisée par Arthur Segard, sous ma direction, au cours de son stage de DEA en 2003.

Les premiers travaux que nous avons menés sur la conception de plates-formes hétérogènes ont porté précisément sur la présence de couches opératives (de systèmes d'exploitation), embarquées dans les plates-formes, et servant de couche d'abstraction des ressources afin de résoudre le problème de l'exécution d'une application sur des architectures mixtes (des processeurs séquentiels associés à des unités de traitement cablées).

L'aspect précis des communications qu'il est nécessaire d'établir entre les différentes unités de traitement (possédant des modèles de calculs différents) nous a paru constituer le premier point dur de la conception de telles architectures. Notre approche a consisté à développer, au sein d'un système d'exploitation temps-réel simple, les mécanismes matériels et logiciels nécessaires à l'abstraction complète de l'hétérogénéité des ressources sur lesquelles s'exécute une application et permettant d'établir des communications tranparentes entre les différentes tâches logicielles et matérielles d'une application.

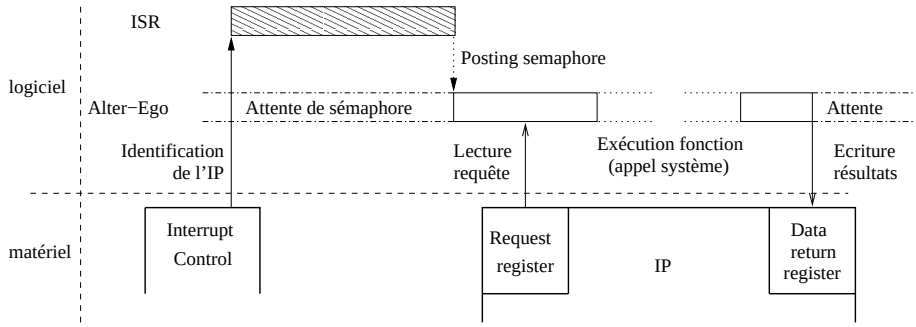


Fig. 3.1 – Illustration du fonctionnement d'une communication matérielle/logicielle grâce au modèle d'*Alter Ego*

Le modèle des *Alter Ego* logiciels

Dans un système d'exploitation temp-réel, une tâche est une portion de code à laquelle est associée, parmis d'autres informations, un identificateur unique et une priorité. Le rôle de la tâche d'ordonnancement est de partager le temps processeur (et les ressources associées) entre toutes les tâches applicatives en fonction de leurs priorités. Une tâche peut demander la création d'une autre tâche qui sera alors elle aussi ordonnancée sur le processeur. Le modèle des *Alter Ego* est basé sur un système d'exploitation temps-réel simple et portable (μC/OS-II, [89]) et permet à n'importe quel module dans un SoC de faire des appels système pour la création de tâches, la requête d'une ressource, l'exécution d'une fonction système, etc. Dans ce modèle, toutes les tâches matérielles (IP) sont encapsulées dans une tâche logicielle – l'*Alter Ego*, fonctionnant comme un *proxy* – qui est gérée par l'OS lui-même comme une tâche classique. Chaque tâche matérielle peut communiquer avec l'*Alter Ego* qui lui est associé et qui sera chargé d'exécuter les appels systèmes.

L'encapsulation des tâches matérielles est réalisée en trois niveaux : un *wrapper* matériel permettant de connecter l'IP sur les ressources de communication du SoC, une routine d'interruption spécifique d'interception des requêtes de l'IP vers l'*Alter Ego* et l'*Alter Ego* lui-même. Le fonctionnement des niveaux d'encapsulation est illustré sur la figure 3.1.

Nous avons démontré la portabilité du modèle en développant des applications sur deux types de plates-formes hétérogènes :

- un circuit FPGA Xilinx XC2VP7 contenant un processeur PowerPC 405 et deux tâches matérielles simulées par deux processeurs microBlaze,
- un circuit FPGA Altera Apex20k avec un processeur Nios 32 bits et deux IP propriétaires pour les calculs respectifs des fonctions mathématiques *log* et *sqrt*.

Sur ces deux types de plates-formes, nous avons montré l'efficacité de ce modèle pour des exemples de partage de ressources critiques et d'allocation de mémoire partagée. Les coûts d'implémentation de ce modèle (en termes de longueur de code additionnel, de cycles processeur supplémentaires et de transferts sur les bus) étant très limités, l'objectif de proposer une solution simple pour les communications hétérogènes a été rempli. De plus, il est apparu que le principe des *proxy* logiciels des tâches matérielles peut s'utiliser dans le cas des architectures reconfigurables dynamiquement, gérées par un système

d'exploitation. Cette partie des travaux a été publié en 2004 [99].

3.2.2 Les problèmes soulevés par la conception des plates-formes (R)SoC : identification des points durs

Contexte
Les travaux présentés dans cette section correspondent aux conclusions qui ont été tirées des travaux du groupe de travail « Intégration Système des SoCs » dans l'EPML POMARD ainsi qu'au premières pistes ayant conduit au dépôt du projet ANR OverSoC.

Les architectures que nous visons sont destinées à contenir l'ensemble des ressources matérielles et logicielles nécessaires à la réalisation complète d'un système de traitement de l'information (système de communication, système de contrôle industriel, système de pilotage et guidage automatique d'engin, vision emabrquée active...). Ces ressources sont classiquement :

- Un ou des processeur(s) de traitement généralistes (RISC ou pas),
- Un ou des processeurs à usage spécifiques (DSP, micro-contrôleurs),
- Les unités de contrôle des entrées / sorties (canaux de communication, convertisseurs AD ou DA, interfaces de capteurs),
- Les ressources de mémorisation (caches, hiérarchie mémoire),
- Les ressources d'interconnexion (bus, hiérarchies de bus, réseaux sur puce)

De telles plates-formes de type SoC (*System-on-Chip*) possèdent des degrés de complexité et d'hétérogénéité qui font que leur conception devient impossible par les techniques classiques. Les contraintes économiques de leur mise sur le marché (*time-to-market* de plus en plus court) et les contraintes de conception multiple (*co-design* ou conception conjointe matériel/logiciel) impliquent de nouvelles approches de conception inspirées de plus en plus par le génie logiciel : spécification de plus en plus abstraite, description des contraintes « métier » avant conception, spécification UML, transformation de modèles, etc.

De plus, les besoins industriels impliquent de plus en plus de flexibilité de ces plates-formes. Flexibilité opérationnelle d'une part, selon laquelle les plates-formes construites doivent pouvoir s'adapter à différentes applications ou à différentes missions au sein d'une même application. C'est le cas notamment des plates-formes envisagées pour la radio logicielle ou *Software Defined Radio* (voir §3.3, page 83). Flexibilité du flot de conception d'autre part, où l'on cherche à ré-utiliser au maximum des parties de ces plates-formes qui auraient déjà été conçues et validées. C'est le concept de la ré-utilisation d'IP (*Intellectual Properties*) et de la virtualisation des composants.

La flexibilité de ces plates-formes s'envisage aujourd'hui par l'introduction de ressources reconfigurables dans le système. L'adjonction de blocs de type FPGA par exemple permet de concevoir l'architecture matérielle de la plate-forme sans figer sa fonctionalité. De plus, on sait aujourd'hui que de nombreuses applications peuvent se réaliser de manière optimale sur de telles structures (convolutions, filtrages, applications de contrôle). On parle alors d'architectures RSoC (*Reconfigurable System on Chip*) comme illustré sur la figure 3.2.

La conception de ces architectures doit également pouvoir se faire au sein d'un flot démarrant à très haut niveau d'abstraction (au niveau de la spécification des besoins)

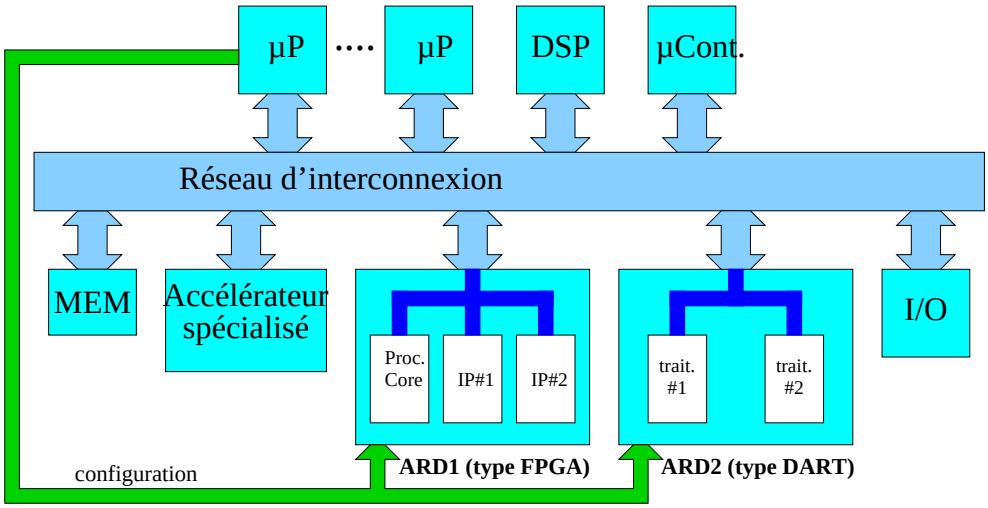


Fig. 3.2 – Architecture typique d'un RSoC organisé autour d'une resources d'interconnexion non précisée ici (bus ou réseau). Les accélérateurs reconfigurables dynamiquement (ARD) permettent d'obtenir la flexibilité complète de la plate-forme.

autorisant une validation à chaque étape de la conception.

L'approche que nous défendons consiste donc à intégrer dans le flot de conception de la plate-forme l'ensemble des éléments constitutifs de celle-ci, à savoir :

- les ressources matérielles nécessaires à l'exécution d'une application,
- les ressources de communication (en utilisant d'ailleurs l'approche de modélisation à plusieurs niveaux d'abstraction comme prônée par TLM),
- l'application à déployer (sous forme d'un graphe de tâches par exemple),
- le système d'exploitation de la plate-forme.

Le flot de conception lui-même que nous proposons n'est pas orienté vers la synthèse automatique d'architectures SoC (compilation d'architecture) mais plutôt vers une méthode de raffinement par étapes successives d'un modèle de départ, spécifié à un niveau d'abstraction qu'il reste à définir, vers un niveau de description capable d'alimenter les outils de génération architecturale (niveau RTL par exemple). En conséquence, les verrours scientifiques à lever sont de deux ordres : exécutif et de modélisation.

Les verrous exécutifs concernent la recherche des solutions à mettre en œuvre pour qu'une plate-forme RSoC puisse exécuter de manière optimale l'application choisie tout en respectant les contraintes qui ont été fixées. Ces points durs ont été identifiés en partie et sont :

- l'organisation et la (ou les) hiérarchie(s) de mémoire dans la plate-forme. Les problèmes non triviaux à résoudre sont la gestion de la cohérence du ou des caches dans l'architecture, la migration de blocs de données entre les structures mémoire, le partage des données sur des strutures complexes au travers de moyens de communication à définir, etc.
- La mise en œuvre de ressources de communication flexibles (c'est à dire reconfigurables) au sein des zones reconfigurables pour permettre l'allocation dynamique d'unités de traitement,

– la définition d'un modèle de programmation global compatible avec l'aspect hétérogène, distribué et reconfigurable de la plate-forme,
– la mise en œuvre de la pré-emption matérielle le cas échéant.

La modélisation, dans un flot de conception cohérent et efficace, de ces différentes solutions constitue le deuxième aspect des verrous scientifiques. Plus particulièrement, à cause du caractère très spécifique à une certaine application de ces techniques, le flot de conception doit pouvoir autoriser très tôt l'exploration de différentes solutions. La question est donc de déterminer comment on peut modéliser, explorer, raffiner et valider de telles plates-formes.

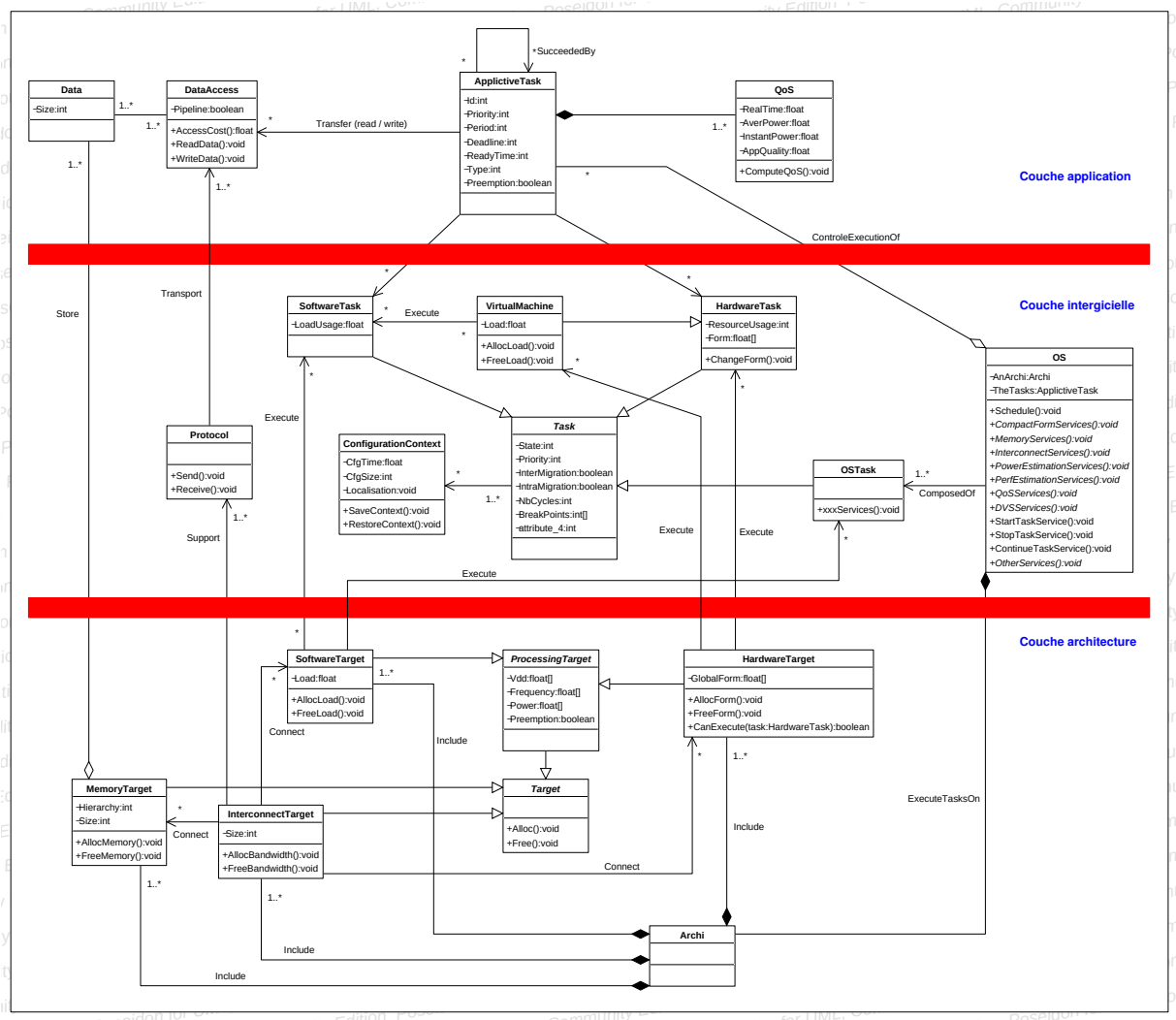


Fig. 3.3 – Diagramme de classes du modèle général d'une plate-forme RSoC.

L'argument que nous mettons en avant est celui de défendre la présence d'un système d'exploitation, temps-réel ou non, au sein des plates-formes. Cette couche serait non seulement chargée d'assurer l'exécution correcte de l'application sur la plate-forme mais constituerait également une abstraction de la plate-forme elle-même (une interface de programmation) facilitant le déploiement d'applications. De plus, il est apparu qu'un

tel système d'exploitation ne peut pas être construit à partir d'OS classiques auxquels on ajouterait certains services particuliers mais doit être conçu de manière *ad hoc* en même temps que la plate-forme en exploitant toutes les ressources de celle-ci y compris son aspect reconfigurable.

Une première ébauche de modélisation d'une plate-forme RSoC a donc été proposée dans le cadre de ces travaux et constitue le point de départ du projet OveRSoC. Cette modélisation a été faite sous la forme d'un diagramme de classes UML et est illustrée figure 3.3 (ce travail a été publié dans [74], [75] et [100]). Cette ébauche de modélisation complète d'une plate-forme RSoC est encore incomplète mais est destinée à fournir le cadre de départ de tous nos travaux ultérieurs.

3.2.3 Construction d'un modèle d'OS temps-réel de haut niveau

Contexte
Cette partie décrit les développements les plus récents de mes travaux qui s'intègrent en partie dans le cadre du projet OveRSoC *et plus généralement qui abordent le problème de la modélisation des plates-formes embarquées complexes.*

En ce qui concerne précisément le problème de la modélisation des systèmes d'exploitation pour les plates-formes RSoC, nous avons décidé de nous inspirer de la méthodologie prônée, en SystemC, par TLM pour la modélisation des communications en plusieurs niveaux d'abstraction. En effet, cette modélisation envisage l'abstraction des ressources de communication **en tant que fournisseur de services**. Le principe TLM consiste à séparer les problèmes de conception (donc de modélisation et de raffinement) des parties opératives et de communication d'une architecture. De ce point de vue, le système d'exploitation d'une plate-forme est bien une entité fournisseur de services (ceux de l'OS) qu'il doit être possible de modéliser (et de raffiner) selon plusieurs niveaux d'abstraction.

La suite de cette partie traite du problème de la modélisation d'un OS et comment cette modélisation peut intervenir dans un flot global de conception d'une plate-forme. Cette partie de mes travaux fait l'objet d'une soumission d'article actuellement en cours de révision [101].

Modélisation du système d'exploitation

On constate depuis l'année 2000 un certain nombre de travaux portant sur la problématique des OS embarqués dans les architectures SoC [76] et plus généralement sur la modélisation et la conception de ces OS [92], [81], [90], [86], [84].

L'un des travaux les plus intéressants sur le principe de modélisation des systèmes d'exploitation a été réalisé par l'équipe de D. Gajski (Université de Irvine, Californie) [103]. Ces travaux s'appuient sur le langage SpecC et non pas sur SystemC et ont comme finalité la synthèse automatique d'architecture. Ces travaux proposent la modélisation d'un OS sous forme de trois niveaux d'abstraction (figure 3.4) :

1. le modèle de spécification : l'OS n'est pas explicite et l'application s'exécute sur l'API du langage de description (ici SpecC). Seuls les canaux de communication et de synchronisation sont explicités grâce aux primitives du langage.

2. le modèle architectural : l'OS est présent dans le modèle sous la forme d'un ordonnanceur. Les communication et synchronisation se font toujours à l'aide des primitives du langage.
3. le modèle d'implémentation : c'est le modèle final, l'application est programmée « sur » l'API de l'OS qui, lui-même, s'exécute sur un simulateur de processeur.

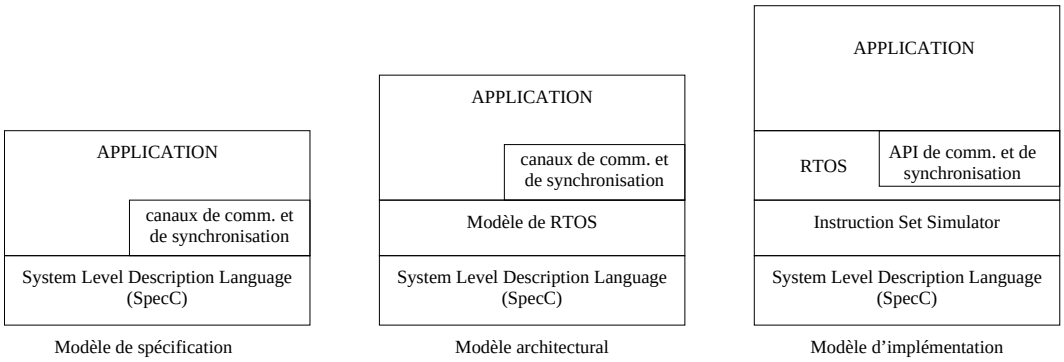


FIG. 3.4 – La modélisation d'un OS proposée dans les travaux de D. Gajski

Les travaux de l'université de Darmstadt sont également bien avancés sur la construction des modèles d'OS (cette fois en utilisant SystemC) [82] et [83]. En revanche, il n'y a pas de proposition de niveaux de raffinement de l'OS dans ces travaux et l'exploration de l'OS lui-même n'apparaît pas évidente.

L'avancée principale de nos travaux a consisté à développer un modèle SystemC d'un OS autorisant un comportement totalement dynamique, à savoir permettant la création et la destruction de tâches applicatives, à l'image exacte du comportement classique d'un OS temps-réel strictement logiciel. De plus, notre modèle supporte une démarche de raffinement qui permet à un concepteur d'explorer progressivement l'implémentation de cet OS.

L'aspect dynamique des tâches (*process* ou *threads* en SystemC) a été rendu possible avec les mécanismes introduits dans la version 2.1 du noyau officiel grâce à la bibliothèque publique `boost`[3]. Grâce à un mécanisme supplémentaire d'attachement de processus aux modules, il est désormais possible de créer et de supprimer des processus SystemC **après** la phase d'élaboration du modèle à simuler. En revanche, une forte limitation persiste dans le noyau SystemC : celle de la construction des modules et de leur hiérarchie au cours de la phase d'élaboration uniquement. En d'autres termes, le noyau SystemC ne peut simuler qu'une liste prédéfinie et statique de modules. Il n'est pas possible de créer dynamiquement (pendant la simulation) les modules SystemC symbolisant les IP. On identifie d'ailleurs là la principale limite à la simulation d'architectures reconfigurables dynamiquement.

À des fins d'illustration, la figure 3.5a décrit une partie d'une application de vision, découpée en tâches, qui a été utilisée pour mettre en évidence les mécanismes proposés. Cette application de vision est utilisée pour la navigation d'un robot et est inspirée de travaux en intelligence artificielle sur les boucles sensorimotrices ([93], [91]). La tâche

[3] `www.boost.org`

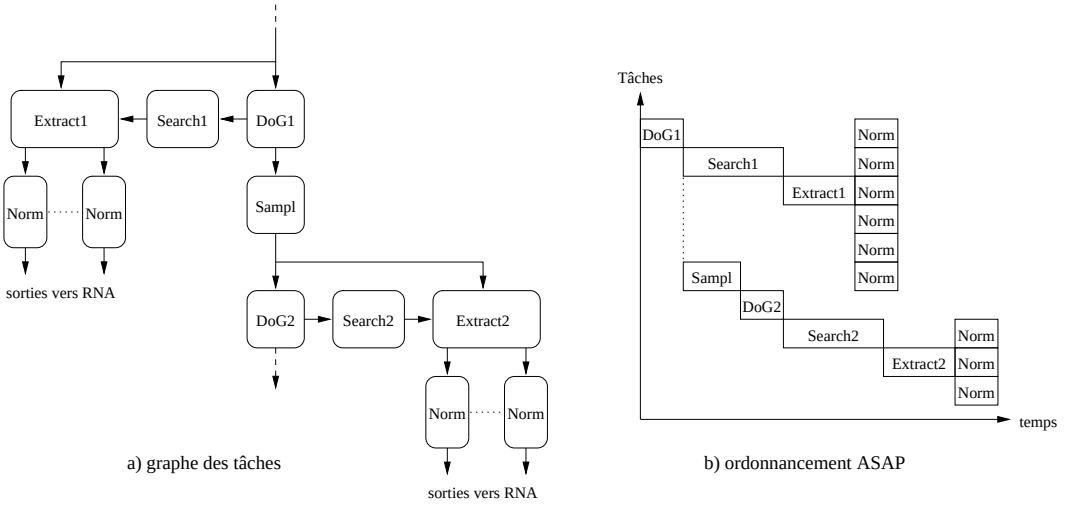


Fig. 3.5 – Exemple d'une application de vision représentée par son graphe de tâches. L'application qui est utilisée ici est basée sur un traitement multi-échelles, deux échelles uniquement sont représentées (à gauche) sur cette figure. Si les ressources disponibles sont en nombre infini et si un ordonnancement ASAP (au plus tôt) est utilisé, ce graphe de tâches pourrait être exécuté comme illustré à droite.

`DoG` effectue un filtrage de l'image par une différence de gaussiennes. La tâche `Search` recherche, trie et extrait les coordonnées de points d'intérêt dans l'image (après seuillage de leurs intensités). Cette tâche prend comme entrée les résultats des filtrages et ses paramètres sont N (le nombre maximum de points à extraire) et γ (le seuil de détection des points). La tâche `Extract` construit les voisinages circulaires autour des points identifiés et `Norm` effectue une transformation log-polaire des imagettes. Les imagettes ainsi construites sont destinées à alimenter une boucle sensorimotrice plus globale de contrôle des mouvements d'un robot autonome.

Nous avons démontré que la partie de traitement visuel de l'application robotique est soumise à une dynamique globale très fortement dépendante des conditions externes (l'environnement visuel). Cette dynamique influe fortement sur les différents paramètres du traitement visuel : les priorités éventuelles entre les traitements à effectuer aux différentes échelles, le nombre maximum de points d'intérêt à extraire à chaque échelle, le seuil de détection des points, la cadence de traitement imposée au système visuel. Nous sommes ici à la fois dans un cas de flexibilité extrême et de contrainte temps-réel forte où le problème majeur sera celui d'assurer l'ordonnançabilité de l'application. Assurer les conditions d'ordonnançabilité est hors du propos de nos travaux. Nous nous sommes concentrés sur les mécanismes qu'il sera nécessaire de fournir afin de pouvoir valider et simuler l'OS temps-réel qui sera responsable de la gestion de la plate-forme. La stratégie exacte d'ordonnancement qu'il faudra développer restant à définir.

De manière à obtenir le comportement dynamique espéré, l'application nécessite la création dynamique de différents traitements, avec des taux de parallélisme variables, en fonction de certains paramètres ou en fonction des données en cours de traitement. Ce type de fonctionnement ne peut pas être prévisible durant la phase de développement de

l'application ni au moment de la compilation. Le noyau de simulation SystemC ne permet pas non plus de gérer ce genre de dynamisme dans les traitements. En effet, le nombre et le niveau de parallélisme des processus classiques en SystemC doivent être connus avant le début de la simulation et ne peuvent pas varier en cours de simulation. Cette limite constitue le premier obstacle à la modélisation de systèmes fortement dunamiques à partir des processus SystemC.

Le second problème lié à la modélisation d'un système d'exploitation en SystemC consiste à trouver les mécanismes permettant d'**expliciter** l'ordonnancement des tâches d'une application plutôt que d'utiliser l'ordonnanceur évènementiel du simulateur. Le noyau de simulation natif de SystemC agit en effet comme une couche d'abstraction et fournit au développeur un modèle virtuel d'une architecture possédant un nombre infini de ressources. Tous les processus sont simulés de manière concurrente (comme l'ordonnancement illustré figure 3.5b). Si un nombre fini de ressource est envisagé, il faut dès lors partager le temps de chacune de ces ressources et ceci ne peut pas être simulé avec SystemC.

De plus, ordonnancer une tâche dans un OS temps-réel consiste non seulement à créer, démarrer ou terminer cette tâche mais aussi, si le principe de préemption est appliqué, à interrompre (suspendre) cette tâche et à changer de contexte. SystemC ne permet pas l'interruption des processus (*process* ou *thread*) sans une déclaration explicite d'un appel à l'ordonnanceur du simulateur avec la primitive `wait()`. Un appel `wait()` n'est pas réentrant et ne permet donc pas la prise en compte de multiples cycles de préemption et de reprise. Là encore, un mécanisme spécifique doit être développé pour ce schéma d'exécution.

Le modèle d'OS temps-réel que nous avons développé inclus l'ensemble de ces mécanisme et a été construit comme un canal SystemC hiérarchique contenant un ensemble de fonctions membres définissant une interface. Cette interface (illustrée figure 3.6) est inspirée à la fois d'une interface de services POSIX et de l'interface du système d'exploitation libre μC/OS-II [89].

Les fonctions membres du modèle de l'OS sont associées aux modules du noyau tandis que l'interface du modèle agit comme la définition de ces services. L'avantage principal du développement de notre modèle d'OS comme un canal hiérarchique réside dans la possibilité de raffiner l'implémentation des services de l'OS – donc d'en explorer les stratégies – sans modifier le prototype des services. Ce point important constitue la clé d'une future méthodologie d'exploration des OS embarqué à partir de ce modèle.

Modélisation des mécanismes de base

La création dynamique de tâches (de processus en SystemC) est désormais possible en utilisant la version 2.1 du noyau de simulation qui est distribué avec une librairie dédiée (la librairie `boost`). Cette librairie offre les primitives nécessaires à la création de processus (mécanisme de type `fork`) et à l'attachement de ceux-ci à des modules pendant la phase de simulation. De plus, grâce aux liste de sensitivité dynamiques, ces processus créés dynamiquement peuvent être synchronisés (démarrés et relancés) par des évènements eux aussi dynamiques. Comme le montre la figure 3.5b, le parallélisme potentiel de la tâche `Norm` dépend du seuil dynamique γ imposé par la boucle de régulation sensorimotrice

```
// POSIX-inspired RTOS interface
class basic_os_if : virtual public sc_interface
{
public:
  // Global interface
  virtual void OSInit()=0;
  virtual void OSStart()=0;

  // Task management
  virtual task* create_task(void (*f)(), int pri, const char *name)=0;
  virtual void kill_task(unsigned int p)=0;

  // Process ID services
  virtual int get_pid(task *t)=0;
  virtual task *get_task(unsigned int p)=0;

  // Preemption and exec. time modelling
  virtual void os_wait(task *t, sc_time t_DELAY)=0;
};
```

Fig. 3.6 – Interface SystemC de notre modèle de base d'un OS temps-réel. Cette interface virtuelle spécifie uniquement les services offerts par le modèle d'OS. L'implémentation de ces services est assurée par les fonctions membres du canal lui-même.

externe. La tâche d'extraction doit donc créer dynamiquement un nombre variable de tâches filles `Norm` en fonction des résultats de la tâche `Search`.

Le service de création de tâches Le service `create_task()` du modèle d'OS a été réalisé grâce à la primitive `sc_spawn()` du noyau. Ce service permet à une tâche applicative de créer dynamiquement plusieurs instances de nouvelles tâches. Le modèle d'OS représente les tâches sous la forme de classes C++ contenant toutes les informations liées aux tâches comme des variables membres : la priorité, la période, l'échéance, un pointeur sur la fonction C++ symbolisant le code fonctionnel de la tâche, etc. Dans ce modèle d'OS, les tâches ne sont pas des instances de `sc_module` SystemC mais des fonctions C++ liées au moment de la compilation du modèle. De cette manière, il n'existe pas de lien statique entre le code fonctionnel d'une tâche et l'instance particulière du modèle d'OS qui aura la charge de l'exécuter. Cette possibilité pourra être exploitée ultérieurement lors de l'exploration d'une version distribuée de l'architecture. Pratiquement, `create_task()` crée une nouvelle instance de *thread* attachée au canal symbolisant l'OS auquel a été fait l'appel à `create_task()`. Un évènement spécifique de synchronisation est également créé pour démarrer ou reprendre l'exécution de cette nouvelle tâche. Enfin, `create_task()` crée toutes les structures internes de l'OS relatives à cette nouvelle tâche.

En fournissant le code applicatif sous la forme d'un programme C++ plutôt que sous la forme de multiples modules SystemC, nous garantissons une indépendance entre le code fournit par le développeur logiciel et l'implémentation de l'exécutif. Cette contrainte permettra une séparation plus claire entre les spécifications d'entrée et l'exploration de l'architecture sous-jacente.

L'ordonnanceur du modèle Le module principal de notre modèle d'OS est l'ordonnanceur des tâches. Ce module est développé comme une méthode membre du canal

hiérarchique. De cette façon, et en utilisant les techniques de polymorphisme, différentes techniques et stratégies d'ordonnancement peuvent être explorées par la surcharge des méthodes. Des travaux intéressants, mais avec une approche plus classique, ont très récemment été publiés par une équipe de l'Université de Porto Alegre au Brésil sur le raffinement de l'ordonnanceur d'un système [85]. La figure 3.7 illustre deux exemples d'ordonnancement avec ou sans le mécanisme de préemption (présenté plus loin) qui peuvent être utilisés pour valider la gestion des priorités des tâches par exemple. L'évaluation de la charge de l'OS, par la prise en compte du nombre de changements de contextes par exemple, peut être faite très tôt dans le cycle de développement grâce à ce modèle.

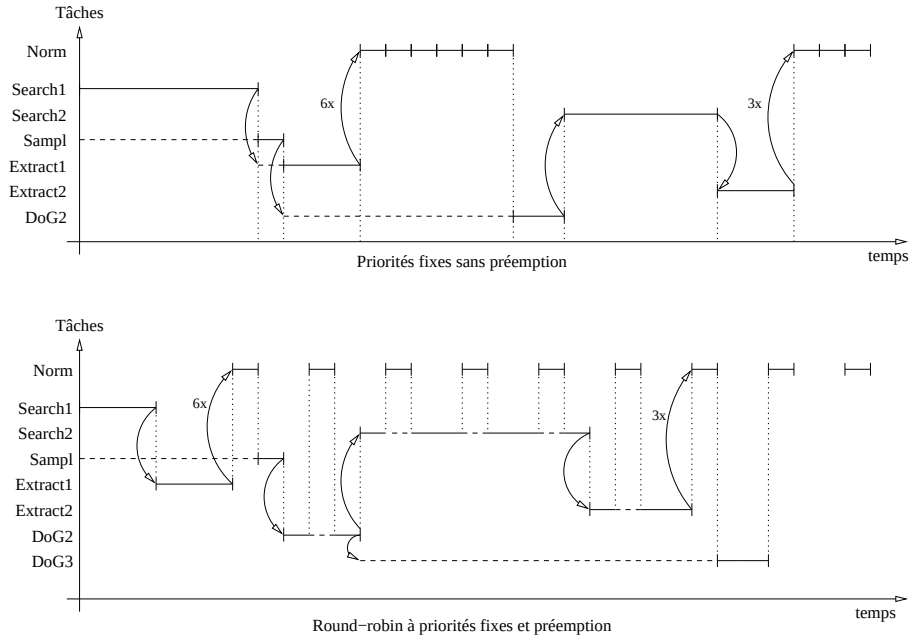


FIG. 3.7 – L'exploration progressive des stratégies d'ordonnancement (avec ou sans préemption) est possible avec le modèle d'OS. Les flèches représentent les appels à `create_task` et les lignes horizontales en pointillés représentent les tâches en attente d'exécution.

La préemption des tâches Pour modéliser l'interruption et la préemption des tâches, nous avons développé un service dédié `os_wait()` qui rend la main au modèle d'OS et qui permet la prise en compte de la préemption et des changements de contexte. La primitive `os_wait()`, qui encapsule la primitive `wait` de SystemC, est écrite comme un appel bloquant qui retourne si le délai précisé comme paramètre s'est écoulé, délai auquel s'ajoute l'ensemble des délais additionnels dûs à d'éventuelles exécutions multiples de l'ordonnanceur. Le service `os_wait()` agit comme une modélisation des temps d'exécution des tâches. Les tâches sont en conséquence décomposées en portions de code non-interruptibles caractérisées chacune par leurs temps d'exécution estimés. Les frontières de ces portions de code peuvent être définies, de manière similaire à ce qui est fait dans la modélisation proposée par l'équipe MCSE [90], par les appels systèmes ou les instructions de rupture de séquence. La figure 3.8 illustre ce découpage.

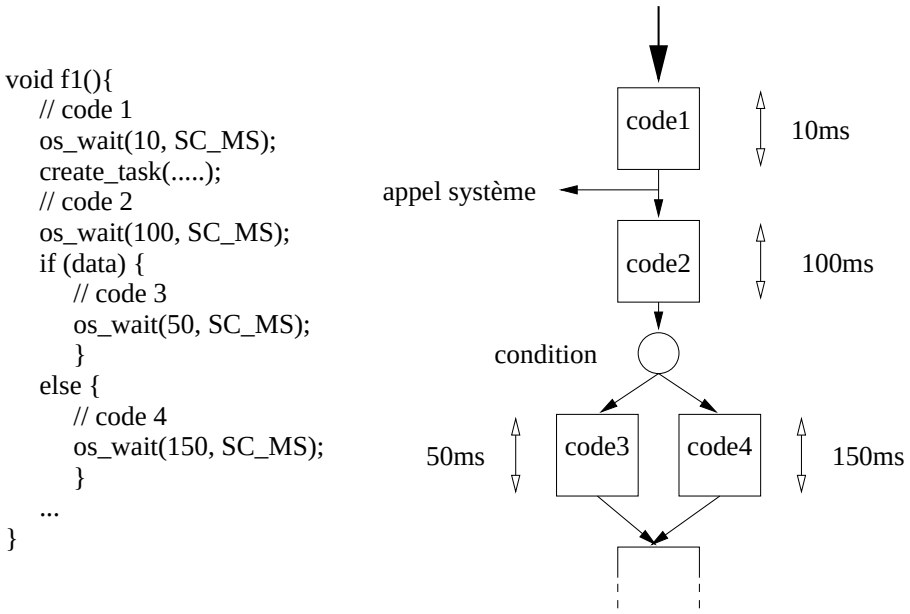


Fig. 3.8 – Modélisation d'une tâche applicative en portions de code non-interruptibles dont les frontières sont les appels systèmes et les ruptures de séquence. Les temps d'exécution sont estimés par l'exécution exclusive de la tâche par une ressource de calcul. Le modèle incluera ces estimations en ajoutant les éventuelles interruptions et durées cumulées des changements de contextes.

Pour illustrer le mécanisme de préemption que nous avons développé, la figure 3.9 donne le détail de la simulation de l'exécution de deux tâches sur une architecture monoprocesseur. Dans cet exemple, lorsque la tâche `taskA` s'exécute, le modèle simule son temps d'exécution (delayA). En réalité, le code associé attend à la fois la terminaison de ce délai et un éventuel évènement temps-réel (le "tick") qui réveillera l'ordonnanceur. Puisque le temps d'exécution de `taskA` est supérieur au temps qui lui a été alloué, le temps écoulé $T = t_2 - t_1$ est pris en compte et soustrait au temps d'exécution restant. au temps t_2, l'ordonnanceur s'exécute et démarre la tâche `taskB` qui possède une priorité supérieure. Après la simulation du temps de changement de contexte $(t_3 - t_2)$, l'évènement de synchronisation de la tâche `taskB` est notifié et l'ordonnanceur attend un nouveau "tick".

Au temps t_3 la tâche `TaskB` exécute son code jusqu'au premier appel à `os_wait()`. L'exécution du code fonctionnel en SystemC se fait en temps nul et n'intervient pas dans la simulation. À son tour, `TaskB` effectue un appel à `os_wait()` pour une durée qui est ici inférieure à la période du "tick" et se termine (*timeout*). Au temps t_4 la tâche `taskA` est réveillée par l'ordonnanceur et le noyau SystemC termine la simulation de son exécution (au temps t_5).

Il est intéressant de noter qu'avec cette modélisation de la préemption, le nombre de changements de contexte entre des processus SystemC lors de la simulation est approximativement égal aux nombre de changements de contexte simulés. Ceci conduit à un modèle efficace de simulation sans *overhead* et permet d'imaginer une simulation suffisamment

rapide.

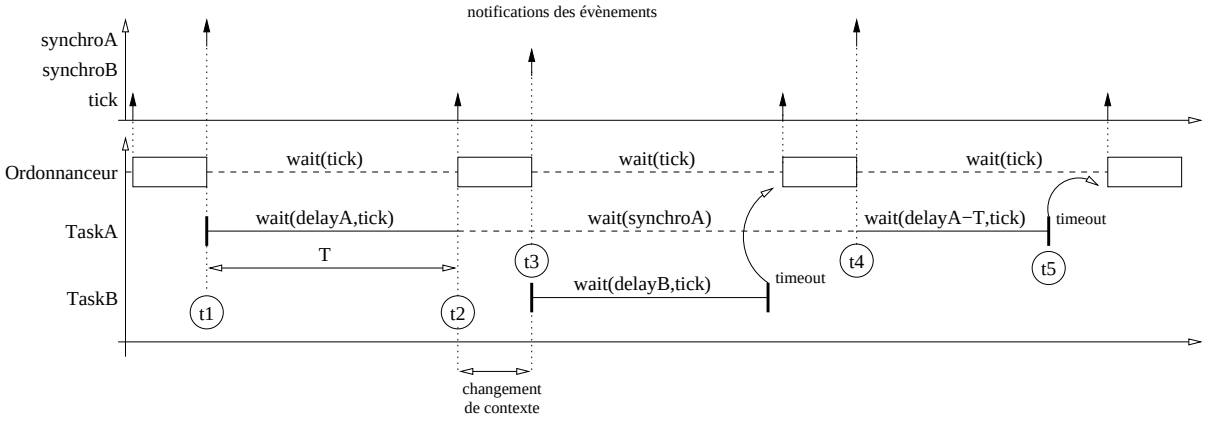


FIG. 3.9 – Details du mécanisme de modélisation de la préemption et des temps d'exécution avec le service `os_wait()`.

Construction modulaire du modèle d'OS Grâce à nouveau à la nature orientée objet de la librairie SystemC, le modèle de notre OS peut être développé de manière modulaire. Ajouter un nouveau service ou modifier un service existant peut être facilement réalisé grâce à la notion d'héritage et les relations entre les objets. L'interface de programmation de notre OS (son API) fonctionne de la même manière et peut être mise à jour ou augmentée en fonction des besoins de l'application. À titre d'exemple, la figure 3.10 donne l'interface d'un module de communication par sémaphores d'exclusion mutuelle (mutex) qui a été développé. D'autres services de communication et d'autres modules du noyau peuvent être modélisés de manière identique (mémoires FIFO, boîtes aux lettres, etc). Il reste cependant à définir précisément un patron de conception (un *design pattern*) qui permettra la mise à jour automatique de l'API du modèle en fonction des services que le développeur incluera dans le modèle.

```
// Communication services interface
class basic_comm_if : virtual public sc_interface
{
  // dynamical creation of mutexes
  virtual OS_sem *OS_create_semaphore()=0;
  // mutex request, blocking call
  virtual void OS_sem_lock(task *t, OS_sem *s)=0;
  // mutex release
  virtual void OS_sem_unlock(task *t, OS_sem *s)=0;
};
```

FIG. 3.10 – Interface de base du module de communication par mutex.

Résultats expérimentaux

Ce modèle d'OS temps-réel a été développé entièrement et a été validé dans une hypothèse mono-processeur avec tous les services et modules présentés ici. Deux stratégies

d'ordonnancement ont été testées avec succès sur ce modèle : un ordonnanceur à priorités fixes et sans préemption et un algorithme temps-réel de type *round-robin*. L'application de vision robotique mentionnée page 71 a également été validée sur ces deux modèles. Grâce à l'API relativement standard, l'effort de reprogrammation de l'application (initialement développée en C) pour le portage sur le modèle d'OS a été réduit au minimum : modification des appels de création de tâche et de manipulation des sémaphores. De plus, la simulation de l'application de vision en SystemC sur notre modèle offre des possibilités intéressantes. Une bibliothèque C++ de fonction graphiques et de manipulation d'images (`gtk+2.0`) a été très facilement incluse dans le code de l'application et a permis une vérification fonctionnelle rapide du code (la figure 3.11 donne les résultats obtenus). Enfin, le modèle produit des traces d'exécution (sous forme brute dans un fichier de debug pour le moment) qui permettent un examen approfondi du comportement du modèle.

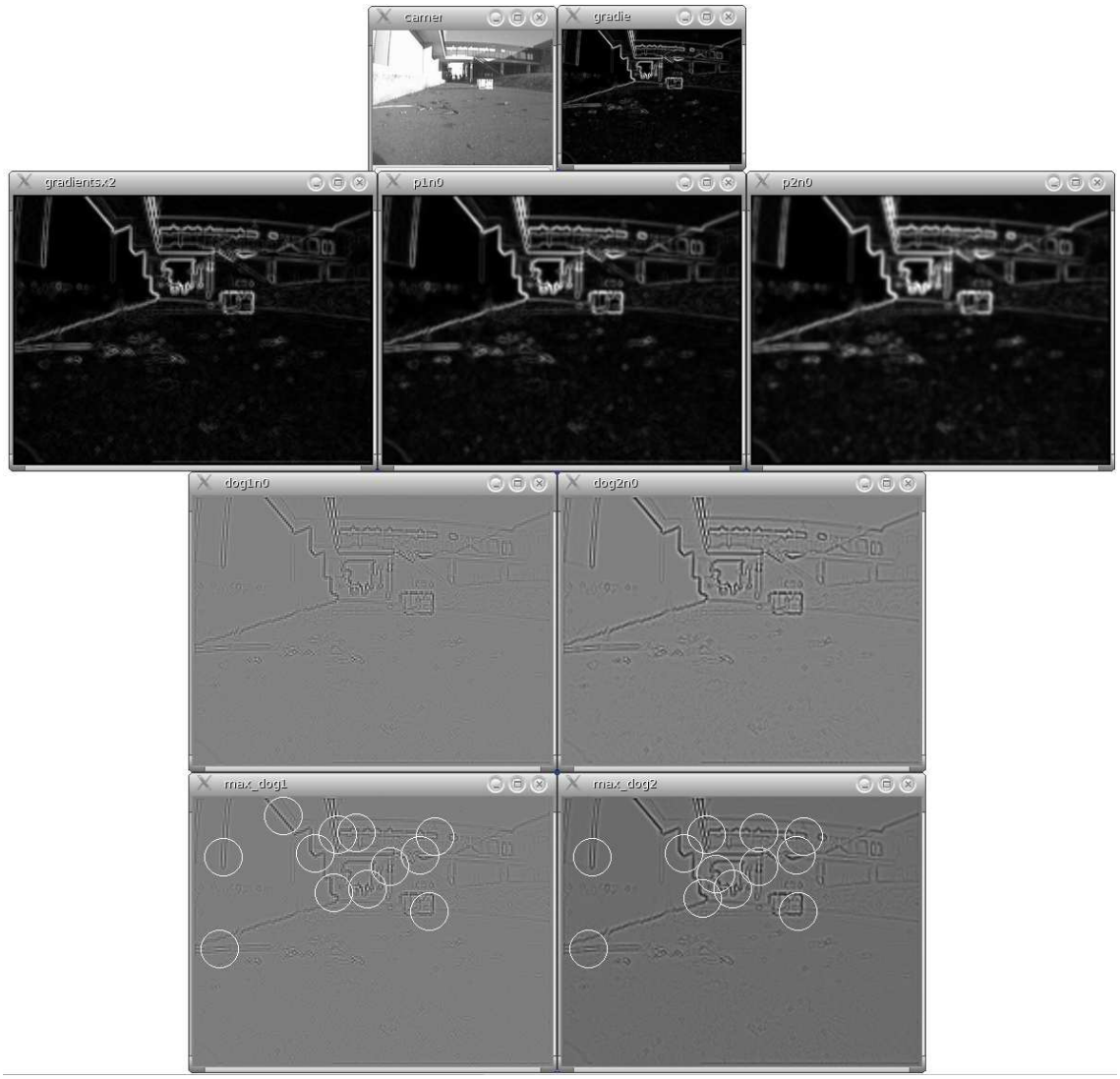


Fig. 3.11 – Résultats graphiques de la simulation en SystemC du modèle exécutif de l'application de vision obtenus avec la bibliothèque C++ `gtk+2.0`.

Une modélisation du comportement temporel d'une architecture exécutant une application au travers d'une couche OS est également possible à partir de ces travaux. En effet, des informations temporelles peuvent être insérées dans le modèle de l'OS lui-même (en utilisant la primitive `wait` de SystemC) pour estimer les délais de changement de contexte. De plus, par le découpage des tâches applicatives en portions élémentaires, il est possible de simplifier l'estimation des temps d'exécution pour des applications fortement dépendantes des données. De cette manière, nous estimons qu'il devient possible d'effectuer une exploration, à très haut niveau d'abstraction, des options architecturales ainsi que des simulations de performances et ce, y compris dans des cas *a priori* difficiles comme pour les architectures reconfigurables dynamiquement.

3.2.4 Conclusion et perspectives

Vers l'exploration des stratégies d'implémentation des OS pour RSoC

À partir de ces travaux et expériences, nous proposons les premières pistes d'une méthodologie d'exploration d'architectures RSoC qui inclue l'exploration et la validation des OS embarqués. Cette méthodologie est basée sur des étapes de raffinement successives à la fois de l'architecture et des stratégies d'implémentation de l'OS temps-réel embarqué. Ce flot, illustré figure 3.12, est partiellement inspiré des travaux de l'équipe de D. Gajski [103] ainsi que de ceux développés au TIMA par l'équipe de A. Jerraya [80] sur la génération automatique d'OS embarqués. Les travaux de [103] proposent une approche de modélisation en trois étapes (niveau spécification, architecture and implémentation). L'inconvénient principal de cette approche est qu'il existe toujours un fossé de conception entre la description architecturale (celle qui fait apparaître l'OS) et la description de l'implémentation (où les processeurs sont modélisés par leurs simulateurs ISS). Pour combler ce fossé, nous proposons un niveau intermédiaire de description distribuée où le parallélisme et les stratégies d'OS peuvent être explorées. L'utilisation de notre modèle d'OS intervient à la deuxième étape de cette méthodologie.

La méthodologie proposée se décompose en quatre niveaux successifs de modélisation (spécification, exécutive, distribution et implémentation) qui sont décrit ci-dessous :

1. Modèle de spécification
   Le modèle de spécification est développé comme un modèle SystemC classique où toutes les tâches applicatives sont des processus ou *threads* SystemC et où les communications entre les tâches sont réalisées avec les primitives de communications de l'API SystemC (`sc_mutex`, `sc_fifo`, etc). Les tâches sont synchronisées avec les `sc_events`. L'ensemble de l'application peut être écrit comme un unique `sc_module` avec toutes les tâches comme fonctions membres. Une vérification fonctionnelle complète de l'application est possible à ce niveau de représentation, à condition que toutes les tâches aient leur représentation fonctionnelle en C++. Le noyau SystemC et son interface de programmation se comportent comme une virtualisation complète de l'architecture qui exécute l'application. Étant donné le principe de concurrence en SystemC, la simulation à ce niveau suppose un nombre infini de ressources. Seules les dépendances de contrôle et de données entre les tâches sont modélisées. Toutes bibliothèque additionnelle permettant une mise au point aisée de l'application peut

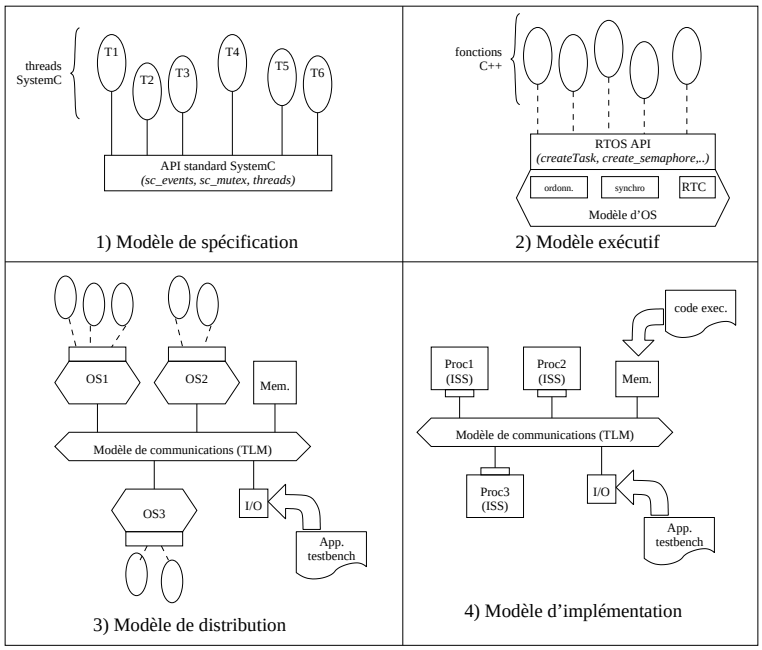


Fig. 3.12 – Le flot de raffinement de plate-forme que nous proposons démarre avec un modèle de spécification et inclue quatre étapes de raffinement.

être incluse dans le modèle (comme la bibliothèque graphique de notre exemple, page 78).

2. Modèle exécutif
   Le second modèle raffine le précédent en ajoutant une couche explicite d'OS à l'architecture. À ce niveau, un module SystemC contient toute l'application. Ce module est "connecté" au canal représentant l'OS par un port spécifique requérant l'interface `basic_os_if`. Ce module principal ne contient que la première tâche de l'application qui symbolise le code de démarrage de la future application embarquée. Cette tâche initialise l'OS (`OSInit()`), crée la première tâche applicative et lance l'ordonnanceur (`OSStart()`). Les éventuels appels aux primitives de communication de SystemC (les `sc_mutex` par exemple) sont remplacés par les appels correspondant de l'interface (`OS_create_semaphore()`, `OS_sem_lock()` et `OS_sem_unlock()` par exemple). De plus, les temps d'exécution des tâches peuvent être modélisés grâce au service `os_wait()` à l'intérieur des frontière élémentaires de code. Aucune autre modification n'est nécessaire sur le code de l'application et une vérification rapide est possible.

   Ce modèle permet une première exploration de certaines stratégies de l'OS : politique d'ordonnancement, modèle de préemption, services optionnels. L'exploration du partitionnement logiciel/matériel de l'application n'est pas encore possible à ce niveau. L'objectif essentiel est l'exploration du séquencement global des opérations dans l'architecture.

3. Modèle distribué
   Le modèle de distribution est construit en instanciant plusieurs OS indépendants

(éventuellement différents). Les différents modèles d'OS représentent une architecture multi processeurs et ont accès aux fonctions C++ de la spécification applicative. Dans ce cas, bien entendu, chaque modèle d'OS doit inclure les modules nécessaires à la synchronisation et à la communication entre les noyaux. La migration éventuelle des tâches entre les processeurs ainsi que l'équilibrage des charges de calcul nécessitent également le développement de modules spécifiques. Tous ces modules ou services peuvent être explorés et validés à ce niveau. De plus, l'architecture matérielle est précisée en faisant apparaître les modèles de mémoire, de ressources de communication et les canaux d'entrée/sortie. En ce qui concerne précisément les ressources de communication, à partir de ce niveau de modélisation il est possible d'utiliser les canaux de communication SystemC proposés par la démarche TLM (*Transaction Level Modeling*) afin de séparer les préoccupations exécutives (les modèles d'OS) et architecturales (les topologies et protocoles de communication).

Nous défendons également l'idée que cette modélisation de l'architecture peut constituer le premier pas de la modélisation d'architectures flexibles et reconfigurables dynamiquement. Un bloc de ressources matérielles reconfigurables – de type FPGA par exemple – peut être modélisé par l'un des modèles d'OS et, à condition de développer les services *ad hoc*, une modélisation complète d'une plate-forme RSoC est possible. Il s'agira bien entendu, dans ce cas, de mettre au point les mécanismes nécessaires à la modélisation en SystemC de ressources matérielles dynamiques. Ces derniers points font d'ailleurs partie des objectifs du projet OverSoC.

4. Modèle d'implémentation
   Ce modèle est la dernière étape avant la synthèse physique de la plate-forme. Les éléments matériels (mémoires, dispositifs d'entrée/sortie, gestionaires d'interruptions, accélérateurs matériels, etc) sont décrits par leurs modèles RTL. Les processeurs sont modélisés par des simulateurs de jeux d'instructions (ISS). Le code source de l'application doit maintenant être compilé et inséré dans le modèle de la plate-forme sous la forme de code exécutable. Les résultats de simulation sont précis au cycle près (*Cycle Accurate*) ou au niveau du signal (*Bit Accurate*) en fonction de la précision des modèles des différents éléments. À partir de cette étape, la plupart des stratégies d'implémentation du ou des OS ont été explorées et validées et la suite de la conception physique de la plate-forme se fera avec les flots classiques.

Cette méthodologie d'exploration peut s'appliquer à plusieurs plates-formes purement logicielles (multi-processeurs hétérogènes ou non) comme à des plates-formes mixtes (réunissant des unités matérielles et logicielles). Elle s'applique à de nombreux domaines applicatifs et est destinée particulièrement aux contextes dynamiques [101].

Explorer et concevoir une plate-forme embarquée dynamique

Sous réserve de compléter et valider toutes les "briques de base" de notre modèle d'OS, il est possible de démarrer l'exploration des stratégies d'ordonnancement hétérogène (tâches matérielles et logicielles) qui sont nécessaires à la réalisation d'une plate-forme embarquée fortement dynamique. L'exemple applicatif naturellement envisagé est celui de la vision robotique pour la navigation et la reconnaissance d'objets. Ces applications sont

effectivement soumises à de fortes contraintes de flexibilité, soit par la boucle de régulation sensorimotrice à l'intérieur de laquelle elles sont insérées, soit par la nature même des traitements qu'elles réalisent et qui peuvent être fortement dépendants des données. L'exemple de la détection et caractérisation en temps réel d'obstacles est significatif de cette classe d'application : ni le nombre d'obstacles (donc la quantité de traitement) ni la contrainte de temps de calcul (liée à la vitesse de déplacement du robot) ne peuvent être connus à l'avance.

La conception complète d'une telle plate-forme embarquée fait d'ores et déjà partie de nos perspectives. Ce travail repose sur plusiseurs axes de recherche qui concernent :

- l'étude de l'ordonnançabilité sous contraintes variables de ce type d'application,
- l'étude des algorithmes d'ordonnancement dynamiques en ligne avec, éventuellement, contrainte de réduction d'énergie puisque la plate-forme est embarquée,
- partitionnement matériel et logiciel des applications afin d'exploiter au maximum les puissances de calcul offertes par les accélérateurs matériel et la flexibilité des traitements logiciels.

Ces axes seront notamment abordés dans le cadre de la thèse de Emmanuel Huck, en collaboration avec Thalès-TRT, qui a commencé en Novembre 2006.

3.3 Le contexte des plates-formes de radio logicielle

Contexte
Cette partie des travaux entre dans le cadre de la thèse de Grégory Gailliard [79], avec un financement CIFRE, sous la forme d'une collaboration en cours avec la société THALÈS-COMMUNICATION.

Les plateformes reconfigurables, à quelque niveau que ce soit (reconfiguration à la volée, dynamique ou pas, plates-formes simplement re-programmables) ont trouvé depuis quelques années un domaine d'application particulièrement attractif. Ce domaine est celui des de la radio logicielle [94], [98]. Une « machine » pour la radio logicielle est un système de communication possèdant des ressources hétérogènes (matérielles et logicielles) capable de supporter l'exécution de plusieurs formes d'ondes. Une forme d'onde est la définition des éléments fonctionnels et de leurs interconnexions réalisant une chaîne complète de communication, du *front-end* radio jusqu'au traitement des informations. Les objectifs de la réalisation de telles architectures sont la **portabilité** et la **flexibilité**. La portabilité est entendue ici comme le portage d'une même forme d'onde sur différents systèmes de communication conduisant à l'interopérabilité des systèmes. La flexibilité s'entend au sens de la reconfigurabilité possible des systèmes permettant de supporter de multiples formes d'ondes. La figure 3.13 illustre l'organisation d'une plate-forme de radio-logicielle.

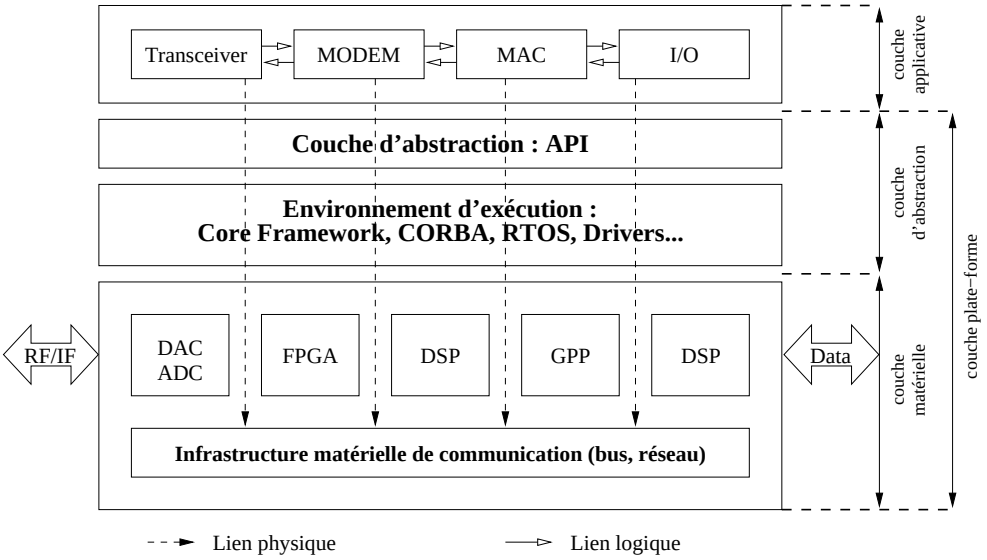


FIG. 3.13 – Modèle d'architecture hétérogène et distribuée pour la radio-logicielle. L'organisation de cette architecture est basée sur les couches d'abstraction multiples entre les composants de la forme d'onde (couche applicative) et l'ensemble des ressources d'exécution (couche matérielle).

Pour atteindre ces objectifs, la reconfigurabilité matérielle tend à trouver ici un cadre d'application idéal. Les circuits de type FPGA devenant de plus en plus performants pour des applications de traitement du signal de plus en plus complexes, ils deviennent des concurrents sérieux aux processeurs de type DSP pour lesquels, paradoxalement, la reprogrammabilité est évidente.

Pour remplir les contraintes de la radio-logicielle, une architecture à la fois matérielle et logicielle capable d'exécuter différentes formes d'ondes doit être conçue. La conception de cette architecture hétérogène nécessite une méthodologie efficace qui permette, à haut niveau d'abstraction, l'exploration de l'espace des solutions et la simulation des modèles de la façon la plus réaliste possible.

La **portabilité** des formes d'ondes peut être obtenue par l'utilisation des couches d'abstraction matérielle, de façon similaire à ce qui est utilisé en développement logiciel : interfaces de programmation (API), environnements d'exécution (services d'OS, intergiciels), drivers, etc. En matériel, de tels concepts existent sous forme de *wrappers* et de convertisseurs de protocoles. Lors de la conception d'une plate-forme, des compromis doivent être trouvés entre la portabilité possible des applications et les performances atteintes. Une méthodologie adéquate doit pouvoir offrir la possibilité de définir et d'évaluer ces différentes couches d'abstraction matérielles et logicielles.

La **reconfigurabilité** logicielle des plates-formes peut être obtenue soit par modification des paramètres d'exécution, soit par remplacement des traitements. Lorsque les traitements sont matériels, les changements de paramètres s'imaginent aisément par écriture dans des mémoires ou registres de paramètres. Le remplacement des traitements, en revanche, implique la reconfiguration matérielle (de manière statique ou dynamique). Certaines applications extrêmes peuvent même impliquer l'**auto-reconfiguration** de la plate-forme elle-même, comme dans le cas de la radio cognitive par exemple [102] où l'usage massif de technologies reconfigurables est envisagé ([104] notamment). Là encore, une méthodologie particulière doit être appliquée pour évaluer au plus tôt dans le cycle de conception les différentes stratégies de reconfiguration.

Les environnements classiques de conception ne sont pas adaptés à ces nouveaux enjeux principalement à cause des ruptures dans les flots de développement : l'évaluation des applications, des contraintes et des stratégies s'effectue à très haut niveau (programmation en C, C++ ou Matlab) tandis que les plates-formes sont conçues à partir de langages de description matériels comme VHDL ou Verilog.

Nos premiers travaux dans ce cadre se sont donc orientés vers le principe de la virtualisation des plates-formes. Par virtualisation, nous entendons ici la modélisation fidèle à très haut niveau d'abstraction d'une plate-forme complète autorisant l'évaluation de stratégies d'exécution ainsi que la réutilisation de composants. La réutilisation est, en effet, le premier pas vers l'évaluation des plates-formes reconfigurables. Notre choix s'est, là encore, orienté vers le langage support SystemC et son noyau de simulation propre. Ces travaux ont été publiés dans [77] et [78].

3.3.1 SCA : le génie logiciel appliqué à l'embarqué

Une initiative du département de la défense américain a permis de proposer une démarche de spécification et de déploiement des applications de radio-logicielle. Le programme d'étude JTRS (*Joint Tactical Radio System*) a été lancé afin de renouveler et rendre inter-opérables l'ensemble des systèmes de radio-communication des forces militaires navales, aériennes et terrestres. Ce programme a conduit, sous l'égide du JPO (*Joint Program Office*) à l'élaboration de règles et d'interfaces (*Framework*) pour déployer, configurer et gérer de multiples formes d'ondes sur des systèmes baptisés JTR (*Joint Tactical*

Radios).

Cette démarche nommée SCA (*Software Commmunications Architecture*) [88] est largement inspirée des techniques de génie logiciel et s'organise à la manière du déploiement d'applications réparties orientées objet. Le découpage d'une forme d'onde, en utilisant la démarche SCA, sur une plate-forme de radio logicielle est illustrée sur la figure 3.14.

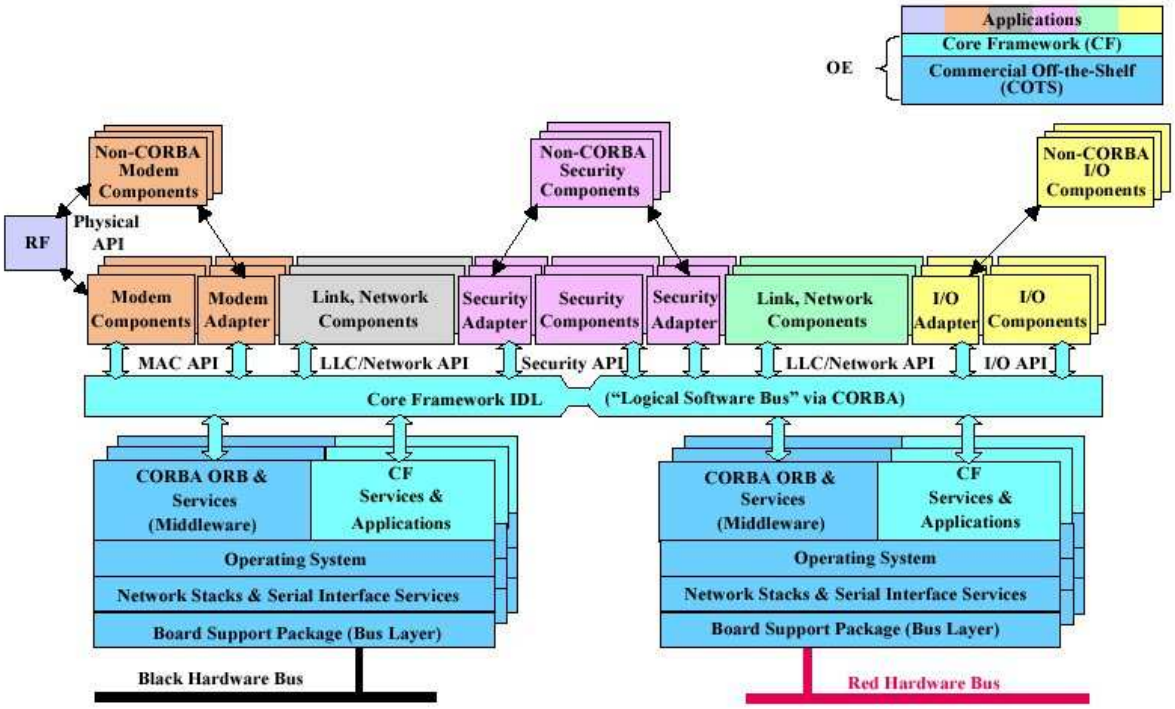


FIG. 3.14 – Organisation d'une application de radio-logicielle selon la démarche SCA. L'ensemble de la forme d'onde est définie par une collection d'objets reposant sur la couche d'abstraction CORBA. Le contexte essentiellement militaire a imposé la séparation de la plate-forme en une partie cryptée (*Black Bus*) et non-cryptée (*Red Bus*).

La méthodologie SCA est basée sur un certain nombre de choix techniques logiciels matures :

- un intergiciel **CORBA** [96] (*Common Object Request Broker Architecture*) destiné à l'abstraction des communications et de la distribution d'une application multi objets,
- un système d'exploitation conforme à **POSIX** (*Portable Operating System Interface*) pour l'abstraction des environnements d'exécution sous-jacents à la forme d'onde et
- le langage **XML** (*eXtensible Mark-up Language*) pour la spécification de l'application et des ressources d'exécution.

L'environnement d'exécution principal, sur lequel repose l'exécution des formes d'ondes sur une plate-forme de radio-logicielle conforme à SCA, est principalement constitué du *Core Framework* (CF). Ce *Core Framework* est composé d'un ensemble d'interfaces de services qui permettent le déploiement, la gestion, l'interconnexion et l'inter-communication

des composants d'application d'une forme d'onde.

D'un point de vue global, la démarche d'architecture logicielle SCA est riche et bien formalisée dans ses choix techniques, son environnement d'exécution et ses interfaces de programmation. Il n'en reste pas moins qu'elle n'est adaptée qu'aux solutions de déploiement, de portabilité et de reconfigurabilité de formes d'ondes **logicielles** uniquement. En revanche, pour faire face aux nouvelles normes de communication, il est nécessaire de fournir aux formes d'ondes des puissances de calcul de plus en plus importantes et cette puissance ne sera disponible qu'à la condition d'intégrer de plus en plus d'accélérateurs matériels au sein des plates-formes. Les aspects de conception **matérielle** des plates-formes ainsi que de leur éventuelle **reconfiguration** ne sont pas du tout abordés dans les choix du SCA.

Au cours de ce travail, nous avons identifié les impacts des applications et plates-formes de radio-logicielle à la fois sur les flots de conception et sur les architectures elles-mêmes. En ce qui concerne les flots de conception, les besoins suivants doivent être pris en compte :

- un haut niveau d'abstraction,
- l'hétérogénéité des cibles d'exécution des composants de forme d'onde (DSP, processeur généraliste, FPGA, ASIC),
- la réutilisabilité des composants matériels et logiciels,
- la reconfiguration des composants,
- la validation d'une forme d'onde vis-à-vis du respect des spécifications fonctionnelles,
- une simulation à la fois précise et rapide.

En ce qui concerne les choix architecturaux, les besoins identifiés sont :

- une architecture de communication hétérogènes, flexible et reconfigurable s'adaptant aux variations de qualité de service,
- des interfaces de communication favorisant la portabilité, la reconfigurabilité et la réutilisation,
- et un environnement d'exécution (matérielle ou logicielle) fournissant les **services** standards du SCA,

3.3.2 Les méthodologies de développement des plates-formes et formes d'ondes

Actuellement, la méthodologie émergente pour le développement des plates-formes et formes d'ondes est basée sur les concepts de l'approche par modèles (MDA : *Model Driven Architecture*) [95] et sur le paradigme composant/conteneur.

L'approche MDA prône la séparation des préoccupations entre les spécifications de la fonctionalité d'un système, sous la forme d'un modèle indépendant de la plate-forme (PIM : *Platform Independant Model*) et l'implémentation de ces spécifications sous la forme d'un modèle spécifique à la cible (PSM : *Platform specific Model*). Les modèles PIM sont généralement fournis sous forme de spécifications UML alors que les modèles PSM sont développés dans des langages de type C, C++, Java, VHDL, etc.

D'autre part, le concept composant/conteneur permet une séparation claire entre les propriétés de comportement et les choix techniques d'implémentation. Le composant est une entité réutilisable qui réalise la fonction désirée (le code "métier" ou *business code*)

et qui fournit ou nécessite des services au travers d'interfaces clairement définies. Le conteneur fait l'adaptation entre le composant et l'interface de l'infrastructure d'exécution. Quelques soient les modifications des formes d'ondes ou des plates-formes, le conteneur garantit la portabilité et la reconfigurabilité nécessaire dans la radio-logicielle.

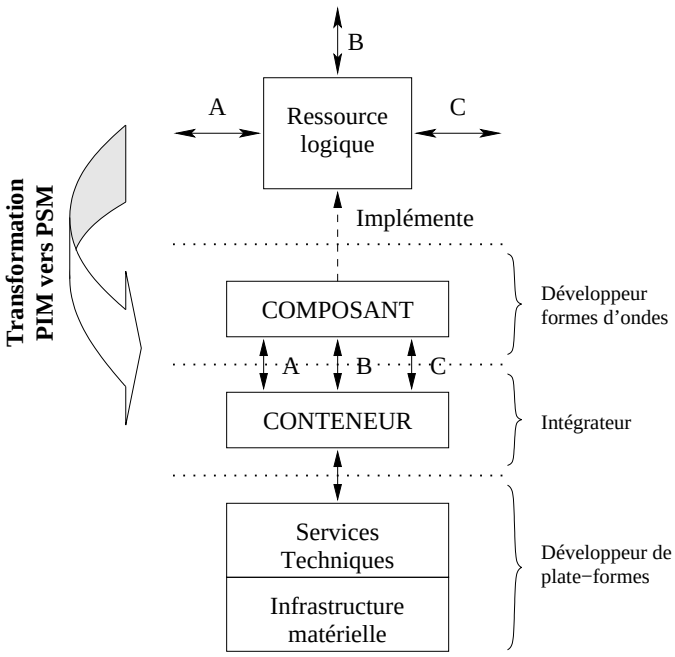


FIG. 3.15 – Méthodologie inspirée de MDA et basée sur le concept composant/conteneur pour le développement d'applications de radio-logicielle.

La figure 3.15 illustre cette démarche autour du PIM d'une ressource de traitement abstraite, contenant des interfaces logiques (A, B et C). La ressource logique, les interfaces logiques ainsi que les éventuelles contraintes temps-réel associées aux communications entre la ressource de traitement et le système radio peuvent être exprimés en UML avec les diagrammes de classes, de scénario, d'état et les cas d'utilisation (*use cases*). La transition PIM vers PSM consiste à définir le composant qui implémente cette logique de traitement et qui communique avec son environnement, grâce à son conteneur, au travers des interfaces A, B et C. Le conteneur transpose la sémantique des interfaces logiques vers les services fournis par la plate-forme.

Cette approche est compatible avec la philosophie SCA lorsqu'elle est déployée sur des processeurs généralistes (GPP) et a été également adaptée avec succès sur des cibles DSP. Elle semble pouvoir également se décliner sur des cibles matérielles telles que les FPGA.

3.3.3 Vers un flot de conception exploitant la virtualisation des plates-formes

L'inconvénient que nous avons identifié dans cette approche est qu'une validation complète de l'application vis-à-vis des contraintes de la forme d'onde ne peut être effectuée

qu'une fois l'ensemble des ressources de traitement conçues. Nous proposons donc d'utiliser dans ce flot des modèles PIM et PSM exécutables qui permettraient des évaluations (simulations) au plus tôt dans la démarche. L'utilisation de modèles SystemC TLM peut remplir cet objectif pour virtualiser les plates-formes d'exécution, y compris dès la spécification des modèles PIM de l'application (figure 3.16).

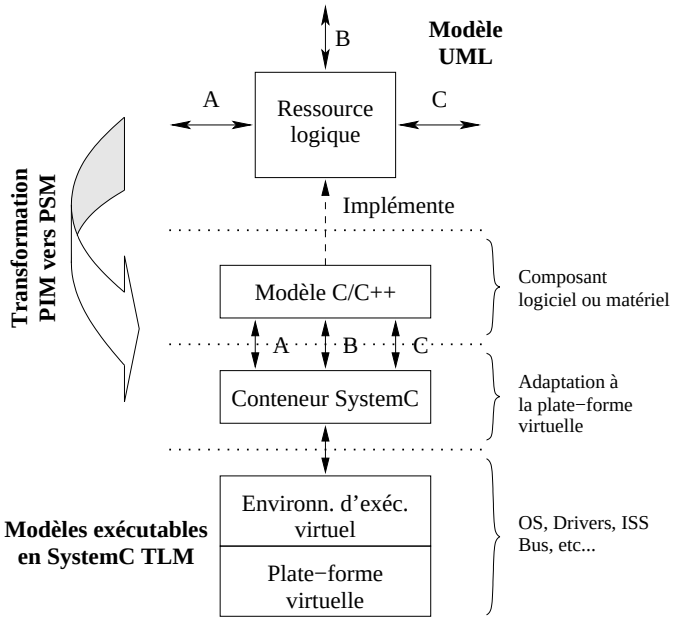


Fig. 3.16 – Extension de l'approche MDA par l'insertion de modèles exécutables PIM et PSM en SystemC TLM.

Comme nous l'avons défendu déjà dans la partie précédente, SystemC est un langage de modélisation qui permet une représentation unifiée des ressources matérielles et logicielles d'une plate-forme. C'est donc une solution de représentation qui s'impose naturellement dans le cas de la conception des plates-formes de radio-logicielle. De plus, l'utilisation des modèles SystemC dans la démarche de simulation transactionnelle (TLM) offre un bon compromis entre précision et vitesse de simulation.

Nous défendons donc là encore l'idée de la virtualisation des plates-formes comme solution à l'exploration efficace des solutions de conception. On a vu que dans le cas de la modélisation des systèmes d'exploitation (§3.2), un modèle SystemC et son interface représentent une couche de virtualisation des ressources et services offerts par l'entité modélisée. Dans le cas de la radio-logicelle, les aspects qui seront prépondérants pour satisfaire les contraintes de portabilité et de reconfigurabilité sont l'abstraction des communications dans les architectures hétérogènes ainsi que la réutilisation d'IP (*Intellectual Properties*). L'utilisation de modèles SystemC TLM doit permettre d'atteindre ces objectifs.

Un modèle PSM en SystemC : virtualisation d'un SoC

Ces concepts ont été appliqués à la modélisation d'une IP au sein du flot de conception et de raffinement d'un modèle PSM en SystemC TLM. Le modèle PSM qui a été conçu

est celui d'une plate-forme existante de SoC (DiMITRI [97] développé chez THALÈS) pour une application de radio-diffusion numérique DRM (*Digital Radio Mondiale*) [87]. L'architecture en question est celle d'un récepteur de radio numérique AM/FM possédant un *front-end* numérique, une démodulation OFDM et plusieurs codecs multimédia. Le SoC qui a été développé est architecturé autour de deux processeurs ambarqués ARM (figure 3.17).

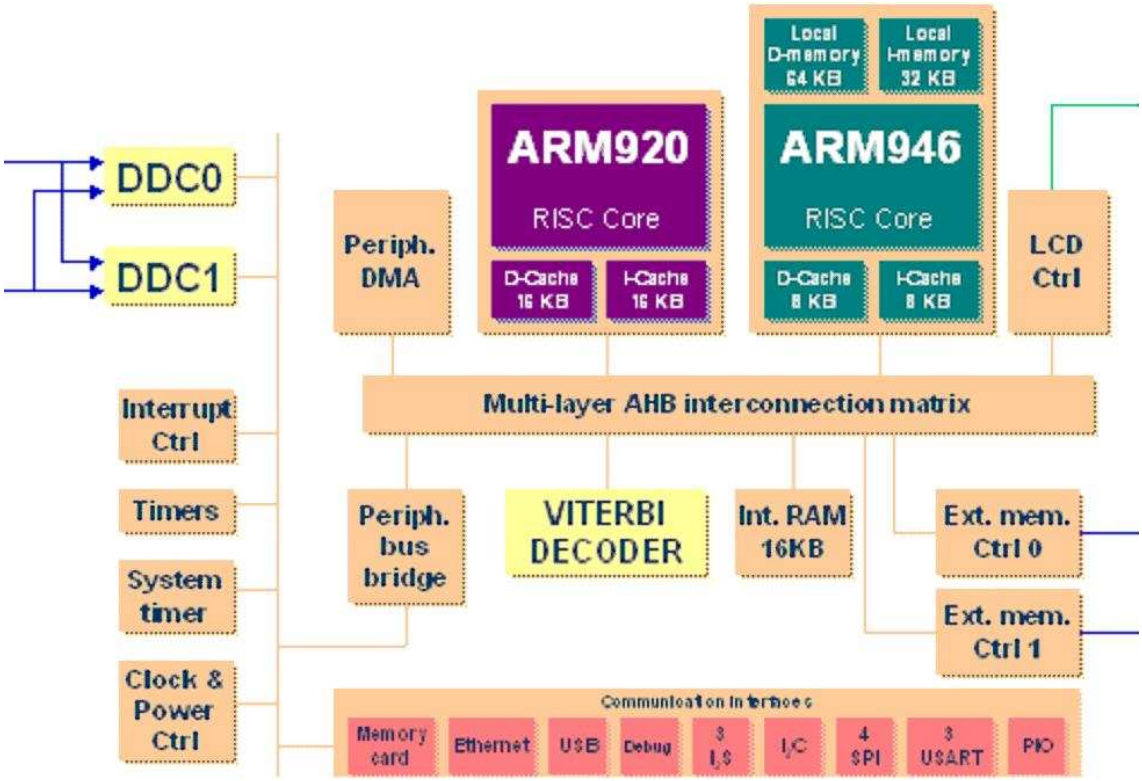


FIG. 3.17 – Architecture du SoC DiMITRI conçu chez THALÈS pour la réception de radio numérique AM/FM.

L'expérience qui a été menée a consisté à modéliser en SystemC le bloc Viterbi précédemment validé (l'IP a été conçue, testée et on dispose de son code VHDL) et à l'inclure dans un modèle virtuel de la plate-forme complète. À partir de la description fonctionnelle de l'IP Viterbi, un modèle SystemC a été développé au niveau *Programmer's View* (PV). Le niveau de description PV correspond à une modélisation fonctionnelle des ressources matérielles d'une architecture permettant la validation logicielle. Les ressources matérielles sont modélisées par le banc de registres de leur interface et leur description comportementale. À ce niveau de description, le modèle ne contient aucun détail temporel ni de performance. Seuls l'espace d'adressage de l'IP, l'adresse des bancs de registres et les interruptions sont représentés (figure 3.18).

Dans cette modélisation, le code fonctionnel en C du comportement de l'IP représente la logique "métier" et ses services (la fonction `ViterbiDecode()` entre autres) sont appelés par le conteneur via les fonctions *callback* associées au registres et champs de bits. Le conteneur implémente la logique "technique" de synchronisation, de stockage et d'in-

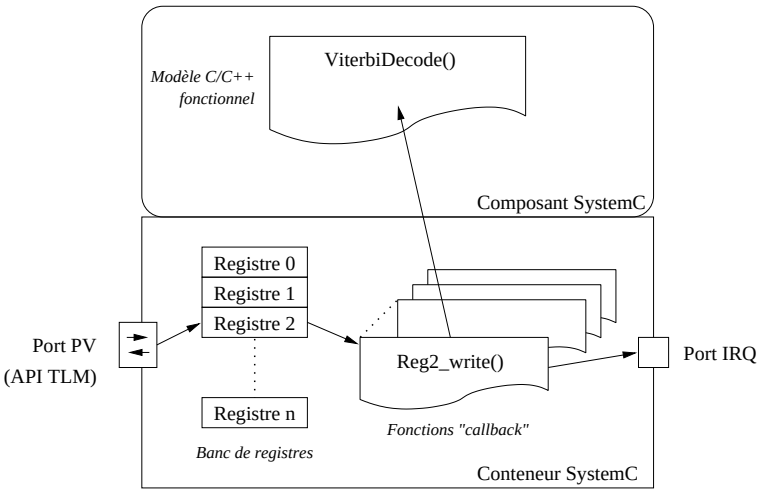


Fig. 3.18 – Modélisation SystemC au niveau PV d'une IP en appliquant la démarche composant/conteneur.

terface. Le code fonctionnel en C de l'IP est directement inclus dans le modèle, compilé avec le modèle et reste intact. Ce code constitue d'une certaine manière la spécification de référence telle que celle, par exemple, qui aurait été utilisée dans une spécification purement logicielle (en Matlab).

Ce modèle de l'IP Viterbi a été inclus dans une première ébauche de la plate-forme virtuelle du SoC (figure 3.19). Cette architecture est entièrement modélisée en SystemC et contient un modèle de processeur ARM, sous la forme d'un simulateur ISS (*Instruction Set Simulator*), les blocs de mémorisation et un canal de communication OCP. Un transacteur PV↔OCP permet la connexion de l'IP sur le bus OCP. Le transacteur effectue uniquement la traduction des protocoles.

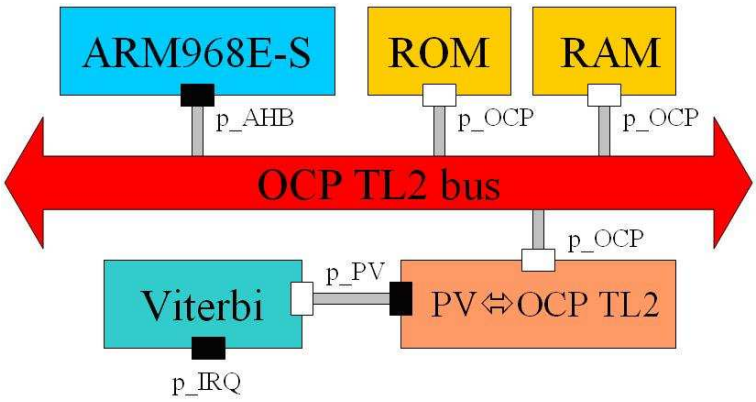


Fig. 3.19 – Première plate-forme virtuelle contenant l'IP Viterbi décrite au niveau PV.

À ce niveau de description, le modèle de la plate-forme est fonctionnel et la simulation évènementielle ne permet pas encore la prise en compte des informations temporelles. En revanche, la validation fonctionnelle de l'application complète est possible : le code source compilé et exécuté par le simulateur ARM correspond au test unitaire utilisé sur le circuit

réel, le driver Viterbi original a été utilisé sans modification. La première conclusion partielle de ce travail est que, au prix d'un effort de modélisation relativement limité (ré-écriture légère de la spécification fonctionnelle de l'IP en code C), il a été possible d'effectuer une simulation complète de la plate-forme, en SystemC, avec une séparation claire entre les préoccupations de la logique comportementale (fonction et interface de l'IP) et de la logique technique (communication, synchronisation et stockage) tout en appliquant les concepts composant/conteneur.

Cette expérience a été poursuivie en raffinant le modèle du SoC par le remplacement du réseau de communication virtuel OCP par un modèle plus proche de l'implémentation (un bus AMBA AHB) et par l'exploration d'un second modèle de processeur (cœur ARM926-EJS en place du cœur ARM968-ES). La seconde plate-forme virtuelle est décrite figure 3.20. Un nouveau transacteur PV↔AHB a été développé. De plus, un modèle de contrôleur d'interruptions a été ajouté dans la plate-forme.

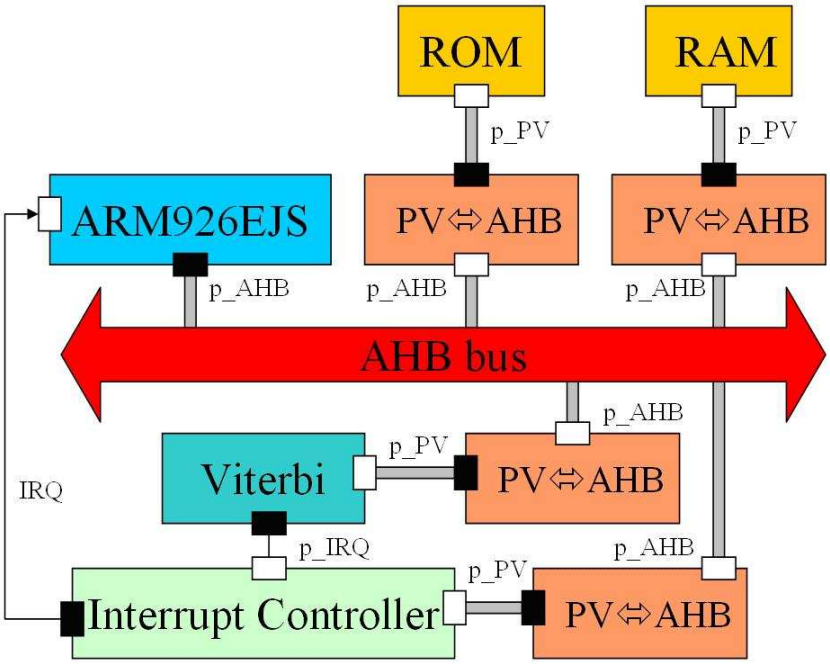


FIG. 3.20 – Raffinemant de la plate-forme virtuelle par insertion d'un bus de communication AMBA et d'un contrôleur d'interruptions. Le code PV de l'IP Viterbi a été réutilisé sans modifications dans ce modèle.

L'utilisation de modèles de processeurs ISS précis au niveau des cycles d'horloge (CA : *Cycle Accurate*) autorise cette fois la prise en compte d'informations temporelles et permet l'exploration et la validation des ressources de communication (charge du bus, estimation de performances, etc).

3.3.4 Conclusion et perspectives

Nous avons démontré avec ce travail qu'il est possible d'adopter la démarche composant/conteneur dans la phase de spécification, de modélisation et d'exploration d'une

plate-forme, à condition d'employer les outils de modélisation et les niveaux de description adéquats. Cette démarche offre la possibilité d'effectuer une réelle ré-utilisation de ressources matérielles pré-conçues lors de la conception d'une plate-forme SoC.

La suite de ce travail va se concentrer cette fois sur les ressources matérielles à prévoir dans la plate-forme pour que la philosophie composant/conteneur soit applicable d'une façon plus générale dans la démarche SCA pour les plates-formes de radio-logicielle :

Décliner en matériel les services CORBA pour la radio-logicielle

Nous avons vu que les concepts clés des applications et plates-formes de radio-logicielle sont la **portabilité** et la **reconfigurabilité**. Nous avons développé une méthode basée sur SystemC TLM pour assurer la portabilité (dans le flot de conception) des composants matériels de formes d'ondes. La portabilité exécutive des composants (c'est à dire dans la plateforme elle-même) est censée être assurée par l'utilisation d'une couche d'abstraction des communications et de la distribution des applications (CORBA). D'autre part, la reconfigurabilité des plates-formes pourra être assurée par l'utilisation de composants reconfigurables (FPGA ou structures à gros grain). Il reste donc à décliner au niveau des ressources matérielles de la plate-forme les services et interfaces de distribution des traitements.

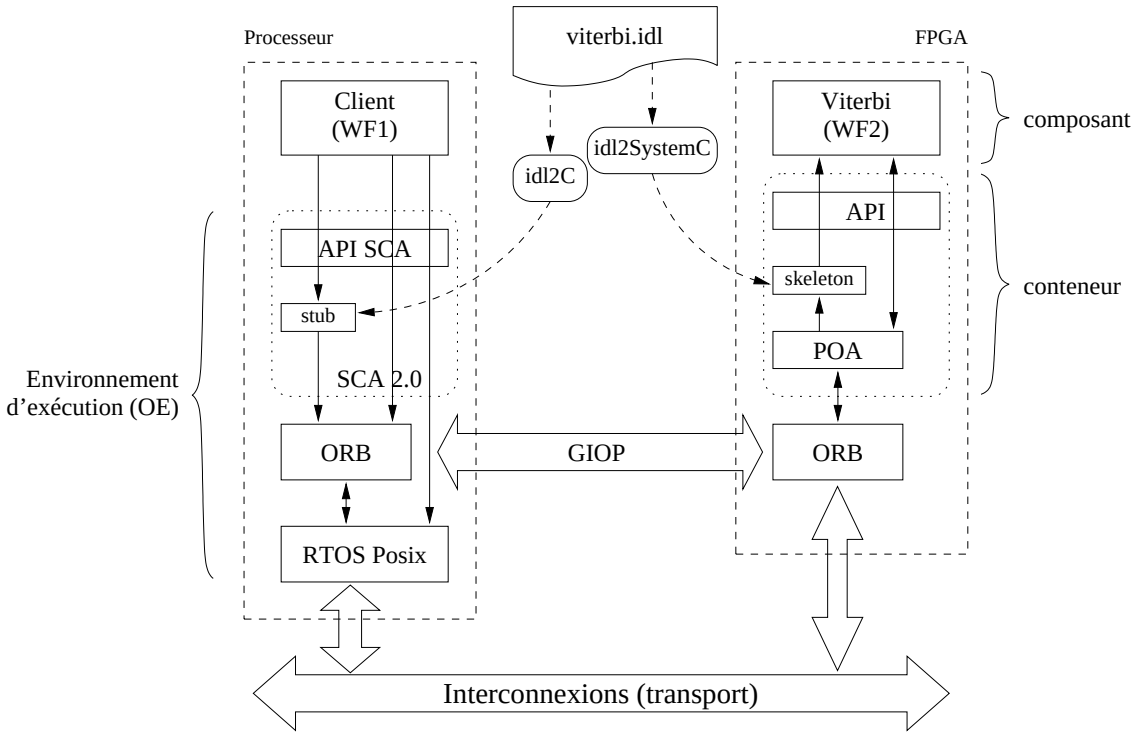


Fig. 3.21 – Déclinaison de l'architecture globale d'une application de radio-logicielle sur des ressources de traitement hétérogènes au travers de la couche CORBA. Les conteneurs matériels assurent l'interfaçage GIOP vers les composants éventuellement reconfigurables dynamiquement.

Nous défendons l'idée que cette déclinaison d'un modèle de type CORBA est possible

à la fois dans les ressources d'interconnexion et dans les composants matériels reconfigurables. La figure 3.21 décrit une vision de l'architecture dans laquelle une application de radio-logicielle, vue comme un ensemble d'objets répartis communicants (clients et serveurs) peut se déployer à la fois sur des unités de traitement programmées (des processeurs) et sur des circuits de type FPGA.

La déclinaison des couches d'abstraction CORBA dans du matériel reconfigurable peut s'envisager à la manière de ce que nous avons développé dans la modélisation des plates-formes, à savoir par l'utilisation de conteneur ou *wrappers* implémentant les services nécessaires. Dans cette proposition, et afin de séparer les préoccupations de communication d'une part et d'implémentation des traitements d'autre part, les communications au sein de l'architectures sont envisagées au niveau GIOP (*General Inter-ORB Protocol*, de niveau client-serveur) et assurées par une couche transport sous-jacente (un bus AHB avec interfaces OCP par exemple). Une proposition similaire a été faite par l'un des partenaires importants du consortium CORBA (la société Prismtech[4]) mais n'apporte que très peu d'élements de solution concernant l'implémentation des nombreux services nécessaires d'un réel CORBA matériel. Plus précisément, nos recherches s'intéresseront aux services de résolution des références (*naming service*), de dépôt des implémentations (*implementation repository*) et de déploiement qui sont absolument nécessaires si les ressources sont dynamiques comme dans le cas envisagé de reconfiguration matérielle des plates-formes.

Ces perspectives constituent les objectifs à court terme de la thèse de Grégory Gailliard, en collaboration avec Thalès-Communications.

[4] `www.prismtechnologies.com`

Références Bibliographiques

[74] I. Benkhermi, A. Benkhelifa, D. Chillet, S. Pillement, J-C. Prévotet, and F. Verdier. Modélisation niveau système de SoC reconfigurables. In *8ème Symposium en Architectures nouvelles de machines (SympA'05)*, Le Croisic, France, April 2005.

[75] I. Benkhermi, A. Benkhelifa, D. Chillet, S. Pillement, J-C. Prévotet, and F. Verdier. System-Level Modelling for Reconfigurable SoCs. In *20th Conference on Design of Circuits and Integrated systems (DCIS'05)*, Lisboa, Portugal, November 2005.

[76] D. Desmet, D. Verkest, and H. De Man. Operating System based Software Generation for Systems-on-Chip. In *Proceedings of the IEEE Design Automation Conference (DAC)*, pages 396–401, Los Angeles, CA, USA, June 2000.

[77] G. Gailliard, B. Mercier, M. Sarlotte, B. Candaele, and F. Verdier. Towards a SystemC TLM based Methodology for Platform Design and IP Reuse : Application to Software Defined Radio. In *Second European Workshop on Reconfigurable Communication-centric SoCs (ReCoSoC'06)*, Montpellier, France, July 2006.

[78] Gregory Gailliard, Eric Nicollet, Michel Sarlotte, and François Verdier. Transaction Level Modelling of SCA compliant Software Defined Radio Waveforms and Platforms PIM/PSM. In *International Conference on Design and Test in Europe (DATE 2007)*, Nice, France, April 2007.

[79] Grégory Gailliard. Extension de la démarche SCA au niveau plate-forme SoC. Thèse CIFRE Thalès-Communication, depuis le 1er Novembre 2005.

[80] L. Gauthier, S. Yoo, and A. Jerraya. Automatic generation and targeting of application specific operating systems and embedded systems software. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE)*, pages 679–685, Munich, Germany, 2001. IEEE Press.

[81] Andreas Gerstlauer, Haobo Yu, and Daniel Gajski. RTOS Modeling for System Level Design. In *International Conference on Design, Automation and Test in Europe (DATE)*, pages 10130–10135, Munich, Germany, March 2003.

[82] Prih Hastono, Stephan Klaus, and Sorin A. Huss. An Integrated SystemC Framework for Real-Time Scheduling Assesments on System Level. In *25th IEEE International Real-Time Systems Symposium (RTSS 2004)*, Lisbon, Portugal, December 2004.

[83] Prih Hastono, Stephan Klaus, and Sorin A. Huss. Real-Time Operating System Services for Realistic SystemC Simulation Models of Embedded Systems. In *Forum on Specification & Design Languages (FDL'04)*, pages 380–391, Lille, France, September 2004.

[84] Z. He, A. Mok, and C. Peng. Timed RTOS Modeling for Embedded System Design. In *Proceedings of the 11^{t}h IEEE Real Time Embedded Technology and Applications Symposium (RTAS)*, pages 448–457, March 2005.

[85] F. Hessel, V. da Rosa, C. Reif, C. Marcon, and T. Dos Santos. Scheduling Refinement in Abstract RTOS Models. *ACM Transactions on Embedded Computing Systems*, 5(2) :68–69, May 2006.

[86] F. Hessel, V. da Rosa, I. Reis, C. Marcon, and A. Susin. Abstract RTOS Modeling for Embedded Systems. In *Proceedings of the 15^{t}h IEEE International Workshop on Rapid System Prototyping (RSP)*, Geneva, Switzerland, June 2004.

[87] F. Hofmann, C. Hansen, and W. Schafer. Digital Radio Mondiale (DRM) digital sound broadcasting in the AM bands. *IEEE Trans. on Broadcasting*, 49(3) :319–328, Sept. 2003.

[88] Joint Tactical Radio System (JTRS) Joint Program Office (JPO). Software Communications Architecture Specification v2.2.2. available at `http://jtrs.spawar.navy.mil/sca/`.

[89] J. J. Labrosse. *MicroC/OS-II, The Real-Time Kernel*. CMP Books, Lawrence, Kansas, USA, 2002.

[90] R. Le Moigne, O. Pasquier, and J-P. Calvez. A Generic RTOS Model for Real-time Systems Simulation with SystemC. In *Proceedings of Design and Test in Europe (DATE)*, volume 3, pages 82–87, Paris, France, February 2004.

[91] S. Leprêtre, P. Gaussier, and J.P. Cocquerez. From Navigation to Active Object Recognition. In *Proceedings of the Sixth International Conference on Simulation for Adaptive Behavior (SAB)*, pages 266–275, Paris, France, 2000.

[92] J. Madsen, K. Virk, and M. Gonzales. Abstract RTOS modeling for multiprocessor system-on-chip. In *Proceedings of the International Symposium on System-on-Chip*, pages 147–150, November 2003.

[93] M. Maillard, O. Gapenne, P. Gaussier, and L. Hafemeister. Perception as a dynamical sensori-motor attraction basin. In M. Capcarrere et al., editor, *Advances in Artificial Life (8th European Conference, ECAL)*, volume LNAI 3630 of *Lecture Note in Artificial Intelligence*, pages 37–46. Springer, sep 2005.

[94] Joseph Mitola. *Software Radio Architecture : Object-Oriented Approaches to Wireless Systems Engineering*. Wiley Interscience publisher, November 2000.

[95] OMG. MDA Guide, version 1.0.1 . available at `http://www.omg.org/mda/`, June 2003.

[96] OMG. CORBA 3.0.3, Common Object Request Broker Architecture (Core Specification), 2004-03-01, March 2004.

[97] J. Quévremont, M. Sarlotte, and B. Candaele. Démarche de conception d'une plateforme monopuce de réception de radiodiffusion numérique DRM. *Annales des Télécommunications*, 59(7-8), Juillet-Août 2004.

[98] Jeffrey H. Reed. *Software Radio : A Modern Approach to Radio Engineering*. Prentice Hall, 1st edition, May 2002.

[99] A. Segard and F. Verdier. SOC and RTOS : Managing IPs and tasks communications. In *Field-Programmable Logic and its Applications : 14th International conference (FPL)*, pages 710–718, Antwerp, Belgium, September 2004.

[100] F. Verdier, J-C. Prévotet, A. Benkhelifa, D. Chillet, and S. Pillement. Exploring RTOS issues with a high-level model of a reconfigurable SoC platform. In *European Workshop on Reconfigurable Communication-centric SoC (ReCoSoC 2005)*, Montpellier, France, June 2005.

[101] François Verdier, Mickaël Maillard, and Benoît Miramond. Designing a custom SoC for robotic vision : the problem of modelling the embedded real-time operating system. In *EURASIP Journal on Applied Signal Processing*, Soumis en septembre 2006.

[102] J. Walko. Cognitive radio. *IEE Review*, 51(34–37), May 2005.

[103] Haobo Yu and Daniel Gajski. RTOS Modeling in System Level Synthesis. Technical Report 02-25, University of California, Irvine, USA, August 2002.

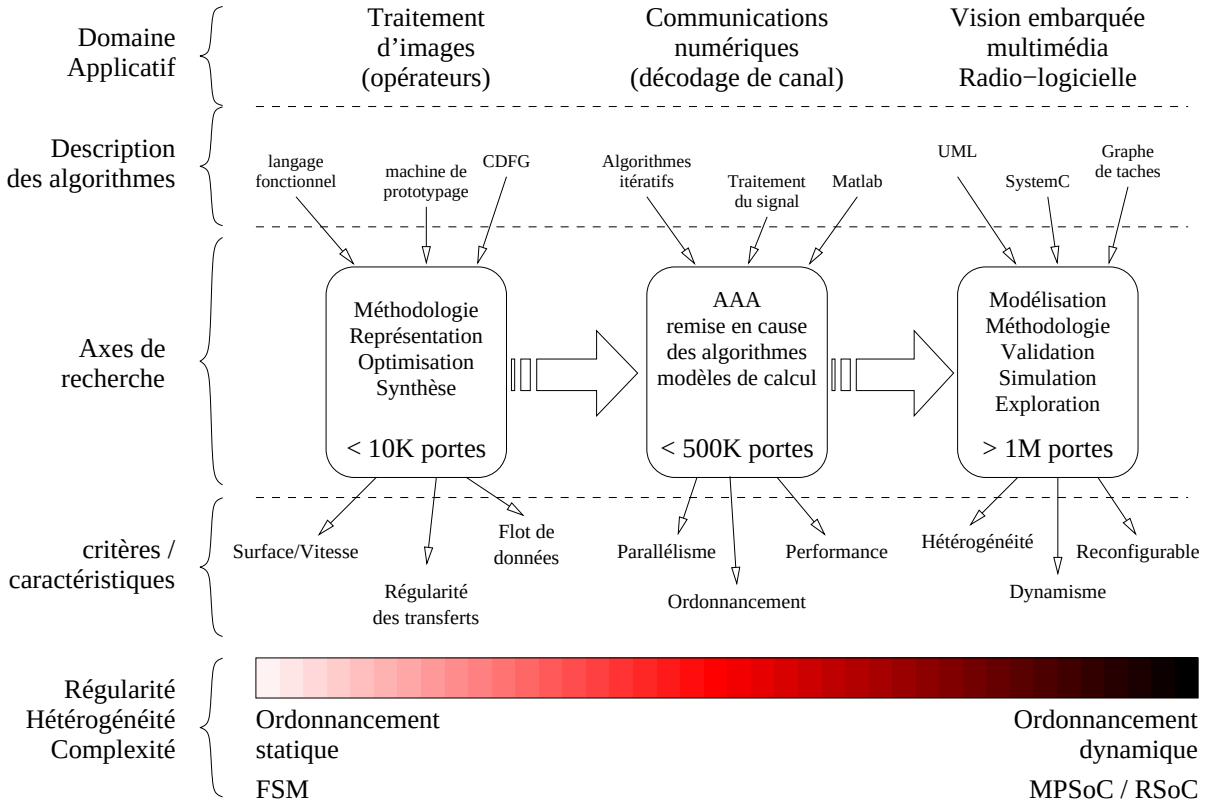
[104] Q. Zhang, G.J.M. Smit, L.T. Smit, A. Kokkeler, R.W. Hoeksema, and M. Heskamp. A reconfigurable platform for cognitive radio. In *2nd International Conference on Mobile Technology, Applications and Systems*, Nov. 2005.

4

Bilan et Perspectives de Recherche

4.1 Bilan scientifique

Pour conclure la synthèse de mes travaux, je présente ici la mise en perspective de l'ensemble de mes activités de recherche, incluant mes travaux de doctorat. Je précise que, dans mon analyse, la chronologie exacte des différents projets sur lesquels j'ai pu travailler n'est pas fondamentale, même si elle est respectée.



J'exerce mes activités scientifiques depuis plus de dix ans dans le domaine de la concep-

tion d'architectures numériques optimales pour résoudre des problèmes applicatifs précis. Dans le cadre de mon doctorat, le problème posé était celui de la conception **automatique** d'opérateurs de traitement d'image (ce que l'on qualifie de *synthèse d'architecture*). Le fait qu'à l'époque on cherchait à construire ces circuits VLSI, de manière automatique et par l'application d'heuristiques optimisant de nombreux critères, est accessoire dans mon analyse. En revanche, le point intéressant est qu'il était apparu qu'exploiter un certain degré de régularité dans les traitements à intégrer pouvait être utilisé pour optimiser la réalisation matérielle de ces traitements. Ces travaux ont eu lieu au début des années 1990 et, à l'époque, les outils et les langages utilisés pour décrire les applications exhibaient eux aussi une certaine régularité : langages fonctionnels, graphes de contrôle et de flot de données (CDFG), machines à états finis.

Dans le cadre de mes travaux sur les décodeurs LDPC, la conception de leurs architectures s'est trouvée, sur de nombreux aspects, confrontée à un certain manque de régularité. Les matrices de décodage, à cause de leurs propriétés pseudo-aléatoires, imposent principalement dans les architectures des transferts de données souvent très irréguliers. On peut noter d'ailleurs que dans une approche de conception conjointe des codes et des architectures de décodeurs, les architectes tentent de réintroduire une certaine dose de déterminisme pour optimiser les structures matérielles. D'autre part, on peut constater qu'un second facteur architectural devient prépondérant, l'augmentation très importante de degré de parallélisme. Enfin, comme je l'ai mentionné en tant que perspective de cette activité, une approche prometteuse pour franchir un nouveau cap de performance pourrait consister à adopter un modèle architectural un peu moins "classique" et envisager les architectures de type *fermes de processeurs* afin de supporter et d'exploiter au maximum, cette fois, l'irrégularité des matrices et le dynamisme des algorithmes de décodage.

Cet aspect de dynamisme dans les traitements et dans les architectures correspondantes – cas des machines à reconfiguration partielle dynamique ou celui des plates-formes de radio-logicielle – est justement un facteur nouveau auquel on s'intéresse aujourd'hui. La classe des applications visées (vision active, radio cognitive, plates-formes multi-applications) se porte de plus en plus difficilement sur les modèles architecturaux habituels (parallélisme, pipeline, SIMD / MIMD, modèles synchrones et autres). La mise en échec des méthodes de conception et l'évolution de la technologie nous pousse à développer des plates-formes multi-processeurs, hétérogènes et souvent reconfigurables. Il n'y a plus, dans ce cas, la moindre régularité à exploiter et la plupart du temps, les comportements espérés de ces futurs systèmes sont flexibles, dynamiques et adaptables (voire auto-adaptables).

4.2 Perspectives

J'ai déjà détaillé dans ce document les suites possibles de mes différents projets (pages 46, 56, 81 et 92). Certaines de ces pistes sont des perspectives à court terme dans la mesure où elles constituent les objectifs de travaux en court ou à venir, notamment dans le cas des thèses de Grégory Gailliard sur la réalisation matérielle des mécanismes CORBA et de Emmanuel Huck sur l'exploration du service de gestion des tâches d'un OS distribué hétérogène.

La suite discute de perspectives à plus long terme des mes activités.

La première conclusion de l'analyse des mes travaux est que, à la faveur de nouvelles applications possédant de nouvelles caractéristiques, j'ai été amené régulièrement à développer de nouveaux concepts architecturaux qui m'inspirent aujourd'hui de nouvelles opportunités sur des cas déjà traités. Revisiter, par exemple, les architectures de décodeurs LDPC avec des modèles architecturaux dynamiques, flexibles et, pourquoi pas, reconfigurables en ligne peut constituer de nouveaux axes de recherche prometteurs. Fournir à un récepteur de communication des capacités d'auto-adaptation aux conditions du canal et du contexte applicatif est un défi architectural et algorithmique qui semble à notre portée aujourd'hui. Les solutions à explorer consisteraient probablement à abandonner l'ordre partiel et statique des calculs, le découpage statique en partitions parallèles et pourquoi pas l'adaptation dynamique des équations de décodage. Quelques travaux dans ce sens explorent déjà des techniques de quantification non linéaire des calculs et méritent certainement d'être étendus.

La deuxième conclusion est que nous sommes aujourd'hui, probablement plus que jamais d'ailleurs, face à un mur architectural et technologique : les outils de conception, les langages de description comme les modèles de calcul actuels sont inefficaces pour résoudre les problèmes du traitement embarqué optimal. Cet échec est certainement à mettre sur le compte des multiples contraintes orthogonales qui s'imposent à ces systèmes : performance maximale, consommation d'énergie minimale, flexibilité, *time to market*, généricité, robustesse aux pannes et aux fautes, etc.

En ce qui concerne les outils de conception, on met en doute aujourd'hui l'efficacité des approches par synthèse automatique dans le cas des circuits SoC très complexes. La complexité et l'hétérogénéité des problèmes posés ne peut plus se résoudre par une heuristique, si complexe soit-elle. Les approches défendues aujourd'hui se basent plutôt soit sur l'exploration et le raffinement (manuel mais formalisé) des spécifications, soit par des mises en correspondance de modèles, des transformations de modèles et des manipulation de méta-modèles (comme les travaux autour des approches MDA/MDE par exemple). Les outils de synthèse de haut niveau et de génération automatique de code n'ont pas disparu mais tendent à se limiter uniquement à la conception de sous-parties de ces systèmes.

Les spécifications et les langages qui supportent la description des systèmes s'éloignent de plus en plus du traditionnel code VHDL. On s'aperçoit, par exemple, que pour des applications de radiocommunication – pourtant traitées pendant longtemps avec les outils classiques – on adopte des description XML, des diagrammes de classes UML, du code C++ et des spécifications orientées objet ! Ceci s'explique bien sûr par le fait que le déploiement des applications s'envisage aussi bien de manière logicielle que matérielle.

C'est finalement sur ce dernier point que de nouvelles innovations peuvent avoir lieu : comment concilier des modèles de calculs si différents que le calcul programmé et le calcul câblé ? La solution qui semble s'imposer est d'exploiter la reconfiguration des structures de calcul. Du coté des circuits reconfigurables (quelque soit le grain de la reconfiguration), on a cherché à appliquer sur des structures câblées le concept de la programmation sans résoudre pour le moment d'ailleurs le problème du modèle de calcul qui peut correspondre. Du coté des processeurs programmables, les avancées les plus originales consistent à doter ces unités de capacités d'implémentation matérielle de portions de code grâce à des zones reconfigurables (Tensilica Xtensa, Philips TriMedia, Chameleon, etc).

Sans chercher à inventer un nouveau modèle de calcul universel, c'est dans cet axe que j'envisage les perspectives de mon travail scientifique. Plus précisément, mes travaux les plus récents me permettent d'envisager l'élaboration des principes matériels et logiciels ainsi que des techniques de modélisation et de réalisation qui permettraient de doter les futurs systèmes de traitement de capacités d'auto-réorganisation de leurs structures de calcul pour développer des architectures de type *polymorphiques*. Pour illustrer de telles architectures, je ne prendrai qu'un exemple des pistes que nous avions dressé à l'occasion du démarrage de notre projet OverSoC. Une architecture RSoC composée d'au moins un processeur de calcul et d'une zone reconfigurable dynamiquement est capable de créer, à la demande, de multiples cœurs de processeurs (*soft-cores*) – donc d'augmenter son propre taux de parallélisme – et de redéployer ses calculs sur ces nouvelles unités. Le redéploiement et le ré-ordonnancement en temps-réel d'une application sur une architecture de ce type pose de nombreux problèmes à la fois techniques et de formalisation. C'est technologiquement possible mais les problèmes scientifiques sous-jacents sont nombreux et restent à résoudre.

Résumé

Les travaux de recherche dont la synthèse est présentée dans ce document portent sur deux aspects de la conception d'architectures numériques embarquées pour des applications de traitement de l'information. Le premier axe concerne l'étude et la conception de modèles architecturaux pour les décodeurs de canal utilisés dans les communications numériques. Les décodeurs étudiés sont basés sur les codes LDPC (Low Density Parity Check codes) qui, depuis quelques années, sont proposés comme codes correcteurs d'erreurs dans plusieurs normes de transmission. On s'intéresse en particulier à la norme DVB-S2 de radio-diffusion de programmes multimédia. Ces architectures de décodeurs mettent en oeuvre des algorithmes dont les réalisations matérielles reposent sur une adéquation fine entre le taux de parallélisme, l'ordonnancement des calculs et les quantités de ressources nécessaires. Une étude sur la réduction de complexité des algorithmes de décodage LDPC non binaires, préalable à la définition d'une architecture associée est également présentée.

Le deuxième axe de recherche étend la problématique aux architectures très fortement intégrées, de type SoC (systèmes sur puces), et qui disposent de capacités de flexibilité, d'adaptabilité et de reconfiguration matérielle dynamique. La présence d'un système d'exploitation temps-réel embarqué devient alors nécessaire pour gérer de telles architectures et rend inadaptées les méthodes classiques de conception. Le deuxième axe des travaux porte sur de nouvelles méthodologies d'exploration et de conception d'architectures reconfigurable. Le cas de la modélisation des systèmes d'exploitation embarqués est abordé ainsi que le cas de la conception des applications et plates-formes pour la radio-logicielle.

Mots-clés: Conception d'architectures embarquées, décodeur LDPC, architectures reconfigurables, méthodologies de conception, systèmes d'exploitation embarqués, modélisation à haut niveau

Abstract

The research work presented in this document addresses two different problems of designing embedded digital architectures for information processing applications. The first axis concerns the study and design of architectural models for channel decoding systems used in digital communication applications. These decoders are based on LDPC (Low Density Parity Check) codes that are currently proposed as error correcting codes in several transmission standards. We are notably interested in the wireless digital video broadcast standard DVB-S2. The hardware implementation of decoding algorithms used in these architectures necessitates a fine tradeoff between the parallelism degree, computation scheduling and available resources. Another study on reducing the complexity of the decoding algorithms working on non binary codes is also presented, and could help to define the associated architecture.

The second research axis generalises the problem to the design of highly integrated SoC (System-on-Chip) architectures exhibiting flexibility, adaptability and dynamic hardware reconfiguration capacities. The use of an embedded real-time operating system becomes necessary for managing such architectures. Designing these architectures by using classical design methods would thus fail. The second part of the work concerns new exploration and design methodologies for reconfigurable architectures. The problem of modelling embedded operating systems and the design of SDR (Software-Defined Radio) applications and platforms are presented.

Keywords: Embedded architecture design, LDPC decoders, reconfigurable architectures, design methodologies, embedded operating systems, high-level modelling