Etude du bloc de réception dans un terminal UMTS-FDD et développement d'une méthodologie de codesign en vue du fonctionnement en temps réel.

Eric Batut

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THESE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : Signal, Image, Parole, Télécom
préparée au laboratoire **LIS de Grenoble**
dans le cadre de l'Ecole Doctorale
Electronique, Electrotechnique, Automatique, Télécommunications, Signal
présentée et soutenue publiquement

par

Eric BATUT

le Lundi 3 Juin 2002

Etude du bloc de réception dans un terminal UMTS-FDD et développement d'une méthodologie de codesign en vue du fonctionnement en temps réel.

Directeur de thèse : Geneviève JOURDAIN

Composition du Jury :

M.	Jean-Marc DOLMAZON	Président
M.	Pierre DUHAMEL	Rapporteur
M.	Eric MARTIN	Rapporteur
Mme	Geneviève JOURDAIN	Directeur de thèse
Mme	Marylin ARNDT	Co-directeur de thèse
M.	Jean-Claude BIC	Examinateur

Remerciements

En premier lieu, je tiens à remercier chaleureusement Patrice Senn et Marylin Arndt de m'avoir accueilli au sein du laboratoire DIH/OCF, anciennement DTM/CET, et de m'y avoir offert un environnement de travail extrêmement enrichissant.

Merci à Geneviève Jourdain d'avoir accepté d'encadrer ce travail de thèse à distance en plus de ses autres charges au sein du laboratoire LIS, et d'avoir fait le nécessaire pour faciliter autant que possible la laborieuse finalisation de ce mémoire.

Merci à Eric Martin et Pierre Duhamel d'avoir accepté d'être rapporteurs de ce travail, ainsi qu'à Jean-Claude Bic et Jean-Marc Dolmazon d'avoir accepté de faire partie du jury.

Merci à tous les thésards (Yvan, Pascal, Christophe, Thierry, Sandrine, Alexandre, Hélène, Sébastien, Joumana, Laurent, Bruno, Vincent, Fayçal, Carolynn, David et Salvatore) et stagiaires (Myriam, Yves (l'homme polyphase), Mathieu, Vincent le movie-addict, Dimitri, Jean, Florent, Jérôme et *last but not least*, Stéphane) ayant transité par DTM/CET puis DIH/OCF d'avoir entretenu et de continuer à entretenir un espace d'échange et de discussion aussi bien technique que non-technique, ainsi qu'à Cathy et Martine (avec une pensée pour Annick) pour leur essentielle contribution au fonctionnement général du laboratoire.

Enfin, sans ordre de préférence, et sûrement avec quelques oublis, merci à ma famille et à ma belle-famille pour l'encouragement et le support actif, à Benoît pour son talent de relecteur et sa généreuse contribution à l'atmosphère du bureau E002, aux Miscopes en général pour les trips Ikéa, aux Roux-Rob' pour les déménagements, aux Dabés pour les apéros, aux Ducs pour le Jorky, aux Ugrantes, à Yvan et au Mab' pour le squash, à Lionel et Peggy pour le wooby-call, aux Fraissinets, à Joke et Rabia pour la soirée at Home, à Mit et Milooze pour la demi-beubar, à Olibli (champion de tennis) pour ses mails désinhibants, à Manoul pour la coupe disco de sa fille, à Stéph (la Fouine "foot on the desk + œil révulsé") et Anne-Laure (mon royaume pour une Lotus Elise) pour les pixel sessions (spéciale dédicace à Gontso et à "Il est pas bon ton café !"), à la Voix du Denis (merci la chaîne rouillée), et enfin à Kevin, Steven, Abdel-Aziz et Katja de la Place d'Clichy pour des éclats de rire téléphoniques réitérables à l'infini.

Enfin, un immense merci à Caroline et Augustin pour m'avoir supporté à travers tout ça, et surtout pour tout le reste.

Résumé

L'UMTS est un nouveau standard de radiocommunications mobiles destiné à résoudre les problèmes des actuels réseaux de deuxième génération, proches localement de la saturation et limités dans leur offre de services multimédias par les faibles débits utiles supportés. L'UMTS représente une rupture technologique importante et nécessite un effort particulier pour la réalisation des équipements, car la complexité des traitements à effectuer a augmenté dans des proportions considérables. Les terminaux 3G, par exemple, devront embarquer une puissance de calcul supérieure de plus d'un ordre de grandeur à celle embarquée par leurs prédécesseurs.

Après avoir introduit l'UMTS et une de ses interfaces radios, le Wideband CDMA, nous avons identifié l'estimation par le terminal du canal radiomobile par lequel a transité le signal émis par la station de base comme étant une des tâches susceptibles d'entraîner le plus grand nombre d'opérations à effectuer. Une solution originale à ce problème est proposée sous la forme d'un algorithme d'estimation itérative de canal à suppression de trajets. La complexité calculatoire de cet algorithme a l'inconvénient majeur de varier avec le carré du facteur de suréchantillonnage, ce qui empêche de travailler avec une valeur élevée de celui-ci, et par conséquent ne permet pas d'obtenir une grande précision quant à l'estimation des instants d'arrivée des trajets. Ce problème est résolu en introduisant une version optimisée de cet algorithme, dont la complexité varie linéairement avec le facteur de suréchantillonnage. Conserver une complexité raisonnable tout en travaillant avec des facteurs de suréchantillonnage élevés devient réaliste, ce qui permet d'accéder à coût égal à une précision plus élevée qu'avec l'algorithme origininal. De plus, cette optimisation simplifie les opérations élémentaires effectuées par l'algorithme, ce qui a pour conséquence de rendre son implémentation sur une architecture hybride matérielle-logicielle plus efficace que son implémentation sur un seul processeur de signal.

Une méthodologie de conception au niveau système est ensuite proposée pour réaliser cette architecture hybride dans un but de prototypage rapide. Cette méthodologie, bâtie autour du logiciel N2C, de la société CoWare, utilise un langage de haut niveau, surensemble du langage C auquel ont été rajoutées les constructions nécessaires pour décrire des architectures matérielles. L'algorithme est partitionné en une partie logicielle s'exécutant sur un cœur de DSP ST100 et un coprocesseur réalisé en logique câblée. De sévères incompatibilités logicielles ont empêché la réalisation de cette architecture hybride selon la méthodologie proposée, mais des résultats intéressants ont néanmoins été obtenus à partir d'une implémentation purement logicielle de l'algorithme proposé. L'architecture obtenue avec l'application des premières étapes de la méthodologie proposée à l'algorithme d'estimation de canal est décrite, ainsi que quelques suggestions faites à la société CoWare, Inc. pour l'amélioration de leur outil. Enfin, l'adéquation de la méthodologie proposée à un environnement de prototypage rapide est discutée et des pistes pour la réalisation d'un éventuel démonstrateur sont données.

Abstract

UMTS is a new radiocommunication standard aimed at solving today's second generation networks' problems that are local saturation and slow bitrates. UMTS and previous networks sharing close to nothing regarding the radio link, network equipments as well as user equipments must be rebuilt from scratch. User equipments, in particular, must embed a lot more processing power than their older counterparts.

After having introduced UMTS and the first to-be-deployed of its radio interfaces, the Wideband CDMA, we determined that the channel estimation prior to the Rake combining, assuming the receiver uses this well-known receiver structure, is the most complex task to be performed by the user equipment. An algorithmic solution to this problem is proposed through an iterative channel estimation algorithm which suppresses identified paths before trying to find new ones in the considered time window. This algorithm has a major drawback : its computational complexity varies quadratically with the oversampling factor, and thus forbids to work with high oversampling factors, which are the key to a precise channel delay estimation. An optimized version of this algorithme is proposed, whose complexity scales linearly with the oversampling factor, against a quadratic variation for the original one, and who does not cause a noticeable performance loss. The optimized channel estimation algorithm is therefore suitable for a constrained environment such as a user equipment. Furthermore, the performed optimization has the side effect of making the proposed algorithm much more suitable for a hybrid hardware-software implementation than for a pure software one.

A system-level design flow is then proposed to realize this hybrid architecture while keeping a fast prototyping approach in mind. This methodology revolves around CoWare's N2C environment and uses a superset of the C language called CoWareC. CoWareC includes several constructs needed to describe partitionned systems and hardware structures. The proposed algorithm is mapped onto a hybrid architecture composed of a ST100 DSP core and a hardware coprocessor. Due to major software issues, this hybrid implementation could not be developped following the proposed methodology, but interesting results were nonetheless obtained from a pure software implementation. The application of the first steps of the proposed methodology to the channel estimation algorithm yielded interesting results and suggestions for the improvement of the N2C tool. Then, hints are given for the development of a real hybrid prototype and the adequation of the proposed methodology to a fast prototyping environment is discussed.

Table des matières

Table des figures

Liste des tableaux

Introduction

Motivation

Le marché des radiocommunications mobiles est en croissance soutenue depuis plusieurs années, et le taux de pénétration des terminaux mobiles en France est aujourd'hui supérieur à 60 %. La situation est cependant loin d'être idyllique. Le réseau GSM, victime de son succès, souffre de saturation prononcée dans certains centres urbains, et n'offre que des débits dérisoires à l'utilisateur désireux de transférer des données.

Le problème étant sensiblement le même dans de nombreux pays, l'Union Internationale des Télécommunications a lancé au début des années 90 un appel d'offre pour le prochain standard de télécommunications mobiles cellulaires, dénommé IMT-2000. La réponse européenne à cet appel d'offre est l'UMTS, qui doit à long terme supplanter les réseaux de deuxième génération, en offrant des débits bien supérieurs à ceux-ci, une interopérabilité accrue, et surtout une capacité en termes de nombre d'abonnés les mettant à l'abri de tout problème de saturation.

Dans la bande dite appairée, libérée et ayant été vendue sous licence aux opérateurs européens, va être déployée la première version de l'UMTS, dite UMTS-FDD, aussi appelée du nom de son interface radio, le Wideband CDMA (WB-CDMA). La couche physique de ce nouveau standard employant des technologies très différentes de celles utilisées par GSM, il est nécessaire pour les équipementiers de développer de nouveaux terminaux embarquant toutes les fonctionnalités prévues par la norme, et, pour les opérateurs, d'acquérir la connaissance de l'architecture de ces terminaux qui va leur permettre de proposer une gamme pertinente de services réalistes et adaptés à l'environnement matériel de leur exécution.

Ce travail de thèse s'est déroulé dans le laboratoire DIH/OCF (Direction des Interactions Humaines / Objets Communicants et Faisabilité de systèmes) de France Télécom R&D, dans lequel les fonctions critiques et la complexité des terminaux mobiles sont examinées, ainsi que leur éventuel degré d'intégration.

Après un tel examen du lien descendant de l'interface radio de l'UMTS-FDD, les travaux rapportés ici se sont orientés vers le développement d'un algorithme d'estimation de canal *raisonnablement embarquable*. En effet, ce point est très souvent négligé par les nombreuses publications sur l'égalisation, alors que l'estimation de canal constitue justement une de ces tâches critiques, qui influe sur le reste du traitement de réception si elle est mal effectuée ou si elle n'est pas effectuée du tout.

Les terminaux évoluant de manière à incorporer des puissances de calcul sans cesse croissantes, les *méthodologies de conception* de ces équipements évoluent elles-aussi. La complexité des algorithmes à embarquer d'une part et le besoin de flexibilité dans le cadre d'une norme non finalisée d'autre part rendent délicate l'obtention d'un partitionnement qui resterait pertinent

au cours du cycle de développement et *après* le déploiement des équipements.

Le caractère hétérogène des architectures embarquées nécessitant l'adoption de nouvelles méthodologies de conception qui intègrent les fonctionnalité de cosimulation et de synthèse d'interface, le présent travail de thèse a aussi abordé ces nouvelles méthodologies de conception et leur adéquation à un environnement de prototypage rapide. En effet, la plupart de ces méthodologies ayant pour but la conception d'un système-sur-puce complet, il est nécessaire de procéder à certains ajustements dans l'utilisation de ces outils dès lors qu'on ne s'intéresse qu'à un *prototype* des blocs fonctionnels incriminés.

Contribution

Ce travail de thèse comporte un volet algorithmique et un volet plus méthodologique ayant rapport au domaine de la conception microélectronique.

La principale contribution algorithmique a été le développement d'un algorithme itératif d'estimation de canal à suppression de trajet. Cet algorithme est chargé de l'identification des trajets secondaires crées par la propagation dans le canal radiomobile. La complexité de cet algorithme s'étant révélée trop élevée, une version optimisée en a été développée sans dégradation apparente des performances, une étude poussée des performances de ces deux algorithmes restant à effectuer. Cette étude sera réalisée à l'aide du prototype devant être développé à l'aide de la méthodologie proposée dans la deuxième partie de cette thèse, afin de mimer aussi fidèlement que possible l'architecture d'accueil de l'application, à savoir celle d'un terminal de troisième génération. Un critère innovant d'identification des faux trajets est également proposé. Ce critère est basé sur une évaluation de la qualité de chaque trajet identifié par l'intermédiaire du nombre de symboles potentiellement démodulables sans erreur à partir du trajet en cours d'identification.

Une méthodologie innovante de conception au niveau système représente le deuxième « volet » de ce travail. Cette méthodologie s'appuie sur une temporisation progressive du comportement des blocs composant le système. Utilisant le logiciel N2C de la société CoWare pour faciliter l'exploration architecturale et l'obtention d'un partitionnement pertinent, elle est basée sur un langage de haut niveau dérivé du langage C et favorise la migration du code déjà existant d'un niveau d'abstraction à un autre plutôt que sa réécriture. Les points critiques de l'application de cette méthodologie dans le but du prototypage rapide sont identifiés pour un environnement de prototypage donné, et des solutions à ces problèmes sont proposées.

Structure du mémoire

Le présent mémoire de thèse est divisée en deux parties.

La première partie présente le cadre applicatif et les développements algorithmiques ayant été effectués. Le chapitre I décrit l'UMTS en tant que norme de téléphonie mobile de troisième génération, et présente les innovations apportées par ce nouveau standard en comparaison des systèmes précédents. Le lien descendant, qui constitue le facteur limitant de la transmission, est détaillé, ainsi que les divers systèmes de codage utilisés dans l'interface radio.

Le chapitre II traite quant à lui de l'estimation de canal au sein du terminal. Les défauts introduits par la propagation dans le canal radiomobile sont présentés, avant de souligner l'im-

portance de l'estimation de canal dans le cadre d'un système CDMA. Après un bref examen de l'état de l'art de l'estimation de canal, un algorithme itératif à faible complexité est proposé, basé sur la suppression des divers trajets identifiés au cours du processus d'estimation. Cet algorithme est ensuite optimisé puis comparé à l'algorithme original. La version optimisée faisant intervenir des opérations réalisables efficacement par un opérateur câblé, la décision est prise d'implémenter cet algorithme sur une architecture hétérogène comprenant un DSP et un opérateur câblé.

La deuxième partie traite de la conception des systèmes embarqués tels que celui proposé dans la première partie. Après avoir présenté les méthodologies classiques de conception et leurs limitations, le chapitre III illustre l'impact des fonctionnalités telles que la cosimulation ou la synthèse d'interface sur les flots de conception dits « au niveau système », qui partent d'une vue comportementale du système avant d'évoluer vers une vue temporisée et partitionnée du même système. Une méthodologie innovante est alors proposée dans le cadre d'une application à forte composante « signal ».

Le chapitre IV présente l'implementation entièrement logicielle qui a été faite de l'algorithme développé, faute d'avoir pu appliquer en totalité la méthodologie proposée, du fait de sévères incompatibilités logicielles. Enfin, l'adéquation de la méthodologie proposée à un environnement de prototypage rapide détaillé dans ce chapitre est discutée, et les points critiques de la réalisation d'un tel prototype à partir d'une chaîne de cosimulation sont identifiés.

Première partie

L'estimation de canal au sein du terminal UMTS-FDD

Chapitre I

L'UMTS : la téléphonie mobile « 3G »

On entend souvent parler de l'UMTS comme une norme de téléphonie mobile cellulaire de troisième génération. L'objectif de ce chapitre est de préciser le sens des termes «troisième génération» et de répondre aux questions qu'ils suscitent : quelles étaient les générations précédentes, quelles sont les nouveautés apportées par l'UMTS, quelles sont les spécificités de l'interface radio de l'UMTS, en quoi diffère-t-elle de celle de GSM ...

La première partie de ce chapitre, après avoir rappelé brièvement les principales étapes du développement de la téléphonie mobile, présente les possibilités offertes par l'UMTS, tant du point de vue de l'utilisateur que de celui plus rarement exposé de l'opérateur. La deuxième partie de ce chapitre détaille les mécanismes spécifiques du lien radio de l'UMTS : la ou les techniques de duplex et d'accès multiple employées, la technique d'étalement de spectre mise en œuvre, et enfin le format des données circulant sur le lien descendant[1].

I.1 Qu'est-ce que la « 3G » ?

I.1.1 Rappels historiques et généralités

Bien qu'issue de travaux vieux de plus de 100 ans, la radiotéléphonie ne connaît son heure de gloire que depuis peu, et est encore en pleine croissance aujourd'hui. Les avancées théoriques et techniques ayant permis le développement de la radiotéléphonie telle que nous la percevons aujourd'hui sont les suivantes :

- En 1876, le canadien Graham Bell invente le téléphone : une communication est possible entre deux postes reliés par deux fils de cuivre (la paire téléphonique).
- En 1887, l'allemand Heinrich Hertz découvre les ondes radio, qui porteront son nom par les suites : ce sont les ondes hertziennes.
- En 1896, Guglielmo Marconi réalise la première transmission radio dans son grenier : il commande une sonnette éléctrique à quelques mètres de distance.
- De 1897 à 1901, Marconi séjourne en Angleterre où il réalise la première liaison radio transatlantique entre les Cornouailles et Terre-Neuve, en 1901. Il reçoit le prix Nobel de physique en 1909 pour ses travaux sur les ondes hertziennes.

[1] Les stations de base étant traditionnellement situées en hauteur pour avoir une meilleure «visibilité», la communication de la station de base vers le mobile est appelée lien *descendant* (downlink, en anglais), tandis que la communication du mobile vers la station de base est appelée lien *montant* (uplink en anglais).

Dès le début du 20ème siècle, les services de police de différents pays d'Europe et d'Amérique du Nord se dotent de moyens radio pour communiquer avec des véhicules en patrouille, mais ce n'est qu'en 1950 que le radiotéléphone se banalise : c'est en effet à cette date que la compagnie Bell Telephone propose à ses abonnés un service de téléphonie mobile. Mais le réseau, dont l'expansion est limitée par le nombre de fréquences radio disponibles, ne peut accueillir qu'un nombre limité d'abonnés.

En 1964, le concept de partage des ressources est introduit dans les réseaux de radiotéléphonie : le réseau alloue dynamiquement un canal radio à une nouvelle communication pendant sa durée. Le système choisit dans l'ensemble des canaux libres une fréquence qu'il attribue à une nouvelle communication. La gestion des fréquences, jusque-là statique, devient dynamique : un réseau peut désormais compter plus d'abonnés que de canaux radio.

I.1.1.1 Le réseau cellulaire

En 1971, aux Etats-Unis, la compagnie Bell Telephone présente, en réponse à une demande sans cesse croissante, le concept de *réseau cellulaire* et teste, à partir de 1978, un système de radiotéléphonie mettant en œuvre ce concept : le système AMPS (Advanced Mobile Phone System).

Le principe sous-jacent au réseau cellulaire est la division d'une zone géograhique en plusieurs zones de taille inférieure, non recouvrantes (ou alors le moins possible), appelées *cellules*. Ainsi, là où le réseau non-cellulaire couvre une surface S, le réseau cellulaire couvre la même surface, mais avec N cellules de surface S/N.

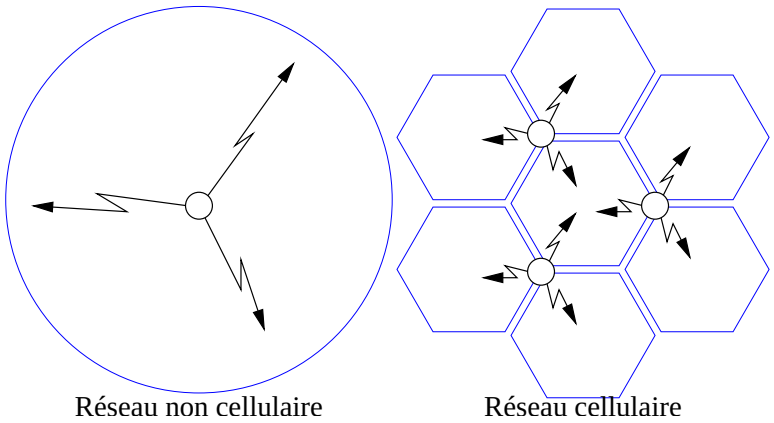


FIG. I.1 – Topologies de réseau

Ce découpage, malgré le fait qu'il entraîne l'augmentation du nombre de stations de base (les émetteurs/récepteurs situés aux intersections des cellules), offre surtout la possibilité d'optimiser l'utilisation de la ressource fréquentielle allouée au réseau. En effet, si deux stations de base proches utilisent le même ensemble de fréquences, leurs émissions vont se parasiter et créer de l'*interférence inter-cellules*. Le nombre de mobiles qu'elles pourront gérer correctement sera alors réduit. La multiplication des stations de base permet de réduire leur puissance d'émission et de répartir plus finement les fréquences utilisées, permettant ainsi leur réutilisation dans des cellules non-adjacentes. De cette manière, l'interférence inter-cellules est réduite et la capacité du réseau augmentée. Le motif de réutilisation des fréquences doit être judicieusement choisi de

manière à limiter autant que possible l'interférence inter-cellules tout en utilisant au maximum la bande de fréquence allouée au réseau. Sur la figure I.2, la ressource fréquentielle a été divisée en 3 groupes de fréquences, on dit que le facteur de réutilisation (le *reuse factor*, en anglais) est égal à 3.

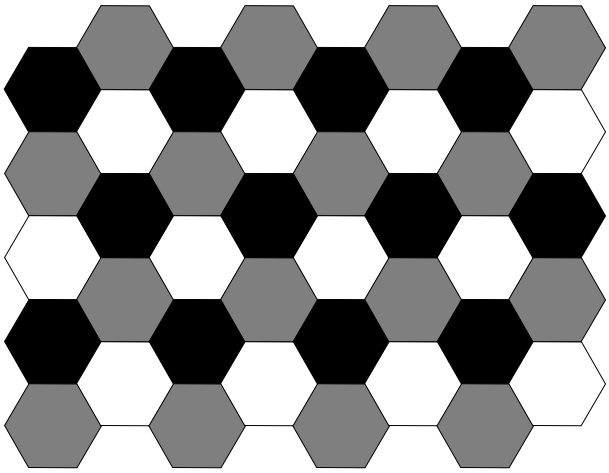


FIG. I.2 – Exemple de motif de réutilisation de la ressource fréquentielle : motif à 3 sous-ensembles

En contrepartie, le réseau doit être capable de gérer correctement le passage d'un mobile d'une cellule à une autre (pour la téléphonie mobile, on parle alors de *handover*) : quand un abonné veut émettre un appel, la station lui attribue une fréquence d'émission, et lorsque l'abonné change de cellule, il passe sous le contrôle d'un autre émetteur, lequel lui donne une nouvelle fréquence d'émission éventuellement différente de la première. Il libère alors la fréquence utilisée dans sa cellule d'origine. Pour pouvoir suivre les mobiles sans interrompre les communications, le réseau a donc besoin de savoir à chaque instant vers quelle cellule il peut aiguiller tel ou tel abonné, ce qui se traduit par une gestion plus complexe et une infrastructure sensiblement allourdie, incluant notamment un contrôleur de stations de base.

Le concept de réseau cellulaire, exposé plus en détail dans [Pra98, chapitre 3], introduit donc les innovations suivantes :

- le changement dynamique de la fréquence de communication entre la station de base et le terminal mobile *en cours* de communication en fonction des déplacements de celui-ci,
- la réutilisation des fréquences dans des cellules suffisamment éloignées les unes des autres et donc l'augmentation sensible de la capacité du réseau,

tout en ayant les inconvénients suivants :

- le nombre de stations de base à gérer et à relier entre elles augmente sensiblement, et par conséquent la complexité de la phase de planification du réseau aussi,
- le réseau doit offrir au terminal mobile la possibilité de savoir avec quelles stations de base il peut communiquer, ce qui implique la présence d'un dispositif de signalisation,
- le réseau doit gérer correctement le passsage d'un mobile d'une cellule à une autre, c'est-à-dire sans interruption de la communication.

I.1.1.2 Systèmes radiocellulaires « 1G »

En 1982, aux Etats-Unis, la FCC (Federal Communications Commission) normalise les spécifications du système AMPS, qui devient le standard unique du radiotéléphone en Amérique du Nord. Ainsi, dans les années 80, de nombreux réseaux cellulaires sont mis en service dans le monde entier, mais ils utilisent tous un mode de transmission de l'information analogique et sont pour la plupart incompatibles entre eux. En Europe, les systèmes suivants sont utilisés :

- NMT (Nordic Mobile Telephone), en Belgique, Finlande, France, Norvège, Suède, Danemark, et aux Pays-Bas,
- TACS (Total Access Communication System), au Royaume-Uni et en Italie,
- Radiocom 2000 en France, développpé par la société Matra et inauguré en 1986,
- Mobilphone 2 en Belgique,
- C-450 en Allemagne.

Ces systèmes représentent la « 1G », la première génération de systèmes de radiotéléphonie : ils n'offrent qu'une panoplie limitée de services pour les communications vocales, et rarement la possibilité de transférer des données sans en plus garantir leur protection contre les interceptions pirates ; ils utilisent tous une transmission globalement analogique (le signal modulant *et* son taitement sont analogiques).

Dans toutes les grandes métropoles, le radiotéléphone est victime de son succès : les opérateurs ne peuvent plus répondre aux demandes d'abonnements, supérieures aux capacités des réseaux. Pour surmonter cette situation, deux techniques de gestion ont été ajoutées : le passage d'un traitement analogique à un traitement numérique de l'information d'une part, et le multiplexage temporel d'autre part (plus de détails sur le multiplexage sont donnés à la section I.2.1).

I.1.1.3 Systèmes radiocellulaires « 2G »

Dans les années 90, les systèmes analogiques de première génération ont majoritairement été remplacés par des systèmes numériques, tels GSM en Europe, normalisé entre 1982 et 1988 par un groupe de travail appelé Groupe Spécial Mobile et déployé en 1991, ou encore IS 54 [TIA98] (proposé par la TIA[2] en 1990) et IS 95 [TIA99] (normalisé en 1993) en Amérique du Nord et au Japon. GSM n'est pas compatible avec les systèmes utilisés précedemment en Europe, ce qui n'a pas posé de problème particulier lors du déploiement, au vu de la très faible couverture 1G de l'époque et de son hétérogénéité (alors que les deux systèmes américains offrent une compatibilité descendante dans certaines conditions). Les avantages des systèmes numériques sur leurs prédécesseurs analogiques sont multiples :

- la résistance au bruit est accrue ; en effet, la moindre distortion subie par un signal modulé analogiquement introduit le plus souvent une distortion irréversible sur le signal original (le signal modulant), alors que dans le cas d'un signal modulé numériquement, des petites perturbations n'entraîneront aucune erreur sur le symbole décodé à la réception. De plus, des techniques d'entrelacement et de codage canal (introduction de redondance à l'émission dans le but de détecter et éventuellement de corriger à la réception les erreurs survenues pendant la transmission) peuvent être mises en œuvre pour faire baisser la

[2]Telecommunications Industry Association, `www.tiaonline.com`

probabilité d'erreur[3], au prix bien sur d'une baisse de débit utile,
- l'utilisation d'un système numérique permet d'employer des techniques d'accès multiple (plus de détails à la section I.2.1) plus efficaces en termes de partage de la ressource spectrale que dans le cas d'un système analogique,
- la communication peut être chiffrée, rendant ainsi plus difficiles les éventuelles tentatives d'interception,
- la phonie est de bien meilleure qualité,
- il devient possible de transmettre de manière robuste des données, et non plus seulement de la voix, à des vitesses comparables à celles des premiers modems pour ordinateurs personnels. GSM, par exemple offre un débit maximum de 9.6 kb/s (kilo-bits par seconde) et IS-95 offre 14.4 kb/s dans certaines conditions.

Ces systèmes représentent la « 2G » de la radiotéléphonie, et leur déploiement a surtout été rendu possible par les progrès de la technologie des semiconducteurs. En effet, les rapides avancées de la microéléctronique ont entraîné une réduction drastique des coûts de fabrication des circuits intégrés et des processeurs de signaux en même temps qu'une amélioration de leurs performances : fréquences de fonctionnement plus élévées, consommations de plus en plus faibles ... Les années 90 ont donc marqué l'avènement de la radiotéléphonie numérique et pour la première fois, il est possible de transférer des données *et* de la voix avec le même appareil (mais pas simultanément).

I.1.1.4 Systèmes radiocellulaires « 2,5G »

De la même manière que la 1G, la 2G arrive elle-aussi à saturation. Cette saturation est à la fois vraie pour la *couverture* mais également pour les *services* proposés aux abonnés qui aspirent à une augmentation des débits utiles afin d'avoir accès à des taux de transferts de données plus élevés.

Deux systèmes communément appelés de « 2,5G » sont le GPRS (*General Packet Radio Service*) et le EGPRS (ou EDGE, *Enhanced Data Rates for GSM Evolution*). Ils font apparaître la notion de communication en mode paquet car un utilisateur peut se voir attribuer plusieuers slots là où en 2G il n'en avait qu'un. L'innovation majeure de ces systèmes réside aussi dans le fait que différents schémas de codage sont proposés pour les communications, apportant ainsi une protection *variable* des données en fonction de la qualité du lien radio (ce qui permet dans de bonnes conditions de propagation d'augmenter le débit utile des données en diminuant la redondance apportée par le codage canal).

GPRS est une « surcouche » logicielle de GSM permettant à *un* utilisateur de se voir attribuer les ressources temporelles de *plusieurs* utilisateurs fictifs et de multiplier ainsi le débit de sa connexion (dans certaines limites). De plus, plusieurs schémas de codage des données sont envisagés, comme indiqué sur le tableau I.1.

Ainsi, avec un débit utile de 13.4 kbps par time slot (CS 2), un utilisateur utilisant 2 slots de la trame en réception bénéficiera d'un débit utile égal à 26.8 kbps, presque égal à *trois fois* le débit utile de GSM. Le débit maximal atteignable a été fixé à 115.2 kbps (4 time slots en réception avec utilisation du CS 2).

[3]La limite vient alors de la nécessité de transmettre de *l'information*, et non pas de ne transmettre que de la redondance destinée à *protéger* cette information.

Schéma de codage	Débit utile par slot (kbps)
CS 1	9.05
CS 2	13.4
CS 3	15.6
CS 4	21.4

TAB. I.1 – Débit utile par slot en GPRS en fonction du schéma de codage utilisé

EDGE s'appuie sur GPRS en offrant non seulement un nombre plus élevé de schémas de codage (9 pour EDGE contre 4 pour GPRS) mais aussi la possibilité de remplacer la modulation GMSK[4] par une modulation MDP8[5], portant ainsi le débit utile par slot à 48 kbps. Ce changement de modulation est mis en œuvre lorsque les conditions de propagation sont suffisamment bonnes. De la même manière qu'en GPRS, le débit offert à un utilisateur utilisant les 8 time slots de la trame atteindrait 384 kbps.

On passe ainsi d'un réseau « circuit » avec allocation statique des ressources à un réseau « paquet » avec allocation *dynamique* des ressources. Par rapport aux systèmes 2G, l'évolution fondamentale pour l'opérateur est la possibilité d'appliquer une gestion beaucoup plus fine des ressources temporelles et spectrales, et ainsi de satisfaire une gamme de besoins beaucoup plus large avec une souplesse accrue. On note aussi une évolution des méthodes de tarification, avec l'apparition d'une tarification au *volume transmis*, semblable à celle pratiquée par certains fournisseurs d'accès à Internet, accompagnée éventuellement d'une tarification évoluant avec la *qualité de service* offerte en plus des traditionnelles tarifications au *temps écoulé*.

I.1.2 Naissance de la « 3G »

Le problème incontournable des réseaux de radiotéléphonie aujourd'hui est celui de la saturation : dans certaines agglomérations françaises, il devient difficile, voire impossible de se connecter au réseau aux heures de pointe, et la situation tend à se généraliser à la plupart des grandes métropoles des pays développés. Une solution temporaire à ce problème a été l'ouverture des réseaux DCS 1800 en Europe et IS-136 [TIA00] aux Etats-Unis. Ces réseaux reprennent exactement le fonctionnement de GSM et IS-54 [TIA98], respectivement, à ceci près qu'ils occupent des bandes de fréquences plus élevées (autour de 1.8 GHz pour DCS 1800 contre 900 MHz pour GSM, par exemple). Ayant prévu et anticipé cet état de fait, l'UIT[6] a lancé dès 1995 un appel d'offres pour un système de troisième génération universel appelé IMT2000 (International Mobile Telecommunications 2000). En Europe, c'est l'ETSI[7] qui contribue dès 1996 à l'UMTS (Universal Mobile Telecommunications System) en réponse à l'appel d'offres de l'UIT, et notamment sur sa composante terrestre : l'UTRA (UMTS Terrestrial Radio Access).

En Europe, quatre interfaces radio candidates ont été étudiées, toutes basées sur des techniques d'accès multiple différentes présentées dans [OP98a]. L'une d'elles, basée sur l'OFDM (Orthogonal Frequency Division Multiplexing) a été rapidement abandonnée du fait de son importante complexité et de sa relative inadéquation à un environnement d'accès multiple non

[4]Gaussian Minimum Shift Keying
[5]Modulation De Phase à 8 états.
[6]Union Internationale des Télécommunications, `www.itu.int`
[7]European Telecommunications Standards Institute, `www.etsi.org`

synchronisé tel que celui de la radiotéléphonie mobile, mais réapparaît timidement aujourd'hui, sans doute grâce aux avancées théoriques et techniques dans le domaine. La solution purement TDMA (Time Division Multiple Access) a été elle-aussi abandonnée, malgré le fait qu'elle présentait de grandes similitudes avec GSM, ce qui aurait permis dans une certaine mesure aux opérateurs de minimiser le volume de modifications à apporter au cœur de réseau et au réseau radio lors de la migration vers la 3G.

Il ne reste donc que deux interfaces radio en développement pour l'UTRA, qui n'est que la composante *terrestre*[8] de l'UMTS, qui n'est lui-même qu'*une* des réponses *européennes* à l'appel d'offres de l'UIT, l'autre étant le DECT (Digital European Cordless Telephone). Force est donc de constater que le "U" de UMTS sera quelque peu usurpé. En effet, le développement de l'UMTS a été transféré de l'ETSI au 3GPP[9] en 1998, la priorité ayant été donnée à l'harmonisation de l'UMTS avec la proposition japonaise, les deux solutions ayant beaucoup de points communs. Mais on trouve aussi un organisme appelé 3GPP2[10], chargé de développer la solution américaine appelée cdma2000 [Den98]. Pressés de voir un standard aboutir, les organismes de normalisation chinois, japonais et coréens sont membres des deux organismes. En effet, l'Asie souffre d'un retard non négligeable (par rapport à l'Europe ou aux Etats-Unis) au niveau de sa couverture 2G, et a encouragé très tôt les développements 3G [Cha01].

Une des causes de cette multiplicité de standards candidats est la multiplicité des systèmes 2G les ayant précédés. En effet, les opérateurs, désireux de préserver une partie des investissements colossaux réalisés lors du déploiement des réseaux 2G, ont tendance à préférer des solutions dont les architectures sont proches de celles des prédécesseurs. Ainsi, UWC-136, TD-CDMA et cdma2000 sont en grande partie les héritiers directs de, respectivement, IS-54, GSM et IS-95. La conséquence de cette multiplicité est qu'il existera très probablement 3 ou 4 normes principales pas toujours compatibles entre elles, comme indiqué figure I.3. Plus de détails sur les divers organismes de normalisation intervenant dans ces groupes de travail sont données dans [OP98b, chapitre 14] et dans [Cha01].

I.1.3 Quelles sont les nouveautés apportées par l'UMTS ?

Les débits atteignables seront plus élevés que ceux disponibles actuellement : jusqu'à 384 kb/s en permanence, et jusqu'à 2 Mb/s (megabits par seconde, ou millions de bits par seconde) en situation de mobilité réduite. Atteindre ces débits dans n'importe quel environnement a nécessité le développement d'une interface radio complètement nouvelle, l'UTRA, détaillée section I.2.

L'utilisation de services véritablement multimédia, tels que la visiophonie, la vidéoconférence, le jeu en réseau, ou Internet se fera dans un confort accru (du moins supérieur à celui de la navigation sur un mobile WAP aujourd'hui, par exemple). On envisage mal la consultation confortable d'Internet sur un écran de téléphone semblable à ceux vendus aujourd'hui, et c'est pour cela qu'on parle désormais de *terminal*, et non plus de simple *téléphone*. De plus, cette nouvelle appellation correspond mieux à la fonction de l'objet auquel elle se réfère : en effet, la

[8] Une composante satellite est prévue pour l'UMTS, le S-UMTS, mais sa standardisation avance lentement, probablement suite aux récents déboires d'Iridium et Globalstar, deux compagnies proposant un service de radiotéléphonie par satellite et ayant toutes les deux connu des difficultés financières.

[9] Third Generation Partnership Project, `www.3gpp.org`

[10] Third Generation Partnership Project 2, `www.3gpp2.org`

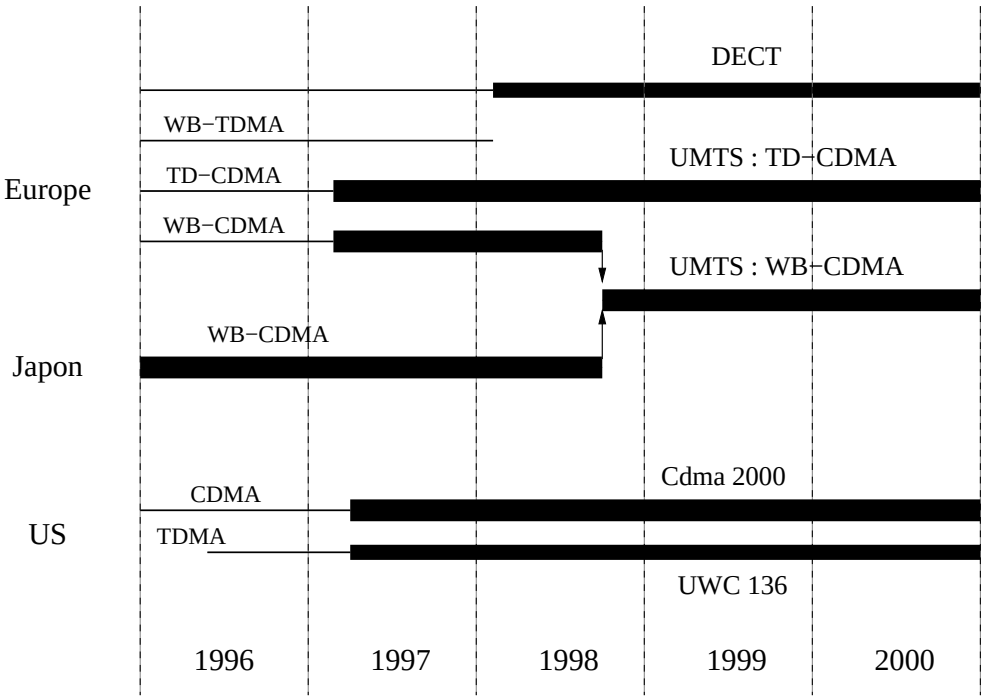


FIG. I.3 – Evolution des différents standards 3G

part des données dans le trafic radiomobile augmente sans cesse, traduisant une utilisation de plus en plus *informatique* du terminal, et donc de moins en moins *téléphonique*. Les prospectives indiquent qu'en 2015, les services 1G/2G (la voix et les données "bas débit") ne représenteront plus que 5% du trafic, alors que les contenus multimédia se partageront les 95% restant [Pej01], dont plus de 80% de contenu multimédia asymétrique (téléchargement de données, streaming vidéo, consultation peu ou pas interactive d'Internet).

Le réseau devra assurer une compatibilité descendante avec les réseaux de deuxième génération (GSM, en Europe). Pour l'opérateur, cela permet de déployer le réseau UMTS par îlots, et de ne pas assurer une couverture totale dès l'ouverture du service, ce qui serait bien trop coûteux et ne permettrait pas de valider une véritable montée en charge progressive des équipements. La compatibilité avec GSM offre donc à l'opérateur une solution de repli et aux abonnés une continuité de la communication dans le cas d'un déplacement vers une zone non-couverte par le réseau UMTS. Cette solution est illustrée figure I.4. Cette compatibilité a un prix : non seulement les cœurs de réseaux (le *core network*, en anglais) doivent être compatibles entre eux, connectés et les transferts de l'un à l'autre supervisés, mais les terminaux doivent aussi, pour pouvoir bénéficier de cette possiblité de repli, embarquer les fonctionnalités UMTS *et* GSM, ce qui fait augmenter dans une faible mesure[11] leur complexité déjà non négligeable.

Une autre nouveauté importante est l'apparition d'une garantie de *qualité de service* (QoS, Quality of Service en anglais). Cela signifie qu'un abonné voulant faire une visioconférence,

[11]La complexité du traitement numérique de l'UMTS étant supérieure d'au moins un ordre de grandeur à celle de GSM, c'est surtout au niveau des composants du front-end radio, responsables à la réception de la transformation du signal *analogique* capté par l'antenne en un signal *numérique*, et de la transformation inverse à l'émission, que cette contrainte de fonctionnement bimode va se ressentir.

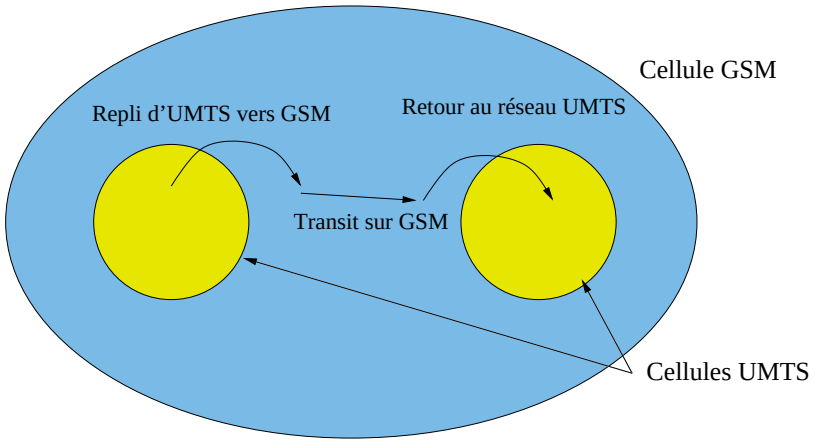


FIG. I.4 – Exemple de mise en œuvre de la compatibilité descendante UMTS/GSM

par exemple, pourra demander au réseau de se voir attribuer les ressources nécessaires pour une communication à haut débit pendant une durée précisée ou non durant la négociation de la communication, éventuellement moyennant des restrictions sur sa mobilité. En cas d'acceptation par le réseau de la requête de l'abonné, c'est le réseau qui a la responsabilité de lui fournir le débit demandé et de *maintenir* cette fourniture. On devine qu'un tel utilisateur "privilégié" dans une cellule nuira à la capacité de celle-ci, monopolisant une quantité importante de ses ressources, et donc à l'opérateur, qui doit donc se poser la question de savoir s'il vaut mieux accepter beaucoup d'utilisateurs à bas débit et refuser les utilisateurs demandant des débits élevés, au risque de ne pas satisfaire les abonnés ayant besoin de débits importants, ou s'il est préférable d'accorder aux utilisateurs qui le demandent des débits plus importants au risque de sous-employer sa cellule et éventuellement de ne pas satisfaire les abonnés voulant juste téléphoner. Le choix de la politique d'attribution des ressources de la cellule est aussi guidé par des contraintes économiques : la réponse à la question posée dépend surtout du prix de l'abonnement et du prix auquel sera proposé le service "débit à la demande".

Mais la plus grosse différence entre les réseaux 2G et les réseaux 3G est plus conceptuelle que technique. En effet, les réseaux 2G ont été conçus pour tirer le meilleur parti de ce que la technologie pouvait offrir à un instant t, et les services proposés sur ces réseaux ont été calqués et limités par les choix technologiques effectués, alors que pour la conception des réseaux 3G, le processus a été inversé : les organismes de normalisation ont décidé des services à fournir et ont effectué les choix technologiques qui s'imposaient quant à la fourniture de ces services. En cela, pour la 2G, *le réseau a tiré les services* alors que pour la 3G, l'inverse va se produire : *les services vont tirer le réseau*, et c'est sans doute là que réside la différence majeure entre ces deux générations de réseaux de radiotéléphonie.

I.2 La couche physique de l'UMTS : l'UTRA

Toutes les innovations présentées à la section précédente et résumées dans le tableau I.2, ont nécessité le développement d'une nouvelle interface radio et d'une nouvelle infrastructure de réseau. Cette dernière, l'UTRAN (UMTS Terrestrial Radio Access *Network*) apporte comme

Génération	**1G**	**2G**	**3G**
Exemples	NMT, AMPS	GSM, IS-95, IS-54, DCS 1800, IS-136	UMTS, cdma2000
Service ayant guidé la conception du réseau	Voix	Voix, puis Données	Données, puis Voix
Modulation	Analogique	Numérique	Numérique
Qualité de la phonie	Moyenne	Bonne	Qualité de la phonie sur téléphone fixe
Qualité des services multimédias	Aucun	Moyenne (limités par le réseau)	Bonne (QoS garantie)
Débits offerts	Très faibles (2.4 kbps maxi.)	Moyens (9.6 kbps pour GSM, 14.4 kbps pour IS-95, 144 kbps pour GPRS et jusqu'à 384 kbps pour EDGE)	Elevés (384 kbps en permanence et jusqu'à 2 Mbps en mobilité réduite)
Compatibilité avec la ou les générations antérieures	Pas de génération précédente	Très rare	Descendante (solution de repli en l'absence de couverture 3G)

TAB. I.2 – Comparaison des systèmes 1G, 2G et 3G

principales nouveautés par rapport à l'infrastructure du réseau GSM la gestion de la qualité de service et l'introduction du protocole IP au sein du cœur de réseau[12]. Les spécifications techniques sont disponibles sur le site Internet du groupe de travail 3GPP, `www.3gpp.org`. C'est à l'UTRA (l'interface radio de l'UMTS) que nous allons désormais nous intéresser.

I.2.1 Duplex et accès multiple : les deux modes de l'UMTS

Lors de la définition de l'interface radio d'un réseau de communication mobile multi-utilisateurs, les deux premiers points à préciser sont :

- la façon dont sont séparées la voix montante et la voix descendante d'une même communication entre le terminal et la station de base : c'est le *duplex*,
- la façon dont est partagée la ressource fréquentielle entre les divers utilisateurs du système : c'est l'*accès multiple*.

I.2.1.1 Le duplex

Le duplex correspond à la séparation des voies montante et descendante d'une même communication. Pour l'UMTS, deux possibilités ont été identifiées : le duplex *fréquentiel* (FDD, Frequency Division Duplex) et le duplex *temporel* (TDD, Time Division Duplex).

Le duplex fréquentiel correspond à la situation où le terminal et la station de base émettent à des fréquences différentes, comme indiqué sur la figure I.5. Ce mode de duplex offre la possibilité

[12]Le *core network* spécifié dans la version 1999 est mixte (réseau commuté/IP), alors que celui spécifié dans la version 2000 est *tout IP*.

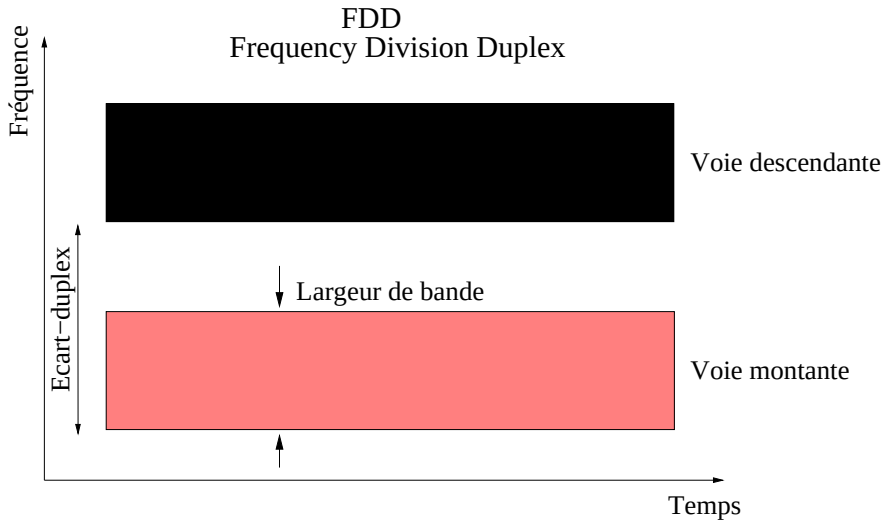


FIG. I.5 – Duplex fréquentiel

d'émettre et de recevoir des données en même temps, mais impose de disposer du matériel suffisant pour émettre et recevoir à deux fréquences différentes en même temps. D'autre part, la mise en place d'un tel mode de duplex nécessite l'attribution de *deux* fréquences, et plus généralement la division de la bande de fréquence attribuée à l'opérateur en deux sous-bandes appairées, une réservée à la voie montante, l'autre à la voie descendante. La différence entre les fréquences d'émission et de réception est appelée *écart-duplex*, et peut être fixe, comme pour GSM, ou variable, comme pour l'UMTS[13].

Pour le duplex temporel, les deux voies empruntent la même fréquence porteuse : des deux côtés de la liaison, l'émission et la réception se font à la même fréquence, mais jamais en même temps, comme indiqué sur la figure I.6. Un système de temps de parole et d'écoute est mis en place pour éviter la collision des émissions : le mobile émet pendant un temps Δt, puis c'est au tour de la station de base d'émettre pendant le même temps Δt, et ainsi de suite ... Ce mode de duplex a l'avantage évident par rapport au FDD de ne nécessiter l'allocation que d'une seule fréquence au lieu des 2 requises par le FDD, mais le débit brut du canal radio est alors divisé par 2, seule la moitié du temps étant passée à émettre des données. De plus, le fait que les voies montantes et descendantes soient véhiculées par des porteuses de même fréquence offre la possibilité aux deux extrémités de la liaison de partager une estimation du canal commune : la station de base peut ainsi communiquer au terminal les résultats des estimations de canal qu'elle réalise, lui évitant de reproduire les mêmes efforts, et diminuant ainsi la redondance nécessaire sur la voie descendante. En contrepartie, il est nécessaire d'établir et de maintenir une synchronisation aussi précise que possible entre le mobile et la station de base, de manière à éviter l'écrasement des données de l'un par l'autre en cas de superposition des émissions.

I.2.1.2 L'accès multiple

L'accès multiple est la façon dont une même ressource fréquentielle est rendue accessible à plusieurs utilisateurs.

[13]L'écart-duplex « par défaut » en UMTS-FDD est fixé à 190 MHz.

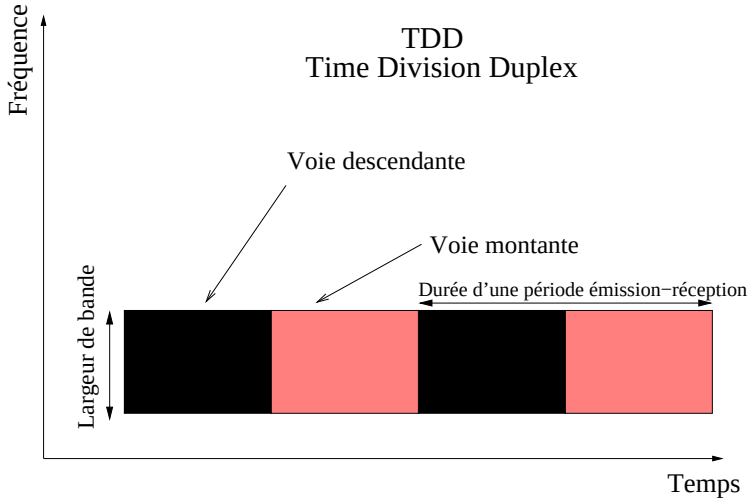


FIG. I.6 – Duplex temporel

Accès multiple à répartition en fréquence : le FDMA

Les premiers systèmes de radiotéléphonie, comme Radiocomm 2000 par exemple, ont introduit l'*accès multiple à répartition en fréquence* (AMRF, ou FDMA[14] en anglais, pour Frequency Division Multiple Access), qui consiste à découper la bande de fréquence allouée à l'opérateur en canaux d'une certaine largeur fréquentielle et à attribuer *un* canal fréquentiel à chaque utilisateur dans le cas d'un duplex temporel, et *deux* canaux fréquentiels à chaque utilisateur dans le cas d'un duplex fréquentiel (un canal pour la voie montante, et un autre pour la voie descendante). La figure I.7 représente les deux cas d'accès multiple à répartition en fréquence. Pour un duplex en fréquence à écart-duplex fixe, les deux bandes sont divisées en canaux alloués aux utilisateurs, et séparées par l'écart-duplex. Pour un duplex en temps, lorsqu'un canal est alloué à un utilisateur, la base et l'utilisateur émettent sur ce canal tour à tour.

Accès multiple à répartition en temps : le TDMA

Les systèmes plus récents (essentiellement GSM et IS-54/IS-136) ont introduit un autre mode d'accès multiple utilisé *conjointement* au FDMA : l'*accès multiple à répartition en temps* (AMRT, ou TDMA en anglais, pour Time Division Multiple Access), qui consiste à découper l'axe temporel en intervalles de temps élémentaires (time slots, en anglais[15]) de même durée et parfaitement consécutifs (il n'y a pas d'intervalle de temps entre 2 slots), regroupés en *trames* (frames, en anglais). Une trame étant composée d'un nombre fixé de slots, on attribue un slot à un utilisateur, et il émettra (respectivement il écoutera) uniquement à chaque occurrence de son slot, à chaque trame, et ce jusqu'à la fin de la communication.

Les avantages du TDMA sont multiples :

– la capacité du réseau est augmentée, au prix d'une baisse des débits par utilisateur : pour une trame de 10 slots, par exemple, un canal fréquentiel n'accueillera plus un seul

[14]Les sigles français n'ayant pas encore été largement adoptés dans la littérature, ce sont les sigles anglais qui seront employés ici.

[15]La traduction française du mot *time slot* étant le peu pratique *intervalle de temps*, les mots *slot* ou *time slot* seront utilisés par la suite.

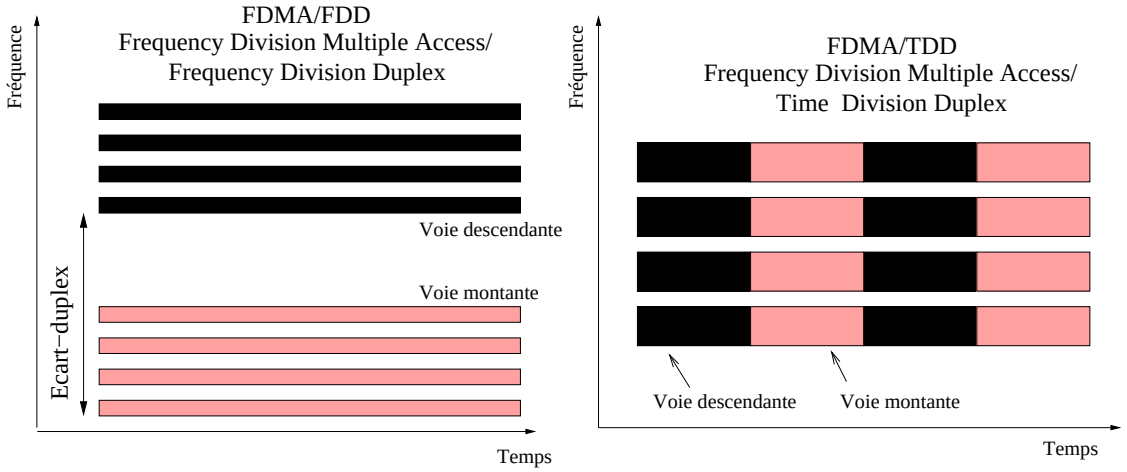


FIG. I.7 – Accès multiple à répartition en fréquence

utilisateur mais une dizaine d'entre eux, avec bien sûr un débit par utilisateur dix fois moindre,

- la durée de vie des batteries des terminaux augmente, car le mobile n'émet (et respectivement n'écoute) désormais qu'un dixième du temps,
- les stations de base nécessitent moins de matériel : alors qu'une station de base purement FDMA a besoin d'un bloc émetteur-récepteur par fréquence (et donc par utilisateur), une station de base TDMA peut servir *plusieurs* uilisateurs avec *un seul* bloc émetteur-récepteur[16].

Ceci dit, la mise en œuvre du TDMA implique une synchronisation parfaite entre les mobiles et la station de base, le plus souvent acquise à la mise sous tension du mobile et maintenue pendant la durée de la communication, ainsi qu'une augmentation de la largeur des canaux fréquentiels dans le cas où on veut garantir le même débit qu'en FDMA.

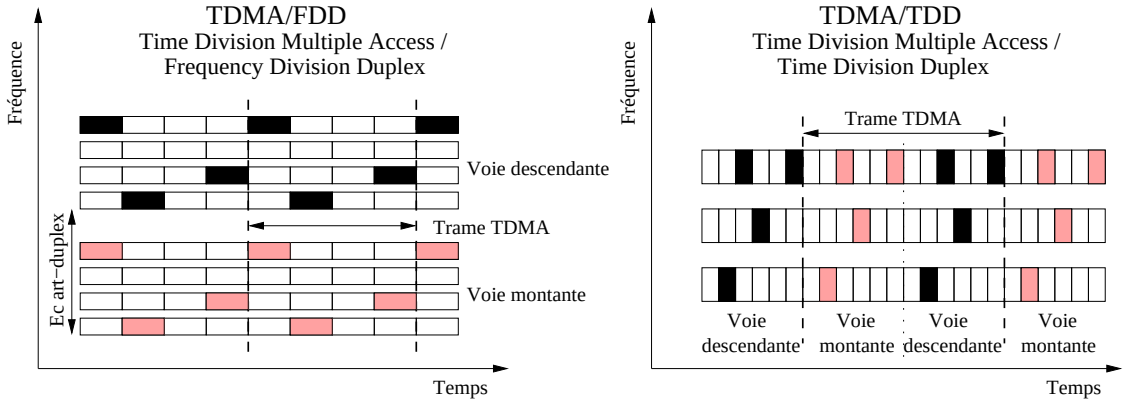


FIG. I.8 – Accès multiple à répartition en temps

[16]Un tel bloc émetteur-récepteur est appelée *transceiver*, de l'anglais transmitter-receiver, que l'on pourrait traduire par transcepteur. Cette traduction n'étant pas ou peu usitée, le mot transceiver sera utilisé par la suite.

La figure I.8 illustre les deux cas possibles d'accès multiple à répartition en temps. Pour un duplex fréquentiel (c'est le cas de GSM[17]), un canal de transmission entre le terminal et la station de base est constitué d'une paire de fréquences séparées par l'écart-duplex et d'un numéro de slot. Pour un duplex temporel, la communication est portée par une seule fréquence, mais une trame sur deux est réservée à la voie montante tandis que l'autre est réservée à la voie descendante. Les slots sont récurrents, en ce sens qu'un utilisateur garde le même time slot d'une trame à l'autre. La fréquence, quant à elle, peut changer en cours de communication : on a alors ce qu'on appelle un *saut de fréquence* (frequency hopping, en anglais), qui est une technique d'*étalement de spectre* parmi d'autres (l'étalement de spectre est introduit section I.2.2).

Accès multiple à répartition par code : le CDMA

IS-95 [TIA99] a été le premier système de radiotéléphonie à utiliser l'*accès multiple à répartition par code* (AMRC, ou CDMA en anglais, pour Code Division Multiple Access). Le CDMA se distingue des techniques d'accès multiple présentées ci-dessus par le fait que les signaux émis par les différents utilisateurs présents dans la cellule sont émis *en même temps* et *à la même fréquence*. La séparation des signaux se fait selon une «troisième dimension[18]», le code, comme illustré sur la figure I.9.

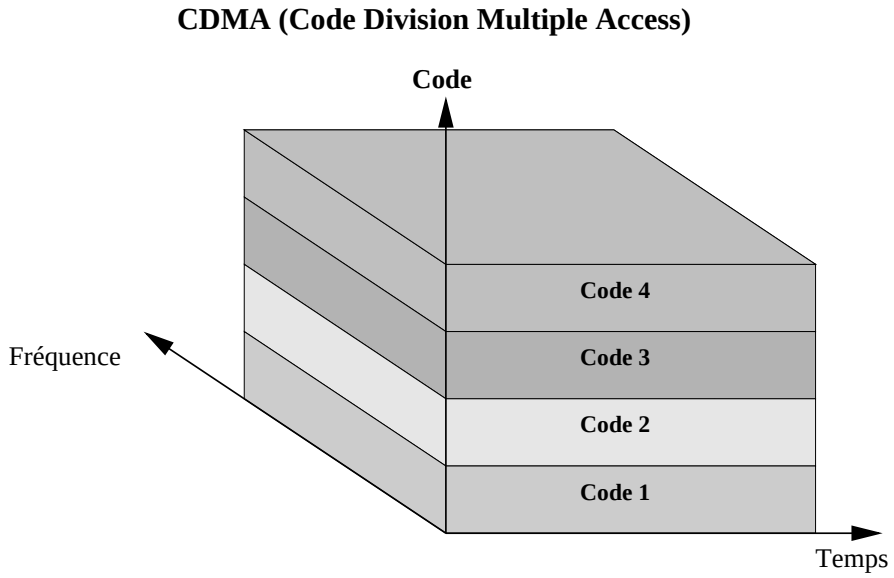


FIG. I.9 – Accès multiple à répartition par code

Les terminaux présents dans la cellule émettent leurs signaux en les « tatouant » d'une certaine manière qui les rend (dans des conditions idéales) parfaitement séparables par la station

[17]Sur la figure I.8, les frontières temporelles des trames montantes et descendantes sont synchronisées : ce n'est pas forcément le cas : GSM, par exemple, introduit un retard de 3 time slots entre la voie montante et la voie descendante.

[18]Parler ici de *dimensions* au sens classique du terme, que ce soit pour le code, le temps ou la fréquence, est un abus de langage traditionnellement commis à des fins purement didactiques, mais n'en reste pas moins un abus de langage. En effet, le temps et la fréquence sont liés par la transformée de Fourier, et le code s'exprime lui-aussi en termes de temps et de fréquence, ne constituant donc en aucun cas une dimension à part entière.

de base, malgré le fait qu'ils se soient superposés pendant leur passage dans le canal radiomobile. Ce résultat est atteint grâce aux codes employés pour « tatouer » les données émises : dans le meilleur des cas, les codes employés sont orthogonaux au sens du produit scalaire dans une base des codes émis. Cela signifie que si la station de base décode la somme des signaux de tous les utilisateurs avec un code particulier, elle ne récupérera que le signal issu de l'émetteur ayant utilisé ce code en annulant les signaux tatoués par d'autres codes. La façon dont est réalisé ce codage et les codes employés sont décrits section I.2.2.

I.2.1.3 Les deux modes de l'UMTS

Pour l'UMTS, deux combinaisons de duplex et d'accès multiple ont été retenues, donnant ainsi naissance aux deux modes de l'UMTS. Ces deux modes sont l'*UMTS-FDD*, aussi appelé *Wideband CDMA* (CDMA large bande, ou WB-CDMA par la suite), et l'*UMTS-TDD*, aussi appelé TD-CDMA. Le mode FDD utilise un duplex fréquentiel et un accès multiple à répartition par code (CDMA) par-dessus une division de la ressource spectrale en porteuses disjointes (FDMA). Le mode TDD, lui, utilise un duplex temporel et un accès multiple hybride, tenant à la fois du TDMA et du CDMA : l'axe temporel est divisé en trames elles-mêmes divisées en slots, mais plusieurs utilisateurs peuvent se voir allouer le même time slot. Leurs communications sont « empilées » selon le principe du CDMA, et utilisent donc les mêmes ressources fréquentielles et temporelles. Le plan des fréquences allouées à ces deux modes de l'UMTS est donné figure I.10.

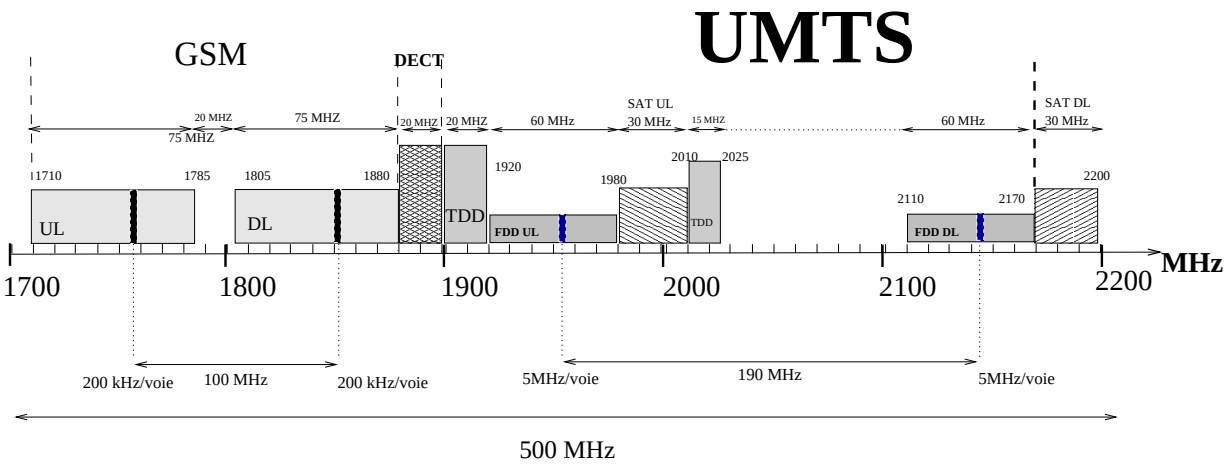


FIG. I.10 – Allocation des fréquences aux systèmes de radiotéléphonie mobile autour de 2 GHz

La normalisation s'étant d'abord orientée vers la solution WB-CDMA, celle-ci est aujourd'hui plus avancée que la solution TD-CDMA et sera la première à être déployée, probablement en 2002. Le déploiement de l'UMTS-TDD suivra avec 1 à 2 ans de retard. D'autre part, on peut penser que les premiers services déployés n'exploiteront pas toutes les possibilités de l'UMTS et que l'on ne pourra pas établir une liaison à 2 Mb/s le lendemain de l'ouverture commerciale du réseau. On s'achemine donc à long terme vers une large couverture FDD proposant des services à moyen débit (jusqu'à 384 kb/s) et une couverture TDD plus ponctuelle (centres urbains, intérieurs de bâtiments ...), qui proposera un peu plus tard les fameux services à 2 Mb/s, comme

indiqué figure I.11. Ce point de vue est conforté par les codes utilisés et les spécificités des deux interfaces radio, qui font que l'UMTS-TDD est plus adapté aux hauts débits.

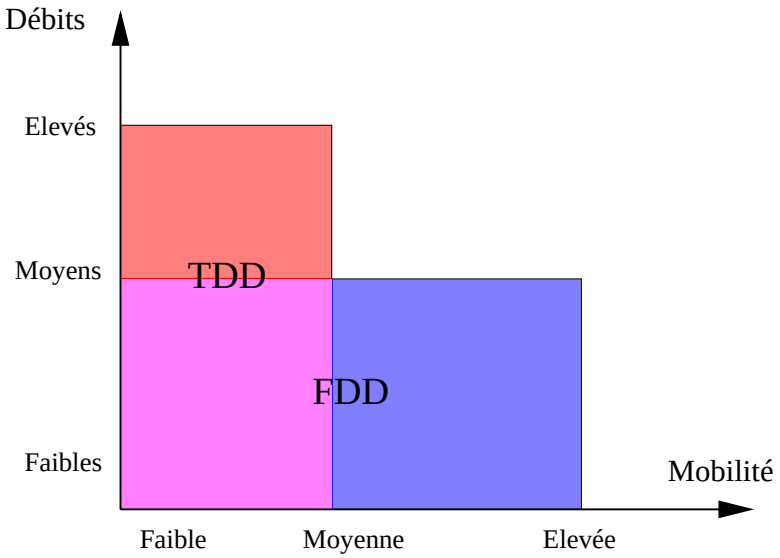


FIG. I.11 – Utilisations privilégiées des deux modes de l'UMTS

I.2.2 CDMA et étalement de spectre

La clef de voûte de l'UTRA semble donc résider dans le CDMA, où *étalement de spectre à séquence directe* (DS-SS en anglais, pour Direct Sequence Spread Spectrum), cette technique d'accès multiple qui permet à tous les utilisateurs de la cellule d'émettre en même temps et à la même fréquence. Cet « exploit » est l'aboutissement de nombreux travaux de recherche, principalement à des fins militaires, qui n'ont vu leur première application civile au domaine des télécommunications[19] qu'avec le déploiement d'IS-95, aux Etats-Unis, au début des années 90. On trouve dans [Sch82] un panorama assez complet des avancées théoriques et techniques ayant eu lieu jusqu'au milieu des années 40 dans les domaines suivants :

- les systèmes RADAR[20],
- la théorie des communications et la théorie de l'information,
- l'implantation matérielle des corrélateurs,
- la protection des communications (cryptographie et stéganographie[21]),
- le guidage de missile,

ainsi qu'un recensement détaillé des premiers systèmes à étalement de spectre.

[19] Les premières utilisations du CDMA ont concerné les transmissions en environnement industriel fortement bruité, dans les centrales nucléaires par exemple.

[20] *RAdio Detection And Ranging*

[21] Alors que la cryptogtaphie vise à protéger le contenu du message en cas d'interception, la stéganoraphie vise à dissimuler la communication elle-même, ou du moins le fait qu'il y ait communication, diminuant ainsi la probabilité d'une tentative d'interception.

I.2.2.1 Principe du CDMA

Emettre des données en CDMA consiste à remplacer l'émission de symboles à un débit donné par l'émission à un rythme supérieur de symboles élémentaires appelés *chips* obtenus en multipliant les symboles de départ par les éléments d'un code pseudo-aléatoire *périodique*, dont la période est égale à la durée symbole, choisi au sein d'une famille de codes *orthogonaux* et propre à chaque utilisateur.

Soit $\{b_k\}_{k\in[0,N-1]}$ les N symboles de départ, et $D = 1/T$ leur débit (T étant la période symbole). Après passage des *symboles* aux *chips*, la séquence émise sera la séquence $\{c_n\}_{n\in[0,N_c-1]}$ de N_c *chips*, émise au débit $D_c = D \cdot SF$. Les chips ont donc une durée $T_c = T/SF$, et sont obtenus en multipliant les symboles originaux $\{b_k\}$ par les éléments $\{d_n\}_{n\in[0,SF-1]}$ composant une séquence pseudo-aléatoire periodique de période T, dont le débit est SF fois supérieur à celui de la séquence $\{b_k\}$. Le schéma-bloc d'un tel émetteur est représenté figure I.12.

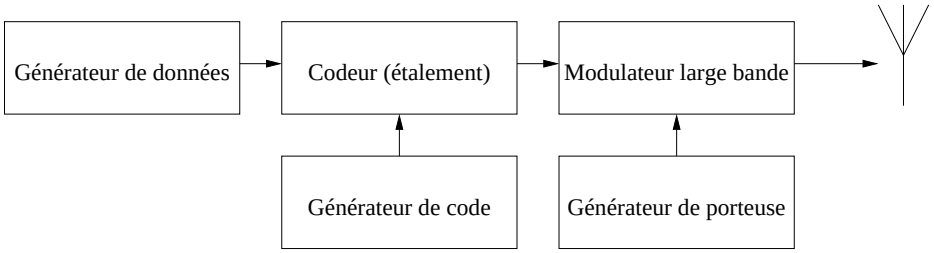


FIG. I.12 – Schéma-bloc d'un émetteur CDMA

L'opération effectuée est la suivante :

$$\forall k \in [0, N-1], \forall i \in [0, SF-1], c_{k\times SF+i} = b_k \cdot d_i \tag{I.1}$$

Cette opération est appelée *étalement*, et avec des formes d'onde en lieu et place des séquences binaires, elle s'écrit :

$$u_T(t) = \begin{cases} 1 \text{ pour } 0 \leq t < T \\ 0 \text{ sinon} \end{cases} \tag{I.2a}$$

$$b(t) = \sum_{k=-\infty}^{+\infty} b_k u_T(t - kT) \tag{I.2b}$$

$$d(t) = \sum_{k=-\infty}^{+\infty} \sum_{n=0}^{SF-1} d_n u_{T_c}(t - nT_c - kT) \tag{I.2c}$$

$$c(t) = b(t) \cdot d(t) \tag{I.2d}$$

La même opération peut être perçue comme un filtrage linéaire. En effet, il suffit d'une part de remplacer la suite de *créneaux* $b(t)$ par une suite de *Diracs* $b'(t)$ et d'autre part de considérer un filtre G dont la réponse impulsionnelle $g(t)$ est égale à *une* période du code $d(t)$ pour retrouver un processus de filtrage linéaire, comme indiqué figure I.13.

L'équation d'un tel filtrage est la suivante :

$$b'(t) = \sum_{k=-\infty}^{+\infty} b_k \delta(t - kT) \tag{I.3a}$$

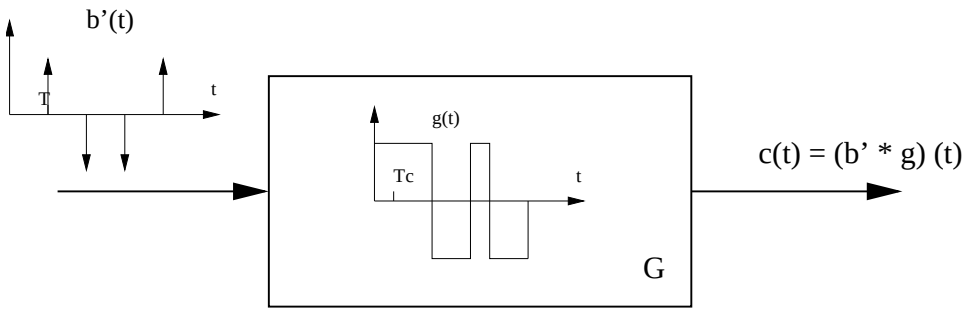


FIG. I.13 – Le CDMA vu comme un filtrage linéaire

$$g(t) = \sum_{n=0}^{SF-1} d_n u_{T_c}(t - nT_c) \quad \text{(I.3b)}$$

$$c(t) = (b' * g)(t), * \text{ dénotant ici la convolution.} \quad \text{(I.3c)}$$

SF est appelé le *facteur d'étalement*, car à *un* symbole correspondent SF chips. La séquence $\{d_n\}_{n\in[0,SF-1]}$ est appelée *séquence d'étalement*, ou encore *séquence étalante*. Un exemple de passage des symboles aux chips avec un facteur d'étalement égal à 8 est donnée figure I.14.

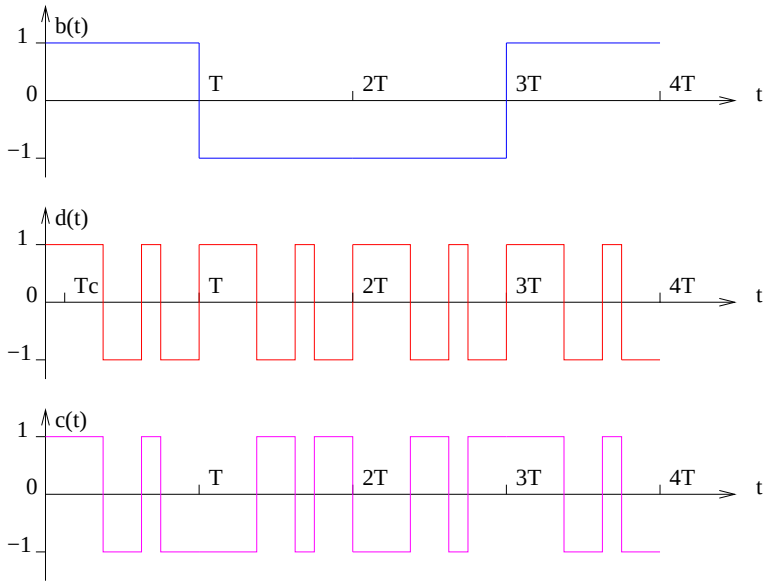


FIG. I.14 – Illustration du principe de l'étalement (SF=8)

Le choix des séquences d'étalement n'est pas anodin. En effet, pour conférer au signal étalé les caractéristiques spectrales d'un bruit blanc, il est d'usage de choisir des séquences pseudo-aléatoires[22] particulières. Les séquences *de longueur maximale*, ou séquences *MLLSR* pour *Maximal Length Linear Shift Register Sequences*, présentées dans [Gol92], sont souvent utilisées. Leur construction est abordée dans [Vit95, chapitre 2]. D'autres familles de séquences

[22]Les séquences purement aléatoires ne conviennent pas car elles ne peuvent pas être régénérées de manière synchrone à la réception.

binaires utilisables dans le cadre du CDMA sont présentées dans [DJ98]. On y trouve les séquences de Gold [Gol67], [Gol68], de Kasami [Kas66] ou encore de Walsh-Hadamard [ASO97]. Ces familles de séquence ont en commun des fonctions d'autocorrélation très étroites, présentant un pic élevé en 0, et des fonctions d'intercorrélation très faibles relativement à la hauteur du pic d'autocorrélation, comme indiqué figure I.15.

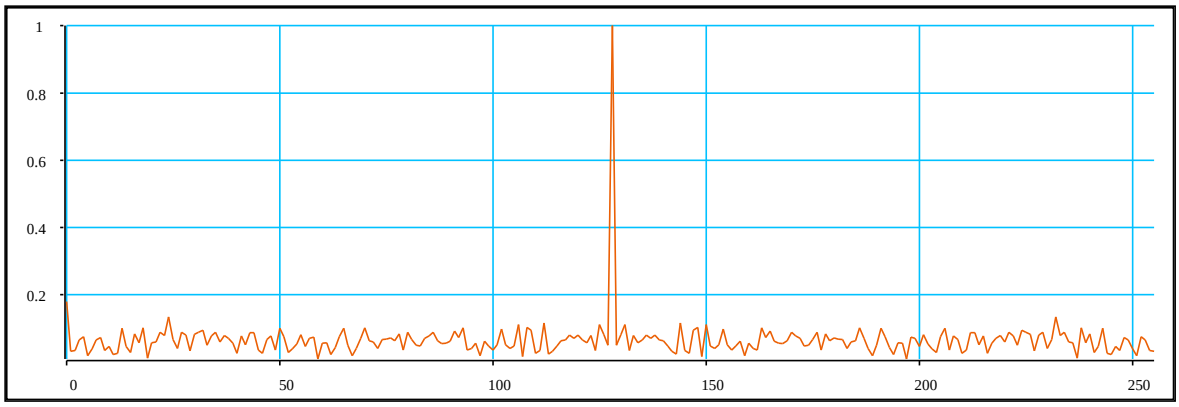


FIG. I.15 – Autocorrélation normalisée d'un fragment de séquence de Gold complexe de 256 chips de long

Les séquences MLLSR sont obtenues à partir d'un registre à décalage dont une combinaison des sorties est rebouclée sur l'entrée, comme illustré figure I.16. Lorsque c_i est égal à 1, la branche correspondante est rebouclée sur l'entrée. L'addition représentée sur le schéma est une addition modulo 2, ou encore une porte XOR (Ou Exclusif), dont la sortie ne vaut 1 que dans le cas où les deux entrées sont différentes (0 et 1 ou 1 et 0).

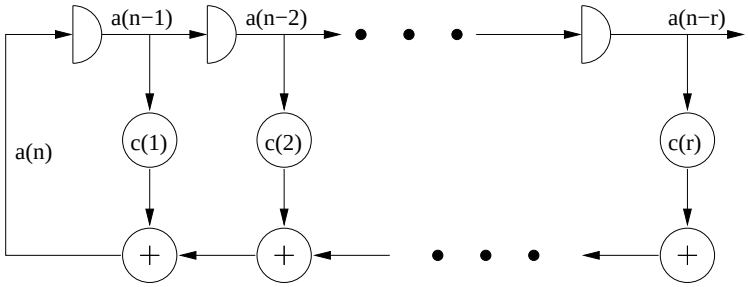


FIG. I.16 – Registre à décalage servant à générer des séquences d'étalement

La séquence générée par un tel registre peut s'écrire :

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_r a_{n-r} = \sum_{i=1}^{r} c_i a_{n-i}$$

Elle ne dépend que de deux choses : le vecteur des rebouclages $\{c_i\}_{i\in[1,r]}$ et le vecteur des conditions initiales $\{a_{-r}, a_{-r+1}, \cdots, a_{-1}\}$, juste avant que le terme a_0 ne soit généré. On émet l'hypothèse qu'au moins c_r est égal à 1, sinon le registre n'est plus de longueur r. Il existe d'autres manières de trouver ou de construire des ensembles de séquences adaptées au CDMA, comme le recuit simulé ([Den96]) ou la recherche taboue ([BO98]), par exemple.

I.2.2.2 Technique de réception associée

Pour estimer les symboles émis à partir du signal reçu ramené en bande de base et filtré, le récepteur va *désétaler* les chips en appliquant le même type de transformation qu'à l'émission : il va multiplier les chips $\{\hat{c}_n\}_{n\in[0,N_c-1]}$ reçus et les chips du code $\{d_n\}_{n\in[0,N_c-1]}$ (le code doit être connu du récepteur, soit dans une mémoire, soit par régénération), intégrer le signal obtenu sur la durée d'un symbole, et effectuer une décision sur la valeur du symbole émis. Dans le cas de séquences binaires, l'opération effectuée est la suivante :

$$\forall k \in [0, N-1], \hat{b}_k = sgn\left(\frac{1}{SF}\sum_{i=0}^{SF-1} \hat{c}_{k\times SF+i} \cdot d_{k\times SF+i}\right) \tag{I.4}$$

Les deux données absolument nécessaires au succès du désétalement sont la connaissance du code qui a servi à l'étalement et la connaissance de l'instant du début de l'émission. En effet, un seul chip de décalage entre le signal reçu et le code régénéré à la réception suffit pour rendre les symboles émis irrécupérables. C'est pour cela qu'on trouve dans les récepteurs CDMA un organe responsable de la poursuite (*tracking* en anglais) du code, c'est-à-dire du maintien de la synchronisation, comme indiqué figure I.17. Le fonctionnement de tels organes est expliqué dans [Vit95, chapitre 3].

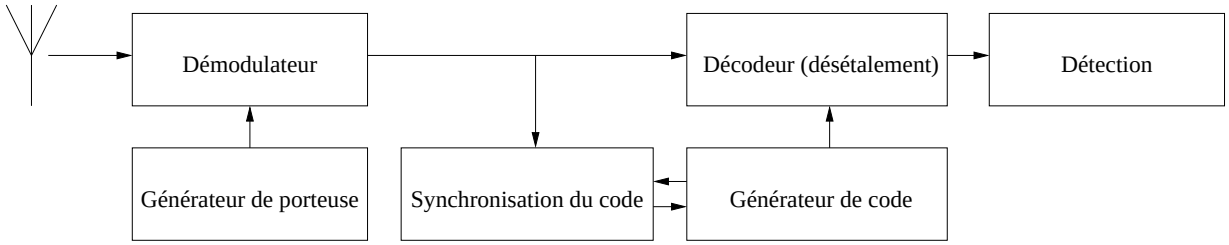


FIG. I.17 – Schéma-bloc d'un récepteur CDMA

I.2.2.3 Lien entre CDMA et étalement de spectre

Une conséquence de l'augmentation du rythme est l'*étalement* déjà mentionné du spectre du signal ainsi obtenu, expliqué dans [PSM82]. Les créneaux se succédant à un rythme SF fois plus élevé, le signal occupe en effet une bande SF fois plus importante, et c'est sans doute ce qui explique le succès du CDMA auprès des militaires : la transmission est en effet *noyée* dans le bruit d'une part (si la séquence étalante est bien choisie, le signal étalé ressemble à un bruit blanc), et *protégée* contre les éventuelles interceptions d'autre part (sans la connaissance de la séquence étalante *et* de l'instant du début de la transmission, il est impossible d'accéder au contenu du message transmis).

Il est important pour la suite de noter que l'opération d'étalement est presque involutive : mis à part l'intégrateur en bout de chaîne qui permet de passer du rythme chip au rythme symbole lors du désétalement, l'opération effectuée à l'émission est la même qu'à la réception, à savoir une multiplication chip à chip d'un signal (les données à émettre, suréchantillonnées au rythme chip à l'émission ou le signal reçu et filtré à la réception) par un code généré localement. Cette opération, appliquée à un signal à bande étroite tel que le signal à transmettre, étale son

spectre et le transforme ainsi en signal à bande large. Appliquée à un signal à bande large, elle ne modifie pas la largeur de son spectre, sauf si le signal en question est le signal émis à l'aide du même code. Dans ce cas, elle fait ressortir le signal à bande étroite du bruit dans lequel il a été transmis.

Pour le lecteur intéressé, on peut citer d'autres formes d'étalement de spectre, dont le *saut de fréquence* (FH-SS en anglais, pour *Frequency Hopping Spread Spectrum*), employé en GSM. Une émission à saut de fréquence a ceci de particulier qu'elle change régulièrement de porteuse au cours du temps. En GSM, par exemple, les données émises pendant un slot ne sont pas émises sur la même porteuse d'un slot à l'autre, de manière à limiter les risques de tomber dans un trou de *fading* (le *fading* est introduit section II.1.1.2) d'une part, et à protéger la communication des interceptions éventuelles (pour réussir une interception de longueur significative, il faut arriver à *suivre* l'émission au fur et à mesure des changements de fréquence porteuse, ce qui peut s'avérer relativement complexe). Ainsi, plus les intervalles de temps entre deux changements de porteuse sont de courte durée, plus la transmission donne l'illusion de s'effectuer sur une bande plus large que celle qu'elle occupe en réalité à un instant donné.

Bien que l'étalement à séquence directe et le saut de fréquence soient les deux techniques d'étalement de spectre les plus couramment utilisées, il existe d'autres techniques plus ésotériques, telles que le saut en temps (*time hopping spread spectrum*), dans lequel le signal est transmis de manière discontinue, en rafale, à des instants déterminés par le code, ou encore le *chirp spread spectrum*, uniquement utilisé pour l'instant dans les radars, sans compter les possibilités de systèmes hybrides. Plus de détails sur la technique de l'étalement de spectre et les méthodes possibles sont donnés dans [Pra98, chapitre 8].

I.2.2.4 Avantages et inconvénients du DS-CDMA

Du fait de l'étalement et du spectre élargi qui en résulte, les signaux DS-CDMA possèdent un certain nombre de propriétés qui les différencient des signaux à bande étroite :

- Possibilité d'accès multiple : si plusieurs utilisateurs transmettent simultanément des signaux à spectre étalé, le récepteur peut quand même distinguer les utilisateurs à condition que les codes employés aient des intercorrélations suffisamment faibles. Le désétalement du signal global avec un code particulier ne fera en effet ressortir que le signal étalé avec ce code-ci. Si il n'y a pas trop d'utilisateurs simultanément, la puissance du signal d'intérêt dans la bande utile est alors bien supérieure à la puissance des interférences dans cette même bande.
- Réjection des interférences dues aux trajets multiples : les trajets multiples, introduits à la section II.1, sont des copies du signal émis qui se superposent au signal original au cours du passage dans le canal radiomobile et qui sont présents en tant qu'interférences à l'entrée du récepteur. Or, s'ils sont distants d'au moins la largeur de l'autocorrélation du code employé, ils seront considérés par le récepteur comme des signaux d'autres utilisateurs et ne verront qu'une petite partie de leur puissance conservée dans la bande utile, relativement à la puissance du signal d'intérêt.
- Réjection des interférences bande étroite : que les interférences bande étroite soient d'origine physique (évanouissement ...) ou humaine (brouilleurs), les signaux causant ces interférences se voient appliquer à la réception le même traitement que celui qu'a subi le signal d'intérêt à l'émission : la corrélation avec le code d'intérêt va étaler le spectre des

signaux d'interférences bande étroite et ainsi diminuer leur puissance dans la bande utile d'un facteur SF, comme illustré figure I.18.

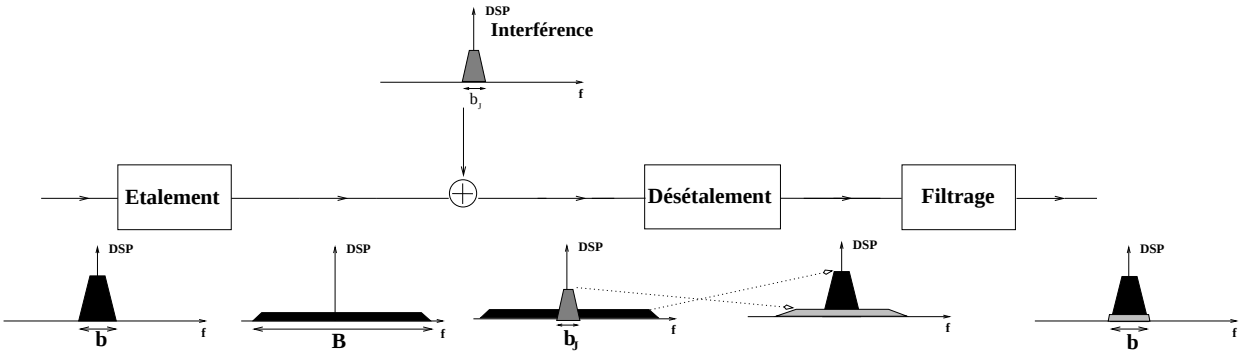


FIG. I.18 – Illustration de la réjection des interférences à bande étroite

– Faible probabilité d'interception : du fait de sa faible densité spectrale de puissance, le signal à spectre étalé est intrinsèquement difficile à détecter, ce qui explique avec le point précédent le succès du CDMA auprès des militaires.

De plus, les signaux DS-CDMA ont les avantages suivants :

– La génération des codes employés est en général très simple, de même que le procédé de codage : uns simple multiplication suffit, qui peut même être réduite à une équation binaire à quelques bits.
– Une seule fréquence porteuse est employée, ce qui relâche grandement les contraintes sur les composants analogiques en termes de rapidité de synchronisation et sur les composants numériques en termes de contrôle des composants analogiques.
– Aucune synchronisation entre les utilisateurs n'est nécessaire.

En contrepartie, l'utilisation d'un système DS-CDMA impose la mise au point de dispositifs de synchronisation très précis. En effet, la synchronisation entre le code régénéré à la réception et les chips reçus doit se faire avec une erreur inférieure à une fraction de chip, sous peine de ne plus pouvoir désétaler correctement le signal du fait de l'étroitesse de l'intercorrélation du code, voir figure I.15.

D'autre part, la puissance reçue par la station de base de la part des utilisateurs proches est bien supérieure à la puissance reçue de la part des utilisateurs qui en sont plus éloignés. Comme la transmission à spectre étalé utilise la totalité de la bande disponible, un utilisateur très proche de la station de base peut créer de très fortes interférences pour les utilisateurs éloignés. Ce phénomène est connu sous le nom de *near-far effect*, que l'on peut traduire par problème proche-lointain. En pratique, ce problème est traité par une boucle de *contrôle de puissance* dont l'objectif est que les signaux émis de chaque utilisateur arrivent avec une puissance égale à la station de base, de manière à créer le minimum d'interférences. La mise en œuvre de ces algorithmes de contrôle de puissance fait l'objet de nombreux travaux qui dépassent le cadre de cette thèse.

I.2.3 Le lien descendant de l'UMTS-FDD

L'UMTS-FDD sera le premier mode de l'UMTS à être déployé commercialement, probablement au cours de l'année 2002. Par abus de langage, ce mode est aussi appelé Wideband

CDMA, ou CDMA large bande[23].

Les équipements UMTS, que ce soit du côté du réseau ou du côté des terminaux, sont actuellement en phase de prototypage chez les constructeurs. Le lien descendant étant en général le facteur limitant de l'écoulement des données du fait des limites de la puissance embarquable dans les terminaux, c'est à l'optimisation des blocs du lien descendant que l'on s'intéressera.

I.2.3.1 Caractéristiques du lien descendant WB-CDMA

Les caractéristiques temporelles et fréquentielles du lien descendant de l'interface radio WB-CDMA sont résumées dans le tableau I.3.

Débit	3.84 Mcps (méga-chips par seconde)
Largeur d'un canal fréquentiel	5 MHz
Espacement des canaux fréquentiels	200 kHz
Durée d'un slot	666 μs (2560 chips)
Durée d'une trame	10 ms (15 slots)
Modulation	QPSK (Quaternary Phase Shift Keying)
Mise en forme	Racine de Nyquist (facteur d'excès de bande : 0.22)
Facteurs d'étalement possibles (en chips)	4, 8, 16, 32, 64, 128, 256, 512
Débits bruts correspondants	1920, 960, 480, 240, 120, 60, 30, 15 kbps
Multiplexage contrôle-données	Multiplexage en temps

TAB. I.3 – Paramètres significatifs du lien descendant WB-CDMA

Les principaux points qui diffèrent avec le lien montant sont :
- les facteurs d'étalement et les débits possibles,
- la façon dont sont multiplexés le contrôle et les données : sur le lien montant, le multiplexage se fait par voie (les données sont émises sur la voie I, le contrôle sur la voie Q), ce qui permet des débits différents pour les données et le contrôle,
- le processus de génération des signaux émis (présenté plus avant).

Il convient d'insister sur le fait que malgré l'emploi de *trames* et de *slots*, le mode WB-CDMA est bien un système CDMA-FDD. L'axe temporel a été divisé en trames elles-mêmes divisées en slots pour donner une structure logique au système et pour fournir une base de temps aux organes d'émission et de réception. Ainsi, un utilisateur émet en continu pendant

[23]La largeur d'un canal fréquentiel en WB-CDMA est de 5 MHz, ce qui justifie l'appellation « large bande » en comparaison avec la largeur d'un canal IS-95, par exemple, qui n'est que de 1.25 MHz. L'utilisation sans référence du qualificatif « large bande » constitue elle aussi un abus de langage habituellement toléré.

tous les slots de toutes les trames, et écoute sans discontinuer les émissions de la station de base. Les slots et les trames fournissent en particulier la base de temps du système de contrôle de puissance chargé de réguler les puissances d'émission des terminaux [3GP01e].

La modulation employée est une modulation QPSK, où un symbole émis véhicule deux bits d'information, un sur la voie I et un sur la voie Q, comme indiqué sur la constellation illustrée figure I.19.

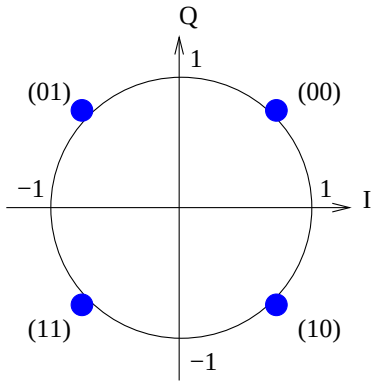


FIG. I.19 – Exemple de relation liant les bits à transmettre et les symboles QPSK émis

L'exemple illustré figure I.19 se traduit par :

Bits	Symbole QPSK émis
00	$\frac{1}{\sqrt{2}}(1+j)$
01	$\frac{1}{\sqrt{2}}(-1+j)$
10	$\frac{1}{\sqrt{2}}(1-j)$
11	$\frac{1}{\sqrt{2}}(-1-j)$

TAB. I.4 – Exemple de table de correspondance QPSK

Les symboles QPSK représentent des impulsions complexes, que l'on met en forme avant de moduler la porteuse émise dans le canal radiomobile. En effet, on ne sait pas émettre des créneaux parfaits d'une part, et d'autre part, le spectre du signal résultant serait beaucoup trop large et donnerait lieu à d'importantes interférences inter-symboles. Pour remédier à ce problème, les impulsions QPSK sont mises en forme avec un filtre dit de Nyquist, dont le spectre a l'allure d'un cosinus surélevé. Ce filtre, construit pour minimiser l'interférence inter-symbole dans le cas d'un canal à bande passante finie tel que le canal radiomobile, est équipartitionné entre l'émetteur et le récepteur, qui comportent donc chacun un filtre en *racine* de Nyquist, aussi appelé *demi-Nyquist*, dont le spectre est en *racine* de cosinus surélevé. Pour l'UMTS, ce filtre est décrit dans [3GP01a]. La réponse impulsionnelle du demi-Nyquist est donnée par l'équation I.5.

$$g\left(\frac{t}{T_c}\right) = \frac{4\beta\frac{t}{T_c}\cos\left((1+\beta)\,\pi\frac{t}{T_c}\right) + \sin\left((1-\beta)\,\pi\frac{t}{T_c}\right)}{\pi\frac{t}{T_c}\left(1-\left(4\beta\frac{t}{T_c}\right)^2\right)} \tag{I.5}$$

T_c est la durée d'un chip et β est le facteur de *roll-off*, ou d'adoucissement du spectre. Plus β est proche de 0, plus le filtre a un gabarit proche du créneau et une réponse impulsionnelle proche du sinus-cardinal, et plus β est proche de 1, plus le filtre a un gabarit aplati et large et une réponse impulsionnelle très courte (les lobes secondaires s'atténuent très vite). La réponse impulsionnelle donnée est anti-causale, mais est souvent tronquée à 3 ou 4 symboles de chaque côté de l'origine et rendue causale par introduction d'un retard. Ceci est illustré sur la figure I.20. Pour l'UMTS, le facteur de roll-off retenu est $\beta = 0.22$.

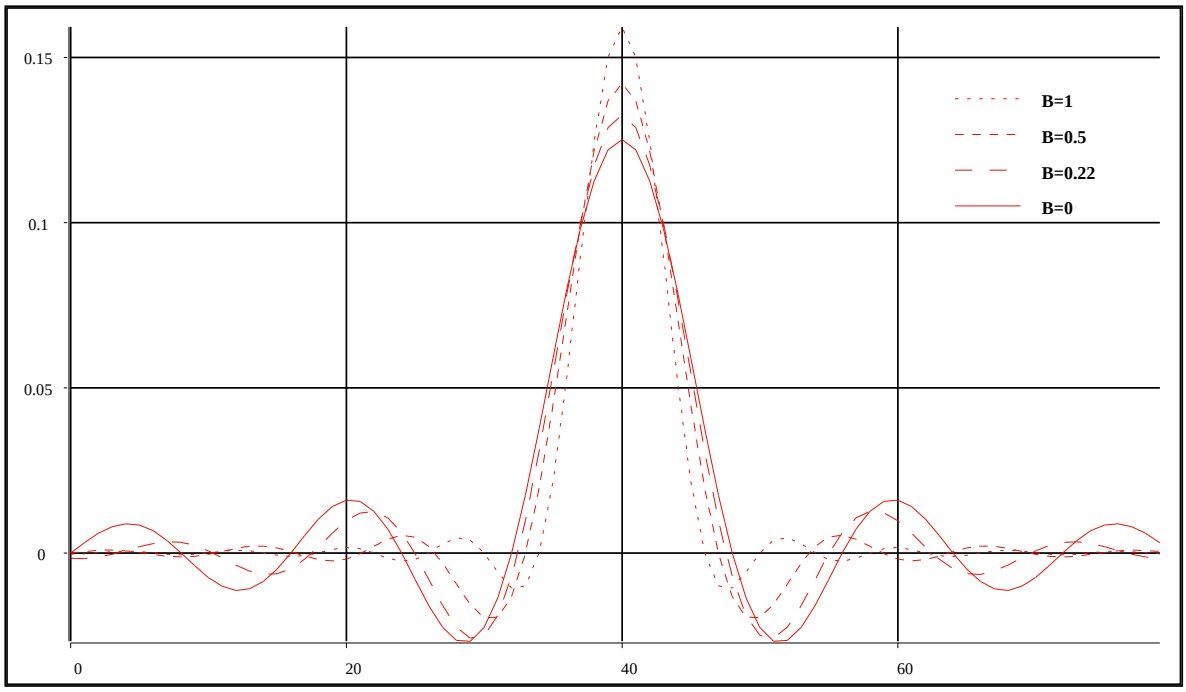


FIG. I.20 – Réponse impulsionnelle du filtre en racine de Nyquist pour diverses valeurs de β

I.2.3.2 Format des données sur le lien descendant

Comme tout système radiocellulaire, l'UMTS dispose de plusieurs types de canaux, appelés *canaux de transport*, pour véhiculer des informations de la station de base au terminaux et vice-versa. Il existe deux sortes de canaux de transport : les canaux *communs* et les canaux *dédiés*. Les canaux communs servent à la diffusion générale d'informations sur toute la cellule, sans viser un utilisateur particulier. Ces canaux véhiculent l'état du système, les fréquences utilisées par la cellule et ses voisines ... Les canaux dédiés, par contre, supportent effectivement une communication particulière entre la station de base et un terminal. Les différents types de canaux de transports, au nombre de sept (six types de canaux communs et un type de canal dédié) sont détaillés dans [3GP01b, section 4.1].

Toute information circulant sur le canal radio étant portée par un canal *physique*, il est nécessaire de multiplexer d'une manière ou d'une autre les différents canaux de transport sur les canaux physiques. On s'intéressera dans le cadre de cette thèse au canal physique dédié qui permet de faire transiter des informations de la station de base au terminal : le *DPCH* (*Dedicated Physical Channel*). Pour le bon fonctionnement de la communication, des informations

de contrôle sont envoyées en parallèle avec les données propres à la communication en cours. Ainsi, le contrôle est porté par le canal *DPCCH* (*Dedicated Physical* Control *CHannel*), tandis que les données sont portées par le canal *DPDCH* (*Dedicated Physical* Data *CHannel*). Ces deux canaux sont multiplexés en temps comme indiqué figure I.21 pour former le susdit *DPCH*.

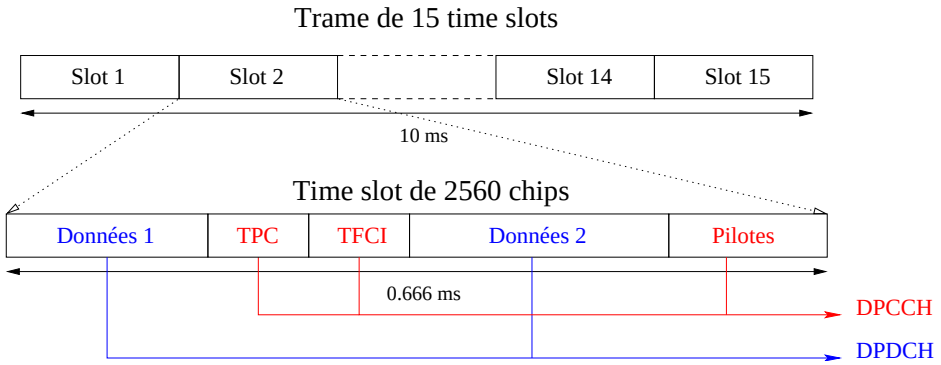


FIG. I.21 – Format des bursts émis sur le canal dédié du lien descendant en WB-CDMA

Sur le canal de contrôle DPCCH circulent trois types d'information :

- les bits *TPC* (Transmit Power Control) : ces bits indiquent au mobile s'il doit augmenter ou réduire sa puissance d'émission dans le cadre du contrôle de puissance déjà mentionné. Cette commande est calculée par la station de base en fonction des puissances reçues de la part des utilisateurs de la cellule de manière à recevoir les signaux de tous les mobiles avec la même puissance, ceci afin de limiter le *near-far effect*.
- les bits *TFCI* (Transport Format Combination Indicator) : ces bits indiquent de quel façon ont été codés les bits contenus dans les parties de données du burst, et renseignent le mobile sur la marche à suivre pour les décoder correctement. En effet, certains paramètres de codage tels que le facteur d'étalement ou le taux du code correcteur utilisé peuvent changer au cours de la communication (jusqu'à une fois par trame). Le mobile, n'étant pas forcément l'initiateur de ces changements, doit en être informé sous peine de ne plus pouvoir décoder les données reçues correctement. De plus, dans la situation où le service utilisé par le mobile est un service à bas débit dont les caractéristiques ne varient pas ou peu, il peut parfois être plus intéressant de ne pas transmettre ces bits TFCI pour récupérer un peu de débit utile. La présence des bits TFCI importe surtout lors de transmissions multiservices au cours desquelles plusieurs types de données (audio, vidéo, internet) codées de différentes manières circulent en même temps sur le canal dédié. En l'absence de ces bits TFCI, le mobile met en œuvre une procédure de *détection aveugle de débit* (Blind Rate Detection, en anglais) pour trouver seul le processus de codage ayant servi à l'émission. De tes algorithmes sont proposés dans [TL01], [TL99], ou encore [SM01].
- les bits *pilotes*, utilisés par le terminal pour réaliser l'estimation du canal radiomobile nécessaire à l'égalisation.

Les longueurs des différents champs du burst dépendent du débit des données transportées et de la présence ou non des bits *TFCI*. Les 17 possibilités sont récapitulées dans le tableau I.5, extrait de [3GP01b]. Pour les formats 12 à 16, si les bits TFCI ne sont pas transmis, leur emplacement reste réservé dans le burst, mais ils sont remplacés par des zéros.

Format de Slot	Débit Brut (kbps)	Débit Données (kbps)	SF	Bits / Slot	DPDCH Bits / Slot		DPCCH Bits / Slot		
					N_{Data1}	N_{Data2}	N_{TPC}	N_{TFCI}	N_{Pilot}
0	15	6	512	10	0	4	2	0	4
1	15	3	512	10	0	2	2	2	4
2	30	24	256	20	2	14	2	0	2
3	30	21	256	20	2	12	2	2	2
4	30	21	256	20	2	12	2	0	4
5	30	18	256	20	2	10	2	2	4
6	30	15	256	20	2	8	2	0	8
7	30	12	256	20	2	6	2	2	8
8	60	51	128	40	6	28	2	0	4
9	60	48	128	40	6	26	2	2	4
10	60	45	128	40	6	24	2	0	8
11	60	42	128	40	6	22	2	2	8
12	120	90	64	80	12	48	4	8	8
13	240	210	32	160	28	112	4	8	8
14	480	432	16	320	56	232	8	8	16
15	960	912	8	640	120	488	8	8	16
16	1920	1872	4	1280	248	1000	8	8	16

TAB. I.5 – Longueur des différents champs d'un burst en fonction du format de slot sur le lien descendant WB-CDMA

I.2.3.3 Les codes utilisés en WB-CDMA

Le processus d'étalement utilisé pour le lien descendant du WB-CDMA est légèrement plus complexe que celui décrit section I.2.2.1, mais peut aisément se ramener au cas général.

Le système de codage est un système hiérarchique à deux couches :

- une couche dite d'étalement (*spreading*, en anglais) chargée de multiplexer les différents DPDCH de la cellule en un seul flux. C'est lors de cette étape qu'est effectué l'étalement proprement dit, c'est-à-dire le passage du débit symbole (variable suivant les services utilisés) au débit chip (fixé par la norme à 3.84 Mcps). Les codes utilisés sont appelés codes *OVSF*, pour *Orthogonal Variable Spreading Factor*. Ces codes sont *à longueur variable*, ce qui permet le multiplexage de services de débits *différents* par simple sommation. De plus, l'utilisation de codes OVSF assure *l'orthogonalité* de ces divers services dans le cas de l'utilisation simultanée de codes de longueurs différentes.
- une couche dite d'embrouillage (*scrambling*, en anglais), dont le rôle est d'atténuer l'interférence inter-cellule. C'est cette couche qui confère le caractère pseudo-aléatoire au signal devant être émis par la station de base en multipliant chip à chip le flux issu de la couche d'étalement et une séquence particulière appelée code d'embrouillage. Aucun changement de rythme n'intervient lors de cette étape.

Pour un service donné, c'est le produit du code d'étalement utilisé lors de la première phase par le code d'embrouillage utilisé lors de la seconde phase qui constitue véritablement la *séquence étalante* introduite section I.2.2.1.

Les codes d'étalement

Les codes d'étalement, ou codes OVSF, sont attribués aux utilisateurs en fonction des débits que leur alloue le réseau. Le processus de génération de ces codes, détaillé dans [ASO97], est le suivant : soit C_N l'ensemble de N séquences binaires de longueur N, notées $\{C_{N,n}\}_{n\in[1,N]}$, avec $N = 2^{K-1}$ (K est un entier strictement positif). C_N est obtenu à partir de $C_{N/2}$ d'après l'équation I.6, où $\tilde{}$ dénote la complémentation binaire.

$$C_N = \begin{bmatrix} C_{N,1} \\ C_{N,2} \\ C_{N,3} \\ C_{N,4} \\ \vdots \\ C_{N,N-1} \\ C_{N,N} \end{bmatrix} = \begin{bmatrix} C_{N/2,1}C_{N/2,1} \\ C_{N/2,1}\tilde{C}_{N/2,1} \\ C_{N/2,2}C_{N/2,2} \\ C_{N/2,2}\tilde{C}_{N/2,2} \\ \vdots \\ C_{N/2,N/2}C_{N/2,N/2} \\ C_{N/2,N/2}\tilde{C}_{N/2,N/2} \end{bmatrix} \tag{I.6}$$

La structure hiérarchique obtenue avec un tel processus de génération est illustrée figure I.22.

Le système peut être initialisé avec $C_{1,1} = 1$, pour construire des séquences de Walsh-Hadamard, ou à partir d'une couche supérieure avec d'autres codes orthogonaux, comme des codes de Gold, par exemple.

L'ensemble C_N forme un ensemble de séquences de Walsh othogonales les unes aux autres. De plus, deux codes appartenant à des couches différentes C_N et C_M sont orthogonaux en l'absence de relation père-fils entre ces deux codes. Par exemple, l'allocation à un canal d'un quelconque code de la suite $\{C_{1,1}, C_{2,1}, C_{4,1}, C_{8,1}, C_{16,2}, C_{32,3}, C_{64,5}\}$ empêche l'allocation de tout

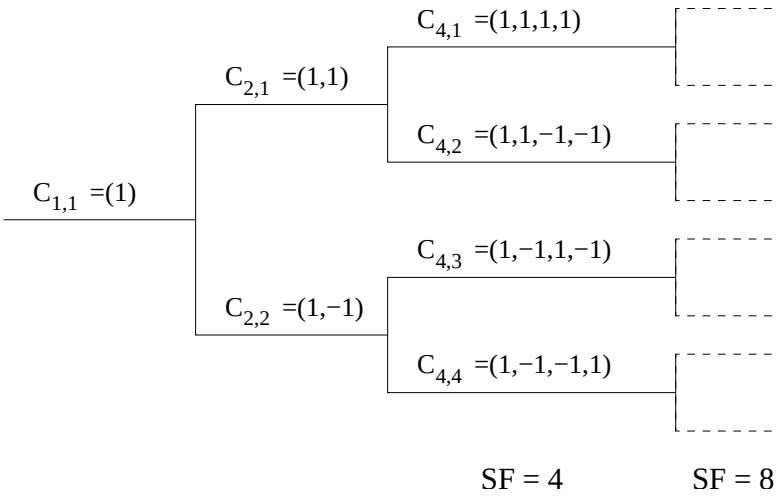


FIG. I.22 – Structure hiérarchique de l'ensemble des codes d'étalement en WB-CDMA

autre code de la suite à un autre canal, l'orthogonalité n'étant alors plus préservée. Les codes courts sont alloués aux utilisateurs ayant besoin de débits importants, alors que les codes longs, disponibles en plus grand nombre, sont alloués aux utilisateurs n'ayant besoin que de faibles débits, typiquement équivalents à celui d'une communication téléphonique.

La limite du débit total transmissible provient du nombre de codes disponibles. Une solution à ce problème est l'utilisation de plusieurs codes de scrambling dans une seule et même cellule, envisagée dans [3GP01d, section 5.2.2], qui multiplierait le nombre d'arbres de codes OVSF disponibles dans la cellule.

Les codes d'embrouillage

Les codes d'embrouillage (ou de scrambling) sont des séquences complexes constituées à partir de segments de séquences de Gold réelles elles-mêmes générées à l'aide de registres à décalage configurés rebouclés comme indiqué figure I.23. Ils sont utilisés pour atténuer l'interférence intercellule et lui donner les caractéristiques spectrales d'une interférence qui serait due uniquement à un bruit blanc.

Les étapes de génération des codes de scrambling sont les suivantes :

– deux séquences x et y sont construites comme suit :

$$x(i+18) = x(i+7) \oplus x(i), i = 0, \ldots, 2^{18} - 20 \quad \text{(I.7a)}$$

$$y(i+18) = y(i+10) \oplus y(i+7) \oplus y(i+5) \oplus y(i), i = 0, \ldots, 2^{18} - 20 \quad \text{(I.7b)}$$

– avec les conditions initiales suivantes :

$$x(0) = 1, x(1) = x(2) = \ldots = x(16) = x(17) = 0 \quad \text{(I.8a)}$$

$$y(0) = y(1) = \ldots = y(16) = y(17) = 1 \quad \text{(I.8b)}$$

– les séquences $z_n, n = 0, 1, 2, \ldots, 2^{18} - 2$ sont définies par :

$$z_n(i) = x((i+n) \text{ modulo } (2^{18} - 1)) \oplus y(i), i = 0, 1, 2, \ldots, 2^{18} - 2 \quad \text{(I.9)}$$

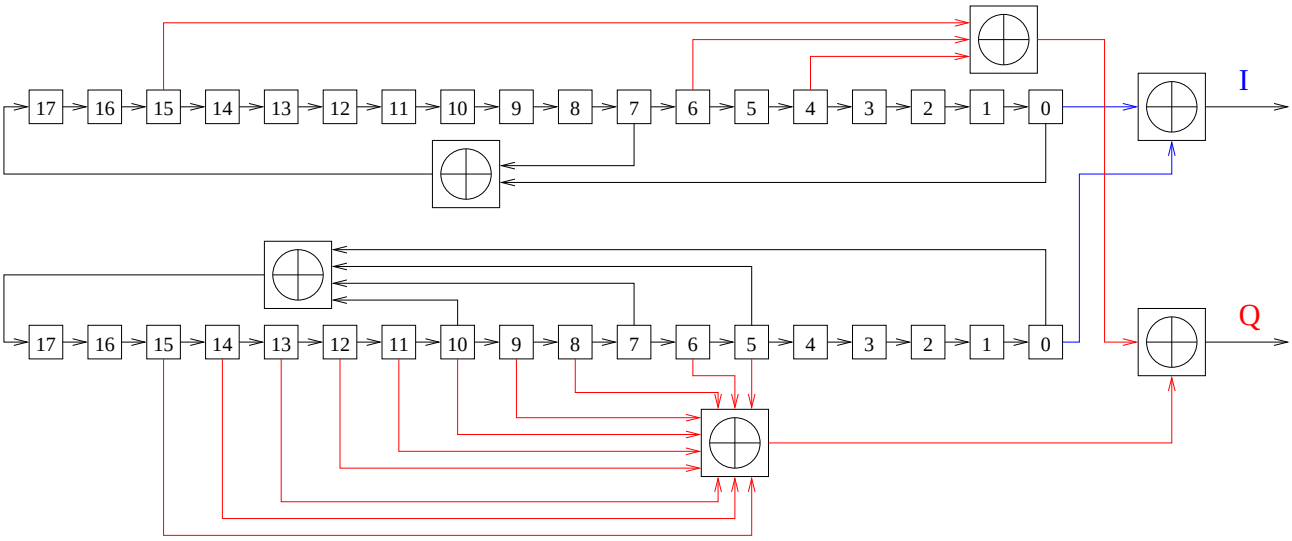


FIG. I.23 – Configuration des registres à décalage générant les codes de scrambling

– et sont converties en séquences à valeurs réelles Z_n par la transformation suivante :

$$Z_n(i) = \begin{cases} +1 & \text{if } z_n(i) = 0 \\ -1 & \text{if } z_n(i) = 1 \end{cases} \quad \text{pour } i = 0, 1, 2, \ldots, 2^{18} - 2 \tag{I.10}$$

– pour finir, le n-ième code de scrambling complexe S_n est défini par :

$$S_n(i) = Z_n(i) + jZ_n((i + 131072) \text{ modulo } (2^{18} - 1)), i = 0, 1, 2, \ldots, 38399 \tag{I.11}$$

Le code de scrambling est répété à chaque trame, et est spécifique à la cellule. Un seul code de scrambling est normalement utilisé dans chaque cellule, mais d'autres codes peuvent lui être associés en cas d'insuffisance de codes d'étalement, comme expliqué dans [3GP01d, section 5.2.2].

I.2.3.4 Processus de génération des symboles émis

Le processus de multiplexage de plusieurs services de débits éventuellement différents est illustré figure I.24. Chaque service est étalé avec un facteur d'étalement adapté au débit, puis c'est la somme des services étalés qui est embrouillée en une seule étape. Dans un premier temps, les divers DPDCH et l'*unique*[24] DPCCH subissent une conversion série-parallèle, puis l'étalement, puis les signaux issus de l'étalement des bits pairs (resp. impairs) sont ajoutés les uns aux autres. Une suite de chips complexes est alors formée avec comme partie réelle la somme des signaux issus de l'étalement des bits pairs et comme partie imaginaire la somme des signaux issus de l'étalement des bits impairs. Cette suite complexe est alors embrouillée avec le code attribué à la cellule, avant d'être mise en forme puis émise.

[24]Il n'y a qu'un seul DPCCH par utilisateur. En effet, il ne peut y avoir qu'une seule commande de contrôle de puissance, de même qu'un seul indicateur de format. Les autres flux intervenant lors du multiplexage comportent des zéros en lieu et place du DPCCH du premier flux. Si la station émet plusieurs DPDCH, elle peut éventuellement augmenter la puissance du DPCCH afin de faciliter l'estimation de canal et le décodage des bits de contrôle en général.

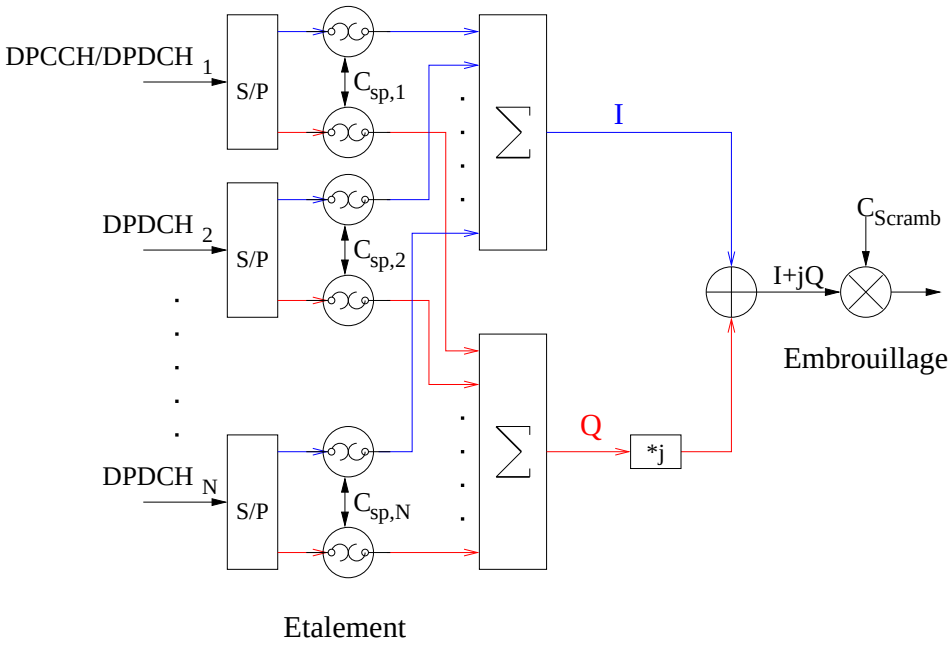


FIG. I.24 – Multiplexage de différents flux sur le lien descendant WB-CDMA

En appelant
- $N > 0$ le nombre de flux à multiplexer,
- $\{SF_i\}_{i\in[1,N]}$ les facteurs d'étalement des divers services à multiplexer, dépendant directement des débits de ceux-ci,
- $\{n_i\}_{i\in[1,SF_i]}$ les numéros des codes d'étalement $\{C_{SF_i,n_i}\}_{i\in[1,N]} = \left\{(C_{i,k})_{k\in[0,SF_i-1]}\right\}_{i\in[1,N]}$ attribués à chaque service,
- $\{b_i\}_{i\in[1,N]} = \left\{(b_{i,j})_{j\in\left[0,\frac{5120}{SF_i}-1\right]}\right\}_{i\in[1,N]}$ les séquences réelles bivaluées à transmettre pour chaque service,
- $C_{SC} = (C_{SC,l})_{l\in[0,38399]}$ le code de scrambling attribué à la cellule,
- $r \in [0, 14]$ la position du slot dans la trame,

les flux pairs $\left\{\left(b^I_{i,j}\right)_{j\in\left[0,\frac{2560}{SF_i}-1\right]}\right\}_{i\in[1,N]}$ et impairs $\left\{\left(b^Q_{i,j}\right)_{j\in\left[0,\frac{2560}{SF_i}-1\right]}\right\}_{i\in[1,N]}$ s'écrivent :

$$\forall i \in [1,N], \forall j \in \left[0, \frac{2560}{SF_i} - 1\right], \begin{cases} b^I_{i,j} = b_{i,2j} \\ b^Q_{i,j} = b_{i,2j+1} \end{cases} \tag{I.12}$$

Les mêmes flux deviennent, après étalement :

$$\forall i \in [1,N], \forall j \in \left[0, \frac{2560}{SF_i} - 1\right], \forall k \in [0, SF_i - 1], \begin{cases} \hat{b}^I_{i,j\times SF+k} = b^I_{i,j} * C_{i,k} \\ \hat{b}^Q_{i,j\times SF+k} = b^Q_{i,j} * C_{i,k} \end{cases} \tag{I.13}$$

Les flux pairs (resp. impairs) sont sommés :

$$\forall l \in [0, 2559], \begin{cases} I_l = \sum_{i=1}^{N} \hat{b}_l^I \\ Q_l = \sum_{i=1}^{N} \hat{b}_l^Q \end{cases} \tag{I.14}$$

Puis, la séquence complexe résultante est embrouillée avec la portion du code de scrambling correspondant au slot courant :

$$\forall l \in [0, 2559], s_l = (I_l + jQ_l) C_{SC,r\times 2560+l} \tag{I.15}$$

Les chips s_l sont ensuite mis en forme avec un filtre en racine de Nyquist réel puis aiguillés vers le modulateur RF.

Le processus présenté ci-dessus ne concerne que les canaux dédiés à un abonné utilisant éventuellement plusieurs services en même temps. Il faut garder à l'esprit que ce n'est qu'une partie des informations émises à destination du mobile, l'autre partie étant constituée de la multitude de canaux diffusant des informations de contrôle, d'état des cellules, de synchronisation…, ce qui laisse présager de la complexité globale du système et de la puissance de calcul devant être embarquée dans le mobile.

Chapitre II

Le bloc de réception du terminal UMTS-FDD

Le mobile doit démoduler en permanence un certain nombre de canaux, tous étalés selon le principe du WB-CDMA présenté au chapitre précédent. La complexité de cette tâche est accrue par le fait que le canal radiomobile dégrade les informations émises et introduit des erreurs de transmission, qui peuvent être détectées et parfois corrigées par les traitements effectués par le terminal.

Sans aborder les dispositifs de détection et correction d'erreurs décrits dans [3GP01c], le présent chapitre présente le canal radiomobile et les dégradations qu'il fait subir aux signaux émis, ainsi que les moyens habituellement employés pour diminuer l'influence de ces dégradations sur la qualité du signal fourni au dispositif d'estimation des symboles transmis. L'accent est mis sur le bloc de réception car c'est dans cette partie du terminal qu'ont lieu le plus de calculs en bande de base : l'égalisation et le décodage canal sont de loin les deux tâches les plus complexes effectuées par le terminal. Un dispositif classique d'égalisation en CDMA, le récepteur Rake, est ensuite présenté en détail.

Du fait de l'importance qu'elle revêt en CDMA, l'estimation du canal fait l'objet de la dernière partie de ce chapitre. Une solution algorithmique à ce problème d'identification est proposée. Sa complexité est évaluée, puis réduite par une optimisation de l'algorithme lui-même *et* une simplification des opérations élémentaires effectuées.

II.1 Le canal radiomobile

Les signaux émis par les stations de base et les mobiles transitent par le canal radiomobile sous la forme d'une onde électromagnétique dont la fréquence porteuse est située dans les zones identifiées figure I.10. Les conditions d'utilisation des terminaux mobiles (débits transmis, fréquences utilisées, mobilité des utilisateurs ...) rendent délicate la réception optimale des signaux émis par la station de base. En effet, plusieurs phénomènes viennent entâcher le signal reçu d'erreurs que le mobile et la station de base doivent détecter et prendre en compte lors du processus de réception.

II.1.1 La propagation multi-trajet

Les positions relatives des antennes de la station de base, en hauteur le plus souvent, et de celle du mobile, entre un et deux mètres au-dessus du sol, font que celui-ci reçoit dans la bande qui l'intéresse les contributions de tout un ensemble de réflecteurs et de diffracteurs. Ce phénomène est connu sous le nom de *propagation multi-trajet*, et est illustré figure II.1.

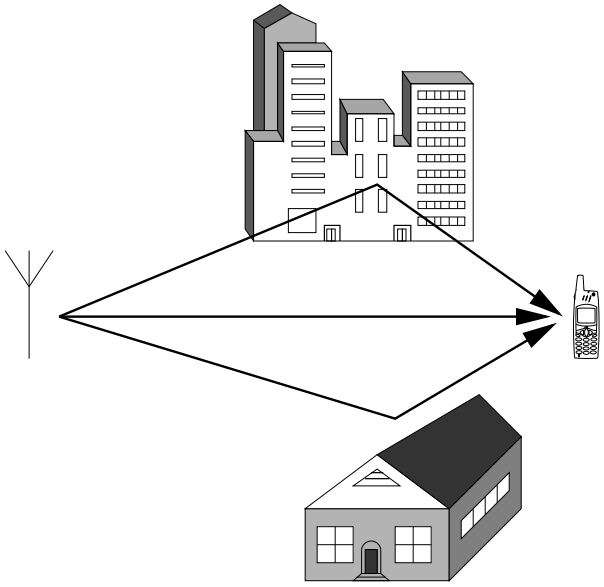


FIG. II.1 – Exemple de propagation par trajets multiples

La réponse impulsionnelle d'un tel canal est tracée figure II.2. Les différentes répliques du signal original qui sont reçues par le mobile ont des amplitudes, des phases, et des retards différents. Le temps de propagation d'une information selon les différents trajets est variable avec la longueur de chaque trajet, ce qui entraîne une distortion préjudiciable à la reconnaissance du message transmis. A l'arrivée dans le récepteur, on peut avoir une somme constructive de trajets lorsqu'ils sont relativement en phase les uns avec les autres, ou bien une somme destructive lorsqu'ils sont en opposition de phase. Ce phénomène est appelé *évanouissement*, ou *fading* en anglais ([OP98b, section 4.2]). Les deux principales sortes de canaux à fading sont les canaux à fading de *Rayleigh*, où aucun trajet ne prédomine vraiment, et les canaux à fading de *Rice*, où on trouve un trajet dominant et des trajets secondaire de plus faible puissance. Ce dernier cas se produit principalement à l'intérieur des bâtiments, dans des micro- ou pico-cellules, où il existe une ligne de vue directe, qui constitue ainsi le trajet dominant.

La propagation par trajets multiples affecte les paramètres de la transmission de la façon suivante ([RCS88, chapitre 4]) :

– la fréquence par l'effet Doppler,
– l'amplitude par le fading de Rayleigh (ou de Rice, pour les types de fading les plus connus),

II.1.1.1 L'effet Doppler

L'effet Doppler est un décalage fréquentiel de toute forme d'onde transmise ou reçue par un mobile en déplacement.

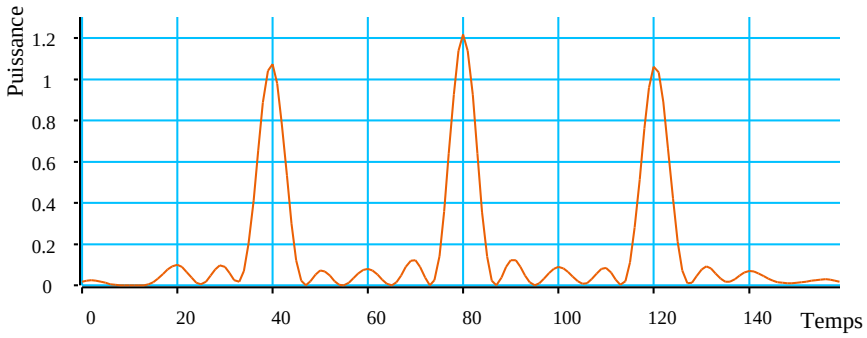


FIG. II.2 – Exemple de module de réponse impulsionnelle d'un canal à trajets multiples

Si F est la fréquence de l'onde transmise, v la vitesse du mobile, c la vitesse de propagation de l'onde[1] et α l'angle formé par le vecteur vitesse du mobile avec le vecteur de propagation de l'onde, l'effet Doppler déplace cette fréquence d'une quantité :

$$f_d = F\frac{v}{c}\cos\alpha \tag{II.1}$$

On appelle *spectre Doppler* le spectre observé en réception en réponse à une fréquence porteuse non modulée (soit un signal $s(t)$ constant en bande de base à l'émission). L'écart entre la fréquence du signal reçu et la fréquence émise est au maximum égal à f_d. La forme du spectre dépend de la disposition des obstacles et de la vitesse du mobile.

De la même façon que des trajets multiples élargissent une impulsion, l'effet Doppler, dans le cas d'un mobile en déplacement, va élargir une raie spectrale en une répartition d'énergie sur tout un intervalle de fréquence.

II.1.1.2 Le fading

C'est Rice qui, à partir de 1944 ([Ric44], [Ric45], [Ric48]), s'est intéressé à la réception d'un signal de fréquence F assorti d'un bruit à bande étroite centré en F.

L'amplitude du champ électrique reçu par le mobile en déplacement présente des évanouissements profonds régulièrement espacés, qui se produisent à une fréquence à peu près égale à 2 fois la fréquence Doppler f_d correspondant à la vitesse de déplacement du véhicule.

En effet, le mobile se déplace dans la géométrie des ondes stationnaires créées par interférence entre les ondes d'amplitude variée, diffractées ou réfléchies à partir de l'onde émise par toute une série d'obstacles. La distance entre noeuds de l'amplitude du champ reçu est ainsi de $\lambda/2$ pour une onde monochromatique de longueur d'onde λ. Le mobile parcourant cette distance à la vitesse v, l'espacement temporel entre deux évanouissements est donc de l'ordre de $\lambda/2v$.

Rayleigh s'est quant à lui intéressé au problème du mobile « encaissé », dans lequel aucune ligne de vue directe n'est présente. Ce modèle, simplifié par rapport à celui de Rice, correspond plus au milieu urbain, ou l'onde directe est souvent absente du continuum de fréquences reçu par le mobile. Ce modèle est aussi plus sévère que celui de Rice, donnant naissance à des évanouissements plus profonds que la réalité observée, et fournissant ainsi un bon exemple de « pire cas ».

[1] Pour une onde électromagnétique se déplaçant dans l'air, c est égal à la vitesse de la lumière dans le vide, c'est-à-dire 3.10^8 m/s.

En considérant un environnement isotrope, le spectre Doppler en bande de base associé à un trajet d'atténuation complexe α_i est donné [Cla68] par :

$$S_{\alpha_i}(\nu) = \begin{cases} \frac{\sigma^2_{\alpha_i}}{\pi f_d \sqrt{1-\left(\frac{\nu}{f_d}\right)^2}} & \text{pour } |\nu| \leq f_d, \\ 0 & \text{pour } |\nu| > f_d \end{cases} \tag{II.2}$$

Ce spectre, classiquement appelé *spectre en U*, est représenté figure II.3.

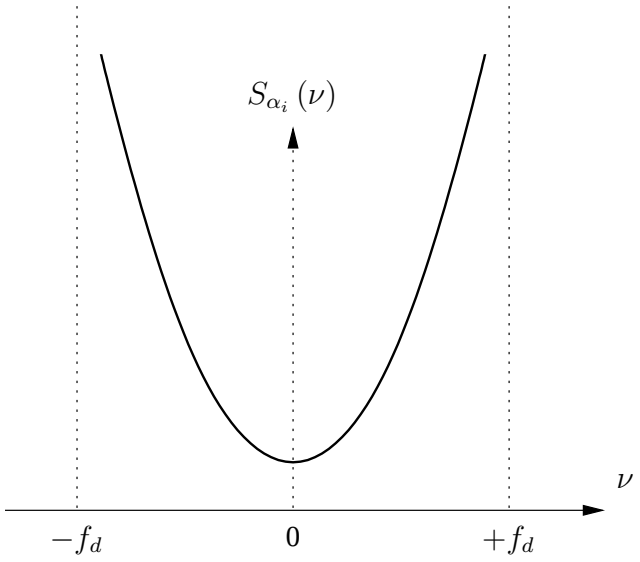


FIG. II.3 – Spectre Doppler en U (en bande de base) associé à un "trajet" en environnement isotrope

II.1.2 Caractérisation du canal radiomobile

Les grandeurs traditionnellement utilisées pour caractériser le canal radiomobile sont ([BB99, chapitre 13]) :

- la *fenêtre de retard* ou *durée d'étalement* (*delay spread* en anglais) : dans un environnement à trajets multiples en présence de mobiles, le canal est caractérisé par sa réponse impulsionnelle $h(\tau, t)$, où t désigne le temps et τ désigne le paramètre de retard de la corrélation. Une impulsion de courte durée émise à l'instant t va être reçue sous la forme d'une impulsion élargie de durée T_m, appelée fenêtre de retard ou durée d'étalement. Lorsque la durée de cette fenêtre d'étalement est supérieure à la durée d'un symbole élémentaire[2], les symboles transmis « bavent » les uns sur les autres : c'est l'interférence inter-symbole.
- la *bande de cohérence* B_{coh}, habituellement prise égale à l'inverse de la fenêtre d'étalement, est la largeur de la plus petite bande qui sépare deux composantes fréquentielles du signal ne subissant pas la même atténuation. Un canal dont la bande de cohérence B_{coh} est plus petite que la bande B_x occupée par un signal donné altérera différemment les diverses composantes fréquentielles de ce signal : le canal est alors *sélectif en fréquences*.

[2]Dans le cas du CDMA, la durée à considérer est celle d'un chip.

– l'*étalement Doppler* : les trajets multiples arrivant au mobile en mouvement sont perçus avec des décalages fréquentiels dûs à l'effet Doppler. L'étalement Doppler est défini comme le décalage Doppler le plus important par rapport au signal d'origine. L'étalement Doppler est le dual de la fenêtre de retard dans le domaine fréquentiel.
– le *temps de cohérence* T_{coh}, dual en temps de la bande de cohérence, est égal à l'inverse de l'étalement Doppler, et représente la plus courte durée qui doit séparer deux impulsions pour qu'elles soient atténuées différemment par le canal. Le temps de cohérence traduit la vitesse de variation des caractéristiques du canal. Si le temps de cohérence du canal est inférieur au débit symbole (au débit chip pour le CDMA) T_x, le signal transmis subit de la distortion intra-symbole. Le canal est alors *sélectif en temps*.

Les canaux radiomobiles peuvent donc être caractérisés selon deux grandeurs : la bande de cohérence B_{coh} et le temps de cohérence T_{coh}. Cette classification est illustrée figure II.4.

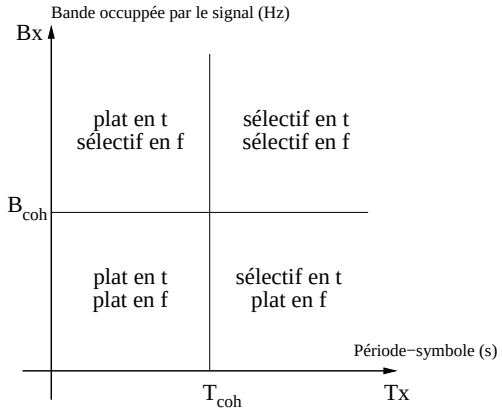


FIG. II.4 – Classification des canaux radiomobiles selon leur sélectivité en temps et en fréquence

Ni le modèle plat en temps *et* en fréquence ni le modèle sélectif en temps *et* en fréquence ne sont particulièrement proches du canal radiomobile perçu par un mobile en déplacement à la surface du sol. Seuls les canaux sélectifs, en temps *ou* en fréquence, seront donc pris en compte par la suite.

II.1.3 Modélisation et approximation du canal radiomobile

Pour un canal à N trajets multiples dont les caractéristiques ne varient pas dans le temps, la réponse impulsionnelle s'écrit :

$$h(\tau) = \sum_{i=1}^{N} a_i e^{j\theta_i} \delta(\tau - \tau_i) \quad \text{(II.3a)}$$

$$= \sum_{i=1}^{N} c_i \delta(\tau - \tau_i) \quad \text{(II.3b)}$$

a_i représente l'amplitude positive du trajet i, θ_i son déphasage, et τ_i son retard. c_i représente ici l'atténuation complexe du trajet i, et est égal à $a_i e^{j\theta_i}$. Une représentation graphique d'un tel modèle de canal est représenté figure II.5. L'atténuation complexe de chacun des trajets est

représentée dans un repère complexe *propre au trajet* et centré sur le point de l'axe des retards correspondant au retard du trajet en question[3].

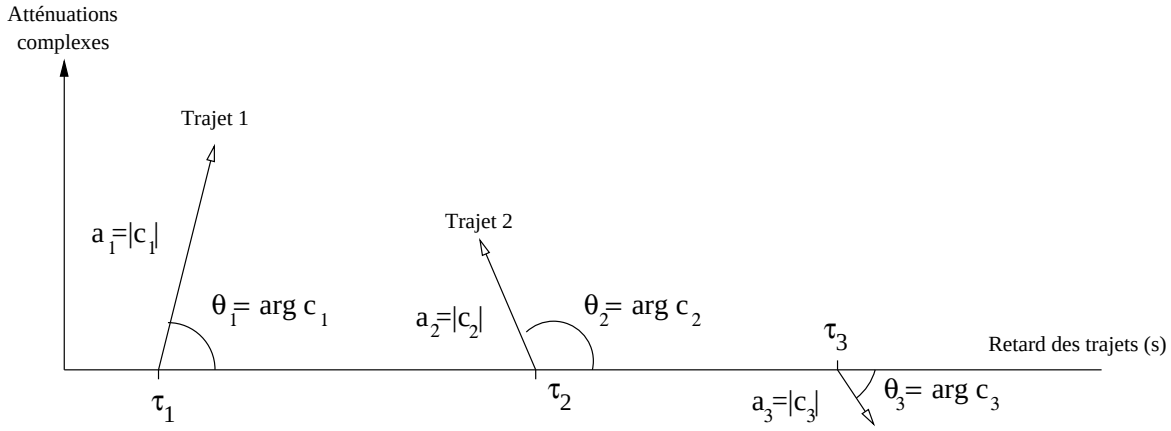


FIG. II.5 – Réponse impulsionnelle complexe d'un canal multi-trajet

Les canaux réels, en particulier en environnement radiomobile, ayant typiquement une réponse impulsionnelle variant dans le temps, il faut étendre le modèle formulé précédemment. La réponse impulsionnelle à l'instant t_0 d'un canal à trajets multiples variant dans le temps s'écrit :

$$h(\tau, t_0) = \sum_{i=1}^{N(t_0)} a_i(t_0)\, e^{j\theta_i(t_0)} \delta(\tau - \tau_i(t_0)) \quad \text{(II.4a)}$$

$$= \sum_{i=1}^{N(t_0)} c_i(t_0)\, \delta(\tau - \tau_i(t_0)) \quad \text{(II.4b)}$$

Le paramètre N est rendu variable dans le temps par la mobilité du terminal et éventuellement par celle des obstacles qui contribuent à créer des trajets secondaires. En effet, le passage d'un coin d'immeuble par un terminal peut faire apparaître très rapidement une ligne de vue directe ou au contraire la supprimer, selon que le mobile entre ou sorte de la zone ombragée (au sens radioélectrique du terme) par l'immeuble en question. Les canaux donnés en exemple dans les documents de normalisation [3GP01a, annexe B] comprennent, outre des canaux à trajets multiples classiques, un canal à deux trajets dont l'un s'éloigne et se rapproche de l'autre sinusoïdalement au cours du temps, et un canal à deux trajets qui disparaîssent et réapparaîssent périodiquement à des retards différents.

Après le passage à travers les différents trajets secondaires, un bruit $n(t)$ s'ajoute à la somme de copies du signal original. Ce bruit est supposé blanc et gaussien.

Pour des raisons de simplification des simulations, certaines approximations du modèle de canal établi ci-dessus ont été faites :

– l'effet Doppler n'a pas été pris en compte. Celà limite la validité des simulations à des situations de faible mobilité, ou du moins à des vitesses réduites. La variation de phase

[3]Sur la figure, les trajets les plus puissants apparaissent avec les retards les plus faibles, ce qui n'est pas toujours le cas dans la réalité.

maximale due à l'effet Doppler est de 1,4^o pour une vitesse de 3 km/h, et de 50^o pour 120 km/h [Ros01]. Le récepteur envisagé n'estimant le canal qu'une fois par slot, les variations du canal intra-slot ne seront pas prises en compte, ce qui limite effectivement le fonctionnement aux vitesses réduites. Il sera par contre possible de mettre le récepteur en défaut en simulant une transmission à vitesse élevée donnant lieu à un important écart Doppler.

– pour les mêmes raisons, on considère que le canal est stationnaire pendant la durée d'un slot. Les c_i et les τ_i sont donc fixes pour la durée d'un slot et sont estimés une fois par slot pour chaque trajet. Le très faible rapport de la vitesse du mobile sur la vitesse de l'onde radioélectrique permet de considérer les *retards* comme fixes pendant un slot, mais oblige à considérer les atténuations complexes comme variables au cours du temps dans le cas d'une vitesse élevée.
– le nombre de trajets N est aussi considéré comme invariable pendant un slot. L'erreur commise en cas d'apparition ou de disparition d'un trajet est suffisamment faible pour légitimer cette hypothèse, et est de plus minimisée par le processus de recombinaison des trajets, du moins pour un récepteur Rake comme celui envisagé par la suite.

II.2 Le récepteur Rake

Il a été montré précédemment que du fait de la propagation dans le canal radiomobile, le signal émis arrive au récepteur sous la forme d'une suite de répliques retardées et déphasées qui peuvent s'additionner de manière aussi bien constructive que destructive.

Nous allons voir dans cette partie consacrée au récepteur Rake qu'il est possible de tirer parti de la présence de cet ensemble de répliques. En effet, en les recombinant judicieusement, on peut *exploiter* la diversité offerte [PG58].

II.2.1 Notions de diversité

Les courbes de probabilités d'erreur indiquent qu'une transmission à travers un canal sélectif en fréquence est beaucoup plus sensible au RSB (rapport signal-à-bruit, ou SNR en anglais, pour *signal-to-noise ratio*) qu'une transmission à travers un canal à bruit blanc additif gaussien. Pour obtenir la même probabilité d'erreur, il va donc falloir émettre avec une puissance supérieure en présence de fading, sous peine de voir le récepteur incapable de démoduler correctement le signal transmis du fait du fading.

Pour combattre le fading et réduire ainsi les besoins en puissance d'émission, une technique efficace consiste à fournir de la *diversité* au récepteur. Le fading transformant le RSB à la sortie du canal en un processus aléatoire ([BB99, chapitre 13]), l'idée sous-jacente à l'introduction de la diversité est de transmettre le signal à travers L canaux indépendamment sélectifs, et de choisir ces canaux de manière à fournir L répliques du signal émis au récepteur, ces répliques ayant subi des fadings aussi indépendants que possible. Ainsi, à chaque instant, la probabilité est élevée qu'au moins *une* des L branches de diversité ne subisse pas un évanouissement profond. Le SNR final a donc des chances raisonnables d'être suffisamment élevé pour permettre une démodulation du signal émis de bonne qualité.

Exploiter la diversité revient donc à profiter du fait que plusieurs copies *discernables* et

décorrélées de la même information sont disponibles à l'entrée de l'organe responsable de l'estimation de cette information.

II.2.1.1 Les différentes formes de diversité

Les quatre principales formes de diversité sont :
- la diversité d'espace : la réception du signal transmis s'effectue sur L antennes simultanément. Les antennes doivent être suffisamment espacées relativement à la longueur d'onde de la porteuse pour assurer une décorrélation suffisante. Cette technique est difficilement appliquable à un terminal de taille réduite, pour des raisons évidentes d'encombrement (voir à ce sujet les études menées dans [Pig00] et [Ros01]).
- la diversité de polarisation : dans le cas où le canal radiomobile atténue différemment des signaux transmis selon des polarisations orthogonales, la diversité peut être exploitée en utilisant deux antennes polarisées différemment sur le récepteur. Ces antennes n'ont alors pas besoin d'être aussi distantes que pour la diversité d'espace, ce qui constitue un avantage certain du point de vue du terminal. Par contre, cette technique ne fournit que deux branches de diversité, alors que la diversité d'espace peut théoriquement en fournir autant que l'on souhaite. De plus, elle est difficile à appliquer à un récepteur *mobile*.
- la diversité de fréquence : cette forme de diversité est obtenue en transmettant le même signal sur plusieurs porteuses différentes, séparées par une bande de fréquence plus large que la bande de cohérence du canal, toujours pour assurer une décorrélation suffisante. Cette technique, gourmande en ressources spectrales, est rarement utilisée.
- la diversité en temps : cette diversité, duale de la diversité en fréquence, est obtenue en transmettant le même signal à différents instants, séparés par un temps supérieur au temps de cohérence du canal. Les récepteurs mobiles se déplaçant lentement percevant typiquement un temps de cohérence assez important, cette technique ne peut être mise en œuvre qu'au prix de délais importants.

II.2.1.2 La diversité de trajet

Grâce au CDMA, il est possible de faire apparaître la propagation multi-trajet comme une forme de diversité.

En effet, en effectuant une corrélation glissante avec la séquence étalante, qui va faire ressortir un trajet particulier en atténuant les autres, on peut parvenir à traiter chaque trajet de manière isolée puis à les recombiner par la suite. Les différents trajets, une fois isolés les uns des aures, peuvent donc être considérés comme autant de branches de diversité.

Dans la suite de ce document, on assimilera le terme « diversité » à « diversité de trajet », cette dernière étant la forme de diversité exploitée par le récepteur Rake.

II.2.1.3 Les schémas de recombinaison

Pour exploiter une quelconque forme de diversité, il faut décider de la manière dont on construit le signal optimal à partir des L branches de diversité présentes à l'entrée du récepteur, le but étant de fournir le RSB le plus élevé possible au récepteur proprement dit.

On peut écrire le signal reçu $s_r(t)$ comme la somme des diverses copies $r_i(t)$ bruitées et

retardées du signal original $s(t)$ de la manière suivante :

$$r_i(t) = \beta_i(t)\, s(t-\tau_i) + n_i(t) \tag{II.5a}$$

$$s_r(t) = \sum_{i=1}^{N(t)} r_i(t) \tag{II.5b}$$

La recombinaison s'effectue en général sous la forme d'une combinaison linéaire des signaux reçus sur les différentes branches de diversité. Si on émet l'hypothèse que les retards des différentes copies entre elles sont inférieurs à la durée d'un symbole, on obtient :

$$\hat{s}(t) = \alpha_1 r_1(t) + \cdots + \alpha_{N(t)} r_{N(t)}(t) \tag{II.6a}$$

$$= \sum_{i=1}^{N(t)} \alpha_i r_i(t) \tag{II.6b}$$

Dans le cas contraire, il faut resynchroniser les copies entre elles avant la recombinaison sous peine d'additionner des portions de signal ne se rapportant pas au même symbole émis.

Dans la pratique, les coefficients α_i peuvent éventuellement dépendre des statistiques de $r_i(t)$. La combinaison devient alors « quasi-linéaire ».

Les principaux schémas de recombinaison sont les suivants :

– la recombinaison par sélection : à chaque instant, un seul α_i est différent de 0, de préférence celui qui correspond à la branche ayant le RSB le plus élevé. Cette technique offre l'avantage d'une sélection instantanée de la meilleure branche, au prix d'une limitation des performances. En effet, le RSB en sortie n'est au mieux qu'égal au plus élevé des SNR en entrée. Ce schéma de recombinaison est illustré figure II.6.

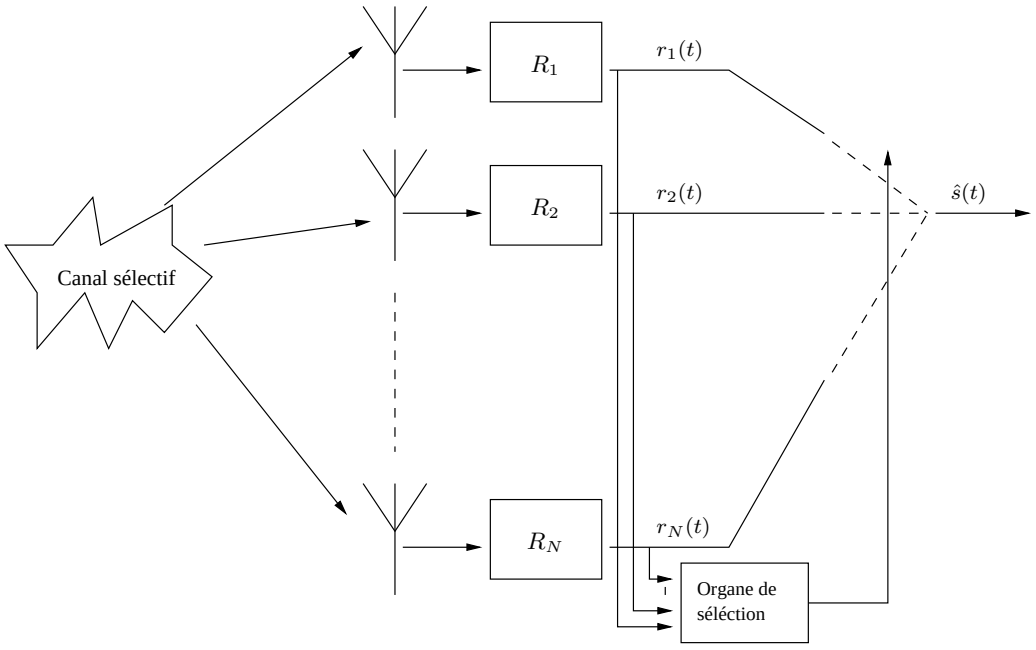


FIG. II.6 – Recombinaison par sélection

– la recombinaison MRC (*Maximum Ratio Combining*), ou recombinaison « à pondération optimale[4] » : les α_i sont choisis proportionnels au RSB de la branche qu'ils pondèrent. Une mesure pertinente du RSB n'étant pas toujours faisable, les α_i sont choisis proportionnels à la puissance totale (signal + bruit) de la branche qu'ils pondèrent. Cette technique impose de connaître les déphasages de chaque branche pour pouvoir effectuer une somme *cohérente* de signaux. Ce schéma de recombinaison est illustré figure II.7.

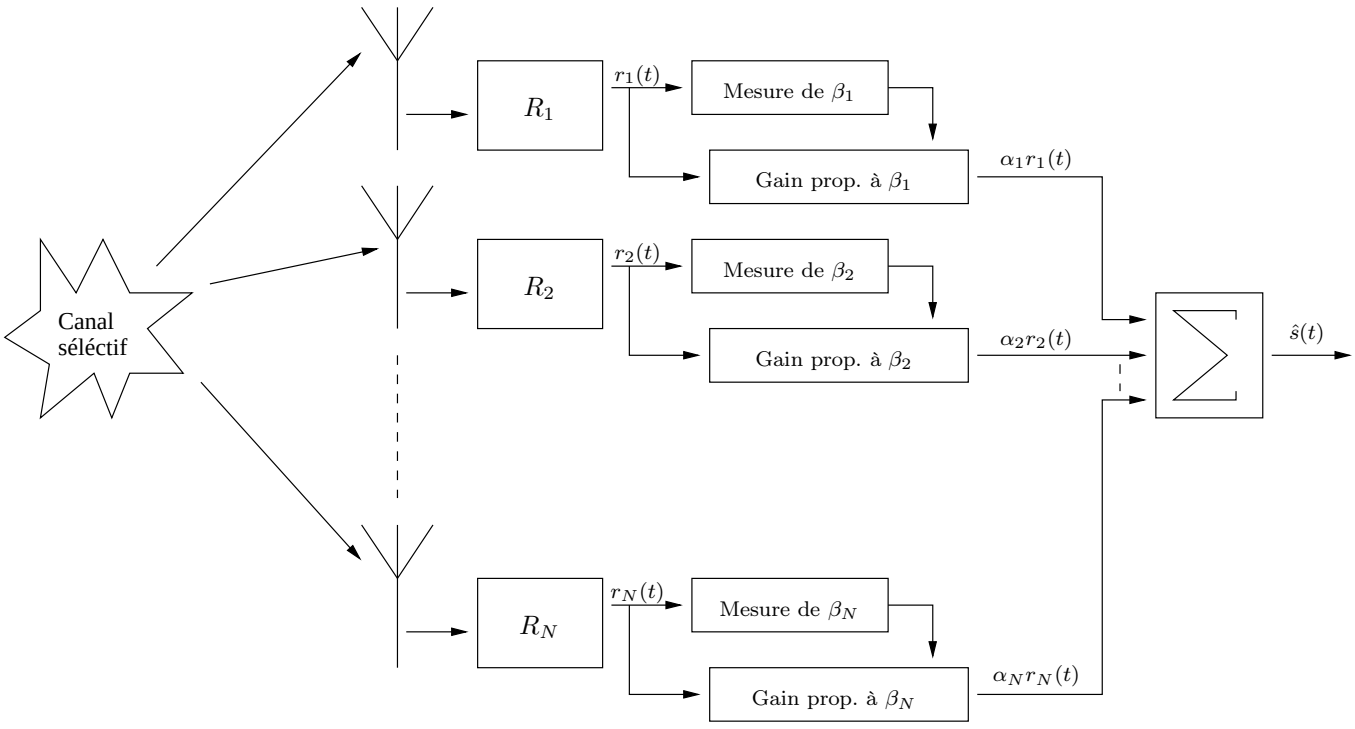


FIG. II.7 – Recombinaison « à pondération optimale » (Maximum Ratio Combining)

– la recombinaison à gains égaux (de l'anglais Equal Gain Combining, ou EGC) : les α_i sont choisis égaux à 1, et la même importance est ainsi donnée à chaque branche de diversité lors du processus de recombinaison. Cette méthode est plus simple que le MRC, mais le RSB en sortie est moindre qu'avec le MRC. De plus, cette technique est plus sensible aux erreurs de surestimation/sous-estimation du nombre de branches de diversité réellement disponibles, ce qui peut être problématique dans le cas d'un récepteur Rake, par exemple.

La méthode MRC est celle qui est utilisée au sein du récepteur Rake, car c'est elle qui offre le meilleur RSB en sortie. En effet, si on néglige les retards τ_i, on peut écrire :

$$\forall i \in [1, N], r_i = \beta_i s + n_i \tag{II.7}$$

On émet l'hypothèse que les bruits n_i sont indépendants et ont la même densité spectrale de puissance $2N_0$. De plus, les β_i sont considérés comme parfaitement connus et pour la simplicité de l'écriture, on considère que s est un signal à enveloppe constante qui vérifie $|x|^2 = 2\varepsilon$. La

[4]Traduction hasardeuse de l'anglais *Maximum Ratio Combining*.

combinaison linéaire utilisée lors du MRC s'écrit :

$$\hat{s} = \sum_{i=1}^{N} \alpha_i r_i = \sum_{i=1}^{N} \alpha_i \beta_i s + \sum_{i=1}^{N} \alpha_i n_i \tag{II.8}$$

La densité spectrale du bruit après recombinaison est donnée par :

$$2N_0 \sum_{i=1}^{N} |\alpha_i|^2 \tag{II.9}$$

tandis que la puissance instantanée du signal est donnée par :

$$2\varepsilon \left| \sum_{i=1}^{N} \alpha_i \beta_i \right|^2 \tag{II.10}$$

D'après l'inégalité de Cauchy-Schwartz,

$$\left| \sum_{i=1}^{N} a_i b_i^* \right|^2 \leq \sum_{i=1}^{N} |a_i|^2 \sum_{i=1}^{N} |b_i|^2 \tag{II.11}$$

le rapport de ces deux valeurs

$$\eta = \frac{\varepsilon}{N_0} \frac{\left| \sum_{i=1}^{N} \alpha_i \beta_i \right|^2}{\sum_{i=1}^{N} |\alpha_i|^2} \tag{II.12}$$

peut être maximisé de la manière suivante :

$$\eta \leq \frac{\varepsilon}{N_0} \frac{\sum_{i=1}^{N} |\alpha_i|^2 \sum_{i=1}^{N} \beta_i^2}{\sum_{i=1}^{N} |\alpha_i|^2} = \frac{\varepsilon}{N_0} \sum_{i=1}^{N} \beta_i^2 \tag{II.13}$$

L'égalité dans II.13 est obtenue pour l'égalité $\alpha_i = \beta_i, \forall i$ ($\alpha_i = \beta_i^*, \forall i$, pour des signaux complexes). Chaque branche est pondérée proportionnellement à son atténuation. Ainsi, les branches qui subissent les évanouissements les plus profonds contribuent dans une moindre mesure au signal reconstitué que les branches qui ne subissent presque pas d'atténuation. Cette technique fournit un RSB égal à :

$$\eta_{MRC} = \frac{\varepsilon}{N_0} \sum_{i=1}^{N} \beta_i^2 \tag{II.14}$$

Or, $\varepsilon\beta_i^2/N_0$ est aussi le RSB sur la branche i. II.14 montre donc que le RSB en sortie de la recombinaison MRC est égal à *la somme* des RSB sur chaque branche de diversité, et peut donc atteindre des valeurs importantes même si les RSB individuels ne sont pas très élevés.

La recombinaison « à pondération optimale », ou recombinaison MRC, est donc la méthode fournissant le RSB le plus élevé à la sortie du bloc assurant la recombinaison. Lorsque son implémentation se révèlera trop complexe, on pourra se tourner vers la recombinaison à gains égaux, qui ne s'embarrasse pas de la mesure des coefficients d'atténuation. C'est la technique MRC qui est utilisée au sein du récepteur Rake pour la recombinaison des différentes répliques du signal original arrivant à des instants différents.

D'autres schémas de recombinaison offrant de meilleures performances, en particulier dans le cas d'un canal à trajets multiples et d'une transmission à facteur d'étalement faible, sont proposés dans [TWS01] ou dans [OKT98], mais nécessitent des traitements mathématiques plus complexes, tels que des inversions de matrices, et se prêtent donc plus difficilement[5] à des implémentations en environnement contraint. Le récepteur Rake considéré par la suite utilisera donc la recombinaison MRC.

II.2.2 Principe du récepteur Rake

Comme mentionné en introduction de cette section consacrée au récepteur « en râteau », la fonction de ce dernier est de tirer parti de la diversité fournie par les différentes trajets ayant propagé le signal. A cette fin, le récepteur Rake a besoin de connaître, au strict minimum, leurs *instants d'arrivée*. Leur estimation est réalisée par un bloc annexe du récepteur Rake appelé le *Searcher*, qui dans sa forme la plus simple se compose d'un organe réalisant une corrélation glissante avec le code de l'utilisateur d'intérêt et d'un organe de décision décrétant ou non la présence d'un trajet à un instant donné. Ce bloc sera détaillé dans la section II.3.

En supposant que le récepteur Rake n'a connaissance que des *instants d'arrivée* des différents trajets, il doit effectuer en parallèle :

- la resynchronisation des différents trajets,
- le désétalement de ces différents trajets, pour revenir du rythme chip au rythme symbole, et ainsi faire « ressortir » le signal utile du bruit dans lequel il a été noyé,
- la mesure du coefficient d'atténuation du trajet en train d'être désétalé, nécessaire à la reconstruction du signal optimal par sommation cohérente des différents trajets, dans le cas où celui-ci n'est pas fourni au récepteur Rake par le dispositif d'estimation de canal,
- la sommation cohérente des différents trajets resynchronisés, désétalés, remis en phase, et pondérés proportionnellement à leur atténuation respective (recombinaison MRC).

II.2.2.1 Architecture fonctionnelle

Le récepteur Rake est composé d'un certain nombre de *doigts*, chargés de désétaler un trajet chacun. L'architecture fonctionnelle du récepteur Rake est illustrée figure II.8. Le Rake recombine ensuite selon la méthode MRC présentée section II.2.1.3 les sorties au rythme symbole des différents doigts. Le désétalement est effectué en multipliant le signal reçu par le conjugué du code complexe utilisé lors de l'étalement (pour faire revenir le signal désétalé à une phase « de référence »), puis en intégrant le produit du signal et du code sur la durée d'un symbole. Le signal reçu et le code sont synchronisés avec une précision supérieure à un chip, de manière à ce que le désétalement s'effectue correctement.

Pour pouvoir resynchroniser les trajets correctement, le récepteur Rake a besoin d'une estimation aussi précise que possible des instants d'arrivée de ces trajets. Cette estimation est réalisée par le bloc responsable de l'estimation de canal, appelé Searcher, et décrit section II.3. De plus, le Rake a besoin d'une estimation des coefficients complexes d'atténuation (amplitude pour la recombinaison MRC et phase pour la sommation cohérente) de chaque trajet. Cette estimation, réalisée par le Rake lui-même ou par le Searcher, suivant les implémentations, est rendue possible par la présence des bits pilotes (voir section I.2.3.2) au sein du slot.

[5] Du moins pour l'instant ...

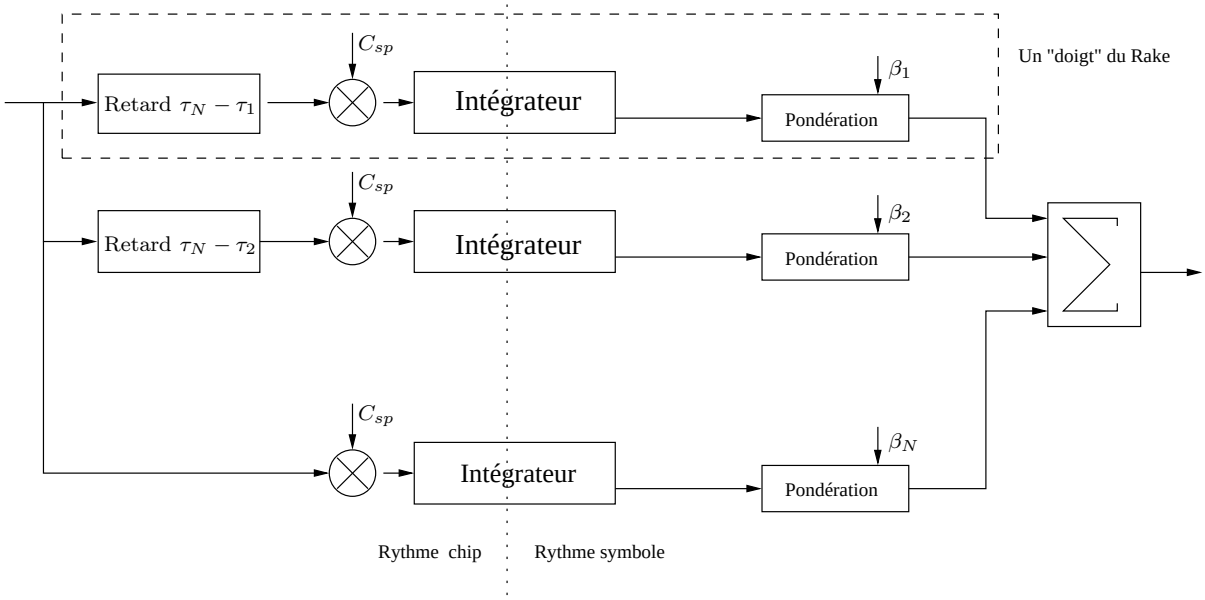


FIG. II.8 – Architecture fonctionnelle d'un récepteur Rake classique (τ_i, β_i supposés connus)

Il existe plusieurs variantes de l'architecture générale présentée figure II.8. Une première possibilité est de remplacer les blocs désétaleurs de chaque doigt, composés d'un multiplieur complexe simplifié[6] et d'un échantillonneur-bloqueur au rythme symbole, par un seul désétaleur, situé *en sortie* du bloc effectuant la recombinaison cohérente des différents trajets, comme illustré figure II.9. Cette modification a l'avantage de ne pas nécessiter une multiplication des ressources (cycles d'horloge dans le cas d'une implémentation logicielle, millimètres carrés de silicium dans le cas d'une implémentation matérielle), mais présente l'inconvénient majeur d'augmenter d'un facteur SF la cadence à laquelle doivent s'effectuer les multiplications complexes *non*-dégénérées lors du processus de recombinaison cohérente. Par contre, dans le cadre d'une transmission multi-code, cette méthode ne reconstruit le signal optimal qu'une seule fois, et le désétale ensuite autant de fois qu'il y a de canaux logiques transmis simultanément, alors que le récepteur Rake classique désétale les différents trajets et recombine les *symboles* (et non plus les *chips*) pour chaque canal logique.

Là où le récepteur Rake théorique fonctionne en temps continu et dispose donc des valeurs du signal reçu à tous les instants, une réalisation numérique de ce récepteur fonctionne en temps discret, avec un signal échantillonné à un certain rythme. Le problème qui se pose alors est celui de la précision de l'estimation de l'instant d'arrivée des trajets et de la disponibilité des échantillons correspondants. En effet, si l'échantillon correspondant à cet instant d'arrivée n'est pas disponible en mémoire, il faut le fabriquer, ou en fabriquer un autre suffisamment proche (en temps) pour que l'erreur commise soit faible. La méthode de fabrication la plus couramment utilisée dans ces circonstances est l'interpolation : les données sont échantillonnées à deux fois le ryhtme chip, pour respecter le critère de Shannon, et le récepteur Rake fabrique à l'aide d'un filtre interpolateur l'échantillon dont il a besoin pour désétaler un certain trajet. Pour ce faire,

[6] Le calcul effectué est la multiplication du signal reçu resynchronisé par le conjugué du code complexe produit du code d'étalement et du code d'embrouillage, qui a dans le cadre du WB-CDMA une valeur de la forme $\pm 1 \pm j$.

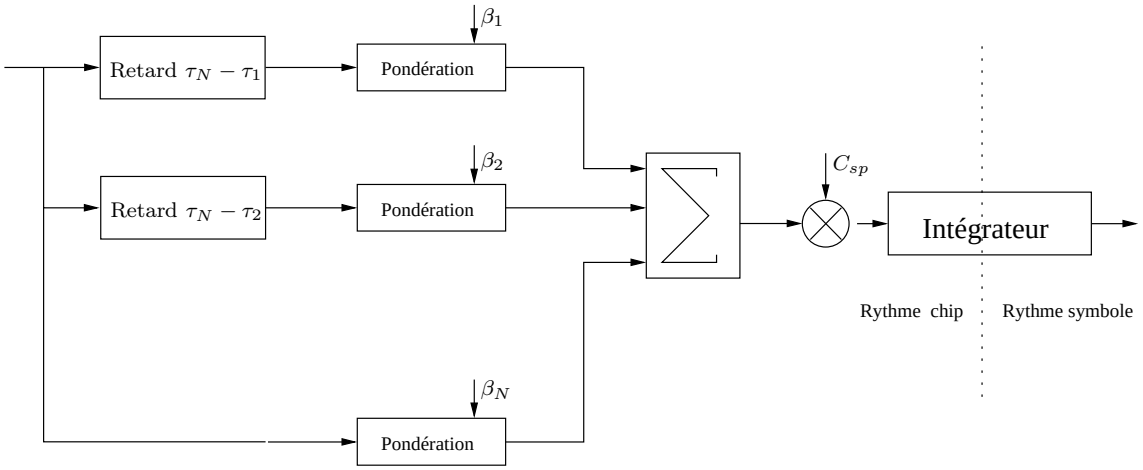


FIG. II.9 – Architecture d'un récepteur Rake à un seul désétaleur (τ_i, β_i supposés connus)

le récepteur Rake a besoin d'un interpolateur par doigt, ce qui n'est guère économe en termes de puissance de calcul : en effet, les filtres interpolateurs ne sont pas fixes, et doivent être recalculés à chaque fois qu'une nouvelle estimation de canal est disponible. A cela s'ajoute le problème de la conception d'un dispositif d'estimation de canal suffisamment sophistiqué pour estimer des instants d'arrivée avec une précision supérieure à celle des données qui lui sont présentées. Cette solution offre cependant l'avantage de ne pas multiplier la quantité de données nécessaire au bon fonctionnement du récepteur Rake.

L'autre alternative consiste à ne pas utiliser d'interpolateur et à travailler avec des signaux suréchantillonnés à un rythme multiple du rythme chip, typiquement d'un facteur 4 ou 8. Les doigts du récepteur Rake sont alors simplifiés, puisqu'ils n'ont plus besoin d'intégrer des filtres interpolateurs à coefficients variables recalculés à la volée. Par contre, ce choix impose une cadence de fonctionnement plus élevée aux blocs antérieurs au récepteur Rake, en particulier au bloc responsable de l'estimation de canal, le Searcher. La cadence de fonctionnement des doigts reste la même, car les doigts n'ont besoin que d'un seul échantillon par chip pour le désétalement. La précision de l'estimation des instants d'arrivée est ainsi de l'ordre du huitième ou du quart de chip, ce qui légitime le fait de se passer d'interpolateur, sans que la complexité du traitement numérique ne change pour autant[7]. C'est pourquoi les récepteurs Rake travaillant à un rythme multiple du rythme chip, et donc dépourvus d'interpolateurs, sont les plus utilisés aujourd'hui.

II.2.2.2 Equations du récepteur Rake

Soit N_T le nombre de trajets identifiés par le Searcher, $\{\beta_i\}_{i\in[1,N_T]}$ les atténuations complexes de ces N_T trajets, supposées parfaitement connues et $\{\tau_i\}_{i\in[1,N_T]}$ leurs retards respectifs par rapport à une base de temps entretenue dans le mobile, supposés parfaitement connus, $s(t)$ le signal émis par la station de base et $n(t)$ un bruit additif supposé blanc et gaussien. Le signal

[7]On parle ici du traitement numérique du récepteur Rake : la charge de calculs du Searcher augmente naturellement avec le volume d'information à traiter en entrée.

$r(t)$ présent à l'entrée du récepteur s'écrit donc :

$$r(t) = \sum_{i=1}^{N_T} \beta_i s\left(t - \tau_i\right) + n(t) \tag{II.15}$$

Pour la simplicité de l'écriture, on considèrera que les trajets sont classés par ordre de retard croissants, i.e. $\tau_1 < \tau_2 < \cdots < \tau_{N_T}$, et que le récepteur Rake est idéalement doté d'autant de doigts que de trajets identifiés par le Searcher. Les doigts du récepteur Rake sont d'abord calés sur les différents trajets, puis retardés chacun d'une durée différente de manière à resynchroniser les différents trajets avec le dernier arrivé, afin de ne sommer que des portions de signal se rapportant bien aux mêmes chips. Les signaux sur les doigts du récepteur Rake sont alors :

$$\begin{aligned}
\forall i \in [1, N_T], r_i(t) &= r\left(t - (\tau_{N_T} - \tau_i)\right) && \text{(II.16a)}\\
&= \sum_{j=1}^{N_T} \beta_j s\left(t - (\tau_{N_T} - \tau_i) - \tau_j\right) + n(t) && \text{(II.16b)}\\
&= \beta_i s\left(t - \tau_{N_T}\right) + \sum_{\substack{j=1\\ j\neq i}}^{N_T} \beta_j s\left(t - (\tau_{N_T} - \tau_i + \tau_j)\right) + n(t) && \text{(II.16c)}
\end{aligned}$$

Soit $C(t)$ le code complexe à valeurs dans $\pm 1 \pm j$ ayant étalé le signal à l'émission. $C(t)$ est donné par :

$$C(t) = \sum_{n=-\infty}^{+\infty} \left(C_n^I + jC_n^Q\right) u_{T_c}\left(t - nT_c\right), \text{ avec } u_T(t) = \begin{cases} 1 \text{ pour } 0 \leq t < T \\ 0 \text{ sinon} \end{cases} \tag{II.17}$$

Soit $\tau_{i,j} = \tau_{N_T} - \tau_i + \tau_j$. Les produits des signaux resynchronisés sur le dernier trajet par le conjugué du code complexe s'écrivent alors :

$$\begin{aligned}
\forall i \in [1, N_T], v_i(t) &= r_i(t) C^*\left(t - \tau_{N_T}\right)\\
&= \underbrace{\beta_i s\left(t - \tau_{N_T}\right) C^*\left(t - \tau_{N_T}\right)}_{\substack{\text{terme portant l'information : le code et le signal sont synchronisés}}}\\
&\quad + \underbrace{\left(\sum_{\substack{j=1\\ j\neq i}}^{N_T} \beta_j s\left(t - \tau_{i,j}\right)\right) C^*\left(t - \tau_{N_T}\right)}_{\text{interférence due aux trajets multiples}}\\
&\quad + \underbrace{n(t) C^*\left(t - \tau_{N_T}\right)}_{\text{terme dû au bruit}}\\
&= \beta_i s\left(t - \tau_{N_T}\right) C^*\left(t - \tau_{N_T}\right) + w_i(t)
\end{aligned} \tag{II.18}$$

$w_i(t)$ représente ici l'interférence globale due aux trajets multiples et au bruit. Les trajets y contribuant ne sont pas synchronisés avec le code, et ne « ressortent » pas du bruit lors du désétalement.

Les sorties des désétaleurs de chaque doigt sont obtenues de la façon suivante :

$$\forall i \in [1, N_T], \hat{s}_i(t) = \sum_{n=-\infty}^{+\infty} \left(\frac{1}{SF} \sum_{k=0}^{SF-1} v_i\left(nT + kT_c\right)\right) u_T\left(t - nT\right) \tag{II.19}$$

En notant $e(t)$ le signal étalé à l'émission, le signal optimal reconstitué à la sortie du récepteur Rake après recombinaison MRC est donné par :

$$\begin{aligned}
\hat{s}(t) &= \sum_{i=1}^{N_T} \beta_i^* \hat{s}_i(t) \\
&= \sum_{n=-\infty}^{+\infty} \sum_{i=1}^{N_T} \beta_i^* \left(\frac{1}{SF} \sum_{k=0}^{SF-1} \beta_i s\left(nT + kT_c - \tau_{N_T}\right) C^*\left(nT + kT_c - \tau_{N_T}\right) \right) u_T(t - nT) \\
&\quad + \sum_{n=-\infty}^{+\infty} \sum_{i=1}^{N_T} \beta_i^* \left(\frac{1}{SF} \sum_{k=0}^{SF-1} w_i\left(nT + kT_c - \tau_{N_T}\right) \right) u_T(t - nT) \\
&= \sum_{n=-\infty}^{+\infty} \left(\sum_{i=1}^{N_T} |\beta_i|^2 \right) \left(\frac{1}{SF} \sum_{k=0}^{SF-1} s\left(nT + kT_c - \tau_{N_T}\right) C^*\left(nT + kT_c - \tau_{N_T}\right) \right) u_T(t - nT) \\
&\quad + \sum_{n=-\infty}^{+\infty} \sum_{i=1}^{N_T} \beta_i^* \left(\frac{1}{SF} \sum_{k=0}^{SF-1} w_i\left(nT + kT_c - \tau_{N_T}\right) \right) u_T(t - nT) \\
&= \sum_{n=-\infty}^{+\infty} \left(\underbrace{\left(\sum_{i=1}^{N_T} |\beta_i|^2 \right) e\left(nT - \tau_{N_T}\right)}_{\text{terme portant l'information}} + \underbrace{\sum_{i=1}^{N_T} \left(\frac{1}{SF} \sum_{k=0}^{SF-1} w_i\left(nT + kT_c - \tau_{N_T}\right) \right)}_{\text{terme d'interférence due aux trajets multiples et au bruit}} \right) u_T(t - nT)
\end{aligned} \tag{II.20}$$

On peut voir dans II.20 que la forme d'onde émise est récupérée avec un retard égal au retard maximum des trajets l'ayant propagée, entachée d'une erreur provenant à la fois du désétalement non synchronisé, pour chaque trajet identifié, des trajets autres que le trajet d'intérêt, et du désétalement du bruit. C'est l'inconvénient majeur du récepteur Rake : en estimant les trajets séparément les uns des autres, l'interférence due aux trajets multiples n'est pas traitée en tant que telle mais plutôt considérée de la même façon que le bruit.

C'est là une des différences fondamentales entre le récepteur Rake et les récepteurs à *détection multi-utilisateur*, aussi appelés à *détection conjointe*. Ce type de récepteur est basé sur des algorithmes qui exploitent spécifiquement la connaissance des intercorrélations des codes présents dans le signal et cherchent à minimiser leur contribution. En assimilant les trajets secondaires à des contributions parasites d'autres utilisateurs[8], les algorithmes du type *MUD* (*Multi-User Detection*) traitent la contribution des interférences liées à ces codes en tentant de les minimiser sous différents critères tels que le critère *ZF* (*Zero Forcing*, forçage à zéro) ou le critère *MMSE* (*Minimum Mean Square Error*, erreur quadratique moyenne minimum) [Pig00] [Ros01].

Ainsi, ces récepteurs présentent des performances meilleures que le récepteur Rake mais leur utilisation est soumise à quelques contraintes :

- Dans le cas de codes d'étalement longs, la complexité algorithmique devient un sérieux obstacle à leur implémentation [Pig00] ; c'est d'ailleurs pour celà que l'utilisation de récepteurs MUD n'est envisagée pour l'instant que pour le mode TDD de l'UMTS, pour lequel la longueur des codes est au maximum égale à 16.
- Pour fonctionner correctement, ces algorithmes nécessitent la connaissance de l'ensemble

[8]Cette assimilation est légitimée par le fait qu'un trajet secondaire se présente par rapport au trajet principal de la même manière que le signal d'un autre utilisateur, c'est-à-dire comme un code venant *interférer* avec la détection du code d'intérêt.

des codes présents dans la cellule. Ainsi, s'il paraît possible de les employer dans la station de base, qui dispose nécessairement de cette connaissance, leur implémentation au sein des terminaux est pour l'instant compromise pour des raisons évidentes de confidentialité des communications. En effet, la norme ne prévoit pas explicitement la transmission de l'ensemble des codes présents dans la cellule au terminal, empêchant ainsi ce dernier d'utiliser les algorithmes sus-mentionnés.
– Les gains en performance apportés par l'emploi d'algorithmes MUD sont plus importants lorsque les algorithmes de détection conjointe sont mis en œuvre sur des systèmes exploitant la diversité spatiale ([JB95]), ce qui est rarement le cas du terminal pour l'instant.

II.2.3 Complexité algorithmique du récepteur Rake

Les différentes tâches effectuées par le récepteur Rake sont les suivantes :
– l'estimation des coefficients d'atténuation de chaque trajet, lorsque celle-ci n'a pas déjà été effectuée par le Searcher,
– la multiplication des signaux resynchronisés sur chaque doigt par le conjugué du produit du code d'embrouillage complexe par le code d'étalement réel,
– le désétalement de ces mêmes signaux, qui se fait par intégration des signaux de chaque doigt sur la durée d'un symbole,
– la pondération des symboles obtenus par le conjugué du coefficient d'atténuation estimé,
– la sommation cohérente des symboles récupérés sur chaque doigt.

La complexité algorithmique évaluée ici se chiffre en multiplications et additions de nombres réels, sans prendre en compte d'éventuelles astuces d'implémentation. Ceci dit, la simplification de certains calculs, qui découle du fait que les valeurs des codes appartiennent à un ensemble bien particulier, est prise en compte car elle permet de changer la *nature* des opérations effectuées, ce qui peut avoir un impact important sur la complexité globale de l'algorithme et encore plus sur son éventuelle implémentation.

II.2.3.1 Notations utilisées

Les variables utilisées pour expliciter le calcul de la complexité sont les suivantes :
– N_d, le nombre de doigts du récepteur Rake mis en œuvre, idéalement égal au nombre de trajets identifiés par le Searcher N_T, typiquement compris entre 3 et 6[9],
– N_{ch}, le nombre de canaux transmis simultanément par la station de base, typiquement compris entre 1 et une petite dizaine, suivant la classe du terminal,
– SF, le facteur d'étalement des canaux transmis, supposé unique pour tous les canaux (transmission monoservice mais multicode), égal à une puissance de 2 entre 4 et 512 (voir tableau I.5),
– N_p le nombre de symboles pilotes dans le slot, égal à une puissance de 2 entre 2 et 16 (voir tableau I.5) ; pour 1 symbole pilote, on a SF bits pilotes et $SF/2$ *chips* pilotes,
– $N_{data} = \frac{5120}{SF} - N_p$ le nombre de symboles de données au sens large[10].

[9]Le facteur limitant est ici le niveau de performance du Searcher, qui doit alors être capable d'identifier sans erreur ces trajets.

[10]Les données au sens large sont les données proprement dites, les commandes de contrôle de puissance (les bits TPC) et les indicateurs de format (les bits TFCI).

II.2.3.2 Phase 1 : Estimation par le Rake des coefficients de pondération

Lorsqu'elle n'est pas réalisée par le Searcher, l'estimation des coefficients β_i peut se faire grâce à la présence des chips pilotes au sein du burst. Il faut mesurer, pour chaque chip connu à la réception, le rapport complexe entre le signal reçu et le chip pilote en question. Ce calcul est simplifié par la particularité des valeurs des chips pilotes, qui appartiennent toutes à l'ensemble $\{+1, -1, +j, -j\}$.

En notant $p(n)$ les chips pilotes et $r(n)$ les chips reçus correspondant aux chips pilotes, la division par $p(n)$ devient :

$$\frac{r(n)}{p(n)} = \frac{1}{|p(n)|^2} r(n)p^*(n) = r(n)p^*(n), \text{ car } |p(n)| = 1 \tag{II.21}$$

β_i est alors estimé de la manière suivante :

$$\hat{\beta}_i = \frac{2}{N_p * SF} \sum_{n=0}^{\frac{N_p * SF}{2}-1} r(n)p^*(n) \tag{II.22}$$

$p(n)$ appartenant par construction à l'ensemble $\{1, -1, j, -j\}$, le produit $r(n)p^*(n)$ ne peut prendre que les valeurs suivantes :

$$r(n)p^*(n) = \begin{cases} \mathcal{R}\{r(n)\} + j\mathcal{I}\{r(n)\} & \text{si } p(n) = 1 \\ -\mathcal{R}\{r(n)\} - j\mathcal{I}\{r(n)\} & \text{si } p(n) = -1 \\ \mathcal{I}\{r(n)\} - j\mathcal{R}\{r(n)\} & \text{si } p(n) = j \\ -\mathcal{I}\{r(n)\} + j\mathcal{R}\{r(n)\} & \text{si } p(n) = -j \end{cases} \tag{II.23}$$

La multiplication complexe du calcul d'un produit $r(n)p^*(n)$ est donc remplacée par un éventuel échange des parties réelle et imaginaire de $r(n)$ et par deux additions/soustractions réelles, selon la valeur de $p(n)$. La division par $\frac{N_p * SF}{2}$ est remplacée par un décalage à droite du résultat de la somme, le nombre de chips traités étant toujours égal à une puissance de 2. La charge de calcul entraînée par l'estimation des coefficients d'atténuation est donc de :

$$N_d \times (N_p \times SF + 1) \text{ additions réelles par slot.} \tag{II.24}$$

On peut également effectuer cette estimation à partir des *symboles* pilotes, une fois ceux-ci désétalés.

II.2.3.3 Phase 2 : Multiplication des signaux de chaque doigt par les codes conjugués

Cette opération préalable au désétalement proprement dit est semblable dans son principe à la précédente. En effet, pour chaque canal de données présent dans le signal reçu, il faut multiplier chaque chip considéré par le conjugué du code complexe ayant servi à l'étalement, avant d'intégrer le résultat obtenu sur la durée d'un symbole.

Le système de codage utilisé étant un système hiérarchique à deux couches, on peut soit désétaler N_{ch} fois avec des codes complexes, soit désembrouiller une seule fois avec le code d'embrouillage complexe, puis désétaler séparément les voies I et Q avec les codes d'étalement réels de chaque canal de données.

On peut de nouveau remplacer avantageusement les multiplications complexes apparentes par des additions/soustractions réelles. En effet, pour chaque chip de données, il faut calculer $r(n)C^*(n)$, où $r(n)$ est le chip reçu et $C(n)$ le code complexe, produit du code d'étalement par le code d'embrouillage. La valeur de $C(n)$ pouvant par construction s'écrire $\pm 1 \pm j$, le produit $r(n)C^*(n)$ ne peut prendre que les valeurs suivantes :

$$r(n)C^*(n) = \begin{cases} (\mathcal{R}\{r(n)\} + \mathcal{I}\{r(n)\}) & + & j\,(\mathcal{I}\{r(n)\} - \mathcal{R}\{r(n)\}) & \text{si } C(n) = 1 + j \\ (\mathcal{R}\{r(n)\} - \mathcal{I}\{r(n)\}) & + & j\,(\mathcal{I}\{r(n)\} + \mathcal{R}\{r(n)\}) & \text{si } C(n) = 1 - j \\ (\mathcal{I}\{r(n)\} - \mathcal{R}\{r(n)\}) & - & j\,(\mathcal{I}\{r(n)\} + \mathcal{R}\{r(n)\}) & \text{si } C(n) = -1 + j \\ -(\mathcal{R}\{r(n)\} + \mathcal{I}\{r(n)\}) & + & j\,(\mathcal{R}\{r(n)\} - \mathcal{I}\{r(n)\}) & \text{si } C(n) = -1 - j \end{cases} \quad \text{(II.25)}$$

Le calcul d'un produit $r(n)C^*(n)$ nécessite donc 2 additions/soustractions réelles ainsi qu'une sélection des parties à additionner/soustraire. La charge de calcul entraînée par la multiplication des signaux sur chaque doigt par les conjugués des codes ayant servi à l'étalement est donc de :

$$N_d \times N_{ch} \left(\left(2560 - \frac{N_p \times SF}{2} \right) \times 2 \right) = N_d \times N_{ch} \times N_{data} \times SF \text{ additions réelles par slot.} \quad \text{(II.26)}$$

II.2.3.4 Phase 3 : Désétalement des signaux de chaque doigt

Pour revenir au rythme symbole après avoir multiplié de façon synchrone les signaux au rythme chip par le conjugué du code complexe ayant servi à l'étalement, il faut intégrer ces signaux sur la durée d'un symbole, qui dépend du facteur d'étalement utilisé. Chaque chip comportant deux bits d'information, on passe de 2560 chips par slot à $\frac{2\times 2560}{SF}$ symboles, et donc à $\frac{2\times 2560}{SF} - N_p = N_{data}$ symboles de données, pour un seul canal de données. On obtient en fait $\frac{2560}{SF} - \frac{N_p}{2} = \frac{N_{data}}{2}$ nombres complexes contenant chacun une quantité d'information équivalente à 2 symboles réels, qui sont récupérés séparément lors d'une phase ultérieure.

Le désétalement d'un symbole nécessitant SF additions réelles, la charge de calcul entraînée par le désétalement de tous les canaux de données sur tous les doigts est de :

$$N_d \times N_{ch} \times \left(\frac{5120}{SF} - N_p \right) \times SF = N_d \times N_{ch} \times N_{data} \times SF \text{ additions réelles par slot.} \quad \text{(II.27)}$$

II.2.3.5 Phase 4 : Pondération des symboles obtenus avant sommation cohérente

Une fois les différentes copies du signal original désétalées, il faut les pondérer par le conjugué de l'atténuation complexe estimée lors de la première phase, de manière à réaliser une sommation cohérente et éviter ainsi de perdre de l'information lors de la recombinaison.

Pour chaque canal de données et chaque doigt, il faut multiplier les $\frac{2560}{SF} - \frac{N_p}{2} = \frac{N_{data}}{2}$ nombres complexes désétalés précédemment par les atténuations conjuguées, sans qu'il soit possible ici de remplacer les multiplications complexes par des calculs moins coûteux. On note l'avantage de travailler ici au rythme symbole, et non plus au rythme chip.

La charge de calcul entraînée par la pondération des symboles désétalés est de :

$$N_d \times N_{ch} \times \left(\frac{2560}{SF} - \frac{N_p}{2} \right) = N_d \times N_{ch} \times \frac{N_{data}}{2} \text{ multiplications complexes par slot.} \quad \text{(II.28)}$$

II.2.3.6 Phase 5 : Sommation cohérente des doigts pour chaque symbole

Pour fournir une estimation des symboles émis au dispositif de détection proprement dit, il faut sommer de manière cohérente les symboles de chaque doigt. L'étape précédente ayant assuré la cohérence de la somme en alignant les symboles des doigts sur une phase de référence, il ne reste plus qu'à calculer la somme des doigts pour chaque symbole à estimer.

La charge de calcul entraînée par la sommation cohérente est de :

$$N_{ch} \times \left(\frac{2560}{SF} - \frac{N_p}{2}\right) \times (2 * N_d) = N_d \times N_{ch} \times N_{data} \text{ additions réelles par slot.} \quad \text{(II.29)}$$

II.2.3.7 Bilan de complexité

La charge de calcul globale d'une itération du récepteur Rake sur un slot, une fois les traitements d'identification effectués par le Searcher, sans prendre en compte le processus de détection[11], dépend du nombre de doigts mis en œuvre N_d, du nombre de canaux à démoduler simultanément N_{ch}, du nombre de symboles pilotes N_p, du facteur d'étalement utilisé SF, et du nombre de symboles de données au sens large $N_{data} = \frac{5120}{SF} - N_p$.

La charge de calcul globale pour un slot est de :

$$\begin{array}{c} N_d \times (N_p \times SF + 1) + N_d \times N_{ch} \times N_{data} \times (2 \times SF + 1) \text{ additions réelles,} \\ \text{et } N_d \times N_{ch} \times \frac{N_{data}}{2} \text{ multiplications complexes.} \end{array} \quad \text{(II.30)}$$

Pour évaluer quantitativement cette complexité algorithmique dans plusieurs cas de figure, il est confortable de se ramener à une seule unité, l'opération réelle dans le cas présent. L'introduction d'une unité générique et la conversion des multiplications complexes dans cette unité soulèvent le problème de l'implémentation. En effet, il existe plusieurs façons d'effectuer une multiplication complexe, certaines plus adaptées à une implémentation logicielle sur architecture programmable, et d'autres plus adaptées à une architecture matérielle ([BwT98], [SSB98]). S'affranchir de l'implémentation proprement dite et la dissocier de l'évaluation de la complexité algorithmique impose de se placer à un niveau d'abstraction relativement élevé et de considérer que toutes les opérations réelles sont équivalentes, sans distinguer la nature des opérations véritablement effectuées. Ainsi, on considèrera pour le moment qu'une *addition* réelle est équivalente à une *multiplication* réelle, même si ce n'est pas forcément le cas pour l'architecture ciblée par l'implémentation finale de l'algorithme.

Bien que le produit complexe $(a + ib)(c + id)$ puisse se calculer de différentes manières, on ne considèrera ici que la plus directe, $(a + ib)(c + id) = (ac - bd) + i(ad + bc)$, qui nécessite 6 opérations réelles (4 multiplications et 2 additions).

Cette conversion effectuée, la charge de calcul globale pour un slot est de :

$$N_d \times (N_p \times SF + 1) + 2 \times N_d \times N_{ch} \times N_{data} \times (SF + 2) \text{ opérations réelles.} \quad \text{(II.31)}$$

Il est intéressant de chiffrer cette charge de calcul pour les différents couples $\{N_p, SF\}$ possibles, pour un seul canal de données ($N_{ch} = 1$) et un seul doigt du récepteur Rake ($N_d = 1$). Les valeurs de ces couples sont données en fonction du format de burst dans le tableau I.5. Les

N_p	SF	N_{data}	Débit	Comp. (op. r.)	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
4	512	6	15 kbps	8217	24,94 %	37,39 %	37,39 %	0,22 %	0,07 %
2	256	18	30 kbps	9801	5,23 %	47,02 %	47,02 %	0,55 %	0,18 %
4	256	16	30 kbps	9281	11,04 %	44,13 %	44,13 %	0,52 %	0,17 %
8	256	12	30 kbps	8241	24,86 %	37,28 %	37,28 %	0,44 %	0,15 %
4	128	36	60 kbps	9873	5,20 %	46,67 %	46,67 %	1,09 %	0,36 %
8	128	32	60 kbps	9345	10,97 %	43,83 %	43,83 %	1,03 %	0,34 %
8	64	72	120 kbps	10017	5,12 %	46 %	46 %	2,16 %	0,72 %
8	32	152	240 kbps	10593	2,43 %	45,92 %	45,92 %	4,30 %	1,43 %
16	16	304	480 kbps	11201	2,29 %	43,42 %	43,42 %	8,14 %	2,71 %
16	8	624	960 kbps	12609	1,02 %	39,59 %	39,59 %	14,85 %	4,95 %
16	4	1264	1920 kbps	15233	0,43 %	33,19 %	33,19 %	24,89 %	8,30 %

TAB. II.1 – Complexité du récepteur Rake, en opérations réelles, pour 1 doigt et 1 canal, en fonction du facteur d'étalement et du nombre de symboles pilotes présents dans le burst

charges de calcul obtenues, exprimées en opérations réelles, ainsi que les contributions des différentes phases à la complexité totale de l'algorithme sont présentées dans le tableau II.1.

On note la prédominance de l'apport en termes de complexité des phases 2 et 3, qui montre bien l'importance des traitements au rythme chip (multiplication des signaux des doigts par les conjugués des codes complexes et désétalement de ces mêmes signaux) par rapport aux traitements au rythme symbole (pondération et sommation des doigts).

II.3 Estimation de canal et complexité

Pour reconstituer de manière optimale le signal utile à partir du signal reçu, composé de plusieurs copies retardées, déphasées et atténuées du signal émis, le récepteur Rake a besoin d'une estimation aussi précise que possible de trois types de paramètres [Tur80] :

- le nombre $\hat{N}$ de copies du signal émis composant le signal reçu,
- les retards respectifs $\{\hat{\tau}_i\}_{i\in[\![1,\hat{N}]\!]}$ de ces trajets secondaires par rapport à une base de temps,
- les atténuations complexes $\left\{\hat{\beta}_i = \hat{a}_i e^{j\hat{\theta}_i}\right\}_{i\in[\![1,\hat{N}]\!]}$ de chacun de ces trajets.

Ce problème d'estimation est différent d'une identification complète de la réponse impulsionnelle du canal radiomobile. En effet, à la différence d'un égaliseur classique, le récepteur Rake n'essaie pas de compenser les effets du canal radiomobile dans leur totalité, mais plutôt d'exploiter les propriétés des signaux à spectre étalé pour récupérer autant d'énergie utile que possible.

[11] La détection proprement dite est la décision finale de la valeur du symbole transmis à partir de la sortie du récepteur Rake. Le détecteur le plus simple, dans le cas de séquences binaires, est le détecteur à seuil, qui se contente de renvoyer le *signe* des symboles fournis par le Rake.

II.3.1 Algorithmes d'estimation de canal

Le Searcher est le bloc responsable de l'estimation des paramètres nécessaires à l'utilisation efficace du récepteur Rake. Pour faciliter cette estimation, le signal émis comporte une partie connue du récepteur : les *symboles pilotes* déjà mentionnés plus haut. Ces symboles particuliers se retrouvent dans les systèmes de deuxième génération (GSM, IS-95, D-AMPS ...) et dans la plupart des systèmes de transmission radiomobile, sous forme de midambule la plupart du temps.

Leur présence généralisée au sein des systèmes radiocellulaires se justifie par la sévérité des conditions de propagation rencontrées : la réception y est en effet suffisamment difficile pour ne pas avoir en plus à mettre en œuvre des algorithmes aveugles. Leur utilisation pour l'estimation de canal se traduit donc souvent par l'emploi d'algorithmes dits *entraînés*, qui exploitent spécifiquement la connaissance de cette partie du signal émis afin de faciliter la démodulation de la partie inconnue du message.

Les erreurs qui peuvent être commises lors de l'estimation des paramètres ci-dessus n'ont pas toutes la même importance. Pris indépendamment les uns des autres et classés par ordre d'importance croissante, on obtient la liste suivante :

1. Le module est le moins important des paramètres. En effet, lors de la recombinaison, l'amplitude sert à pondérer les différentes branches de diversité les unes par rapport aux autres. Accorder à une branche un peu plus d'importance que ce qu'elle ne mérite est relativement peu dommageable à la qualité globale de la démodulation. Le RSB obtenu n'est pas le RSB optimal atteignable par le récepteur, mais aucune *perte* d'information à proprement parler ne peut découler d'une erreur commise sur l'amplitude.

2. Le nombre de trajets discernables et utiles présents à l'entrée du récepteur peut, s'il est sous-estimé ou surestimé, dégrader dans de plus grandes proportions la qualité de la démodulation effectuée par le récepteur Rake. Le sous-estimer revient à ne pas exploiter *toute* l'information présente à l'entrée du récepteur en laissant de côté une ou plusieurs copies secondaires du signal émis. En se privant ainsi d'une quantité d'information utile, le récepteur n'atteint pas son RSB optimal. Le surestimer revient par contre à introduire une quantité de bruit supplémentaire au sein du processus de recombinaison. Des symboles issus du désétalement d'une portion non synchronisée du signal reçu sont pondérés et pris en compte lors de la reconstruction des symboles émis. Là aussi, le RSB en sortie du récepteur Rake est dégradé par rapport à sa valeur optimale.

3. La phase est une donnée plus importante encore que l'amplitude ou le nombre de trajets. En effet, lors de la recombinaison, c'est la phase estimée qui détermine l'exactitude de l'alignement des symboles désétalés avec la constellation de référence avant la sommation cohérente des branches de diversité. Si la phase est fausse, la sommation n'est plus parfaitement cohérente, et une partie de l'information utile est perdue. Le non-alignement d'une branche de diversité avec la constellation de référence signifie qu'une partie de l'énergie véhiculée par la voie I vient bruiter la voie Q, et vice-versa. Il y a donc une perte explicite d'information.

4. Enfin, les retards des différents trajets identifiés relativement à une base de temps maintenue au sein du récepteur sont les paramètres les plus importants intervenant dans la recombinaison. Les propriétés d'auto- et d'intercorrélation des codes employés lors des

phases d'étalement et d'embrouillage imposent en effet une erreur de synchronisation inférieure à un chip. Dans le cas où l'erreur commise est *supérieure* à un chip, les symboles obtenus en sortie des désétaleurs ne sont porteurs d'aucune information, et viennent bruiter purement et simplement le processus de recombinaison. La dégradation causée par une erreur d'estimation du retard d'un trajet est proportionnelle à son amplitude estimée. En effet, si le trajet a été identifié comme relativement peu puissant par rapport aux autres, la pondération des symboles en provenant mitigera leur influence sur le résultat final de la recombinaison. Par contre, si le trajet est considéré comme un des plus puissants, les symboles qui en sont extraits, intrinsèquement faux puisqu'obtenus à l'aide d'un désétalement non synchronisé, vont introduire une fois pondérés une erreur relativement importante sur les symboles issus de la recombinaison. Il y a là aussi une perte explicite d'information. L'impact d'une erreur d'estimation des retards est présenté en détail dans [PSO96] et [HVNP96].

La littérature sur le sujet étant pour le moins fournie, il peut être utile de regrouper les algorithmes proposés en un certain nombre de classes, selon les caractéristiques qu'ils partagent.

II.3.1.1 Pourquoi l'estimation de canal ?

On trouve dans la littérature un nombre élevé d'articles traitant de l'égalisation du canal en environnement radiomobile. Ceci dit, peu d'entre eux rentrent dans les détails de l'estimation de canal proprement dite. La plupart du temps, celle-ci est supposée déjà effectuée d'une part, et parfaite d'autre part. Une des différences fondamentales entre les algorithmes d'estimation de canal et les algorithmes traitant conjointement l'égalisation et l'estimation est le calcul *explicite* des caractéristiques du canal, indispensable au bon fonctionnement du récepteur Rake.

Les algorithmes d'égalisation tentent en général de compenser les effets néfastes du canal radiomobile dans sa totalité, ce qui n'est pas l'objectif ici. L'objectif ici est d'obtenir non pas une estimation aussi précise que possible de la réponse impulsionnelle du canal radiomobile à un instant donné, mais d'en extraire les quantités pertinentes dans le cadre de la mise en œuvre d'un récepteur Rake, à savoir le nombre de trajets, leurs délais de propagation et leurs atténuations complexes. D'une certaine manière, le Searcher est responsable de la fourniture au récepteur Rake d'une version *épurée* (un *modèle*) de la réponse impulsionnelle du canal.

Le calcul explicite des caractéristiques du canal étant en général absent des algorithmes dits *aveugles*, où une certaine quantité en relation avec les statistiques du canal radiomobile par lequel a transité le signal émis est entretenue et réactualisée au cours du temps, ces derniers ne seront pas considérés par la suite. De plus, l'architecture hypothétique envisagée pour le bloc de réception n'utilisant qu'un récepteur Rake de la plus simple espèce, on ne considèrera que les algorithmes d'estimation de canal ne nécessitant pas d'aller-retour des données entre le récepteur Rake et le Searcher. Ainsi, les algorithmes « guidés par les données[12] », tels [APL98], ou *semi-aveugles* [KS98] ne sont pas considérés comme des solutions adéquates au problème posé.

Enfin, on ne s'intéressera pas aux algorithmes de *poursuite* [SBR99a], chargés de réactualiser de slot en slot les caractéristiques d'un ensemble de trajets déjà identifiés. La mise en œuvre de

[12]Ces algorithmes sont aussi appelés *decision-directed*, car les décisions en sortie du récepteur Rake sont réinjectées dans l'estimateur de canal pour parfaire au fil des itérations l'estimation de canal transmise au récepteur Rake.

tels algorithmes pouvant s'effectuer indépendamment du dispositif d'estimation, ils constituent une classe d'algorithmes distincte de celle des algorithmes d'estimation pure et ne seront donc pas développés.

II.3.1.2 Vitesse de variation des caractéristiques du canal

Le modèle de canal considéré, ou plutôt les simplifications qui y ont été apportées, conditionne considérablement le choix de la structure du récepteur et le degré de réactivité des organes qui le composent.

Par exemple, si l'on considère que le temps de cohérence du canal est suffisamment faible pour permettre à N, τ_i, a_i et θ_i de varier significativement pendant la durée d'un slot (666 μs pour le Wideband CDMA), ou même d'un symbole, il est illusoire d'espérer atteindre des performances correctes en n'utilisant qu'un récepteur Rake classique, qui réactualise ses paramètres (le nombre de trajets, les délais de propagation et les atténuations complexes) au rythme auquel se font les estimations successives du canal, et donc au rythme d'apparition des symboles pilotes dans la transmission, c'est-à-dire une fois par slot. Il faut alors augmenter le degré de complexité du récepteur et permettre au récepteur Rake de voir ses paramètres varier en cours de slot. De plus, il faut rendre variable le rythme auquel sont effectuées les estimations de canal, malgré le fait que les symboles pilotes ne soient émis qu'une fois par slot, et implémenter au sein du bloc de réception un dispositif d'estimation suffisamment rapide et/ou peu complexe pour pouvoir être activé plusieurs fois par slot.

On considère habituellement que pendant la durée de l'estimation (et pendant sa durée de validité), les retards ne varient pas, tandis que les amplitudes et les phases peuvent varier lentement (à un rythme moins élevé que le rythme symbole ou le rythme chip, selon les hypothèses émises). Suivant la situation de fading envisagée, les amplitudes, alors considérées comme des variables aléatoires, peuvent suivre une distribution de Rayleigh ou de Rice, par exemple.

II.3.1.3 Station de base ou terminal ?

Les caractéristiques du problème de l'estimation des paramètres du canal changent selon l'extrémité de la liaison auprès de laquelle on se place. Dans le cadre d'un système de radiotéléphonie cellulaire, l'estimation de canal effectuée par la station de base doit en particulier tenir compte d'un certain nombre de facteurs que le terminal peut s'abstenir de considérer.

Le *near-far effect*, par exemple, néfaste pour la qualité du lien radio CDMA dès lors qu'un émetteur est perçu comme sensiblement plus puissant que les autres, ne concerne que la station de base, qui reçoit simultanément les signaux émis de manière non synchronisée et avec des puissances différentes par tous les terminaux présents dans la cellule. Le problème est alors un problème d'estimation de canal *multi-utilisateur*. En effet, pour améliorer la qualité de la démodulation simultanée des signaux de tous les utilisateurs, la station de base a intérêt à estimer *en même temps* les canaux suivis par les signaux de tous les utilisateurs avant de commencer à estimer les données émises par chacun d'entre eux : c'est le principe de la détection conjointe, déjà évoquée II.2.2.2. Le terminal, quant à lui, n'a pas ce problème à résoudre. En effet, la majeure partie du temps, il ne voit qu'un seul émetteur : la station de base de sa cellule, qui émet normalement suffisamment fort pour ne pas être bruitée par une autre émission à la même fréquence. Le même problème, vu du terminal, est devenu un problème d'estimation *mono-utilisateur*. Cependant, en bordure de cellule, le terminal peut recevoir avec

des puissances sensiblement équivalentes les émissions de sa cellule d'origine et des cellules voisines, qui se bruitent alors mutuellement. Un processus de *soft handover* [3GP01e] est alors mis en œuvre. Pendant un court laps de temps, les deux (ou plus) cellules incriminées vont émettre les *mêmes informations* à destination du mobile, pour lui permettre d'améliorer la réception des données envoyées par le réseau. Dès que le terminal a quitté la bordure de la cellule et qu'il reçoit les émissions de sa nouvelle cellule suffisamment fort, le *soft handover* est interrompu. Le *soft handover* pose à lui tout seul un certain nombre de problèmes théoriques, dont l'étude dépasse de loin le cadre de cette thèse.

Sur un plan plus pratique, les contraintes s'exerçant sur les algorithmes implantés dans les stations de base, en particulier les contraintes de complexité et de connaissance a priori de l'environnement radiomobile, sont nettement plus lâches que celles s'exerçant sur les terminaux. La station de base, n'ayant pas à se soucier de problèmes d'autonomie ou de durée de vie d'éventuelles batteries, dispose d'une puissance de calcul virtuellement illimitée pour résoudre les problèmes algorithmiques qui lui sont posés. De plus, la station de base étant directement connectée au réseau sous-jacent, elle dispose d'une connaissance largement plus étendue que celle du terminal : la géographie de la cellule lui est connue, ainsi que le nombre d'utilisateurs présents simultanément ou encore les codes utilisés par chacun d'entre eux. Ce sont autant d'informations dont la connaissance peut être mise à profit par la station de base pour résoudre les problèmes algorithmiques posés. Le terminal, ne disposant pas de ces informations, doit la plupart du temps recourir à des solutions algorithmiques différentes de celles employées par la station de base.

Un procédé couramment utilisé pour améliorer la qualité de l'estimation de canal est l'accumulation et/ou le moyennage des données sur un certain intervalle de temps, préalablement à l'estimation effective des paramètres du canal [SBR99b]. Dans [ASA98], par exemple, l'estimée de la réponse impulsionnelle du canal est construite comme une combinaison linéaire non triviale des réponses impulsionnelles calculées par corrélation au slot présent ainsi qu'aux slots précédents. Les coefficients du filtre réalisant la combinaison linéaire ne sont pas exprimés formellement, mais plutôt déterminés par simulation, et dépendent fortement de la situation radiomobile (fading lent ou rapide, utilisation du contrôle de puissance...), ce qui rend délicate l'utilisation en situation réelle de l'algorithme présenté. Une telle accumulation/pondération a pour principal effet la diminution des erreurs d'estimation, au prix d'une augmentation considérable de la quantité de mémoire nécessaire au fonctionnement du système. Cette technique introduit de plus un délai supplémentaire dans la communication, ce qui n'est pas forcément tolérable[13].

Par ailleurs, les algorithmes destinés à un récepteur exploitant la diversité spatiale [LLM98] sont par leur conception plus adaptés aux stations de base qu'aux terminaux, du moins pour l'instant [Pig00]. Les terminaux d'aujourd'hui n'ont en effet qu'une seule antenne, et ce pour principalement deux raisons :

– le volume toujours plus réduit des terminaux mobiles d'aujourd'hui, qui rend très difficile l'intégration d'une seconde antenne au mobile. En effet, pour pouvoir exploiter la diversité spatiale offerte par la présence de plusieurs antennes sur le récepteur, il faudrait que les

[13]Pour un système TDD, où l'utilisateur a presque toute la *trame* pour démoduler les données émises à son intention par la station de base, l'accumulation des données sur plusieurs trames a vite fait d'introduire un délai considérable dans la communication, alors que pour un système FDD, il est possible, au prix d'une augmentation de la quantité de mémoire embarquée, d'accumuler les données sur plusieurs *slots* sans trop ralentir les échanges.

antennes soient distantes d'au moins une demi-longueur d'onde les unes des autres, ce qui équivaut à peu près à 7 cm pour l'UMTS. L'intégration d'une autre antenne ne peut donc se faire sans une augmentation substantielle de la taille des terminaux, ce qui va à l'encontre de l'usage strictement *téléphonique* qui est fait aujourd'hui de la majorité des terminaux. Des terminaux plus sophistiqués, dont l'usage ne serait pas *uniquement* téléphonique, mais incluerait l'utilisation de fonctions du domaine des PDA[14], seraient plus à même de voir leur taille (déjà supérieure à celle d'un téléphone mobile) augmenter dans des proportions raisonnables, pour peu que cette augmentation de taille soit accompagnée d'une augmentation similaire des débits offerts ou de la qualité des communications [Ros01].

– le coût de plus en plus faible des terminaux actuels, qui augmenterait considérablement si d'autres antennes étaient présentes sur le terminal. En effet, deux antennes nécessitent deux chaînes RF de traitement, qui ne peuvent être intégrés au même prix que des fonctions supplémentaires sur une architecture programmée, par exemple. De plus, le traitement analogique représentant une part importante de la consommation électrique globale d'un terminal, l'autonomie en souffrirait considérablement.

Ceci dit, le gain en performances apporté par l'exploitation de la diversité spatiale n'est en général significatif qu'à partir d'un nombre d'antennes typiquement supérieur à 3, ce qui n'est pas envisageable avant au moins la prochaine génération de terminaux..

Le problème posé ici concernant l'estimation de canal *au sein du terminal*, le problème d'estimation sera réduit à sa dimension mono-utilisateur et mono-capteur, on s'affranchira de l'étude de la sensibilité des algorithmes présentés au *near-far effect* et on n'utilisera pas de techniques "retardantes", telles l'accumulation/pondération des estimations déjà effectuées, malgré le gain potentiel en performances.

II.3.1.4 Estimation ou synchronisation ?

Il ne faut pas confondre le problème posé, qui consiste à *estimer* les paramètres du canal, avec un problème de *synchronisation*.

Acquérir une synchronisation revient en simplifiant à caler le récepteur sur le trajet le plus puissant, en minimisant à tout prix l'erreur commise sur l'instant d'arrivée du trajet en question. L'estimation va au-delà : le terminal doit non seulement estimer aussi précisément que possible l'instant d'arrivée du trajet le plus puissant, celui-ci étant en général porteur de la plus grande quantité d'information, mais aussi estimer avec la meilleure précision possible les instants d'arrivée des divers trajets secondaires présents à l'entrée du récepteur, sans surestimer ou sous-estimer leur nombre. On trouve dans la littérature un certain nombre d'algorithmes d'estimation volontairement restreints à des situations mono-trajets [BA98], qui tiennent donc plus de la *synchronisation* que de l'*estimation* de canal proprement dite.

Une solution à ce problème est l'emploi d'algorithmes dits *itératifs*, qui estiment *successivement* les caractéristiques de *tous* les trajets discernables à l'entrée du récepteur.

La mise en place d'un tel estimateur n'est pas sans poser un certain nombre de problèmes : comme dans tous les systèmes itératifs, le critère d'arrêt des itérations doit être judicieusement choisi. D'autre part, au fil des itérations, il peut s'avérer utile d'utiliser les éléments déjà iden-

[14] *Personal Digital Assistant* : assistant numérique personnel, appareil mobile réunissant les fonctions d'agenda, de répertoire, de bloc-notes ... Les appareils Palm ou Psion, par exemple, appartiennent à la catégorie des PDA.

tifiés lors des itérations précédentes pour maximiser la probabilité d'une estimation correcte pour l'itération courante.

II.3.1.5 Bilan

Les algorithmes qui nous intéressent rassemblent les caractéristiques suivantes :
- ils estiment le nombre de trajets discernables à l'entrée du récepteur, les retards et/ou les atténuations complexes de *chacun* de ces trajets, et réalisent ainsi une véritable *estimation* du canal, ce qui représente bien plus qu'une simple *synchronisation* du récepteur sur un trajet donné,
- l'estimation de canal effectuée est une estimation entraînée : la connaissance des symboles *pilotes* présents dans le burst est exploitée par le récepteur, ainsi que celle des codes d'étalement et d'embrouillage utilisés à l'émission, tandis qu'aucune estimation de *données* n'est effectuée,
- ils ne tentent pas d'estimer plusieurs réponses impulsionnelles de canaux différents comme le ferait une station de base, mais plutôt de se concentrer sur une estimation mono-utilisateur,
- les caractéristiques des principaux trajets discernables présents à l'entrée du récepteur sont ne sont estimées qu'une seule fois par *activation* de l'algorithme, et ne sont pas remises à jour par un quelconque dispositif de *poursuite* pendant la démodulation des données proprement dites[15],
- aucune technique de moyennage/pondération des estimations précédemment effectuées n'est utilisée. Seule la portion de signal correspondant à la partie *pilotes* du slot courant est traitée par l'algorithme.

On peut classer les nombreux algorithmes rencontrés dans la littérature remplissant ces conditions en trois grandes familles :
- les algorithmes à maximum de vraisemblance (algorithmes *ML*, pour *maximum likelihood*), et à maximum de vraisemblance approximé [ZLMS97] [SPMO96b], qui testent d'une manière exhaustives toutes les valeurs possibles des paramètres caractéristiques du problème d'estimation afin de minimiser une fonction de coût optimale ou une approximation de cette fonction de coût. Ces algorithmes, en plus d'effectuer une recherche exhaustive des valeurs optimales de certains paramètres, ont en général recours à des calculs matriciels très lourds, comme des inversions de matrice ou des calculs de valeurs propres et des vecteurs propres associés, et ne sont donc pas adaptés à une implémentation en environnement contraint.
- les algorithmes à projection, ou algorithme des sous-espaces [BA96] [Ris99] [SPMO96a], basés sur l'adaptation à une situation monocapteur de l'algorithme MUSIC (Multiple Signal Classification) [Sch81], qui estiment les délais de propagation en décomposant l'espace des observations en un sous-espace engendré par le signal d'intérêt, communément appelé *sous-espace signal*, et un sous-espace engendré par le bruit, orthogonal au premier, et en maximisant ensuite la projection du signal reçu sur le sous-espace signal. Ces algorithmes comme les précédents souffrent de l'utilisation de calculs matriciels lourds, et ne ciblent que rarement le terminal comme organe de calcul.

[15] La mise en œuvre d'un tel dispositif peut être réalisée indépendamment du dispositif d'estimation de canal employé.

– les algorithmes à corrélation, qui n'exploitent que la corrélation du signal reçu avec le signal pilote régénéré au sein du récepteur [BSRC98] [KM95] [FG99] [FSA98], avec ou sans utilisation de techniques d'accumulation des données ou des estimations effectuées [ASA98] [SBR99b]. Moins sophistiqués et souvent moins performants, ces algorithmes sont moins complexes que les algorithmes à projection ou à maximum de vraisemblance, et sont les seuls aujourd'hui à pouvoir être raisonnablement implantés dans un terminal. C'est donc vers cette classe d'algorithme que va s'orienter la recherche d'un algorithme d'estimation de canal à faible complexité.

II.3.2 Principe de l'estimation par corrélation

L'utilisation d'un algorithme estimant les caractéristiques du canal radiomobile à partir de la corrélation entre le signal reçu et un signal pilote régénéré à la réception répond à un impératif de minimisation de la complexité embarquée. On ne cherchera pas à atteindre les meilleures performances possibles, mais plutôt à réduire la complexité opératoire tout en conservant des performances correctes, afin de viser une implémentation en environnement contraint capable de fonctionner en temps réel.

L'inconvénient majeur des algorithmes basés sur le calcul de l'intercorrélation entre le signal reçu et le signal pilote régénéré à la réception est leur pouvoir de résolution limité [GJP+91]. Du fait du filtre de mise en forme utilisé, la largeur d'un pic de corrélation traduisant la présence d'un trajet au retard considéré est égale à la durée d'un chip, ce qui empêche le simple calcul de l'intercorrélation de distinguer deux trajets séparés en temps d'une durée inférieure à celle d'un chip, même en cas de travail avec une fréquence d'échantillonnage multiple de la fréquence chip. Il existe des algorithmes capables de distinguer des trajets éloignés de moins d'un chip [KP99] [Luo00], mais de l'aveu des auteurs de [KP99], ces algorithmes sont hautement irréalistes à l'heure actuelle, et ce pour trois raisons :

– ils requièrent une fréquence d'échantillonnage très élevée, typiquement de l'ordre de 25 fois la fréquence chip,
– les manipulations de matrices effectuées sont non triviales,
– la méthode proposée est itérative, ce qui augmente d'autant la complexité calculatoire de l'ensemble.

Cette limitation acceptée, le Searcher a deux tâches à réaliser : le calcul de l'intercorrélation entre le signal reçu et le signal pilote régénéré sur une fenêtre de longueur donnée, et le *dépouillement* de cette corrélation, c'est-à-dire l'extraction proprement dite des caractéristiques du canal qui doivent être transmises au récepteur Rake, comme indiqué figure II.10.

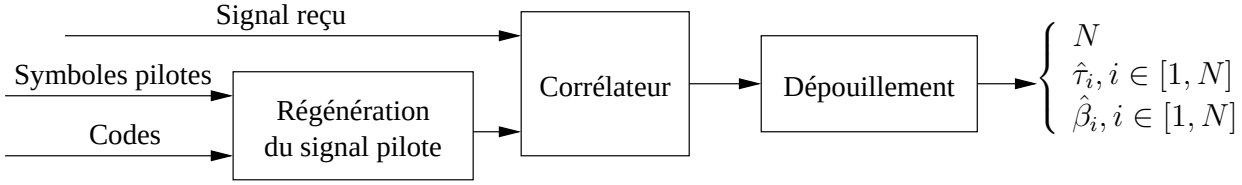


FIG. II.10 – Schéma-bloc d'un Searcher estimant le canal par corrélation

L'estimation du canal commence par l'estimation du retard et de l'atténuation complexe du trajet principal. En notant $r(t)$ le signal reçu, $s_p(t)$ le signal pilote régénéré à la réception, τ_1 le

retard du trajet principal[16], β_1 son atténuation complexe et $n(t)$ le terme global d'interférence due au bruit et aux trajets secondaires, le modèle adopté est :

$$r(t) = \beta_1 s_p(t - \tau_1) + n(t), \tag{II.32}$$

Les estimateurs au sens du maximum de vraisemblance de τ_1 et β_1 sont obtenus à l'aide de la fonction d'intercorrélation $\gamma(\tau)$ entre le signal reçu et le signal pilote régénéré, qui peut s'écrire :

$$\forall \tau \in [0, \tau_{max}], \gamma(\tau) = \frac{1}{T_p} \int_{T_p} r(t) s_p^*(t - \tau) dt, \tag{II.33}$$

en notant T_p la durée du support su signal pilote régénéré.

Les estimateurs $\hat{\tau_1}$ et $\hat{\beta}_1$ sont alors donnés [VT71] par :

$$\hat{\tau_1} = \arg\max_{\tau} |\gamma(\tau)|^2 \tag{II.34a}$$

$$\hat{\beta}_1 = \frac{\gamma(\hat{\tau}_1)}{\frac{1}{T_p} \int_{T_p} |s_p(t)|^2 dt} \tag{II.34b}$$

Ces estimateurs sont *non corrélés*, et leurs variances vérifient :

$$Var\left[\hat{\beta}_1\right] \geq \frac{1}{SNR} \tag{II.35a}$$

$$Var\left[\hat{\tau_1}\right] \geq \frac{1}{SNR} \frac{1}{B_{eff}^2} \tag{II.35b}$$

$$B_{eff}^2 = \frac{4\pi^2 \int \nu^2 S^2(\nu) d\nu}{\int |S(\nu)|^2 d\nu}, \tag{II.35c}$$

en notant SNR le rapport signal-à-bruit et B_{eff} la *bande efficace* du signal pilote. De plus, l'estimateur $\hat{\beta}_1$ de β_1 est biaisé à cause du terme en $n(t)s_p^*(t - \tau)$, qu'on ne peut pas estimer. Ce point sera repris par la suite.

Cette opération de « dépouillement » du profil obtenu par corrélation doit être effectuée pour l'identification de chacun des trajets, ce qui est réalisé grâce à la suppression successive des trajets estimés ; ce point est présenté dans le paragraphe II.3.3. Dans le cas d'une estimation volontairement limitée à un faible nombre de trajets, des solutions non itératives peuvent exister ([PJ86] pour un canal à deux trajets proches, par exemple).

Le dépouillement efficace du profil du canal obtenu par le calcul du module carré de l'intercorrélation entre le signal reçu et le signal pilote régénéré est rendu plus délicat par le fait de travailler à un rythme supérieur au rythme chip. En effet, comme indiqué figure II.11, l'extraction analytique des pics de corrélation observables à l'œil nu est loin d'être évidente. Dans le cas le plus favorable, où les pics sont suffisamment distants les uns des autres, et où le niveau de bruit est relativement faible, un algorithme de « partage des eaux » [NS96, section 2] [Gom01, section 4.2.2], par exemple, peut suffire à les identifier. Dans la majorité des cas, cependant, il y a une réelle difficulté algorithmique à surmonter : comment estimer le plus précisément possible

[16] On ne considère pas que le trajet principal est le trajet qui arrive au récepteur avec le retard le plus faible, comme l'indice $_1$ utilisé dans la notation (τ_1, β_1) peut le faire croire. L'indice est utilisé pour figurer l'ordre dans lequel sont *détectés* les trajets, et non pas le classement de leurs retards respectifs.

les caractéristiques des trajets les plus faibles et comment identifier les véritables trajets des pics de corrélation dus à la non-orthogonalité des répliques secondaires du signal transmis entre elles ?

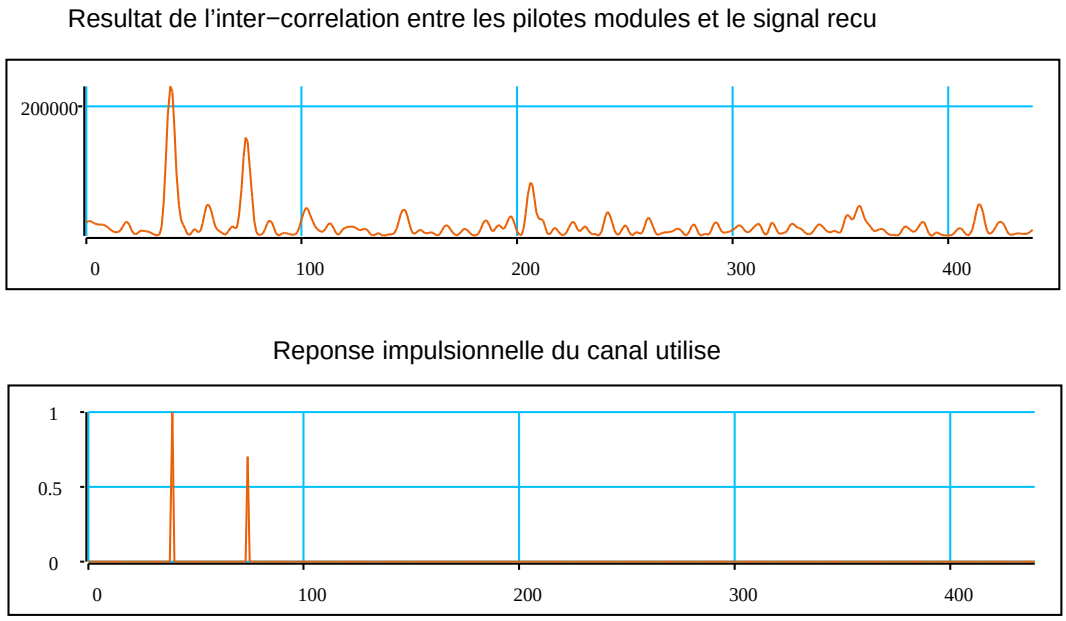


FIG. II.11 – Profil de canal obtenu par intercorrélation et trajets simulés à l'origine du profil

Il est intéressant d'observer la courbe d'autocorrélation du signal pilote, ou plus précisément de la portion du signal reçu correspondant à l'émission des symboles pilotes, pour différentes longueurs de cette séquence pilote, avant et après la mise en forme. Cette autocorrélation dépend des codes d'étalement et d'embrouillage utilisés, mais l'allure générale des courbes est la même. Pour obtenir les figures ci-dessous, les codes utilisés ont été choisis au hasard, et les autocorrélations des séquences pilotes employées sur les différents slots ont été moyennées. L'autocorrélation de la séquence de 32 chips pilotes étalés et embrouillés est représentée figure II.12 *avant* et *après* la mise en forme. Le facteur de suréchantillonnage considéré est égal à 8. L'étroitesse du pic de corrélation est conservée une fois le signal mis en forme, sans toutefois empêcher l'apparition de faibles lobes secondaires. Ainsi, si la superposition des séquences pilotes retardées et déphasées par le canal radiomobile n'empêche pas l'identification des trajets, c'est la superposition des *séquences de données* aux séquences pilotes de faible longueur qui va véritablement gêner le Searcher, comme on peut le voir sur la figure II.15.

Concernant le nombre de trajets à prendre en compte, la solution la plus simple consiste à seuiller le module de la corrélation obtenu et de ne conserver comme trajets valides que les pics de corrélation dépassant le seuil. Cette technique a l'inconvénient majeur de nécessiter le *réglage* du seuil en question, ce qui peut s'avérer relativement délicat de par la variété des environnements radiomobiles dans lesquels le mobile peut évoluer. Fukumoto *et al.* proposent dans [FSA98] une technique plus perfectionnée employant non pas *un* mais *deux* seuils. Les trajets conservés sont ceux dont le module carré est supérieur au maximum S de deux seuils S_{min} et S_{max} calculés comme suit :

$$S_{min} = \min_{\tau} \left\{ |\gamma(\tau)|^2 \right\}_{\tau \in [0, \tau_{max}]} \times 10^{\frac{\Delta_m}{10}} \quad \text{(II.36a)}$$

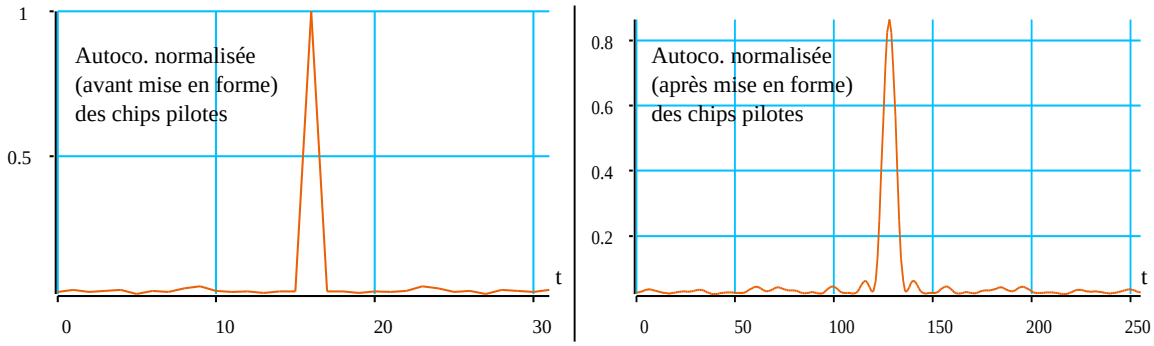


FIG. II.12 – Autocorrélation normalisée de la séquence pilote *avant* et *après* la mise en forme ($N_{cp} = 32$, $OSF = 8$)

$$S_{max} = \max_{\tau} \left\{ |\gamma(\tau)|^2 \right\}_{\tau \in [0, \tau_{max}]} \times 10^{-\frac{\Delta_M}{10}} \quad \text{(II.36b)}$$

$$S = \max \{ S_{min}, S_{max} \} \quad \text{(II.36c)}$$

Les deux paramètres à régler ici sont Δ_m et Δ_M. Δ_M sert à éliminer les trajets présentant un RSB trop faible : un trajet est considéré comme valide uniquement si le module carré de son atténuation complexe est supérieur à Δ_M dB *de moins* que le module carré de l'atténuation complexe du trajet le plus puissant. Δ_m, quant à lui, sert à éliminer des trajets validés par le premier seuil les trajets constitués majoritairement ou uniquement de bruit : ce deuxième seuil va éliminer les trajets dont le module carré de l'atténuation complexe n'est pas *au moins égal* à Δ_m dB *de plus* que le plancher de bruit, matérialisé ici par la valeur minimum prise par le module carré du profil du canal obtenu par corrélation. Autrement dit, Δ_M représente la tolérance de l'algorithme vis-à-vis du trajet le plus puissant, tandis que Δ_m représente sa tolérance vis-à-vis du plancher de bruit (ou plutôt d'une estimation du plancher de bruit). Cette technique intéressante renferme cependant l'inconvénient classique des algorithmes reposant uniquement sur des seuils, à savoir les seuils eux-mêmes et la nécessité de les régler. Fukumoto *et al.* ont obtenu un couple (Δ_m, Δ_M) optimal (3 dB, 5 dB) validé pour différents profils de canaux, mais même si le problème est résolu pour ces cas là, il reste entier pour des situations plus générales. Une solution légèrement plus complexe mais plus efficace est l'emploi de méthodes *itératives* estimant les caractéristiques des trajets les uns après les autres.

II.3.3 Estimation itérative de canal à suppression de trajets

Une solution au problème de l'estimation des caractéristiques des trajets autres que le trajet principal est l'utilisation de méthodes itératives estimant les caractéristiques des trajets les uns après les autres, de manière à tirer parti de la connaissance des caractéristiques des trajets déjà estimés lors de la recherche d'un nouveau trajet.

II.3.3.1 Exemple : estimation de canal itérative à maximum de vraisemblance approximé

Un exemple de version *itérative* d'algorithme au départ *non-itératif* est donné par l'algorithme sous-optimal à maximum de vraisemblance approximé présenté dans [BSRC98] et

fonctionnant de la manière suivante :

1. La position du premier trajet dans la fenêtre de recherche est estimée par minimisation exhaustive d'une fonction de coût donnée (approche ML), et son atténuation complexe est estimée.
2. Un deuxième trajet est recherché de manière exhaustive dans la fenêtre traitée *en considérant la position du premier trajet comme fixée*, en recalculant la métrique utilisée pour chaque combinaison de délais. Les atténuations complexes des deux trajets sont réévaluées pour chaque combinaison de délais testée.
3. Les délais des deux premiers trajets sont ensuite fixés, et la position du troisième est éstimée en ne réévaluant pour chaque combinaison que les atténuations complexes des trois trajets alors en cours d'évaluation.
4. Le procédé est renouvelé jusqu'à l'arrêt des itérations, quel que soit le critère d'arrêt choisi.

Ce procédé permet ici de réduire la complexité totale de la recherche des délais dans la mesure où à chaque itération, le nombre de degrés de liberté est égal à 1. En effet, les positions des trajets déjà identifiés étant fixées, il n'y a qu'un seul nouveau délai à estimer. Par contre, les coefficients de *tous* les trajets (ceux déjà identifiés et celui en cours d'estimation) sont réévalués tout au long de la recherche du nouveau trajet, ce qui rend peu réaliste l'implémentation d'un tel algorithme.

II.3.3.2 Estimation à suppression de trajets

Une autre manière d'exploiter la connaissance des caractéristiques des trajets $1, \cdots, n-1$ pour faciliter la recherche et l'estimation du trajet n est la *suppression* de ces trajets au fur et à mesure. Une variante de cette technique est connue dans le domaine du radar-sonar sous le nom de méthode « *clean* », car elle consiste à « nettoyer » le profil obtenu par corrélation de manière à faciliter les estimations successives.

En effet, on peut voir sur la figure II.11 que l'obstacle majeur à l'identification immédiate du *second* trajet est le pic de corrélation dû à la présence du *premier* trajet. De plus, il est raisonnable d'émettre l'hypothèse que l'estimation de l'atténuation complexe du second trajet sera bruitée du fait de la présence du premier trajet dans le signal servant de base à l'estimation.

Une solution possible est de détecter les trajets à l'aide d'un critère simple, typiquement par la recherche du pic de corrélation de plus grande amplitude, puis de supprimer du vecteur de corrélation obtenu le pic dû au trajet identifié. Un moyen simple d'arriver à ce résultat est de pondérer le signal pilote complexe servant au calcul de l'intercorrélation par l'atténuation complexe du trajet précédemment estimée, puis de soustraire ce signal pilote pondéré au signal servant de base au calcul de l'intercorrélation. De cette façon, les trajets sont soustraits au fur et à mesure de l'estimation des différentes composantes du signal reçu. Il convient de noter que les trajets sont identifiés par ordre de puissance décroissante, et non pas par ordre d'arrivée au récepteur, ce qui permet à l'algorithme de fonctionner quelle que soit la position du trajet principal par rapport aux trajets secondaires. Cette approche est mentionnée dans [FG99] dans le cas d'un canal *indoor* et d'un canal pilote transmis avec une puissance supérieure à celle des canaux destinés aux utilisateurs par la station de base. Le déroulement de l'algorithme est représenté figure II.13.

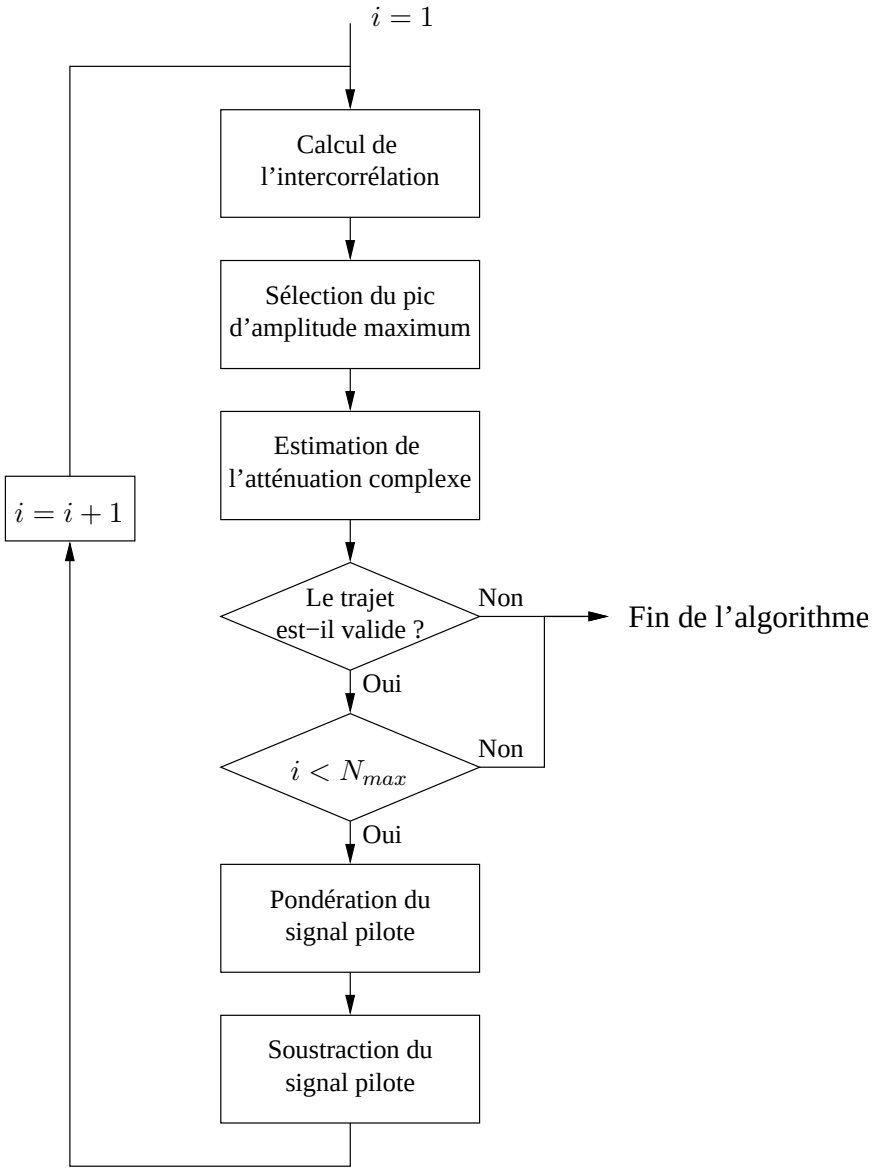


FIG. II.13 – Déroulement de l'algorithme itératif d'estimation de canal à suppression de trajet

En notant $s_p(t)$ le signal pilote obtenu en étalant puis en embrouillant les symboles pilotes émis et en filtrant deux fois (une fois pour l'émission et une fois pour la réception) le résultat obtenu, T_p la durée de son support et $r_i(t)$ les *versions* successives du signal reçu au fur et à mesure des itérations effectuées, les étapes de l'algorithme sont les suivantes :

1. L'intercorrélation pour l'itération courante est calculée comme suit :

$$\gamma_i(\tau) = \frac{1}{T_p} \int_{T_p} r_i(t-\tau) s_p^*(t) dt \tag{II.37}$$

2. Le trajet d'amplitude maximum est sélectionné :

$$\hat{\tau}_i = \arg\max_{\tau} |\gamma_i(\tau)|^2 \tag{II.38}$$

3. L'atténuation complexe du trajet identifié est estimée préalablement à la pondération du signal pilote :

$$\hat{\beta}_i = \frac{\gamma_i(\hat{\tau}_i)}{\frac{1}{T_p} \int_{T_p} |s_p(t)|^2 \, dt} \tag{II.39}$$

   Il existe une autre technique d'estimation de $\hat{\beta}_i$, nettement plus complexe, qui consiste à calculer le rapport complexe moyen du signal reçu sur le signal pilote, comme suit :

$$\hat{\beta}_i = \frac{1}{T_p} \int_{T_p} \frac{r_i(t-\hat{\tau}_i)}{s_p(t)} \tag{II.40}$$

4. La validité du trajet identifié ainsi que le nombre de trajets déjà identifiés sont testés. En cas d'échec à l'un ou l'autre test, l'algorithme est interrompu.
5. En cas de succès aux deux tests, le signal pilote $s_p(t)$ est pondéré et soustrait au signal reçu avant de réitérer à l'étape 1.

$$\begin{cases} r_{i+1}(t) = r_i(t) - \hat{\beta}_i s_p(t-\hat{\tau}_i) \\ i = i+1 \end{cases} \tag{II.41}$$

La figure II.14 montre la variation de la courbe du module carré de l'intercorrélation $\gamma_i(\tau)$ en fonction des itérations et illustre le bénéfice qu'on peut tirer de la suppression des trajets déjà identifiés à chaque itération pour l'estimation des paramètres du trajet courant. Les courbes représentées ont été obtenues en simulant un canal à 5 trajets distants les uns des autres de plus de la période chip, sans addition de bruit. La seule interférence est donc ici l'interférence d'accès multiple, ou *MAI* (*Multiple Access Interference*).

Les principaux désavantages qu'on peut trouver à un tel algorithme d'estimation de canal sont :

- le caractère non-borné par défaut de son temps d'exécution : ceci est dû au caractère *itératif* de l'algorithme,
- la potentielle propagation des erreurs commises tôt dans les itérations : une fois soustrait, un trajet aux caractéristiques mal estimées va détériorer considérablement l'estimation des caractéristiques du trajet suivant.

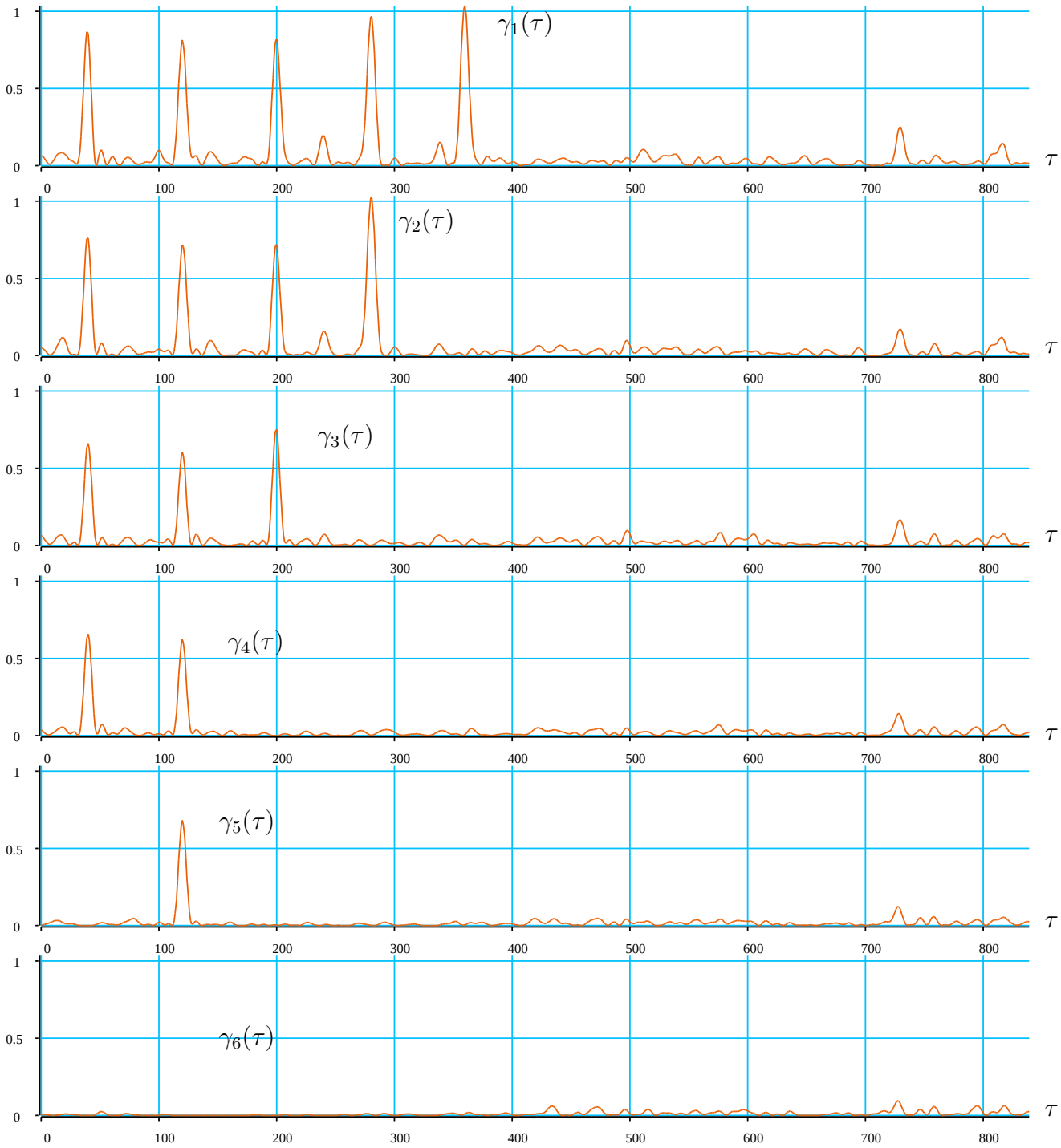


FIG. II.14 – Variation de l'intercorrélation calculée en fonction des suppressions de trajets effectuées à chaque itération, pour 5 trajets simulés et 128 chips pilotes

II.3.3.3 Eléments de complexité

Les paramètres qui conditionnent la complexité de l'algorithme décrit ci-dessus sont les suivants :

- le facteur de suréchantillonnage OSF (abbréviation de l'anglais *OverSampling Factor*), égal au rapport du rythme auquel travaille le Searcher sur le rythme chip (3.84 Mcps). Typiquement égal à 2, 4 ou 8, ce facteur de suréchantillonnage peut être atteint à l'aide de filtres interpolateurs en tête de récepteur ou bien directement à l'aide des convertisseurs analogiques-numériques.
- le nombre de *chips* pilotes N_{cp}, égal au demi-produit du facteur d'étalement SF en vigueur pendant la transmission par le nombre de *symboles* pilotes N_{sp} présents dans le burst. N_{cp} est égal à une puissance de 2 comprise entre 32 et 1024 (les différents cas sont énumérés tableau I.5).
- le *delay spread* DS (la durée de la fenêtre temporelle au sein de laquelle les trajets sont recherchés), considéré fixe pour l'instant. Il est tentant de prendre une valeur élevée du delay spread pour ne rater aucun trajet, mais un tel choix n'est pas adapté à un algorithme itératif. En effet, de cette valeur va directement découler le nombre de points de l'intercorrélation à calculer, et donc la complexité globale de l'algorithme. Une valeur de quelques dizaines de chips, typiquement 50 ou 60 chips (ce qui équivaut à environ 15 μs), est suffisante dans la plupart des cas. Il est bien sûr possible de rencontrer une situation où les trajets d'intérêt seront retardés de plus de 15 μs, mais ce devrait être suffisamment rare pour ne pas trop influer sur les performances du récepteur. Le *delay spread* des canaux de test proposés par la norme [ETS98] est en effet presque toujours inférieur à cette durée.

A des fins de comparaison, il est intéressant d'évaluer la complexité de l'algorithme présenté. Celle-ci dépendant évidemment du nombre d'itérations effectuées, on se limitera à *une seule itération typique (pondération et suppression incluses)*. La charge de calcul entraînée par les différentes phases *d'une seule itération* de l'algorithme est donc évaluée comme suit :

1. Le calcul de l'intercorrélation se fait un point à la fois, sachant que l'intercorrélation complète comprend $DS \times OSF$ points. Le calcul d'un point de l'intercorrélation nécessite $N_{cp} \times OSF$ multiplications complexes, ce qui amène la complexité de cette phase à

$$\begin{array}{c} N_{cp} \times DS \times OSF^2 \text{ multiplications complexes, soit} \\ 6 \times N_{cp} \times DS \times OSF^2 \text{ opérations réelles.} \end{array} \tag{II.42}$$

   La conversion des multiplications complexes en opérations réelles étant explicitée au paragraphe II.2.3.7.

2. Le calcul du module carré des valeurs prises par l'intercorrélation en fonction du retard avec lequel elle est calculée nécessite 3 opérations réelles pour chacun des $DS \times OSF$ points ($|a+ib|^2 = (a+ib)(a-ib) = a^2+b^2$). La charge de calcul entraînée par cette phase est donc de

$$3 \times DS \times OSF \text{ opérations réelles.} \tag{II.43}$$

3. La sélection du retard entraînant le module maximum se fait avec $DS \times OSF$ comparaisons, ce qui donne une charge de calcul égale à :

$$DS \times OSF \text{ opérations réelles.} \tag{II.44}$$

4. L'estimation de l'atténuation complexe du trajet en cours d'identification nécessite deux divisions réelles. La norme du signal pilote $s_p(t)$ n'a besoin d'être calculée qu'une fois par slot, et ne rentre donc pas en compte ici. A l'inverse des additions et des multiplications, la division ne peut être ramenée abstraitement à une seule opération réelle. Les algorithmes de division, qu'ils soient plus adaptés à une implémentation logicielle ou matérielle, sont la plupart du temps itératifs, augmentant la précision du résultat obtenu à chaque itération. Pour garder la cohérence de l'unité choisie, à savoir l'opération réelle, un coefficient pondérateur doit donc être appliqué lors de la conversion des divisions réelles en multiplications réelles. Ce coefficient est fixé arbitrairement[17] à 10. Cette phase contribue donc à la complexité globale de l'algorithme à hauteur de :

$$20 \text{ opérations réelles.} \tag{II.45}$$

5. Le test de validité du trajet identifié peut revêtir plusieurs formes et être plus ou moins sophistiqué, c'est pourquoi son cas est traité plus loin. On considèrera pour l'instant la complexité de cette phase comme nulle.
6. La pondération du signal pilote régénéré par l'estimée de l'atténuation complexe du trajet en cours d'identification, lorsqu'elle doit être effectuée, nécessite une multiplication complexe par point, et engendre au total :

$$\begin{array}{c} N_{cp} \times OSF \text{ multiplications complexes, soit} \\ 6 \times N_{cp} \times OSF \text{ opérations réelles.} \end{array} \tag{II.46}$$

7. De même, en cas de non-arrêt de l'algorithme, la soustraction du signal pilote pondéré à la portion de signal servant de base au calcul de l'intercorrélation requiert une soustraction complexe, soit deux opérations réelles, par point, et nécessite donc :

$$2 \times N_{cp} \times OSF \text{ opérations réelles.} \tag{II.47}$$

Une itération typique de l'algorithme nécessite donc :

$$6 \times N_{cp} \times DS \times OSF^2 + 4 \times (2N_{cp} + DS) \times OSF + 20 \text{ opérations réelles.} \tag{II.48}$$

La valeur numérique de cette complexité, qui varie avec *le carré du facteur de suréchantillonnage (OSF^2)* est calculée pour les différentes valeurs des paramètres la conditionnant ($N_{cp} = \{32, 64, 128, 256, 512, 1024\}$, $OSF = \{2, 4, 8\}$, $DS = 60$). Les résultats sont présentés tableau II.2. Le chiffre entre parenthèses est la contribution du terme en OSF^2, uniquement dû au calcul de l'intercorrélation. Ces chiffres sont donnés pour *une seule itération.*

Il apparaît clairement que cet algorithme est beaucoup trop gourmand pour pouvoir supporter toutes les situations : pour N_{cp}=1024, par exemple, travailler à OSF=2 engendre un million et demi d'opérations réelles *par itération*, ce qui rend l'utilisation de cet algorithme inenvisageable dans sa version actuelle. Il faut donc envisager un raffinement de cet algorithme pour éviter cette variation en OSF^2. D'autre part, l'étape de validation du trajet en cours d'estimation soulève un problème qui peut grandement affecter le comportement de l'algorithme décrit, à savoir le choix d'un critère d'arrêt des itérations. En effet, tout algorithme itératif peut être exécuté autant de fois qu'on le souhaite, mais il est intéressant de se pencher sur la pertinence de certaines stratégies de gestion de ce critère d'arrêt.

[17] Ce coefficient n'est là que pour la justesse du raisonnement, mais sa valeur n'a que peu d'importance étant donné la fréquence des divisions en question.

	$OSF = 2$	$OSF = 4$	$OSF = 8$
N_{cp}=32	47092 (97,9 %)	186324 (98,9 %)	741268 (99,5 %)
N_{cp}=64	93684 (98,4 %)	371668 (99,2 %)	1480596 (99,6 %)
N_{cp}=128	186868 (98,6 %)	742356 (99,3 %)	2959252 (99,7 %)
N_{cp}=256	373236 (98,8 %)	1483732 (99,4 %)	5916564 (99,7 %)
N_{cp}=512	745972 (98,8 %)	2966484 (99,4 %)	11831188 (99,7 %)
N_{cp}=1024	1491444 (98,9 %)	5931988 (99,4 %)	23660436 (99,7 %)

TAB. II.2 – Complexité en opérations réelles d'une itération de l'algorithme d'estimation de canal par corrélation et suppression de trajets pour différentes valeurs de N_{cp} et OSF, et contribution du calcul de l'intercorrélation

II.3.3.4 Quel(s) critère(s) utiliser pour l'arrêt des itérations ?

Différents critères peuvent être envisagés pour déterminer si oui ou non il est intéressant de continuer à itérer un algorithme donné. Ces critères ne s'excluent pas mutuellement, et on peut choisir d'en associer plusieurs, ou d'effectuer une combinaison linéaire de certains d'entre eux afin d'accorder différents poids aux décisions issues de différents critères. De telles combinaisons de critères ne seront pas examinées ici et ne sont mentionnées qu'à titre de suggestion. Une simple association de plusieurs d'entre eux sera retenue par la suite. Les critères cités ci-après sont classés par ordre de sophistication croissante[18].

La méthode la plus simple est d'itérer l'algorithme d'estimation un nombre fixe de fois, quels que soient les résultats obtenus, les conditions de propagation identifiées, les capacités de traitement de l'organe de calcul responsable de l'exécution de l'algorithme ou encore le temps disponible pour l'accomplissement de cete tâche. Cette méthode a l'avantage d'une extrême simplicité, mais est largement sous-optimale. En effet, dans le cas où le nombre d'itérations effectuées est supérieur au nombre de trajets réellement présents à l'entrée du récepteur, de *faux* trajets sont fournis au récepteur Rake afin qu'il les incorpore au sein de la recombinaison, ce qui a pour conséquence la dégradation du RSB en sortie du récepteur Rake. Dans le cas inverse, des trajets potentiellement significatifs ne sont pas pris en compte lors de la recombinaison, causant ainsi une dégradation certaine des performances. La seule application de cette méthode est la *limitation* du nombre d'itérations effectuées dans le but de respecter des contraintes de temps d'exécution. Un autre critère doit alors être associé à cette limite du nombre d'itérations, pour arrêter l'algorithme en cas de besoin.

Un critère plus fin utilisable conjointement au précédent est le seuil sur le module de la corrélation. Les seuls trajets retenus sont ceux dont l'atténuation complexe a un module qui dépasse une certaine limite fixée à x dB au-dessous du module de l'atténuation complexe du trajet le plus puissant (le premier trajet détecté). Plusieurs possibilités sont offertes quant au réglage du seuil en question : le seuil peut être fixé indépendamment de tout autre paramètre, ou en relation avec le type d'environnement radiomobile lorsque celui-ci est connu, par exemple. Ainsi, le profil radiomobile d'un environnement *indoor* étant riche en trajets proches les uns des autres et de puissances équivalentes, le seuil sur l'amplitude peut être réglé de manière assez stricte à une valeur assez faible sans entraîner de perte d'information importante, un nombre suffisant

[18]La liste donnée ici ne se veut en aucune manière exhaustive, le nombre de critères correspondant au problème présent n'étant limité que par l'imagination des concepteurs.

de trajets significatifs étant relativement puissants par rapport au trajet principal. Cependant, lorsque le nombre de chips pilotes est faible, la longueur des séquences intervenant dans le calcul de l'intercorrélation *sur un seul slot* (en l'absence de technique d'accumulation/pondération) est telle que les pics de corrélation dûs à des vrais trajets et les pics de corrélation uniquement dûs au bruit peuvent être difficiles à distinguer, gênant ainsi l'identification des trajets secondaires, comme indiqué figure II.15. De plus, ce critère pose le problème du réglage du seuil en question, et de la définition d'une stratégie d'ajustement de la valeur de ce seuil en fonction des caractéristiques de l'environnement radiomobile dans lequel le terminal évolue.

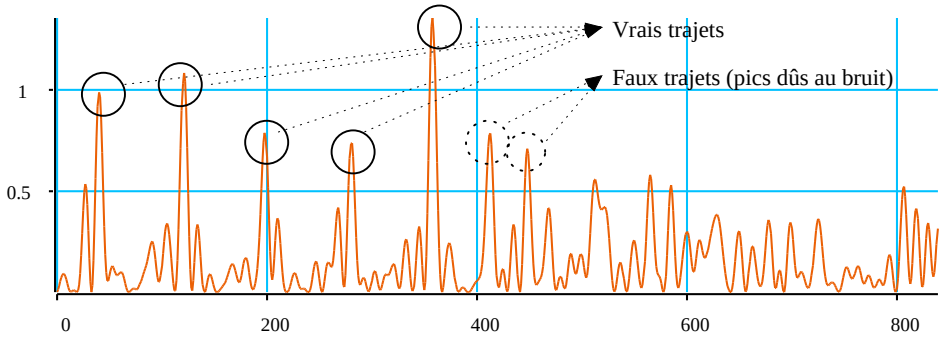


FIG. II.15 – Profil de canal obtenu par intercorrélation avec un nombre de chips pilotes faible ($N_{cp} = 32$) et cinq trajets de même puissance

Dans les environnements fortement bruités et dans les cas où le nombre de chips pilotes émis est faible, il est possible d'utiliser non pas *un* mais *deux* seuils sur le module de la corrélation [FSA98]. Ainsi, les trajets ne se détachant pas assez du niveau de bruit moyen sont éliminés du processus de recombinaison. Le problème du réglage des seuils est alors doublement présent.

Critère innovant : calcul du nombre de symboles pilotes correctement démodulés

Dans le cadre de ce travail, nous avons imaginé qu'un autre critère possible pour juger de la validité d'un pic de corrélation en tant qu'indicateur éventuel de la présence effective d'un trajet à l'entrée du Searcher est le nombre de symboles ou de chips pilotes correctement démodulés par un récepteur Rake théorique qui serait calé sur ce trajet potentiel. Ainsi, en ne démodulant que le trajet en cours d'évaluation, on peut définir une politique de validation basée sur la variation de cette quantité de pilotes (chips ou symboles) correctement démodulés au fur et à mesure des itérations [Bat01].

En notant
- i le numéro du trajet en cours de validation,
- $r(t)$ le signal reçu,
- T_c la durée d'un chip,
- $\hat{\tau}_i$ le retard associé au trajet i relativement à une base de temps donnée,
- $\hat{\beta}_i$ l'atténuation complexe estimée du trajet i,
- $C(k)$ le k-ème élément du code complexe ayant été utilisé à l'émission (combinaison du code d'étalement et du code d'embrouillage),

les symboles pilotes $\{\hat{a}_l\}_{l\in[0,N_{sp}-1]}$ sont reconstitués de la manière suivante :

$$\hat{b}_k = \frac{\hat{\beta}_i^*}{SF}\sum_{n=0}^{SF-1} r(k\cdot SF\cdot T_c + n\cdot T_c - \hat{\tau}_i)\times C^*(k\cdot SF+n),\ k\in\left[0,\frac{N_{sp}}{2}-1\right] \tag{II.49}$$

$$\{\hat{a}_l\}_{l\in[0,N_{sp}-1]} = \bigcup_{k=0}^{k=\frac{N_{sp}}{2}-1}\{\hat{a}_{2k},\hat{a}_{2k+1}\} = \bigcup_{k=0}^{k=\frac{N_{sp}}{2}-1}\left\{sign(\mathcal{R}(\hat{b}_k)), sign(\mathcal{I}(\hat{b}_k))\right\} \tag{II.50}$$

Le nombre q_i de symboles pilotes correctement démodulés à partir du i-ème trajet est obtenu en comparant les symboles pilotes reconstitués $\{a_l\}_{l\in[0,N_{sp}-1]}$ aux symboles pilotes originellement émis $\{a_l\}_{l\in[0,N_{sp}-1]}$:

$$q_i = card\left\{\hat{a}_l\,/\hat{a}_l\neq a_l\right\}_{l\in[0,N_{sp}-1]} \tag{II.51}$$

En se basant sur le fait que les symboles issus d'un désétalement non synchronisé avec un trajet véritable n'auront rien à voir avec les véritables symboles pilotes émis, on ne peut retenir par exemple que les trajets qui permettent de démoduler *au minimum* une certaine part des symboles ou des chips pilotes émis (la moitié étant le minimum[19]). On peut aussi mettre ce seuil en relation avec la quantité de symboles pilotes correctement démodulés par le premier trajet identifié, et ne pas autoriser cette métrique à décroître dans l'absolu, ou alors ne pas l'autoriser à décroître de plus d'une certaine valeur au fur et à mesure des itérations, et rejeter ainsi les trajets qui entraîneraient une dégradation trop brutale de la qualité de la démodulation des symboles pilotes.

La complexité calculatoire de ce critère est raisonnable :

- le désétalement de deux symboles pilotes nécessite SF multiplications du signal complexe reçu par le conjugué du chip original correspondant, de la forme $\pm 1 \pm j$, ce qui engendre $2\times SF$ opérations réelles par symbole pilote[20], et l'accumulation de ces SF nombres complexes. Le désétalement de tous les symboles pilotes nécessite donc :

$$2\times SF\times N_{sp} \text{ opérations réelles.} \tag{II.52}$$

- la remise en phase des symboles pilotes désétalés par leur multiplication par le conjugué de l'atténuation complexe estimée effectue $\frac{N_{sp}}{2}$ multiplications complexes, soit

$$3\times N_{sp} \text{ opérations réelles.} \tag{II.53}$$

- l'extraction des signes des symboles obtenus et la comparaison des symboles pilotes obtenus aux symboles originaux coûtent chacune

$$N_{sp} \text{ opérations réelles.} \tag{II.54}$$

La complexité calculatoire de l'évaluation de cette métrique pour *une itération* est donc de :

$$(5+2\times SF)\times N_{sp} \text{ opérations réelles.} \tag{II.55}$$

Le tableau II.3 donne la complexité en opérations réelles d'*une* évaluation de cette métrique dans les différents cas de figure possibles, ainsi que le rapport de cette complexité à celle d'une itération du Searcher pour différentes valeurs du facteur de suréchantillonnage.

[19]En cas de désétalement d'un faux trajet, les symboles obtenus ont les caractéristiques d'un bruit blanc, et ont donc en moyenne une chance sur deux d'être corrects.

[20]La division par SF est ignorée, elle ne pose en effet qu'un problème d'échelle facilement contourné.

N_{sp}	SF	N_{cp}	Complexité	$OSF = 2$	$OSF = 4$	$OSF = 8$
4	512	1024	4116	0,276 %	0,069%	0,017 %
2	256	256	1034	0,277 %	0,070%	0,017 %
4	256	512	2068	0,277 %	0,070%	0,017 %
8	256	1024	4136	0,277 %	0,070%	0,017 %
4	128	256	1044	0,280 %	0,070%	0,018 %
8	128	512	2088	0,280 %	0,070%	0,018 %
8	64	256	1064	0,285 %	0,072%	0,018 %
8	32	128	552	0,295 %	0,074%	0,019 %
16	16	128	592	0,317 %	0,080%	0,020 %
16	8	64	336	0,359 %	0,090%	0,023 %
16	4	32	208	0,442 %	0,112%	0,028 %

TAB. II.3 – Complexité en opérations réelles de l'évaluation du nombre de pilotes correctement démodulés et comparaison avec la complexité d'une itération du Searcher

Cette métrique est d'autant plus intéressante dans le cadre de l'utilisation d'un récepteur Rake pour la raison suivante : le fait que le Searcher traite chaque trajet en ayant *soustrait* les trajets plus puissants, ce que ne fait pas le récepteur Rake, permet d'affirmer que le récepteur Rake recupèrera *moins* d'information d'un trajet bruité que le Searcher. Un trajet jugé trop bruité par le Searcher car ne permettant pas la démodulation correcte d'un *minimum* de symboles pilotes n'a donc que peu d'intérêt aux yeux du récepteur Rake, et il est alors préférable d'exclure ces trajets trop bruités du processus de recombinaison.

Pour gagner en flexibilité, il est possible de choisir comme métrique le nombre de *chips* pilotes mal démodulés, et non le nombre de *symboles* pilotes, mais ce choix implique un nombre de multiplications complexes SF fois plus importants, faisant augmenter dans des proportions non négligeables la complexité calculatoire de l'évaluation du critère, diminuant par là même son utilité.

Pour pouvoir gérer correctement le plus grand nombre de situations possibles, une combinaison de critères semble nécessaire. Il apparaît judicieux d'utiliser conjointement un seuil sur le module carré de l'intercorrélation, pour empêcher la prise en compte de trajets trop faibles lors du processus de recombinaison et un critère basé sur la qualité de la démodulation offerte par le trajet en cours d'identification. Limiter le nombre d'itérations effectuées par l'algorithme est aussi nécessaire à l'éventuelle satisfaction de contraintes temps réel.

II.3.3.5 Evaluation des performances de l'algorithme proposé

Avant de vérifier la correction fonctionnelle de l'algorithme proposé, il est nécessaire de préciser qu'on n'effectuera pas d'étude de performances au sens classique du terme. En effet, on ne s'intéressera pas ici à l'évaluation du taux d'erreurs binaires en sortie du récepteur Rake utilisant les données issus du Searcher proposé dans différents environnements radiomobiles, avec différents niveaux de bruits, à différentes vitesses ... On évaluera plutôt la qualité de l'estimation de canal en comparant les coefficients utilisés lors de la simulation aux résultats obtenus par l'algorithme proposé.

Ces études de performances exhaustives étant en effet très coûteuses en termes de temps de calcul, on préfèrera les confier à un *prototype* réalisant la fonction décrite. Le principal bénéfice à tirer d'une telle migration des calculs d'une station de travail à un prototype mimant une véritable architecture contrainte est la possibilité d'examiner les points caractéristiques d'une certaine implémentation de l'algorithme qui seraient dissimulés par une exécution sur une machine de type PC : degré d'adéquation à une architecture de type DSP (fondamentalement différente de celle des processeurs équipant les stations de travail actuelles), consommation de bande passante, quantité d'accès à la mémoire ... Effectuer l'étude architecturale *avant* les études de performance est donc un moyen *d'augmenter* la quantité d'information obtenue par l'exécution des études de performances, de compléter les résultats numériques par des considérations architecturales pouvant impacter des caractéristiques plus concrètes de l'implémentation de l'algorithme, telles que le temps de réponse ou l'utilisation des bus mémoires, par exemple.

Bien sur, cette démarche n'a d'intérêt *que* dans le cas où une implémentation en environnement contraint est envisagée. Dans le cas contraire, un PC effectuera les mêmes calculs plus vite et offrira plus de possibilités de visualisations et/ou post-traitements que l'exécution de certains scénarios sur un prototype.

Les trois défauts majeurs *identifiés* de l'algorithme proposé sont :

- l'impossibilité de séparer les trajets séparés par une durée inférieure à un chip. Ceci est dû à la nature du processus de calcul de la métrique servant à identifier les trajets, à savoir le calcul de l'intercorrélation entre le signal reçu et le signal pilote idéal, régénéré au sein du récepteur.
- l'erreur commise sur l'estimation du module des trajets identifiés. Le module du premier trajet identifié étant inévitablement entâché de l'erreur due à l'interférence d'accès multiple, la phase de pondération/suppression va *propager* une partie de cette erreur lors des itérations successives de l'algorithme. En effet, l'orthogonalité des canaux entre eux étant rompue par la présence de répliques retardées du signal émis à l'entrée du récepteur, l'intercorrélation entre le trajet principal et les autres trajets n'est pas nulle[21]. L'atténuation complexe du premier trajet détecté est erronée car les autres trajets contribuent comme du bruit au calcul de ce point de l'intercorrélation. De ce fait, lors de la suppression du signal pilote pondéré par le premier coefficient estimé, une erreur est introduite, le module du coefficient estimé étant différent du module du coefficient original [FG99]. Ce point justifie d'autant plus l'introduction d'un système de limitation du nombre d'itérations effectuées par l'algorithme : la quantité de signal soustraite lors du premier trajet étant erronée, une fois tous les autres trajets significatifs soustraits, le processus de corrélation peut éventuellement resélectionner le premier trajet en considérant l'erreur introduite lors de la soustraction comme un trajet à part entière.
- la variation importante de son temps d'exécution. Comme tous les algorihtmes itératifs n'effectuant pas un nombre fixé d'itérations, le temps nécessaire à l'algorithme pour effectuer l'estimation du canal peut varier dans de grandes proportions, même avec tous ses paramètres fixés (N_{cp}, OSF, DS). Le système utilisant cet algorithme doit prendre cet aspect en compte et adopter un comportement *globalement asynchrone* vis-à-vis de la tâche d'estimation de canal, sous peine de devoir se contenter d'une estimation « pire-cas » du temps d'exécution, et donc de se comporter de manière hautement sous-optimale.

[21] On retrouve l'interférence d'accès multiple, ou *MAI*, mentionnée plus haut

Une fois ces défauts précisés, il est intéressant d'examiner le comportement de l'algorithme proposé dans certaines situations caractéristiques. Le facteur de suréchantillonnage OSF a été pris égal à 8 pour tous les exemples présentés ci-dessous.

Le premier exemple illustre une situation où l'algorithme fonctionne bien malgré un nombre de chips pilotes employés assez faible ($N_{cp} = 32$). Deux trajets de même puissance sont simulés, séparés par une durée légèrement inférieure au temps-symbole (3.75 chips). Les résultats obtenus dans le cas défavorable $N_{cp} = 32$ et dans un cas plus favorable $N_{cp} = 512$, ainsi que les erreurs relatives aux valeurs originales, sont donnés par les tableaux II.4 et II.5. La valeur entre parenthèses correspond à l'erreur relative donnée sous la forme d'un pourcentage pour le module, et à la valeur absolue de l'erreur commise sur pour la phase, exprimée en radians. On note qu'avec un nombre de chips pilotes plus importants, la surestimation du module du premier trajet identifié est plus flagrante.

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
0	1	-1,5740	1,0384 (3,8 %)	-1,6768 (0,103)
30	1	0,7712	1,0546 (5,5 %)	0,8111 (0,040)

TAB. II.4 – Résultats obtenus par l'algorithme original dans le cas de 2 trajets de même puissance, $N_{cp} = 32$

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
0	1	-1,5740	1,1068 (10,7 %)	-1,6661 (0,092)
30	1	0,7712	0,9762 (2,4 %)	0,7694 (0,002)

TAB. II.5 – Résultats obtenus par l'algorithme original dans le cas de 2 trajets de même puissance, $N_{cp} = 512$

Dans le cas où le nombre de trajets augmente sensiblement (six trajets d'égale puissance sont désormais simulés), les limites de la politique de gestion du critère d'arrêt se font sentir. Ainsi, l'adoption d'une politique imposant la stricte décroissance du nombre de symboles pilotes mal démodulés au fur et à mesure des itérations a pour conséquence l'arrêt anticipé de l'algorithme comme illustré par les tableaux II.6 ($N_{cp} = 1024$) et II.7 ($N_{cp} = 32$). Avec 32 chips pilotes, l'algorithme s'arrête trop tôt et n'identifie que 2 trajets sur les 6 présents à l'entrée du récepteur, conduisant ainsi le récepteur Rake à négliger une grande partie de l'information potentiellement utilisable. De plus, l'identification effectuée avec 32 chips pilotes est loin d'être de bonne qualité. D'autres simulations effectuées avec $N_{cp} = 32$ ont mis en évidence l'incapacité de l'algorithme à identifier plus de 3 trajets sur les 6 présents, qui plus est en commettant des erreurs importantes lors de l'estimation de la phase du trajet considéré.

Un modèle de canal plus réaliste (canal de type 3 dans [3GP01a, annexe B]) dans lequel la puissance des trajets décroît avec leur retard par rapport au trajet principal ne pose que peu de problème à l'algorithme proposé, du fait du caractère itératif de l'estimation. L'identification correcte des trajets les plus faibles est rendue possible par la suppression préalable des trajets

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
400	1	-1,6539	1,0194 (1,2 %)	-1,6059 (0,048)
320	1	-2,5346	1,0017 (0,2 %)	-2,5546 (0,020)
160	1	-2,2882	1,0030 (0,3 %)	-2,2357 (0,053)
0	1	1,8848	0,9773 (2,3 %)	1,9070 (0,022)
80	1	0,5187	0,9528 (4,7 %)	0,5334 (0,015)
240	1	1,3816	0,9888 (1,1 %)	1,3910 (0,009)

TAB. II.6 – Résultats obtenus par l'algorithme original dans le cas de 6 trajets de même puissance, $N_{cp} = 1024$

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
0	1	1,8848	1,2118 (21 %)	1,6987 (0,186)
398	1	-1,6539	1,1963 (20 %)	-1,1903 (0,464)

TAB. II.7 – Résultats obtenus par l'algorithme original dans le cas de 6 trajets de même puissance, $N_{cp} = 32$

plus puissants. Le canal utilisé ici est constitué de 4 trajets distants d'un chip les uns des autres, dont les atténuations relatives sont 0, -3, -6 et -9 dB. Les résultats obtenus pour $N_{cp} = 64$ et $N_{cp} = 512$ sont donnés tableaux II.8 et II.9. Avec $N_{cp} = 32$, le critère de stricte décroissance du nombre de pilotes mal démodulés provoque l'arrêt anticipé des itérations après l'identification du seul trajet principal.

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
0	1	1,8076	1,043 (4,3 %)	1,7032 (0,104)
6	0,708	0,3331	0,596 (15,8 %)	0,0575 (0,276)
15	0,501	-2,1713	0,449 (10,4 %)	-1,9602 (0,211)
24	0,355	-0,1941	0,344 (3,1 %)	-0,1592 (0,035)

TAB. II.8 – Résultats obtenus par l'algorithme original dans le cas de 4 trajets de puissances décroissantes, $N_{cp} = 64$

II.3.4 Estimation itérative optimisée

Il a été montré précédemment que la complexité calculatoire de l'algorithme d'estimation itérative de canal à suppression de trajets est trop importante pour permettre une utilisation efficace de cet algorithme dans de bonnes conditions. La phase de calcul de l'intercorrélation, répétée à chaque itération, pèse en effet trop lourd dans le bilan de complexité. Sa complexité varie avec *le carré* du facteur de suréchantillonnage, ce qui empêche l'utilisation d'un facteur de suréchantillonnage supérieur ou égal à 4, et alourdit dans une moindre mesure les calculs

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
0	1	1,8076	1,046 (4,6 %)	1,6257 (0,182)
6	0,708	0,3331	0,681 (3,8 %)	0,1224 (0,211)
16	0,501	-2,1713	0,391 (22 %)	-2,0331 (0,138)
24	0,355	-0,1941	0,343 (3,4 %)	-0,2142 (0,020)

TAB. II.9 – Résultats obtenus par l'algorithme original dans le cas de 4 trajets de puissances décroissantes, $N_{cp} = 512$

lorsque beaucoup de chips pilotes sont utilisés ($N_{cp} = 512$ ou 1024). Il est cependant possible de modifier l'algorithme présenté ci-dessus afin d'en réduire drastiquement la complexité.

II.3.4.1 Modification du procédé de calcul de l'intercorrélation

En notant $\{s_p(n)\}_{n\in[0,N_{cp}\times OSF-1]}$ le signal pilote complexe régénéré échantillonné à OSF fois le rythme chip, et $\{r(n)\}_{n\in[0,(N_{cp}+DS)\times OSF-1]}$ la portion de signal reçu correspondant aux pilotes échantillonnée à ce même rythme chip, les points de l'intercorrélation $\{\gamma(k)\}_{k\in[0,DS\times OSF-1]}$ sont calculés de la manière suivante :

$$\forall k \in [0, DS \times OSF - 1], \gamma(k) = \frac{1}{N_{cp} \times OSF} \sum_{i=0}^{N_{cp}\times OSF-1} r(k+i)s_p^*(i) \qquad \text{(II.56)}$$

L'optimisation proposée ici est le passage d'une corrélation entre deux séquences échantillonnées à un rythme *multiple* du rythme *chip* (typiquement d'un facteur 2, 4, ou 8) à une corrélation entre deux séquences échantillonnées au rythme *chip*. Il est important de noter que le rythme de calcul des points de l'intercorrélation ne change pas : l'intercorrélation est calculée pour des retards exprimés en *échantillons*, mais chaque point est calculé comme l'intercorrélation de deux séquences échantillonnées au rythme *chip*, et non plus à OSF fois ce rythme.

Le signal pilote intervenant dans le calcul de l'intercorrélation est maintenant échantillonné au rythme chip, *sans prise en compte de la forme d'onde*. Il est maintenant constitué de chips "binaires" obtenus par l'étalement et l'embrouillage des symboles pilotes. Les valeurs de ces chips appartiennent à l'ensemble $\{+1, -1, +j, -j\}$, comme expliqué dans la section II.2.3. En notant $\{\tilde{s}_p(n)\}_{n\in[0,N_{cp}-1]}$ ces chips pilotes, la phase de calcul de l'intercorrélation devient :

$$\forall k \in [0, DS \times OSF - 1], \tilde{\gamma}(k) = \frac{1}{N_{cp}} \sum_{i=0}^{N_{cp}-1} r(k + i \times OSF)\tilde{s}_p^*(i) \qquad \text{(II.57)}$$

Ainsi, les coefficients estimés ne sont plus les coefficients du canal radiomobile seul, mais les coefficients "nets" du canal global constitué du canal radiomobile et des deux filtres de mise en forme, à l'émission et à la réception [BSRC98].

Une autre solution est de calculer l'intercorrélation du signal pilote et du signal reçu non pas dans le domaine temporel, mais dans le domaine fréquentiel, à l'aide d'une transformée de Fourier rapide. Ce processus nécessite néanmoins le calcul de la transformée de Fourier de la portion de signal reçu (évoluant au fil des itérations) à chaque itération, ainsi que le calcul

de la transformée de Fourier inverse du produit fréquentiel pour pouvoir extraire la position et les caractéristiques du trajet le plus puissant. Cette approche a beau réduire la complexité d'une itération, elle reste riche en multiplications complexes et ne se prête que relativement peu à une implémentation hybride matérielle-logicielle[22]. De plus, la transformée de Fourier d'une séquence de nombres complexes de la forme $\pm 1, \pm j$ n'a aucune propriété particulière qui faciliterait son calcul.

II.3.4.2 Impact de l'optimisation proposée sur la complexité de l'algorithme

Cette modification du processus de calcul de l'intercorrélation a deux conséquences majeures, qui sont de natures différentes : la première consiste en une réduction du nombre d'opérations à effectuer, tandis que la seconde se traduit par la modification de la *nature* même des opérations effectuées : les suites de multiplications complexes sont en effet remplacées par des accumulations de produits du type *nombre* $\times$ *bit*.

Tout d'abord, la complexité de la phase de calcul diminue dans des proportions non négligeables. En effet, le nombre de points à calculer reste le même, mais le calcul d'un point coûte beaucoup moins cher : N_{cp} multiplications complexes, ou $6 \times N_{cp}$ opérations réelles, par point avec le nouveau procédé de calcul, contre $6 \times N_{cp} \times OSF$ opérations réelles avec l'ancien procédé. La complexité totale de la phase de calcul de l'intercorrélation ne coûte donc plus que

$$6 \times N_{cp} \times DS \times OSF \text{ opérations réelles,} \tag{II.58}$$

ce qui réduit la complexité totale d'une itération à

$$2 \times (3N_{cp} \times DS + 4N_{cp} + 2DS) \times OSF + 20 \text{ opérations réelles.} \tag{II.59}$$

De plus, en examinant de plus près le nouveau procédé de calcul de l'intercorrélation, on s'aperçoit que les produits intervenant dans le calcul de $\tilde{\gamma}(k)$ sont de la forme $r(k+i\times OSF) \times \tilde{s}_p^*(i)$, où $r(k + i \times OSF)$ est un nombre complexe dont les parties réelle et imaginaire sont à valeurs réelles[23], tandis que $\tilde{s}_p(i)$, qui correspond au chip pilote étalé et embrouillé, est par construction de la forme $\pm\{1, j\}$. Le produit à calculer est donc *formulé* comme une multiplication complexe, qui nécessite théoriquement 6 opérations réelles, alors qu'il n'y a pas de *véritable* multiplication à effectuer. Le produit $r(k+i\times OSF) \times \tilde{s}_p^*(i)$ ne peut en effet prendre que certaines valeurs bien particulières. En notant $r(k,i) = r(k+i\times OSF)$, ces valeurs sont :

$$r(k,i) \times \tilde{s}_p^*(i) = \begin{cases} \mathcal{R}\{r(k,i)\} & + & j\mathcal{I}\{r(k,i)\} & \text{si } \tilde{s}_p^*(i) = 1 \\ -\mathcal{R}\{r(k,i)\} & - & j\mathcal{I}\{r(k,i)\} & \text{si } \tilde{s}_p^*(i) = -1 \\ \mathcal{I}\{r(k,i)\} & - & j\mathcal{R}\{r(k,i)\} & \text{si } \tilde{s}_p^*(i) = j \\ -\mathcal{I}\{r(k,i)\} & + & j\mathcal{R}\{r(k,i)\} & \text{si } \tilde{s}_p^*(i) = -j \end{cases} \tag{II.60}$$

L'accumulation de produits de cette forme ne nécessite donc pas 1 multiplication complexe par élément à accumuler, mais 2 opérations réelles, aucun produit de nombres réels n'étant véritablement calculé. La complexité de la phase de calcul de l'intercorrélation est donc de

$$2 \times N_{cp} \times DS \times OSF \text{ opérations réelles,} \tag{II.61}$$

[22] Sauf bien sûr si l'on envisage l'utilisation d'une FFT câblée, ce qui revient à nier la potentielle efficacité du DSP, dont la raison d'être première est justement le calcul des corrélations et des FFT (trop lourd pour pouvoir être effectué par des processeurs classiques).

[23] Réelles prend ici le sens de *non-entières*.

et la complexité totale d'*une seule* itération de l'algorithme devient

$$2 \times (N_{cp} \times DS + 4N_{cp} + 2DS) \times OSF + 20 \text{ opérations réelles.} \tag{II.62}$$

Le principal avantage de cette évolution en termes de nombre d'opérations à effectuer est que la complexité d'une itération ne varie plus avec *le carré* du facteur de suréchantillonnage, mais *linéairement* avec celui-ci. Les complexités d'une itération de l'ancien et du nouvel algorithme sont comparées dans le tableau II.10, qui chiffre la complexité d'une itération de l'algorithme optimisé et évalue le ratio de cette complexité à celle d'une itération de l'algorithme original.

	$OSF = 2$	$OSF = 4$	$OSF = 8$
N_{cp}=32	8692 (18,5 %)	17364 (9,3 %)	34078 (4,7 %)
N_{cp}=64	16884 (18 %)	33748 (9,1 %)	67476 (4,6 %)
N_{cp}=128	33268 (17,8 %)	66516 (9 %)	133012 (4,5 %)
N_{cp}=256	66036 (17,7 %)	132052 (8,9 %)	264084 (4,5 %)
N_{cp}=512	131572 (17,6 %)	263124 (8,9 %)	526228 (4,4 %)
N_{cp}=1024	262644 (17,6 %)	525268 (8,9 %)	1050516 (4,4 %)

TAB. II.10 – Complexité en opérations réelles d'une itération de l'algorithme optimisé, et comparaison avec la complexité d'une itération de l'algorithme original

Une fois l'optimisation proposée implémentée, la phase de calcul de l'intercorrélation ne fait plus intervenir de multiplication réelle. Les seules phases ayant recours à de telles opérations sont les phases de pondération du signal pilote régénéré avant sa suppression ainsi que l'évaluation du critère déclenchant l'arrêt des itérations. De la même manière, la phase de pondération/suppression du signal pilote régénéré est la seule à être effectuée au rythme *échantillon*, toutes les autres phases de calcul se déroulant désormais au rythme *chip*.

Il est important de noter que l'optimisation proposée permet de réaliser l'estimation de l'atténuation complexe du trajet en cours d'identification *en même temps* que le calcul de l'intercorrélation. En effet, une fois le retard $\hat{\tau}_i$ déterminé, l'atténuation complexe $\hat{\beta}_i$ est estimée de la manière suivante par l'algorithme original :

$$\hat{\beta}_i = \frac{\gamma_i(\hat{\tau}_i)}{\frac{1}{T_p}\int_p |s_p(t)|^2 \, dt} \tag{II.63}$$

Une fois l'optimisation proposée implémentée, l'atténuation complexe $\hat{\beta}_i$ est estimée comme suit :

$$\hat{\beta}_i = \frac{\tilde{\gamma}_i(\hat{\tau}_i)}{\frac{1}{N_{cp}}\sum_{k=0}^{N_{cp}-1} |\tilde{s}_p(k)|^2} \tag{II.64}$$

Or,

$$\begin{aligned} \forall k \in [0, N_{cp}-1], \tilde{s}_p(k) \in \{1, -1, j, -j\} \quad &\Rightarrow \quad \forall k \in [0, N_{cp}-1], |\tilde{s}_p(k)| = 1 \\ &\Rightarrow \quad \frac{1}{N_{cp}} \sum_{k=0}^{N_{cp}-1} |\tilde{s}_p(k)|^2 = 1 \\ &\Rightarrow \quad \hat{\beta}_i = \tilde{\gamma}_i(\hat{\tau}_i) \end{aligned} \tag{II.65}$$

La phase d'estimation de l'atténuation complexe est désormais superflue, ce qui fait disparaître par la même occasion le besoin d'effectuer deux divisions réelles. De la même manière, le récepteur Rake n'aura pas à effectuer ces calculs lors de la démodulation des données.

II.3.4.3 Exemples de résulats comparés des deux versions de l'algorithme

L'algorithme optimisé estimant l'intercorrélation et les atténuations complexes des trajets sur des populations d'échantillons OSF fois plus petites que l'algorithme original, il est normal de s'intéresser à la dégradation éventuelle des performances qui s'en suit. Les exemples présentés ci-dessous ne sont pas issus d'une véritable étude de performances exhaustive, mais doivent plutôt être considérés comme raisonnablement représentatifs du comportement des deux algorithmes. Aucun bruit additif n'ayant été simulé, les seules interférences présentes sont dues aux différents trajets se bruitant les uns les autres.

Les résultats obtenus par l'algorithme optimisé vont donc être comparés à ceux obtenus par l'algorithme original dans divers cas de figure. Les résultats obtenus en considérant un canal composé de deux trajets de même puissance pour les deux valeurs de N_{cp} testées précédemment sont donnés tableaux II.11 et II.12. Les deux versions de l'algorithme exhibent sensiblement le même comportement, et fournissent des estimations entâchées d'erreurs dont les ordres de grandeur sont comparables, avec un léger avantage pour la version de l'algorithme incorporant l'optimisation proposée.

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
30	1	2,885	0,989 (1,1 %)	2,766 (0,119)	0,996 (0,4 %)	2,803 (0,082)
0	1	-2,123	0,985 (1,5 %)	-2,127 (0,004)	0,994 (0,6 %)	-2,129 (0,006)

TAB. II.11 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 2 trajets de même puissance, $N_{cp} = 512$

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
30	1	2,885	1,041 (4,1 %)	2,801 (0,084)	1,040 (4,0 %)	2,809 (0,076)
0	1	-2,123	0,969 (3,1 %)	-2,028 (0,095)	0,970 (3,0 %)	-2,058 (0,065)

TAB. II.12 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 2 trajets de même puissance, $N_{cp} = 32$

Avec un environnement radiomobile plus chargé (6 trajets) et donc plus bruité, un nombre de chips pilotes $N_{cp} = 32$ et une politique de stricte décroissance du nombre de symboles pilotes mal démodulés, les deux versions de l'algorithme n'identifient correctement que le premier trajet, se heurtant ensuite à un critère d'arrêt des itérations trop rigide. Les résultats obtenus par les deux algorithmes après l'inhibition de la métrique basée sur le nombre de symboles pilotes mal démodulés sont présentés tableaux II.14 et II.15. Les trajets n'étant pas détectés dans le même ordre, les résultats obtenus par les deux algorithmes sont présentés sur deux tableaux. Le modèle du canal utilisé lors de la simulation est présenté tableau II.13. Le critère d'arrêt des

itérations utilisé ici est le seuil sur le module, inopérant dans un environnement aussi bruité, et la fixation d'une limite au nombre d'itérations effectuées (6, ici). L'algorithme original détecte deux faux trajets, dont un qui précède un véritable trajet dans l'ordre de détection, alors que l'algorithme optimisé identifie 4 trajets aux bonnes positions temporelles avant de détecter lui aussi de faux trajets. On peut voir à la très lente décroissance des modules des trajets détectés que le seul critère du seuil d'amplitude ne constitue pas un critère valable dans le cas présent. Il est intéressant de noter que les trajets situés aux retard 160 et 320, présents dans le modèle de canal utilisé, ne sont pas détectés avant au minimum la septième itération de l'algorithme. On peut en effet observer leur relative insignifiance sur la figure II.16, ainsi que l'insuffisance du simple seuil sur le module de la corrélation pour décider ou non d'interrompre les itérations.

	Coefficients simulés	
$\hat{\tau}_i$ (éch.)	Module	Phase
0	1	0,955
80	1	-0,350
160	1	2,655
240	1	2,253
320	1	-1,887
400	1	-3,044

TAB. II.13 – Caractéristiques des trajets composant le canal utilisé lors des simulations présentées tableaux II.14 et II.15

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
83	1	-0,350	1,318 (31,8 %)	0,134 (0,484)
403	1	-3,044	1,023 (2,3 %)	-2,946 (0,098)
241	1	2,253	0,957 (4,3 %)	2,296 (0,043)
415	N/A	N/A	0,842	2,036
0	1	0,955	0,833 (16,7 %)	0,895 (0,060)
707	N/A	N/A	0,816	0,302

TAB. II.14 – Résultats obtenus par l'algorithme original dans le cas de 6 trajets de même puissance, $N_{cp} = 32$

Dans la même situation mais avec un nombre de chips pilotes $N_{cp} = 128$, les deux versions de l'algorithmes parviennent à identifier correctement les six trajets avant que le critère de stricte décroissance du nombre de symboles pilotes mal démodulés ne mette fin au déroulement de l'algorithme, comme indiqué tableau II.16.

Enfin, la simulation d'un environnement radiomobile plus réaliste, où le canal est composé de quatre trajets de puissances décroissantes, illustre une fois de plus la similitude des comportements des deux versions de l'algorithme, comme indiqué tableaux II.17, II.18 et II.19. Avec $N_{cp} = 64$, le dernier trajet n'est pas détecté du fait du critère de stricte décroissance du nombre de symboles pilotes mal démodulés, mais dès que $N_{cp} \geq 128$, les quatre trajets sont correctement identifiés. Les résultats obtenus pour $N_{cp} = 1024$, donnés tableau II.19, illustre

	Coefficients simulés		Coefficients estimés	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase
82	1	-0,350	1,269 (26,9 %)	0,109 (0,459)
240	1	2,253	1,005 (0,5 %)	2,416 (0,163)
401	1	-3,044	0,986 (1,4 %)	-2,768 (0,276)
0	1	0,955	0,856 (14,4 %)	0,904 (0,051)
414	N/A	N/A	0,846	2,072
575	N/A	N/A	0,820	-2,317

TAB. II.15 – Résultats obtenus par l'algorithme optimisé dans le cas de 6 trajets de même puissance, $N_{cp} = 32$

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
400	1	-3,067	1,168 (16,8 %)	-3,025 (0,042)	1,229 (22,9 %)	-3,080 (0,013)
80	1	-0,352	1,179 (17,9 %)	-0,439 (0,087)	1,219 (21,9 %)	-0,436 (0,084)
0	1	0,969	0,993 (0,7 %)	0,993 (0,024)	0,993 (0,7 %)	1,017 (0,048)
320	1	-2,046	1,024 (2,4 %)	-2,076 (0,030)	0,994 (0,6 %)	-2,111 (0,065)
160	1	-2,165	0,975 (2,5 %)	-2,150 (0,015)	0,973 (2,7 %)	-2,146 (0,019)
240	1	3,095	0,929 (7,1 %)	-3,052 (0,043)	0,908 (9,2 %)	-3,072 (0,023)

TAB. II.16 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 6 trajets de même puissance, $N_{cp} = 128$

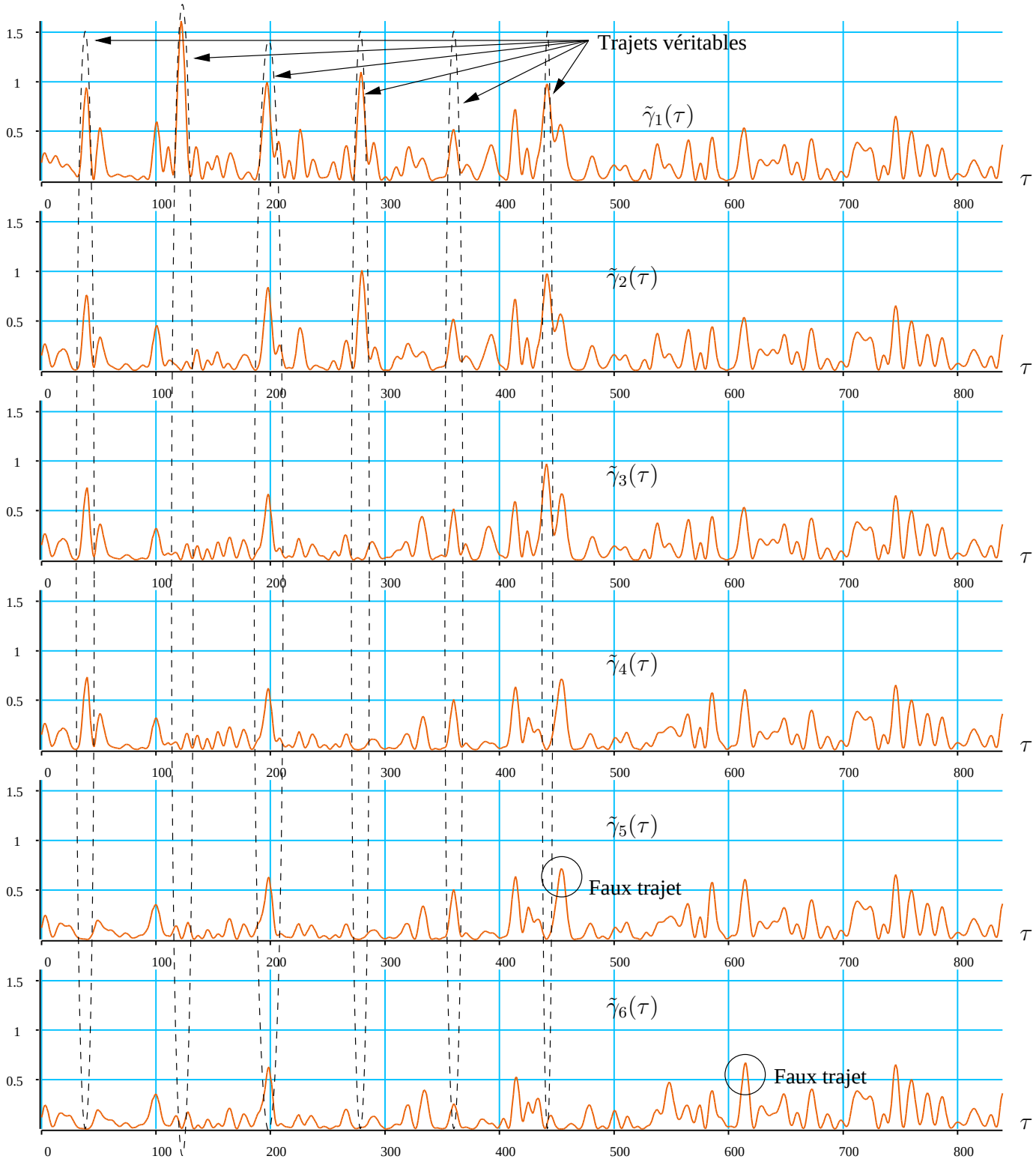


FIG. II.16 – Variation de l'intercorrélation calculée par l'algorithme optimisé dans le cas de six trajets de même puissance, $N_{cp} = 32$

l'augmentation logique avec N_{cp} de la précision des estimations réalisées sur les trajets les plus significatifs.

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
0	1	-1,970	1,051 (5,1 %)	-1,937 (0,033)	1,052 (5,2 %)	-2,043 (0,073)
8	0,708	-1,291	0,669 (5,5 %)	-1,546 (0,255)	0,667 (5,8 %)	-1,301 (0,010)
16	0,501	-2,615	0,458 (8,6 %)	2,794 (0,855)	0,442 (11,8 %)	-2,871 (0,256)

TAB. II.17 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 4 trajets de puissances décroissantes, $N_{cp} = 64$

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
0	1	-1,970	0,972 (2,8 %)	-1,892 (0,078)	0,972 (2,8 %)	-1,930 (0,040)
8	0,708	-1,291	0,692 (2,3 %)	-1,591 (0,300)	0,662 (6,5 %)	-1,582 (0,291)
16	0,501	-2,615	0,472 (5,8 %)	3,058 (0,610)	0,473 (5,6 %)	3,123 (0,545)
24	0,355	1,160	0,299 (15,8 %)	1,487 (0,327)	0,307 (13,5 %)	1,442 (0,282)

TAB. II.18 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 4 trajets de puissances décroissantes, $N_{cp} = 128$

	Coefficients simulés		Alg. original		Alg. optimisé	
$\hat{\tau}_i$ (éch.)	Module	Phase	Module	Phase	Module	Phase
0	1	-1,970	1,001 (0,1 %)	-1,874 (0,096)	0,999 (0,1 %)	-1,909 (0,061)
8	0,708	-1,291	0,698 (1,4 %)	-1,615 (0,324)	0,674 (5,0 %)	-1,610 (0,319)
16	0,501	-2,615	0,460 (8,2 %)	3,059 (0,609)	0,468 (7,1 %)	3,120 (0,548)
24	0,355	1,160	0,283 (20,3 %)	1,385 (0,225)	0,289 (18,6 %)	1,369 (0,209)

TAB. II.19 – Comparaison des résultats obtenus par l'algorithme original et l'algorithme optimisé dans le cas de 4 trajets de puissances décroissantes, $N_{cp} = 1024$

II.3.4.4 Impact « architectural » de l'optimisation proposée

Les multiplications réelles effectuées par l'algorithme original lors du calcul de l'intercorrélation ont été remplacées par des opérations plus simples qui sont en fait des opérations *pseudo-réelles*. Les opérations effectuées sont en fait des accumulations/soustractions de nombres réels dont la valeur (la partie réelle ou imaginaire du chip traité) et le signe (addition ou soustraction à l'accumulateur) lors de l'opération sont commandés par le signe et le caractère complexe ou réel du chip pilote correspondant. Cette opération, exécutable par un organe de type DSP, n'est cependant pas optimale en termes d'utilisation des ressources de la machine. En effet, l'implémentation d'une telle suite d'opérations sur une architecture de DSP peut s'envisager de deux manières différentes :

1. Programmer le calcul de chaque point de l'intercorrélation comme une suite d'opérations élémentaires de type multiplication-accumulation complexe dont les multiplicandes seraient les chips du signal reçu, à valeurs réelles, et les chips pilotes, considérés de la même manière comme étant à valeurs réelles. L'avantage d'une telle implémentation est que le DSP peut alors mettre à profit sa capacité à effectuer rapidement des sommes de produits. On peut émettre la remarque que le DSP est alors sous-employé, utilisant son multiplieur pour multiplier des nombres par ± 1 avant de les additionner à un accumulateur interne. Un tel déploiement de moyens est à coup sûr sous-optimal en termes de consommation d'énergie et d'utilisation des ressources.
2. Programmer uniquement les additions/soustractions en les entourant de tests successifs pour déterminer la nature de l'opération (addition/soustraction) et son opérande (la partie réelle ou imaginaire du chip traité). Le code équivalent à une telle suite de calculs conditionnés est relativement inadapté aux DSP actuels, plus à l'aise lors de l'exécution de code *exempt* de tests.

L'implémentation de l'algorithme optimisé sur un seul DSP ne semble donc pas être la solution la plus pertinente au problème de l'embarquement de l'algorithme proposé. Par contre, la fait que les opérations effectuées soient des produits du type *nombre réel* $\times$ *bit* conduit à envisager l'adjonction au DSP d'un *coprocesseur* spécialisant ne réalisant que cette tâche, en consommant moins de ressources que le DSP. La méthodologie permettant de réaliser une telle architecture est présentée dans la deuxième partie.

Conclusion

L'UTRA (l'interface radio de l'UMTS) se décline selon deux modes, dont l'UTRA-FDD (ou Wideband CDMA), qui sera la première « version » déployée de l'UMTS. L'évolution majeure par rapport à GSM est l'utilisation du CDMA, ou Accès Multiple à Répartition par Codes, comme technologie d'accès multiple. L'UMTS et le WB-CDMA sont brièvement décrits dans le chapitre I.

Les dégradations du signal émis par la station de base induites par le passage dans le canal radiomobile ainsi que les moyens de restaurer dans la mesure du possible la qualité du signal original sont présentés au début du chapitre II.

L'estimation du canal radiomobile a été identifiée comme étant l'une des tâches les plus complexes, avec le décodage canal et l'égalisation, à effectuer par le terminal. Cependant, bien que de nombreux algorithmes sur le sujet soient présents dans la littérature, la majorité d'entre eux traite de l'égalisation dans son ensemble et délaisse l'estimation de canal proprement dite, la supposant parfaitement effectuée le plus souvent. Un algorithme original d'estimation de canal est donc proposé dans la suite du chapitre II.

L'algorithme en question procède par itérations successives au cours desquelles la position du trajet le plus significatif est détectée à partir de l'intercorrélation du signal reçu et du signal pilote régénéré à la réception, préalablement à la suppression éventuelle du trajet en question du buffer de travail. Un nouveau critère de validation des trajets détectés basé sur le nombre de symboles pilotes mal démodulés sur un trajet particulier est proposé par la même occasion.

La complexité de cet algorithme est ensuite évaluée, puis réduite par optimisation du calcul de l'intercorrélation, plus exactement par le passage d'une corrélation au rythme échantillon, typiquement égal à deux, quatre ou huit fois le rythme chip, à une corrélation au rythme chip. La complexité de l'algorithme passe d'une variation quadratique avec le facteur de suréchantillonnage en vigueur dans la chaîne de réception à une variation linéaire, ce qui permet à complexité comparable de travailler avec un facteur de suréchantillonnage plus élevé avec la version optimisée de l'algorithme qu'avec sa version originale. La précision atteinte dans l'estimation des instants d'arrivée des trajets est ainsi augmentée, sans dégradation apparente des performances. Des simulations plus poussées sont bien évidemment nécessaires pour pouvoir véritablement juger de la dégradation apportée par l'optimisation suggérée.

En lieu et place des multiplications complexes intervenant en majorité dans la version originale de l'algorithme, l'optimisation proposée fait intervenir des produits de type *nombre réel* $\times$ *bit*, certes implémentables sur un cœur de DSP, mais pas de manière efficace. Le postulat de départ étant celui d'une réduction de la complexité, il semble plus pertinent de confier ces calculs à un coprocesseur commandé par le DSP plutôt qu'au DSP lui-même. La méthodologie permettant de réaliser une telle architecture hybride est décrite dans la deuxième partie.

Deuxième partie

Codesign en vue du prototypage rapide

Chapitre III

Méthodologie de codesign

L'implémentation d'un algorithme tel que celui développé dans la première partie sur une architecture hétérogène pose un certain nombre de problèmes. L'architecture devant comporter une partie programmable et une partie réalisée en logique câblée, sa conception relève à la fois du génie logiciel et de la conception microélectronique, alors que les outils actuels sont plutôt tournés soit vers l'un soit vers l'autre. Ce point est détaillé section III.1.1.

Pour remédier à celà, des outils de conception dits "au niveau système" ont fait leur apparition vers la fin des années 90. Le point commun de tous ces outils est l'approche du système dans sa globalité dans un premier temps, dans le but de parvenir rapidement à un partitionnement efficace de l'algorithme à implémenter, avant de détailler isolément les parties logicielles et matérielles du système. Les fonctionnalités novatrices de ces outils sont detaillées section III.1.2.

Une méthodologie de conception innovante est ensuite proposée section III.2. Cette méthodologie, qui s'appuie sur l'outil N2C de CoWare [CoW01], utilise un langage de haut niveau dérivé du langage C pour la description des différents blocs composant le système, et favorise le raffinement du code plutôt que sa réécriture lors du passage aux différents niveaux de description supportés par l'outil. Les différentes phases de cette méthodologie, qui utilisent chacune un ensemble bien distinct des possibilités de l'outil N2C, ainsi que la fonctionnalité clé de l'outil N2C, à savoir la synthèse d'interface, sont illustrées sur un exemple simple qui montre le bénéfice qu'on peut tirer de l'introduction progressive du caratère temporel de l'architecture au sein de la description du système.

Enfin, l'adéquation de l'outil utilisé à des applications à forte composante « signal » est discutée, ainsi que les points faibles de la méthodologie proposée et des solutions potentielles.

III.1 Evolution des flots de conception

Concevoir des systèmes embarqués en ce début de millénaire n'a que peu de choses en commun avec la même activité pratiquée une dizaine d'années plus tôt. Les flots de conception alors utilisés à la limite de leurs capacités ont du évoluer et faire place à de nouveaux outils pour accompagner les nombreux changements de méthodologie de conception ayant eu lieu depuis. Vers la fin des années 90, de nouveaux flots sont apparus, offrant des possibilités inédites en matière de conception d'architectures hétérogènes. Cette section présente les problèmes liés à l'utilisation des méthodes traditionnelles de conception lors du développement de telles architectures,

ainsi que les solutions proposées par les flots les plus récents.

III.1.1 Les défis posés aux concepteurs aujourdhui

Le métier de concepteur de système embarqué a largement évolué du fait de plusieurs facteurs, dont entre autres les profondes modifications du marché ciblé par les produits développés. Des équipements de plus en plus complexes construits autour d'architectures hétérogènes ont en effet rendu désuètes les méthodes de conception des années passées. Les principaux points durs que rencontrent aujourd'hui les concepteurs de systèmes embarqués sont ici décrits rapidement.

III.1.1.1 Augmentation de la complexité des équipements microélectroniques

Le premier défi que doivent relever les concepteurs d'aujourd'hui est bien évidemment celui de la complexité sans cesse croissante des systèmes microélectroniques. En effet, les équipements microélectroniques grand public tels que les processeurs généralistes (Pentium II, III ou IV) ou spécialisés (GeForce II ou III, Radeon 8500, pour citer deux exemples tirés du monde des cartes graphiques pour PC) ou les processeurs destinés au monde de l'embarqué (présents dans les téléphones portables et les PDA) ont aujourd'hui une durée de vie relativement courte. Les différentes versions d'un même circuit se succédent à un rythme élevé, intégrant à chaque itération du processus de nouvelles fonctions ou des améliorations de celles qui étaient déjà présentes dans la version précédente. Les concepteurs se retrouvent ainsi avec des échéances de plus en plus proches les unes des autres et des circuits de plus en plus compliqués à concevoir.

C'est la technologie dans laquelle sont fabriqués ces circuits qui vient perturber ce rythme de fonctionnement et imposer une limite bientôt atteinte à l'expansion des circuits intégrés. Ainsi, pour les télécommunications mobiles, par exemple, la complexité des algorithmes requis par les différents standards augmente plus vite que les performances fournies par les processeurs. Cette situation est présentée figure III.1, extraite de [Fox01]. La loi de Moore, au départ une prédiction qui s'est avérée exacte et dont la validité perdure, dit que la puissance des processeurs double tous les 18 mois. L'augmentation de la complexité des standards de radiocommunication a été plus rapide que celle de la puissance des processeurs, et la conséquence principale de cette différence de rythme est que pour les industriels, attendre la montée en puissance des processeurs pour s'attaquer à un marché à un instant donné n'est plus commercialement viable. Le recours à des circuits câblés, bien souvent juxtaposés à des cœurs de processeurs plus classiques, est donc aujourd'hui indispensable.

Une autre contrainte commerciale est le *time-to-market*, le temps nécessaire au développement d'un produit avant de pouvoir le vendre. Sur des marchés évoluant de plus en plus rapidement, saturés de produits de plus en plus complexes à la durée de vie de plus en plus courte, le *time-to-market* revêt une importance sans cesse croissante : les circuits doivent être développés rapidement et être fonctionnels le plus tôt possible dans le processus de fabrication, limitant ainsi le nombre d'itérations coûteuses entre les concepteurs du circuit et ses fabricants. Des circuits de plus en plus flexibles incorporant un léger degré de liberté par rapport à leurs spécifications initiales sont donc nécessaires pour pouvoir corriger d'éventuels défauts à n'importe quel moment durant le développement.

Le concepteur doit donc effectuer un compromis entre la haute performance et la faible consommation des circuits câblés d'une part et la flexibilité et la facilité de développement des

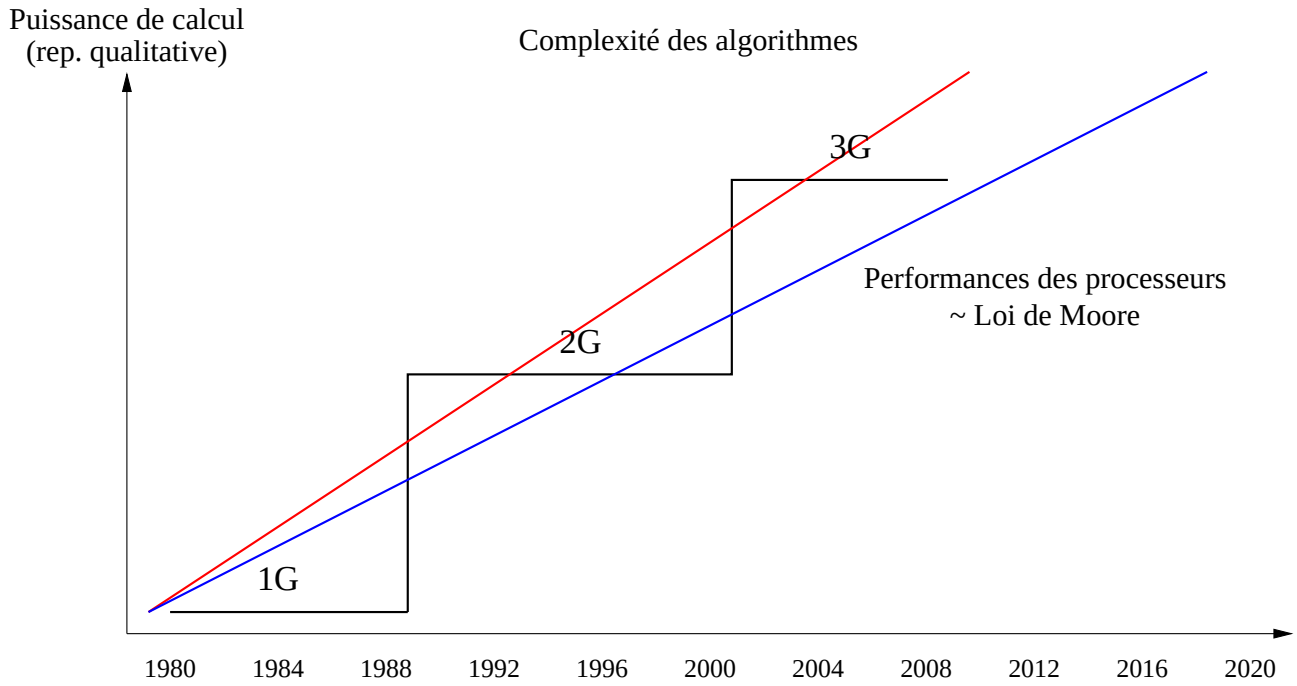


FIG. III.1 – Evolutions comparées des complexités algorithmiques et des puissances de calcul

architectures câblés de type MCU (microcontôleurs) ou DSP (processeurs de signaux) d'autre part, comme indiqué figure III.2. Le degré de performance requis par les spécifications d'un circuit destiné à l'embarqué est tel que le concepteur est obligé de recourir à des blocs de logique câblée en plus du processeur initialement prévu, gagnant ainsi en performances ce qu'il va perdre en flexibilité.

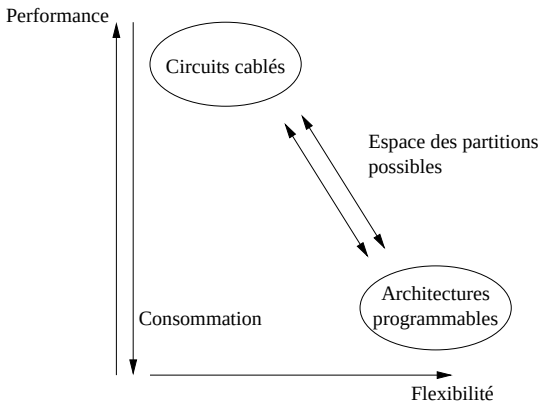


FIG. III.2 – Délimitation de l'espace des partitionnements selon les performances requises, la consommation d'énergie et la flexibilité offerte

On aboutit ainsi à un nouveau type de circuit essayant de réunir le meilleur des deux mondes, à savoir la flexibilité des architectures programmables et le niveau de performance des architectures câblées. Ce type de circuit est appelé ASSP, pour Application Specific Standard Product. A travers le nom même de ces circuits transparaît le désir contradictoire des industriels de

disposer de circuits spécifiques aux applications ciblées tout en restant suffisamment standardisés pour offrir un environnement de développement cohérent aux concepteurs des équipements devant embarquer ces circuits.

III.1.1.2 Hétérogénéité des architectures

Un autre facteur ayant largement contribué à l'évolution du métier de concepteur et des flots de conception utilisés est l'hétérogénéité des systèmes embarqués actuels, désormais appelés *System On Chip* (SOC), ou systèmes sur une puce. Les différentes organes du système, auparavant éclatés en plusieurs composants montés sur une carte, sont aujourd'hui intégrés sur *une seule* puce, diminuant ainsi les temps de circulation des données sur les bus les interconnectant, et diminuant du même coup la consommation du système, point crucial pour le domaine de l'électronique embarquée.

Les SOC développés aujourd'hui peuvent embarquer en un ou plusieurs exemplaires les organes suivants :

- un cœur de processeur généraliste de type ARM7, ARM9 ou PowerPC exécutant le code de contrôle de l'application embarquée et assurant souvent le contrôle général de l'application,
- un cœur de processeur spécialisé en traitement du signal (DSP) pour les applications à forte composante signal ou image dont l'exécution sur un processeur généraliste est irréaliste,
- un ou des accélérateurs/coprocesseurs matériels, réalisés en logique câblée, dont le rôle est de décharger les cœurs de processeur des opérations trop complexes ou très répétitives telles que les transformées de Fourier rapides (FFT), les multiplications-accumulations complexes, les transformées discrètes en cosinus directes et inverses (DCT et IDCT) utilisées en compression/décompression d'image, les estimations de mouvement utilisées en compression vidéo ...
- des périphériques divers gérant des liaisons par port série (UART), des bus d'extension permettant la connexion à des blocs de mémoire externes au SOC, des horloges/compteurs comandables par l'utilisateur, un contrôleur d'interruptions ...

Le concepteur de système embarqué doit déterminer une stratégie de partitionnement et effectuer un choix parmi un grand nombre de solutions, sans forcément disposer d'informations sur les performances de telle ou telle combinaison d'organes de calcul.

On conçoit donc que le développement conjoint d'une telle architecture et d'une application s'y exécutant soit d'autant plus difficile à mener à bien à la première tentative que le nombre et surtout la variété des composants embarqués sont grands.

III.1.1.3 Multiplicité des langages et des environnements de développement

Le premier problème qui se pose lors de la conception d'une architecture hétérogène est celui de la multiplicité des langages et des environnements de développement. En effet, les langages utilisés en développement logiciel, traditionnellement le C et le C++, et les langages utilisés en développement matériel, traditionnellement VHDL[1] de ce côté-ci de l'Atlantique et

[1]VHSIC Hardware Description Language, Langage de Description Matérielle de VHSIC, VHSIC signifiant Very High Speed Integrated Circuit, Circuit Intégré à Très Grande Vitesse.

Verilog de l'autre côté, n'ont rien en commun, et ne sont en général pas maîtrisés par les mêmes personnes. De plus, écrire des programmes et concevoir des blocs de logique câblée étant des métiers relativement dissemblables, ces deux tâches sont généralement confiées à deux équipes différentes travaillant avec des outils différents et des méthodologies différentes.

Les temps de simulation endurés par les concepteurs de la partie matérielle étant en général largement supérieurs à ceux auxquels doit faire face l'équipe responsable du logiciel, cette partie est finalisée en premier, et n'est confiée aux développeurs de la partie logicielle de l'application que dans les derniers stages de sa conception, voire une fois fabriquée, pour faire tourner la partie logicielle directement sur la partie matérielle finale. Ce flot est représenté figure III.3. Si, après livraison des prototypes à l'équipe logicielle, des incohérences entre les spécifications de la partie matérielle et son fonctionnement sont découvertes, la contrainte du time-to-market et le coût éventuel d'une seconde itération entre les concepteurs du matériel et d'une seconde fabrication de prototype font que cette solution n'est adoptée que dans le cas où les erreurs détectées ne peuvent pas être d'abord *contournées* par la partie logicielle. Lorsqu'une erreur est découverte au sein de la partie logicielle de l'application, la relative insignifiance de la durée d'une itération au sein de l'équipe responsable du logiciel fait que le problème est beaucoup plus facile à résoudre : l'erreur est recherchée puis corrigée si possible.

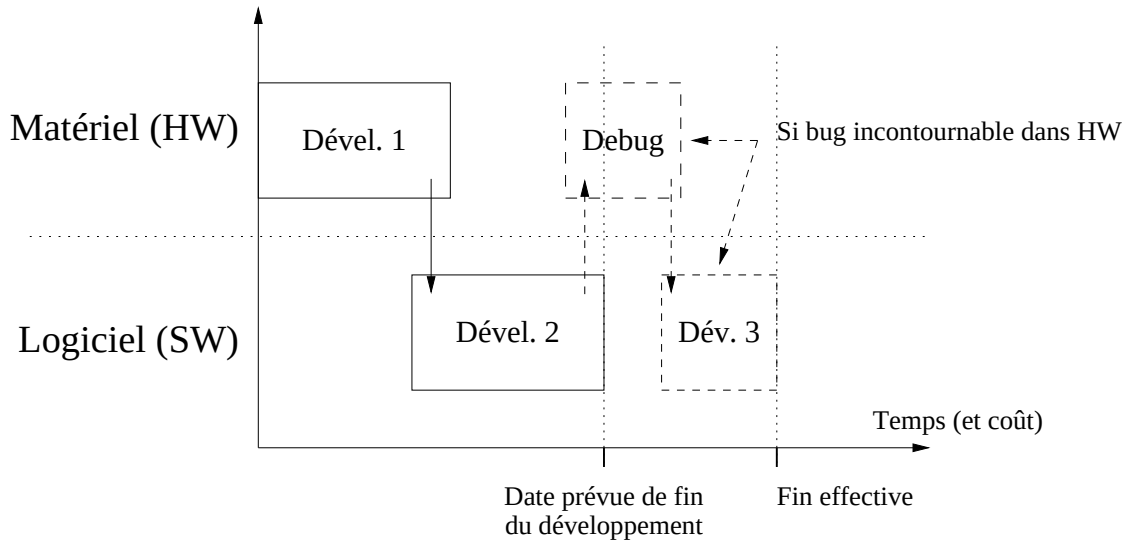


FIG. III.3 – Séquentialité des développements matériels puis logiciels

III.1.1.4 Coût de la vérification

La conséquence principale de l'augmentation de la complexité des équipements microélectroniques et de l'hétérogénéité des architectures est l'augmentation phénoménale du *coût de la vérification*. En effet, il devient de plus en plus difficile et surtout de plus en plus long de simuler une architecture hybride dans son ensemble et à plus forte raison de valider son fonctionnement de manière exhaustive.

Des simulateurs existent pour les parties logicielles et pour les parties matérielles. Pour les parties logicielles, ce sont les simulateurs de jeu d'instruction (ISS, pour Instruction Set

Simulator). Ils existent en général en deux versions pour chaque cœur de processeur, l'une appelée "instruction- accurate", ou "bit-accurate", exacte à l'instruction près et au bit près, qui permet de valider *fonctionnellement* le code produit, mais ne respecte pas *le caractère temporel* de l'exécution du code sur la cible, et l'autre, appelée "cycle-accurate", exacte au cycle près, qui reproduit *les temps d'exécution* de chaque instruction tournant sur la cible. Les simulateurs cycle-accurate sont nettement plus lents que les simulateurs instruction-accurate de par la nécessité de garder une trace logicielle du temps simulé.

Pour les parties matérielles, il existe des simulateurs de VHDL/Verilog chez tous les vendeurs d'outils de conception. Ces outils permettent d'observer les variations au cours du temps de tous les signaux présents à l'intérieur de l'opérateur en cours de développement, d'y appliquer divers stimuli, d'enregistrer et de rejouer des suites de manipulations pour ne pas avoir à répéter à la main les manœuvres d'initialisation, par exemple. La vitesse de ces simulateurs dépend principalement de la taille du design en cours d'élaboration, ainsi bien sûr que de la puissance de la station de travail simulant le circuit.

Les bancs de test des parties logicielles et matérielles ne sont à priori pas écrits dans le même langage, et ne s'adressent pas aux mêmes types de simulateurs. Comment dans ces conditions simuler la *totalité* de l'architecture ? La brique technologique permettant une telle simulation concurrente des parties logicielle et matérielle est appelée *moteur de cosimulation*. Le moteur de cosimulation va contrôler l'ISS *et* le simulateur de VHDL, en assurant la cohérence des horloges rythmant le fonctionnement du(des) processeur(s) et des parties matérielles, et en agissant comme une sorte d'interface traduisant les requêtes émises par un simulateur en stimuli compréhensibles par l'autre et vice-versa. Le problème de ce type de simulation conjointe du matériel et du logiciel est que la vitesse d'une telle cosimulation est asservie à la vitesse du plus lent des simulateurs mis en œuvre, ce qui rend les simulations de scénarios sophistiqués passablement coûteuses. De plus, il peut être nécessaire de réécrire certaines parties des bancs de test utilisés afin de prendre en compte les particularités de la cosimulation, ce qui augmente encore l'effort requis.

Du fait de son coût et malgré les bénéfices qu'on peut en tirer, la vérification n'est pas incluse dans le cycle de développement, mais plutôt effectuée à la fin du cycle, retardant ainsi la détection d'erreurs éventuelles que seule la simulation du système dans sa totalité peut mettre à jour.

III.1.2 Les apports des nouveaux flots de conception

Les années 90 ont vu naître des flots de conception novateurs, qui présentent plusieurs avantages. Le premier est bien évidemment la possibilité offerte par ces nouveaux outils de gérer la création de designs de tailles largement supérieures, dépassant ainsi d'une manière logique leurs prédécesseurs. Mais ces outils présentent aussi un caractère plus ambitieux dans la mesure où les méthodologies qui leur sont associées dépassent le cadre de la stricte conception de circuits intégrés pour donner naissance à une nouvelle discipline : la *conception au niveau système*, ou SLD, pour System-Level Design. Les principales caractéristiques partagées par la plupart des outils de design au niveau système sont les suivantes :

- utilisation d'un langage *commun*, souvent dérivé du C ou du C++, pour décrire les parties matérielle *et* logicielle du système lors des premières phases de la conception,
- amélioration du confort du concepteur lors des phases d'exploration architecturale et de

partitionnement à l'aide d'une description du système orientée « bloc » et de la possibilité d'évaluer *qualitativement* les performances du système en fonction de la stratégie de partitionnement en cours d'examen,
- automatisation partielle du processus de *synthèse* des interfaces entre les domaines logiciel et matériel, même dans le cas de systèmes multi-cœurs,
- possibilité d'intégrer sous forme exécutable des blocs déjà développés et validés (aussi appelés IP, pour *Intellectual Property*), éventuellement par une tierce partie,
- cosimulation au sens large, multi-cœur, multi-langage, multi-niveau d'abstraction, facilitant l'exercice de la *vérification* le long du flot.

III.1.2.1 Unicité du langage de description primaire du système

La majorité des flots de codesign actuels utilisent le *même* langage pour décrire les parties logicielle et matérielle du système, au moins lors des phases de description préliminaire du système. De cette manière, il est possible pour des personnes n'étant ni des spécialistes du logiciel ni des spécialistes du matériel de concevoir les premières versions du système sans avoir recours tout de suite à deux équipes distinctes. Des spécifications exécutables peuvent ainsi rapidement être atteintes, permettant ensuite à des équipes de spécialistes de prendre le relais.

Les langages utilisés lors de cette ébauche du système sont en majeure partie des dérivés du C ou du C++, auxquels on a ajouté les constructions nécessaires à la description de blocs *interagissant* les uns avec les autres (ports d'entrée/sortie, connexions inter-blocs), à la *hiérarchisation* du système (possiblité d'encapsuler des blocs au sein d'un macro-bloc), à l'exécution *concurrente* des portions de code décrivant le comportement des blocs composant le système, et finalement à l'utilisation de *signaux* semblables à ceux utilisés dans les langages de description de matériel classiques (affectation retardée, introduction de registres ...).

Auparavant, ces langages étaient des langages propriétaires. Chaque compagnie du marché des outils de CAO avait développé son propre langage de design au niveau système et vendait l'outil capable de simuler les systèmes décrits dans ce langage et uniquement dans celui-là. Changer d'outil de codesign *pendant* le développement d'un circuit était alors inenvisageable, le changement se traduisant inévitablement par une réécriture complète du système dans un autre langage. Il était alors difficile de comparer les outils entre eux, chacun s'accompagnant de son propre langage.

En 1999 se forme alors l'OSCI[2], un consortium composé des principaux acteurs du marché de la conception microélectronique, dans le but de promouvoir l'utilisation d'un nouveau langage de description : le SystemC [Swa01]. Ce langage se distingue des autres par le fait qu'il est compréhensible par n'importe quel compilateur C++, et qu'il est disponible en *open-source*. Le code source des classes C++ qui composent SystemC, ainsi que celui du moteur permettant la simulation d'un système décrit en SystemC peuvent être téléchargés et utilisés gratuitement. De la même manière que Sun et le langage Java, l'OSCI espère ainsi établir un standard *de facto* et déplacer la compétition sur un autre terrain, celui des outils, désormais tous interopérables, plutôt que sur celui des langages. En retour, les membres du consortium s'engagent à ne pas garder secrètes les modifications qu'ils pourraient apporter aux classes SystemC, mais plutôt à les intégrer au langage lui-même, après approbation préalable des autres membres.

[2]Open SystemC Initiative

SystemC est aujourd'hui disponible en version 2.0, et couvre une large gamme de fonctionnalités :

- SystemC permet le développement rapide de spécifications systèmes complexes et leur simulation,
- ces spécifications peuvent être affinées jusqu'à l'obtention d'implémentaions hybrides logicielles / matèrielles,
- il est possible de décrire les parties matérielles jusqu'au niveau RT (Register Transfer), offrant ainsi la même granularité que VHDL ou Verilog,
- des types de données complexes tels que les nombres en virgule fixe sont utilisables plus facilement qu'avec VHDL, Verilog, ou C
- la réutilisation de la vaste quantité de code C/C++ existant est bien entendu possible.

L'obtention d'une spécification exécutable et *rapidement simulable* du système est donc facilitée par l'utilisation d'un seul et même langage. L'avenir dira si SystemC réussira à s'imposer comme standard *de facto* ou si d'autres tentatives seront nécessaires.

III.1.2.2 Aide à l'exploration architecturale et au partitionnement

Une des premières décisions à prendre lors du développement d'une architecture hybride devant accueillir une application est le *partitionnement* de l'application, c'est-à-dire le découpage de l'application en blocs logiciels s'exécutant sur une architecture programmable de type processeur et en blocs matériels réalisés en logique câblée. On appelle *exploration architecturale* la démarche consistant à comparer divers partitionnements selon différents critères avant de n'en retenir qu'un seul.

Il est crucial d'effectuer ce choix le plus tôt possible dans le cycle de développement, car plus les différents blocs composant le système sont proches de leur implémentation finale, qu'elle soit logicielle ou matérielle, plus ils sont décrits finement, et plus leurs *interconnexions* sont décrites finement elles-aussi. Déplacer une tâche du domaine matériel au domaine logiciel ou inversement prend alors énormément de temps, et nécessite de gros efforts de redéveloppement. Par contre, le fait de tester diverses stratégies de partitionnement relativement tôt minimise le coût de chaque essai, et permet donc de tester un plus grand nombre de solutions dans un laps de temps plus court, au prix d'une précision un peu plus faible lors de l'évaluation des critères comparant les partitionnements entre eux, du fait de l'état non finalisé de la description des blocs du système.

Bien qu'il n'existe à l'heure actuelle aucun outil de partitionnement complètement automatique, certains outils de codesign peuvent aujourd'hui proposer une mesure *qualitative* de la pertinence d'une partition parmi d'autres, l'interprétation de cette mesure étant naturellement à confier au concepteur. Ainsi, en partant d'une description des blocs du système, de leurs éventuelles interconnexions et de leur comportement, ces outils sont capables d'estimer grossièrement le taux d'activité de certains canaux de communication inter-blocs, ce qui peut être relié à des bandes passantes de blocs de mémoire ou à des fréquences d'interruptions pour des architectures programmables ...

De plus, dans le cas de systèmes multi-cœurs, il est possible d'observer l'impact de la migration d'une tâche d'un cœur à un autre sur ces taux d'occupation de bus, tout comme l'impact du déport d'une fonction d'un cœur de processeur du système à un bloc de logique câblée.

III.1.2.3 Synthèse automatique des interfaces

Une fonction nécessaire à l'exploration architecturale mais aussi potentiellement déterminante en ce qui concerne la quantité de code à écrire est la *synthèse automatique d'interface*.

L'exemple le plus simple pour en illustrer l'utilité est celui d'un bloc de logique câblée devant communiquer des données à un cœur de processeur. Dans le cas où c'est l'opérateur câblé qui va initier la communication, le scénario courant, illustré figure III.4 est le suivant :

1. l'opérateur câblé, connecté au contrôleur d'interruptions (ITC, pour *Interrupt Controller*) du processeur, va émettre une requête d'interruption,
2. le contrôleur d'interruption va identifier la requête en question et en déterminer la priorité,
3. si la priorité de l'interruption arrivante est plus élevée que celle de la tâche actuellement exécutée par le processeur, l'ITC va interrompre le processeur pour lui faire exécuter une routine spécifique, la routine de service d'interruption (ISR, pour *Interrupt Service Routine*),
4. cette routine d'interruption va aller lire à une adresse mémoire particulière la donnée mise à disposition par l'opérateur câblé,
5. la requête de lecture est émise sur le bus du processeur,
6. une fois la requête reçue par le périphérique, la donnée est finalement émise sur le bus et récupérée par l'ISR qui doit la traiter,
7. après l'achèvement du traitement, le processeur retourne à la tâche qu'il exécutait préalablement à l'interruption.

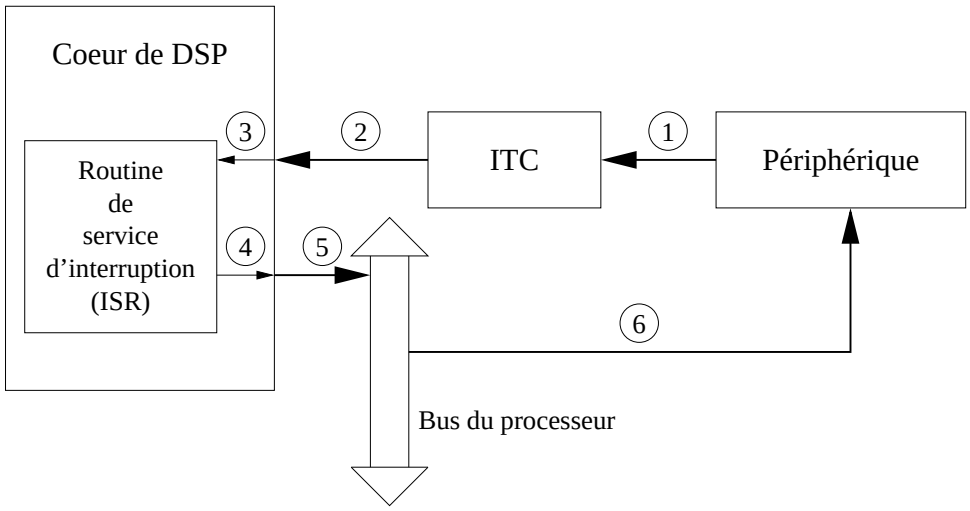


FIG. III.4 – Communication entre un opérateur câblé et un processeur utilisant un mécanisme d'initiation par interruption

En plus de la routine traitant effectivement la donnée reçue, l'utilisateur d'un flot classique devrait écrire le code de la routine d'interruption, déterminer l'adresse où le processeur peut lire la donnée en question, et écrire le code des blocs qui connectent effectivement l'opérateur câblé au contrôleur d'interruption du processeur ainsi qu'à ses bus d'adresse et de données. Ce

travail n'est pas simple, et doit être recommencé à chaque changement de partition, ou à chaque changement de scénario de communication, ce qui peut vite mener à une constante réécriture de l'interface entre le cœur et ses périphériques. Difficile dans ces conditions de choisir sereinement une partition parmi d'autres, sachant que le moindre changement à apporter à la partition en cours nécessite de volumineuses réécritures de portions de code critique. Changer de cœur devient là-aussi une tâche colossale, toute l'interface étant à redévelopper selon des standards différents.

L'apport d'un flot à synthèse automatique d'interface se situe précisément lors de cette phase d'évaluation et de comparaison. Les outils récents utilisent en effet une description des possibilités génériques d'interconnexion d'un cœur de processeur pour générer *au vol* :

- la cartographie des espaces mémoires servant à l'échange de données entre le cœur et les périphériques, couramment appelée la *memory map*,
- les routines d'interruptions adéquates dans le cas de communications initiées par les périphériques, ainsi que les routines de lecture/écriture correspondantes aux accès initiés par le cœur,
- les circuits décodeurs d'adresse surveillant le bus d'adresse pour aiguiller les transactions avec les périphériques vers leurs destinataires,
- la logique nécessaire à l'interconnexion des périphériques au bus de données, dès lors que ceux-ci utilisent un protocole standard (au sens du logiciel effectuant la synthèse de l'interface) pour communiquer.

De cette manière, le déport d'une fonction du cœur au domaine câblé ou inversement ne nécessite plus que quelques changements dans la définition du partitionnement, permettant ainsi de tester rapidement plusieurs solutions sans avoir à réécrire une quelconque ligne du code de l'interface, ni dans le domaine matériel ni dans le domaine logiciel. Cependant, le concepteur garde naturellement un contrôle complet sur les choix par défaut effectués par le logiciel synthétisant l'interface, et est parfaitement libre de modifier la *memory map* ou le scénario d'une communication, sachant que pour l'évaluation rapide de la pertinence d'un partitionnement, les solutions proposées par le logiciel sont en général suffisantes.

Pour pouvoir synthétiser automatiquement ce type d'interface, le logiciel doit avoir une certaine connaissance de la façon dont le cœur à intégrer au design se connecte au monde extèrieur et des diverses manières d'intervenir dans son comportement. Ces données sont généralement présentées sous forme de *plugin* à installer avec le logiciel, en fonction des cœurs dont on désire disposer.

Cette fonctionnalité est aussi utile une fois que le partitionnement a été fixé, car elle permet de garantir une certaine cohérence dans l'écriture des routines d'interruption ainsi que dans le code d'interconnexion des périphériques aux bus divers.

III.1.2.4 Réutilisation d'IP

Bien que la possibilité de réutiliser des blocs déjà développés au sein de projets antérieurs existe depuis longtemps dans les outils de conception, la possibilité d'*exporter* ces blocs vers une tierce partie ou d'en *importer* est clairement innovante.

On trouve depuis l'avènement des outils de codesign des entreprises spécialisées dans la fourniture d'opérateurs spécifiques ou de solutions à des problèmes classiques tels que la correction d'erreurs, par exemple. Ces entreprises ont en effet la possibilité de fournir selon les types de

licence le code source de leur opérateur, alors complètement paramètrable (c'est très rare, le prix d'une telle licence étant souvent supérieur au temps de redéveloppement de l'IP), ou bien d'en livrer un modèle déjà paramétré sous forme compilée de manière à protéger leur propriété intellectuelle.

Pour les contractants, les blocs achetés à de tels fournisseurs d'IP présentent plusieurs avantages :

- leur développement est déjà fait, permettant ainsi d'économiser beaucoup de temps et de diminuer le *time-to-market*,
- ils sont déjà optimisés et ont déjà subi des séries de tests dans plusieurs configurations,
- le support de ces modèles est garanti par le concepteur original de l'IP, déchargeant ainsi le contractant,
- ils coûtent moins cher qu'une équipe interne de concepteurs ayant accompli le même travail, du fait de la capacité du fournisseur à amortir rapidement ses efforts de développement en vendant plusieurs fois son modèle ou des instances déjà paramétrées de son modèle.

Par contre, toute adaptation ou modification mineure du fonctionnement *interne* du bloc IP ne peut se faire sans l'aval du développeur original, qui est souvent le seul à détenir le code source.

Le concepteur d'IP a donc intérêt à proposer une offre la plus paramétrable possible, mais qui présente en même temps un degré d'optimisation suffisant pour rester attractive en termes de performance. On trouve dans les langages de type SystemC les constructions nécessaires pour apporter une certaine généricité aux IP, ainsi que la possibilité d'incorporer des « boîtes noires » au système, dont le comportement n'est pas décrit textuellement, mais géré à l'édition de lien du programme réalisant la simulation.

III.1.2.5 Cosimulation et vérification

L'inconvénient de la cosimulation logicielle-matérielle, déjà introduite plus haut, est la lenteur du procédé dès lors qu'un simulateur VHDL ou Verilog intervient dans la simulation. Le ralentissement occasionné par le passage de la simulation à la *co*-simulation peut atteindre un facteur 10, en fonction bien sûr de la taille de la partie matérielle à simuler, véritable facteur limitant de la cosimulation.

Les méthodologies de conception au niveau système et les langages de description de type SystemC ouvrent cependant la porte à de nouvelles formes de cosimulation.

Cosimulation de plusieurs niveaux de description Les blocs matériels intervenant dans la cosimulation peuvent désormais être décrits de plusieurs manières différentes, parmi lesquelles un mode purement comportemental, où le temps n'intervient ni dans les communications inter-blocs ni dans le comportement des blocs (les échanges inter-blocs se résument à des appels de fonction), un mode semi-comportemental ou semi-temporisé, où seules les communications possèdent un caractère temporel et où l'exécution des traitements au sein d'un bloc est considérée comme instantanée[3], et enfin un mode complètement temporisé, où le comportement des blocs *et* les transactions inter-blocs sont modélisés et simulés au cycle près. Le surcoût introduit par la cosimulation d'un bloc matériel peut donc diminuer fortement, pour peu que celui-ci ne soit pas décrit en mode complètement temporisé.

[3]Cette approche est dite *synchrone* [Hal93], et est adoptée par les langages de type Esterel [Ber00].

L'information apportée par la cosimulation ainsi « simplifiée » est néanmoins pertinente puisqu'elle permet de valider la correction fonctionnelle des échanges entre les domaines logiciel et matériel.

Cosimulation de plusieurs niveaux d'abstraction De la même manière que la multiplicité des niveaux de *description* disponibles, les blocs matériels peuvent être compilés et simulés de deux manières différentes, qui sont les simulations *event-based*, ou basées sur les évènements, et les simulations *cycle-based*, semblables aux simulations VHDL classiques, basées sur les cycles d'horloge. Alors que la simulation cycle-based a recours à un traitement *exhaustif* de l'axe temporel avec une granularité prédéfinie ainsi qu'à un examen de *chaque* signal à *chaque* instant simulé pour déterminer si la situation a évolué au sein des blocs composant le design, consommant ainsi la même quantité de ressources quelle que soit le nombre de signaux ayant évolué au cours de l'instant simulé, la simulation event-based se contente pour accélérer les choses de conserver un inventaire des transitions ayant lieu et de n'évaluer la réaction que des seuls blocs incriminés dans ces mêmes transitions, de manière à ne pas perdre de temps à calculer la réaction des blocs dont aucune entrée n'a récemment changé. Ainsi, si tous les signaux pouvant déclencher une réaction d'un certain bloc restent dans un état ne déclenchant pas de réaction du dit bloc, celui-ci est purement et simplement déconnecté de la simulation jusqu'à ce qu'un évènement extérieur ne vienne le réveiller. L'avantage de cette technique est que les blocs intervenant rarement dans le design ne sont au final que peu simulés, ce qui permet de gagner un temps précieux au prix d'une simulation à la portée réduite. En effet, les blocs dont l'activité est déclenchée en interne ne sont pas des bons candidats à la simulation event-based, qui pourrait éventuellement rater des réactions des blocs, et pervertir ainsi toute la simulation. C'est donc l'utilisation conjointe de ces deux techniques, la simulation event-based pour les blocs à activité marginale, et la simulation cycle-based pour les blocs à activité plus interne, et la répartition des blocs dans ces deux domaines, qui vont permettre d'économiser la ressource temporelle tout en conservant une certaine pertinence dans la simulation.

Cosimulation de plusieurs cœurs Malheureusement, les deux techniques précédentes ne s'appliquent que lorsque les blocs incriminés sont des blocs *matériels*, ou alors, dans le cas de blocs logiciels, lorsque ceux-ci ne tournent pas encore sur les ISS des processeurs auxquels ils sont destinés. Pour la vérification des parties logicielles, les moteurs de cosimulation actuels sont capables de réaliser des simulations multi-cœurs, pour pouvoir simuler conjointement, par exemple, une architecture de type microcontrôleur, une architecture de type DSP, des opérateurs câblés tous rassemblés sous l'égide d'un seul simulateur VHDL ou Verilog, et un banc de test écrit en C s'exécutant sur le processeur de la station hôte de la cosimulation. Ceci dit, plus le nombre de cœurs à simuler est grand, plus la simulation globale ralentit. En effet, non seulement le temps passé dans les ISS augmente considérablement, mais c'est le *poids* de la cosimulation sur la station hôte qui va constituer le facteur limitant des simulations multi-cœurs. C'est le moteur de cosimulation lui-même qui va ralentir le fonctionnement du système avec des changements de contexte importants, l'utilisation d'une quantité de mémoire considérable, et de plus en plus de temps consacré aux *échanges de données* entre les divers simulateurs plus qu'à *l'exécution* de ces simulateurs. Ces simulations sont certes nécessaires, mais il est souhaitable d'avoir recours le plus souvent possible à la simulation mono-cœur comme outil de

vérification généraliste et de ne réserver la simulation multi-cœur qu'à des tests précis et ponctuels ainsi qu'à la vérification finale du système.

Cosimulation de plusieurs langages Utilisant le même principe que la simulation multi-cœur, les moteurs de cosimulation spécialisés, tels CosiMate [Are00], de la société Arexsys, sont capables de faire cohabiter au sein de la même simulation des descriptions de blocs réalisées dans plusieurs langages différents, tels Matlab, SDL, ou d'autres langages plus exotiques, du moins en ce qui concerne la conception microélectronique. L'avantage majeur est la possibilité ainsi offerte de profiter ponctuellement des forces d'un langage qui se prête mal par ailleurs aux contraintes de la simulation telle qu'elle est pratiquée en microélectronique. Ainsi, on peut tirer profit de la simplicité du code Matlab pour effectuer des traitements mathématiques complexes dans un banc de test tout en simulant un circuit connecté à ce banc de test dans un simulateur VHDL. Mixer trop de langages de portées différentes devient problématique pour les mêmes raisons que ci-dessus : au-delà d'une certaine limite, le maintien de la cohérence de l'environnement virtuel au sein duquel cohabitent les simulateurs et les interpréteurs finit par coûter plus cher à la station hôte que l'exécution même de ces outils.

Ces nouvelles formes de cosimulation ont plusieurs avantages :
- elles favorisent la migration du code déjà écrit lors des spécifications exécutables plutôt que sa réécriture, grâce au passage progressif et bloc par bloc du domaine comportemental au domaine temporisé,
- grâce à diverses optimisations du traitement des blocs matériels en simulation, elles rendent la vérification du système (éventuellement la vérification multi-cœur) abordable,
- enfin, elles permettent de ne maintenir qu'*un seul* banc de test utilisable quel que soit la partie du design à vérifier et quelle que soit son niveau de description/d'abstraction.

III.1.2.6 Bilan

Les deux conséquences majeures des avancées réalisées par les outils de design au niveau système présentées ci-dessus sont :
- l'émergence des SOC hétérogènes, véritables systèmes complets pouvant contenir plusieurs cœurs et plusieurs périphériques,
- la parallélisation des travaux des équipes HW et SW lors du développement d'applications sur architectures hétérogènes.

La complexité grandissante des systèmes microélectroniques embarqués répond à des exigences accrues en termes de fonctionnalités fournies par des équipements de plus en plus petits, et de plus en plus connectés. On trouve ainsi des appareils réunissant baladeurs MP3, téléphones portables et PDA, connectables à un PC de plusieurs manières différentes et disposant d'une autonomie suffisante pour que l'aspect "sans fil" soit une véritable fonction du produit et non plus un simple argument de vente sans fondement réel. Ces appareils ayant des durées de vie de plus en plus courtes, leur développement doit s'accélérer et les mises sur le marché des différentes versions se succéder de plus en plus vite. Les architectures hétérogènes composés d'au moins un cœur de processeur et d'un ou plusieurs périphériques correspondent bien à cette structure de développement, permettant de n'optimiser sous forme d'opérateur câblé que les tâches nécessaires, sans pour autant se priver de la flexibilité des architectures programmables. De plus, l'intégration de blocs IP permet d'acquérir des fonctions spécifiques à moindre coût

dans le but d'intégrer rapidement de nouvelles fonctionnalités à un équipement en cours de développement.

L'utilisation d'un seul et même langage pour l'obtention des spécifications exécutables du système permet quant à elle la relative parallélisation des développements logiciels et matériels, ou tout du moins d'éviter la pure séquentialité de ces développements. Les possibilités de (co)vérification sont telles qu'il est maintenant possible de faire évoluer séparément le logiciel et le matériel la majeure partie du temps, et d'effectuer *régulièrement* des simulations jumelées pour évaluer la correction des développements par rapport aux spécifications. La fréquence de ces vérifications doit être choisie de manière à ce que la vérification ne soit pas l'essentiel du travail des concepteurs tout en garantissant que ni le logiciel ni le matériel ne s'écartent des spécifications établies. Cette parallélisation et les vérifications effectuées plus souvent permettent de raccourcir le cycle de développement tout en augmentant la fiabilité du résultat.

La comparaison d'un cycle de développement utilisant les outils présentés ci-dessus et d'un cycle de développement classique semblable à celui présenté figure III.3 est représentée figure III.5.

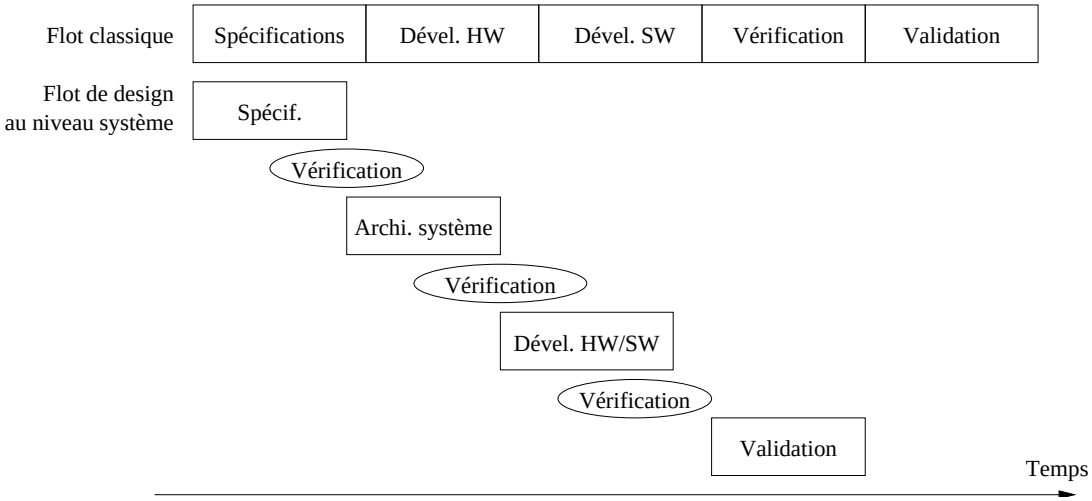


FIG. III.5 – Comparaison d'un flot de conception classique et d'un flot de conception au niveau système utilisant des outils de codesign et de covérification

III.2 Méthodologie innovante de conception au niveau système

La conception au niveau système semblant être l'équivalent de ce que la synthèse logique a été aux années 80, ou de ce que la synthèse physique a été à la fin des années 90, à savoir une sorte de Graal de la microélectronique, les vendeurs d'outils de CAO ont tous développé leur propre outil pour répondre aux problèmes posés ci-dessus, de manière à combler le fossé existant entre les méthodologies de conception classiques et le flot « idéal » de conception au niveau système. On trouve dans [Rou01, chapitre B, section III] un récapitulatif des outils de design au niveau système, parmi lesquels N2C [CoW01] (Napkin To Chip), de CoWare, Inc. (`www.coware.com`), VCC

[Cad01b] (Virtual Component Co-Design), de Cadence Design Systems (`www.cadence.com`), CoCentric System Studio [Syn00a], de Synopsys (`www.synopsys.com`), ou encore Visual Elite [Inn01], d'Innoveda (`www.innoveda.com`).

Il a été mentionné plus haut qu'en général, pour supporter un type de cœur ou un autre, l'outil a besoin d'une caractérisation de ce cœur et de ses possiblités d'interconnexion. Pour des raisons de disponibilité et de compatibilité, c'est l'environnement CoWare N2C qui a été retenu, celui-ci étant le seul à supporter le cœur de DSP étudié par le laboratoire au début de ces travaux, à savoir le cœur ST100 de ST Microelectronics [STM99].

La méthodologie proposée, illustrée figure III.6, repose sur l'introduction *progressive* de la caractérisation temporelle du comportement du système. Le système est d'abord simulé au niveau *transactionnel*, sans aucune considération temporelle. Les différents cœurs de processeur utilisés sont ensuite introduits au sein de l'architecture développée. La génération des diverses interfaces est alors prise en charge par l'outil de codesign utilisé, ici l'outil N2C, de manière semi-automatisée. Grâce à cette prise en charge de la synthèse d'interface, le concepteur peut explorer plus rapidement l'espace des partitionnements possibles et aboutir en un temps réduit au partitionnement final de l'architecture développée. Dans un premier temps, seul le fonctionnement de la partie *logicielle* est temporisé. Une fois le partitionnement finalisé, le concepteur peut procéder à la temporisation des parties *matérielles* du système, en les décrivant à un niveau d'abstraction moins élevé. Tout au long de cette démarche, diverses formes de cosimulation (cosimulation logicielle-matérielle et cosimulation de différents niveaux d'abstraction principalement) sont utilisées pour évaluer la pertinence des décisions architecturales effectuées.

Avant de présenter le fonctionnement de l'outil N2C et son intégration à un flot de conception au niveau système, il est intéressant de préciser quels sont les développements devant déjà être effectués *avant* l'utilisation d'une telle méthodologie. Les points faibles de la méthodologie proposée sont ensuite détaillés dans le cadre d'une application à forte composante « signal ».

III.2.1 Entrées requises préalablement à la phase de codesign

Le codesign étant une démarche pour *implémenter* un algorithme donné sur une architecture a priori inconnue, et non pas pour *mettre au point* un algorithme, il y a deux prérequis au codesign proprement dit :

- l'algorithme lui-même, déjà mis au point, exempt de bugs et correct fonctionnellement, si possible écrit de manière à calculer avec des nombres représentés en précision finie (ce qui implique un certain nombre d'hypothèses sur la largeur disponible du chemin de données),
- une étude de complexité *au sens large*, qui doit fournir au moins des indications quant aux nombres d'opérations requises par les diverses phases de l'algorithme, et au mieux les tailles des espaces de mémoire nécessaires au stockage des données manipulées par l'algorithme, ou encore des informations sur le rythme des transferts entre les divers blocs envisagés pour le découpage de l'algorithme.

III.2.1.1 Simulation du flot de données en précision finie

C'est naturellement une étape indispensable car elle fournit l'algorithme qui va faire l'objet de la démarche de codesign par la suite. Le déroulement d'une méthodologie de codesign étant par nature une tâche relativement complexe, il est souhaitable que l'algorithme qui en fait

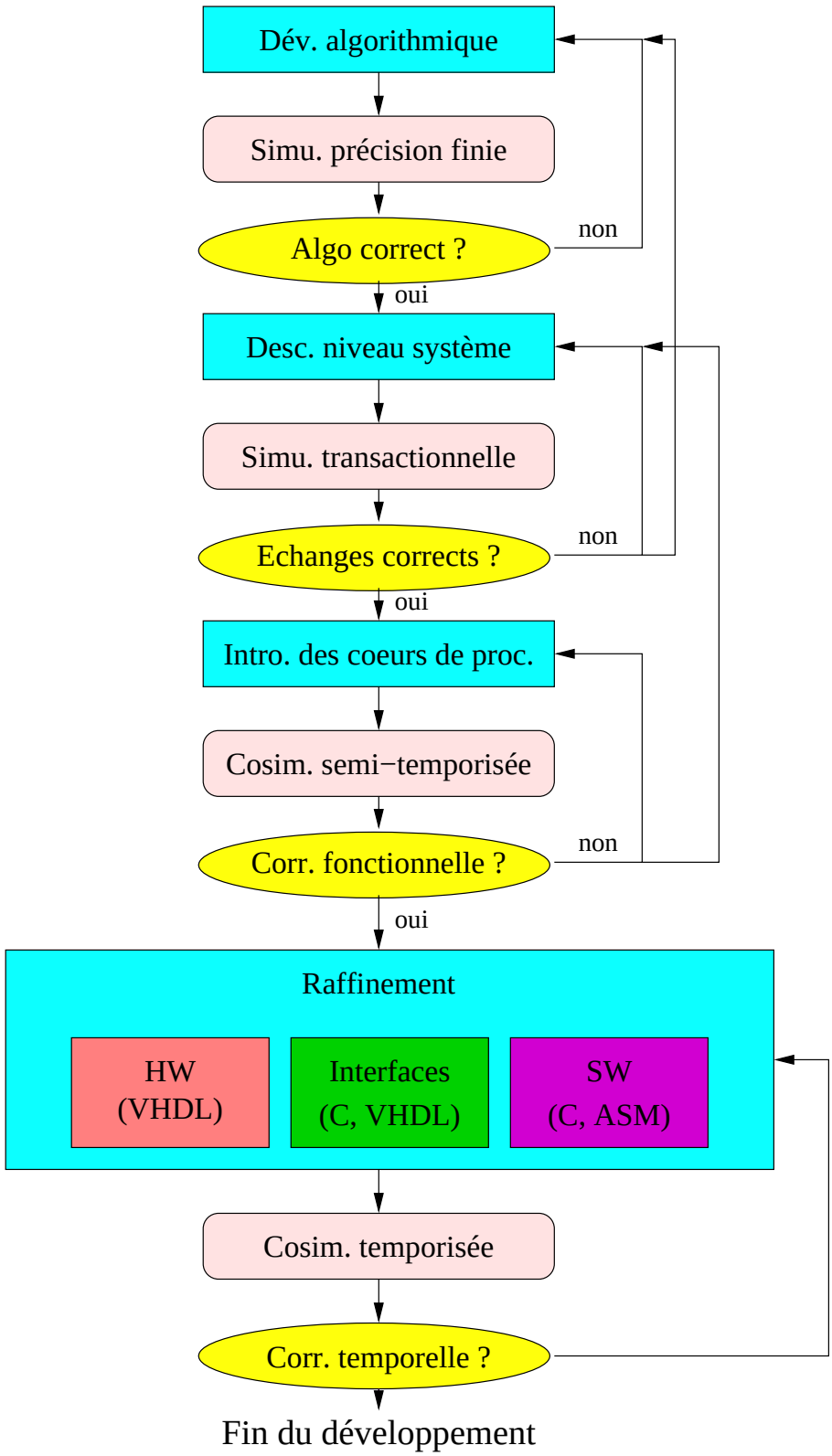


FIG. III.6 – Flot de conception au niveau système

l'objet soit finalisé avant le codesign proprement dit.

Les outils de développement algorithmiques sont légion : MATLAB [The00a] / Simulink [The00b], de The Mathworks (`www.mathworks.com`), COSSAP [Syn], de Synopsys, en cours de dépréciation au profit de CoCentric System Studio [Syn00a], SPW [Cad01a] (Signal Processing Worksystem), de Cadence, Mustig [Gré], de Gresilog (`www.gresilog.com`), ou encore System View [Ela01], de la société Elanix (`www.elanix.com`), pour les principaux. Ces outils ont pour la plupart une entrée graphique qui permet de développer l'algorithme voulu en connectant des « boîtes » les unes aux autres et, ce qui est plus important, offrent tous la possibilité d'encapsuler du code C (ou autre) au sein d'un bloc connecté aux autres, pour peu que le code incorporé respecte certaines conventions d'interfaçage dépendant du logiciel hôte.

Cette possibilité d'encapsulation est fondamentale pour plusieurs raisons :

1. elle permet de réaliser certains traitements dans un langage autre que celui supporté par l'outil, en s'affranchissant ainsi des limites éventuelles exhibées par un outil, ce qui peut s'avérer particulièrement utile pour des outils n'acceptant qu'une entrée *graphique* du système,
2. elle autorise la réutilisation de code déjà écrit autrement qu'en traduisant les fonctions éxecutées d'un langage dans un autre, au prix du développement de l'interface entre la routine déjà écrite et le logiciel hôte, processus en général bien documenté et d'une complexité raisonnable,
3. bien exploitée, elle confère une certaine indépendance vis-à-vis du flot de design utilisé par la suite ; en ne laissant que les fonctions « utilitaires » sous la responsabilité du logiciel hôte et en écrivant du code encapsulé pour le reste, le concepteur n'est pas tributaire d'un flot propriétaire pour le passage de l'algorithme à son implémentation, de la même manière que pendant l'implémentation, un système décrit en SystemC n'est pas lié à un outil particulier, mais peut servir de point de départ à de nombreux flots différents.

Le concepteur a donc globalement intérêt à écrire d'abord les routines critiques dans un langage compilé efficacement, puis à les incorporer dans un environnement graphique qui lui permette de développer rapidement l'algorithme voulu, sans toutefois puiser trop profondément dans les bibliothèques de modules livrés avec le logiciel de développement (souvent sans code source) sous peine de se retrouver avec une quantité importante de code à réécrire lors du passage à l'implémentation. Ce problème ne se pose bien sûr qu'avec des outils fournis par des vendeurs différents et donc incompatibles entre eux, VCC acceptant naturellement en entrée des systèmes développés sous SPW, par exemple ...

La vue du système avec laquelle on travaille lors de la mise au point de l'algorithme doit idéalement être exempte de toute considération architecturale, de manière à ne pas influencer le style dans lequel le code des routines encapsulées est écrit vers une implémentation plutôt qu'une autre. Dans la pratique, lorsque la même personne développe l'algorithme et son implémentation, cet état de parfaite ignorance des considérations architecturales est difficile à atteindre, le concepteur ayant souvent une ou deux idées sur le partitionnement ciblé, et écrivant du code destiné à s'insérer dans ce partitionnement plutôt qu'un autre.

Un point important qui détermine fortement la réutilisabilité potentielle du code écrit pour ces simulations est le choix de la *représentation* des nombres qui y sont manipulés. En effet, ce code étant destiné à un embarquement sur cible programmable ou câblée, l'usage des nombres à virgule flottante [IEE85] y est proscrit pour la bonne et simple raison que la manipulation

des nombres à virgule flottante coûte plus cher, que ce soit sur une architecture programmable ou une architecture câblée, que la manipulation de nombres entiers ou à virgule fixe. C'est d'ailleurs pour cela que la majorité des DSP vendus aujourd'hui ne dispose pas de FPU (unité de calcul flottant, Floating Point Unit en anglais). Cette unité est par contre présente dans tous les processeurs de station de travail ou presque, et permet de manipuler les nombres avec une précision supérieure à celle atteignable avec les nombres en virgule fixe, ce qui explique le fait que la représentation en virgule flottante soit celle qui est adoptée par défaut dans la plupart des logiciels de simulation. Le lecteur intéressé trouvera dans [Nus88, chapitre 1] une discussion approfondie sur les différentes représentations en virgule fixe et leurs diverses caractéristiques.

Il n'y a malheureusement pas plus de démarche classique de migration d'un algorithme d'une représentation en virgule flottante à l'utilisation d'un format plus facilement digérable par les architectures embarquées d'aujourd'hui que d'outil réalisant ce passage automatiquement. Le concepteur dispose de plusieurs solutions pour faciliter cette transition :

- étudier la dynamique des variables manipulées *le long* du chemin de données, puis fixer la largeur *de départ* des données pour ne pas introduire un bruit de calcul significatif le long de l'algorithme, tout en dimensionnant attentivement les opérateurs intervenant de manière à éviter tout problème de saturation,
- contraindre *dès le départ* la taille des opérateurs du chemin de données, et évaluer les dégradations des performances de l'alghorithme induites par cette réduction de la largeur du chemin de données en recherchant un compromis acceptable.

Quelle que soit la démarche choisie, il est souhaitable d'aboutir à des routines ne manipulant que des nombres entiers, ou par extension des nombres entiers considérés comme des nombres à virgule fixe. Le code ainsi écrit pourra être compilé sans problème majeur vers la plupart des DSP, et pourra servir de base à des opérateurs câblés manipulant des données de taille usuelle, typiquement 8, 16 ou 32 bits.

III.2.1.2 Etude de complexité au sens large

L'objectif de cette étude est de fournir des informations préliminaires qui vont pouvoir *guider* le concepteur dans sa recherche d'un partitionnement de l'algorithme développé sur une architecture hétérogène.

La première donnée à déterminer est la complexité *calculatoire* de l'algorithme développé, c'est-à-dire le nombre d'opérations arithmétiques élémentaires (additions, multiplications, divisions ...) effectuées par les différentes étapes de l'algorithme. Associée au temps donné à l'algorithme pour accomplir ses calculs, elle est exprimée en M*O*PS, pour Millions d'*Opérations* Par Seconde. Cette première mesure grossière de la charge de calcul nécessaire à l'algorithme peut déjà servir, après application d'un facteur d'expansion, à dimensionner les organes devant effectuer les calculs. En effet, pour une architecture programmable de type DSP, convertir la mesure de la complexité calculatoire en un nombre d'*instructions* par seconde, exprimé en M*I*PS, pour Millions d'*Instructions* Par Seconde, est loin d'être aisé. Selon les cœurs envisagés, leur architecture, la qualité des outils de compilation qui y sont associés, et surtout selon le type de code pris en compte, la valeur du rapport entre le nombre d'*instructions* requises par un traitement donné sur le nombre d'opérations effectuées par ce traitement peut atteindre un facteur pouvant varier entre 1.5 et 5 ou 6, suivant les architectures visées et les importances respectives des portions de code dédiées au contrôle et au traitement du signal.

Le nombre de MIPS requis par l'algorithme va donc pouvoir être grossièrement comparé aux performances des cœurs potentiellement intégrables au système, et ainsi effectuer une première passe de sélection parmi ces cœurs, en rejetant ceux qui ne sont pas assez performants. Il est important de souligner le caractère *très approximatif* de cette méthode d'évaluation, qui ne fait appel à aucune compilation effective du code en question ni aucun test des performances du code éventuellement obtenu. Le choix d'un cœur parmi d'autres n'est ici qu'*abordé*.

Il est intéressant d'évaluer dès maintenant la quantité de mémoire nécessaire au bon déroulement de l'algorithme ainsi que le volume et le rythme des échanges entre les opérateurs effectuant les calculs et les bancs de mémoire en stockant les résultats. Cette information, dont l'évaluation n'est nécessairement *plus* exempte de toute considération architecturale, peut conduire à des données telles que la *bande passante* de la mémoire du système, son organisation, ou encore ses interconnexions avec les organes du système. L'ordre de grandeur de cette quantité de mémoire peut aussi aider à déterminer la *proximité* de cette mémoire des cœurs éventuellement impliqués. L'influence du partitionnement envisagé par le concepteur lors du développement de l'algorithme étant ici difficilement négligeable, on s'efforcera de ne pas trop rentrer dans les détails et de considérer ces informations en ne leur accordant qu'importance relative.

Un autre point qui peut s'avérer déterminant pour l'architecture du système est la fréquence de l'exécution à l'algorithme. En effet, des systèmes du type application météo (très forte complexité, mais occurrences des traitements très éloignées les unes des autres) ne se conçoivent pas de la même manière que des systèmes effectuant plus souvent des traitements moins complexes. La fréquence globale des activations de l'algorithme peut aussi guider le concepteur dans le partitionnement qu'il va effectuer, en imposant une *borne supérieure* au temps d'exécution de l'algorithme.

Une fois tout ou partie de ces informations déterminées, le concepteur peut passer au codesign de l'application proprement dit.

III.2.2 Codesign à l'aide de N2C

Les différentes étapes de la méthodologie proposée sont décrites par la suite, et illustrées par l'exemple d'un système de ping-pong constitué de deux blocs s'échangeant une valeur en la modifiant à chaque itération, et ce jusqu'à ce que la valeur échangée soit nulle. L'implémentation de ce système sera effectuée avec l'outil N2C, de CoWare, Inc., et incorporera un cœur de DSP ST100 [STM99] pour les besoins de la démonstration.

Ces choix ont été motivés par les raisons suivantes :

- le cœur de DSP ST100 a été retenu pour des raisons d'évaluation de potentiel technologique, étant un des premiers à avoir été conçus dans le but de pouvoir jouer le rôle d'un microcontrôleur *et* d'un DSP[4],
- N2C était le seul outil supportant le cœur de DSP ST100 au début de ces travaux, tout en prônant une démarche suffisamment originale pour motiver l'intérêt du laboratoire.

Les premières versions de N2C n'acceptaient que les entrées *textuelles* d'une part, et *en*

[4]De plus, un partenariat étroit avec STMicroelectronics nous a facilité l'obtention de la documentation et des outils relatifs à ce coeur de DSP.

CoWareC[5] d'autre part. Les exemples seront donc donnés dans le langage natif de l'outil N2C, le CoWareC, et illustrés par des captures d'écran du Workbench, outil permettant d'examiner et de manipuler une vue *graphique* du système en cours de développement. Cet outil permet depuis peu *l'entrée* graphique du système, et la génération de code source *à partir* de l'entrée graphique.

Un des inconvénients de l'outil N2C est le fait que tout au long de la phase de codesign, une grande latitude est offerte au concepteur quant aux niveaux de description des blocs composant le système. La méthodologie proposée *restreint* donc l'espace des niveaux de simulation possibles à un certain nombre de sous-ensembles correspondant chacun à une vue spécifique du système, guidant ainsi le concepteur lors du processus de migration du système d'une vue uniquement *transactionnelle* à une vue entièrement *temporisée*.

Ces différentes étapes, ainsi que les niveaux de description et les structures du CoWareC associés, sont présentés par la suite.

III.2.2.1 Découpage en blocs fonctionnels

La première étape de la méthodologie proposée est l'abstraction du système envisagé sous la forme de blocs interconnectés réalisant un découpage *fonctionnel* du système. Les blocs composant le système sont caractérisés par leurs entrées, leurs sorties, et leur comportement.

Les *ports* des blocs décrits sont dits *maîtres* lorsqu'ils *initient* la communication, ou *esclaves* lorsqu'ils *répondent* à une communication initiée par un port maître, ces deux possibilités s'excluant naturellement mutuellement. De plus, les ports d'entrée-sortie peuvent véhiculer des *données* typées, qui circulent alors dans une *direction* particulière (entrée, sortie, ou entrée-sortie).

Outre la zone de mémoire locale à chaque bloc destinée à stocker des variables internes et la tâche d'*initialisation* du bloc, exécutée au début de la simulation, seul le *comportement* des blocs est décrit, sans aucune caractérisation temporelle. Cette modélisation purement comportementale prend la forme de deux styles de tâches :

- la ou les tâches de fond, dites *autonomes*, qui s'exécutent en boucle,
- les tâches associées aux ports *esclaves* du bloc, déclenchées en réaction à un évènement survenant sur le port en question.

Ces deux types de tâches peuvent naturellement accéder aux ports *maîtres* du bloc au sein duquel elles sont encapsulées.

D'autre part, il est possible de décrire des blocs sous forme d'agrégats de sous-blocs encapsulés, ce qui permet de hiérarchiser proprement le système. C'est une structure encapsulante de ce type qui est utilisée pour décrire le bloc de plus haut niveau considéré, le bloc « système ».

A ce niveau de description, appelé *UnTimed* (Non Temporisé) par l'outil N2C, seule une simulation *transactionnelle* du fonctionnement du système est effectuée. Les échanges qui ont lieu sur les ports des blocs qui composent le système sont considérés comme instantanés, tout comme les exécutions des tâches associées aux ports esclaves. Les accès aux ports sont simulés avec des RPC (*Remote Procedure Call*, Appel de Procédure Distante), et sont donc *bloquants*. En effet, le contrôle n'est rendu à la tâche appelante qu'une fois la tâche appelée terminée. Il est cependant possible de contourner cette limitation et de rendre certains accès *non bloquants*

[5] CoWare, Inc. étant un des membres fondateurs de l'OSCI, comité responsable de la spécification et de l'implémentation de référence de SystemC, N2C supporte depuis peu en entrée les systèmes écrits en SystemC.

en introduisant une machine d'état tournant en tâche de fond dont l'état est modifié par une tâche esclave. Le comportement du bloc est alors semblable à celui d'un processeur qui sert une requête d'interruption à ceci près que la tâche de fond continue de s'exécuter *pendant* le déroulement de la tâche esclave.

Le CoWareC comporte des mots réservés qui servent à décrire les blocs selon la sémantique présentée ci-dessus. Pour les ports, par exemple, le tableau III.1 indique les différents types de ports et leurs connexions possibles. Il est aussi possible de rendre « adressables » les ports véhiculant des données, en leur associant un index semblable à celui utilisé pour accéder aux éléments d'un tableau en C.

Direction	Initiateur	Récepteur
Sans (pas de donnée véhiculée)	*master*	*slave*
Entrée	*inmaster*	*outslave*
Sortie	*outmaster*	*inslave*
Entrée-sortie	*inoutmaster*	*inoutslave*

TAB. III.1 – Compatibilité des types de port possibles en CoWareC

La description en CoWareC des blocs composant le système de ping-pong mentionné ci-dessus est composée des portions de code suivantes :

– l'instanciation du bloc et la déclaration de ses ports d'entrée-sortie caractérisés par leur nom, leur type, et leur direction,
```
blockdef bloc1(
    bool[32] in  : inslave,
    bool[32] out : outmaster )
```
– l'encapsulation comportementale ou non-temporisée (`C` est le langage, `UT` le niveau d'abstraction, pour Untimed, et `version` permet de faire cohabiter au sein d'un seul fichier `.cwr` plusieurs versions d'une même encapsulation),
```
begin
    C.UT.version {
```
– la déclaration au sein du `context` des variables locales au bloc,
```
        context {
            int new_val, val;
        };
```
– la tâche d'initialisation, appelée `thread construct`, qui est exécutée avant le début de la simulation proprement dite,
```
        thread construct {
            val = 1;
            new_val = 10;
        };
```
– la tâche esclave associée au port `in`, à l'intérieur de laquelle on note la *lecture* de la valeur reçue sur le port en question (`new_val = in`),
```
        thread in {
            new_val = in;
            printf("Bloc 1 recoit %d.\n", new_val);
        };
```
– la tâche de fond, qui surveille l'évolution de la valeur reçue avant de la renvoyer incrémentée,
```
        thread {
```

```
                if( val != new_val ){
                    val = new_val;
                    if( val == 0 )
                        exit( 0 );
                    else
                        out = val + 1;
                }
            };
```

– et enfin la clôture de l'encapsulation `C.UT.version` et de la définition du bloc.

```
        };
    end;
```

Les fichiers sources des deux blocs du ping-pong sont :

```
blockdef bloc1(
  bool[32] in  : inslave,
  bool[32] out : outmaster )
begin
  C.UT.version {
    context {
      int new_val, val;
    };
    thread construct {
      val = 1;
      new_val = 10;
    };
    thread in {
      new_val = in;
      printf( "Bloc 1 recoit %d.\n", new_val );
    };
    thread {
      if( val != new_val ){
        val = new_val;
        if( val == 0 ) exit( 0 );
        else out = val + 1;
      }
    };
  };
end;
```

```
blockdef bloc2(
  bool[32] in  : inslave,
  bool[32] out : outmaster )
begin
  C.UT.version {
    context {
      int new_val, val;
    };
    thread construct {
      new_val = 0
      val = 0;
    };
    thread in {
      new_val = in;
      printf( "Bloc 2 recoit %d.\n", new_val );
    };
    thread {
      if( new_val != val ){
        val = new_val;
        out = val - 2;
      }
    };
  };
end;
```

Le fichier source encapsulant les deux blocs déjà décrits et déclarant leurs connexions est quant à lui décrit ci-dessous. On peut noter les deux parties distinctes de bloc `system`, à savoir l'instanciation des blocs utilisés d'une part, et leur inteconnexion à l'aide de canaux de communication nommés d'autre part.

```
// Inclusion des descriptions des blocs composant le système
#include "bloc1.cwr"
#include "bloc2.cwr"
// Définition du système
blockdef system()
begin
    // Declaration d'une structure encapsulante
    STRUCTURE.any.version {
        // Instanciation des blocs utilisés
```

```
        instancedef bloc1 Bloc_1;
        instancedef bloc2 Bloc_2;
        // Création des canaux de communication connectant les blocs
        Bloc_1( b2_vers_b1, b1_vers_b2 );
        Bloc_2( b1_vers_b2, b2_vers_b1 );
    };
end;
```

Une vue graphique du système décrit à l'aide de ces trois fichiers sources est proposée figure III.7, et illustre le formalisme associé à la caractéristique maître-esclave des ports de communication ainsi qu'à la direction des transactions qu'ils supportent.

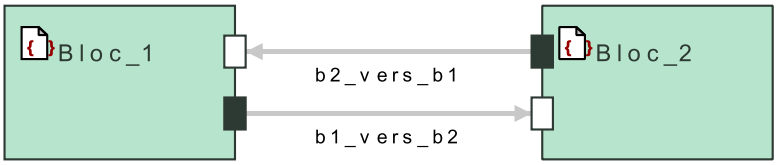


FIG. III.7 – Schéma-bloc du système de ping-pong

A ce niveau d'abstraction, la principale possibilité d'*analyse* du système développé est un diagramme de Gantt illustrant les tâches actives non pas en fonction du temps, qui n'a pas encore été introduit au sein de la simulation, mais en fonction du *nombre d'opérations* effectuées par chaque tâche. Un tel diagramme obtenu en instrumentant le système de ping-pong est présenté figure III.8. Le code C écrit à l'intérieur des tâches de chaque bloc est instrumenté *pendant* la simulation, ce qui permet au moteur de simulation de conserver une trace de chaque type d'opération élémentaire effectuée. Il est ainsi aisé de vérifier que les blocs s'activent bien dans l'ordre voulu d'une part, et que la complexité engendrée par chacun des blocs est *à peu près* conforme à l'estimation qui en a déjà été faite d'autre part. En effet, cette métrique particulière dépend fortement du style dans lequel est écrit le code C, ce qui fait que ce nombre d'opérations ne peut avoir dans l'évalution de la complexité du système qu'une valeur strictement *qualitative*. Il est de plus possible d'observer la succession des tâches exécutées, le passage d'une tâche à une autre étant matérialisé par des flèches symbolisant la transmission du contrôle d'une portion du code à une autre.

Il est de plus possible d'enregistrer les plages de variations de chaque variable instanciée dans un bloc quelconque, ainsi que le nombre de transactions effectuées sur chaque canal de communication inter-bloc, ce qui peut servir à dimensionner des compteurs internes au système ou des bandes passantes de bus.

Le niveau d'abstraction de la description proposé ne sert qu'à valider le découpage qui a été fait de l'algorithme et à s'assurer de la correction *fonctionnelle* du système. Aucune ébauche de partitionnement n'ayant pour l'instant été réalisée, les seules informations que l'on peut déduire des simulations exécutées à ce niveau d'abstraction ont surtout une valeur qualitative et ne peuvent que difficilement être introduites dans un processus *automatisé* d'évaluation d'architecture.

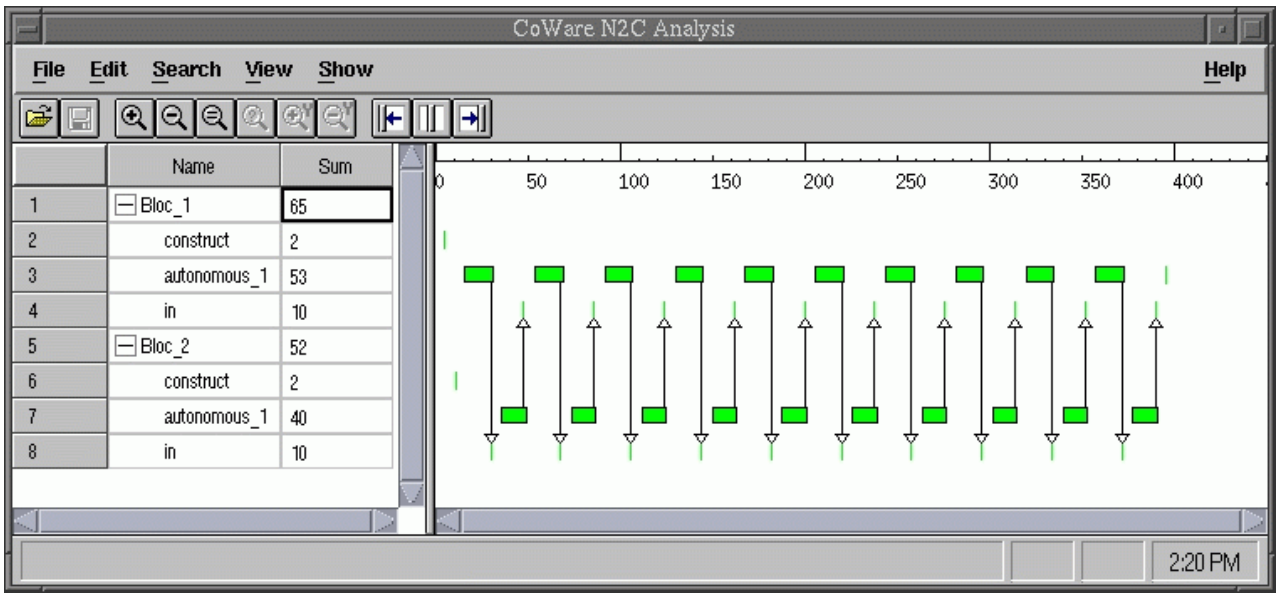


FIG. III.8 – Diagramme de Gantt illustrant l'ordonnancement des tâches au cours de la simulation en fonction du nombre d'opérations instrumentées dans le code

III.2.2.2 Introduction des cœurs et partitionnement

L'étape suivante consiste à introduire au sein du système au moins un cœur de processeur et à effectuer le *partitionnement* proprement dit, c'est-à-dire choisir parmi les blocs décrits à l'étape précédente ceux qui s'exécuteront sur la partie *programmable* du système et ceux qui se matérialiseront sous la forme d'opérateurs câblés.

La principale modification apportée au système par rapport à l'étape précédente est *l'introduction du temps* dans la description du système. Les cœurs de processeur considérés étant en général des circuits *synchrones*, il est nécessaire d'introduire une *horloge* au sein du système, ainsi qu'un signal de remise à zéro (*reset*) servant à déclencher convenablement l'initialisation de ces circuits.

Le système ainsi semi-temporisé se compose d'un cœur de processeur ou de DSP cadencé à une certaine fréquence, sur lequel les opérations mettent un temps *non nul* à s'exécuter, et d'un ensemble d'opérateurs *non temporisés* qui fonctionnent d'une manière semblable à l'étape précédente. La *synthèse d'interface* joue ici un rôle primordial, puisqu'elle va créer automatiquement tous les blocs *logiciels*, comme les routines d'interruption, et *matériels*, comme les arbitres de bus ou les décodeurs d'adresse, nécessaires au bon déroulement des transactions entre le domaine logiciel et le domaine matériel. C'est l'efficacité de ce moteur de synthèse d'interface qui va permettre d'effectuer plusieurs itérations de développement et ainsi de tester plusieurs partitionnement distincts, voire, le processus de synthèse d'interface pouvant être guidé par le concepteur, de tester pour un même partitionnement logiciel-matériel plusieurs possibilités d'interfaçage entre le cœur de processeur et les opérateurs câblés. C'est aussi la synthèse d'interface qui permet de migrer une fonctionnalité d'un domaine à l'autre sans avoir à modifier le code déjà écrit dans de trop grandes proportions. En effet, pour les opérateurs câblés, les accès aux ports d'entrée-sortie se font exactement de la même manière que dans la description non temporisée, le processus de connexion aux bus du processeur étant transparent

pour le concepteur.

D'autre part, les simulations effectuées en mode semi-temporisé forment un compromis suffisamment *rapide* entre la simulation non-temporisée introduite plus haut et la simulation de type RTL, où le fonctionnement *physique* du circuit est simulé, pour réduire le temps d'itération et permettre au concepteur de converger rapidement vers le partitionnement final de l'application.

Les modifications *syntaxiques* à apporter par le concepteur aux *fichiers sources* du système à cette étape sont mineures : des domaines particuliers d'horloge et de reset sont crées d'une part, ce qui permet de ne pas avoir à ajouter des ports aux blocs déjà décrits, et les *générateurs* de ces signaux particuliers sont instanciés d'autre part, de la même manière que le sont les composants usuels. Il existe plusieurs façons d'initialiser ces générateurs, de manière à obtenir des resets actifs à 1 ou à 0, des horloges dont le rapport cyclique est particulier, des horloges sous-multiples d'autres horloges ... Le nouveau fichier système est inclus ci-dessous.

```
// Inclusion des descriptions des blocs composant le système
#include "bloc1.cwr"
#include "bloc2.cwr"
#include "clockreset.cwr"
#include "terminator.cwr"
// Définition du système
blockdef system()
begin
    // Declaration d'une structure encapsulante
    STRUCTURE.any.version {
        // Instanciation des blocs utilisés
        instancedef bloc1 Bloc_1;
        instancedef bloc2 Bloc_2;

        // Instanciation ds générateurs d'horloge et de reset
        instancedef Clock<2>.C.BCA.version clock
        begin
            pragma region {
                clk = SYS_clock;
            };
        end;
        instancedef Reset<37>.C.BCA.version reset
        begin
            pragma region {
                rst = SYS_reset;
            };
        end;

        // Création des canaux de communication connectant les blocs
        Bloc_1( b2_vers_b1, b1_vers_b2 );
        Bloc_2( b1_vers_b2, b2_vers_b1, finsim );
        Terminator( finsim );

        clock();
        reset();
```

```
    };
end;
```

C'est au sein de l'interface utilisateur[6] de N2C que le concepteur va véritablement *créer* les domaines logiciel et matériel de l'architecture développée. La manipulation correspondante est illustrée figure III.9, et le résultat *avant* l'affectation d'un bloc au domaine logiciel et *avant* la synthèse d'interface est représenté figure III.10. La partie gauche de cette figure repsésente le système à son plus haut niveau hiérarchique. Les blocs présents sont les deux blocs s'échangeant les données, les générateurs d'horloge et de reset, un bloc responsable de la fin de la simulation[7], et l'encapsulation du DSP, qui est elle détaillée sur la partie droite de la figure. Dans le cas d'un cœur de DSP ST100, les blocs matériels instanciés dès l'introduction du cœur au sein du système sont une méga-cellule ST100, un contrôleur d'interruption (ITC) et le décodeur d'adresse associé à cet ITC.

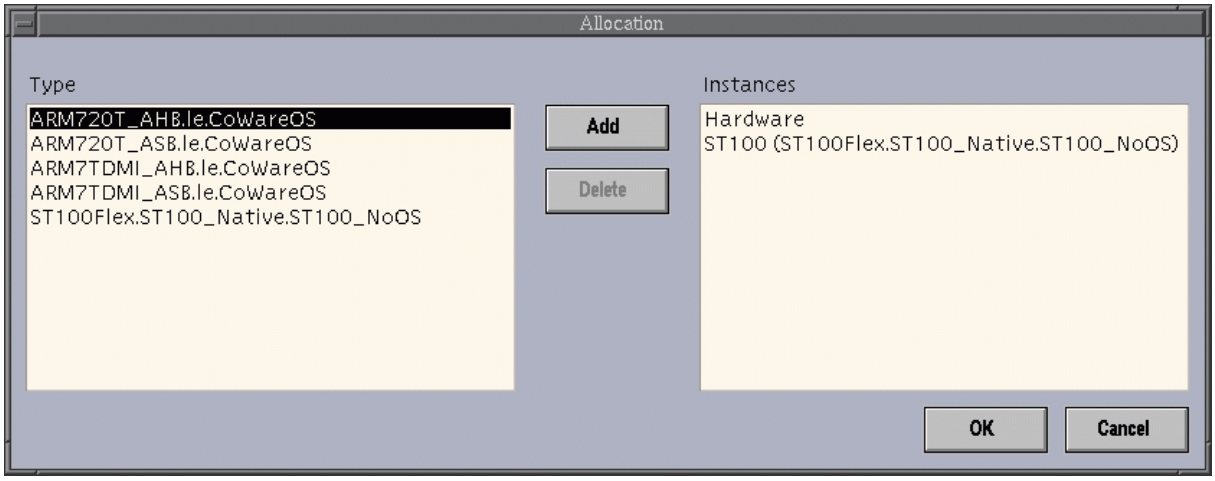


FIG. III.9 – Introduction d'un cœur de processeur au sein de l'architecture développée

Ensuite a lieu l'affectation des blocs composant le système au domaine logiciel ou au domaine matériel et la synthèse proprement dite de l'interface entre les deux domaines. L'affectation des blocs à un cœur de processeur se fait à l'aide de l'interface graphique, et est illustrée figure III.11. Pour chaque bloc, la possibilité est offerte au concepteur de l'affecter au domaine matériel générique ou à chaque cœur de processeur instancié. Le code décrivant le comportement des blocs mappés sur un cœur de processeur est alors traité par la chaîne de compilation du cœur en question, et sera exécuté sur l'ISS concerné lors de la simulation. Le système *après* le mapping du bloc 2 sur le DSP est représenté figure III.12. Le bloc 2 est désormais contenu dans le modèle du ST100, qui se retrouve alors connecté au bloc 1 et au bloc responsable de la fin de la simulation.

La synthèse de l'interface, quant à elle, est illustrée figure III.13. Chaque scénario de communication proposé dispose d'un certain nombre de paramètres qu'il est possible de modifier

[6]Il est naturellement possible d'effectuer ce travail avec des lignes de commande, mais cette fonctionnalité est plutôt destinée à l'*automatisation* du processus qu'à son déroulement manuel.

[7]La présence de ce bloc s'explique par le fait que dans des simulations semi-temporisées faisant intervenir un ISS fonctionnant en mode cycle-accurate, la terminaison de la simulation ne peut provenir que d'un bloc s'exécutant sur la machine hôte, qui ne peut donc pas être mappé sur le cœur de processeur.

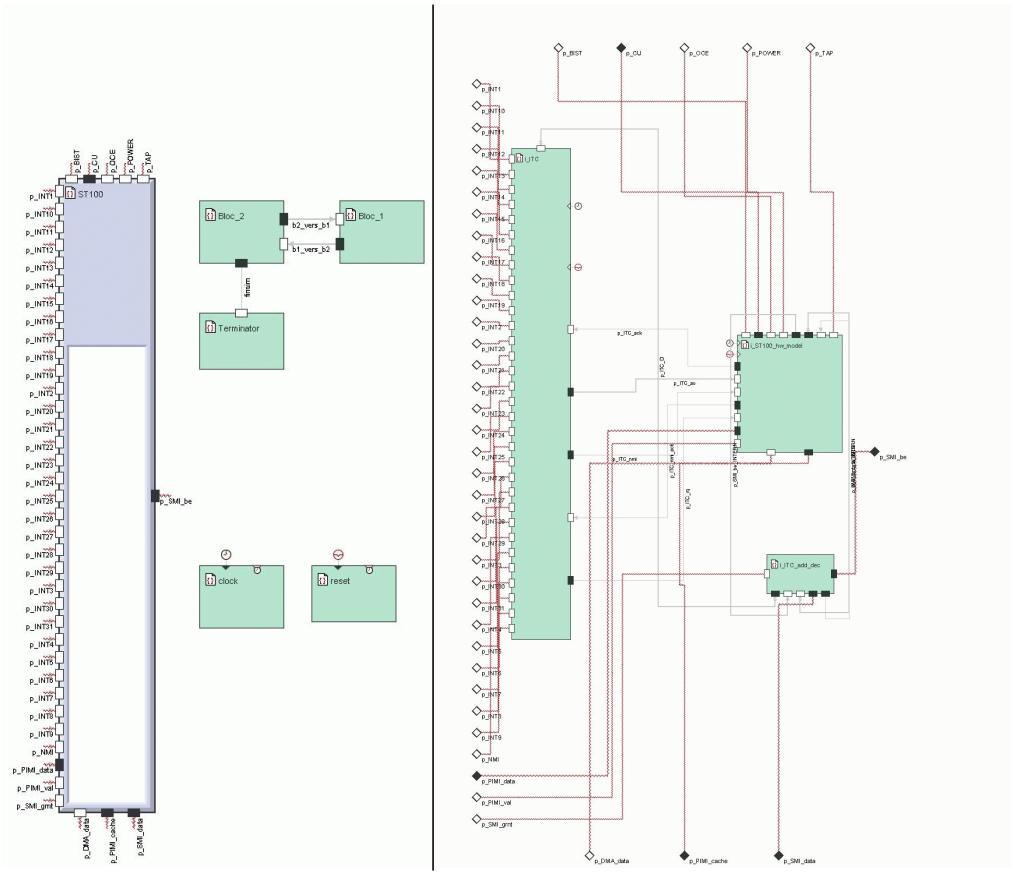


FIG. III.10 – Système du ping-pong après incorporation du cœur de DSP

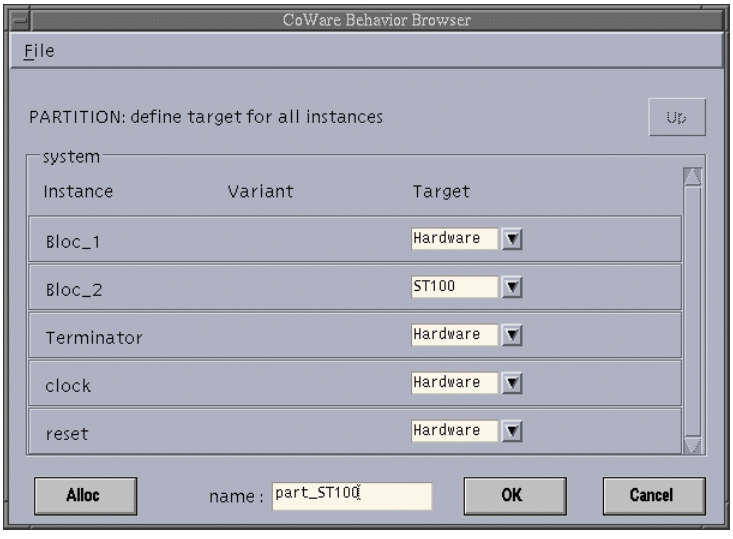


FIG. III.11 – Affectation des blocs aux domaines logiciel et matériel

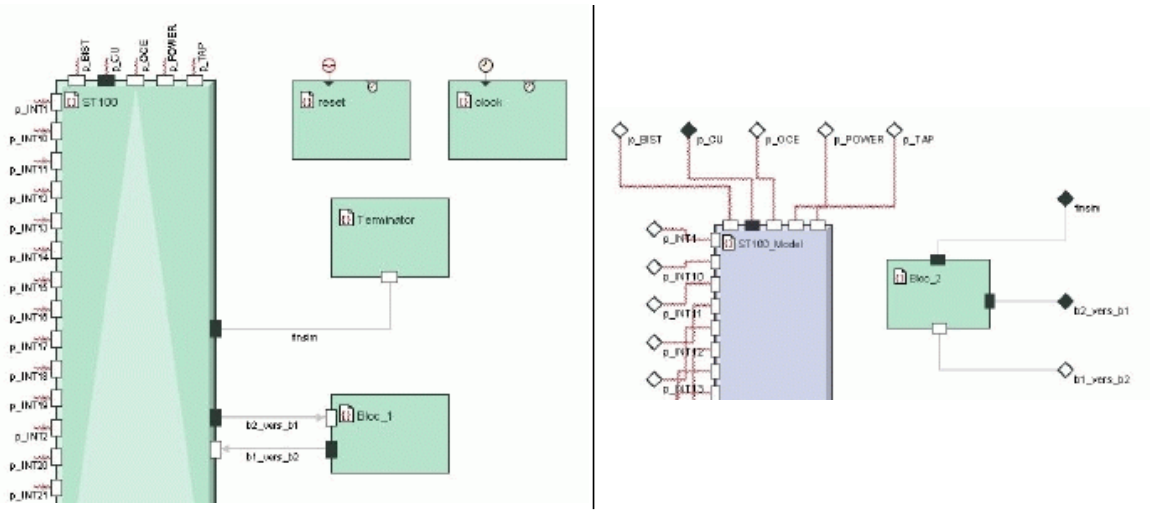


FIG. III.12 – Système du ping-pong après affectation d'une tâche au DSP

pour satisfaire certaines contraintes de l'application, telles que les priorités des interruptions, par exemple. L'outil N2C propose une interface par défaut déterminée à partir des types et des directions des ports de communication entre le logiciel et le matériel. Il est évidemment possible *d'imposer* certains choix d'interfaçage et de laisser le logiciel effectuer les autres.

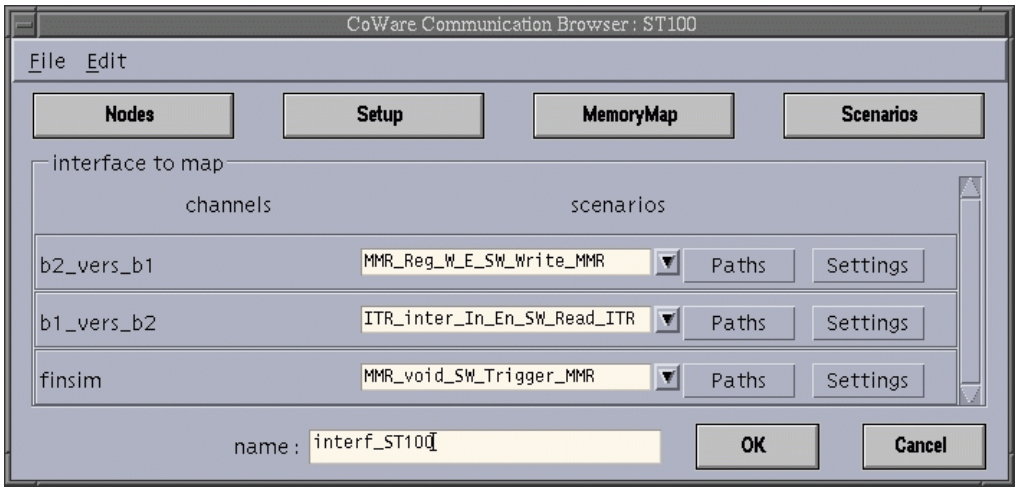


FIG. III.13 – Synthèse d'interface automatique

L'interface proprement dite entre le logiciel et le matériel se compose, pour chaque canal de communication traversant la frontière, d'une partie logicielle et d'au moins une partie matérielle. La partie logicielle, appelée *scénario* est responsable de la lecture ou de l'écriture des données correctes à l'adresse mémoire fixée dans les choix d'interface, tandis que les parties matérielles, appelées *ponts*, sont responsables du transfert des mêmes données du bus du processeur au périphérique visé. Les parties matérielles ainsi synthétisées sont des décodeurs d'adresse, des multiplexeurs et des démultiplexeurs, des convertisseurs de protocoles et des modèles de bus génériques reliant le cœur de processeur aux diverses ressources nécessaires au bon fonctionne-

ment du système, telles que les mémoires RAM et ROM par exemple. Le système de ping-pong obtenu *après* la synthèse d'interface dans le cas où le bloc 1 est réalisé en matériel et le bloc 2 en logiciel est représenté figure III.14.

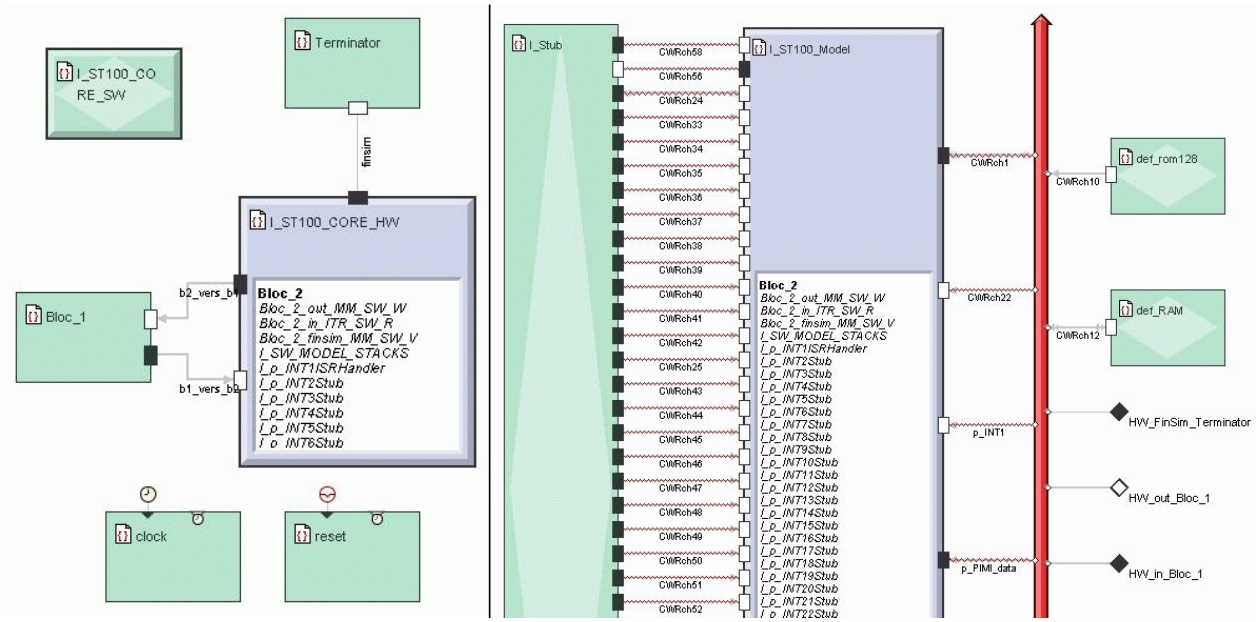


FIG. III.14 – Système du ping-pong après la synthèse d'interface

Pour chaque partitionnement testé et chaque interface générée, la partie logicielle de l'application doit être régénérée, de manière à incorporer les pilotes de périphérique corrects et les bonnes routines d'interruption. Pour les applications requiérant l'utilisation d'un système d'exploitation, il est possible d'intégrer un RTOS (Real Time Operating System) à qui confier l'ordonnancement des différentes tâches s'exécutant sur le cœur de processeur. Le processus de recréation de l'image logicielle génère des routines C compilables avec les outils associés au cœur pour chaque bloc affecté au domaine logiciel, et va compiler ces routines en une seule image mémoire, rajoutant au passage le code de boot, les routines initialisant les contrôleurs d'interruption et le séquenceur des éventuelles tâches de fond, faisant ainsi « disparaître » du système perçu par CoWare les blocs correspondants, leurs fonctionnalités étant désormais assurées par le cœur instancié. Le système *après* la génération de l'image logicielle est représenté figure III.15. Une fois l'image logicielle recréée, il faut affecter un niveau d'abstraction à chaque bloc du système. Les générateurs d'horloge et de reset, responsables de la mise à jour périodique de ces signaux, sont ainsi simulés à leur plus bas niveau d'abstraction. Pour les simulations semi-temporisées, le cœur de processeur est simulé à un niveau d'abstraction médian : son fonctionnement est temporisé, mais les transactions effectuées avec le matériel ne le sont pas, de manière à pouvoir réutiliser les blocs déjà décrits sans avoir à les modifier, ce qui permet de tester en un temps réduit plusieurs stratégies de partitionnement.

Les possibilités d'analyse à ce niveau de description sont surtout axées sur l'activité *logicielle* du système. Le diagramme de Gantt déjà présenté peut être tracé en fonction du nombre d'opérations effectuées *ou* en fonction du temps, du moins pour les tâches logicielles, cette information étant désormais disponible du fait de l'introduction de l'horloge au sein du système.

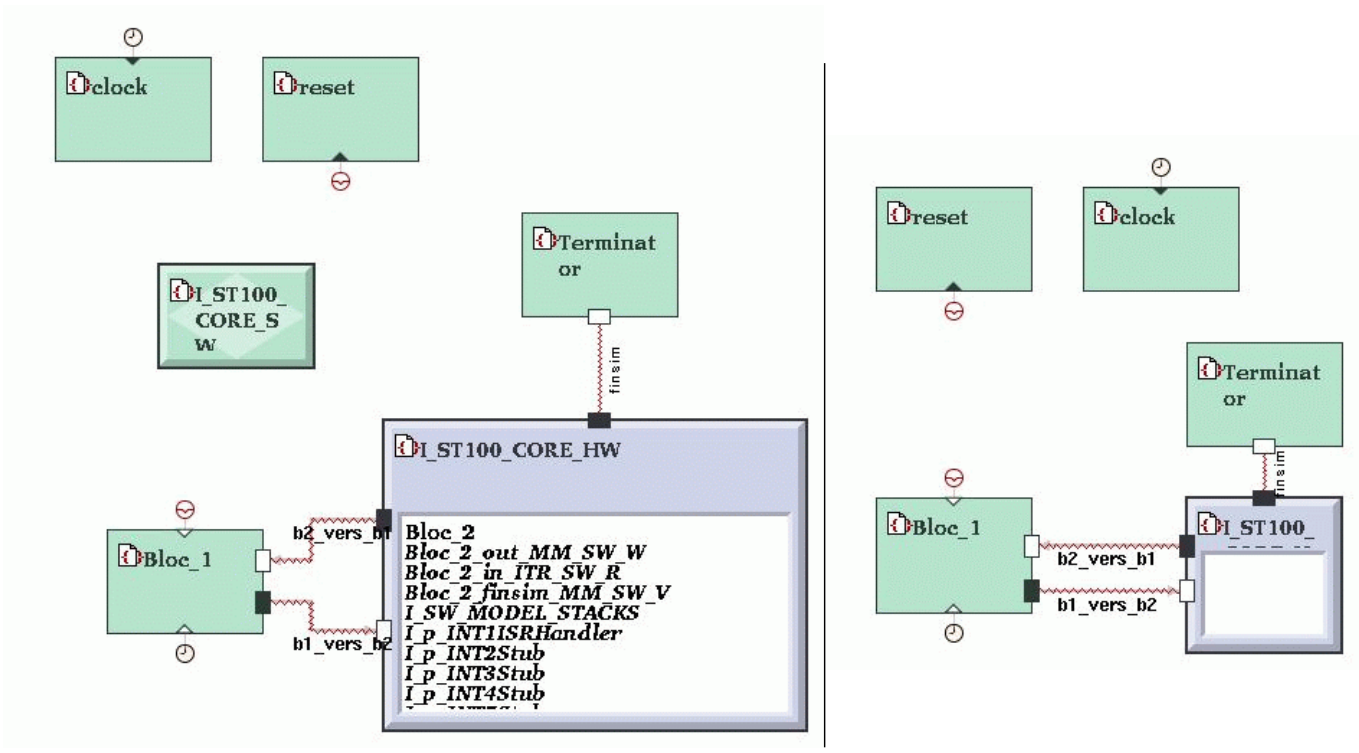


FIG. III.15 – Système du ping-pong avant et après génération de l'image logicielle

De même, il est possible de tracer le niveau de charge du processeur en fonction du temps et ce pour les différentes classes de processus que sont les tâches de fond, les routines d'interruption, les fonctions de l'OS ... Il est aussi possible de garder une trace de chaque accès mémoire en fonction du temps, qu'il s'agisse d'un accès en lecture ou en écriture, et ce vers de la mémoire programme, de la mémoire donnée, ou encore vers une adresse servant à piloter un périphérique.

Une autre donnée utile à l'évaluation de la pertinence du partitionnement testé est la mesure du taux d'activité des canaux de commmunication, dont on peut déduire une estimation grossière de la consommation des interconnections.

Cette étape de la méthodologie permet non seulement au concepteur de *choisir* le cœur de processeur à instancier dans le système, mais aussi de déterminer rapidement et avec relativement peu d'effort de codage le meilleur partitionnement de l'application sur l'architecture hétérogène. La fonctionnalité de synthèse d'interface automatique permet de connecter les blocs décrits de manière comportementale à un ISS temporisé sans avoir à modifier leur code, accélérant ainsi le processus itératif menant au partitionnement définitif.

III.2.2.3 Raffinement des protocoles de communication

Une fois la composante temporelle introduite dans le système pour les tâches logicielles, l'étape suivante consiste à temporiser les communications entre le domaine logiciel et le domaine matériel d'une part, et à semi-temporiser le fonctionnement des blocs matériels d'autre part.

Lors de la temporisation des communications inter-blocs, les ports d'entrée-sortie ne sont plus considérés comme des canaux abstraits de communication, dont le temps de latence est

nul, mais comme un ensemble de signaux véhiculant une signalisation et éventuellement une donnée dans une direction ou une autre. Les transactions effectuées sur ces bus élémentaires ont lieu selon un *protocole* déterminé, plus ou moins sophistiqué selon le degré de synchronisation requis.

En plus des caractéristiques des ports déjà mentionnées, la spécification d'un protocole de communication ajoute les éléments suivants à la déclaration des ports :

- le type de *poignée de main* éventuellement utilisée (sans, poignée de main simple ou poignée de main complète avec acquittement),
- pour les ports véhiculant une donnée, la largeur maximum des données circulant sur le port, en bits,
- pour les ports adressables, de type mémoire, la largeur de la composante d'adresse du port, en bits.

La spécification de ces protocoles se fait lors de la description des blocs à un niveau d'abstraction moins élevé appelé niveau *BCA*, pour *Bus-Cycle-Accurate*, précis au cycle de bus près, ce qui signifie la temporisation exacte des *transactions*, sans sous-entendre celle des *comportements* régis par ces transactions[8]. La manière classique de décrire dans un langage impératif des comportements astreints au respect de protocoles tels que ceux supportés par l'outil N2C est la programmation d'une *machine d'état*, sorte d'automate dont l'évolution d'un état à un autre est gouvernée par les fronts montants et descendants de certains signaux, comme les signaux composant les bus élémentaires de communication inter-blocs, comme indiqué plus bas avec l'exemple du pingpong.

Les différentes broches des protocoles supportés par l'outil N2C sont :

- pour le protocole *NoHndshk* (*No Handshake*, pas de poignée de main), le port P se résume à la seule donnée $P.d$ présentée par le maître à l'esclave, qui doit donc être parfaitement synchronisé avec le maître et savoir quoi faire de la donnée qui lui est présentée,
- pour le protocole *EnHndshk* (*Enable Handshake*, poignée de main simple), les broches du port P sont $P.en$, qui sert à signaler la transmission à l'esclave, et $P.d$ dans le cas d'une donnée typée. L'esclave doit réagir au passage à 1 de $P.en$ et éventuellement consommer la donnée présente sur $P.d$. Cette communication est non bloquante pour le maître car il ne fait que signifier l'*initiation* de la transmission sans attendre de réponse de la part de l'esclave.
- enfin, le protocole *FullHndshk* (*Full Handshake*, poignée de main avec acquittement) instaure un véritable mécanisme de synchronisation entre le maître et l'esclave. Le maître signifie à l'esclave le début de la transmission en mettant à 1 $P.req$ et en présentant l'éventuelle donnée $P.d$ dans le cadre d'un port véhiculant une donnée typée. En cas de non-disponibilité immédiate de l'esclave, le maître doit *maintenir* les signaux activés, et ce jusqu'à ce que l'esclave réponde au maître en mettant à 1 la broche $P.ack$, mettant ainsi fin à la transaction. Ce protocole permet de modéliser des transactions *bloquantes*, où le comportement du maître, si celui-ci respecte le protocole, est suspendu jusqu'à l'acquittement de l'esclave.

[8]Il existe aussi un niveau intermédiaire de description appelé *BCASH*, pour *Bus-Cycle Accurate Shell*, qui permet au concepteur d'utiliser *simultanément* les ports classiques non temporisés et les ports dotés d'un protocole spécifique. Des mécanismes de synchronisation intra-bloc sont alors proposés pour asservir le fonctionnement du bloc à l'horloge lorsque ses ports temporisés sont actifs. Ce niveau de description dont la seule utilité est de faciliter la migration d'un modèle non temporisé à un modèle semi-temporisé ne sera pas abordé par la suite.

Ces mécanismes se compliquent dans le cas de ports bidirectionnels ou de ports adressables, du fait soit de la détection par l'esclave de la *direction* de l'accès, effectuée à l'aide de la broche *P.nRW*, mise à 0 pour une lecture et à 1 pour une écriture, soit de la gestion de *l'adresse* (*P.A*) en plus de celle de *la donnée* pour les ports adressables.

Les protocoles utilisés pour l'exemple du ping-pong sont *EnHndshk* pour le port permettant au logiciel d'envoyer la donnée au matériel, car le matériel n'a pas de tâche particulière à exécuter qui serait plus prioritaire que la réception de la donnée, et *FullHndshk* pour le renvoi de la valeur au logiciel. L'utilisation d'un protocole de poignée de main avec acquittement se justifie par le fait que cette communication se faisant par le biais d'une interruption, la demande de transaction doit être maintenue tant que l'interruption n'a pas été acceptée et servie par le processeur, ce qui peut prendre un temps variable selon les tâches déjà en cours d'exécution sur le cœur. L'exemple ci-dessous illustre aussi le fait que les éventuels signaux d'horloge (ou de reset) cadençant le comportement des parties matérielles du système ne sont pas des ports rajoutés à la description des blocs temporisés mais plutôt des signaux issus des domaines déjà introduits pour le cadencement et la remise à zéro des cœurs de processeur.

```
#include "Protocols/basic.cwr"
blockdef bloc1(
  bool[32] in  : inslave,
  bool[32] out : outmaster )
begin
  C.UT.version { // Encapsulation déjà décrite ci-dessus
  };
  C.BCA.version ( in :: EnHndshk, out :: FullHndshk ){
    context {
      int new_val, val, state;
    };
    thread clk, rst.rst {
      if( rst.rst == 0 ){
        new_val = val = 0;
        state = 0;
        out.req = 0;
        out.d = 0;
      } else if( clk == 1 ){
        switch( state ){
        case 0 :
          if( in.en == 1 ){
            new_val = in.d;
            printf( "Bloc1 recoit %d.\n", new_val );
            state = 1;
          } break;
        case 1 :
          if( new_val != val ){
            val = new_val;
            out.d = val-2;
            out.req = 1;
            state = 2;
          } break;
        case 2 :
          if( out.ack == 1 ){
            out.req = 0;
            out.d = 0;
            state = 0;
          } break;
        }
      }
    };
    pragma region {
      CLOCK clk : slave :: CLOCK1;
      RESET rst : slave :: RESET1;
    };
  };
end;
```

Le respect de ces protocoles et des chronogrammes associés prend toute son importance lors de l'utilisation conjointe des protocoles standards supportés par N2C et de la synthèse d'interface automatique. En effet, à ce niveau de description, les blocs matériels réalisant l'interface entre le cœur de processeur et ses périphériques se conforment par construction à ces protocoles. C'est pourquoi il est crucial d'exécuter en réponse aux transactions initiées par ces blocs d'interface du code respectant les protocoles qu'ils utilisent, sous peine d'inhiber complètement la communication entre les domaines logiciel et matériel.

Un point particulier concernant cette temporisation des transactions concerne la *frontière* entre le logiciel et le matériel. Le cœur de processeur instancié étant perçu par l'outil N2C comme un bloc du plus haut niveau hiérarchique, il n'est possible de le simuler que dans deux modes, qui sont le mode non-temporisé, utilisé lors de l'évaluation des divers partitionnements possibles, et le mode temporisé, qui est utilisé une fois le partitionnement définitif déterminé.

Ceci implique que tous les blocs en contact immédiat avec le cœur du processeur doivent être simulés, et donc décrits, au même niveau d'abstraction. Dans le cas d'un cœur contrôlant un seul périphérique, seuls les ports reliant le périphérique au cœur doivent être temporisés, les blocs contenus à l'intérieur du périphérique peuvent être décrits dans l'une ou l'autre forme, pour peu que certaines précautions soient prises, tandis que dans le cas d'un cœur contrôlant un nombre important de périphériques, tous les périphériques doivent voir leur interface avec le cœur décrite au même niveau d'abstraction, empêchant ainsi la migration bloc par bloc du domaine non temporisé au domaine temporisé. Cette restriction n'est valable que pour les blocs directement interfacés avec le cœur du processeur.

Outre la description plus réaliste du comportement de la partie *matérielle* de l'architecture, cette étape au cours de laquelle les *transactions* entre les domaines logiciel et matériel acquièrent une *durée* et donc une réalité physique sert principalement à confirmer les choix effectués lors du partitionnement. Les traces obtenues lors de l'exécution de la partie logicielle, naturellement perturbées par les délais désormais induits par les communications auparavant instantanées, doivent permettre au concepteur de vérifier que le degré d'intrusion du matériel au sein du logiciel n'empêche pas ce dernier de satisfaire les contraintes temporelles qui lui ont été fixées. En conséquence, il faut désormais éviter toute modification du partitionnement retenu, sous peine de devoir réécrire le code temporisé de certains blocs et de devoir revalider le fonctionnement global de l'application compte tenu des nouveaux retards associés aux interfaces modifiées.

Si de telles contraintes ne sont plus satisfaites alors qu'elles l'étaient avec des communications non temporisées, deux solutions s'offrent au concepteur, qui peut soit choisir de se pencher plus avant sur les communications elles-mêmes, et tenter d'optimiser les échanges pénalisant dans de trop grandes proportions le fonctionnement du système, soit remonter d'un niveau dans la méthodologie de conception et apporter des modifications au *partitionnement* établi.

III.2.2.4 Raffinement des descriptions logicielles et matérielles

Enfin, la dernière étape préalablement à la validation finale par cosimulation est le raffinement des descriptions matérielles et l'optimisation des descriptions logicielles. Les blocs matériels sont décrits à leur plus bas niveau d'abstraction, dans un langage qui est au CoWareC ce que le VHDL synthétisable est au VHDL dans sa globalité, à savoir un sous-ensemble obéissant à des règles plus strictes mais décrivant très concrètement le matériel en cours de conception. Ce langage est appelé *RTC*, pour *Register Transfer C*, par analogie au VHDL *RTL*, pour *Register Transfer Level*. Les routines logicielles critiques sont elles optimisées de manière à tenir compte des spécificités offertes par une machine particulière, ou carrément réécrites dans un langage de plus bas niveau comme l'assembleur, par exemple.

Le but de cette étape est de permettre, lorsque la totalité d'un bloc est décrite en RTC, une traduction *automatique*[9] du code RTC en code VHDL, et la synthèse automatique d'une interface de cosimulation avec le logiciel responsable de la simulation du VHDL généré.

Les constructions ajoutées au C à cet effet ont d'ailleurs des équivalents très proches en VHDL[10], tels que les *signaux*, par exemple, qui, à la différence des *variables*, ne prennent effectivement la valeur qui leur est assignée qu'au front d'horloge *suivant* celui au cours duquel

[9]La traduction effectuée est plus une traduction *syntaxique* qu'une véritable analyse du code, bien que la distinction de certains motifs (processus synchronisés, resets asynchrones ...) soit effectuée.

[10]Et sûrement en Verilog.

la valeur en question leur est assignée. Quand ils rencontrent des constructions de ce type, les outils de synthèse logique instancient en général des *registres*. Les signaux en question sont dits *à affectation retardée*.

Un autre emprunt aux langages de description de matériel que sont VHDL et Verilog est l'apparition des qualificatifs *rise* et *fall* servant à désigner les fronts montants et descendants des signaux qu'ils qualifient, l'usage le plus fréquent qui en soit fait en RTC étant la dénomination de la tâche synchrone principale par `thread rise(clk) {...};`. Un autre de ces ajouts au C est la possibilité non illustrée ci-dessous de ne modifier que certains bits d'un *std_logic_vector* ou encore de former des vecteurs de bits par concaténation.

C'est aussi à cette étape que le concepteur fixe la *taille* des signaux et des variables au bit près, de manière à ce que la simulation reproduise aussi fidèlement que possible le fonctionnement du matériel. Ce passage d'une taille d'entier égale à 32 bits à une taille de donnée inférieure peut révéler la présence d'erreurs invisibles auparavant, uniquement dues à des problèmes de saturation ou de débordement se produisant plus rarement avec des tailles de donnée supérieures.

La migration du code source présenté ci-dessus au langage RTC donne le code source inclus ci-dessous. Les principales modifications sont la précision des tailles des signaux d'entrée-sortie et des signaux ou variables internes, qui seront traduites en *std_logic* ou *std_logic_vector* de la bonne largeur, la déclaration dans le *context* des signaux servant à garder une trace interne de l'état du bloc, l'introduction des mots réservés *rise* et *fall* pour préciser les fronts de l'horloge et du reset sur lesquels le bloc doit réagir, le rajout d'une clause *default* à l'instruction *switch*, de manière à pouvoir générer du code VHDL correct, et l'ajout de la possibilité de traduction automatique en VHDL par l'instruction *pragma translate { SYNTHESIS = "on" }*.

```
#include "Protocols/basic.cwr"
blockdef bloc1(
    bool[4] in  : inslave,
    bool[4] out : outmaster )
begin
  C.UT.version { // Encapsulation déjà décrite ci-dessus
  };
  C.BCA.version ( in :: EnHndshk, out :: FullHndshk ){
    context {
      typedef Signal_int Signal_bool_4;
      typedef Signal_int Signal_bool_2;
      Signal_bool_4 new_val, val;
      Signal_bool_2 state;
    };
    thread rise(clk), fall(rst.rst) {
      if( rst.rst == 0 ){
        new_val = 0;
        val = 0;
        state = 0;
        out.req = 0;
        out.d = 0;
      } else if( clk == 1 ){
        switch( state ){
        case 0 :
          if( in.en == 1 ){
            new_val = in.d;
            state = 1;
          } break;
        case 1 :
          printf( "Bloc1 recoit %d.\n", (int) new_val );
          if( new_val != val ){
            val = new_val;
            out.d = new_val-2;
            out.req = 1;
            state = 2;
          } break;
        case 2 :
          if( out.ack == 1 ){
            out.req = 0;
            out.d = 0;
            state = 0;
          } break;
        default :
          break;
        }
      }
    };
    pragma region {
      CLOCK clk : slave :: CLOCK1;
      RESET rst : slave :: RESET1;
    };
    pragma translate {
      SYNTHESIS="on";
    };
  };
end;
```

Une fois le bloc décrit à ce niveau d'abstraction, il est possible *sans retoucher à son code source* de générer une simulation ou le bloc est simulé en C sur la machine hôte ou bien en

VHDL sur un simulateur de type VSS ou NC-SIM. Le code généré dans le cas d'une traduction du modèle en VHDL est présenté ci-dessous. Le code généré se présente sous la forme de deux fichiers, un pour la déclaration de l'entité, l'autre pour la déclaration de l'architecture générée de cette entité. Ces deux fichiers ont été fondus pour faciliter leur présentation. Il est également possible de synthétiser le code produit avec des outils comme Design Compiler, de Synopsys, puis de réintégrer un modèle VHDL de la netlist générée afin d'effectuer des cosimulations *post-synthèse*.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
library COWARE;
use coware.c2vhdl.all;
-- VHDL-dumper pragma section:

-- Entity Bloc_1:
entity Bloc_1 is
port (
    in_d     : in  STD_LOGIC_VECTOR(3 downto 0);
    in_en    : in  STD_LOGIC;
    out_d    : out STD_LOGIC_VECTOR(3 downto 0);
    out_req  : out STD_LOGIC;
    out_ack  : in  STD_LOGIC;
    clk_clk  : in  STD_LOGIC;
    rst_rst  : in  STD_LOGIC
);
end Bloc_1;

architecture BCA_Translated_Bloc_1 of Bloc_1 is

-- context:
-- pragma translate_off
constant INSTANCENAME : string := "/Bloc_1";
-- pragma translate_on
signal new_val: std_logic_vector(3 downto 0);
signal val: std_logic_vector(3 downto 0);
signal state: std_logic_vector(1 downto 0);

begin  -- architecture

cwr1: process (clk_clk, rst_rst) -- thread clk_clk:
begin
  if (rst_rst = '0') then
    new_val <= "0000";
    val <= "0000";
    state <= "00";
    out_req <= '0';
    out_d <= "0000";
  elsif (clk_clk'event and clk_clk = '1') then
    case state is
    when "00" =>
      if (in_en = '1') then
        new_val <= in_d;
        state <= "01";
      end if;
    when "01" =>
      -- not translated: printf("Bloc1 recoit %d.\n", (int)new_val)
      if (new_val /= val) then
        val <= new_val;
        out_d <= new_val - "0010";
        out_req <= '1';
        state <= "10";
      end if;
    when "10" =>
      if (out_ack = '1') then
        out_req <= '0';
        out_d <= "0000";
        state <= "00";
      end if;
    when others =>
    end case;
  end if;
end process; -- thread clk_clk.

end BCA_Translated_Bloc_1;

configuration C_BCA_Translated_Bloc_1 of Bloc_1 is
for BCA_Translated_Bloc_1
end for;
end C_BCA_Translated_Bloc_1;
```

Une fois ces étapes franchies avec succès, le concepteur dispose d'une chaine de cosimulation composée d'un ou plusieurs ISS simulant les cœurs de processeurs ou DSP présents dans le système, d'un simulateur VHDL ou Verilog gérant les divers opérateurs câblés et d'une interface entre le logiciel et le matériel elle-même composée de blocs décrits en C et simulés au niveau cycle-accurate. Il est parfois possible, selon les fonctionnalités offertes par l'intégration du cœur de processeur au sein de l'outil N2C, de traduire les blocs composant l'interface en VHDL et de les simuler avec la partie matérielle du système, ne laissant au code C exécuté sur la machine hôte que la responsabilité du lancement des divers simulateurs intervenant lors de cette phase finale de cosimulation.

III.2.3 Points durs liés à la méthodologie

La méthodologie proposée, ici démontrée sur un exemple simple, permet de migrer sans rupture flagrante d'une modélisation comportementale à une chaîne complète de cosimulation ISS/VHDL. Cependant, cette méthodologie n'est pas exempte de défauts et son déroulement dans le cadre d'une application à forte composante « signal » présente tout de même quelques point durs qui sont rapidement évoqués ci-après.

III.2.3.1 Faible orientation « traitement du signal » de l'outil

Le premier des défauts exhibés par la méthodologie proposée est l'orientation résolument plus « contrôle » que « données » de l'outil N2C.

En effet, le langage nativement supporté par CoWare ne proposant que les types de données standards supportés par le langage C, il est difficile d'utiliser *simplement* des outils tels que les nombres en précision finie ou des opérations arithmétiques sophistiquées telles que la saturation, la troncature ou encore les diverses formes d'arrondi.

Depuis peu, la bibliothèque de classes C++ SystemC constitue une solution à ce genre de problèmes. SystemC offre en effet la possibilité d'utiliser nativement des nombres à virgule fixe, des nombres en précisison finie à virgule variable, ainsi que les opérations arithmétiques mentionnées ci-dessus. N2C acceptant désormais les systèmes décrits en SystemC, ces outils arithmétiques deviennent accessibles au concepteur à deux précautions près :

- l'utilisation des types de données SystemC est à priori incompatible avec l'affectation du bloc considéré au domaine logiciel. En effet, bien que les classes de SystemC soient écrites en C++ standard, leur compilation et leur exécution sur une architecture destinée à l'embarqué telle qu'un cœur de DSP risque de ne pas être optimale, surtout si leur utilisation vise à reproduire le fonctionnement d'opérateurs et de types de données supportés naturellement par le DSP en question. Les cœurs de DSP disposant bien souvent de leurs propres types de données destinés à assurer ces fonctionnalités et des instructions arithmétiques adéquates, il faut soit développer une couche de conversion entre les types de données SystemC et les types supportés nativement par le compilateur du DSP, soit réécrire le code faisant appel à ces fonctions avec des instructions C (ou directement en assembleur) directement compréhensibles par le compilateur du DSP une fois le partitionnement finalisé. On utilisera dans ce cas ce qu'on appelle des instructions *intrinsèques* (*intrinsics* en anglais), qui permettent d'exploiter *avec la sémantique du C* les types de données et les opérations arithmétiques *spécifiques au DSP*.
- le passage d'une description SystemC à du VHDL synthétisable est encore aujourd'hui loin d'être évident, à plus forte raison lorsque les opérateurs arithmétiques invoqués possèdent des fonctionnalités non standards dont l'équivalent synthétisable n'existe pas forcément. Certes, des outils comme CoCentric SystemC Compiler [Syn00b] existent, mais pour être efficaces avec des blocs décrits à haut niveau, ils font le plus souvent appel à des techniques de synthèse comportementale dont la qualité des résultats n'est pas encore considérée comme globalement satisfaisante.

SystemC permet donc d'accéder *en simulation* à des représentations des donnéess en précision finie, mais le passage de ces types de données à des types *compilables efficacement* ou à des opérateurs synthétisables n'est pas encore acquis, ce qui complique l'utilisation de la méthodologie proposée dans le cadre d'une application à forte composante « signal ».

III.2.3.2 Temporisation fastidieuse des communications

La méthodologie proposée présente lors de la temporisation des communications et de la synchronisation des processus aux horloges du système un caractère relativement fastidieux. Pour *chaque* port de *chaque* bloc affecté au domaine matériel, le concepteur doit réécrire la machine d'état respectant le protocole choisi pour gérer la communication, ce qui peut rapidement s'avérer répétitif.

De plus, dans le cas d'un seul processus synchronisé gérant plusieurs ports temporisés, le nombre d'états peut croître de manière exponentielle avec le nombre de ports gérés simultanément, ce qui complique grandement l'écriture de la machine d'état. La *composition d'automates* offerte par les langages synchrones de type Esterel [Ber00] constitue une solution élégante à ce problème. Dans le cas où les protocoles des ports sont les mêmes, le concepteur peut ne décrire qu'un seul d'entre eux et exécuter *plusieurs* instances de cet automate en parallèle, chaque instance ne gérant qu'un port. Un code décrivant le comportement de la machine d'état globale peut alors être généré sans avoir été écrit par le concepteur, ce qui est particulièrement appréciable pour un grand nombre de ports.

La génération d'un squelette d'automate réagissant aux variations des différentes broches de chaque port a été evoquée avec les fournisseurs de N2C et la faisabilité de l'implémentation de cette fonctionnalité est actuellement à l'étude.

III.2.3.3 Temporisation des blocs au contact du coeur

Les cœurs de processeur sont considérés par N2C comme des modèles temporisés, nécessitant donc une horloge et un signal de reset, dont les connexions sont soit *toutes* instantanées (niveau d'abstraction *BCASH*, *Bus-Cycle-Accurate Shell*) soit *toutes* temporisées (niveau d'abstraction *BCA*, *Bus-Cycle-Accurate*). Le niveau *BCASH*, certes déconnecté de la réalité *physique* du système en cours de conception, est là principalement pour faciliter la phase d'exploration architecturale, au cours de laquelle les blocs matériels non temporisés peuvent être interfacés au cœur de processeur sans intervention du concepteur, pour peu que le cœur soit simulé au niveau *BCASH*.

Une fois le partitionnement établi, la phase suivante consiste à temporiser les communications inter-blocs d'abord et le comportement des blocs matériels ensuite, sachant qu'à chaque étape de ce processus, deux règles doivent être respectées :

- deux ports connectés doivent être décrits au même niveau de temporisation,
- les ports des blocs matériels connectés au cœur de processeur doivent *tous* être décrits au même niveau que le processeur (*BCASH* ou *BCA*).

Pour les blocs ne faisant pas partie de l'interface entre le domaine logiciel et le domaine matériel, la contrainte est relativement faible, puisque le concepteur n'a qu'à maintenir une « frontière » au niveau *BCASH* entre les différents niveaux de description, la déplaçant lorsque des blocs auparavant non-temporisés le sont entièrement. Cette frontière peut aussi se présenter sous la forme d'un périmètre entourant un îlot au niveau *BCA* au milieu d'un système encore décrit au niveau *UnTimed*, ou sous la forme de plusieurs îlots disjoints. Cette solution implique le maintien d'une description au niveau *BCASH* de chaque bloc matériel à un moment ou à un autre, ce qui requiert un effort important, et ne permet la migration souple que de la partie purement *matérielle* du système, la partie réalisant l'interface avec le cœur ne devant migrer qu'au dernier moment au niveau *BCA*, en même temps que le cœur.

Chapitre IV

Réalisations et approche du prototypage

Ce court chapitre a pour objectif de faire le lien entre la méthodologie de conception au niveau système proposée dans le chapitre précédent et le prototypage effectif des architectures et des applications développées à l'aide de cette méthodologie. Tout d'abord, on détaillera l'environnement de prototypage au sens large, c'est-à-dire les *fonctions* que devra assurer le prototype et les *composants* utilisés ainsi que leurs caractéristiques. Ensuite, une incompatibilité logicielle majeure ayant empêché la mise au point de l'application retenue conformément à la méthodologie proposée, l'implémentation *logicielle* réalisée est présentée, avant de mentionner les problèmes spécifiques au prototypage ayant été relevés et des solutions possibles.

IV.1 Environnement de prototypage

L'environnement de prototypage regroupe l'ensemble des *caractéristiques* du prototype à réaliser. L'objectif ici n'étant pas la réalisation d'un terminal UMTS/FDD, mais l'étude de l'implémentation de certains algorithmes critiques sur une architecture définie *en fonction* de ces algorithmes, il s'agit dans un premier temps de déterminer le sous-ensemble des fonctions d'un *véritable* terminal à exécuter, puis dans un second temps les *composants* assurant cette exécution. L'architecture hétérogène envisagée étant divisée en une partie logicielle (l'utilisation d'un processeur généraliste en plus du DSP n'est pour l'instant pas considérée) et une partie matérielle, il faut choisir les organes qui vont être utilisés pour émuler ces deux parties ainsi que leur interconnexion.

IV.1.1 Périmètre du prototypage

Le périmètre du prototypage, ou périmètre d'émulation, est le sous-ensemble des fonctionnalités de l'équipement à prototyper dont on désire valider l'implémentation.

La seule fonction à protoyper ici est l'estimation de canal, qui sera réalisée au moyen de l'algorithme optimisé proposé section II.3.4. En particulier, la démodulation des données (et donc le récepteur Rake) ne sera pas effectuée, pas plus que les étapes suivantes du traitement des données reçues par le terminal (décodage canal, désentrelacemement, décodage source ...). De plus, on ne validera *que* l'estimation de canal, et non pas son *intégration* au sein d'un environnement logiciel d'exécution semblable à celui d'un terminal. On ne traitera donc pas l'intégrabilité de la tâche décrite à un système d'exploitation, pas plus que son interruptibilité

ou que l'éventuel caractère réentrant du code proposé. Pour la simplicité de la démonstration, on ne considèrera aucune couche protocolaire ni aucun système de signalisation éventuel entre la station de base et le terminal.

Les données seront supposées déjà présentes en mémoire, laquelle reste à définir, ainsi que les paramètres nécessaires à l'estimation de canal (facteur d'étalement, numéros des codes d'étalement et d'embrouillage, symboles pilotes ...). La « sortie » de l'algorithme (le nombre de trajets détectés et leurs atténuations complexes) sera écrite en mémoire dans un format restant à définir.

Le prototype devra effectuer les opérations requises, ici l'estimation du canal, aussi rapidement que possible, de manière à se rapprocher autant que possible du fonctionnement en temps réel. Il faut prendre garde au sens que l'on accorde ici à l'expression « temps réel ». La stratégie globale du récepteur n'étant pas encore déterminée, on ne sait pas de combien de temps le Searcher disposera pour effectuer l'estimation du canal. Dans le cas où aucun dispositif de poursuite n'est utilisé, le Searcher devra estimer le canal une fois par slot, ce qui est le pire cas de fonctionnement, alors que si l'estimation complète du canal n'a lieu qu'une fois ou deux par trame, les autres slots étant traités par une *réactualisation* de la réponse impulsionnelle plutôt que par son *recalcul complet*, on pourra allouer un temps plus important au Searcher pour mener à bien l'estimation du canal. On cherchera donc à minimiser autant que possible le temps d'exécution de la routine d'estimation du canal en tenant compte des limitations du matériel et des contraintes globales induites par l'exécution en environnement contraint, où les temps d'exécution se traduisent directement en consommation et donc en réduction d'autonomie.

IV.1.2 Emulation du logiciel

L'exécution de la partie *logicielle* de l'application sera confiée au DSP ST120 de STMicroelectronics [STM00]. C'était en effet au début de ces travaux un des rares DSP supportés par l'outil N2C, ce qui constituait un prérequis à l'utilisation de la méthodologie proposée. D'autre part, divers partenariats avec STMicroelectronics nous ont facilité l'obtention de son kit d'intégration sous CoWare *et* de sa carte d'évaluation une fois son développement terminé.

IV.1.2.1 Introduction de l'architecture ST100

Le cœur de DSP choisi appartient à la famille de DSP-MCU ST100 [STM99], dont les principales caractéristiques architecturales sont présentées ci-dessous :

- support de 3 jeux d'instruction : le GP16[1], jeu d'instructions 16 bits fournissant une bonne densité de code au prix d'un accès restreint aux ressources de la machine, et donc plus adapté à du code de contrôle, le GP32[2], jeu d'instructions 32 bits « nominal » de la machine, offrant l'accès à la totalité des ressources, et le SLIW[3], jeu d'instructions 128 bits (4×32 bits) permettant l'exploitation d'un degré plus élevé de parallélisme, adapté à l'exécution de tâches orientées « flot de données »,
- largeur du chemin de données de 16 bits : les registres de donnée font 40 bits de large, les multiplieurs sont du type 16×16 et les accumulations se font sur 40 bits,

[1] *General Purpose 16 Bits*
[2] *General Purpose 32 Bits*
[3] *Scoreboarded Long Instruction Word*

- 3 compteurs de boucles matériels, évitant la « pollution » des boucles dont le nombre d'itérations est connu par la gestion du compteur,
- utilisation de l'exécution *gardée* : toutes les instructions sont conditionnées par un bit de garde, de manière à éviter les pénalités dues aux branchements conditionnels,
- support de l'arithmétique sur 8, 16, 32 et 40 bits et disponibilité des opérateurs d'arrondi/saturation,
- support des opérations de type *SIMD*[4], comme par exemple les additions séparées des mots de poids fort et faible de deux registres de 32 bits,
- architecture à *accès découplé* minimisant la latence des accès mémoire et réduisant le nombre de registres requis,
- accès logiciel à 4 modes d'économie d'énergie.

Grâce à ses trois jeux d'instructions, l'architecture ST100 est donc en théorie capable d'assurer aussi bien les fonctions de microcontrôleur que de processeur de traitement de signal à hautes performances, mais pas simultanément.

Le cœur de DSP ST100 est composé de trois unités distinctes, comme indiqué figure IV.1 :

- l'unité de données DU (*Data Unit*), responsable des opérations arithmétiques, qui dispose de 16 registres de 40 bits,
- l'unité d'adresses AU (*Address Unit*), responsable de la gestion des pointeurs et des modes d'adressage spécifiques, qui dispose de 17 regsitres de 32 bits,
- ł'unité de contrôle CU (*Control Unit*), qui gère entre autres les compteurs de boucles matériels et les changements de jeu d'instructions.

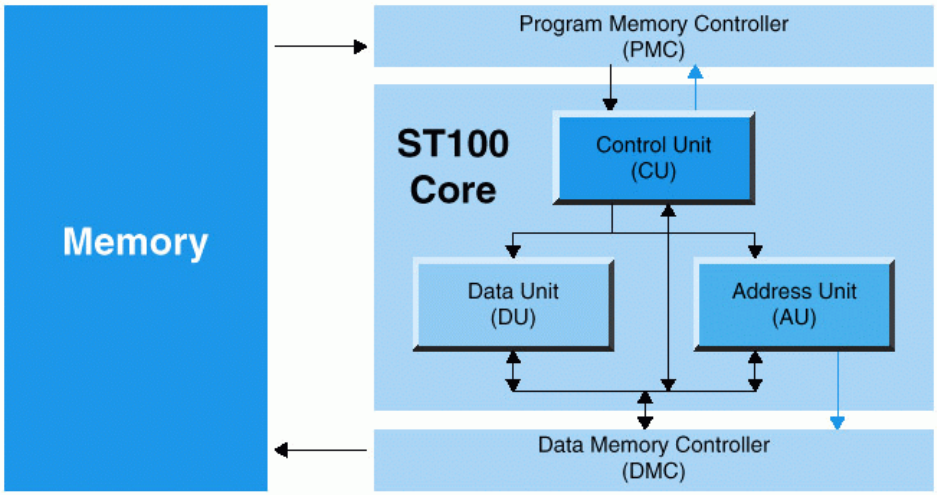


FIG. IV.1 – Système générique bâti autour d'un cœur de ST100

IV.1.2.2 Spécificités du ST120

Le DSP-MCU ST120 est la première implémentation de la famille ST100. Il embarque 128 kilo-octets de mémoire « programme » et deux fois 32 kilo-octets de mémoire « données » (architecture Harvard). Son unité de données est divisée en deux sous-unités identiques, et peut

[4] *Single Instruction Multiple Data*

donc exécuter deux instructions arithmétiques par cycle[5]. Pour alimenter ces deux sous-unités de données, son unité d'adresses est capable de générer deux adresses par cycle. L'architecture du ST120 est représentée figure IV.2. L'utilisation du jeu d'instructions SLIW sur le ST120 ne permet pas l'exécution de *toutes* les combinaisons d'instructions en un cycle. Les combinaisons majoritairement utilisées sont au nombre de 3 :

- Load1, Load2, Arithmetic1, Arithmetic2
- Load1, Arithmetic1, Arithmetic2, Store1
- Arithmetic1, Arithmetic2, Store1, Store2

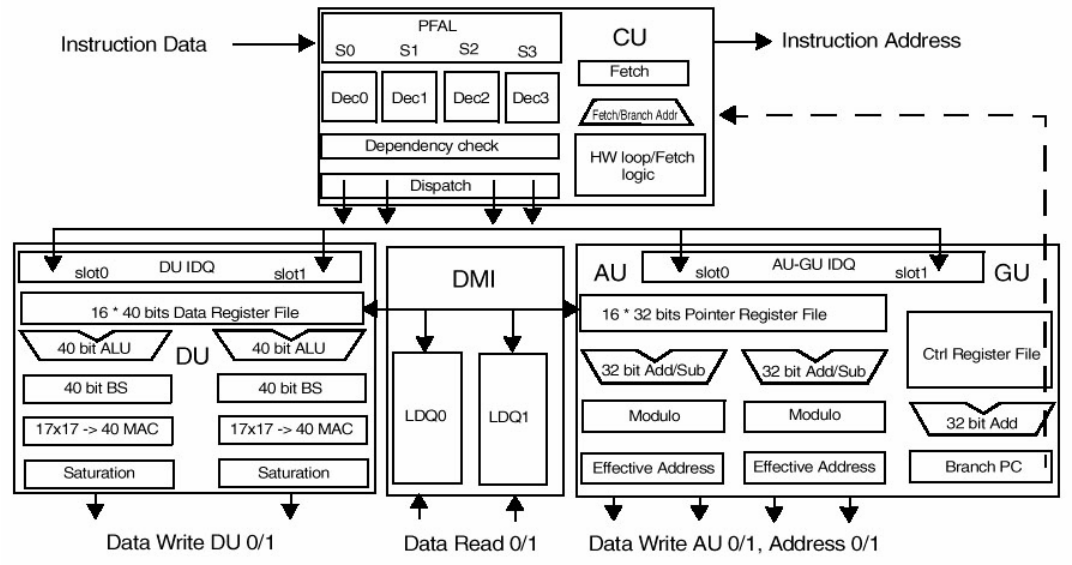


FIG. IV.2 – Architecture interne du DSP-MCU ST120

Le fait que l'architecture ST100 soit à acès découplé permet aux intructions arithmétiques d'utiliser les données chargées au cours du *même* paquet d'instructions SLIW, de même que les opérations de sauvegarde des résultats en mémoire voient les résultats des opérations arithmétiques du *même* paquet d'instructions SLIW.

IV.1.2.3 Chaîne de compilation du ST120

La chaîne de compilation du ST100 comprend les outils de développement classiques, dont un compilateur C qui souffre d'un défaut majeur : il ne génère pas de code SLIW, empêchant ainsi l'accès aux performances réelles de la machine. Les seules particularités architecturales de la famille ST100 mises à profit par le compilateur sont les compteurs de boucles matériels et l'exécution gardée. Ce défaut est particulièrement dommageable, d'autant plus que l'assembleur est parfaitement capable de générer du code utilisant ce jeu d'instructions. Il existe deux solutions pour créer du code exploitant le jeu d'instructions SLIW et exécutant réellement 4 instructions par cycle :

- écrire directement le code voulu en assembleur, ce qui implique une connaissance étendue de la machine et de son fonctionnement au niveau micro-architectural,

[5]Le ST140, prochain membre de la famille ST100, disposera de 4 unités arithmétiques.

– utiliser un outil externe à la chaîne de compilation : le *LAO*[6], développé par STMicroelectronics[7]. Cet outil prend en entrée un programme écrit en *assembleur linéaire* et l'optimise suivant les directives données par le programmeur, écrivant au passage du code SLIW si besoin est.

L'assembleur linéaire est une abstraction de l'assembleur pur et dispose de nombreuses fonctionnalités destinées à faciliter le développement dans ce langage, telles que le renommage de registres, la déclaration simplifiée des espaces d'adressage, le typage des registres (donnée, pointeur ...), ou encore la caractérisation des boucles (boucle déroulable, boucle parallélisable, congruence du nombre d'itérations lorsque celui-ci est connu ...). Développer en assembleur linéaire exige certes une bonne connaissance de la machine, mais les détails d'optimisation sont laissé à la charge du LAO, à la différence du développement en assembleur pur.

Le LAO peut soit s'utiliser tel quel, en mode *standalone*, pour optimiser des programmes directement écrits en assembleur linéaire, soit s'intercaler au milieu de la chaîne de compilation de manière à optimiser le code assembleur généré par le compilateur avant la phase d'assemblage proprement dite.

IV.1.2.4 Prototypage sur ST120 : la carte d'évaluation EV2

Le prototypage d'applications sur le ST120 est possible grâce à la carte d'évaluation EV2, munie du circuit EVA2, qui comprend lui-même un cœur de DSP ST120 et plusieurs périphériques. La carte peut soit opérer déconnectée de tout environnement de travail, grâce à la présence de mémoire Flash pouvant embarquer le code exécuté au boot, soit travailler en liaison avec un poste de développement pour l'émulation de code en cours de mise au point.

Le circuit EVA2 embarqué sur la carte peut fonctionner à des fréquences d'horloge allant de 10 à 200 MHz, ce qui permet d'évaluer ses performances pour une grande gamme d'applications. Deux types de mémoire sont présents sur la carte : 2 méga-octets de mémoire volatile SRAM, et 512 kilo-octets de mémoire Flash permettant le fonctionnement de la carte en mode *standalone*.

De plus, deux connecteurs sont présents sur la carte, permettant la mise au point de systèmes complexes :

– un connecteur « hôte », permettant à un autre organe de contrôler divers aspects du fonctionnement du DSP : gestion des phases de reset et des horloges, chargement des mémoires programme et données ... Ce connecteur fournit en plus un accès prioritaire à la *totalité* de l'espace adressable du DSP, ce qui permet de contrôler son fonctionnement et celui des divers périphériques qui lui sont attachés.
– un connecteur « système », permettant un accès non-prioritaire au bus d'extension du DSP ainsi qu'à un certain nombre de signaux génériques destinés à communiquer avec le DSP. C'est ce connecteur qui va permettre de relier des périphériques non standards au DSP par l'intermédiaire du partage du bus d'accès mémoire.

La carte EV2 et ses connexions possibles sont représentées figure IV.3.

L'utilisation de la carte EV2 comme accélérateur lors du développement d'applications sur ST100 permet d'effectuer des itérations entre les simulations et l'écriture du code proprement dite beaucoup plus rapidement qu'avec les simulateurs tournant sur la machine hôte. Outre

[6] *Linear Assembly Optimizer*

[7] La chaîne de compilation du ST100 n'a pas été développée par STMicroelectronics, mais par Green Hills Software, Inc., omniprésent vendeur d'outils de développement pour l'embarqué.

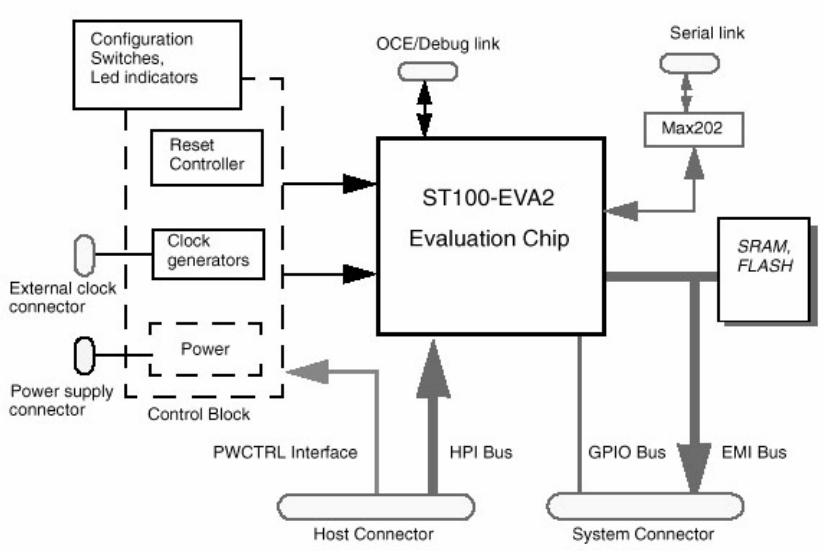


FIG. IV.3 – Carte d'évaluation EV2 et interconnexions

le fait que le programme en cours de développement tourne alors sur sa cible *finale*, ce qui garantit la correction fonctionnelle du code en cas de succès des simulations, il tourne aussi beaucoup plus vite, le circuit embarqué sur la carte pouvant fonctionner jusqu'à 200 MHz. Le contrôle complet de l'application tournant sur la carte est rendu possible grâce à une interface JTAG[8]. De plus, la présence sur la carte de compteurs/timers contrôlables par le logiciel permet d'obtenir des mesures précises de temps d'exécution, ce qui prend habituellement énormément de temps avec des simulateurs cycle-accurate.

IV.1.3 Emulation du matériel

Le problème de l'émulation de la partie *matérielle* de l'architecture devant accueillir l'application est plus compliqué à résoudre que celui de l'émulation du logiciel. Après les phases de conception, de synthèse et de validation par (co-)simulation, le flot de conception ASIC classique enchaîne les phases de placement-routage et de génération du masque. Disposer d'un *prototype* du circuit devant être ainsi réalisé nécessite de se détacher de la technologie visée lors de la fabrication. Le prototypage se fait en effet à l'aide de circuits reprogrammables, comme les *FPGAs*[9] ou les *CPLDs*[10], qui vont permettre de tester le fonctionnement du circuit *sans* l'implanter réellement, et ainsi d'y apporter des modifications et surtout de tester ces modifications bien plus facilement et bien plus rapidement que s'il fallait refabriquer le circuit à chaque itération.

Un FPGA peut être vu comme une grille de cellules élémentaires dont les interconnexions sont programmables. Chaque cellule élémentaire comprend des éléments de logique câblée qui dépendent des fabricants, mais il existe une constante qu'on trouve dans tous les FPGAs : la

[8] *Joint Tag Action Group*, une interface de test standardisée à laquelle la majorité des circuits embarqués et des cartes de développement se conforment.

[9] *Field Programmable Gate Array*

[10] *Complex Programmable Logic Device*

porte combinatoire à 4 entrées. La sortie de ce bloc peut être programmée de manière à émuler le fonctionnement de n'importe quelle fonction combinatoire à 4 entrées, ce qui permet, en interconnectant plusieurs de ces cellules élémentaires programmées chacune avec une fonction différente, d'émuler des circuits logiques de manière *programmable*. Les outils utilisés ne sont pas les outils de synthèse classiques, plus orientés vers les ASIC, mais des outils spécifiques à ce type de circuit, la taille et la fonction des blocs élémentaires manipulés n'étant pas les mêmes. De la même manière que pour les ASIC, il existe des outils de placement-routage et d'implantation pour les FPGAs, vendus par les fabricants des FPGAs concernés. Aujourd'hui, les FPGAs les plus sophistiqués peuvent intégrer un cœur de processeur, ou d'autres macro-blocs dédiés tels que des contrôleurs PCI, des ports de communication, ou encore des mémoires...

La société Aptix commercialise des plates-formes d'émulation de circuits intégrés qui répondent bien au besoin exprimé ici. Ces plates-formes, utilisables en émulation pure ou en cosimulation à l'aide d'une interface séparée, permettent de confier l'exécution de code VHDL à des composants reprogrammables situés sur la machine Aptix.

Les blocs matériels à émuler sont synthétisés vers des technologies FPGA puis mappés sur la plate-forme Aptix par l'outil System Explorer [Apt00b], qui va *router* les interconnexions entre les FPGAs placés sur la plate-forme et permettre de découper le circuit à émuler en plusieurs FPGAs le cas échéant. La plate-forme ainsi configurée va pouvoir être utilisée comme un prototype du circuit final et permettre sa mise au point grâce à des possibilités de debug et d'analyse logique[11]. Une fois les éventuels problèmes de synthèse vers des FPGAs réglés, les systèmes matériels complexes peuvent donc être vérifiés beaucoup plus rapidement qu'en simulation pure sur une station de travail, les vitesses revendiquées par le constructeur approchant les 40 MHz.

L'interface MVP [Apt00a] permet de coupler ces plates-formes à des simulateurs tournant sur une station de travail et de les utiliser comme des accélérateurs matériels de simulation. Ainsi, au sein d'une simulation VHDL, une partie des blocs simulés pourra être mappée sur la machine Aptix, l'interfaçage entre les blocs mappés sur le simulateur VHDL tournant sur la machine hôte et la machine Aptix étant réalisé par le module MVP, comme indiqué figure IV.4. La vitesse atteignable en cosimulation annoncée par Aptix est de 250 kHz.

Cette interface MVP n'est malheureusement pas gérée par l'outil N2C, et n'est donc pour l'instant disponible que dans le cas de simulations VHDL (ou Verilog) pures et dans certains environnements de développements algorithmiques, comme SPW [Cad01a] par exemple. Cependant, une API[12] en langage C étant fournie, seule la complexité de la génération des diverses « coquilles » de cosimulation empêche pour l'instant le déport des blocs matériels d'une cosimulation CoWare vers une machine Aptix. L'implémentation de cette fonctionnalité a été proposée à CoWare et sa faisabilité est actuellement à l'étude.

Dans le cadre du développement d'un prototype d'architecture hétérogène comme celle envisagée, l'utilisation d'une machine Aptix va permettre d'émuler les divers coprocesseurs connectés au DSP. Le fonctionnement de la palte-forme sera alors un fonctionnement en émulation pure, et non pas en cosimulation, réservée aux phases de développement. L'interface entre la machine

[11] Le point de vue exposé ici est volontairement simpliste : les machines Aptix sont parfaitement capables d'émuler des systèmes relativement complexes grâce à la multitude de modules différents pouvant être implantés sur la plate-forme : cœurs de processeur, mémoires, afficheurs LCD, pavés numériques, et bien sûr divers FPGAs.

[12] *Application Progamming Interface*, ensemble de fonctions documentées facilitant le développement d'applications

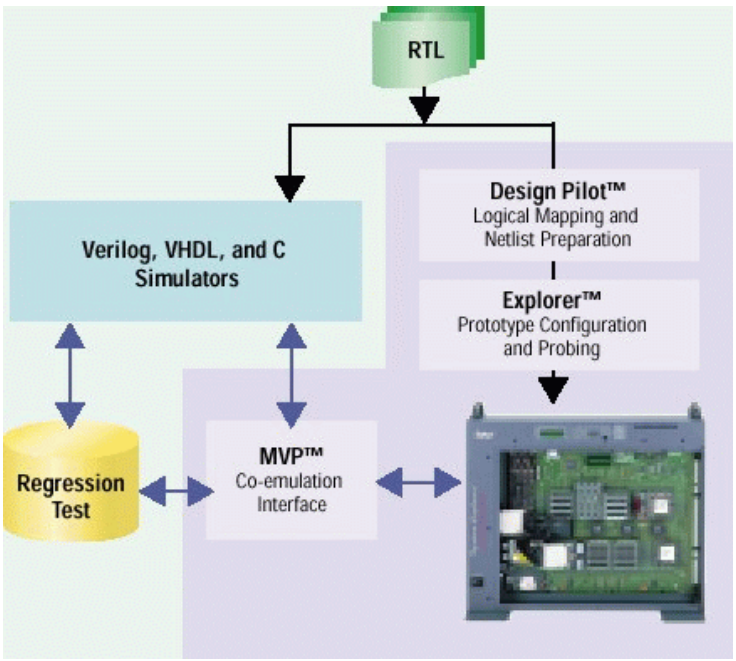


FIG. IV.4 – Utilisation en cosimulation d'une machine Aptix grâce à l'interface MVP

Aptix et la carte EV2 reste à définir et doit tenir compte des diverses possibilités de connexion de chacun des équipements.

IV.2 Implémentation logicielle sur ST120

Pour des raisons expliquées en détail ci-après, la méthodologie proposée au chapitre III n'a pas pu être appliquée dans sa totalité. Une implémentation « tout logiciel » de l'algorithme proposé a néanmoins été réalisée, de manière d'une part à identifier clairement les points critiques de l'implémentation de l'estimateur de canal proposé sur un DSP ST120 et d'autre part à évaluer la qualité de la chaîne de compilation associée au ST100. Diverses simulations effectuées avec cette implémentation entièrement logicielle ont permis d'obtenir des résultats intéressants exposés par la suite et de déterminer sans ambiguïté les principales fonctions chronophages à déporter vers le matériel.

IV.2.1 Incompatibilités entre N2C et les outils Green Hills

Une grave incompatibilité entre l'outil N2C et la chaîne de compilation du ST100 a été découverte peu de temps après la mise à disposition du kit d'intégration du DSP-MCU ST100 dans N2C. Ce problème, résolu dernièrement (novembre 2001), affectait les cosimulations effectuées entre le simulateur du ST100 et du code C tournant sur la machine hôte.

Le code exécuté par le simulateur du ST100 était à ce moment composé de deux sous-ensembles de routines : le code écrit par l'utilisateur, directement extrait des encapsulations composant le système, et le code fourni par le kit d'intégration, comme par exemple le code de boot, les squelettes des routines d'interruption non utilisées, ou encore certains pilotes de

périphériques. La différence majeure entre ces deux ensembles de routines est que les premières, extraites des descriptions fournies par l'utilisateur, étaient compilées à partir du code *source* contenu dans les encapsulations, tandis qu'une partie des secondes était fournie sous la forme de code *déjà compilé*. Les informations nécessaires lors des phases de debug, comme les numéros de ligne de source associés à certaine portions du code, par exemple, étaient apparemment corrompues lors de l'édition de liens, ce qui décorrélait totalement le code source affiché par le debugger du code réellement en cours d'exécution. Il était dans ces conditions impossible de débugger la partie logicielle des simulations autrement que directement au niveau de l'assembleur, l'affichage du code source étant erroné et empêchant en particulier la disposition de points d'arrêt à certains endroits.

Une solution provisoire à ce problème a été de déclencher l'affichage dans la console du debugger de la valeur de certaines variables à des instants précis par la commande classique `printf`. Envoyer les valeurs à afficher vers un bloc matériel simulé en C sur la machine hôte ne résolvait pas le problème et ne permettait pas de débugger proprement la partie *logicielle* de l'application. L'affichage des variables internes à la partie logicielle de l'application était donc invoqué de l'extérieur par une routine d'interruption. Outre le fait que l'exécution des `printf` prenait un certain temps et perturbait notablement le comportement de l'application, un deuxième problème a fait surface, lié cette fois-ci à un bug du compilateur. Ce bug se manifestait par des affichages erronés dans la plupart des cas, et a même perturbé l'affichage de simples constantes numériques. L'origine des perturbations subies par les affichages en question était située dans le compilateur qui ne gérait pas correctement le code destiné à servir des interruptions et entraînait des modifications intempestives des *paramètres* passés aux diverses fonctions appelées au sein de ces routines d'interruption.

L'appel à la fonction `printf`, consituant déjà une solution *de repli* en l'absence d'outil de debug fonctionnel, voyait donc ses paramètres *modifiés*, ce qui empêchait virtuellement toute possiblité de debug du code tournant sur le ST100 autrement qu'en déchiffrant les listings fournis par le désassembleur, seul organe de la châine de compilation à fournir des informations correctes. L'apprentissage de l'assembleur ST100 ne faisant pas partie des objectifs de cette thèse[13], la cosimulation ST100/C a été laissée de côté, et la décision de réaliser une implémentation entièrement logicielle de l'estimateur de canal proposé a été prise.

Depuis le mois de novembre 2001, la cosimulation ST100/C est à nouveau fonctionnelle, grâce à l'implémentation par CoWare d'un nouveau flot de génération de l'image logicielle. Chaque portion de code à compiler peut désormais être isolée dans son propre fichier `.c`, qu'elle provienne de la description du comportement des blocs par l'utilisateur ou du kit d'intégration du cœur de processeur. De cette manière, lorsque la directive est donnée au compilateur d'instrumenter le code pour permettre son debug au niveau source, la totalité de l'application est compilée avec les informations adéquates, et la totalité des fichiers sources impliqués est alors correctement affichée par le debuggeur, ce qui permet de placer des points d'arrêt où l'on veut.

IV.2.2 Résultats obtenus

Cette section a pour but de présenter rapidement les résultats obtenus à partir de simulations effectuées grâce à l'implémentation logicielle de l'algorithme proposé. Il ne s'agit pas ici d'une

[13]Il a cependant constitué une étape de l'implémentation entièrement logicielle de l'algorithme, ce qui n'était pas prévu au départ.

étude des performances de l'algorithme au sens « traitement du signal » du terme : une telle étude sera réalisée une fois le prototype développé. On s'intéressera plutôt aux caractéristiques logicielles de cette implémentation, à savoir les largeur de donnée utilisées, les temps d'exécution des routines de traitement du signal, ou encore à la détermination des fonctions dont le déport vers le matériel profiterait à l'amélioration des performances (réduction du temps d'exécution principalement).

IV.2.2.1 Portage en précision finie

Pour être implémentable de manière réaliste sur un DSP, il a fallu remanier l'algorithme proposé pour le faire calculer avec des nombres *entiers*, seul format compris *nativement* par le DSP. Les tailles de donnée utilisées et leur pertinence dans le cadre d'une implémentation sur le ST120 sont ainsi discutées ci-après.

La largeur des registres de donnée du ST120 est de 40 bits, ce qui permet de manipuler les formats numériques récapitulés sur le tableau IV.1. Les formats marqués N/A ne peuvent pas servir à *déclarer* une variable, mais sont utilisés dans la désignation des instructions pour considérer le contenu des registres concernés comme des nombres compactés ou fractionnaires. En C, le programmeur doit appeler les fonctions *intrinsèques* déjà mentionnées pour pouvoir bénéficier de ces formats.

Nom du type	Largeur	Nom du type en C
Octet	8 bits	`char`
Demi-mot	16 bits	`short`
Mot	32 bits	`int`
Mot long	40 bits	`long`
Octets compactés	2×8 bits	N/A
Demi-mots compactés	2×16 bits	N/A
Fractionnaire Q1.15	16 bits	N/A
Fractionnaire Q1.31	32 bits	N/A
Fractionnaire Q9.31	40 bits	N/A

TAB. IV.1 – Nom et largeur des types de données manipulés par le ST120

Les multiplications majoritairement effectuées par le DSP sont du type 16 bits $\times$16 bits $\rightarrow$ 40 bits, avec sélection possible des parties hautes ou basses des registres intervenant dans le produit.

La conversion de l'algorithme à l'emploi de nombres entiers a été effectuée selon le principe de la maximisation de l'utilisation des capacités de la machine, qui propose d'utiliser à chaque instant le type le plus large possible pour stocker les données tout en évitant toute possibilité de débordement de capacité pouvant résulter d'un calcul mal cadré. L'étude des *dégradations* induites par le passage en précisison finie pourra se faire une fois le prototype développé. De la même façon, on pourra retailler les données circulant au sein des opérateurs câblés pour limiter la complexité de ces opérateurs tout en maintenant un niveau de performances acceptable.

Les valeurs manipulées sont des nombres réels dont on considère une représentation entière sur un nombre donné de bits. Ainsi, on appliquera à chaque valeur réelle un coefficient

pondérateur recadrant la dynamique de la variable réelle à une dynamique supérieure, avant de transformer cette variable de plus grande dynamique en nombre entier. En pseudo-code, la formule utilisée est la suivante :

$$\text{Var. entière} = \text{Partie entière}(\text{Variable réelle} \times 2^N),$$

où N est le nombre de bits utilisé. La régénération du signal pilote, par exemple, se passe de la façon suivante :

1. Les parties réelles et imaginaires du train de chips sont régénérées par étalement et embrouillage des symboles pilotes. Le train de chips avant filtrage est composé de nombres appartenant à l'ensemble $\{+1, -1, +j, -j\}$. Les parties réelles et imaginaires de ces nombres sont chacunes codées sur 1 bit.
2. Le filtrage d'émission de ces chips pilotes se fait en ayant quantifié les coefficients du filtre en racine de cosinus surélevé sur 14 bits, de manière à pouvoir conserver le résultat de ce premier filtrage sur 16 bits. Le coefficient de pondération endossé par ces résultats est alors de 2^{14}.
3. Le filtrage de réception de ces chips pilotes se fait avec les mêmes coefficients, par l'intermédiaire d'un accumulateur sur 32 bits, redimensionné à 16 bits à la fin du traitement par recadrage. Le coefficient pondérateur du signal pilote régénéré est à nouveau de 2^{14}.

La recherche et le traitement des trajets se déroulent comme suit :

1. On considère tout d'abord que les parties réelles et imaginaires du signal reçu sont normalisées :
$$\forall t, \begin{cases} |\mathcal{R}\{s(t)\}| \leq 1 \\ |\mathcal{I}\{s(t)\}| \leq 1 \end{cases}$$
On peut ainsi quantifier le signal reçu sur un certain nombre de bits sans risquer de saturation. En réalité, ce point de vue est légitimé par les circuits de contrôle de gain, qui garantissent la non-saturation des convertisseurs analogiques-numériques à l'entrée de la partie numérique de récepteur. Les parties réelles et imaginaires du signal reçu sont quantifiées sur 16 bits, et endossent donc un coefficient de pondération de 2^{16}.
2. Le calcul de la corrélation entre les chips pilotes étalés et embrouillés mais non mis en forme fait intervenir de nombreuses accumulations/soustractions de nombres quantifiés sur 16 bits, et se fait à l'aide d'un accumulateur de 40 bits pour éviter tout risque de saturation. La division du résultat de la corrélation complexe par le nombre de chips pilotes ramène le résultat dans une gamme quantifiable sur 16 bits. Les parties réelles et imaginaires de l'atténuation complexe du trajet sélectionné sont donc quantifiés sur 16 bits, comme le signal reçu.
3. Après la validation du trajet identifié, il faut pondérer le signal pilote régénéré par l'atténuation complexe adéquate avant de le soustraire au buffer de travail contenant le signal reçu. Le produit complexe du signal pilote régénéré (coefficient de pondération 2^{14}) par l'atténuation complexe (coefficient de pondération 2^{16}), qui doit être ramené dans la dynamique du signal reçu pour que la soustraction effectuée ait un sens, est donc divisé par le coefficient de pondération du signal piloté régénéré.
4. La soustraction se fait entre des valeurs endossant le même coefficient de pondération, ce qui permet à l'algorithme de continuer sans avoir à effectuer un quelconque recadrage.

Les ressources de la machine sont utilisées à leur maximum, ce qui est appréciable pour une implémentation entièrement logicielle, tout en garantissant l'absence de saturation le long du chemin de données. On pourra toutefois diminuer la largeur des représentations entières utilisées lors de la connexion aux opérateurs câblés, de manière à limiter la taille et la complexité des composants instanciés par les outils de synthèse. Le prototype pourra alors servir de support à une étude des dégradations subies par les performances lors de la diminution des tailles d'entiers utilisées, ce qui est plus pénible à réaliser en cosimulation.

IV.2.2.2 Comportement temporel de l'application

Les paramètres dont dépend la complexité d'une itération de l'algorithme optimisé d'estimation de canal sont rappelés sur le tableau IV.2.

Paramètre	Description
N_{cp}	Nombre de chips pilotes, égal à une puissance de 2, compris entre 32 et 1024
DS	Largeur de la fenêtre de recherche des trajets, exprimée en chips, typiquement de l'ordre de quelques dizaines de chips
OSF	Facteur de suréchantillonnage en vigueur dans la chaîne de réception, typiquement égal à 4 ou 8

TAB. IV.2 – Paramètres dont dépend la complexité d'une itération de l'algorithme optimisé

Le déroulement d'une itération de l'algorithme suit globalement les étapes suivantes :

1. L'intercorrélation entre le signal reçu et les chips pilotes étalés et embrouillés est calculée.
2. Le module carré de cette intercorrélation est calculé.
3. Le retard du module maximum est déterminé, et l'atténuation complexe du trajet supposé présent à cet instant est déduite de la valeur de l'intercorrélation à cet instant.
4. La validité du trajet est testée.
5. En cas de validité du trajet, le signal pilote régénéré et mis en forme est pondéré par l'atténuation complexe estimée.
6. Le signal pilote pondéré est alors soustrait du buffer de travail.

L'implémentation réalisée a rassemblé les trois premières étapes en une seule fonction de recherche de pic de corrélation (complexité : $2(N_{cp}+2) \times DS \times OSF$ opérations réelles par itérations), de même que les deux dernières (complexité : $8 \times N_{cp} \times OSF$ opérations réelles par itération). Les temps d'exécution de ces deux fonctions ont été mesurés (en cycles d'horloge) pour quatre versions du programme :

- une version compilée avec la chaine de développement Green Hills, sans aucune optimisation, notée « GHS non optim. »,
- une version compilée avec cette même chaîne, optimisée de manière à minimiser le temps d'exécution de l'application, notée « GHS optim. »
- une version utilisant le LAO pour optimiser l'assembleur linéaire issu du compilateur Green Hills, notée « GHS + LAO »,

– une version où les routines critiques (pondération/suppression du signal pilote régénéré et recherche du trajet principal par corrélation) ont été écrites *à la main* en assembleur linéaire puis confiées au LAO, notée « Ass. lin. + LAO ».

Le procédé de mesure utilise deux des quatre compteurs/timers présents sur la carte EV2. Ces compteurs, larges de 16 bits, peuvent fonctionner indépendamment les uns des autres ou bien être chaînés les uns aux autres pour créer des compteurs virtuels plus larges que ne le sont les circuits eux-mêmes. Leur incrémentation/décrémentation peut être liée aux divers fronts de l'horloge, à des signaux externes, à d'autres périphériques de la carte ou encore donner lieu à une requête d'interruption. Pour chaque partie de l'itération dont on veut mesurer le temps d'exécution, les compteurs chaînés sont déclenchés avant l'appel de la fonction concernée et leur valeur est lue à sa sortie. Les résultats présentés ci-dessous ont été obtenus en moyennant les temps d'exécution de 4 itérations, la construction d'une statistique plus large étant ici relativement dépourvue de sens. En effet, les temps d'exécution des routines concernées dépendent beaucoup plus des *longueurs* des vecteurs traités que des vecteurs eux-mêmes[14]. Le tableau IV.3 donne les temps d'exécution mesurés pour les deux routines profilées, à savoir la recherche du trajet principal, notée « R », et la pondération/suppression du signal pilote régénéré, notée « P/S ». Les deux dernières lignes indiquent, en *opérations réelles*, la complexité algorithmique des phases correspondantes, à savoir $8 \times N_{cp} \times OSF$ opérations réelles pour la pondération/suppression, et $2(N_{cp}+2) \times DS \times OSF$ opérations réelles pour la recherche du trajet le plus puissant. Le facteur de suréchantillonnage a été pris égal à 4 pour des raisons de taille mémoire. Le *delay spread* a été pris égal à 60 chips. Les besoins en mémoire dans le cas où $N_{cp} = 1024$ excédant la taille des mémoires internes du ST120, ce cas de figure n'a pas été testé. Les temps d'exécution obtenus sont représentés figure IV.5 pour la recherche du trajet principal et figure IV.6 pour la pondération/suppression. Les performances de la version « GHS non optim. » n'apparaissent pas sur ces figures du fait de leur faible intérêt pratique.

	$N_{cp}=512$	$N_{cp}=256$	$N_{cp}=128$	$N_{cp}=64$	$N_{cp}=32$	
R	19481135	7574265	4920918	2494815	1281800	GHS non optim.
P/S	227432	113768	56936	28520	14312	
R	1595237	827589	444119	252561	157052	GHS optim.
P/S	49820	24706	12418	6272	3202	
R	667147	360457	207299	131306	93553	GHS + LAO
P/S	22618	11356	5722	2906	1500	
R	370650	186331	94171	48091	25051	Ass. lin. + LAO
P/S	14392	7224	3640	1848	952	
R	246720	123840	62400	31680	16320	Comp. *algorithmique*
P/S	16384	8192	4096	2048	1024	

TAB. IV.3 – Temps d'exécution des routines profilées, en cycles d'horloges, pour différentes valeurs de N_{cp} et différentes optimisations

[14]Cette hypothèse a été validée en calculant la dispersion des temps d'exécution mesurés, qui s'est avérée négligeable.

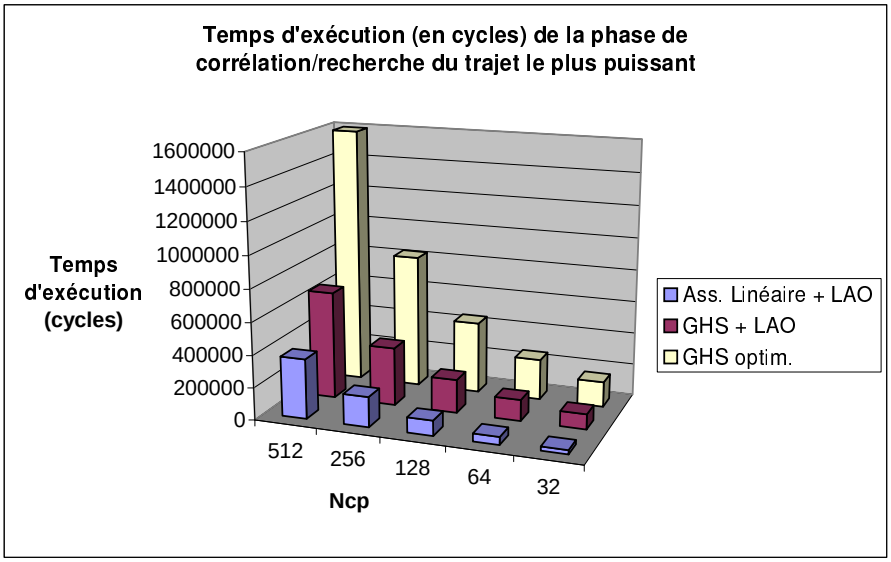


FIG. IV.5 – Temps d'exécution en cycles de la recherche du trajet principal en fonction de N_{cp} et du degré d'optimisation choisi

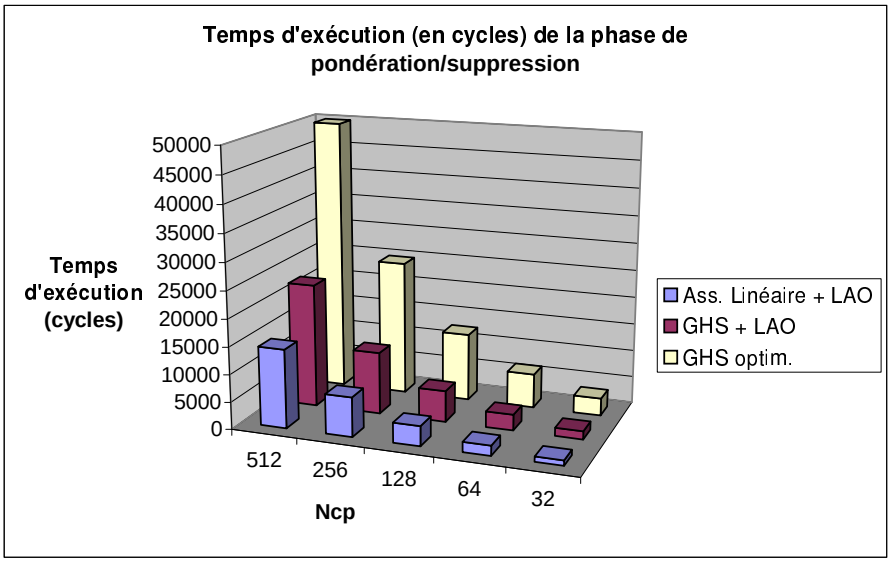


FIG. IV.6 – Temps d'exécution en cycles de la pondération/suppression du signal pilote régénéné en fonction de N_{cp} et du degré d'optimisation choisi

Ces résultats mènent à plusieurs observations :

1. Les temps d'exécution varient relativement *linéairement* avec N_{cp}, tout comme la complexité algorithmique, et ce quel que soit le degré d'optimisation des routines profilées.
2. Le temps d'exécution de la recherche non optimisée est *entre 8 et 12 fois* supérieur à celui de la même routine optimisée par le compilateur GHS, qui *ne génère pas de code SLIW*. On peut en déduire que l'absence d'optimisation ne doit être réservée qu'aux phases de debug potentiellement perturbées par le processus d'optimisation. Hors de ces cas particuliers, la non-optimisation est *absolument à proscrire*.
3. L'utilisation de l'assembleur linéaire et du LAO est la véritable clé ouvrant l'accès aux performances réelles de la machine. Après lecture de la documentation adaptée, la réécriture en assembleur linéaire des deux routines en question n'a pris que deux jours (écriture + test + mise au point) pour à peu près 150 lignes de code. Le temps d'exécution des routines concernées s'en est trouvé divisé par 5 (en moyenne) pour la recherche du trajet principal, et par 3,5 pour la pondération/suppression. La référence prise pour ces résultats est la version « GHS optim. », les mêmes rapports atteignant *50* pour la recherche et *15,5* pour la pondération/suppression dans le cas où l'on compare la version « Ass. lin. + LAO » à la version « GHS non optim. ».
4. Une solution intermédiaire entre la simple optimisation effectuée par le compilateur Green Hills et les optimisations plus agressives permises par la réécriture en assembleur linéaire des routines critiques et l'utilisation du LAO pour optimiser directement la sortie du compilateur Green Hills. De cette manière, le développeur n'écrit pas une ligne d'assembleur tout en bénéficiant néanmoins d'un gain de temps non négligeable. Les temps d'exécution sont cependant entre 1,5 et 4 fois plus importants que dans le cas d'une réécriture à la main. De meilleurs résultats sont peut-être atteignables en paramétrant plus finement le LAO, qui dispose d'un grand nombre de degrés de liberté dans son fonctionnement, ou en lui suggérant *à la main* certaines optimisations capables d'après le développeur d'améliorer sensiblement le comportement du programme, mais une compréhension approfondié de de l'outil est alors nécessaire. Les résultats présentés ici montrent ce qu'on peut tirer d'une utilisation « standard » du LAO, sans consigne particulière.
5. On est encore loin d'un éventuel fonctionnement en temps réel. En faisant fonctionner le cœur du ST120 à 200 MHz, trois itérations de l'algorithme de recherche du trajet principal dans le cas où $N_{cp} = 512$ prennent à peu près 5 ms, ce qui équivaut à *une demi-trame*. Envisager l'exécution complète de l'algorithme d'estimation à chaque slot est alors complètement illusoire. Si le comportement temporel de l'application ne peut pas être modifié, il faudra avoir recours à un dispositif de poursuite ou bien se priver d'estimation de canal une certaine partie du temps.
6. L'étude de complexité algorithmique montre à la fois ses qualités et ses défaut : le rapport entre le nombre de cycles pris par la recherche écrite en assembleur linéaire et le nombre d'opérations théoriquement effectuées par la routine en question vaut à peu près 1,5 pour toutes les valeurs de N_{cp}, et le même rapport pour la pondération/suppression vaut 0,9. Ces deux valeurs illustrent la pertinence particulière de l'étude de complexité algorithmique dans ce cas de figure, ainsi que la faible intrusion du code de contrôle dans les routines concernées. Par contre, le rapport de complexité entre la recherche et la pondération/suppression ne se retrouve pas du tout au niveau des temps d'exécution. Ainsi,

si le rapport des complexités algorithmiques vaut à peu près 15, le rapport des temps d'exécution vaut 26 pour la version écrite en assembleur linéaire, varie de 30 à 60 pour la version « GHS + LAO », entre 30 et 50 pour la version « GHS optim. » et tourne autour de 85 pour la version « GHS non optim.», ce qui illustre la faiblesse des optimisations effectuées. Ainsi, si la routine de pondération/suppression, plus simple, est relativement sous contrôle, la routine de recherche du trajet principal avec ses deux boucles imbriquées pose plus de problèmes au compilateur, ce qui explique l'explosion de son temps d'exécution par rapport à celui de la routine de pondération/suppression.

IV.2.3 Bilan

Le bilan de cette étude du comportement temporel de l'algorithme proposé sur le ST120 est double.

Tout d'abord, cette étude a permis de montrer sans ambiguïté le caractère fortement chronophage de la routine de recherche du trajet principal par corrélation, même dans le cas d'une routine complètement réécrite en assembleur et optimisée par le LAO. Cette fonction est le premier candidat à un déport vers le domaine matériel, où elle serait effectuée par un opérateur câblé activé par le DSP et ayant accès à au moins une partie de sa mémoire, justifiant ainsi la démarche de codesign proposée. Une autre fonction chronophage non présentée ici est la génération des codes d'étalement et des codes d'embrouillage. Le processus de génération des codes d'embrouillage, en particulier, est particulièrement inefficace s'il est implémenté de manière logicielle, alors qu'il ne coûte pratiquement rien s'il est implémenté en matériel. Du fait de son rythme d'activation (une fois par slot), son déport vers le matériel ne se justifie pas autant que pour la routine de recherche de trajet, mais le sujet reste néanmoins à approfondir.

D'autre part, cette étude a montré l'importance absolument *cruciale* de la chaîne de compilation associé à un DSP dès lors que celui-ci présente suffisamment de particularités architecturales pour lui permettre d'exploiter un certain parallélisme dans le flot du programme. Un sérieux problème se pose lorsqu'une architecture potentiellement performante comme celle du ST120 voit ses performances anéanties par la non-sophistication du compilateur. Le LAO vient combler le fossé existant entre les performances théoriques de la machine et celles du code compilé, mais son maniement efficace reste sujet à une connaissance approfondie de son fonctionnement et de la problématique de l'optimisation en général, et sa non-intégration à la chaîne de compilation standard vient perturber l'automatisation du processus. La réécriture des routines critiques en assembleur, malgré le gain de performances flagrant qu'elle procure, ne doit être envisagée qu'en dernier recours, du fait de la non-portabilité du code d'une part et de la maintenance plus difficle de telles portions de code au sein d'un projet majoritairement écrit en langage C d'autre part.

IV.3 Problèmes spécifiques au prototypage

Cette section a pour but de présenter les principaux problèmes susceptibles de se poser lors du passage d'une chaîne de cosimulation sous un enevironnement comme N2C à un prototype composé de cartes diverses et d'émulateurs HW. Dans une démarche de conception complète d'un circuit, relativement peu d'obstacles entravent la fabrication effective du circuit dès lors que l'interface synthétisée par l'outil utilisé est *elle-même* synthétisable. Par contre, dans une

démarche de prototypage rapide, qui vise l'utilisation de composants déjà existant sous la forme de cartes de développement ou d'émulateurs HW, un certain nombre de précautions doivent être prises pour assurer autant que possible la conformité du fonctionnement du prototype à celui de la chaîne de cosimulation. Trois de ces points critiques sont présentés ci-dessous.

IV.3.1 Position de l'interface HW/SW

Une fonctionnalité clé de l'outil N2C est la synthèse d'interface, qui permet de réduire drastiquement les temps de développement d'un système complexe sur une architecture hétérogène.

L'interface synthétisée par CoWare, composée à la fois d'opérateurs câblés synthétisables (décodeur d'adresse, encodeurs de priorité des requêtes d'interruption, convertisseurs de protocole ...) et de parties logicielles (routines d'interruption, routines d'accès aux périphériques par l'intermédiaire d'adresses en mémoire ...), n'est malheureusement pas *directement* transposable dans l'environnement de prototypage. En effet, elle est située très près du cœur du processeur ou du DSP, de manière à la rendre le plus efficace possible, et n'envisage pas la présence d'autres périphériques que ceux déclarés explicitement au sein du système, ce qui correspond au fonctionnement escompté de l'outil de synthèse d'interface.

Le problème vient du fait que sur les cartes de développement, le circuit présent n'est pas le cœur du processeur lui-même, mais plutôt un circuit *incluant* non seulement ce cœur, mais aussi plusieurs périphériques déjà connectés au processeur et contrôlables de manière logicielle. Les possibilités de connexion offertes par ces cartes incluant rarement les bus internes au processeur, il n'est pas possible d'utiliser sans modification l'interface générée par N2C. De plus, la présence d'autres périphériques au sein du système et surtout la prédéfinition de la carte de la mémoire du circuit sont incompatibles avec l'interface synthétisée automatiquement.

La carte EV2, par exemple, offre principalement deux connecteurs externes, l'un permettant à un autre circuit de prendre le contrôle complet de l'espace adressable par le ST120, l'autre permettant la connexion de périphériques esclaves du ST120. En particulier, le bus auquel sont raccordés les opérateurs câblés dans la chaîne de cosimulation ne sont *pas* accessibles par ces connecteurs. C'est le deuxième connecteur (le connecteur « système » présenté plus haut) qui correspond le plus à l'application envisagée ici. Ce deuxième connecteur permet en particulier de se raccorder à l'*EMI* (*External Memory Interface*, interface aux mémoires externes), à laquelle sont déjà raccordées les deux mémoires présentes sur la carte (Flash et SRAM). Cette interface entre le ST120 et l'extérieur respecte un protocole certes documenté mais non compris nativement par l'outil N2C. Pour assurer le bon déroulement des communications entre le DSP et les périphériques non présents sur la carte, il est nécessaire de se conformer au protocole de cette interface, et donc d'insérer un convertisseur aussi rapide que possible entre les protocoles de communication utilisés par le périphérique tel qu'il a été conçu en cosimulation et le protocole de cette interface[15].

Pour assouplir dans la mesure du possible le passage de la cosimulation au prototype, il est donc souhaitable d'une part de *fixer* les adresses des périphériques à des valeurs compatibles avec celles déjà utilisées par la carte et d'autre part de prêter la plus grande attention aux

[15]Une fois cette manipulation effectuée et la démarche maîtrisée, il peut devenir plus intéressant « d'inverser » le convertisseur en question et de l'incorporer à la chaîne de cosimulation, de manière à ce que sa *suppression* lors du passage au prototype réel permette au périphérique d'utiliser directement l'EMI.

protocoles utlisés par les périphériques destinés à être raccordés aux bus offerts par les cartes utilisés.

Une autre solution à ce problème réside dans une nouvelle tendance dans le design au niveau système : le *Platform-Based Design*, ou design basé sur une plate-forme [CoW00a] [CoW00b]. Cette démarche vise la fourniture d'une « base » sur laquelle bâtir des développements futurs. Cette base peut être composée d'un cœur de processeur déjà instancié, de mémoires, éventuellement d'un certain nombre de périphériques conenctés au cœur, et d'un certain nombre de connexions vers l'extérieur.

Plusieurs types de plate-forme sont envisageables :

- la plate-forme dénuée de connexions vers l'extérieur, sur laquelle l'utilisateur ne va développer *que* le logiciel, et éventuellement contrôler de cette manière les périphériques présents sur la plate-forme,
- la plate-forme sur laquelle la partie logicielle est déjà écrite, à laquelle l'utilisateur va connecter des périphériques divers assurant des fonctions appelées par le logiciel tournant sur le cœur,
- la plate-forme dont l'utilisateur va modifier les aspects logiciel *et* matériel, en réécrivant des parties du code éventuellement fourni et en concevant des périphériques à raccorder à la plate-forme.

Le design basé sur une plate-forme peut être considéré comme un équivalent « virtuel » (et plus souple) de la carte de développement associée à un circuit : l'utilisateur de cette plate-forme va embarquer du code et connecter des périphériques *en simulation* de la même manière que l'utilisateur d'une carte de développement va télécharger son code et y connecter son émulateur HW pour valider ses périphériques. Cette méthode favorise la réutilisation d'IP et permet de disposer pour de nombreux développements d'une base modifiable rapidement, et « spécialisable » selon les applications envisagées.

L'utilisation de cette méthode de conception dans notre cas consisterait à recréer sous N2C un modèle de la carte EV2 incluant le ST120, éventuellement les quelques périphériques inclus dans le circuit[16], les mémoires externes, les lignes d'interruptions permettant de détourner le fonctionnement du ST120, et surtout l'EMI connectant le cœur, les mémoires et l'extérieur. De plus, il faut figer une partie de la carte de l'espace adressable du ST120 de manière à faire correspondre les adresses de ces divers composants sur la plate-forme virtuelle à leur adresse réelle sur la carte EV2. De cette manière, il est possible de valider à l'aide de l'outil N2C la connexion des périphériques développés par l'utilisateur de la plate-forme et de garantir leur bon fonctionnement lors du passage au prototype.

Une telle recréation d'une carte existante sous forme de plate-forme virtuelle nécessitant un effort conséquent, cette fonctionnalité pourtant prometteuse n'a pas été explorée au cours de cette thèse. Cependant, sa maîtrise semble absolument nécessaire à l'établissement d'un environnement sophistiqué de prototypage rapide.

IV.3.2 Emplacement des mémoires

Un autre problème pouvant survenir lors du passage d'une chaîne de cosimulation ne faisant intervenir que relativement peu de composants à part le cœur lui- même et les périphériques

[16] Cette étape n'est nécessaire que si le code exécuté par le ST120 fait effectivement appel à ces périphériques.

dédiés à l'application à un réel environnement de prototypage est la détermination des mémoires utilisées.

Qu'on se place en simulation ou dans l'environnement de prototypage, plusieurs types de mémoire peuvent en effet être utilisés pour le stockage des données et le passage de données ou de résultats du domaine logiciel au domaine matériel et inversement :

- les mémoires internes au DSP, de taille relativement faible, sont certes accessibles rapidement, mais leur accès depuis l'extérieur du cœur nécessite l'interruption de son fonctionnement,
- les mémoires externes telles que la Flash ou la SRAM présentes sur la carte EV2 sont beaucoup plus importantes (2 fois 32 kilo-octets de mémoire interne au DSP contre 2 méga-octets de SRAM et 512 kilo-octets de Flash) et leur accès n'empêche pas le fonctionnement du DSP, mais elles sont plus lentes, et leur accès nécessite l'activation de plusieurs composants entre le cœur et la mémoire elle-même,
- les mémoires « double-port », permettant à deux autres composants d'accéder simultanément à deux lignes de mémoire différentes, ont l'inconvénient d'être sensiblement plus coûteuses que les mémoires classiques, et ne sont en conséquence que rarement présentes sur des cartes de développement. Par contre, on en trouve souvent une faible quantité au sein des systèmes multi-cœurs.

Dans le cadre de l'application envisagée ici, l'utilisation des mémoires internes au DSP n'est pas envisageable. En effet, le but avoué du déport de la recherche de trajet vers le domaine matériel étant de soulager le DSP de manière à lui permettre d'exécuter une autre tâche pendant ce temps-là, le bloquer pendant l'exécution de la routine en question n'est pas très intéressant.

La mémoire choisie pour la réalisation du prototype est la SRAM présente sur la carte EV2. C'est la plus rapide des deux mémoires externes, et sa taille est largement suffisante pour stocker non seulement les données nécessaires à l'estimation du canal mais aussi les résultats de la démodulation des données une fois que celle-ci sera implémentée. L'accès à cette mémoire nécessitant l'utilisation de l'EMI, le DSP ne pourra pas y accéder pendant le travail des opérateurs câblés. Le concepteur devra donc prendre garde à rapatrier les données éventuellement utilisées par le DSP pendant la recherche *avant* le déclenchement de celle-ci.

De plus, le déport de la phase de pondération/suppression ne se justifiant pas au vu de l'efficacité de son implémentation sur le DSP, il faudra examiner la manière dont le DSP va modifier le buffer contenant originellement le signal reçu. Selon les différences de temps d'accès entre les mémoires internes au DSP et la SRAM, il pourra être plus avantageux de rapatrier la partie de SRAM à modifier en mémoire interne *avant* de la modifier que de modifier directement la SRAM, même si les résultats devront inévitablement être renvoyés dans la SRAM. Cette décision faisant intervenir la fréquence finale du fonctionnement du DSP, qui n'a pas encore été choisie, ainsi que les temps d'accès à la SRAM *en fonction* non seulement de la fréquence du DSP mais aussi de la fréquence du bus supportant l'EMI, elle a été remise à plus tard.

IV.3.3 Vitesse de l'émulation HW

Le troisième problème qui sera mentionné ici provient de la vitesse de l'émulateur HW utilisé pour le prototype et des éventuelles perturbations du comportement temporel de l'application qui peuvent en résulter.

La vitesse maximale de la plate-forme Aptix qui sera probablement utilisée pour la réalisation du prototype n'étant « que » de 40 MHz, il est possible que cete vitesse ne suffise pas à émuler correctement le fonctionnement des opérateurs câblés. Il faut alors prendre garde lors de l'examen des performances du prototype au décalage entre la fréquence de fonctionnement *souhaitée* des périphériques à connecter au cœur et la fréquence *réelle* (probablement inférieure) de la machine *émulant* ces périphériques, connectée non pas au cœur mais à un bus *externe* au cœur.

IV.3.4 Bilan

Le passage d'un flot de codesign à un environnement de prototypage rapide n'est certes pas indolore, mais il est possible de faciliter les choses en prenant garde principalement au passage d'une interface synthétisée « idéale », rapide et située près du cœur, à une interface imposée par les composants utlisés, nécessairement plus générique et donc plus lente.

Dans la majorité des cas, le nombre de composants venant s'intercaler entre le cœur et les périphériques et ralentir les échanges entre les domaines matériel et logiciel fait que les performances du prototype seront la plupart du temps inférieures aux performances idéales du système sur une puce. Le type de mémoire utilisé, le moyen de s'interfacer à cette mémoire, la vitesse de l'émulateur sont autant de facteurs venant dégrader les performances du prototype sans impacter le système virtuel conçu : seule la réalisation d'un *prototype* à l'aide de cartes de développement et d'émulateurs fait surgir ces difficultés.

Conclusion

La conception d'un système embarqué hétéreogène, traditionnellement effectuée par deux équipes distinctes travaillant séquentiellement avec des outils et des langages différents, vit depuis quelques années une véritable révolution.

Le System-On-Chip, circuit intégrant cœur(s) de processeur, périphériques, mémoires et accélérateur(s) matériel(s), vient faire voler en éclats la conception micro-électronique classique, dont les limties sont présentées au début du chapitre III. Les nouveaux flots de conception, dits "au niveau système", accélèrent le développement de tels circuits et intègrent de nouveaux outils tels que la cosimulation ou la synthèse d'interface. Une méthodologie exploitant un tel flot, basée sur l'utilisation de l'outil N2C, de CoWare, Inc., est présentée dans la seconde moitié du chapitre III.

Cette méthodologie, illustrée par un exemple simple, peut être divisée en trois étapes correspondant globalement aux trois niveaux de description utilisés :

- description comportementale du système, sans aucune considération temporelle,
- instanciation des cœurs de processeur, temporisation de la partie *logicielle* de l'application, partitionnement et cosimulation C/ISS,
- temporisation du système complet, synthèse et cosimulation VHDL/ISS.

L'environnement de prototypage, composé d'une carte de développement du DSP-MCU ST120 et d'un émulateur matériel Aptix, est présenté en détail au début du chapitre IV. Cet environnement sert de support à une implémentation purement logicielle de l'algorithme proposé, l'implémentation hétérogène prévue au départ n'ayant pas pu être réalisée du fait de sévères incompatibilités entre la chaîne de compilation du ST100 et l'outil N2C.

L'étude des performances de cette implémentation logicielle a permis d'exhiber des lacunes majeures de la chaîne de développement du ST100. Seule l'utilisation du LAO permet en effet d'entrevoir les réelles performances de la machine, qui même avec des routines optimisées à la main semble difficilement pouvoir exécuter l'algorithme proposé en temps réel.

Trois points critiques ont été identifiés lors du passage (non réalisé ici) d'une chaîne de cosimulation à un prototype réel. Ces points sont :

- le passage d'une interface idéale située très près du cœur à une interface externe, plus lente, ne respectant pas les mêmes protocoles,
- le choix des mémoires stockant les données et les résultats des traitements effectués,
- la vitesse de l'émulateur matériel utilisé, qui constitue probablement le principal facteur limitant les performances du prototype en regard du System-On-Chip équivalent (ce point est à approfondir).

Synthèse et perspectives

Ce travail de thèse se situant à cheval sur les deux domaines que sont le traitement du signal et la conception de systèmes sur silicium, il peut être intéressant de tirer du travail réalisé un bilan spécifique à chaque domaine dans un premier temps, avant d'en tirer par la suite un bilan conjoint.

Bilan de la partie « traitement du signal »

Après avoir présenté brièvement les apports de l'UMTS par rapport aux systèmes des générations précédentes, le lien descendant d'une de ses interfaces radio, l'UMTS-FDD (aussi appelé par abus de langage Wideband CDMA, ou CDMA Large Bande), a été étudié en détail.

Les diverses dégradations du signal émis induites à la réception par la progatation dans le canal radiomobile ayant été introduites, l'accent a été mis sur le rôle du récepteur embarqué au sein du terminal (classiquement un récepteur Rake), qui doit pour être efficace être capable d'identifier précisément les instants d'arrivée des trajets multiples ainsi que leurs amplitudes complexes. Cette estimation est classiquement effectuée par un organe distinct du récepteur Rake appelé Searcher. Traditionnellement délaissé par la littérature, plus fournie sur l'égalisation en général que sur l'estimation de canal proprement dite, le Searcher représente pourtant une étape algorithmique non négligeable dans la chaîne de réception WB-CDMA. Sa complexité est d'ailleurs prépondérante dans cette même chaîne. Un algorithme d'estimation itérative du canal à suppression de trajet a été proposé avec des résultats satisfaisants. La complexité importante de cet algorithme le rendant inembarquable, une version optimisée de cet algorithme a été développée. Les intercorrélations effectuées par cette version de l'algorithme se font non pas entre deux signaux mis en forme, mais entre un signal mis en forme (le signal reçu) et un signal non mis en forme (les chips pilotes régénérés à la réception). Les estimateurs des modules et phases des trajets sont ainsi moins précis, tandis que la précision de l'estimateur des instants d'arrivée reste inchangée. Les quelques simulations effectuées indiquent que les deux versions de l'algorithme exhibent sensiblement le même comportement, ce qui semble conférer un caractère relativement « inoffensif » à l'optimisation proposée.

Un critère innovant d'arrêt des itérations a été proposé, basé sur le nombre de symboles pilotes correctement démodulables sur chaque trajet potentiellement identifié par le Searcher. Un tel critère, destiné à être utilisé conjointement à des critères plus classiques de seuillage de module, permet d'éviter de prendre en compte les faux trajets lors du processus de recombinaison, et représente un autre angle d'attaque du problème de l'identification des faux trajets, particulièrement lors de l'utilisation de séquences pilotes courtes. Les problèmes liés à la définition de la politique de gestion de ce critère ont été soulevés lors de la comparaison des performances des deux versions de l'algorithme. Le coût algorithmique du calcul de ce critère a été évalué et peut être considéré comme négligeable devant la complexité d'une itération du Searcher, le

rendant donc immédiatement embarquable.

Enfin, l'optimisation proposée supprime une grande partie des multiplications complexes non triviales effectuées par l'algorithme lors du calcul de l'intercorrélation entre le signal reçu et le signal pilote régénéré à la réception, les remplaçant par des accumulations de produits triviaux du type *nombre réel* × *bit*, sûrement effectués plus efficacement par un opérateur câblé que par une architecture programmable de type DSP. L'adoption de l'optimisation proposée a conduit à une redéfinition de l'architecture des organes de calcul du récepteur, le rendant plus adapté à une implémentation sur une architecture hybride que sur une pure architecture programmable.

Bilan de la partie « conception »

La complexité des applications embarquées et la variété des architectures susceptibles d'accueillir ces applications ont précipité l'obsolescence des outils et des méthodologies classiques de conception.

De nouveaux outils dits de conception « au niveau système » sont apparus vers la fin des années 90, facilitant l'exploration architecturale à l'aide de technologies innovantes telles que la cosimulation, l'utilisation de langages de haut niveau en lieu et place des traditionnels VHDL ou Verilog, la réutilisation d'IP ou encore la synthèse automatique d'interface.

Afin de développer un prototype de récepteur implémentant l'algorithme optimisé proposé dans la première partie, une méthodologie innovante de conception au niveau système a été formalisée, en tenant compte de la forte orientation « signal » de l'application à embarquer. Ainsi, les prérequis spécifiques à cette classe d'applications que sont la simulation en précision finie et l'étude de complexité ont été décrits en détail, ainsi que le type d'information qu'on peut en extraire. La méthodologie proposée s'appuie sur une temporisation progressive du système simulé. Le concepteur passe ainsi d'un modèle transactionnel du système à un modèle dont seule la partie logicielle est temporisée, avant de passer à un modèle complètement temporisé, aussi bien au niveau des parties logicielles et matérielles qu'au niveau des interfaces entre les deux domaines.

Cette méthodologie utilise le logiciel N2C, de CoWare, Inc., et s'appuie particulièrement sur la synthèse d'interface offerte par l'outil pour réduire le temps d'itération lors de l'exploration architecturale et ainsi raccourcir la durée de cette exploration en permettant l'évaluation rapide d'un nombre élevé de stratégies de partitionnement. L'outil N2C permettant une grande variété dans les niveaux d'abstraction des différents blocs composant le système, la méthodologie proposée restreint son utilisation à un certain nombre de phases clairement définies permettant d'aboutir rapidement au partitionnement final, puis d'affiner en un minimum d'itérations les descriptions des interfaces et des parties matérielles du système en cours de développement.

L'environnement de prototypage envisagé est composé d'une carte de développement du DSP ST120 de STMicroelectronics et d'une machine Aptix pour émuler les parties matérielles du système. Après avoir présenté ces deux organes, les outils associés et leurs possibilités d'interconnexion, une discussion a été menée concernant leur éventuelle intégration à la méthodologie proposée et à l'outil N2C. Une incompatibilité majeure entre la chaîne de compilation du ST120 et l'outil N2C a empêché l'application envisagée d'être prototypée conformément à la méthodologie proposée, qui n'a ainsi pas été validée sur une application réelle. L'adéquation de la méthodologie proposée à un tel environnement a néanmoins été évaluée, mais de manière qualitative uniquement. Certains problèmes spécifiques au prototypage ont été relevés, tels que

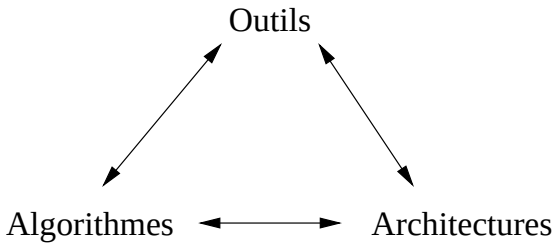
l'inadéquation des interfaces générées par CoWare à l'environnement de prototypage ou encore le choix des mémoires utilisées par l'application.

Une implémentation purement logicielle d'une version en précision finie de l'algorithme développé dans la première partie a été réalisée, permettant ainsi de confirmer certaines conclusions tirées des études de complexité menée lors de son développement. Cette phase de développement a permis de mettre en évidence les lacunes de la chaîne de compilation du ST100, qui n'offre pas l'accès aux véritables performances de la machine. Ce point a pemis de souligner l'importance des outils associés à ces organes de calcul à hautes performances que sont les cœurs de DSP à plusieurs unités d'exécution.

Bilan conjoint

Ce travail de thèse représente une tentative de développement quasi-conjoint d'un algorithme et d'une architecture susceptible d'embarquer cet algorithme de manière pertinente dans le cadre de l'application choisie, tout en formalisant de concert la méthodologie employée. Ainsi, l'algorithme a été développé et optimisé dans le but constant de minimiser sa complexité et par là même son temps d'exécution, l'architecture proposée a été partitionnée de manière à ne pas faire exécuter par le DSP des calculs réalisables à moindre coût par un opérateur câblé, et la méthodologie proposée l'a été dans le but de pouvoir prototyper rapidement l'algorithme en question sur l'architecure en question, et ce avec des composants mimant relativement fidèlement la véritable architecture d'accueil d'une telle application, c'est-à-dire un terminal mobile de troisième génération.

Cet état d'esprit relativement peu académique a néanmoins permis de conserver une certaine ouverture d'esprit lors de cette thèse et ainsi d'appréhender le problème posé sous plusieurs angles d'attaque avant de proposer un début de solution empruntant à plusieurs disciplines habituellement distinctes. Ni totalement focalisé sur la finesse architecturale ou sur la performance algorithmique, ni complètement dévoué au développement ou à l'étude d'outils facilitant le passage de l'algorithme à l'architecture ou améliorant l'adéquation de l'architecture à l'algorithme, le présent travail de thèse espère étendre la démarche classique de l'adéquation algorithme-architecture pour y incorporer une composante « outil », de manière à former le paradigme représenté ci-dessous.



S'éloigner ainsi volontairement des travaux purement algorithmiques ou purement architecturaux empêche certes d'explorer totalement l'ensemble des solutions proposées par chaque discipline à la partie du problème concernée, mais cet aspect est contrebalancé par l'apport que constituent les différentes propositions innovantes formulées dans cette thèse, que ce soit dans un domaine ou dans l'autre.

Perspectives

Ce travail laisse un certain nombre de questions sans réponse, parmi lesquelles la validité de la méthodologie proposée et les véritables performances de l'algorithme optimisé.

Il est certes regrettable qu'une partie du travail effectué n'ait pas été validée malgré le caractère prometteur de la méthodologie innovante proposée, mais ce point figure en première place sur la liste des travaux à mener après la conclusion de cette thèse. L'incompatibilité logicielle à l'origine du problème ayant été levée à la fin de l'année 2001, il est désormais possible d'appliquer la totalité de la méthodologie proposée et de vérifier ainsi sa validité dans le cadre d'une application « signal » d'une part, et dans le cadre d'un environnement de prototypage rapide d'autre part. La méthodologie à appliquer étant formalisée dans le présent mémoire, ce travail relève désormais plus du développement que de la recherche, et devrait rapidement permettre de juger de la véritable pertinence de la méthodologie proposée.

Sur le plan algorithmique, quelques autres optimisations sont restées en suspens, parmi lesquelles l'élimination du recalcul de l'intercorrélation à chaque itération. Celui-ci pourrait en effet être avantageusement remplacé par la soustraction à la corrélation déjà calculée de l'autocorrélation du signal pilote régénéré pondérée par l'atténuation complexe estimée. Le calcul de cette autocorrélation étant rendu non-trivial par l'optimisation proposée, cette piste n'a pas encore été explorée, mais devrait l'être dès que le prototype aura été développé et validé. La gestion du critère d'arrêt des itérations fait aussi l'objet de nombreuses supputations, et sera simulée plus précisément une fois que le prototype aura été développé.

Sur le plan méthodologique, les pistes restant à explorer sont l'intégration du lien MVP sous CoWare, qui pourrait accélérer considérablement les cosimulations des phases finales de la méthodologie proposée, et la possible automatisation du processus de génération d'une interface conforme à l'interface EMI de la carte EV2 grâce à la fonctionnalité de « design basé plateforme » (Platform-Based Design) de l'outil N2C.

Une dernière piste de recherche soulevée par cette thèse est l'intégration au plus tôt dans le flot méthodologique des spécificités du prototypage. Le présent travail de thèse s'étant contenté d'énoncer les points critiques du passage de la cosimulation au prototypage et de proposer certaines solutions à ces problèmes, il pourrait être intéressant d'étudier comment la finalité de la démarche pourrait influer sur les premières étapes de la méthodologie de manière à ne pas créer les problèmes mentionnés plus haut.

Bibliographie

[3GP01a] 3GPP. « TS 25.101 : UE Radio Transmission and Reception (FDD) ». Spécification technique, 3rd Generation Partnership Project, Sophia Antipolis, France, Mar. 2001.

[3GP01b] 3GPP. « TS 25.211 : Physical Channels and Mapping of Transport Channels onto Physical Channels (FDD) ». Spécification technique, 3rd Generation Partnership Project, Sophia Antipolis, France, Mar. 2001.

[3GP01c] 3GPP. « TS 25.212 : Multiplexing and Channel Coding (FDD) ». Spécification technique, 3rd Generation Partnership Project, Sophia Antipolis, France, Mar. 2001.

[3GP01d] 3GPP. « TS 25.213 : Spreading and Modulation (FDD) ». Spécification technique, 3rd Generation Partnership Project, Sophia Antipolis, France, Mar. 2001.

[3GP01e] 3GPP. « TS 25.214 : Physical Layer Procedures (FDD) ». Spécification technique, 3rd Generation Partnership Project, Sophia Antipolis, France, Mar. 2001.

[APL98] Gunther Auer, Gordon J. R. Povey, et David I. Laurenson. « Mobile Channel Estimation for Decision Directed RAKE Receivers Operating in Fast Fading Radio Channels ». *Proceedings of the IEEE Fifth International Symposium on Spread Spectrum Techniques and Applications (ISSSTA)*, Sun City, South Africa, Sep. 1998.

[Apt00a] Aptix, Module Verification Platform Data Sheet. Aptix Corporation, 2000. *http://www.aptix.com/nova/literature/techpubs/MVP.pdf*

[Apt00b] Aptix, System Explorer - Reconfigurable System Prototyping for SoC Emulation. Aptix Corporation, 2000. *http://www.aptix.com/nova/literature/brochures/sysExpl.pdf*

[Are00] Arexsys, CosiMate. Arexsys, 2000. *http://www.arexsys.com/products/Cosimate.pdf*

[ASA98] Hidehiro Andoh, Mamoru Sawahashi, et Fumiyuki Adachi. « Channel Estimation Filter Using Time-Multiplexed Pilot Channel for Coherent RAKE Combining in DS-CDMA Mobile Radio ». *IEICE Transactions on Communications*, vol. E81-B, num. 7, pp. 1517–1526, Juil. 1998.

[ASO97] F. Adachi, M. Sawahashi, et K. Okawa. « Tree-Structured Generation of Orthogonal Spreading Codes with Different Length for Forward Link of DS-CDMA Mobile Radio ». *Electronics Letter*, vol. 33, num. 1, pp. 27–28, Jan. 1997.

[BA96] Stephen E. Bensley et Behnaam Aazhang. « Subspace-Based Channel Estimation for Code Division Multiple Access Communication Systems ». *IEEE Transactions on Communications*, vol. 44, num. 8, pp. 1009–1020, Août 1996.

[BA98] Stephen E. Bensley et Behnaam Aazhang. « Maximum-Likelihood Synchronisation of a Single User for Code-Division Multiple-Access Communication Systems ». *IEEE Transactions on Communications*, vol. 46, num. 3, pp. 392–399, Mar. 1998.

[Bat01] Eric Batut. « A Low Complexity Channel Estimation Scheme for the WB-CDMA Terminal ». *Proceedings of the Third IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 74–77, Taoyuan, Taiwan, Mar. 2001.

[BB99] Sergio Benedetto et Ezio Biglieri. *Principles of Digital Transmission, with Wireless Applications*. Kluwer Academic / Plenum Publishers, 1999.

[Ber00] Gérard Berry, The Esterel Language Primer. CMA, Ecole des Mines de Paris, Juil. 2000. *ftp://ftp.esterel.org/esterel/pub/papers/primer.ps*

[BO98] C. Boulanger et L. Ouvry. « Tabu Search : An Efficient Tool for Designing DS-CDMA Spreading Sequences ». *Proceedings of the IEEE Fifth International Symposium on Spread Spectrum Techniques and Applications (ISSSTA)*, Sun City, South Africa, Sep. 1998.

[BSRC98] Gregory E. Bottomley, Essam Sourour, Rajaram Ramésh, et S. Chennakeshu. « Optimizing the Performance of Limited Complexity Rake Receivers ». *Proceedings of the 48th IEEE Vehicular Technology Conference, 1998 Fall Edition*, pp. 968–972, 1998.

[BwT98] Anders Berkeman, Viktor Öwall, et Mats Torkelson. « A Low Logic Depth Complex Multiplier ». *Proceedings of the 24th IEEE European Solid-State Circuits Conference (ESSCIRC)*, pp. 204–207, La Hague, Pays-Bas, Sep. 1998.

[Cad01a] Cadence, Cadence Signal Processing Worksystem (SPW). Cadence Design Systems, Inc, Sept. 2001. *http://www.cadence.com/datasheets/dat_pdf/spw090601.pdf*

[Cad01b] Cadence, Cadence Virtual Component Co-Design (VCC). Cadence Design Systems, Inc, Mai 2001. *http://www.cadence.com/datasheets/dat_pdf/vcc.pdf*

[Cha01] Jean-Pierre Charles. « 3^{rd} Generation Mobile Systems UMTS/IMT-2000 ». *Annales des Télécommunications*, vol. 56, num. 5-6, pp. 229–235, Mai-Juin 2001.

[Cla68] R. H. Clarke. « A statistical theory of mobile radio reception ». *Bell System Technical Journal*, vol. 47, pp. 957–1000, Juil. - Août 1968.

[CoW00a] CoWare, Flexible Platform-Based Design With the CoWare N2C Design System. CoWare, Inc., Oct. 2000. *http://www.coware.com/pdf/pbdWhitepaper.pdf*

[CoW00b] CoWare, Platform-Based Design Technical Backgrounder. CoWare, Inc., Juil. 2000. *http://www.coware.com/pdf/pbdBackgrounder.pdf*

[CoW01] CoWare, CoWare N2C Design System. CoWare, Inc, 2001. *http://www.coware.com/pdf/n2c.pdf*

[Den96] H. DENG. « Synthesis of binary sequences with good autocorrelation and crosscorrelation properties by simulated annealing ». *IEEE Transactions on Aerospatial and Electronic Systems*, vol. 32, num. 1, pp. 98–107, Jan. 1996.

[Den98] Steve DENNETT. « The cdma2000 ITU-R RTT Candidate Submission ». Document TR45.5/98.04.03.03, Telecommunications Industry Association, Avr. 1998.

[DJ98] Esmael H. DINAN et Bijan JABBARI. « Spreading Codes for Direct Sequence CDMA and Wideband CDMA Cellular Networks ». *IEEE Communications Magazine*, pp. 48–54, Sep. 1998.

[Ela01] ELANIX, SystemView User's Guide. Elanix, 2001. *http://www.elanix.com/pdf/SVUGuide.pdf*

[ETS98] ETSI. « Chapter B.1.4.2 : Channel Impulse Response Model ». Rapport Technique UMTS 30.03 TR101.112, V3.2.0, ETSI, Avr. 1998.

[FG99] Romano FANTACCI et Andrea GALLIGANI. « An Efficient RAKE Receiver Architecture with Pilot Signal Cancellation for Downlink Communications in DS-CDMA Indoor Wireless Networks ». *IEEE Transactions on Communications*, vol. 47, num. 6, pp. 823–827, Juin 1999.

[Fox01] Chuck FOX. « The Coming Era of Field Programmability ». *Proceedings of DesignCon*, Santa Clara, CA, USA, 2001.

[FSA98] Satoru FUKUMOTO, Mamoru SAWAHASHI, et Fumiyuki ADACHI. « Matched Filter-Based RAKE Combiner for Wideband DS-CDMA Mobile Radio ». *IEICE Transactions on Communications*, vol. E81-B, num. 7, pp. 1384–1391, Juil. 1998.

[GJP+91] K. S. GILHOUSEM, I. M. JACOBS, R. PADOVANI, A. J. VITERBI, L. A. WEAVER, et C. E. WHEATLEY. « On the Capacity of a Cellular CDMA System ». *IEEE Transactions on Vehicular Technology*, vol. 40, Mai 1991.

[Gol67] R. GOLD. « Optimal Binary Sequences for Spread Spectrum Multiplexing ». *IEEE Transactions on Information Theory*, vol. IT-B, pp. 619–621, Oct. 1967.

[Gol68] R. GOLD. « Maximal Recursive Sequences with 3-Valued Recursive Cross-Correlation Functions ». *IEEE Transactions on Information Theory*, vol. IT-4, pp. 154–156, Jan. 1968.

[Gol92] S. W. GOLOMB. *Shift Register Sequences*. Aegean Park Press, 1992.

[Gom01] Cristina GOMILA. « *Mise en Correspondance de Partitions en vue du Suivi d'Objets* ». Thèse de doctorat, Ecole des Mines de Paris, Paris, 2001.

[Gré] GRÉSILOG, Le manuel utilisateur MUSTIG. Grésilog. *http://www.gresilog.com/francais/doc_html/index.htm*

[Hal93] N. HALBWACHS. *Synchronous Programming of Reactive Systems*. Kluwer Academic, 1993.

[HVNP96] Bas W.'t. HART, Richard D. J. VAN NEE, et Ramjee PRASAD. « Performance Degradation Due to Code Tracking Errors in Spread-Spectrum Code-Division Multiple-Access Systems ». *IEEE Journal on Selected Areas in Communications*, vol. 14, num. 8, pp. 1669–1679, Oct. 1996.

[IEE85] IEEE. « IEEE Standard for Binary Floating-Point Arithmetic ». Document de standardisation IEEE 754-1985, IEEE, 1985.

[Inn01] INNOVEDA, Visual Elite - System Level Design. Innoveda, Inc, Mar. 2001. *http://www.innoveda.com/products/datasheets/v_elite.pdf*

[JB95] P. JUNG et J. BLANZ. « Joint Detection with Coherent Receiver Antenna Diversity in CDMA Mobile Radio Systems ». *IEEE Transactions on Vehicular Technology*, vol. 44, num. 1, pp. 76–88, Fév. 95.

[Kas66] T. KASAMI. « Weight Distribution Formula for Some Class of Cyclic Codes ». Rapport Technique R-285, Coordinated Science Lab., Univ. IL, Urbana, Avr. 1966.

[KM95] Volker KÜHN et Michael MEYER. « Correlative Channel Estimation in DS-CDMA Systems ». *Proceedings of the Third Annual WIRELESS Symposium*, pp. 279–284. Penton Publishing, Hasbrouck Height, NJ 07604, 1995.

[KP99] Zoran KOSTIĆ et Gordana PAVLOVIĆ. « Resolving Subchip-Spaced Multipath Components in CDMA Communication Systems ». *IEEE Transactions on Vehicular Technology*, vol. 48, num. 6, pp. 1803–1808, Nov. 1999.

[KS98] Young-Hoon KIM et Sanyogita SHAMSUNDER. « Soft Decision Feedback Receiver for CDMA Communication in Multipath ». *Proceedings of the IEEE Vehicular Technology Conference, 1998*, pp. 1705–1709, 1998.

[LLM98] Zheng-She LIU, Jian LI, et Scott L. MILLER. « An Efficient Code-Timing Estimator for Receiver Diversity DS-CDMA Systems ». *IEEE Transactions on Communications*, vol. 46, num. 6, pp. 826–835, Juin 1998.

[Luo00] Tao LUO. « Subchip-Spaced Multipath Diversity in CDMA Communication Systems ». *Proceedings of GLOBECOM*, San Francisco, CA, USA, 2000.

[NS96] L. NAJMAN et M. SCHMITT. « Geodesic Saliency of Watershed Contours and Hierarchical Segmentation ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, num. 12, pp. 1163–1173, Déc. 1996.

[Nus88] Patrice NUS. *Traitement Numérique du Signal : applications du processeur spécialisé DSP56002*. Publitronic - Elektor, 1988.

[OKT98] Henrik OLSON, Daniel KEREK, et Hannu TENHUNEN. « Low-complexity Coherent Rake Combining with a Kalman Phase Tracker ». *Proceedings of the IEEE Fifth International Symposium on Spread Spectrum Techniques and Applications (ISSSTA)*, Sun City, South Africa, Sep. 1998.

[OP98a] Tero OJANPERÄ et Ramjee PRASAD. « An Overview of Air Interface Multiple Access for IMT-2000/UMTS ». *IEEE Communications Magazine*, pp. 82–95, Sep. 1998.

[OP98b] Tero OJANPERÄ et Ramjee PRASAD. *Wideband CDMA for Third Generation Mobile Communications*. Artech House Publishers, 1998.

[Pej01] Milica PEJANOVIC. « Evolving From 2^{nd} To 3^{rd} Generation Cellular Systems ». *Proceedings of the IEEE Vehicular Technology Conference, 2001 Spring Edition*, Rhodes, Grèce, Mai 2001.

[PG58] R. PRICE et P. GREEN. « A Communication Technique for Multipath Channels ». *Proceedings of the IRE*, vol. 46, pp. 555–570, Mar. 1958.

[Pig00] Yvan PIGEONNAT. « *Etude de complexité d'algorithmes de détection conjointe pour les radiocommunications mobiles cellulaires de troisième génération* ». Thèse de doctorat, Institut National Polytechnique de Grenoble, 2000.

[PJ86] Marie-Agnès PALLAS et Geneviève JOURDAIN. « Joint estimation of close delays and application to underwater acoustics ». *EUSIPCO 86*, La Haye, Pays-Bas, Sep. 1986.

[Pra98] Ramjee PRASAD. *Universal Wireless Personal Communications*. Artech House Publishers, 1998.

[PSM82] Raymond L. PICKHOLTZ, Donald L. SCHILLING, et Laurence B. MILSTEIN. « Theory of Spread-Spectrum Communications – A Tutorial ». *IEEE Transactions on Communications*, vol. COM-30, num. 5, pp. 855–884, Mai 1982.

[PSO96] Stefan PARKVALL, Erik STRÖM, et Björn OTTERSTEN. « The Impact of Timing Errors on the Performance of Linear DS-CDMA Receivers ». *IEEE Journal on Selected Areas in Communications*, vol. 14, num. 8, pp. 1660–1668, Oct. 1996.

[RCS88] Jean Gabriel REMY, Jean CUEUGNIET, et Cedric SIBEN. *Systèmes de Radiocommunications avec les Mobiles*. Eyrolles, 1988.

[Ric44] S. O. RICE. « Statistical Properties of the Sine Wave plus Random Noise ». *Bell Systems Technical Journal*, vol. 23, pp. 282–332, 1944.

[Ric45] S. O. RICE. « Statistical Properties of the Sine Wave plus Random Noise ». *Bell Systems Technical Journal*, vol. 24, pp. 46–156, 1945.

[Ric48] S. O. RICE. « Statistical Properties of the Sine Wave plus Random Noise ». *Bell Systems Technical Journal*, vol. 27, pp. 109–157, 1948.

[Ris99] Tapani RISTANIEMI. « Code Acquisition in CDMA Communication System With Fictitious Pilot Signals ». *Proceedings of the 6^{th} IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1273–1277, Paphos, Chypre, Sept. 1999.

[Ros01] Laurent ROS. « *Réception multi-capteur pour un terminal radio-mobile dans un système d'accès multiple à répartition par codes - Application au mode TDD de l'UMTS* ». Thèse de doctorat, Institut National Polytechnique de Grenoble, 2001.

[Rou01] Sébastien ROUX. « *Adéquation Algorithme-Architecture pour le traitement multimédia embarqé* ». Thèse de doctorat, Institut National Polytechnique de Grenoble, 2001.

[SBR99a] Essam SOUROUR, Gregory BOTTOMLEY, et Rajaram RAMÉSH. « Delay Tracking For Direct Sequence Spread SPectrum Systems In Multipath Fading Channels ». *Proceedings of the 49th IEEE Vehicular Technology Conference, 1999 Spring Edition*, pp. 422–426, 1999.

[SBR99b] Essam SOUROUR, Gregory E. BOTTOMLEY, et Rajaram RAMÉSH. « Direct Sequence Spread Spectrum Acquisition With Sample Accumulation In Fading Channels ». *Proceedings of the 50th IEEE Vehicular Technology Conference, 1999 Fall Edition*, pp. 2198–2202, 1999.

[Sch81] R. O. SCHMIDT. « *A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation* ». Thèse de doctorat, Stanford University, Stanford, Californie, 1981.

[Sch82] Robert A. Scholtz. « The Origins of Spread-Spectrum Communications ». *IEEE Transactions on Communications*, vol. COM-30, num. 5, pp. 822–854, Mai 1982.

[SM01] A. Sharma et U. Mitra. « Blind Rate Detection for Multirate UMTS DS-CDMA Signals ». *Proceedings of the IEEE International Conference on Communications*, Helsinki, Finland, Juin 2001.

[SPMO96a] Erik G. Ström, Stefan Parkvall, Scott L. Miller, et Björn E. Ottersten. « DS-CDMA Synchronization in Time-Varying Fading Channels ». *IEEE Journal on Selected Areas in Communications*, vol. 14, num. 8, pp. 1636–1642, Oct. 1996.

[SPMO96b] Erik G. Ström, Stefan Parkvall, Scott L. Miller, et Björn E. Ottersten. « Propagation Delay Estimation in Asynchronous Direct-Sequence Code-Division Multiple Access Systems ». *IEEE Transactions on Communications*, vol. 44, num. 1, pp. 84–93, Jan. 1996.

[SSB98] K. Shin, B. Song, et K. Bacrania. « A 200-MHz Complex Number Multiplier using Redundant Binary Arithmetic ». *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 904–909, Juin 1998.

[STM99] STMicroelectronics, ST100 DSP-MCU CORE - Architecture Overview Handbook. STMicroelectronics, 1999. *http://www.st.com/stonline/prodpres/dedicate/st100/document/pdf/st100bk.pdf*

[STM00] STMicroelectronics, ST120 DSP-MCU CORE Reference Guide. STMicroelectronics, 2000. *http://www.st.com/stonline/prodpres/dedicate/st100/document/pdf/st120rg.pdf*

[Swa01] Stuart Swan, An Introduction to System Level Modeling in SystemC 2.0. Cadence Design Systems, Inc., Mai 2001. *http://www.systemc.org/papers/SystemC_WP20.pdf*

[Syn] Synopsys, COSSAP Datasheet. Synopsys, Inc. *http://www.synopsys.com/products/dsp/cossap_ds.html*

[Syn00a] Synopsys, CoCentric System Studio. Synopsys, Inc, Mar. 2000. *http://www.synopsys.com/products/cocentric_studio/cocentric_studio_dsA4.pdf*

[Syn00b] Synopsys, CoCentric SystemC Compiler. Synopsys, Inc, Mai 2000. *http://www.synopsys.com/products/cocentric_systemC/cocentric_systemC_ds.pdf*

[The00a] The Mathworks, MATLAB 6. The Mathworks, Inc, Nov. 2000. *http://www.mathworks.com/mason/tag/proxy.html?dataid=168&fileid=1388*

[The00b] The Mathworks, Simulink 4. The Mathworks, Inc, Oct. 2000. *http://www.mathworks.com/mason/tag/proxy.html?dataid=171&fileid=964*

[TIA98] TIA. « 800 MHz Cellular System, TDMA Radio Interface, Dual-Mode Mobile Station - Base Station Compatibility Standard ». Document TIA/EIA-627, Telecommunications Industry Association, 1998.

[TIA99] Comité TIA/TR-45.5. « Mobile Station-Base Station Compatibility Standard for Wideband Spread Spectrum Cellular Systems ». Document ANSI/TIA/EIA-95-B-99, Telecommunications Industry Association, Fev. 1999.

[TIA00] Comité TIA/TR-45.3. « TDMA Cellular PCS ». Document ANSI/TIA/EIA-136, Rev. B, Telecommunications Industry Association, 2000.

[TL99] L. Tsaur et D. C. Lee. « Use of OVSF Code For Zero-overhead Symbol Rate Adaptation ». *Proceedings of the International Symposium on Multimedia Information Processing*, Taipei, Taiwan, Dec. 1999.

[TL01] L. Tsaur et D. C. Lee. « Symbol Rate Adaptation and Blind Rate Detection using FOSSIL (Forest for OVSF-Sequence-Set-Inducing Lineages) ». *Proceedings of the IEEE International Conference on Communications*, Helsinki, Finland, Juin 2001.

[Tur80] George L. Turin. « Introduction to Spread-Spectrum Antimultipath Techniques and Their Application to Urban Digital Radio ». *Proceedings of the IEEE*, vol. 68, num. 3, pp. 328–353, Mar. 1980.

[TWS01] S. Tantikovit, M. Z. Wang, et A. U. H. Sheikh. « On Combining Schemes for WB-CDMA Rake Reception in the Presence of Interpath Interference ». *Proceedings of the Third IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, Taoyuan, Taiwan, Mar. 2001.

[Vit95] Andrew J. Viterbi. *CDMA : Principles of Spread Spectrum Communication*. Addison-Wesley, 1995.

[VT71] H. L. Van Trees. *Detection, Estimation and Modulation Theory*. John Wiley and Sons, 1971.

[ZLMS97] Dunmin Zheng, Jian Li, Scott L. Miller, et Erik G. Ström. « An Efficient Code-Timing Estimator for DS-CDMA Signals ». *IEEE Transactions on Signal Processing*, vol. 45, num. 1, pp. 82–89, Jan. 1997.