UNIVERSITÉ DE MONTRÉAL

CONCEPTION AU NIVEAU SYSTÈME DE L'APPLICATION DE PROTOCOLE SANS FIL WIMAX

FATOUMATA LAMARANAH BAH
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2009

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONCEPTION AU NIVEAU SYSTÈME DE L'APPLICATION DE PROTOCOLE SANS FIL WIMAX

présenté par: BAH Fatoumata Lamaranah
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. PIERRE Samuel, Ph.D., président
M. BOIS Guy, Ph.D., membre et directeur de recherche
Mme. NICOLESCU Gabriela, Doct., membre et codirectrice de recherche
M. BOLAND Jean-François, Ph.D., membre

Je dédie ce mémoire à mes parents pour leur amour et leur soutien indéfectible. Je dédie également ce travail à la mémoire de mon oncle Aliou Badra Diallo décédé en 2007 d'un cancer et qui fut mon répétiteur en Mathématiques et en Physique durant mes études secondaires. Qu'il repose en paix.

REMERCIEMENTS

Après deux années de recherche, ma maîtrise s'achève fermant ainsi un chapitre de ma vie et m'ouvrant de nouvelles perspectives. Ces deux années ont été pleines de joie, de frustrations, d'espoir. Mais je n'en serais jamais venue au bout sans l'aide de plusieurs personnes que je souhaiterai remercier.
Tout d'abord, un grand merci à mes professeurs Guy Bois et Gabriela Nicolescu qui m'ont permis d'effectuer mes recherches dans un domaine très porteur que sont les Systèmes embarqués et temps réels, merci pour leur patience et compréhension. Merci aux membres du Jury qui ont bien voulu évaluer le travail réalisé.

J'aimerais remercier également toute l'équipe du Circus, l'équipe au sein de laquelle j'ai effectué mes recherches. Je remercie infiniment Marc-André Cantin qui m'a conseillée et guidée tout au long de mon travail. Un grand merci aussi à Laurent, Sebastien LeBeux, Luc, Maxime, Benoît pour l'aide précieuse qu'ils m'ont apportée. Un grand Merci à Bruno Girodias pour toute l'aide qu'il m'a apportée ainsi qu'à Alain. Merci à Nicolas pour son soutien, à mes collègues de bureau Sébastien, Karim ainsi qu'aux autres membres de Circus Lam, Sylvain, Michel pour leurs encouragements.

J'aimerais remercier aussi mes amis, en particulier Maïmouna qui m'a encouragée et soutenue dans les moments difficiles, Ibrahima mon ami de toujours, Kefil, Franklin et enfin Hassatou.

Je ne serai pas celle que je suis sans ma famille, alors je souhaiterais lui rendre hommage à travers ce mémoire. Un grand merci à mes parents, à mes frères Ismaël et Abder-Rahman, à ma cousine Hawa et au reste de ma famille.

RÉSUMÉ

Communiquer est le principe même de l'homme. Au fil des siècles les modes de communication ont beacoup évolué passant d'un simple échange vocal entre deux personnes à un échange via des appareils (téléphones, ordinateurs,..) d'où la naissance des systèmes de télécommunication. La technologie sous-jacente tient à l'implantation de réseaux de communication qui sont les noyaux de la communication. C'est ainsi que de nos jours on a des réseaux internet, de téléphonie, de télévision, etc. Avec l'évolution des civilisations et l'expansion démographique, on cherche désormais à minimiser la surface occupée par ces réseaux tout en maintenant la même qualité de service. On assiste ainsi à l'émergeance de nouveaux réseaux de communication sans aucune interconnexion filaire.

Le **Wimax** est l'une de ces technologies émergeantes. Grâce à ses techniques de modulation telles que l'OFDM, elle permet une meilleure qualité de service que les réseaux existants. Elle est ainsi utilisée pour de nombreuses applications telles que la communication par vidéo conférence (VoIP), un accès étendu à internet, aux hotspot wifi et aux réseaux cellulaires .
Le problème qui se pose est de savoir comment implanter de nouveaux réseaux de façon rapide et efficace tout en respectant les contraintes de coûts et de temps de conception liées au marché.

La réponse à ces problèmes se trouve au niveau des méthodes de conception des systèmes de traitement de signaux complexes et hétérogènes. Les applications de réseau sans fil sont généralement destinées à être implémentées dans des systèmes sur puce (SOC) tels les ASICs ou les FPGAs. Les méthodes classiques de conception d'application de traitement de signal consistent à réaliser d'abord un modèle algorithmique de simulation à partir duquel un modèle bas niveau de simulation est

généré. Ce modèle est par la suite implémenté puis validé. Cela rend la conception très fastidieuse, car le modèle bas niveau est en général difficile à modifier.

La solution privilégiée consiste à travailler à un niveau intermédiaire afin de tester toutes les performances du système avant de générer le modèle bas niveau. Cette solution basée sur l'ESL (electronic system level) consiste à ramener au niveau système (c'est à dire pas sur la puce) la conception afin de gérer à la fois les spécifications et contraintes logicielles et matérielles.

Ainsi un flot de conception efficace serait de partir d'un modèle algorithmique réalisé à l'aide d'outil visuels de simulation qui évaluerait les performances générales du modèles; par la suite ce modèle serait ramené à un niveau système qui se chargerait de l'estimation d'un partitionnement logiciel/matériel adéquat et du choix d'une architecture optimale à réaliser; enfin la conception se terminerait par l'évaluation d'un modèle bas niveau RTL obtenu à partir du modèle système et évaluant les performances spécifiques voulue par le concepteur telles que la taille occupée sur la puce ou la vitesse de transfert des données.

Étant donné ces différents niveaux de conception, un environnement de conception idéal serait une plateforme permettant l'accès à ces trois niveaux sans aucun effort particulier de la part du concepteur.

La plateforme *Space Codesign* réalisée l'équipe du groupe de recherche **Space** offre un environnement de conception au niveau système. À travers des outils de profilage et de simulation, elle permet d'obtenir un partitionnement optimal de l'architecture finale à implémenter sur puce. De plus, grâce à des outils intégrés de génération de code RTL, elle fournit une architecture RTL complète du modèle à implémenter sur un FPGA de *Xilinx* faisant ainsi le lien avec le niveau RTL. La plateforme suit le schéma classique de conception selon l'ESL. A partir des spécifications du système, on définit un modèle de base de l'application. Ensuite à travers la plateforme ESL et ses outils, on parcours différents niveaux d'abstraction de plus en plus précis et bas jusqu'à l'obtention de l'architecture finale sur puce. Les outils permettent

d'automatiser les étapes de création et de modification des modèles tout au long de la conception et permettent également de faire abstraction des détails d'implémentation de l'architecture. De plus grâce aux outils intégrés de raffinement, elle permet de bâtir de façon rapide et efficace une implémentation du système obtenu sur FPGA (système sur puce).

L'objectif de notre travail consiste d'une part , à concevoir à partir d'une plateforme ESL de partitionnement logiciel/matériel une architecture de réseau sans fil c'est à dire la couche physique OFDM de la norme IEEE 802.16 correspondant **Wimax** et de l'autre à l'interfacer avec l'environnement de développement algorithmique *Matlab-Simulink* afin d'obtenir ainsi un flot de conception entier.

Dans ce mémoire, nous présentons le flot de conception complet de la physique OFDM de la norme IEEE 802.16 que nous appelerons **Wimax** pour simplifier. Au cours de ce travail, trois modèles du **Wimax** ont été réalisés: le premier au niveau algorithmique a été fait sous *Matlab-Simulink*, ensuite le second a été réalisé à l'aide de la plateforme *Space Codesign* et pour finir le modèle RTL obtenu par synthèse du second a été analysé et validé après simulation et synthèse sur FPGA de *Xilinx*. De plus une interface entre la plateforme *Space Codesign* et l'environnement *Matlab-Simulink* a été réalisée afin d'intégrer le modèle système au modèle algorithmique pour une comparaison des deux modèles.

ABSTRACT

Now a days, Wireless systems are everywhere. They had a tremendous growth in recent years. From personnal area networks like bluethooth to local networks like Wifi, most of the communications systems contains wireless systems which have differents networks and users. A major problem of these systems is the interoperability and deployment of wireless networks. This is why a lot of research are made to find solutions and Wimax is one of them. Wimax (wireless interoperability for microwave access) is a metropolitan area network which use different transmission mode for pmp or mesh links to portable access due to OFDM modulation. With the improvement of Wireless technology and because of the tight costs and time-to-market constraints, the main challenge is to get fast and efficient design method. To meet these requirements, a method of conception, ESL is mainly used.
In this report, a design process of the physical layer of the IEEE 802.16 standard with the OFDM modulation using an ESL Eclipse-based platform of software/hardware platform is presented. This ESL environment, with his integrated tools, build embedded applications and architectures by refining the transaction-level communications to some pin and cycle-accurate protocols as well as the generation of synthesizable hardware from the system-level specifications. Those tools automate creation and modification of modules in an ESL design flow and make abstraction of all the implementation details to built an ESL architecture.

The main contribution of this work is the multilevel design on baseband of the PHY-OFDM layer of the IEEE 802.16 standard using an ESL (electronic system Level) environnement of software/hardware partitionning and the use of a multi-system cosimulation technique between the ESL model built with the SpaceStudio platform and a model built on Simulink for validation through cosimulation interfaces of heterogeous systems.

The conception flow of the application is declined on 3 part : the first part is the conception of a Simulink model which is used as a Testbench. The second part which is the core of our work represent the ESL based model. The third and last part is the FPGA implementation of the architecture created by the ESL platform.

This report is organized as follows: Chapter 1 discusses related work, Chapter 2 present the physical OFDM based layer of the IEEE 802.16 standard, Chapter 3 present the implementations of differents models, Chapter 4 discusses the results, the last Chapter present a conclusion of the report.
The results show how an ESL platform can provide an early mock-up of a design to refine it for an FPGA implementation and how it can use a cosimulation method for validation.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT viii

TABLE DES MATIÈRES x

LISTE DES FIGURES xi

LISTE DES ACRONYMES xii

LISTE DES TABLEAUX xiii

LISTE DES ANNEXES xiv

INTRODUCTION 1

CHAPITRE 1 REVUE DE LITTÉRATURE 9

1.1 Conception des DSP : Un exemple, le Wimax 10

1.1.1 Des ASICs aux FPGAs 10

1.1.2 Des niveaux d'abstraction de plus en plus hauts 11

1.2 L'ESL: un choix de conception plus approprié 13

1.2.1 Les sous-niveaux d'abstraction du TLM 16

1.2.2 Réutilisation et automatisation de la conception 18

1.2.3 Méthodologie de Conception dans Space Codesign 20

1.2.3.1 Notions de bases : ports, signaux 20

1.2.3.2 Librairies de Space Codesign 21

1.2.3.3 Exploration architecturale 22
1.2.3.4 Profilage 24
1.2.4 Vers un modèle RTL dans EDK 25
1.3 Validation haut niveau : La cosimulation SystemC/Simulink 27
1.4 Conclusion 31

CHAPITRE 2 ÉTUDE DE LA COUCHE OFDM-PHY 32
2.1 Conceptes de base 33
2.1.1 Modulation OFDM 33
2.1.2 Paramètres 36
2.2 Chaine de codage 37
2.2.1 Mélangeur/De-Mélangeur 37
2.2.2 Encodage FEC (Forward Error Correction) 40
2.2.2.1 Encodeur REED-SOLOMON 40
2.2.2.2 Encodeur Convolution-Perforation 42
2.2.3 Entrelacement/De-Entrelacement 43
2.2.4 Assignation 48
2.2.5 Décodeur FEC 49
2.2.5.1 Décodeur Viterbi 49
2.2.5.2 Décodeur REED-SOLOMON 53
2.3 Conclusion 61

CHAPITRE 3 IMPLÉMENTATION DE LA COUCHE PHYSIQUE OFDM DE LA NORME IEEE 802.16 62
3.1 Vers un modèle en bande de base 63
3.2 Modèle Simulink 65
3.2.1 Envoi et Réception des données 66
3.2.2 Codage et décodage des données 68
3.2.3 Modulation et transmission des données 69

3.2.4 Réglages dans Simulink 71
3.3 Modèle sous Space Codesign 71
3.3.1 Méthodologie de Conception 72
3.3.1.1 Implémentation des modules 72
3.3.1.2 Paramétrage des modules 73
3.3.1.3 Architectures du modèle 75
3.4 Modèle de Co-simulation Space/Simulink 77
3.4.1 Interface entre Space Codesign et Visual Studio 78
3.4.2 Interface entre Visual Studio et Matlab 79
3.4.3 Interface entre Matlab et Simulink 80
3.5 Modèle RTL 81
3.5.1 Du modèle systemC au modèle RTL 83
3.5.2 Génération des éléments de synthèse FPGA 84
3.6 Conclusion 85

CHAPITRE 4 RÉSULTATS ET DISCUSSION 86
4.1 Performances au niveau algorithmique 86
4.2 Processus de co-simulation 90
4.3 Exploration architecturale et synthèse 91
4.3.1 Résultats de simulation selon le type de modulation et le nombre de sous-canaux 92
4.3.2 Profilage du modèle système 93
4.3.3 Résultats de synthèse 97

CONCLUSION ET TRAVAUX FUTURS 100

ANNEXES 103

LISTE DES FIGURES

Figure 1	Les réseaux sans fil	2
Figure 1.1	Approches de Conception	14
Figure 1.2	Eléments de Communication	16
Figure 1.3	Orthogonalité Calculs/Communications	17
Figure 1.4	Communication dans SystemC	20
Figure 1.5	Communications Elix et Simtek dans Space	22
Figure 2.1	Symbole OFDM dans le domaine fréquentiel	34
Figure 2.2	Transmission à travers un canal de communication	38
Figure 2.3	PRBS pour le **Mélangeur**	39
Figure 2.4	Initialisation des rafales en mode ascendant	39
Figure 2.5	Schéma de codage du Reed-Solomon	42
Figure 2.6	Mot code envoyé	43
Figure 2.7	Convolution à rendement de 1/2 et de longueur 7	44
Figure 2.8	Entrelacement à 1 sous-canal et 16-QAM : Avant permutation	47
Figure 2.9	Entrelacement à 1 sous-canal et 16-QAM : Après permutation	47
Figure 2.10	Décodage d'un code convolutif perforé	51
Figure 2.11	Schéma de décodage de codes de Reed-Solomon	55
Figure 2.12	Calcul du Syndrome	57
Figure 2.13	Architecture parallèle du Chien Search	60
Figure 2.14	Architecture de Forney + Correction des erreurs	61
Figure 3.1	Flot de conception du modèle de couche-PHY-OFDM	64
Figure 3.2	Modèle Simulink	66
Figure 3.3	Bloc des parametres	67
Figure 3.4	Bloc des modulations	69
Figure 3.5	Bloc FEC+Assignation/De-FEC+DeAssignation	70
Figure 3.6	Format des données de paramètres	74

Figure 3.7	Architecture synthétisée par ForteCynthesizer	76
Figure 3.8	Interface de co-simulation SpaceStudio-Simulink	78
Figure 3.9	Modèle de co-simulation SpaceCodesign-Simulink	82
Figure 4.1	BER pour un canal AWGN avec une modulation adaptive pour 16, 8, 4, 2 et canaux	87
Figure 4.2	BER pour canal Rayleigh attenuant et un canal Rayleigh dispersif avec une modulation adaptive pour 16, 8, 4, 2 et canaux	88
Figure 4.3	Visualisation des signaux reçus pour une modulation adaptative selon BPSK, QPSK, 16-QAM, 64-QAM pour un canal bruité, dispersif et attenuant	89
Figure 4.4	Vue d'une co-simulation entre Space Studio-Visual Studio-Matlab-Simulink	91
Figure 4.5	Utilisation du processeur	94
Figure 4.6	Utilisation du bus	95
Figure I.1	Catégories de réseaux sans fil	104
Figure I.2	Correspondance OSI-IEEE	112
Figure I.3	Communication Wimax	113
Figure I.4	Structure d'un PDU MAC	113
Figure I.5	Structure de trame TDD pour architecture PMP	116
Figure I.6	Structure de trame TDD pour architecture Mesh	117
Figure I.7	Initialisation de la trame descendante	122
Figure I.8	Initialisation des rafales en mode descendant	123
Figure III.1	Vue de la plateforme SpaceStudio	140
Figure III.2	Configuration de l'architecture	143
Figure III.3	Vue de SpaceMonitor	144

LISTE DES ACRONYMES

BS	**B**ase **S**tation
BWA	**B**roadband **W**ireless **A**ccess
CS	**C**onvergence **S**ublayer
CPS	**C**ommon **P**art **S**ublayer
FDD	**F**requency **D**ivision **D**uplexing
FDM	**F**requency **D**ivision **M**ultiplexing
OFDM	**O**rthogonal **F**requency **D**ivision **M**ultiplexing
OFDMA	**O**rthogonal **F**requency **D**ivision **M**ultiple **A**ccess
PDU	**P**rotocol **D**ata **U**nit
PHY	**C**ouche **P**hysique
Qos	**Q**uality **O**f **S**ervice
RTL	**R**egister **T**ransfert **L**evel
SAP	**S**ervice **A**ccess **P**oint
SDU	**S**ervice **D**ata **U**nit
SS	**S**ubscriber **S**tation
TDD	**T**ime **D**ivision **D**uplexing
TDM	**T**ime **D**ivision **M**ultiplexing
WIMAX	**W**orldwide **I**nteroperability for **M**icrowave **A**ccess
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WMAN	**W**ireless **M**etropolitan **A**rea **N**etwork
WPAN	**W**ireless **P**ersonal **A**rea **N**etwork
WWAN	**W**ireless **W**ide **A**rea **N**etwork

LISTE DES TABLEAUX

Tableau 2.1	Vecteurs de perforation	45
Tableau 2.2	Taille des blocs d'entrelacement	45
Tableau 4.1	Temps d'exécution en ms selon le nombre de sous-canaux et le type de modulation	93
Tableau 4.2	Cycles de Simulation par module et pour un Symbole OFDM	96
Tableau 4.3	Résultats de temps d'exécution par configuration	97
Tableau 4.4	Nombre de Slices par module	98
Tableau I.1	Comparaison Wifi-Wimax	110
Tableau I.2	Tableau des paramètres des symboles OFDM	121
Tableau I.3	Seuils d'erreurs de constellation	122
Tableau I.4	Nombre d'octets requis par modulation	123
Tableau II.1	Eléments de GF(2^8)	129
Tableau II.2	Tableau des constellations BPSK,QPSK, 16QAM,64QAM	136
Tableau II.3	Durée de Trames admises	138
Tableau IV.1	Fonctions de communication SystemC/Matlab/Simulink	149

LISTE DES ANNEXES

ANNEXE I RÉSEAUX SANS FIL 103
I.1 Des réseaux sans fil au Wimax 103
I.2 Définitions 105
I.3 Wimax versus Wifi 108
I.4 Principe de fonctionnement du Wimax 109
I.5 Processus de communication 111
I.6 Processus d'identification d'une BS et d'une SS 114
I.7 Structure des données échangées 115
I.8 Paramètres OFDM 119
I.9 Seuils d'erreurs SNR 122
I.10 Format des vecteurs d'initalisation pour une communication descendante 122
I.11 Nombre d'octets obligatoire par modulation 123

ANNEXE II CODES CORRECTEURS D'ERREURS 124
II.1 Codes systématiques 124
II.2 Théorie de Galois 125
II.3 Polynône primitif 127
II.4 Multiplication dans GF(2^8) 127
II.5 constellations 136
II.6 Durée des Trames 138

ANNEXE III CONCEPTION DANS SPACE CODESIGN 139
III.1 Présentation de Space Codesign 139
III.2 SpaceMonitoring 144
III.3 Section GenX 145
III.4 Bancs d'Essai 145

ANNEXE IV UTILISATION DE SIMULINK 148
IV.1 Simulation 148
IV.2 S-Functions 149
IV.3 Interfaces de cosimulation 151

INTRODUCTION

Présentation du mémoire

La communication sans fil remonte à l'origine de l'humanité. Depuis toujours l'Homme cherche à transmettre des informations. Pour cela il a développé des codes, langages, alphabets; il a utilisé tous les moyens à sa portée pour véhiculer ses messages: tam-tam, fumée, signaux sonores, etc....

En 1896[1] Giuliano Marconi établit la première liaison de téléphonie sans fil quelques années après l'invention du téléphone par Graham Bell. Les travaux de G.Marconi ont mené à la première conversation radio entre deux bateaux en 1915 [**?**] établissant ainsi le premier réseau sans fil. Le terme réseau définissant un ensemble d'éléments (ordinateur, pda, téléphones, etc...) connectés entre eux par des liens physiques (fibre optique, paires torsadées, air) et échangeant des données numériques. Dans la communication sans fil, ces données sont soumises à un ensemble de traitements appelés traitement de signal afin de les transmettre sans encombres à travers le support utilisé qu'est l'air.

De nos jours, l'industrie des télécommunications a pris un grand essor grâce aux réseaux sans fil; les systèmes de communication sans fil tels que les cellulaires, les téléphones sans fil, satellites, les ordinateurs portables sont de plus en plus indispensables à notre société (voir figure 1). Les réseaux mobiles permettent de passer d'un endroit à un autre sans interruption de la communication. Ils permettent également de réduire les coûts d'établissement de réseaux. Cela est fort utile pour les pays en voie de développement où il n'y a presque pas de réseaux filaires. Du fait de sa croissance exceptionnelle, les acteurs de l'industrie veulent donc développer rapidement et à faible coût des systèmes sans fil dont la flexibilité permet de répondre

[1] voir http://www.httr.ups-tlse.fr/pedagogie/cours/intro/histo.htm

à l'importante demande en termes de largeur de bande et de débit. De tels systèmes tel que le **Wimax**[**?**] qui fait l'étude de ce projet, sont très utiles pour le déploiement de réseau dans des zones à accès difficiles, interdits ou temporaires. Néanmoins, cette extension des réseaux sans fil entraîne l'accroissement des problèmes d'interférences dus à la nature électromagnétique des signaux transmis.

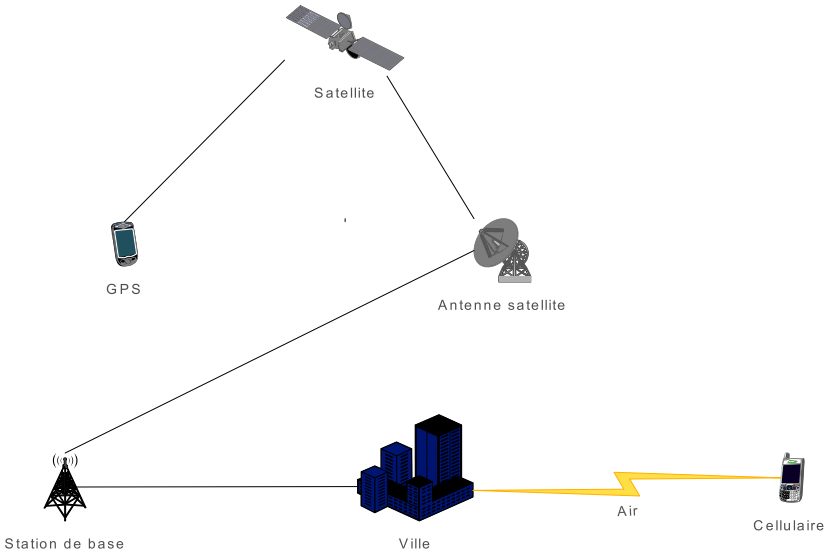


Figure 1 Les réseaux sans fil

En effet, en 1905 Albert Einstein[2] a développé la théorie des photons qui explique que les ondes électromagnétiques contiennent de petites particules appelées photons; lesquelles déterminent l'amplitude de l'onde par leur nombre et la fréquence de l'onde par leur énergie. Toutefois cette transmission électromagnétique entraîne des problèmes d'interférence qui constituent le grand désavantage de ce type de communication par rapport aux réseaux filaires. Cela entraîne un taux d'erreur binaire élevé qu'on essaie de minimiser à l'aide de mécanismes de protection des

[2]source Wikipédia

données. Pour diminuer les risques d'interférence, on assigne des bandes de fréquences aux utilisateurs dans lesquelles ils peuvent émettre et recevoir des informations. et on utilise des techniques de multiplexage des données telles que l'OFDM afin de limiter ces interférences et de minimiser le délai de propagation du signal induit (selon qu'il rencontre un obstacle ou non). Le **Wimax** tel que spécifié dans sa norme propose différents modèles de couches physiques utilisant différents types de modulation pour le transfert des informations. Ainsi, notre choix de conception s'est porté sur **la couche physique-OFDM de la norme IEEE 802.16** [3] qui utilise la modulation OFDM.

Problématique La nécessité d'implanter ce type de système complexe et hétérogène pose le problème de la méthode de conception à utiliser afin de remplir les exigences en terme de coûts production et de temps de conception. Ainsi,de nouvelles techniques de conception de SOCs ont été développées pour atteindre ces conditions. Ces techniques consistent à travailler à des niveaux d'abstraction de plus en plus haut et de plus en plus précis pour respecter les spécifications du système.

Toutes les applications reposant sur un support matériel étaient conçues au départ à l'aide de processeurs d'usage général. Ces processeurs s'avérant trop lents pour des applications nécessitant des débits de transfert très élevé, la solution a été de leur associer des circuits intégrés, les ASICs (Application Specific Integrated Circuits) permettant ainsi d'accélérer ces systèmes.

Cependant, les ASICs offrant peu de flexibilité en terme de reprogrammabilité et limitant les possibilités de réutilisation des modèles précédents, de nouveaux circuits intégrés appelés systèmes sur puce (SOCs) reconfigurables tels que les FPGA (Field Programmable Gate Array) sont apparus. Ces systèmes combinent les avantages d'un processeur à usage général (offrant une grande flexibilité de développement)

[3] Rq:Pour des raisons de simplicité nous utiliserons le terme **Wimax** pour désigner la norme IEEE 802.16

et ceux des ASICs à travers un partitionnement logiciel/matériel. Ils sont également conçus pour optimiser les implémentations des systèmes et fournir toutes les ressources permettant de remplir les fonctionalités de ces systèmes. Ils permettent au concepteur de paralléliser les algorithmes en allouant autant de ressources que possible et ainsi de réaliser les performances requises du système.
Au cours des dernières années, le domaine des circuits intégrés a donc subi une croissance phénoménale; désormais il est possible d'intégrer sur une même puce jusqu'à un million de transistors par millimetres carrés. La loi de Moore [**?**] établie en 1965 nous prédit en effet que leur nombre doublera tous les 18 mois ce qui offre des possibilités incroyables d'intégration de systèmes embarqués sur puce (par exemple, la possibilité d'avoir sur une même puce plusieurs réseaux wireless).

Une première avancée a été le passage du niveau des portes logiques à celui du transfert de registre (RTL). La simulation à ce niveau est plus rapide tout en restant efficace car permettant de représenter les comportements des signaux et registres et fournit une modélisation du circuit à partir de laquelle les câblages réels peuvent être dérivés. De plus, ce niveau offre des possibilités de synthèse automatique et de modularité de l'application permettant de réduire les erreurs inévitables de conception.
Toutefois la conception à ce niveau nécessitant beaucoup trop d'efforts pour la conversion des algorithmes DSP en un langage de description du matériel (VHDL,Vérilog) en rendant l'application dépendante de la plateforme RTL; l'une des solutions privilégiée des concepteurs est de travailler au niveau algorithmique puis de raffiner jusqu'au niveau RTL. Le niveau algorithmique n'offrant aucune information sur le comportement réel de l'application du fait de son niveau d'abstraction, il s'agit de le ramener au niveau système qui par des raffinages successifs aboutit à l'implémentation RTL.

L'ESL (Electronic System Level) [?], [?] à travers l'un de ses dérivés le TLM (Transaction Level Modeling) [?], [?] est apparue pour pallier ces inconvénients en permettant de travailler à des niveaux d'abstraction (UTF/TF et BCA) plus élevés que le RTL mais supportant toutes les spécifications logicielles et matérielles des composants de ce niveau. Les auteurs[?] ont développé une approche ESL de conception logicielle/matérielle basée sur une plateforme virtuelle nommée Space Codesign. Cette plateforme comprend un ensemble de processeurs, liens de communications, memoires et modules IP modelisés par des appels de fonctions, des accés mémoire et transfert de données ainsi qu'une représentation des instructions du processeur. Cette méthodologie basée sur quatre étapes principales : spécification du système, exploration, analyse et intégration combinée aux atouts des plateformes virtuelles tels que le développement au niveau système, l'analyse et l'optimisation des résultats de simulation ainsi que leur vérification et validation offre d'énormes avantages en matière de conception d'applications complexes.

Un autre moyen de respecter les exigences de développement se trouve au niveau de la vérification. En effet, cette étape cruciale est très coûteuse en temps et en moyen. Dans l'industrie, on estime que cette étape correspond à environ 70% du temps total de conception et donc nécessite des méthodes très rapides et efficaces. Plusieurs techniques de vérification existent selon les domaines d'application. Selon l'approche ESL, la vérification du système suit une étape Top-Down (de haut en bas) c'est-à-dire que l'on effectue une validation entre différents niveaux d'abstraction à travers la création d'interfaces de cosimulation.

Que se soit donc pour la conception ou la vérification du système, il s'avère donc que l'accélération du temps de développement nécessite de travailler à différents niveaux d'abstraction, d'être le plus haut possible et indépendant de la plateforme d'implémentation et enfin d'automatiser le processus de passage d'un niveau à un autre. Ainsi, pour une conception plus efficace, il faudrait pouvoir valider un modèle

conçu à un certain niveau avec le niveau adjacent. La plateforme *Space Codesign* étant associée à la plateforme *EDK* de *Xilinx*, cela permet une validation entre les modèles système et RTL. Un moyen de valider le modèle système au modèle algorithmique serait d'associer à la plateforme *Space Codesign* l'environnement *Matlab-Simulink* en lui associant des interfaces de cosimulation.

Objectif : Notre travail s'est porté sur la conception de la couche physique OFDM de la norme IEEE 802.16 par l'utilisation d'une plateforme ESL, nommée Space Codesign, afin de respecter les requis du cycle de développement, puis l'amélioration de la méthode de conception à travers l'utilisation de la cosimulation pour la vérification entre le niveau algorithmique conçu à l'aide de l'environnement *Matlab-Simulink* et le niveau système réalisé au niveau de la plateforme *Space Codesign*. Le modèle réalisé au niveau de *Space Codesign* a été raffiné en modèle RTL qui représente le modèle final à implémenter sur FPGA. *Space Codesign* intègre différents outils qui facilitent la création, la modification des modèles et permet de génèrer toutes les composantes matérielles de l'architecture à implémenter sur FPGA. D'autre part, une librairie de traitement de signal étant disponible sur *Matlab-Simulink*, il s'agit de réutiliser les blocs existants afin de bâtir le modèle algorithmique utilisé à des fins de validation. L'environnement *Simulink* est bâti sur la plateforme algorithmique *Matlab*.

Contribution: Les apports de ce travail peuvent se décliner en 2 parties.

La première est la réalisation des trois modèles correspondant aux trois niveaux d'abstraction de la conception : le niveau algorithmique, le niveau système et le niveau RTL. Ces modèles ont été conçus de façon à respecter tous les requis obligatoires de l'application.

- Un premier modèle a été entièrement conçu sous *Simulink* afin de valider les spécifications de l'application telles que définies dans la norme IEEE 802.16. Cette architecture fonctionnelle a été réalisée uniquement à l'aide de blocs **Simulink** et ne tient pas compte des contraintes temporelles liées à l'implémentation matérielle du modèle.

- Le second modèle, correspondant au coeur de ce projet a été construit à partir de la plateforme ESL Space Codesign qui offre différents niveaux de conception (Elix, Simtek) correspondant aux niveaux d'abstraction du TLM. Les éléments de l'architecture ont été entièrement réalisés en *SystemC* [**?**] sous forme de modules selon l'approche ESL. Ces modules ont été réalisés de façon à optimiser le volume des calculs et des communications. La plateforme Space Codesign comprend différentes composantes électroniques (microprocesseurs, contrôleurs, mémoires,bus, timers) sur une même puce et nécessaires à l'implémentation sur FPGA. Durant cette étape, un partitionnement logiciel/matériel optimal a été défini grâce aux outils de la plateforme à partir des informations collectées sur le temps de calcul et de communication des composants du système.

- Finalement le dernier modèle situé au niveau RTL et obtenu à partir du partitionnement a été implémenté sur le FPGA. Les éléments matériels de ce modèle ont été obtenus par synthèse comportementale [**?**] ou à partir des librairies de la plateforme *Xilinx* d'implémentation du FPGA. Les éléments de l'application non fournis par cette plateforme ont été codés manuellement.

La deuxième contribution représente l'intégration de la co-simulation au modèle ESL à des fins de vérification et de validation. La technique de cosimulation utilisée existant au préalable, elle a été adaptée à la plateforme Space Codesign. Cette technique utilise la librairie fournie par l'outil *Matlab-Simulink* permettant une cosimulation entre deux systèmes hétérogènes et appliquée au domaine du traitement du signal.

Plan du mémoire

Ce mémoire comporte 4 chapitres (sans compter les chapitres d'introduction et de Conclusion). Le premier chapitre présente la revue de littérature du mémoire. Il aborde les travaux relatifs au Wimax , à la cosimulation *SystemC-Matlab-Simulink* et à la technologie de conception ESL. Le chapitre 2 présente les spécifications de la couche physique OFDM du Wimax telles que définies dans la norme 802.16 de IEEE ainsi que les éléments non spécifiés nécessaires à la construction du modèle complet. Le chaptire 3 nous donne l'implémentation de la couche physique à l'aide des outils présentés au chapitre 1. Le chapitre 4 présente les résultats de simulation et de synthèse des modèles implémentés au chapitre 3. Finalement le dernier chapitre conclut ce projet et nous présente les possibilités de recherche futures du modèle.

CHAPITRE 1

REVUE DE LITTÉRATURE

Le terme **Télécommunications** désigne l'ensemble des moyens techniques (ondes électromagnétiques) permettant l'acheminement (transmisssion et commutation) fidèle et fiable d'informations entre deux points quelconques pour un coût raisonnable[1]; ce coût (appelé qualité de service) est mesuré en terme de débit ou largeur de bande, taux d'erreur, durée d'établissement de la communication. L'information peut prendre diverses formes : son (paroles, musique), textes (retraitables ou non), données, images (fixes ou animées). L'ensemble des moyens physiques utilisables par des usagers qui bénéficient d'un même service s'appelle un réseau. Au départ, les informations transmises étaient principalement des signaux sonores pour la téléphonie et la télévision, qui étaient codés puis acheminés à travers des procédés de transmission analogique.
Toutefois les progrès en informatique (internet) ont vu la nécessité de relier ces signaux sonores à des équipements numériques donnant ainsi accès à plus d'applications (textes, images, données, etc..). La coexistance des deux techniques comportant des coûts, on a donc cherché à définir des réseaux purement numériques. Ainsi, une découverte majeure des télécommunications a été celle des réseaux sans fil dont le support de transmission est l'air contrairement aux réseaux filaires utilisant des câbles ou fibres optiques.

Cette évolution des techniques et du support de transmission a complètement bouleversé le domaine des télécommunications en ouvrant de nombreux horizons par l'association des techniques informatiques à celles des télécommunications.

[1] voir http://www.volle.com/ENSPTT/introtcom.htm

1.1 Conception des DSP : Un exemple, le Wimax

L'émergeance de nouveaux réseaux sans fil tels que le **Wimax** pose la nécessité d'améliorer les méthodes de conception afin de produire dans des délais très courts des systèmes opérationnels et surtout extensibles. Le grand engouement provoqué par le Wimax au niveau des industries de télécommunication (Sprint, Nortel) ont mené à de nombreux travaux [?], [?] relatifs à son déploiement et entraîné de nombreuses réalisations effectuées par les sociétés de conception de systèmes sur puces tels que les FPGA.

1.1.1 Des ASICs aux FPGAs

En effet, l'évolution des réseaux sans fil a mis en évidence les limites des méthodes classiques de conception utilisant des processeurs d'usage général et les ASICs (niveau portes logiques). Les applications de réseaux sans fil nécessitant une énorme puissance de traitement difficile à réaliser à partir des processeurs classiques; un grand nombre de vendeurs d'outils de conception (*Altera, Xilinx, Coware*) sont donc arrivés à la conclusion qu'il est nécessaire d'utiliser un autre type de plateforme, en l'occurence les FPGAs (niveau RTL). Ces systèmes sur puce (SOCs) permettent un degré très élevé de parallélisme des calculs (multiplications, additions,....). De plus, l'apparition en 1980 des langages de description du matériel tels que le Vérilog et le VHDL permettent de décrire entièrement des SOCs au niveau RTL en spécifiant l'architecture matérielle des applications au niveau des registres sans avoir à détailler les composantes logiques. Les applications du domaine des télécommunications communément appelées DSP (applications de traitement du signal) sont généralement conçues à partir d'algorithmes d'encodage/décodage et de modulation des données. Afin de remplir leurs contraintes en terme de temps de mise sur le marché, les concepteurs préfèrent se concentrer sur la conception des algorithmes et minimiser le temps d'implémentation

du matériel.
Le développement au niveau RTL d'applications telles que la couche physique OFDM de la norme IEEE 802.16 [?], [?], [?] comporte toutefois certaines limitations.
En effet, les conceptions volumineuses et complexes de systèmes tels que les algorithmes des télécommunications rendent difficile la mise en oeuvre efficace des méthodologies classiques. Bien que le niveau d'abstraction du RTL soit suffisamment élevé pour une conception matérielle, il n'est cependant pas adapté à du logiciel et ne permet donc pas d'exploiter un possible partitionnement logiciel/matériel optimal pour des algorithmes complexes et volumineux. En effet, convertir des algorithmes DSP en un langage tel que le Vérilog ou VHDL peu s'avérer très fastidieux et suciter des erreurs sans compter que l'architecture FPGA reste dépendante de la plateforme FPGA. Cela peut donc rendre difficile la mise en oeuvre optimale et efficace de l'application dont le but principal est la portabilité et l'interopérabilité entre différents opérateurs des télécommunications car de simples modifications apportées à l'algortihme au niveau système peuvent avoir d'immenses répercussions sur la mise en oeuvre FPGA.

1.1.2 Des niveaux d'abstraction de plus en plus hauts

L'utilisation de bibliothèques DSP indépendantes de la technologie contribue de manière significative à la productivité du concepteur. Une conception constituée de fonctions DSP génériques ne comporte pas les paramètres architecturaux qui risqueraient d'encombrer inutilement les premières phases de la conception.

On retrouve de nombreux exemples de conception au niveau algorithmique de la couche physique étudiée, réalisés principalement à partir d'outils de simulation tels

que *Matlab-Simulink*[2] ou labview [?]. L'intérêt principal de ce type de conception vient du fait que ces outils possèdent déjà des librairies entières dédiées aux systèmes de traitement de signal ou de réseaux de télécommunication. L'architecture de la couche physique étant constituée d'une partie codage/décodage et d'une partie modulation/démodulation de données, il s'agit d'un système hétérogène comportant un aspect numérique et un autre analogique. Les différents blocs de la librairie correspondent aux différentes étapes du processus de transmission des données. Les systèmes sont généralement représentés en bande de base étant donné la nature des signaux numériques et la partie analogique peut être représentée par des blocs prédéfinis de la librairie. De plus les nombreux éléments de visualisation qu'offrent ces outils [?], [?], [?] permettent une bonne estimation des performances (taux d'erreur binaire, rapport signal sur bruit selon les modulations supportées) des sytèmes. Les modèles de la couche physique simulés à partir d' outils comme *Matlab-Simulink* utilisent un découpage du système en blocs qui communiquent entre eux à travers des signaux. Ce niveau constitue une représentation graphique de la spécification du système et permet de valider les paramètres du système à réaliser. Toutefois, bien que très avantageux du point de vue souplesse de conception, ce niveau ne permet pas de représenter la réalité physique du système c'est à dire les délais de transmission entre les différents signaux, le temps d'exécution des opérations des différents blocs ainsi que la vitesse d'exécution du système qui est destiné à être porté sur une puce et donc représenté physiquement.

Il existe des systèmes permettant de combler l'écart entre le niveau algorithmique et le niveau RTL et certains concepteurs privilégient cette approche plus simpliste. Dans certains cas, la plateforme matérielle [?],[?] intègre des outils de synthèse matérielle de blocs algorithmiques de librairies comme celles de *Matlab-Simulink*. Ces méthodes nécessitent toutefois le re-codage des différents blocs algorithmiques en un langage de description tel que *VHDL/Vérilog* ou *SystemC* car ces plateformes ne fournissent

[2]voir Annexe IV

que les interfaces de communication entre les simulateurs algorithmiques/matériels. Ainsi le seul avantage de ces méthodes par rapport au niveau RTL est la possibilité d'effectuer les ajustements au niveau algorithmique avant la synthèse RTL car en réalité le système réalisé est dépendant de la plateforme RTL de conception et l'automatisation de la synthèse n'est pas prise en compte.
Afin de pallier à ces inconvénients, une nouvelle approche de conception, l'ESL est apparue. Elle consiste à travailler au niveau système et représente un intermédiaire entre le niveau algorithmique et le niveau RTL. En effet, à travers l'un de ses dérivés, le TLM, elle permet de s'assurer que le passage d'un niveau supérieur à un niveau inférieur ne requiert que peu d'efforts et que le système final peut être validé de manière rapide et efficace.

1.2 L'ESL: un choix de conception plus approprié

Il s'agit d'une méthodologie de conception à différents niveaux d'abstraction et permettant la réutilisation à travers le concept de plateforme. Dans l'approche classique de conception d'une application telle que le Wimax (voir partie 1 figure 1.1), on effectue dès le départ un partitionnement logiciel/matériel à partir des spécifications. Le développement logiciel étant totalement indépendant du développement matériel, c'est seulement après intégration des deux parties que l'on peut se rendre compte des modifications à apporter au système initial. Cela rend donc le développement très fastidieux et coûteux en terme de temps et d'argent. A travers l'ESL (voir partie 2 figure 1.1), bien qu'un partitionnement logiciel/matériel soit effectué le développement des deux parties se fait en collaboration grâce à l'insertion d'un modèle transactionnel entre l'étape des spécifications et celle du partitionnement.

De plus, l'ESL offre de nombreuses possibilités :

- création de la plateforme de simulation contenant les processeurs et co-

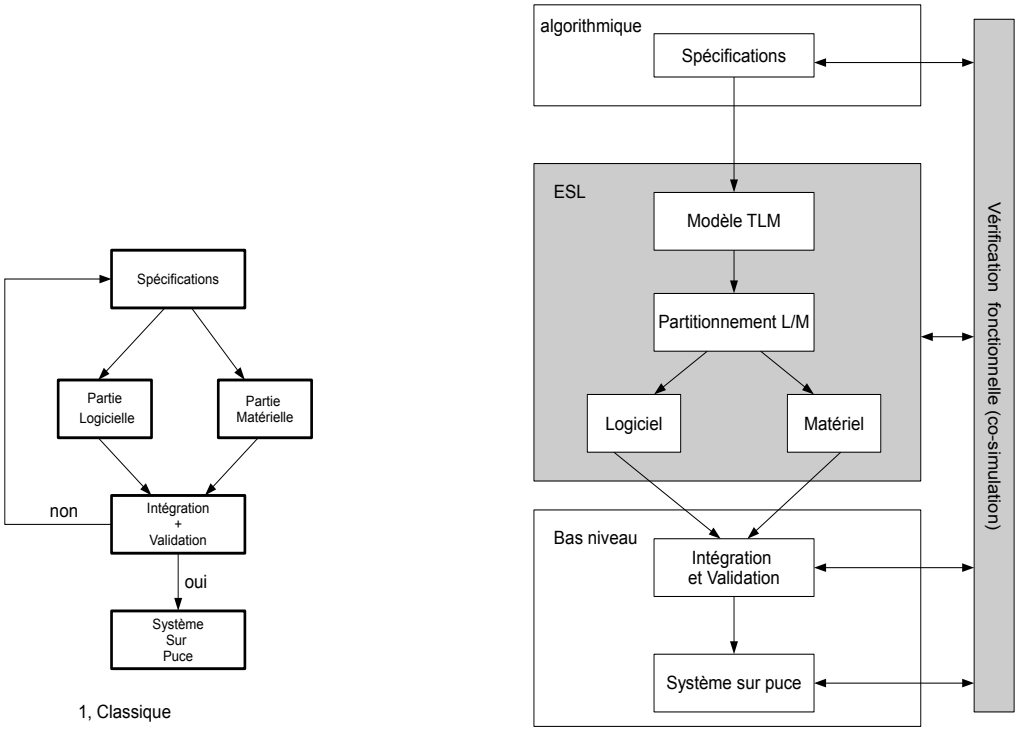


Figure 1.1 Approches de Conception

processeurs, canaux de communication, mémoires et périphériques à différents niveau d'abstraction.

- création de modèles fonctionnels de vérification et exploration architecturale selon l'architecture choisie

- raffinement à bas niveau au cycle d'horloge près (par ex. RTL) pour des implémentations matérielles.

L'ESL [?] permet aussi d'obtenir une analyse des performances du système à travers l'intégration de différents simulateurs et outils graphiques et l'utilisation de langages de programmation tels que C/C++, *SystemC* (basé sur le C/C++), SytemVerilog. L'un de ses dérivés très utilisé, le TLM est principalement axé sur l'abstraction des communications et donc des transactions afin d'accélérer les simulations et de fournir une analyse la plus précise du système. Cette approche de modélisation consiste à découper le système à concevoir en modules qui communiquent entre eux

sous forme de transactions; on peut y voir une certaine analogie avec un réseau de télécommunication qui se définit par un ensemble d'éléments qui communiquent entre eux via des canaux de communication. Dans notre cas les éléments sont des modules et les canaux des bus.

Ainsi, l'avantage de l'ESL réside dans la conception de modèles transactionnels lesquels sont par la suite partitionnés et implémentés grâce à de nombreux mécanismes d'abstraction et de réutilisation. Cela permet d'accélérer le processus de conception, de simulation et de vérification et donc de développement du système.

Les modèles transactionnels utilisent un découpage en blocs du système à partir des spécifications. Ces blocs appelés modules communiquent entre eux par des canaux via des processus transactionnels. Dans chaque communication inter-modules, il y a un initiateur et une cible. L'initiateur appelé Maître de la communication effectue une requête qui est traitée par la cible appelée Esclave. Le canal de communication se charge d'arbitrer et donc de gérer le transfert de l'un à l'autre à travers des appels de fonctions. La figure 1.2 présente les éléments de communication à différents niveaux d'abstraction.

La modélisation transactionnelle (TLM) permet d'améliorer le processus de conception en permettant une validation du système à chaque étape sans avoir à attendre l'étape d'intégration comme c'est le cas dans l'approche classique. La validation est effectuée soit par simulation des différents niveaux ou par comparaison entre un niveau transactionnel et un autre par le biais des interfaces de communication, cela permet ainsi une exploration architecturale des possibilités d'implémentation du système. Le découpage en module permet un parallélisme d'applications de type flot de données telles que celle qui fait l'objet de notre étude. De plus, la communication des modules s'effectuant à l'aide de canaux, cela permet une communication des modules à travers des appels de fonction contrairement aux

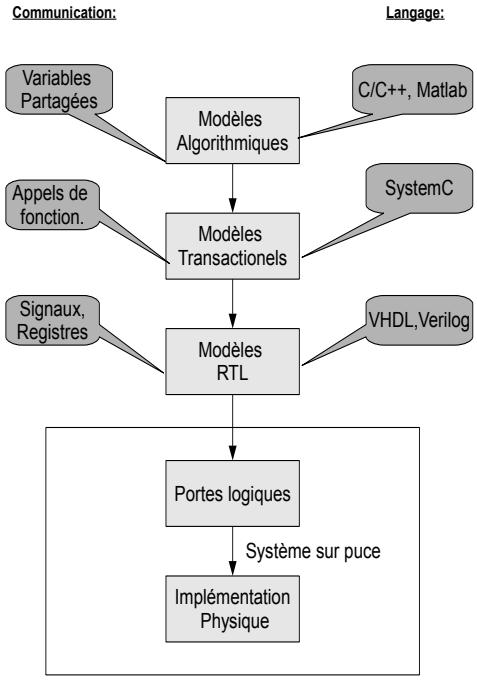


Figure 1.2 Eléments de Communication

modèles RTL qui communiquent par signaux. C'est cette abstraction qui représente le principal intérêt du TLM car on peut ainsi passer d'un haut niveau (presque au niveau des spécifications) à un bas niveau (presque au niveau RTL) sans avoir à détailler la communication.

1.2.1 Les sous-niveaux d'abstraction du TLM

Les modèles transactionnels [?] peuvent être classifiés en fonction des calculs et des communications. En effet, le niveau le plus précis correspond à celui où les calculs et les communications sont évalués au cycle près tandis que le niveau le moins précis correspond à celui où on décrit les fonctionnalités du système sans tenir compte de leur temps d'évaluation. Lorsqu'on observe la figure 1.3, on constate une orthogonalité entre la précision des calculs et celle des communications.

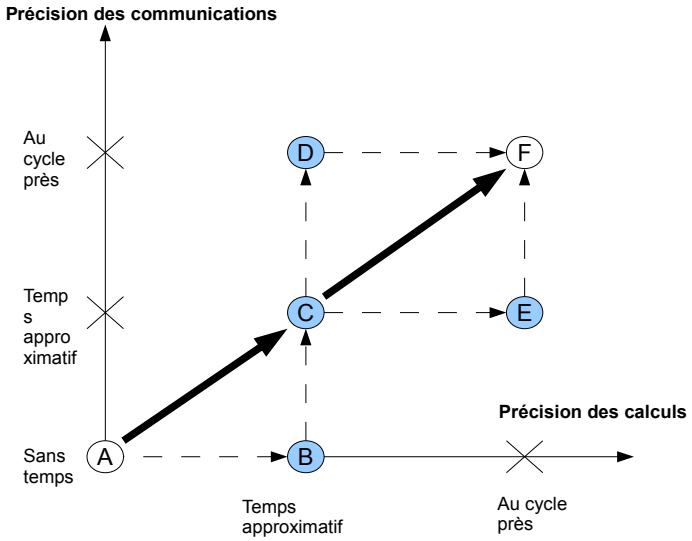


Figure 1.3 Orthogonalité Calculs/Communications

Ces niveaux peuvent être regroupés en trois grandes catégories : A correspondant aux spécifications, B, C et D représentant les sous-nivaux transactionnels et F et E le niveau FPGA.

- La première catégorie correspond au niveau A et ne tient compte d'aucune notion de temps au niveau des calcluls et des communications. Ce niveau sert uniquement à décrire les fonctionalités du système.

- La seconde catégorie correspond aux niveaux B, C et D. Le niveau B est presque identique au niveau A excepté qu'on effectue une appoximation du temps des calculs (pour une conception classique d'une application comme le Wimax, il pourrait s'agir d'un modèle *Simulink*). Cela nécessite donc d'évaluer le temps de simulation de chaque module du système. Les niveaux C et D font intervenir les canaux de communications représentés par des bus et nécessitant donc une arbitration des communications. Les temps de calcul sont approximatifs à ces niveaux; quant aux temps de communication, ils sont approximatifs

pour le niveau C et au cycle près pour le niveau D permettant ainsi de se rapprocher d'un modèle RTL synthétisable sur FPGA. Cette catégorie offre un bon compromis entre la précision des simulations et la rapidité de simulation et de développement permettant ainsi d'explorer différentes architectures selon le type de communication (bus, liens directs, etc..) et le choix de partitionnement logiciel/matériel.

- La dernière catégorie, représente les niveaux E et F quasimment identiques au modèle RTL excepté que le niveau E est moins raffiné au niveau des communications. Les transactions sont estimées au cycle près que ce soit au niveau des calculs ou au niveau des communications. Cela permet ainsi de faire une simulation au niveau RTL tout en faisant abstraction des signaux et des ports. La simulation est ainsi plus rapide tout en restant aussi précise. De plus des outils de synthèse comportementale tels que ForteCynthesizer [**?**] permettent de transcrire les langages des niveaux transactionnels tel que *SystemC*, en langage de matériel tels que Vérilog ou VHDL synthétisable sur FPGA. On peut ainsi valider très rapidement des architectures à différents niveaux d'abstraction tout en réduisant les erreurs induites par le passage d'un niveau à un autre.

Ce découpage en niveaux d'abstraction, grâce à l'orthogonalité des calculs et des communications, permet ainsi de raffiner progressivement l'architecture choisie en ayant un partitionnement optimal qui peut être modifié selon les résultats obtenus sans avoir à attendre l'étape d'intégration du processus. Cela représente la base du codesign qui est la méthode choisie pour la réalisation de la couche physique OFDM de la norme IEEE 802.16.

1.2.2 Réutilisation et automatisation de la conception

Un autre avantage de la modélisation transactionnelle est la possiblité de réutilisation qu'elle offre grâce au découpage en modules de l'application et la séparation des

communications et du traitement. Cela est très pratique pour des applications de réseaux sans fil où les protocoles sont en général étendus d'une norme à une autre. On peut ainsi réutiliser des blocs réalisés pour des applications similaires en changeant uniquement les paramètres de simulation. De plus, le code étant réalisé de façon indépendante de la plateforme d'implémentation, cela le rend indépendant et donc très portable ce qui est très utile pour des applications de réseaux sans fil où on veut pouvoir passer d'une plateforme donnée à une autre avec un minimum d'efforts. On peut également appliquer à l'ESL des mécanismes d'automatisation permettant de passer d'un niveau d'abstraction à un autre.
Il existe de nombreux outils permettant d'automatiser le passage d'un niveau algorithmique au niveau RTL (par ex.[?]) ou d'un niveau transactionnel au niveau RTL[?],[?]. Au niveau algorithmique, des outils graphiques (e.g. *Simulink, Labview*) permettent de visualiser les architectures et les simulations. De nombreux outils de conception bas niveau tels que *XPS* de *Xilinx*, *Quartus II* de *Altera* permettent d'automatiser la synthèse sur FPGA des éléments RTL des systèmes tout en intégrant des outils visuels de simulation des échanges de processus et de signaux au niveau RTL comme c'est le cas de l'outil Modelsim qui supporte les langages VHDL et Vérilog.

Un environnement optimal [?] serait donc celui offrant des interfaces de communication avec les outils visuels algorithmiques et les outils de synthèse RTL tout en offrant des outils graphiques de simulation au niveau transactionnel. *Space Codesign* à travers ses outils de profilage et de simulation associé à des outils intégrés de génération de code RTL, fournit une architecture RTL complète du modèle à implémenter sur un FPGA de *Xilinx* faisant ainsi le lien avec le niveau RTL. De plus, en lui intégrant des interfaces de co-simulation avec l'environnement *Matlab-Simulink*, on obtient une passerelle entre le niveau algorithmique et le niveau système.

1.2.3 Méthodologie de Conception dans Space Codesign

Space Codesign est une plateforme ESL de conception au niveau système, basée sur le langage de description *SystemC* permettant une modélisation de systèmes au niveau comportemental. Bien que souvent considéré comme un langage, *SystemC* [?] est en fait un ensemble de librairies C++(classes, fonctions, macros) permettant une modélisation au niveau système. Il permet de représenter les signaux, ports et intègre la notion de processus concurrents tels que les modules et threads associés très utiles pour une modélisation matérielle, mais également purement logicielle.

1.2.3.1 Notions de bases : ports, signaux

Un module *SystemC* est généralement représenté comme une boîte comportant des ports ou des interfaces de communication. Dans une communication *SystemC*, il y a généralement un maître qui envoie les données et une cible qui les reçoit ou vice-versa. Ce type de communication est généralement appelé une transaction (voir figure 1.4). Ainsi le premier module est connecté au second et peut faire des appels de fonctions à partir de ses ports qui implémentent les interfaces du second module ou l'inverse. De plus le code source du moteur de simulation étant accessible, il est possible de

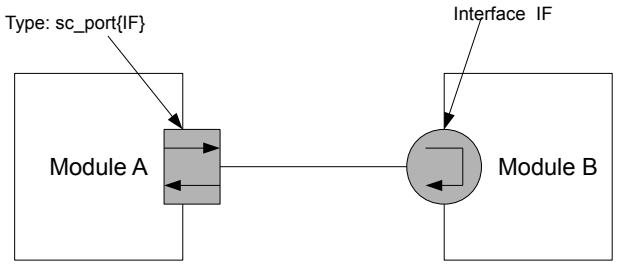


Figure 1.4 Communication dans SystemC

l'étendre et de l'adapter selon des besoins spécifiques tel que c'est le cas dans *Space Codesign*.

1.2.3.2 Librairies de Space Codesign

La plateforme *Space Codesign* [**?**] se subdivise en deux parties : la librairie *SpaceLib* et l'environnement virtuel de développement *SpaceStudio*. La librairie *SpaceLib* comprend tous les éléments de base nécessaires à une modélisation transactionnelle sous *Space Studio*

Ainsi, une application sous *Space Codesign* sera divisée en modules comportant un thread principal et des fonctions. Tous les aspects de communications des modules (ports, interfaces, bus, mémoires, timers, signaux,...) sont gérés par les librairies de *Space Codesign*. Dans *Space Codesign* deux modules peuvent communiquer soit à travers un bus ou par des liens directs. La communication se fait par un mécanisme de rendez-vous (transaction), c'est à dire que chaque module écrit et/ou lit dans un autre module des messages. Lorsqu'il s'agit d'une communication par bus, les modules sont associés à des adaptateurs recevant les messages et les stockant dans des files (fifos); ce sont les adaptateurs qui s'occupent de tous les aspects de communication avec le bus tels que le protocole de bus associé, la disponibilité du bus, la synchronisation des transferts, etc... Cela permet ainsi d'abstraire l'application de l'implémentation. Il existe deux niveaux d'abstraction dans la plateforme. Le niveau Elix et le niveau Simtek. Le premier niveau est le nivau fonctionnel (avec ou sans la présence du temps) correspondant à la première catégorie TLM; il permet de réaliser l'aspect algorithmique du système sans rentrer dans les détails d'implémentation (mémoires, processeurs, bus, etc..). Le deuxième quant à lui est situé juste au dessus du niveau RTL (circuits logiques) correspondant aux catégories 2 et 3 du TLM. C'est le niveau BCA (Bus Cycle Accurate) faisant intervenir le temps et la simulation au cycle

d'horloge près. Au niveau Elix, le protocole de bus associé n'est pas défini de même que les éléments associés tel que le microprocesseur, les mémoires, les contrôleurs d'interruptions, les UART etc. C'est au niveau Simtek que l'exploration architecturale est effectuée (voir figure 1.5).

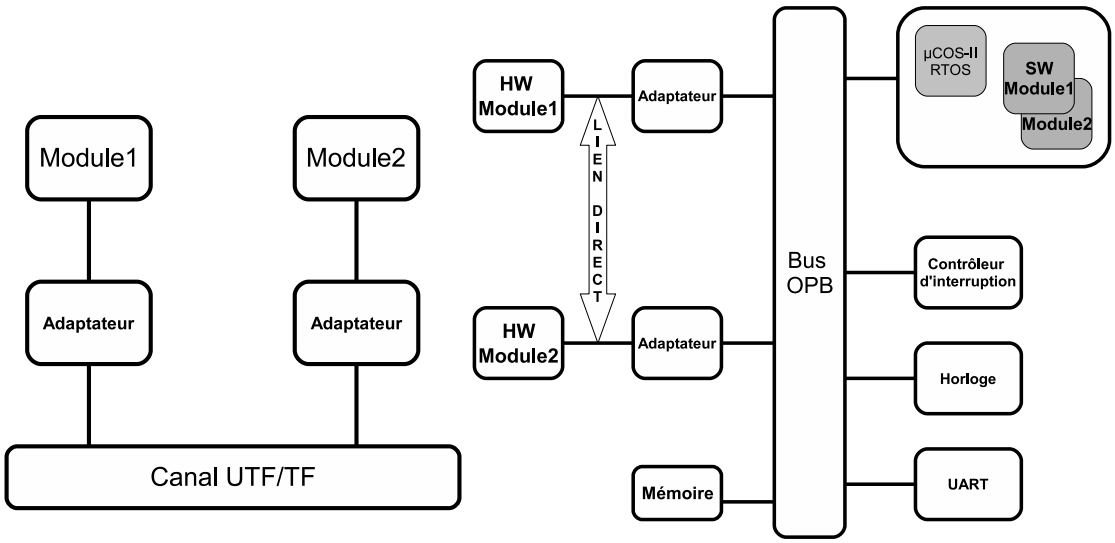


Figure 1.5 Communications Elix et Simtek dans Space

1.2.3.3 Exploration architecturale

La plateforme supporte plusieurs protocoles de bus dont le bus OPB(On-chip Peripheral Bus) et le bus PLB (Processor Local Bus) de IBM CoreConnect. Étant donné que la plateforme *Space Codesign* est conçue de telle sorte à générer des composantes nécessaires à une implémentation dans la plateforme *EDK* de *Xilinx*, elle supporte également les processeurs MicroBlaze et PowerPC associés. De plus, des adaptateurs permettant d'encapsuler les modules (de les abstraire) ainsi que les composantes architecturales telles que les mémoires, les contrôleurs de bus, les interrupteurs, les timers et les simulateurs de jeux d'instruction des processeurs sont

automatiquement générés permettant ainsi une exploration architecturale efficace du système. L'exploration architecturale peut ainsi être décomposée en 3 parties :

1. Partitionnement logiciel/matériel.

   À ce niveau, on détermine les modules implémentés en matériels et ceux implémentés en logiciel (c'est-à-dire dans le processeur)

2. Allocation et instantiation des tâches du processeur.

   On détermine le nombre de processeurs du système et on lie chaque tâche définie au processeur correspondant.

3. Alllocation des bus et instantiation des composantes matérielles.

   On détermine le nombre de bus du système auquels on lie les éléments matériels (modules et composantes architecturales).

D'autre part, les messages échangés dans *Space Codesign* sont de deux sortes: bloquants ou non-bloquants en lecture ou en écriture et chaque module est muni d'un identifiant permettant ainsi de faciliter les communications. Lorsqu'un module envoie un message bloquant à un module ou composant, il ne peut continuer son processus qu'une fois que le receveur lit le message envoyé ce qui n'est pas le cas si l'opération est non-bloquante. Les opérations effectuées sur les messages sont des lectures et écritures. Le format d'une opération dans *Space Codesign* est du type suivant:

```
        status = operation(ID destinataire,
                priorité du message,
                type,
                pointeur vers le message
                );
où :
        priorité du message : la variable SPACE_NORMAL_PRIORITE
        type : SPACE_BLOCKING ou SPACE_NON_BLOCKING
        ID destinataire :identifiant du module destinataire.
```

En plus d'offrir une librairie complète permettant une exploration architecturale, *Space Codesign* permet l'accès à des outils graphiques de simulation tel que *SpaceMonitor* [**?**]. Ce logiciel facilite l'évaluation des performances des configurations des architectures définies.

1.2.3.4 Profilage

L'analyse des résultats est basée sur une observation continue et de façon non-intrusive des calculs et des communications au niveau matériel et logiciel. De cette façon, le temps de simulation interne n'est pas perturbé et le code ne nécessite aucune modification de la part du concepteur.

Au cours d'une simulation, des évènements sont envoyés au contrôleur *SystemC* sous forme de notifications. Celui-ci évalue et collecte les informations qui sont stockées dans un fichier ou envoyés vers un générateur de métriques externe à la simulation. Ce dernier évalue les performances et émet des statistiques qui sont par la suite envoyées soit vers une interface graphique (GUI) ou vers un programe d'analyse (API). La plateforme se charge de générer des macros permettant l'observation et l'analyse de ces résultats.

De plus, les modules matériels n'influencent pas le temps de simulation alors que

dans les modules logiciels, les instructions sont exécutées cycle par cycle sur l'ISS (processeur) et peuvent de ce fait modifier le temps global de simulation. Ainsi, la plateforme s'assure de préserver le temps de simulation en générant de façon automatique les macros chargées de collecter les données générées lors de la simulation. À travers ces données, on obtient des information sur le degré de parallélisme du système permettant ainsi de déterminer son orientation de type flot de données (matériel) ou contrôle (logiciel) ce qui détermine le choix du partitionnement logiciel/matériel destiné à être intégré sur un FPGA..

1.2.4 Vers un modèle RTL dans EDK

La plateforme fournit toute l'architecture matérielle destinée à la plateforme *EDK* (Embedded Development Kit) de *Xilinx*. Elle inclut tous les outils nécessaires à une implémentation au niveau RTL. L'outil Forte Cynthesizer, intégré à *Space Codesign*, permet de passer du niveau système au niveau RTL. Il peut synthétiser les opérations arithmétiques et logiques , le flot de contrôle, les boucles, les appels de fonctions, les tableaux de même que les structures définies par le concepteur. Toutefois, l'allocation dynamique de mémoire n'est pas supportée ce qui fait que les modules TLM et les générateurs des communications doivent supporter ces restrictions. De plus, les modules synthétisés par l'outil occupent généralement plus de ressources que ceux qui auraient pu être faits manuellement, car ces derniers sont optimisés de façon spécifique pour chaque module alors que les optimisations sous Forte Cynthesizer sont générales à tous les modules.

Le passage de la plateforme *Space Codesign* à la plateforme RTL *EDK* est assurée par l'outil GenX de Space Codesign. Il remplace tous les modèles fonctionnels des bus, adaptateurs, processeurs, mémoires et périphériques par leurs équivalents RTL

existant dans la plateforme.

GenX génère automatiquement un projet *EDK*, *XPS* avec tous les éléments de l'architecture. Le fichier *MHS* décrivant l'architecture est généré et tous les ports globaux sont établis. Un dossier *pcores* comportants toutes les composantes matérielles du système est créé; c'est dans ce dossier que les modules matériels (obtenus soit matérellement soit par Forte Cynthesizer) de l'application sont placés. Tous ces éléments sont destinés à être placés sur un FPGA comportant une horloge, un système de réinitialisation. Au niveau système, les horloges sont de simples composants possédant un port de sortie qui fournit l'horloge. La réinitialisation est effectuée par un *reset* qui représente un port du FPGA. GenX se charge de générer les paramètres de ces éléments mais l'utilisteur a la possibilité de les modifier. Les modules logiciel ne sont pas modifiés étant donné que le processeur est capable d'appeler le code *SystemC* grâce aux macros de *Space Codesign* ajoutés au linker_script. La simulation s'effectue à l'aide de l'outil Modelsim accessible à partir de *EDK* et permettant de simuler un code VHDL ou Vérilog. Le banc d'essai est généré automatiquement par GenX ce qui permet ainsi de prendre en compte les composantes telles que les bus, les adaptateurs, etc. Il est ainsi possible de simuler le modèle logiciel/matériel. Pour les blocs matériels codés manuellement, il est nécesaire de rajouter des interfaces de communication avec les adaptateurs (voir figure 1.5).

On constate que la simulation à ce niveau peut être très lente étant donné la prise en compte des signaux tels que l'horloge, le *reset* ou encore les signaux internes des modules. C'est pourquoi il est important de s'assurer que le modèle système a été correctement validé et ne nécessite pas d'être redéfini. Pour cela, une façon efficace de valider ce niveau passe par une cosimulation et une vérification avec le modèle

algorithmique.

1.3 Validation haut niveau : La cosimulation SystemC/Simulink

Tel que précisé plus haut, le choix de conception privilégié des développeurs d'applications de traitement de signal passe par l'utilisation des librairies de Simulink qui permettent de réaliser et de valider à haut niveau.

Matlab est un environnement de développement créé par la société MathWorks. Il s'agit également d'un langage de programmation axé sur le calcul numérique. Quant à Simulink, il s'agit d'une plateforme de simultaion multidomaine et de modélisation de systèmes dynamiques de MathWorks basé sur *Matlab*. Il fournit un environnement graphique et un ensemble de bibliothèques notamment celles comportant des blocs de modélisation permettant le design, la simulation et le contrôle de systèmes de communications et de traitement du signal. Il est intégré à *Matlab* fournissant ainsi un accès immédiat à ses nombreux outils de développement algorithmique, de visualisation et d'analyse de données.

Tel que présenté auparavant, la méthode préférée des concepteurs d'applications de traitement de signal tel que le **Wimax** consiste en une réalisation haut niveau de modèles en bande de base (c'est à dire sans la partie modulation du signal) qui seront par la suite convertis en blocs RTL. Ainsi sur le site de MathWorks, il existe de nombreux exemples de modélisations de réseaux sans fil tel que le **Wifi** et le **Wimax**. Les références [**?**] et [**?**] montrent respectivement un exemple de modèle en bande de base de la couche physique IEEE 802.11a représentant le Wifi et un autre modèle en bande de base de la couche physique et MAC IEEE 802.16e représentant le **Wimax**. Ces deux modèles ne diffèrent que par les paramètres et certains blocs. Ces modèles servent de référence à l'implémentation de notre modèle de couche physique OFDM sous Simulink..

De nombreux outils existent, permettant de combler l'écart entre le niveau algorithmique et le niveau RTL [?]. Grâce aux librairies disponibles, il est possible de bâtir une architecure fonctionnelle complète d'un modèle de réseau sans fil à un niveau algorithmique. Toutefois les outils à ce niveau, ne permettent pas de descendre à des niveaux de conception plus bas (BCA ou RTL); c'est pourquoi de nombreuses recherches ont été effectuées afin d'interfacer un modèle fonctionnel (*Simulink*) et un modèle BCA (*SystemC*) permettant ainsi de valider les spécificités du système à concevoir.

Plusieurs travaux ont été effectués au préalable afin de mettre en rapport des systèmes hétérogènes. Ainsi en [?], les auteurs nous présentent des techniques de cosimulation de systèmes hétérogènes. Il s'agit d'une part d'interfacer *SystemC* avec Esys.Net et de l'autre d'interfacer *Matlab-Simulink* avec Esys.Net. Dans cette approche, les simulateurs communiquent par différentes méthodes: soit par COM, par TCP/IP ou par mémoire partagée pour l'échange de données. A tour de rôle, la synchronisation est contrôlée soit par rendez-vous (TCP/IP) soit à travers une horloge commune (COM, mémoire partagée) par les différents systèmes en présence. Cette approche est très intéressante, car elle permet de mettre en rapport des systèmes à priori incompatibles tel que c'est le cas de notre étude.

Le modèle présenté en [?] est une technique de co-simulation par évènements: la partie *SystemC* est discrète et celle de *Simulink* continue. La co-simulation s'effectue donc par synchronisation des simulateurs *SystemC* et *Simulink*. Le modèle *SystemC* peut envoyer deux types d'évènements : un évènement d'échantillonnage et un de mise à jour tandis que le modèle *Simulink* n'a qu'un évènement d'état qui précise l'état du système à un instant donné. Il s'agit d'un évènement imprévisible.

Cette co-simulation nécessite un certain nombre d'interfaces (S-Functions) permettant à *SystemC* et *Simulink* de communiquer. Du côté *SystemC*, deux interfaces sont nécessaires: la première reçoit les informations venant de *Simulink* et se synchronise

avec les évènements arrivants c'est à dire les signaux de données (de type sc_signal ou sc_fifo) et les évènements associés à ces signaux. Elle convertit les données (type double) venant de *Simulink* en un type *SystemC* (sc_bit ou sc_vector). La seconde interface *SystemC* est celle de sortie c'est à dire celle qui envoie les informations vers *Simulink*. Elle se charge également de la synchronisation et de la conversion des données.

Du côté *Simulink*, il y a quatre interfaces: les deux premières sont les interfaces d'entrée et sortie réciproques de celles de *SystemC*; la troisième est l'interface d'état, c'est à dire celle chargée d'envoyer les évènements détectés lors de la simulation et qui effectue les changements de contexte; quant à la dernière, elle se charge de la synchronisation en plaçant des points d'arrêt utilisables par le solveur *Simulink* (temps discret ou continu) lors de la configuration des paramètres de simulation. Cette méthode bien qu'intéressante nécessite toutefois de nombreux changements de contexte (passage de *SystemC* à *Simulink* et vice versa).

En [**?**] et [**?**], les auteurs utilisent une librairie *Matlab Engine* disponible dans l'outil *Matlab* afin d'échanger des données entre un code *SystemC* et un code *Matlab*. Dans cette méthode, le partage de données s'effectue par mémoire partagée à travers l'espace de travail *Matlab* où les variables du système sont stockées. Le code *SystemC* appelle *Matlab Engine* afin que l'outil exécute des commandes *Matlab* et la synchronisation est contrôlée par la partie *SystemC*. Les auteurs utilisent cette méthode pour valider à l'aide de *Matlab* les sorties du code *SystemC* par des comparaisons et des analyses des résultats obtenus. Cette approche bien que très intéressante n'exploite pas les possibilités offertes par les librairies *Simulink* car elle nécessite d'être étendue.

Ainsi en [**?**], les auteurs vont plus loin en rajoutant une interface entre *Matlab* et *Simulink* grâce aux fonctions intégrées dans ces outils. Les auteurs utilisent la librairie *Matlab Engine* de *Matlab* permettant à un programe C d'appeler des routines *Matlab* et donc d'employer ces fonctions dans un code C/C++. De plus, il

existe des blocs *Simulink* (S-Functions) permettant d'interfacer *Simulink* et *Matlab*. Ainsi le processus de co-simulation *SystemC/Simulink* se déroule de la façon suivante:

- SystemC appelle/reçoit Matlab

  La communication est initiée par le modèle *SystemC* qui place ses données dans le Workspace *Matlab* à l'aide des routines engOpen(), engput() et engEvalString(). Ainsi *SystemC* est le maître de la communication et *Matlab* l'esclave; c'est également lui qui contrôle la synchronisation de la communication à l'aide des commandes *Matlab* set_param() et get_param() qui permettent d'accéder à des paramètres *Simulink*. Ces deux commandes servent à lancer, arrêter et continuer une simulation. D'autre part cette technique suppose que la solveur de *Simulink* admette des évènements discrets (contrairement à la méthode 1). Cela simplifie énormément la co-simulation car à chaque cycle de *SystemC* correspond un seul cycle de *Simulink*.

- Matlab communique avec Simulink

  Comme décrit plus tôt, cette communication se fait à l'aide de S-Functions. Il s'agit de pseudo-langages de description de blocs *Simulink* écrits soit en langage *Matlab, C/C++, Ada ou Fortran*. Ils sont compilés en fichier MEX (exécutable *Matlab*) et linkés dynamiquement au moment de l'exécution dans *Matlab*. Un template de S-Functions est fournit pour chaque langage. On peut accéder à des exemples en tapant la commande :> sfundemos dans la fenêtre *Matlab* et au template C/C++ en allant à :$(Matlabroot)/simulink/src/sfuntmpl_doc.c. Les données *Matlab* sont récupérées par les S-Functions puis envoyées au modèle *Simulink* sous forme de signaux.

Notre modèle ne nécessitant pas énormément de changements de contexte, nous avons donc opté pour cette dernière approche basée sur *Matlab Engine* plus simple à réaliser et suffisante pour les besoins de notre étude; notre but étant de concevoir notre

application grâce à la plateforme ESL *Space Codesign* tout en exploitant les outils de *Simulink* pour une validation fonctionnelle du système.

1.4 Conclusion

On peut retenir de ce chapitre que l'évolution des systèmes de télécommunications a mené à l'émergence d'une nouvelle technologie de réseaux sans fil appelée le **Wimax**. Cette évolution rapide a nécessité la modification du flot de conception classique d'applications consistant en une évaluation d'un modèle algorithmique qui est ensuite raffiné en modèle RTL. Désormais, un nouveau niveau de conception intermédiaire appelé le niveau système basé sur l'approche ESL offre de meilleurs avantages de conception en permettant à l'utilisateur de travailler à un niveau suffisamment élevé tout en tenant compte des spécificités liées à la conception de système sur puce.

La plateforme ESL *Space Codesign* à travers ses outils de simulation, de visualisation et de profilage permet d'estimer une architecture optimale du système à travers un partitionnement logiciel/matériel. De plus, elle est associée à l'environnement RTL de *Xilinx*, permettant de passer d'une architecture système à une architecture RTL de façon automatique. Pour faire le lien avec le niveau algorithmique, la plateforme *Space Codesign* peut aussi être associée à l'environnement algorithmique *Matlab-Simulink* à travers des interfaces de cosimulation basées sur la librairie *Matlab Engine*.

CHAPITRE 2

ÉTUDE DE LA COUCHE OFDM-PHY

Dans un réseau de télécommunications, la couche physique est chargée de la transmission des signaux électriques entre usagers, c'est à dire la conversion des données binaires en signaux électriques transmissibles à travers un canal de communication. A cause des erreurs de transmission dues au canal de communication (l'air), le réseau doit se munir de mécanismes de détection et de correction des erreurs. Cela se traduit par une protection des données à la transmission et à la réception. Ces mécanismes sont appelés FEC (forward error correction) et sont constitués de 3 méthodes: RS-CC (reed-solomon concatenated with convolutional coding), BTC (block turbo coding) ou CTC (convolutional coding). Seule la première est indispensable, les deux autres étant optionnelles. Le canal de communication étant l'air, les informations sont transmises sous forme d'ondes électromagnétiques; des opérations de mise en forme des données binaires en signaux continus sont effectuées grâce à des techniques de modulation diverses. Le choix de la modulation est défini en fonction du type de données à transmettre donc du débit à atteindre. La norme IEEE 802.16 supporte les spécificités de 4 types de couches physiques; celle étudiée est la troisième et est définie pour des transmissions en dessous de 11 GHz.

Dans ce chapitre, nous présentons les spécifications du modèle de couche physique OFDM telles que ses paramètres, la chaîne de codage et décodage des données et leur mise en forme avant l'envoi sur le support de communication (air).

2.1 Conceptes de base

Il s'agit de définir les différents éléments et paramètres permettant de qualifier un canal de communication. La couche physique OFDM comme son nom l'indique est basée sur l'OFDM. Tel que présenté auparavant, la transmission sans fil nécessite la transformation des informations binaires en ondes électromagnétiques. Cette opération s'appelle modulation et la transmission associée est apppelée transmission radiofréquences. Le canal ne se comporte pas toujours de la même manière; il est sélectif car il dépend de la fréquence du signal. Certaines fréquences sont transmissibles plus rapidement que d'autres et peuvent être atténuées lors de la transmission du fait de la nature électromagnétique de l'onde; le signal peut donc être dispersé ou subir une distorsion dans le temps. Cela entraîne un phénomène nommé interférence inter-symboles (ISI). Cette interférence peut également être due aux trajets multiples d'un signal à cause des nombreux obstacles qu'il peut rencontrer (arbres, immeubles, etc..). Ainsi pour des applications de réseaux à débits très élevés donc à large bande, cette sélectivité du canal entraîne une dégradation exponentielle des performances et les processus d'égalisation censés compenser ces effets sont d'une grande complexité et nécessitent à tout instant la connaissance de la fonction de transfert du canal. Les modulations multiporteuses permettent de résoudre ces problèmes car au lieu d'utiliser une porteuse unique pour transmettre tous les symboles du signal, elles utilisent des sous-porteuses peu sensibles aux multitrajets et aux atténuations fréquentielles (effet Doppler) et ont des techniques d'égalisation plus simples.

2.1.1 Modulation OFDM

Une porteuse est un signal sinusoïdal de fréquence et amplitude constantes. Elle est généralement représentée dans le domaine fréquentiel car elle correspond à un pic

de largeur quasi nulle. Les modulations multi-porteuses comme l'OFDM[1] consistent à répartir les symboles à transmettre sur un grand nombre de sous-porteuses à bas débit. Chaque symbole occupant toute la bande passante disponible , la transmission se fait de façon parallèle contrairement aux modulations mono-porteuses.

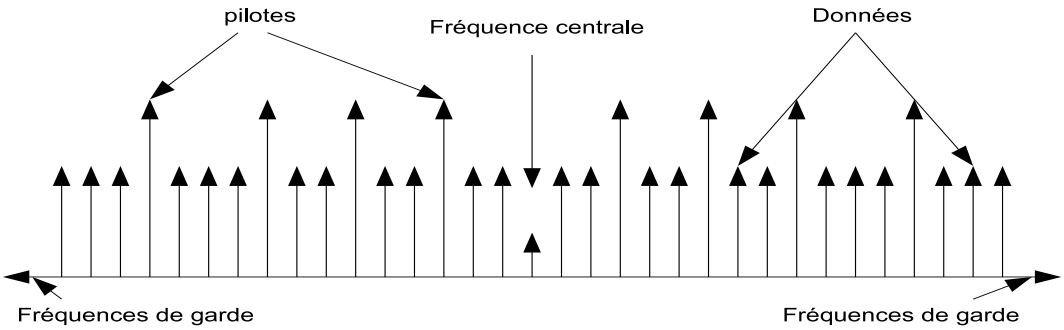


Figure 2.1 Symbole OFDM dans le domaine fréquentiel

Ainsi dans le cas de l'OFDM, pour un train de symboles initial de période $T_s i$, les symboles seront répartis en N trains plus lents et auront alors une durée $T_s = N\Delta T_s i$. Pour répartir les données à transmettre sur les N porteuses, on groupe les symboles c_k par paquets de N. Les c_k sont des nombres complexes définis à partir des éléments binaires par une constellation souvent à modulation de phase (PSK) ou d'amplitude (QAM). La séquence de N symboles $c_0, c_1, ..., c_{n-1}$ forme un symbole OFDM. Le k-ième train de symboles parmi les N trains module un signal de fréquence f_k. Le signal modulé du train k s'écrit sous la forme complexe : $c_k e^{j2\pi f_k t}$ avec $f_k = f_0 + \frac{k}{T_s}$

Le signal total s(t) correspondant à l'ensemble des N symboles réassemblés en un symbole OFDM :

$$s(t) = \sum_{k=0}^{N-1} c_k e^{j2\pi f_k t} \tag{2.1}$$

[1]source : [?]

Cette expression représente la transformée inverse de fourier du signal c[n]=$(c_0, c_1, ..., c_{n-1})$.

Il y a trois types de porteuses d'un symbole OFDM : les données, les pilotes et les gardes. Les porteuses de données contiennent le message à transmettre, les pilotes servent à l'égalisation du signal transmis à travers le canal; quant aux gardes elles servent à empêcher les interférences entre deux symboles OFDM, elles ont une amplitude nulle. Le nombre de porteuses utilisé est donc inférieur au nombre de porteuses disponibles dans le canal. Toutefois les interférences entre symboles (éléments modulés) sont évitées par le fait que les porteuses soient orthogonales. Etant donné que les porteuses sont équidistantes, la distance entre deux porteuses adjacentes est $\Delta f = F_s/N_{FFT}$ où F_s est la fréquence d'échantillonnage du signal et N_{FFT} le nombre total de porteuses. C'est cette orthogonalité qui permet de se prémunir contre les ISI induites par la propagation. Pour éviter les distorsions dues aux multitrajets suivis par le signal, une copie de la fin du symbole OFDM appelée CP est placée au début du symbole. Les paramètres suivants sont ceux fournis par la norme.

Les pilotes sont générés à partir de 11 registres à décalage dont le polynôme générateur est $X^{11} + X^9 + 1$ et les séquences d'initialisation *DL≡11111111111* et *UL≡10101010101*. La transmission à travers un canal peut induire un déplacement de la fréquence centrale du signal et l'influence de la phase du bruit; en insérant les pilotes, on s'assure de retrouver ainsi sa position initiale. Ils sont donc utilisés lors de l'égalisation du signal.

Les préambules qui servent à la synchronisation de la SS par rapport à la BS ont été générés à partir des deux séquences P_{ALL} et P_{SUB}, section Structure des données échangées).

Pour une communication downlink, le préambule est formé de deux symboles OFDM.

Le premier est constitué de 4 fois P_{4x64}, la séquence issue de P_{ALL} formée par les fréquences multiples de 4. Le second est constitué de 2 fois P_{EVEN}, la séquence issue de P_{ALL} formées par les fréquences multiples de 2. Pour une communication UL lorsqu'il n'y a pas de subchannelisation (les 16 canaux sont utilisés) alors on utilise le second symbole OFDM; dans le cas contraire on utilise le symbole obtenu à partir de la séquence P_{SUB}

2.1.2 Paramètres

Soit BW la bande fréquentielle du canal de transmission, N_u le nombre de porteuses utilisées, n le facteur d'échantillonnage de la bande fréquentielle, G le ratio entre la période de CP et celle du symbole OFDM. Les valeurs de ces paramètres sont disponibles à l'Annexe I.4, section Paramètres OFDM.

On obtient finalement l'équation du signal transmis :

$$s(t) = Re\left\{ e^{j2\pi f_c t} \sum_{k=-N_u/2_{k\neq 0}}^{N_u/2} c_k e^{j2\pi k\Delta f(t-Tg)} \right\} \tag{2.2}$$

Le tableau I.2 à la section I.8 de l'Annexe 1 présente les allocations des fréquences en fonction du nombre de sous-canaux utilisés.

La bande fréquentielle est découpée en 16 sous-canaux (subchannelisation en anglais) dans lesquels il est possible d'émettre et il existe 31 combinaisons possibles de sous-canaux. Ces 16 sous-canaux comprenant chacun 12 porteuses peuvent être regroupés selon les cas en 2, 4, 8 ou 16 blocs de sous-canaux. L'allocation de la bande passante se fait de telle sorte que les mêmes sous-canaux ne sont jamais alloués à deux SS en même temps. Le nombre de symboles OFDM dépend donc du nombre de sous-canaux utilisé car les données sont uniquement transmises sur les porteuses correspondantes,

les autres porteuses comportant des données nulles. Ainsi sur les 256 porteuses de la modulation, 8 correspondent aux pilotes (ils ne sont utilisés que lorsqu'on utilise plus d'un sous-canal, sinon leur valeur est nulle), 55 correspondent aux gardes (valant 0), 1 correspond à la porteuse centrale (valant 0). Les 192 restantes correspondent aux données et sur ces 192 porteuses, seules celles correspondant aux sous-canaux sont utilisées. Ce différents paramètres sont utilisés tout au long de la transmission.

2.2 Chaine de codage

Lors d'une transmission de données, la partie la plus importante se trouve au niveau du codage des données car c'est ce qui assure que les données envoyées par le transmetteur pourront être récupérées par le recepteur. Le codage du canal consiste à rajouter de la redondance afin de diminuer et de corriger les erreurs de transmission. La chaîne de codage est constituée de 3 parties: Mélangeur, FEC et Entrelaceur. Elle précède la mise en forme avant la transmission à travers le canal de communication Les deux paramètres les plus importants de la chaîne de codage sont la modulation et le taux de transfert. Ces paramètres determinent la vitesse de transmission des données.

2.2.1 Mélangeur/De-Mélangeur

Le **Mélangeur** (en anglais **Randomizer**) est le premier élément de la chaîne. Il permet de disperser l'énergie des symboles modulés transmis dans l'air. Autrement un pic de puissance serait introduit lors de la transmission ce qui induirait des erreurs au moment du decodage des symboles car il interférerait avec les pilotes. Le **De-Mélangeur** (opération inverse du Mélangeur) consiste à repasser par le Mélangeur.

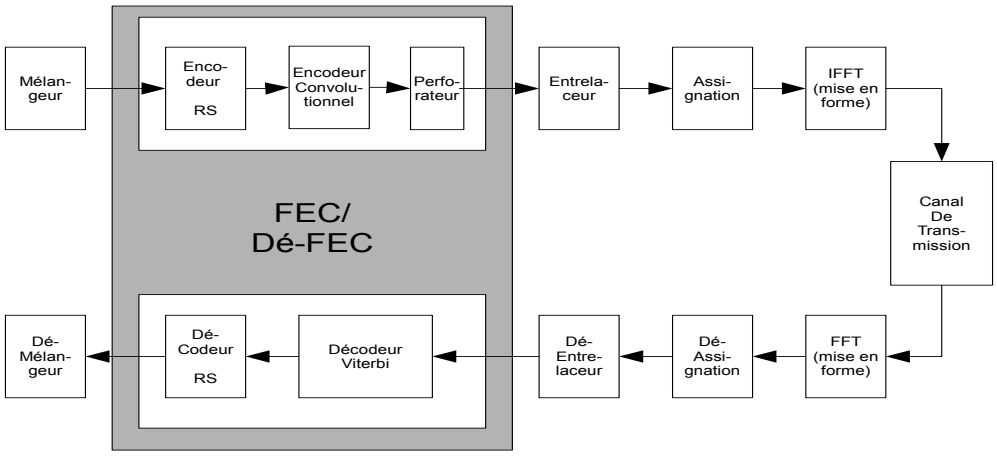


Figure 2.2 Transmission à travers un canal de communication

Le codage est réalisé sur un bloc de données (burst) à transmettre. La taille[2] (nombre d'octets) de ce burst est déterminée en fonction de la modulation, du taux de transfert, du nombre de canaux alloués et du nombre de symboles à transmettre:

$$taille \leq \left\lceil \frac{12 * nbSubchannel * nbSymbols * assignation * taux\_de\_transfert}{8} \right\rceil \tag{2.3}$$

Note: *nbSubchannel* est le nombre de sous canaux, *nbSymbols* le nombre de symboles OFDM à transmettre, *assignation* est le nombre de bits par symbole de la constellation (selon la modulation). La valeur *12* correspond au nombre de porteuses d'un sous-canal.On divise par *8* car même s'il s'agit d'une transmission binaire, les calculs sont effectués sur des octets.On retranche *1 octet* à ce nombre car il sert à revenir à l'état initial (voir partie Convolution).

Selon que la source transmet plus ou moins d'octets que ce nombre, on tronque ou on complète par des octets nuls(*0x00*).

L'opération est effectuée à l'aide de 15 registres à décalage dont le polynôme

[2] $\lceil x \rceil$ est la partie entière supérieure de x

générateur est le suivant:

$$PRBS = 1 + X^{14} + X^{15} \tag{2.4}$$

Ce qui donne l'architecture matérielle suivante:

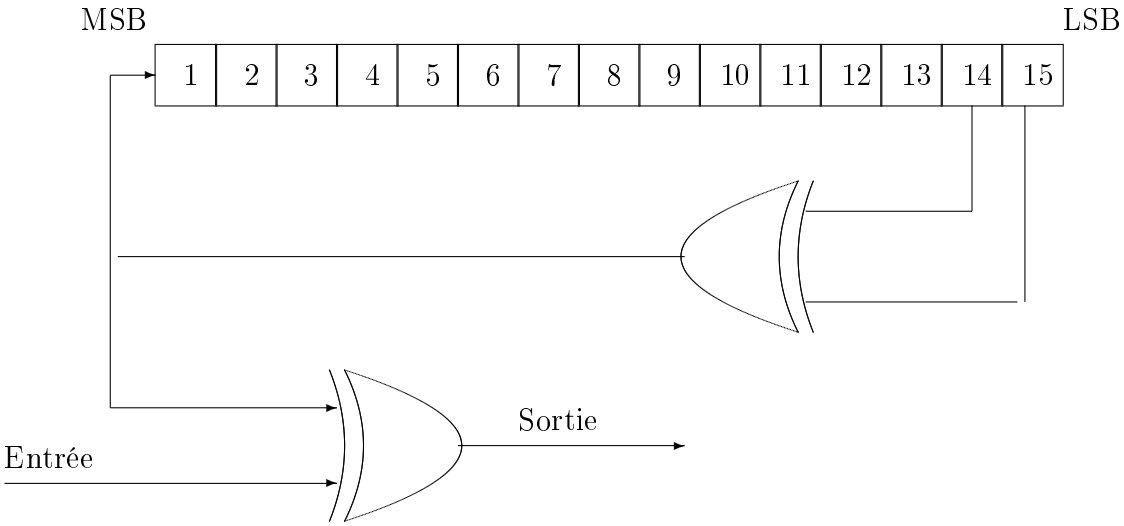


Figure 2.3 PRBS pour le **Mélangeur**

Ces registres sont initialisés par un vecteur d'initialisation ayant différentes valeurs selon la direction de la communication(DL[3] ou UL): La figure 2.4 illustre le format de trame utilisé en mode UL :

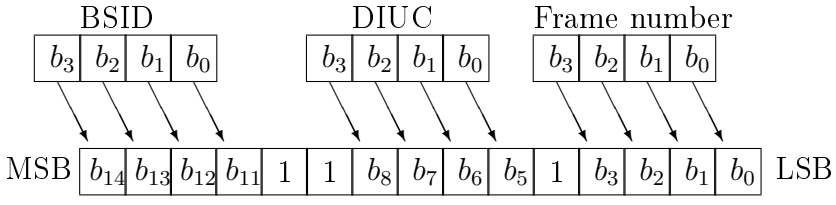


Figure 2.4 Initialisation des rafales en mode ascendant

[3]DL: de la BS vers la SS; UL: de la SS vers la BS

Pour une communication UL (ascendant) le vecteur a le même format que celui des burst DL (descendant).

2.2.2 Encodage FEC (Forward Error Correction)

C'est la partie la plus importante de la chaîne de codage car c'est celle qui effectue les opérations de protection, détection et correction des erreurs. Les codes correcteurs d'erreurs sont des mots d'information auquels on a rajouté de la redondance ce qui donne une meilleure protection aux erreurs car la probabilite d'erreurs est ainsi réduite. Cette partie comporte l'encodeur **Reed-Solomon** et le codeur convolutionnel.

2.2.2.1 Encodeur REED-SOLOMON

Il s'agit d'un code correcteur d'erreurs utilisé dans tous les domaines requérant des données fiables. Il permet de corriger des erreurs et des effacements grâce à de la redondance réalisée par le suréchantillonnage des informations transmises. Une erreur représente le changement survenu sur un 0 ou un 1 après démodulation alors qu'un effacement représente la suppression d'un bit ou octet après passage dans l'encodeur **Reed-Solomon** car seulement une partie des octets générés sont effectivement transmis. Cette opération est réalisée à l'aide d'un polynôme basé sur la théorie des corps de galois *GF*.

Principe de codage: Le Reed-Solomon est dérivé d'un code systématique RS(N,K,T) utilisant $GF(2^8)$. Où:

N est le nombre total d'octets après encodage et vaut 255

K est le nombre d'octets avant encodage et vaut 239

T est le nombre d'octets pouvant etre corrigés et vaut N-K=8

Le code systématique utilise les deux polynômes suivants:

* Pour la génération des symboles de contrôle :

$$g(x) = (x - \alpha^0)(x - \alpha^1)...(x - \alpha^{2*T-1}) \tag{2.5}$$

* Pour la génération des champs de galois :

$$P(x) = 1 + x^2 + x^3 + x^4 + x^8 \tag{2.6}$$

Ce code est raccourci pour avoir une capacité de correction des erreurs variable en fonction du taux de transfert. Ceci nous permet d'avoir des codes RS dérivés du type RS(*N'*,*K'*,*T'*) avec:

$$N' \leq N$$
$$K' \leq K$$
$$T' \leq T$$

D'après [**?**], l'équation clé définissant le codage systématique du Reed-Solomon(n,k,t) est :

$$c(x) = i(x)x^{n-k} + [i(x)x^{n-k}]mod(g(x)); \tag{2.7}$$

Avec :

$c(x)$: polynôme du mot-code de degré *n-1*

$i(x)$: polynôme d'information de degré *k-1*

$[i(x)x^{n-k}]mod(g(x))$: polynôme de contrôle de degré *n-k-1=2T-1*

$g(x)$: polynôme générateur de degré *n-k*

ce qui nous donne le schéma de codage de la figure 2.5 :

Le codage consiste ainsi en une série de décalages des octets reçus qui sont stockés dans un FIFO et qui seront envoyés après les octets de contrôle (*2T'*):

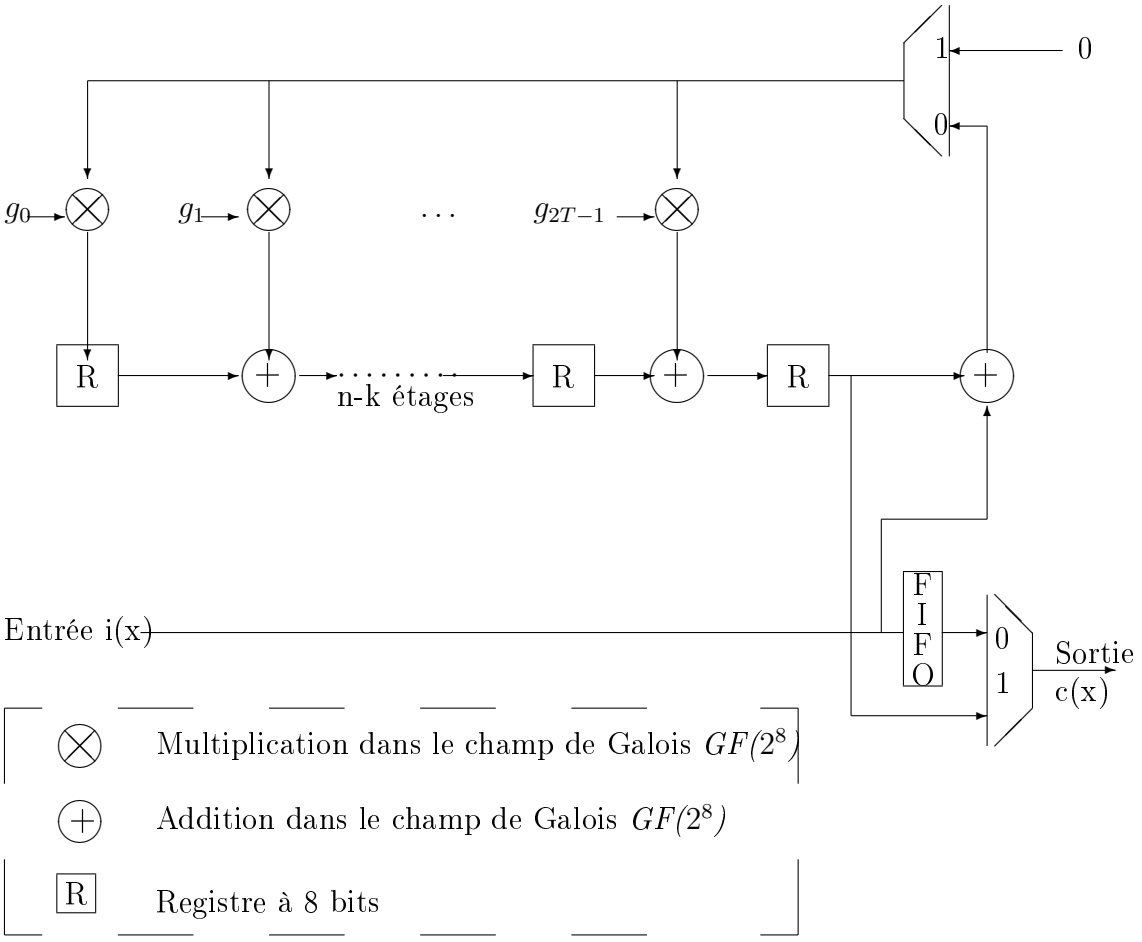


Figure 2.5 Schéma de codage du Reed-Solomon

2.2.2.2 Encodeur Convolution-Perforation

Les codes convolutifs sont des codes systématiques récursifs correcteurs d'erreurs utilisés dans les communications numériques (e.g. téléphonie mobile et communications sans fil). Ils s'appliquent sur des séquences de mots d'information infinies et génèrent des séquences de mots codés infinies.

Ce sont des cas particuliers de codes en blocs avec comme différence la valeur de leur dimension et la longueur. Pour ces codes, chaque bloc de n éléments binaires en sortie dépend non seulement des k éléments binaires présents en entréee mais aussi des m blocs de k éléments binaires précédents,souvent k vaut 1. m+1 s'appelle la

Figure 2.6 Mot code envoyé

longueur de contrainte. Ainsi, le taux de codage R est égal à k/n. De plus, en rajoutant de la redondance cela nous donne des codes plus robustes aux erreurs de communication. Dans notre cas la norme **IEEE 802.16** exige des codes de longueur 7 (donc m=6 et 2^m= 64 états) avec un rendement de 1/2 et sont dérivés de deux polynômes générateurs:

$$G_1 = 1 + D + D^2 + D^3 + D^6 \quad ou\ en\ octal \quad G_1 = 171_{OCT} \tag{2.8}$$

$$G_2 = 1 + D^2 + D^3 + D^5 + D^6 \quad ou\ en\ octal \quad G_2 = 133_{OCT} \tag{2.9}$$

Cette opération se fait à l'aide de registres à décalage pour le delai et de OU exclusifs (XOR) pour l'addition. On obtient ainsi la figure 2.7 à une entrée et deux sorties X et Y pour un taux de 1/2.

Quant à la perforation ou poinçonnement (**Puncturing** en anglais) consiste à supprimer un certain nombre de bits parmi les bits du mot code. Cela permet de modifier le taux de transfert des bits à travers le canal de communication. Ainsi cela augmente la flexibilité du système sans rajouter de complexité. Dans notre cas, les vecteurs de perforation sont donnés dans le tableau 2.1 selon les taux de transfert à obtenir.

2.2.3 Entrelacement/De-Entrelacement

L'**Entrelacement** (en anglais **Interleaving**) est une méthode de permutation des informations utilisée dans les communications numériques. Cette opération ne fait

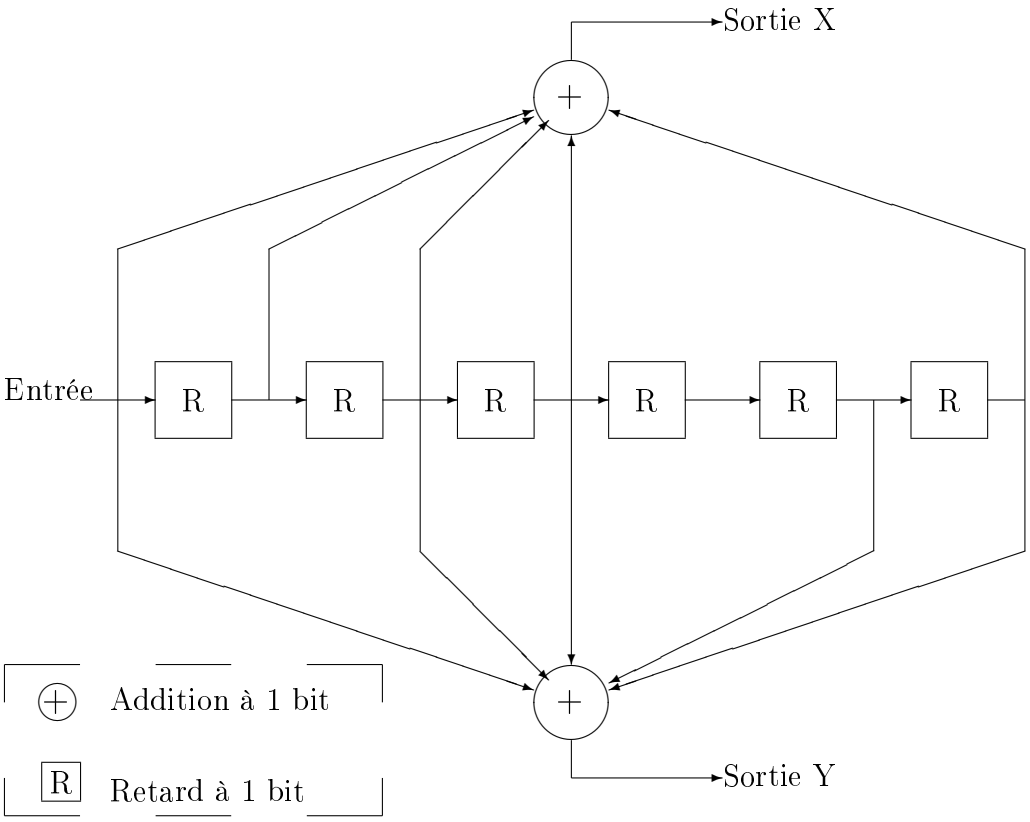


Figure 2.7 Convolution à rendement de 1/2 et de longueur 7

pas de correction ni de détection d'erreurs; elle permet d'éviter d'avoir deux éléments binaires contigus envoyés sur des fréquences adjacentes. Autrement dit, l'Entrelaceur protège les éléments transmis contre les erreurs en changeant de position des bits voisins. L'entrelacement est effectué sur des blocs de mots-codes, dont la dimension varie en fonction de la modulation appliquée et du nombre de sous canaux utilisés. Le tableau 2.2 nous donne la taille des blocs d'entrelacement N_{cbps}.

Deux permutations sont effectuées sur les blocs:

- La première permutation consiste à changer de position deux bits voisins de telle sorte qu'à la modulation ces bits ne se retrouvent pas sur 2 fréquences

Tableau 2.1 Vecteurs de perforation

	Mots de code			
taux de transfert	1/2	2/3	3/4	5/6
distance libre	10	6	5	4
X	1	10	101	10101
Y	1	11	110	11010
XY(Vecteurs de perforation)	X_1Y_1	$X_1Y_1Y_2$	$X_1Y_1Y_2X_3$	$X_1Y_1Y_2X_3Y_4X_5$

Tableau 2.2 Taille des blocs d'entrelacement

	Défaut(16 sous-canaux)	8 sous-canaux	4 sous-canaux	2 sous-canaux	1 sous-canal
	N_{cbps}				
BPSK	192	96	48	24	12
QPSK	384	192	96	48	24
16-QAM	768	384	192	96	48
64-QAM	1152	576	288	144	72

porteuses adjacentes. Elle est donnée par la formule:

$$m_k = (N_{cbps}/12).k_{mod(12)} + \lfloor (k/12) \rfloor \ ^4 \quad k = 0, 1, ..., N_{cbps} - 1 \tag{2.10}$$

Où :

* N_{cbps} est le nombre de bits de transmission d'un bloc d'entrelacement
* N_{cpc} est le nombre de bits codés par porteuse(c'est a dire 1,2,4 ou 6 pour BPSK, QPSK, 16QAM, 64QAM) voir 2.2.4

[4] ce symbole correspond à l'entier inférieur le plus proche

* $s = \lceil N_{cpc}/2 \rceil$ [5]
* k est l'indice du bit codé avant la première permutation
* m_k est l'indice du bit codé après la première permutation et avant la seconde permutation
* j_k est l'indice du bit codé après la seconde permutation

- La seconde permutation consiste à changer de position deux bits voisins de telle sorte qu'à la modulation ces bits soient assignés alternativement entre des symboles de la constellation plus ou moins significatifs; autrement dit entre les porteuses les plus importantes et les porteuses les moins importantes du symbole OFDM. Plus précisémment on obtient :

$$j_k = s.floor(m_k/s) + (mk + N_{cbps} - \lfloor (12.m_k/N_{cbps}) \rfloor)_{mod(s)} \quad k = 0, 1, ..., N_{cbps} - 1 \tag{2.11}$$

La figure 2.8 présente un exemple d'entrelacement pour une modulation 16-QAM et 1 sous-canal utilisé ou une modulation QPSK et 2 sous-canaux utilisés. On obtient d'après le tableau 2.2, $N_{cbps} = 48 bits$.

Soit b_k un bit de la séquence d'entrée avec k =0,1,...,47, la première permutation est présentée à la figure 2.8.

L'entrelacement consiste ainsi à remplir de la gauche vers la droite puis de haut en bas, un tableau dont la taille varie en fonction de la modulation souhaitée et du nombre de sous canaux utilisés tel que c'est le cas dans la norme 802.16. Une fois le tableau rempli on effectue des permutations dans ses colonnes. Pour les modulations BPSK, QPSK il n'y a pas de permutation dans les colonnes. Pour 16-QAM, les permutations se font deux par deux et toutes les deux colonnes tandis que pour 64-QAM les permutations se font trois par trois et cela toutes les trois colonnes. Ainsi

[5] ce symbole correspond à l'entier supérieur le plus proche

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}
b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}
b_{36}	b_{37}	b_{38}	b_{39}	b_{40}	b_{41}	b_{42}	b_{43}	b_{44}	b_{45}	b_{46}	b_{47}

Figure 2.8 Entrelacement à 1 sous-canal et 16-QAM : Avant permutation

b_0	b_{13}	b_2	b_{15}	b_4	b_{17}	b_6	b_{19}	b_8	b_{21}	b_{10}	b_{23}
b_{12}	b_1	b_{14}	b_3	b_{16}	b_5	b_{18}	b_7	b_{20}	b_9	b_{22}	b_{11}
b_{24}	b_{37}	b_{26}	b_{39}	b_{28}	b_{41}	b_{30}	b_{43}	b_{32}	b_{45}	b_{34}	b_{47}
b_{36}	b_{25}	b_{38}	b_{27}	b_{40}	b_{29}	b_{42}	b_{31}	b_{44}	b_{33}	b_{46}	b_{35}

Figure 2.9 Entrelacement à 1 sous-canal et 16-QAM : Après permutation

pour 16-QAM, la colonne 1 reste inchangée, on permutte les lignes de la colonne 2 deux par deux en partant du haut, la colonne 3 reste inchangée et ainsi de suite. Dans le cas 64-QAM: la colonne 1 reste également inchangée; dans la colonne 2 on permute dans l'ordre les ligne 1, ligne 2 et ligne 3 ce qui donne ligne 2, ligne 3 et ligne 1; quant à la colonne 3, après permutation des ligne 1, ligne 2 et ligne 3 on obtient ligne 3, ligne 1, ligne 2 puis on continue avec les lignes suivantes (4,5,6, etc...). Si l'on considère le tableau rempli à la figure 2.8, Pour 16-QAM on aurait après permutation la figure 2.9.

Après avoir terminé les permutations sur le tableau on récupère les bits en sortie en partant de la premiere colonne jusqu'à la dernière et cela de haut en bas. Ce qui donne: le vecteur $[b_0, b_{12}, b_{24}, b_{36}, b_{13}, ..., b_{25}, ..., b_{23}, .., b_{35}]$
L'architecture de l'entrelaceur comporte donc un multiplexeur qui sert à determiner quelle colonne sera permutée en fonction de sa position dans le tableau. La structure des données en sortie est la même que celle des données en entrée; seule leur position dans la trame est changée.

De la même façon, le dé-entrelacement est effectué par l'opération inverse de l'entrelacement. Les équations[6] de désentrelacement sont les suivantes :

$$m_j = s.\lfloor (j/s) \rfloor + (j + \lfloor (12.j/N_{cbps}) \rfloor)_{mod(s)}, \quad j = 0, 1, ..., N_{cbps} - 1 \quad (2.12)$$

$$k_j = 12.m_j - (N_{cbps} - 1).\lfloor (12.m_j/N_{cbps}) \rfloor, \quad j = 0, 1, ..., N_{cbps} - 1 \quad (2.13)$$

Où :

* j est l'indice du bit reçu avant la première permutation
* m_j est l'indice du bit reçu après la première permutation et avant la seconde permutation
* k_j est l'indice du bit reçu après la seconde permutation

2.2.4 Assignation

L'**Assignation** (**Mapping** en anglais) est une opération de mise en forme avant la modulation par la transformée de Fourier. Les symboles OFDM étant obtenus en insérant les pilotes entre les symboles obtenus à partir de l'assignation des constellations; ces symboles sont ensuite modulés à partir d'une transformée inverse de fourier. Cette opération consiste donc à associer à chaque symbole binaire un symbole complexe de la modulation utilisée lors de la communication.

[6] $\lfloor a \rfloor$ est la partie entière de a

Les tableaux des constellations correspondant aux modulations BPSK, QPSK, 16QAM, 64QAM sont présentés à l'Annexe II, Section constellations.

2.2.5 Décodeur FEC

La partie décodage[7] étant laissée à la discrétion de l'utilisateur nous avons choisi des décodeurs inverses des encodeurs c'est à dire un décodeur reed-solomon pour un codeur reed-solomon, un viterbi pour l'encodeur convolutif etc... En effet dans le cas du décodage du Convolueur par exemple, le décodeur LDPC aurait pu être appliqué au lieu du Viterbi. Son efficacité impliquant l'utilisation d'un codeur LDPC, le Viterbi a été jugé plus adapté aux besoins de l'application. De plus, dans le cas du Dé-Modulateur en sortie de la transformée de fourrier, l'utilisation de sorties fermes nécessite que l'entrée du Viterbi soit à entrées fermes même si le code convolutif perforé demande des entrées souples. Pour cela nous avons implémenté un code de Viterbi à entrées fermes modifié. Dans le cas du Reed-Solomon, deux algorithmes sont généralement appliqués : celui d'Euclide et celui du Berlekamp-Massey; nous avons préféré le Berlekamp-Massey dont le code ne nécessite pas une division euclidienne dificile à implémenter. Quant aux décodeurs Mélangeur et Entrelacement, leur structure étant la même que celle de leurs encodeurs respectifs, ils ont déjà été décrits en même temps que ces derniers.

2.2.5.1 Décodeur Viterbi

C'est un décodeur basé sur l'algorithme de **Viterbi** d'**Andrew Viterbi** qui décode un flot de données encodées à l'aide d'un encodeur convolutionnel détecteur et correcteur d'erreurs survenues à travers un canal bruité.

[7] voir Figure 2.2

Le module convolutionnel utilisé dans la partie codage (voir Section 2.2.2.2) étant de longueur 7, cela nous donne un diagramme à 64 états .
L'algorithme de Viterbi [**?**, **?**] peut se faire avec des entrées fermes ou souples, c'est à dire que l'on peut recevoir uniquement les bits codés (entrées fermes) ou les recevoir avec des bits de fiabilité (entrées souples). Etant donné que les codes convolutionnels sont perforés, l'algorithme à entrées souples serait plus réaliste. Toutefois, pour simplifier son implémentation, nous avons utilisé un algorithme à entrées fermes modifié au moment du calcul des distances entre les bits reçus et ceux affectés au treillis. Rappelons que l'algorithme de Viterbi à entrées dures ou souples utilise une représentation en treillis qui est la linéarisation du diagramme à 64 états du codeur Convolutif.

Le treillis est formé de noeuds reliés par des branches. Les noeuds représentent les états du codeur : il y a 2^{m-1} états soit 64 pour m=7 et une entrée à 1 bit valant 0 ou 1. Les branches représentent les transitions d'un état à l'autre selon la valeur de l'entrée. Ainsi, comme on part toujours de l'état[8] initial e_0 et on revient à l'état e_0. Selon l'entrée 0 ou 1, on change d'état et on a une sortie à 2 bits (voir figure 2.7). Le calcul des distances consiste donc à évaluer la distance de Hamming entre 2 bits codés reçus et les 2 bits de sortie correspondant aux 64 états du treillis.

Le décodage par l'algorithme de Viterbi consiste à rechercher dans l'arbre le chemin correspondant à la séquence se trouvant à la distance minimale de la séquence reçue. Ce chemin étant formé des entrées utilisées pour aboutir à la dite séquence. Ainsi le décodage se fait de la manière suivante : à un instant donné t on évalue la distance de Hamming entre 2 bits du code et les sorties pour chaque état du codeur, ensuite on ajoute cette distance évaluée à la valeur des états/noeuds du treillis évaluée à l'instant précédent *t-1* et on détermine le minimum de ces distances. Finalement, on revient à l'état initial en parcourant en sens inverse le chemin minimal (constitué

[8] l'indice de l'état correspond à la valeur de l'état sur m-1 bits

des minimums des états à chaque instant t). Les valeurs décodées sont les entrées correspondant à ce chemin.

Le mot codé étant poinçonné en sortie de la convolution(voir Section 2.2.2.2), le décodage de la convolution comporte 2 sous parties : la dé-perforation et le décodeur viterbi :

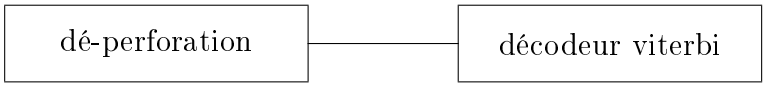


Figure 2.10 Décodage d'un code convolutif perforé

La dé-perforation consiste à remplacer par des zéros les bits supprimés dans la séquence reçue. La position de ces bits est donnée par le vecteur de perforation (voir Tableau 2.1).

Le décodeur Viterbi comporte quant à lui 3 éléments:

- **BMU** (**B**ranch **M**etric **U**nit)

  Il s'agit du module de calcul des distances. A un instant t donné on a 2 bits de la séquence reçue. On évalue la distance de Hamming[9] entre ce mot de 2 bits et la sortie du codeur au niveau de chaque état partant de l'état courant. A l'instant initial on part de l'état e_0. Selon que l'entrée est 0 ou 1 on reste dans l'état e_0 ou on va à l'état e_{31}, ce qui correspond aux sorties : **00** ou **11**. On calcule ensuite la distance de Hamming entre la sortie et le mot de 2 bits reçu à cet instant. On refait l'opération pour chaque mot de reçu.

  L'opération se déroule donc comme suit: à un instant donné t on reçoit en entrée un mot de 2 bits puis on compare ce mot à chaque sortie des branches du treillis ; cela consiste à calculer la distance de Hamming entre ce mot et

[9]voir Annexe 2 section codes correcteurs

la sortie de la branche correspondante. Ces distances sont stockées dans des registres ou des mémoires.

- **PMU** (**P**ath **M**etric **U**nit)

  C'est le module qui accumule les distances évaluées aux différents instants. Pour chaque état on ajoute à la valeur précédente la distance minimale obtenue. Ainsi, à un instant donné t , une fois les distances de Hamming calculées par le BMU, on choisi la valeur la plus probable qu'on rajoute à la valeur du noeud/état du treillis. L'entrée du codeur étant sur 1 bit, il y a 2 valeurs possibles c'est à dire que pour chaque état du codeur, il y a 2 états suivants possibles donc 2 branches de treillis qui arrivent et partent d'un noeud. Le **PMU** évalue donc la somme de la valeur du noeud courant et du minimum des distances de Hamming des 2 branches qui arrivent au noeud. Une fois la valeur des noeuds calculées, le PMU détermine le minimum de tous les noeuds à l'instant donné t.

- **TB** (**T**raceback **U**nit)

  Comme son nom l'indique, c'est le module qui permet de recouvrir le chemin parcouru en sens inverse. Ainsi, on revient à l'état initial e_0 en passant par les minimums des états aux différents instants t. L'information décodée est constituée des entrées du codeur correspondant aux branches permettant d'aboutir aux états de ces minimums. La latence du décodeur Viterbi est donnée par l'instant au bout duquel on commence à recouvrir le chemin; plus elle est élevée, meilleur sera le décodage. On choisi une latence très élevée (7 fois la longueur du code m).

2.2.5.2 Décodeur REED-SOLOMON

Un codeur Reed-Solomon (RS) étant utilisé selon les cas dans le FEC, un décodeur Reed-Solomon est requis pour détecter et corriger des erreurs. L'utilisation d'un codeur Reed-Solomon permet de protéger les informations contre les erreurs dues à la propagation dans l'air d'où le décodeur Reed-Solomon permettant de détecter et de corriger les erreurs induites par la propagation à travers le canal grâce aux symboles de contrôle rajoutés lors du codage. Du fait de la modulation, les codes de Reed-Solomon sont parfois tronqués; les bits effacés sont remplacés par des zéros lors du décodage. Le décodeur Reed-Solomon permet donc de détecter à la fois les erreurs et les effacements, et de les corriger selon certains cas (à condition que le nombre d'erreurs n'excède pas un certain seuil). Rappelons que les erreurs sont les modifications (0 remplacé par 1 ou vice versa) qui sont induites par la transmission à travers le canal de propagation bruité tandis que les effacements sont dus au code RS tronqué en sortie de l'encodeur; en effet, sur les $2T$ octets générés par l'encodeur seuls $2T'$ sont transmis, les autres sont effacés.

Dans la partie codage (Section 2.2.2.1), on voit qu'un code RS est un code systématique[10](n,k,t) où t=(n-k)/2 est le nombre d'octets qu'on peut corriger. $d_{min} = n - k + 1$ est la distance minimale[11] d'un code RS. Pour un code tronqué, on a $\nu \leq t$ le nombre d'octet qu'on peut corriger et ρ le nombre d'octets effacés tel que $\rho + 2\nu \leq 2t$.

[10] voir Annexe I
[11] voir Annexe II

Soit le mot code reçu r(x) de la forme :

$$r(x) = c(x) + e(x) \tag{2.14}$$

où c(x) est le mot code en sortie de l'encodeur Reed-Solomon et *e(x)* est le polynôme des erreurs et effaçements.

A son tour *e(x)* se défini comme:

$$e(x) = \epsilon(x) + \nu(x) \tag{2.15}$$

où :

$\epsilon(x)$ est le polynôme des erreurs

$\nu(x)$ est le polynôme des effaçements

le code RS pouvant corriger ν erreurs tel que:

$\rho + 2\nu \leq 2t$ ou $\rho + 2\nu \leq d - 1$ et $t = \frac{n-k}{2}$

Le décodage d'un code RS se fait en quatre étapes :

1) Calcul du Syndrome et du polynôme de localisation des effacements

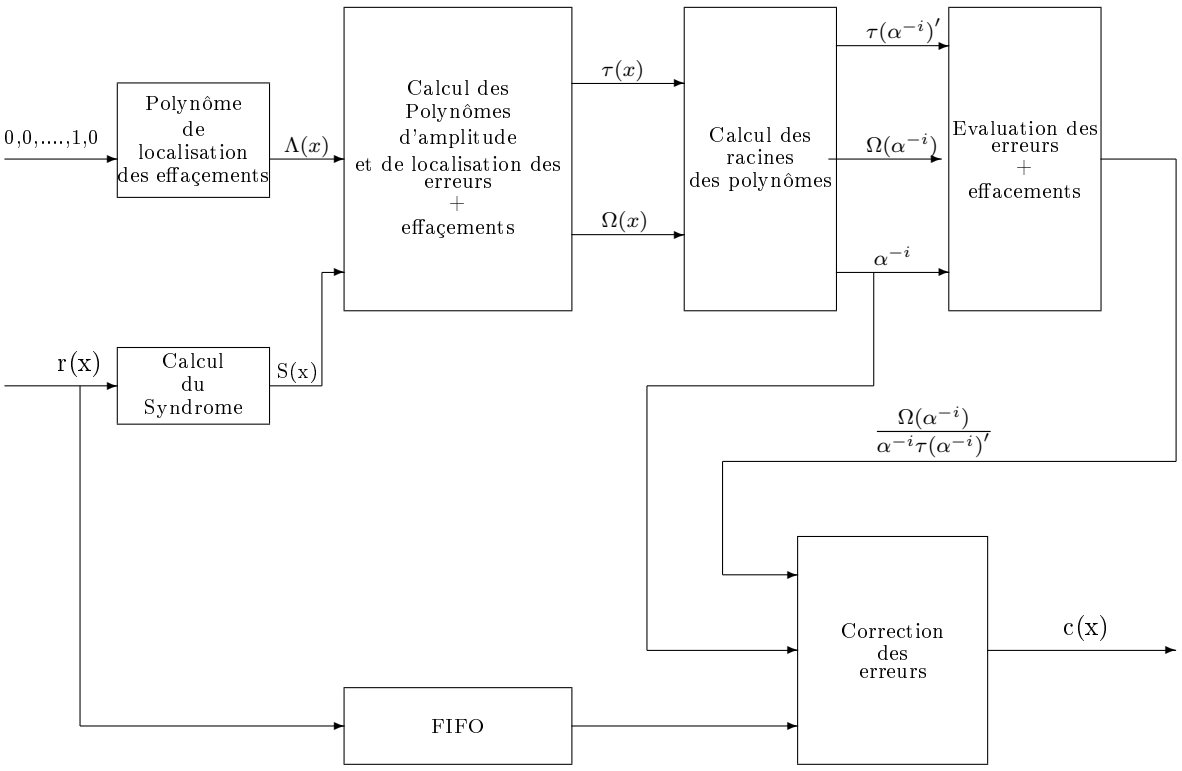
2) Calcul des polynômes de localisation de l'amplitude, des erreurs+ effacements

3) Calcul des racines et évaluation des polynômes

4) Correction de l'erreur

A l'aide de [?] ,[?], [?] et [?] on peut généraliser le schéma de décodage tel qu'illustré à la figure 2.11.

Ainsi le décodage d'un code RS consiste à vérifier si le mot reçu *r(x)* comporte des erreurs et/ou des effaçements (calcul du syndrome), à déterminer des polynômes de localisation de ces erreurs et/ou effaçements, à trouver leur amplitude et leur position puis enfin à les corriger. Il existe différents algorithmes permettant d'effectuer ces opérations.



Légende

r(x) : mot code reçu
c(x) : mot code décodé
S(x) : syndrome calculé
$\Lambda(x)$: polynôme de localisation des effaçements
$\tau(x)$: polynôme de localisation des erreurs+effaçements
$\Omega(x)$: amplitude des erreurs+effaçements
α^{-i} : inverse de la racine α^i du polynôme de localisation des erreurs+effaçements

Figure 2.11 Schéma de décodage de codes de Reed-Solomon

1. Calcul du Syndrome

   Le Syndrome est le reste de la division entre le polynôme reçu r(x) et le polynôme générateur *g(x)* [12]. Il s'agit d'un processus itératif qui s'effectue une fois tous les éléments du code reçus. Etant donné que le codeur RS est basé sur un code systématique *RS(255,239)* , l'opération doit s'effectuer sur *255 octets* . Toutefois, pour un code tronqué on complète par x zéros où *x=239-nombre d'octets reçus* que l'on place en début de séquence ce qui revient à commencer à partir du premier octet reçu les zéros ne changeant pas les registres au départ. Etant donné le format du mot reçu (figure 2.6), il faudra d'abord effectuer les opérations sur les octets d'information et ensuite sur les octets de contrôles. Le code étant tronqué, certains octets de contrôles seront effaçés; ils seront remplaçés par des zéros. Toutefois la norme précise que les octets de contrôle sont envoyés avant les bits d'information, il faudra les stocker avant de les envoyer dans le décodeur. Ainsi de (l'équation 2.14), on obtient:

$$S_k = r(\alpha^k) = \sum_{i=0}^{n-1} r_i \alpha^{ik} = \sum_{i=1}^{\rho} W_i Z_i^k + \sum_{i=1}^{\nu} Y_i X_i^k \tag{2.16}$$

$$\forall k, tel\ que\ 0 \leq k \leq d-2$$

   Où:

   α est un élément primitif de GF(2^m)[13]
   W_i est la valeur du i^{me} élément effaçé
   Z_i est la position connue de cet élément
   Y_i est la valeur de la i^{me} erreur
   X_i est la position connue de cette erreur

   Ainsi on défini le Syndrome par :

$$S(x) = \sum_{k=0}^{d-2} S_k x^k \tag{2.17}$$

[12] voir Section 2.2.2.1
[13] voir Annexe II

Les indices du syndromes (0 à 2t-1) sont issus du polynôme générateur (voir Equation 2.5) De [?] on obtient le schéma de la figure 2.12 du syndrome calculé de façon itérative :

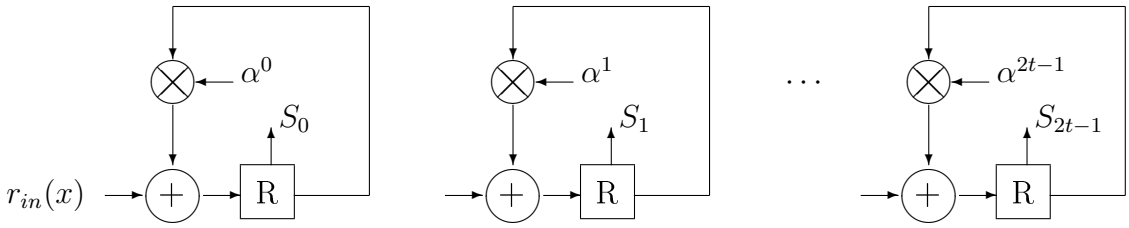


Figure 2.12 Calcul du Syndrome

Le polynôme de localisation des effaçements est quant à lui défini par :

$$\Lambda(x) = \prod_{j=1}^{\rho}(1 - Z_j x) = \sum_{j=0}^{\rho} \Lambda_j x^j \quad o\grave{u}\ \Lambda_0 = 1 \tag{2.18}$$

$\Lambda(x)$ est le polynôme de localisation des effaçements dont les zéros sont les inverses des positions des effaçements. Etant donné la modulation et le taux de transfert, les positions des effaçements sont connues à partir des octets de contrôle tronqués.

2. Calcul des polynômes de localisation de l'amplitude, des erreurs+ effacements
   Le calcul des polynômes est effectué une fois le syndrome et le polynôme de localisation des effaçements trouvés. Il existe différents algorithmes de détermination de ces polynômes; les plus connus étant l'algorithme d'**Euclide** et le **Berlekamp-Massey** . L'algorithme d'Euclide étant un algorithme récursif basé sur une division euclidienne de polynômes nous avons estimé le Berlekamp-Massey plus approprié à notre problème. Pour plus de détails sur l'algorithme d'Euclide consulter la référence [?]. Le Berlekamp-Massey est un algorithme itératif de résolution d'une équation clé à partir de registres à décalage :

$$\Omega(x) \equiv S(x)\tau(x) mod x^d \tag{2.19}$$

$$Avec : d = (n-k+1)\ et\ \ \tau(x) = \Lambda(x)\sigma(x)$$

Plusieurs travaux ont été effectués afin d'éviter la division de polynômes. Un algorithme sans inversion est proposé en [?] et en [?] pour la résolution de cette équation clé. Il est constitué des étapes suivantes:

(a) Conditions Initiales :

$$\mu^{(0)}(x) = \Lambda(x),\ \lambda^{(0)}(x) = \Lambda(x),\ l^{(0)} = 0,\ k = 0\ et\ \gamma^{(k)} = 1\ si\ k \leq 0$$

(b) Pour k=1 à $2T - \rho$:

$$\delta^{(k)} = \sum_{j=0}^{l^{(k-1)}} \mu_j^{(k-1)} S_{k+\rho-j} = \sum_{j=0}^{(d-1)} \mu_j^{(k-1)} S_{k+\rho-j}$$

$$\mu^{(k)}(x) = \gamma^{(k-1)}\mu^{(k-1)(x)} - \delta^{(k)}\lambda^{(k-1)(x)}x$$

$$If\ \ \delta^{(k)} = 0\ or\ 2l^{(k-1)} > k-1$$

$$\begin{cases} \lambda^{(k)}(x) = x.\lambda^{(k-1)}(x) \\ l^{(k)} = l^{(k-1)} \\ \gamma^{(k)} = \gamma^{(k-1)} \end{cases}$$

$$Else$$

$$\begin{cases} \lambda^{(k)}(x) = \mu^{(k-1)}(x) \\ l^{(k)} = k - l^{(k-1)} \\ \gamma^{(k)} = \delta^{(k)} \end{cases}$$

Avec :

$$\mu(x) \equiv \tau(x),\ d-1 = 2T$$

, le code étant dérivé de RS(255,239), alors

$$2T \leq 16$$

Une fois $\tau(x)$ obtenu on remplace dans l'équation 2.19 pour avoir:

$$\Omega(x) = \sum_{j=0}^{\rho+t} \Omega_j x^j$$

3. Calcul des racines

   Le calcul des racines des polynômes se fait à l'aide de l'algorithme du **Chien-Search** . Il s'agit d'un processus itératif qui prend $\tau(x)$, $\Omega(x)$ et les évalue en α^i pour i allant de 1 à 255. Cet algorithme calcule donc $\tau(\alpha^i)$, $\Omega(\alpha^i)$ et $\tau(\alpha^i)'$ la dérivée première de $\tau(x)$ qui est sa partie impaire.

   En se basant sur [**?**] et [**?**], on obtient ainsi l'architecture matérielle de la figure 2.13.

   Cette architecture permet de diviser la latence du ChienSearch en 4, mais augmente sa complexité d'autant. On évalue donc $\tau(x)$ sur les 255 valeurs primitives de GF(2^8). Cette architecture est également utilisée pour l'évaluation de $\Omega(x)$.

   Ce polynôme étant de degré inférieur à $\tau(x)$ ($\deg(\Omega(x)) < \deg(\tau(x)) \leq$ t, t $\leq$ 8).

4. Evaluation des polynômes et correction de l'erreur

   Une fois l'évaluation de $\Omega(x)$ et $\tau(x)$ effectués à l'aide du **ChienSearch**, on utilise l'algorithme de **Forney** pour calculer l'amplitude des erreurs situées dans le mot code reçu. La position des erreurs étant donnée par l'inverse des

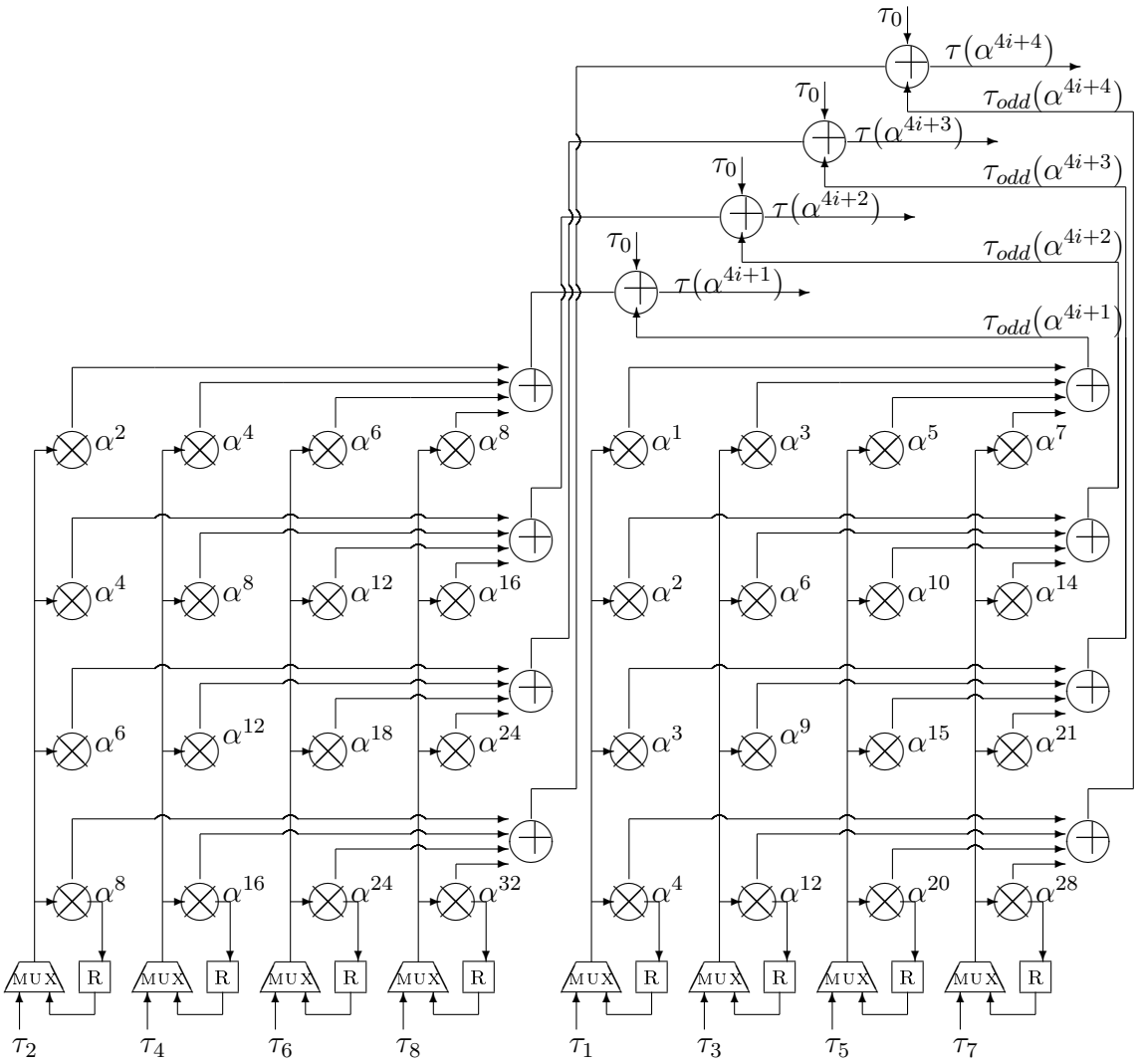


Figure 2.13 Architecture parallèle du Chien Search

racines de $\tau(x)$, autrement dit les α^{-i} pour lesquels $\tau(\alpha^{-i})$ est nul.

En se basant sur ([**?**]) on obtient la figure 2.14.

Etant donné que les effaçements sont dus à la modulation, on n'a pas besoin de les corriger.

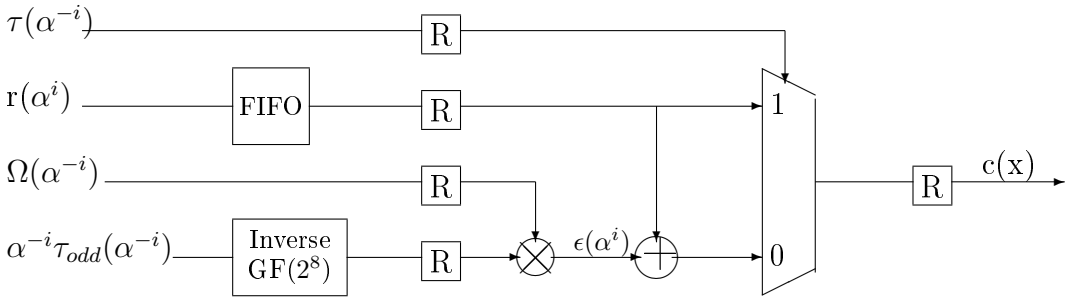


Figure 2.14 Architecture de Forney + Correction des erreurs

2.3 Conclusion

On peut retenir de ce chapitre que la transmission des données nécessite leur protection par la mise en place de mécanismes de codage et de décodage des données. Les paramètres de transmission sont pris en compte par ces mécanismes qui sont constitués d'algorithmes récursifs basés sur des théories mathématiques telles que les nombres de galois dans le cas des codes de Reed-Solomon, les codes convolutifs, l'algorithme de Viterbi. Les éléments de la chaîne de codage ainsi que les paramètres sont spécifiés dans la norme; les éléments de la chaîne de décodage ont été obtenu à travers une recherche bibliographiqueet sur une base complémentaire c'est-à-dire que l'on a choisi un décodeur Reed-Solomon pour un codeur Reed-Solomon par exemple.

CHAPITRE 3

IMPLÉMENTATION DE LA COUCHE PHYSIQUE OFDM DE LA NORME IEEE 802.16

Dans ce chapitre, nous décrivons la façon dont nous réalisons la couche physique OFDM du IEEE 802.16[1] à différents niveaux de conception et à l'aide d'environnements de conception hétérogènes.

La couche physique des réseaux sans fil permet l'accès au médium de communication qui est l'air. Elle est constituée d'une partie numérique effectuant tous les mécanismes de protection des données transmises et d'une partie analogique effectuant les mécanismes de mise en forme des données transmissibles dans l'air sous forme d'ondes électromagnétiques. Grâce aux progrès des SOC, il est désormais possible de construire une architecture entièrement matérielle des réseaux permettant ainsi une réalisation optimale des réseaux sans fil.
Dans cette section nous expliquons la mise en application de la chaîne de communication WMAN-OFDM-PHY spécifiée au chapitre 2 à l'aide des outils de conception présentés au chapitre 1. Nous expliquons aussi le rôle central qu'occupe la plateforme ESL *Space Codesign* dans notre méthodologie de réalisation tout au long du cycle de conception.

Nous sommes partis d'un modèle haut niveau *Simulink* qui nous a permis d'évaluer les performances générales d'une aplication de réseaux sans fil telles que les erreurs de constellation dues à l'estimation du canal de propagation ou encore le taux d'erreurs binaires par paquet envoyé.

[1] que nous simplifierons par **Wimax**

Partant de ce modèle algorithmique, nous avons réalisé un modèle système à l'aide de la plateforme ESL *Space Codesign*. Ce modèle représente le coeur de notre travail, car il nous a permis, à travers des simulations successives et un profilage avancé des résultats d'obtenir un modèle optimal en latence et en surface à synthétiser sur FPGA. Le modèle RTL (obtenu soit par synthèse du second ou manuellement à l'aide de librairies existantes) a été validé par synthèse sur un FPGA de *Xilinx*. Afin de faire une validation entre le niveau algorithmique et le niveau système, des interfaces de cosimulation ont été réalisées.

3.1 Vers un modèle en bande de base

Pour simplifier l'application, nous l'avons représentée en bande de base c'est à dire que nous n'avons pas représenté la modulation RF (filtres passe bas, passe haut, convertisseurs analogiques numériques, modulation RF, etc) mais uniquement la transmission numérique. En outre, tel que le prévoit la norme, il existe trois façons de représenter la couche phyique : en mode TDD, FDD et Mesh. Etant donné les deux modes de communication possibles: une liaison descendante de la BS vers la SS (Downlink en anglais) et une liaison ascendante de la SS vers la BS (Uplink en anglais), nous avons réalisé un modèle TDD Uplink (UL) englobant toutes les spécificités de la couche physique étudiée et permettant l'application du découpage en sous-canaux[2]. Toutefois le modèle réalisé supporte la communication Downlink ce qui pourrait permettre une communication PMP (un sender et plusieurs receveurs).

La figure 3.1 présente le flot de conception de l'application. La partie «Spécifications» correspond à l'étude approfondie de la norme ainsi que des blocs constitutifs de la chaîne de codage. Le modèle *Simulink* correspond au modèle algorithmique de

[2]de l'anglais subchannelization

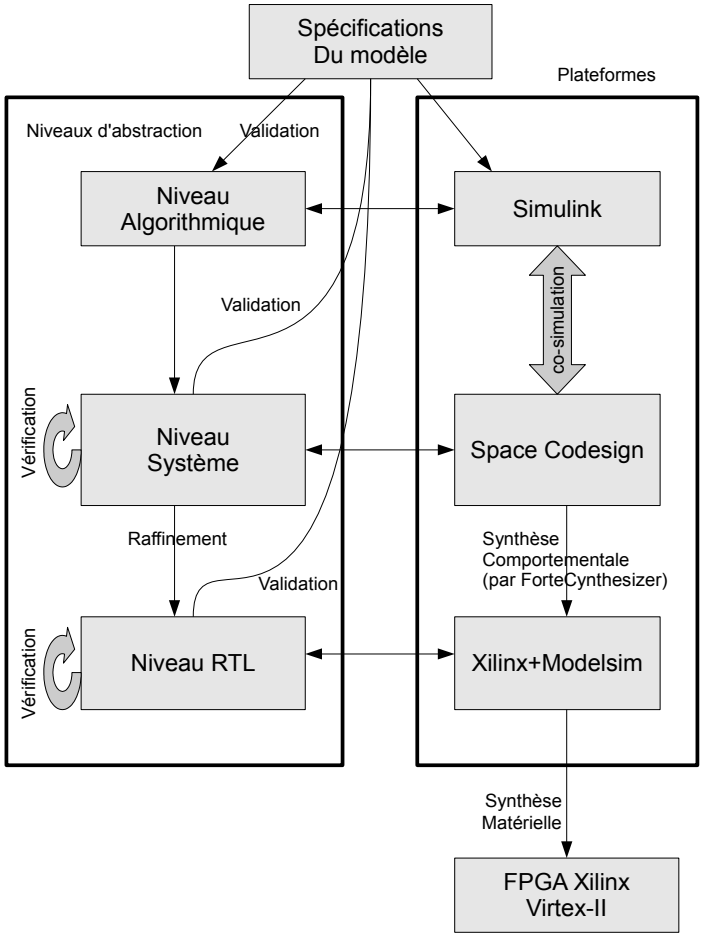


Figure 3.1 Flot de conception du modèle de couche-PHY-OFDM

l'application. Elle comporte une partie numérique correspondant à la chaîne de codage et une partie analogique représentant la modulation et la transmission à travers un canal de bruit blanc gaussien (AWGN) dispersif. Les modèles Elix et Simtek correspondent au niveau système et sont constitués uniquement de la partie numérique du modèle. La cosimulation avec *Simulink* permet de valider le modèle à haut niveau en rajoutant au modèle Elix de *Space Codesign* la partie analogique de *Simulink*. Cela permet ainsi d'obtenir une simulation complète de l'application. Les modèles RTL sont obtenus de deux façons différentes: le premier par synthèse comportementale du niveau Simtek à l'aide de l'outil Forte Cynthesizer, le second à l'aide des blocs de la librairie de *Xilinx* (IP cores en anglais).

Le modèle *Simulink* a été implémenté selon les spécifications suivantes :

1. une communication en liaison ascendante UL pour une seule station de base en supposant que l'initialisation a déjà été réalisée (en effet un modèle à plusieurs stations aurait nécessité trop de ressources). Cela permet d'utiliser le découpage en sous-canaux nécessaire uniquement pour une communication UL (voir tableau I.2 présenté au chapitre 2). La communication UL pour une seule station consiste en une seule rafale (burst en anglais) équivalant à une trame et comportant un ou plusieurs symboles OFDM.

2. les 7 types de modulation spécifiés dans la norme (BPSK-1/2, QPSK-1/2, QPSK-3/4, 16QAM-1/2, 16QAM-3/4, 64QAM-2/3, 64QAM-3/4) ont été représentés donnant ainsi différents débits et donc vitesse d'exécution.

3. le modèle utilise une transformée de fourier de 256 points pour la mise en forme des données.

4. les tests ont été effectués pour les différents sous-canaux 1, 4, 8 et 16 tel que prévu par la norme.

3.2 Modèle Simulink

Il s'agit du modèle fonctionnel haut niveau du système. C'est également le modèle servant à valider l'application. A la chaîne de codage et décodage ainsi qu'aux opérations de mise en forme des données, ont été rajoutés certains blocs servant à la transmission. Le schéma peut se décliner en trois parties: envoi et réception des données, codage et décodage des données, modulation et transmission des données. Ce modèle présenté à la figure 3.2 a été entièrement réalisé à partir des éléments disponibles dans les bibliothèques de traitement de signal et de communication disponibles sous *Simulink*.

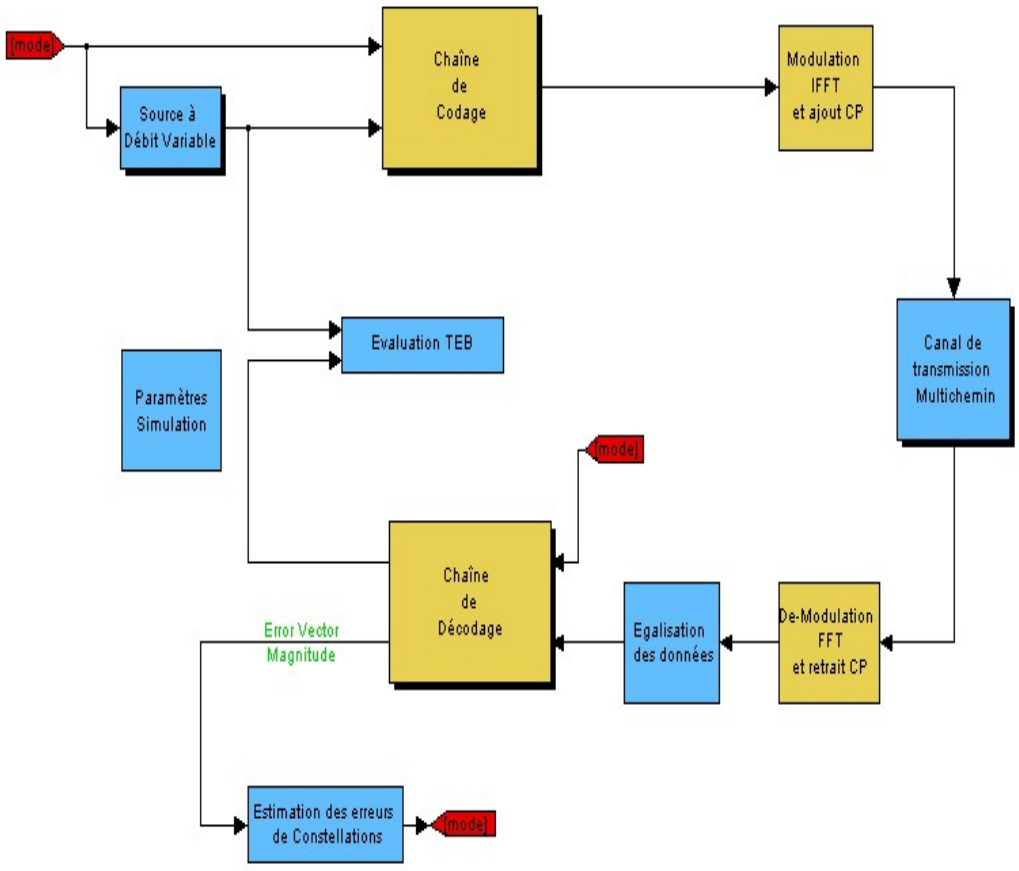


Figure 3.2 Modèle Simulink

3.2.1 Envoi et Réception des données

Dans la figure 3.2, la partie de gauche est constituée des blocs Paramètres, Source à débit variable, Évaluation du taux d'erreurs binaires par paquets et Estimation des erreurs de constellations après démodulation.

- Paramètres

  Ce bloc calcule tous les paramètres de la simulation (voir figure 3.3). Il fait appel à une fonction *Matlab* qui reçoit en entrée les valeurs d'initialisation de la station de base telles que le facteur hystérésis nécessaire au calibrage du canal de transfert, le seuil SNR admis par le canal et utilisé lors de l'estimation des erreurs de démodulation permettant une modulation adaptive, la fréquence

de transmission ainsi que le nombre de sous-canaux et de symboles OFDM à transmettre afin déterminer le débit de transfert. Bien que le modèle soit réalisé en liaison ascendante UL, nous l'avons réalisé de telle sorte qu'il soit possible de l'utiliser en liaison descendante DL en changeant le mode de communication. Les paramètres évalués sont accessibles à tous les blocs de la simulation grâce à la communication par mémoire partagée.

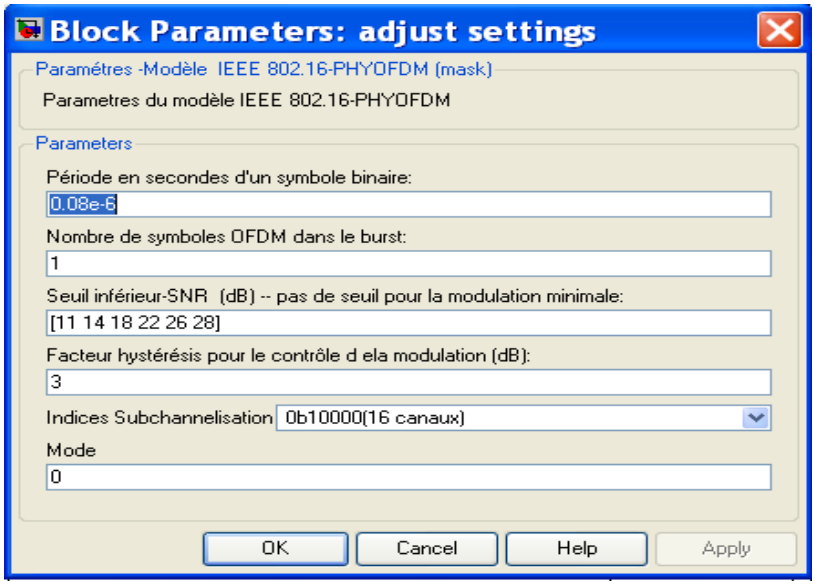


Figure 3.3 Bloc des parametres

- Source à débit variable

  Ce bloc est constitué d'un générateur d'éléments binaires aléatoires dont le nombre est déterminé par le bloc Paramètres. Les données sont générées de façon continue tout au long de la simulation.

- Évaluation des erreurs binaires par paquets. Il s'agit d'un bloc de la librairie de traitement de signal de *Simulink* permettant d'évaluer le taux d'erreurs binaire de la différence entre les données envoyées et celles reçues.

- Éstimation des erreurs de constellation

Ce bloc permet d'adapter la modulation au canal de propagation. Cela entraîne à la fois un changement du débit de transfert et donc une modification du codage du canal. Il reçoit en entrée l'estimation du canal caractérisée par la valeur RMS (variance ou root mean square en anglais), de la différence entre le signal modulé reçu et le signal démodulé puis modulé à la réception. Cela permet ainsi d'estimer l'influence du canal dans la dégradation du signal reçu.

3.2.2 Codage et décodage des données

Cette partie représente l'étape entièrement numérique de la transmission. Il s'agit du modèle en bande de base proprement dit. Comme le système est conçu à travers une modulation adaptive (lorsque le rapport signal à bruit SNR est inférieur ou supérieur à un certain seuil[3], la modulation est augmentée/diminuée afin d'agir sur la puissance du signal), la chaîne de codage est différente selon le taux de transfert utilisé, la modulation évaluée ainsi que le nombre de sous-canaux utilisés.
Ainsi par exemple dans le cas d'une modulation BPSK-1/2, le Reed-Solomon n'est pas employé. La modulation étant adaptée dynamiquement en fonction du rapport signal à bruit (SNR) évalué à partir du canal de communication, il est donc nécessaire d'avoir des blocs de modulation pour chaque type. Cela nécessite l'emploi d'un multiplexeur afin de choisir la chaîne de codage approprié aux paramètres de la communication.
Les figures 3.4 et 3.5 représentent les blocs codeurs/décodeurs correcteurs d'erreur et les différentes modulations/démodulation supportées. Chacun de ces blocs comporte les sous-blocs correspondant aux éléments FEC de la chaîne de codage ainsi qu'à l'assignation des éléments en fonction des constellations correspondantes.

[3]voir Annexe I, section I.9

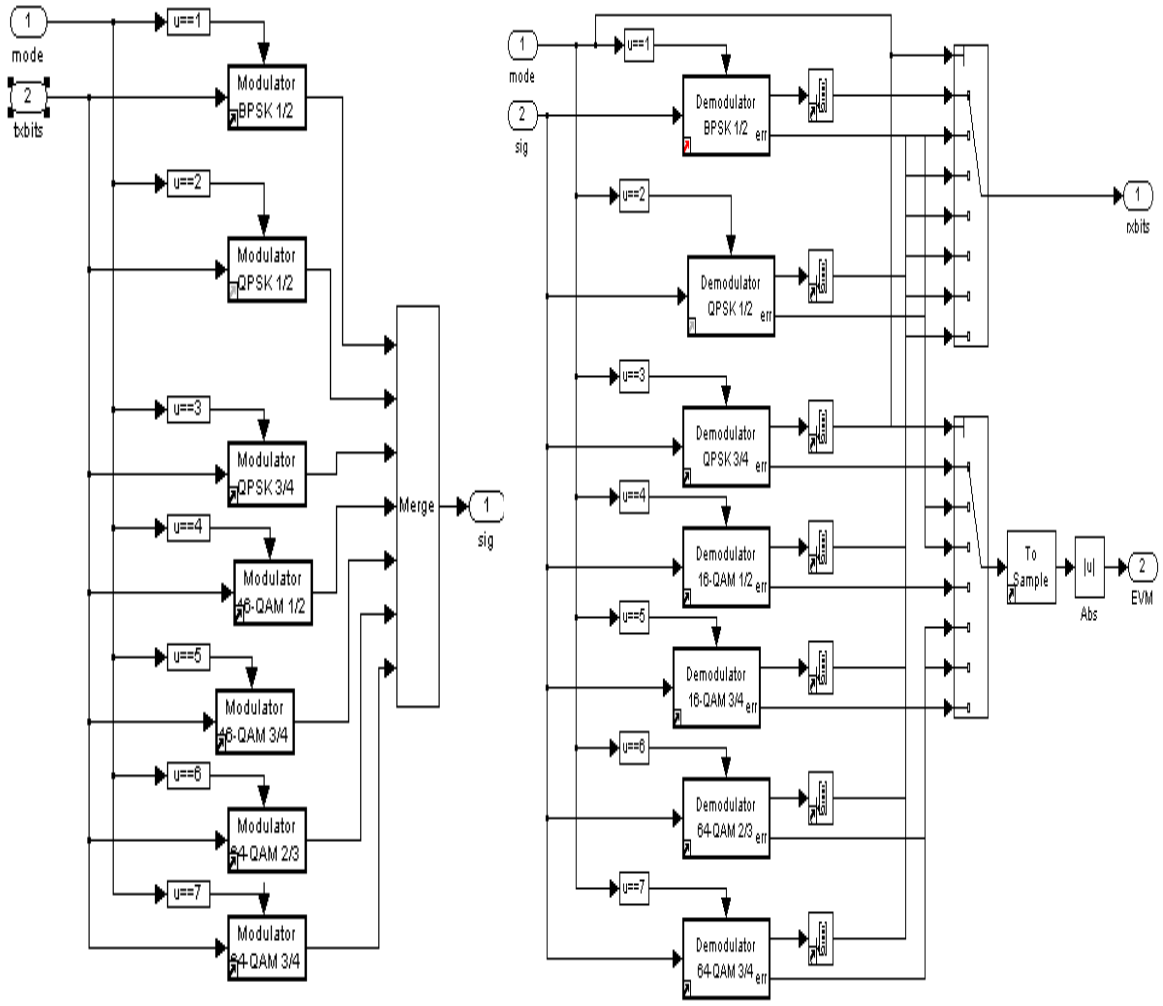


Figure 3.4 Bloc des modulations

3.2.3 Modulation et transmission des données

Comme notre modèle a été réalisé en bande de base, nous n'avons pas représenté la modulation radio fréquence (RF) nécessitant l'utilisation de filtres, de convertisseurs analogique/numérique, de paramétrer les antennes de transmission et de réception, d'effectuer une synchronisation temporelle et fréquentielle des signaux reçus. Dans notre cas nous avons représenté la modulation, la transmission à travers le canal de propagation (air) et la démodulation.

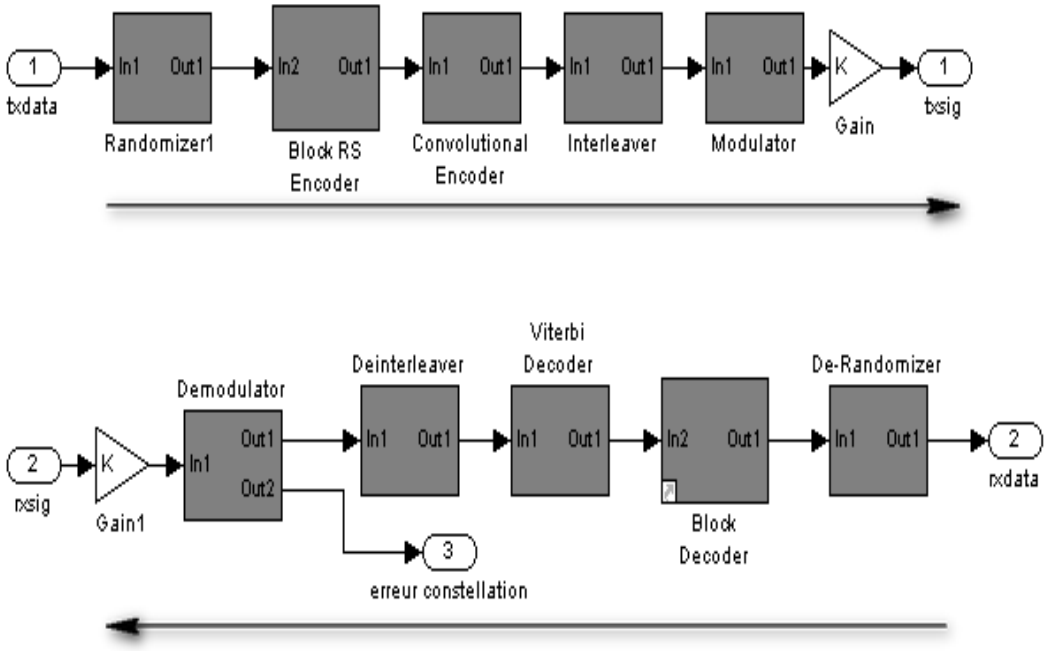


Figure 3.5 Bloc FEC+Assignation/De-FEC+DeAssignation

- La modulation a été réalisée à l'aide d'une transformée inverse de fourier IFFT. Pour cela nous avons utilisé un démultiplexeur car les données sérielles sont modulées en parallèle et envoyées en même temps. Ainsi la trame de bits codés est démultiplexée en fonction des 256 porteuses fréquentielles d'où l'utilisation d'une transformée de fourier inverse à 256 points. La démodulation est effectuée de façon inverse en utilisant cette fois un multiplexage fréquentiel des 256 porteuses par l'utilisation d'une transformée de fourier FFT à 256 points.

- Le canal de propagation représente trois types de propagation : à travers un canal bruité (AWGN), à travers un canal atténuant et/ou dispersif pour modéliser la présence d'obstacles ou la longueur de la propagation. Ces canaux sont supposés traduire le monde ambiant qui se trouve être bruité et rempli d'obstacles. Les librairies de *Simulink*, nous fournissent des modèles trés précis de canaux AWGN et de Rayleigh dont les paramètres sont obtenus à partir d'une estimation du rapport signal à bruit souhaité.

3.2.4 Réglages dans Simulink

Ce niveau servant à valider de façon globale l'algorithme de l'application et permettant d'estimer les performances du point de vue de l'utilisation, de nombreux tests ont été effectués. La première difficulté rencontrée a été au niveau du choix des blocs du modèle. La bibliothèque de *Simulink* étant très fournie, il existe différents blocs pour une même fonctionalité. Nous nous sommes inspirés de modèles disponibles pour le **Wifi** sur le forum MathWorks. Cette application ayant à peu près les mêmes fonctionnalités que le **Wimax**, nous avions juste à changer les paramètres et à adapter les blocs.
Au cours de la simulation, nous avons appliqué un solveur discret à l'ensemble du système c'est-à-dire que l'horloge interne évolue à des temps discrets; la période ou fréquence de l'horloge étant obtenue à partir de la bande fréquentielle utilisée puis calculée selon les paramètres définis au Chapitre 2. Ainsi l'horloge de simulation est égale à la fréquence d'échantillonage du système.

Néanmoins, ces résultats ne nous renseignent pas sur les performances du modèle en terme de vitesse de simulation, de temps de calculs des différents blocs, de temps de transfert entre ces blocs ou encore de surface occupée sur le matériel. Pour cela, il est nécessaire de descendre à un niveau système.

3.3 Modèle sous Space Codesign

Il s'agit du niveau système du flot de conception présenté à la figure 3.1 réalisé en langage SystemC à l'aide de la plateforme ESL Space Codesign. Le modèle a été réalisé aux niveaux Elix et Simtek correspondant aux niveaux 1 et 2 de la modélisation transactionnelle TLM selon l'approche ESL. Le niveaux Elix ne faisant

pas intervenir la notion du temps, il peut correspondre au niveau algorithmique réalisé dans *Simulink*. Le niveau Simtek fait intervenir la notion de temps et ainsi introduit les processeurs et bus associés, les mémoires, les timers, les interrupteurs etc. Dans cette section nous présentons la réalisation au niveau Simtek à l'aide du processeur Microblaze et de son bus OPB.

3.3.1 Méthodologie de Conception

Space Codesign se charge de la génération des adaptateurs et automatise le passage d'une architecture à l'autre sans aucune modification du code. De plus, les outils de profilage de la plateforme permettent d'obtenir une analyse des performances au point de vue logiciel et matériel. Ainsi, il s'agit de réaliser les modules de l'application que l'on souhaite étudier à l'aide de ces outils.

3.3.1.1 Implémentation des modules

Une des difficultés majeures de la réalisation en SystemC au niveau de la plateforme se trouve dans la complexité des algorithmes à implémenter; les modules les plus complexes étant le décodeur Reed-Solomon, le Viterbi et l'entrelaceur/de-entrelaceur. Une façon de représenter ces modules aurait été l'utilisation de mémoires externes afin de stocker les différents tableaux inclus dans les algorithmes. Étant donné le nombre de modules compris dans l'application, nous n'avons pas décomposé les algorithmes et avons préféré conserver toutes les fonctionnalités d'un bloc algorithmique au sein du même module. Cela donne des volumes de calculs élevés par modules mais évite les accès mémoires nécessaires au stockage de ces tableaux.

Le modèle réalisé sous *Space Codesign* a la même structure que celui présenté à la figure 3.2 c'est à dire qu'un bloc représente un module. Il comprend les éléments

suivants:

- un module Send qui envoie les informations à encoder
- les différents blocs de codage/décodage ainsi que d'assignation/dé-assignation et formation des symboles OFDM à envoyer. Ces modules sont au nombre de 12 : Melangeur, Codeur Reed-Solomon, Convolueur, Perforateur[4], Entrelaceur, Assignation, Aiguilleur (insertion des pilotes), De-Assignation, De-Entrelaceur, Viterbi, Decodeur Reed-Solomon, De-Melangeur.
- un module Param de gestion des paramètres
- un module Receive qui reçoit les informations décodées

Un dernier, le module Matlab a été rajouté et correspond à l'interface de co-simulation avec *Simulink*.

3.3.1.2 Paramétrage des modules

L'émetteur Send représente la Sous-Station SS[5] et le recepteur Receive la station de base BS. Les paramètres sont ceux qui ont été récupérés lors de la période d'initialisation de la SS.

De par la nature de la plateforme ESL basée sur des nombres fixes, le canal de communication a été modélisé par le module matlab qui est l'interface de co-simulation avec *Simulink* car la modulation IFFT/FFT renvoie des nombres flottants dus au facteur multiplicatif de modulation: 1 pour BPSK, $1/\sqrt{2}$ pour QPSK, $1/\sqrt{10}$ pour 16-QAM, $1/\sqrt{42}$ pour 64-QAM. Le module Param communique avec tous les

[4]ce module est généralement combiné au Convolueur, nous avons toutefois préféré les séparer afin d'alléger le code

[5]voir liste des acronymes

autres modules puisqu'il contient les paramètres de modulation, de transfert, de découpage en sous-canaux, le numéro de la SS et de la trame. Les données transférées entre deux modules étant de type «unsigned long», la transmission des données se fait sous forme d'octets et non de bits afin de réduire les transactions sur les bus.

Le module Send initialise la communication : il envoie d'abord les paramètres reçus au module Param puis il envoie les données au module Mélangeur. Ensuite, le module Param transmet en une seule fois ces paramètres à tous les modules de la chaîne. Une fois les opérations terminées, le module Receive récupère les informations décodées. Le module Param a un rôle de distribution, il n'effectue aucune opération; on aurait pu le supprimer, toutefois cela permet de montrer que les paramètres sont communs à tous les modules alors que les données sont envoyées modules par modules. Le format des données transférées par le module Param est le suivant :

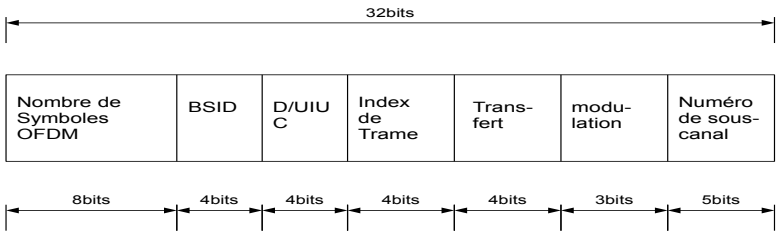


Figure 3.6 Format des données de paramètres

où:

- Nombre de Symboles OFDM est le nombre de symboles OFDM qui seront modulés,
- BSID est le numéro de la station de base BS,
- D/UIUC est l'interval d'usage du code,

- Index de trame est l'index de la trame,
- taux de transfert est le taux associé à la modulation (soit 1/2, 2/3 ou 3/4). Bien que nécessitant 2 bits, il est représenté sur 4 bits afin de pouvoir l'étendre à d'autres taux de transfert si nécéssaire (5/6 ou 7/8 par exemple).
- modulation est le type de modulation utilisé (soit BPSK, QPSK, 16QAM ou 64QAM). Elle est représentée sur 3 bits au cas où il serait nécessaire d'étendre les modulations pour des modelisations futures.
- numéro du sous-canal est le numéro du sous-canal utilisé (allant de 1 à 31).

Tel que spécifié par la norme, le codeur/décodeur RS n'intervient que lorsque l'on utilise les 16 sous-canaux en même temps. En se référant au tableau I.2 du chapitre 2, on trouve le nombre de sous-canaux correspondant au numéro de sous-canal utilisé soit 16 pour 0b10000, 2 pour 0b00010 et 1 pour 0b00001. Les paramètres UIUC, BSID et Numéro de Trame sont utilisés au niveau du mélangeur tandis que les autres sont employés par tous les modules de la chaîne.

La partie décodage n'étant pas définie dans la norme, les modules de décodage ont été vérifiés en comparant les sorties aux entrées des encodeurs correspondants : Melangeur<>De-Melangeur, Codeur RS<>Decodeur RS, Convolueur+Perforateur<>Viterbi, Entrelaceur<>De-Entrelaceur, Modulateur<>De-Modulateur.

3.3.1.3 Architectures du modèle

La figure 3.7 représente le niveau Elix ou une architecture logicielle/matérielle au niveau Simtek (dans ce cas on rajoute le processeur Microblaze et les périphériques associés). Dans ces architectures, toutes les transactions passent par le bus. Le passage d'une architecture à l'autre se fait dans *Space Codesign* en définissant le type

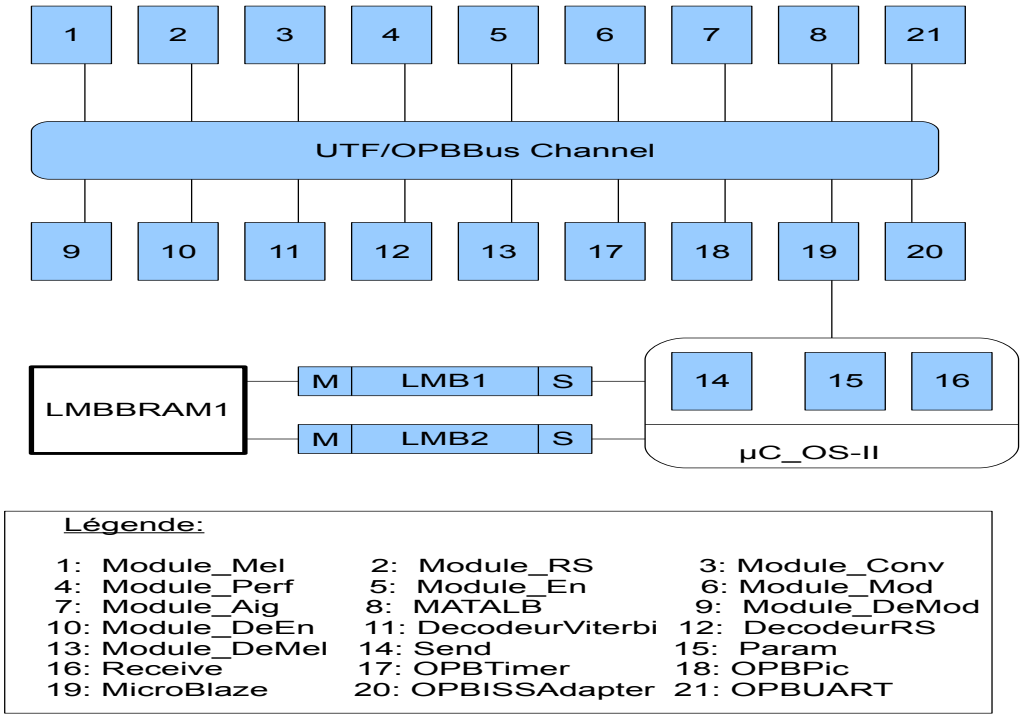


Figure 3.7 Architecture synthétisée par ForteCynthesizer

de bus (UTF pour Elix, OPBBus pour Simtek).

Une architecture logicielle/matérielle a également été réalisée au niveau Simtek. Des simulations ont été effectuées avec différentes combinaisons de modules dans le processeur Microblaze utilisé. Le modèle a été testé avec les 3 bancs d'essai fournis par la norme pour différentes configurations; les paramètres de modulation, les données à encoder et les différentes sorties des modules d'encodage sont fournis par le banc d'essai tel que le produirait la Station de Base dans une communication complète. Les résultats de ces simulations ainsi que les profilages associés sont présentés au chapitre 4.

De par la nature de l'application qui est du type flot de données, une architecture point-à-point en passant par des liens directs a été implémentée au niveau Simtek entre

les différents modules. Dans un premier temps, des communications non bloquantes ont été utilisées afin d'accélérer le débit de transfert. En effet dans une communication non bloquante, une fois qu'un module transmet un message, il peut continuer ses calculs sans attendre que l'autre module ait lu le message. Dans une communication bloquante, le module receveur doit lire le message avant que le module initiateur ne puisse continuer ses opérations. Cela entraîne des délais d'attente de part et d'autre des modules qui changent le débit de transfert des données.

Toutefois, à cause des débits de transfert très élevés, les communications bloquantes ont été privilégiées car les files d'attente se remplissent très vite dans des communications non bloquantes pour des applications à débit élevé. L'utilisation de communications bloquantes influe sur le temps de calcul des données, mais est cependant minime par rapport à l'acquis en terme de temps global du système.

Finalement grâce aux outils de profilage, des métriques et des statisiques d'estimation de l'architecture ont été extraites et permettent ainsi de définir un partitionnement optimal de l'application. Néanmoins afin de valider à haut niveau l'application mais également d'intégrer la partie modulation et transmission au modèle numérique (comprenant uniquement la chaîne de codage/décodage) il est nécessair de faire une co-simulation à haut niveau entre *Space Codesign* et *Simulink*.

3.4 Modèle de Co-simulation Space/Simulink

Tel que présenté au chapitre 2, la co-simulation d'un module *SystemC* avec *Simulink* nécessite l'utilisation de la librairie *Matlab Engine* et de S-Functions. La librairie *Matlab Engine* étant pré-compilée pour la plateforme Windows et Unix avec le compilateur de *Visual Studio* et *Space Codesign* étant compilé sous gcc dans Windows; l'utilisation de *Matlab Engine* a nécessité une interface entre le GUI *SpaceStudio* et *Visual Studio*.

Tel qu'illustré à la figure 3.8, l'interface *Space-Matlab* est subdivisée en deux parties:

SpaceStudio vers *Visual Studio* et *Visual Studio* vers *Matlab-Simulink*. La méthode de cosimulation *Visual Studio* <> *Matlab-simulink* a déjà été présentée en [**?**], nous l'avons adaptée à la plateforme *Space Codesign* en lui rajoutant des interfaces basées sur une communication par socket.

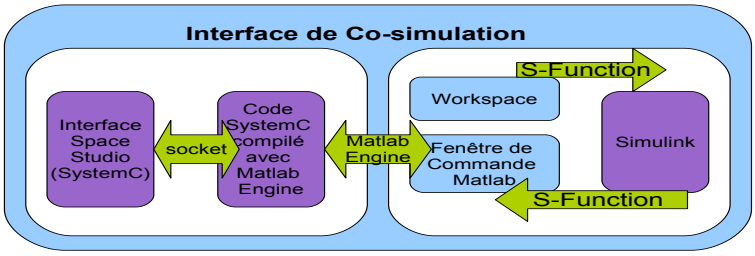


Figure 3.8 Interface de co-simulation SpaceStudio-Simulink

3.4.1 Interface entre Space Codesign et Visual Studio

L'interface *Space* vers Visual a été réalisée à partir d'une communication par socket. On inclut le fichier ServerClient.h puis à l'instantiation du module Matlab de *Space Codesign* on ouvre un port. On envoie/reçoit des données venant de visual par le port SERVER_NAME, section IV.3).

Pour éviter que les temps de connection n'interviennent dans le temps de simulation, on lance d'abord *Matlab engine* avant l'initialisation du simulateur SpaceStudio et on ouvre les sockets à l'instanciation des modules *SystemC*. Les données reçues/envoyées ont une taille de 32 bits comme celles d'une communication dans *Space Codesign*. Étant donné le débit de transfert des données d'un octet, les 24 bits restants sont nuls sauf dans le cas du transfert des paramètres. Cette interface communique avec les modules Send, Receive, Aiguilleur et DéModulateur, elle correspond à un module de *Space Codesign*.

Sa fonction send_param_to_mel() reçoit les paramètres initialisés dans *Simulink* et

l'envoie au module Send puis la fonction send_data_to_mel() reçoit les données à transmettre et les envoie également au module Send. Dans ce cas si le module Send aurait pu être supprimé, nous avons préféré le conserver dans le cas ou l'on étendrait le modèle à plusieurs SS nécessitant donc des chaînes de codage distinctes.
Une fois les opérations de codage effectués, la fonction send_ifft_to_mat() récupère les données assignées et les envoie à *Simulink* via l'interface. Ces données sont modulées (IFFT), transmises (canal AWGN, Rayleigh) puis dé-modulées (FFT) dans *Simulink* et renvoyées dans *Space Studio*. Les données dé-modulées sont ensuite envoyées pour décodage à l'aide de la fonction receiv_fft_to_mat(). Les informations décodées sont par la suite renvoyées par la fonction receiv_data_from_mel() vers *Simulink* pour être évaluées (calcul du taux d'erreur par paquets).
La communication dans *Space Codesign* étant non-bloquante, l'interface utilise une communication bloquante (handshake) afin de maintenir la synchronisation avec *Matlab*, cela ne change rien aux performances générales qui permettent de valider le modèle mais influence le temps de simulation au niveau système.

3.4.2 Interface entre Visual Studio et Matlab

La deuxième interface est celle qui communique avec *Matlab*. Il s'agit d'un module *SystemC*. On inclut aussi le fichier ServerClient.h puis on définit un port de communication par socket. Tel qu'expliqué auparavant, on lance *Matlab Engine* avant d'ouvrir la socket, cela permet à *Simulink* de s'initialiser avant le début de la communication. La communication est initiée par le module Visual qui est le Client de *Matlab*. Cela est dû au fait que *Matlab Engine* est très lent à démarrer. Cette interface comporte des fonctions réciproques des fonctions définies dans l'interface de *Space Studio*. Ainsi envoi_param() et envoi_data() transmettent les paramètres et les données à coder *Space Studio*, put_ifft() et get_fft() reçoivent les informations à moduler et renvoient les informations démodulées après passage dans le canal; enfin

receiv_data() récupère les données finales à évaluer. Chacune de ces fonctions fait appel aux variables et fonctions de *Matlab Engine* permettant de placer et récupérer des valeurs dans l'espace de travail de *Matlab* (Workspace en anglais).
Toutefois bien que les données soient générées dans *Simulink*, la communication est contrôlée par l'interface Space car c'est elle qui demande qu'on lui envoie des informations et cela à tout moment. A ce niveau, la communication entre le modèle *SystemC* et le Workspace de *Matlab* se fait par l'utilisation des routines de *Matlab Engine* engEvalString, EngGetVariable, EngPutVariable. Ces routines font ensuite appel aux fonctions *Matlab* pour l'évaluation des variables *Matlab* dans un environnement *SystemC*. Ainsi par exemple, pour commander au modèle *Simulink* de lancer la simulation on emploie la formulation suivante :

```
engEvalString(ep,"set_param('Cosim','SimulationCommand','start')");
```

Dans ce cas-ci, EngEvalString est la commande de *Matlab Engine* qui fait appel à la commande *Matlab* set_param et aux paramètres de simulation (start, stop, continue) d'un modèle *Simulink*.

3.4.3 Interface entre Matlab et Simulink

Il ne s'agit pas à proprement parler d'une interface mais plutôt de plusieurs S-Functions disponbiles dans les librairies *Simulink* et permettant d'intégrer un code C++ dans un modèle *Simulink*. Nous avons défini 4 S-Functions correspondant à l'envoi des données et des paramètres à coder, la réception des données à moduler, l'envoi des données démodulées puis la réception des données décodées. La taille et le nombre de ports dépend de la nature des informations. Les données sont soit réelles (source) ou complexes(FFT).

L'échange de données entre le module *SystemC* de visual et *Matlab* se fait par mémoire partagée. On utilise le Workspace de *Matlab* pour placer et récupérer des informations: on Alloue un espace mémoire aux données à placer. Les fonctions mxCreateNumericArray et mxCreateDoubleMatrix permettent de créer un pointeur vers un tableau *Matlab*. Les données venant de *Simulink* sont envoyées/récupérées une à une par paquets de 8 bits (octet). Puisque la source génère des éléments binaires, les données envoyées vers Space ont une période 8 fois plus longue que celle de *Simulink*. On récupère les paramètres de la simulation (numéro du sous canal, bsid, duiuc, bsid, mode (modulation+transfert)) à partir du fichier « params.m » initialisé par le bloc *Simulink* «Params». Les éléments sont récupérés par leur nom. Puis, on contrôle la simulation par les fonctions set_param et get_param de la librairie *Matlab*.

La figure (voir figure 3.9) présente les modifications apportées au modèles *Simulink* de la figure 3.2 pour y intégrer les interfaces de communications. Ainsi, on remplace toute la partie codage/décodage du canal par les blocs de Space (interfaces en Vert sur la figure 3.9) et on effectue la simulation.

3.5 Modèle RTL

C'est le dernier modèle du flot de conception présenté à la figure 3.1. Les modèles haut niveau, bien que définissant les métriques de l'application, ne fournissent pas les détails d'implémentation. Avant la réalisation du modèle de transmission sur puce, plusieurs simulations sont effectuées à haut niveau et à bas niveau pour des fins de validation.

Les cartes sont constituées de plusieurs composantes: un processeur contenant la partie logicielle (application, OS,etc.) du modèle et des blocs à même la carte contenant les éléments matériels.

L'implémentation sur carte FPGA permet d'obtenir des métriques beaucoup plus

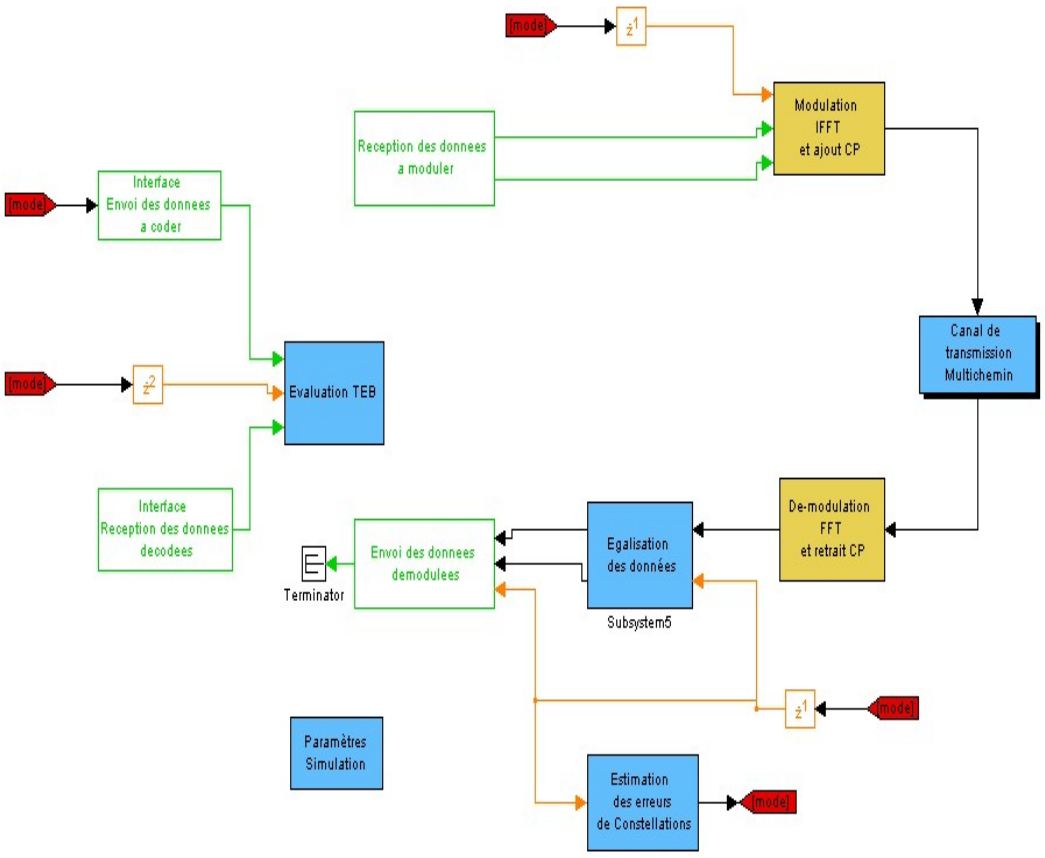


Figure 3.9 Modèle de co-simulation SpaceCodesign-Simulink

précises qu'à haut niveau. En effet, on tient compte du fait que les bits représentent des signaux logiques, qu'il y a une horloge du système et que tous les éléments (registres, entitées) sont synchronisés sur cette horloge. Toutefois cela entraîne des temps de simulation beaucoup trop élevés nécessitant d'avoir une architecture validée au préalable.

3.5.1 Du modèle systemC au modèle RTL

On doit effectuer auparavant une simulation bas niveau (RTL) c'est à dire au niveau portes logiques. Les deux langages principaux pour ce niveau sont le VHDL et le Verilog et il existe certains outils de synthèse comportementale permettant de passer d'un niveau système au niveau RTL. Deux méthodes ont donc été utilisées pour passer du modèle conçu en *SystemC* dans Simtek au modèle RTL.
La première a été réalisée à l'aide de l'outil Forte Cynthesizer[**?**]. Pour cela, les boucles logicielles du code Simtek ont été déroulées et les types des variables ont été modifiés pour être supportés par l'outil.
La seconde méthode consiste en l'utilisation des IP[6] cores (composantes matérielles) disponibles sous *Xilinx* ISE ou de modules VHDL codés manuellement. Toutefois afin de pouvoir communiquer avec l'architecture globale, des adaptateurs (wrappers en anglais)[7] VHDL ont été créés pour ces IPs. Certains blocs ont été remplacés par ceux synthétisés avec Forte et seuls ceux de *Xilinx* ont été comparés à leurs équivalents Forte. Le module Mélangeur a été fait à la main car son IP n'existe pas dans la librairie *Xilinx*.
L'exploration architecturale du modèle (voir chapitre 4) a montré qu'un modèle entièrement matériel était plus adapté à ce type d'application qui est du type flot de données. Cela permet en effet d'accélérer la communication. L'application contenant 16 modules, nous avons jugé préférable de ne pas les diviser même si cela offait des possibilités de paralélisme. Pour ce modèle nous avons laissé de côté la modulation par transformée de fourier. Le module Matlab représentant l'interface de cosimulation n'a pas été pas synthétisé puisque la cosimulation entre le niveau RTL et le niveau algorithmique n'entrait pas dans nos objectifs.

Comme on le voit dans la figure 3.7, il y a trois bus pour cette architecture:

[6] intellectual property
[7] sortes d'encapsulateurs permettant d'interfacer deux éléments différents

- OPBBUS, le bus principal du processeur MicroBlaze de *Xilinx* par lequel communiquent les modules
- LMB1, le bus de données de la mémoire externe.
- LMB2, le bus d'instruction de la mémoire externe.

Ces éléments servent à la communication des différents blocs du système.

3.5.2 Génération des éléments de synthèse FPGA

L'outil GenX (voir chapitre 1 section 1.2.4) de *Space Studio* a généré tous les éléments disponibles de la carte FPGA. Il a généré les fichiers principaux d'un projet de *Xilinx* Platform Studio (*XPS*) en fonction d'une configuration d'un système *Space Codesign* au niveau Simtek. Ensuite, *XPS* a pu être exécuté.Les fichiers générés ont été simulés à bas niveau sous Modelsim puis synthétisés et programmés dans le FPGA Virtex-II disponible. GenX étant en cours de création, la taille des cartes disponibles ne permettait pas de contenir une application aussi importante que celle-ci, les différents modules ont été générés manuellement (c'est à dire un par un).
Ainsi, pour passer de *Space Studio* à *Xilinx EDK*, nous avons lancé le Wizard GenX après avoir configuré les éléments de l'architecture puis nous avons choisi la carte FPGA utilisée pour la synthèse. Les paramètres suivants ont été choisis:

- la taille de la mémoire, elle est utilisée par la partie logicielle
- le débit et les modes de l'UART (pour la communication sérielle) utilisée pour le déboggage de lors de l'exécution.

Les éléments matériels (Vérilog, VHDL ou IPcores de *Xilinx*) ont été rajoutés à ce projet pour la simulation puis pour la synthèse.

3.6 Conclusion

Dans ce chapitre, nous avons présenté notre implémetentation de l'application **Wimax** à trois niveaux de conception à l'aide des différentes plateformes présentées au chapitre 1. L'implémentation au niveau algorithmique à l'aide de *Simulink* a nécessité de nombreuses simulations étant donné que le modèle correspondant sert de référence. Le second modèle au niveau système dans *Space Codesign* a consisté en l'implémentation au niveau système des algorithmes. Le dernier modèle RTL a pu être implementé sur FPGA. Finalement, l'interface de cosimulation permettant de valider le modèle système avec le modèle de référence algorithmique a été réalisée. Elle permet de co-simuler un code *SystemC* avec un modèle sous *Simulink* ce qui dans notre cas permet de tester conjointement les modèles numériques et analogiques. Le chapitre suivant présente les résultats de simulation des différents modèles et analyse les performances obtenues.

CHAPITRE 4

RÉSULTATS ET DISCUSSION

Ce chapitre présente les résultats de simulation et de synthèse des trois modèles réalisés pour un émetteur et un récepteur (point-à-point P2P) et pour une communication Uplink (ascendante). Il présente également les résulats de l'utilisation de la cosimulation afin de valider les modèles à haut niveau tel que présenté au chapitre précédent.

4.1 Performances au niveau algorithmique

Dans cette partie, nous analysons les performances générales d'une application de réseaux sans fil. Ce modèle étant utilisé comme modèle de référence, toutes les configurations possibles ont été testées pour différents sous-canaux. Afin de respecter les requis de la norme IEEE 802.16 concernant le niveau de puissance lors de la transmission et lors de l'arrivée, de nombreux essais ont été effectués selon différentes modulations et pour différents chemins (canal bruité avec/sans attenuation et/ou dispersif). Le modèle étant réalisé en bande de base, nous avons retenu deux critères d'évaluation des performances: le rapport signal à bruit (Eb/No) et le taux d'erreurs binaires (BER).

La simulation a été effectuée sous *Simulink* pour un canal de bande 22MHz en transmission UL et une trame (préambules + symboles) de durée 2.5ms[1]. Le ratio G correspond à la proportion du symbole qui est copiée au début et est fixé à 1/4 (voir annexe I, section I.8).

[1]les durées de trames sont fixées au tableau II.3 de l'annexe 1

En se référant à la section I.8 de l'annexe I concernant les paramètres OFDM nécessaires à la BS (station de base) pour l'initialisation de la communication, on obtient une fréquence du signal Fs de 25.08MHz et un temps d'échantillonnage T_ech de 39.87 ns. La durée d'un symbole OFDM T_b équivaut à 1.02×10^{-5}s. Sachant qu'une trame de durées 2.5 ms peut envoyer X+2 symboles OFDM pour 16 canaux ou X+1 symboles pour moins de 16 canaux de durée T_b chacun, cela donne 220 symboles envoyés par seconde soit 218 ou 219 symboles de données.

La figure 4.1 présente le taux d'erreurs binaires TEB (BER en anglais) pour un canal bruité AWGN. La figure 4.2 montre le taux d'erreurs binaire pour des canaux de Rayleigh attenuants et dispersifs. Les courbes du TEB sont représentées en fonction du rapport signal à bruit pour différents sous canaux selon une modulation adaptative. La modulation adaptative est calculée en fonction du rapport signal à bruit.

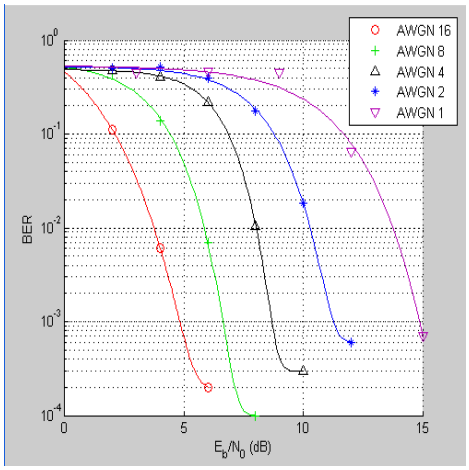


Figure 4.1 BER pour un canal AWGN avec une modulation adaptive pour 16, 8, 4, 2 et canaux

Etant donné les différents chemins parcourus (à travers un canal bruité ou attenuant ou dispertif), on constate que plus on diminue le nombre de sous canaux utilisés, plus le rapport signal à bruit est élevé. Cela se retrouve dans la littérature, car le fait d'utiliser moins de sous canaux entraîne une moins bonne tolérance aux bruits,

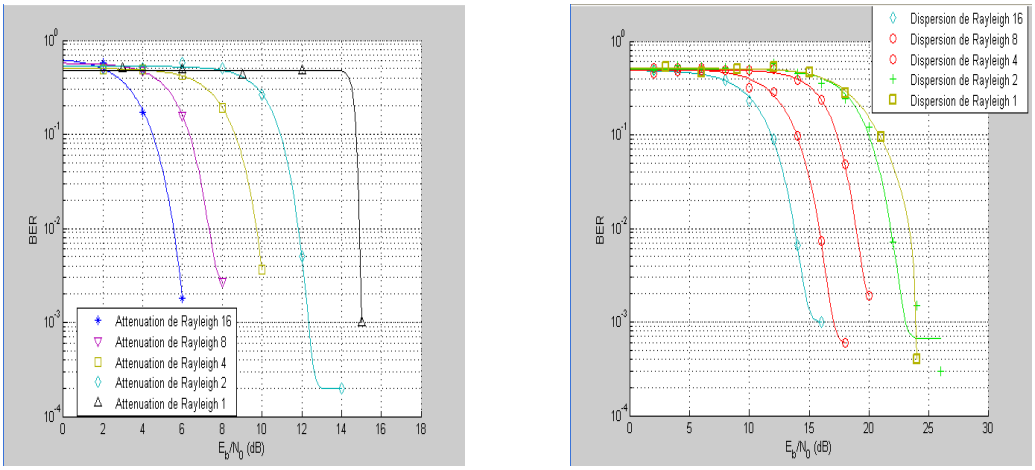


Figure 4.2 BER pour canal Rayleigh attenuant et un canal Rayleigh dispersif avec une modulation adaptive pour 16, 8, 4, 2 et canaux

à l'attenuation du signal et à la dispersion, dû au fait qu'il y a moins de pilotes et de porteuses utilisés.

D'autre part, on peut observer à la figure 4.3 que le rapport signal à bruit évolue en fonction de la modulation; plus le rapport est faible, plus forte est la modulation (la modulation la plus forte étant BPSK et 64QAM-3/4 la moins élevée). Cela s'explique par le fait que l'attenuation du signal et le nombre d'erreurs de constellations sont accentués par la réduction de la bande passante. On observe au niveau de ces figures les états de la constellation à la réception avant et après égalisation. Le signal égalisé apparaît mieux que le signal non égalisé; de plus le rapport signal à bruit a la même évolution que le taux de transfert. Cela est dû à la modulation adaptive qui modifie le taux de transfert en fonction du rapport signal à bruit évalué. Le taux d'erreurs par paquet correspond au cumul des taux d'erreurs binaires au cours du temps. On constate qu'il est très faible ce qui correspond à la situation souhaitée au niveau de la

norme. On observe de gauche à droite et de haut en bas, l'évolution de la simulation. Les captures d'écran ont été prises pour les modulations respectives BPSK, QPSK, 16-QAM, 64-QAM.

En observant l'évolution des constellation et les courbes du TEB selon les canaux

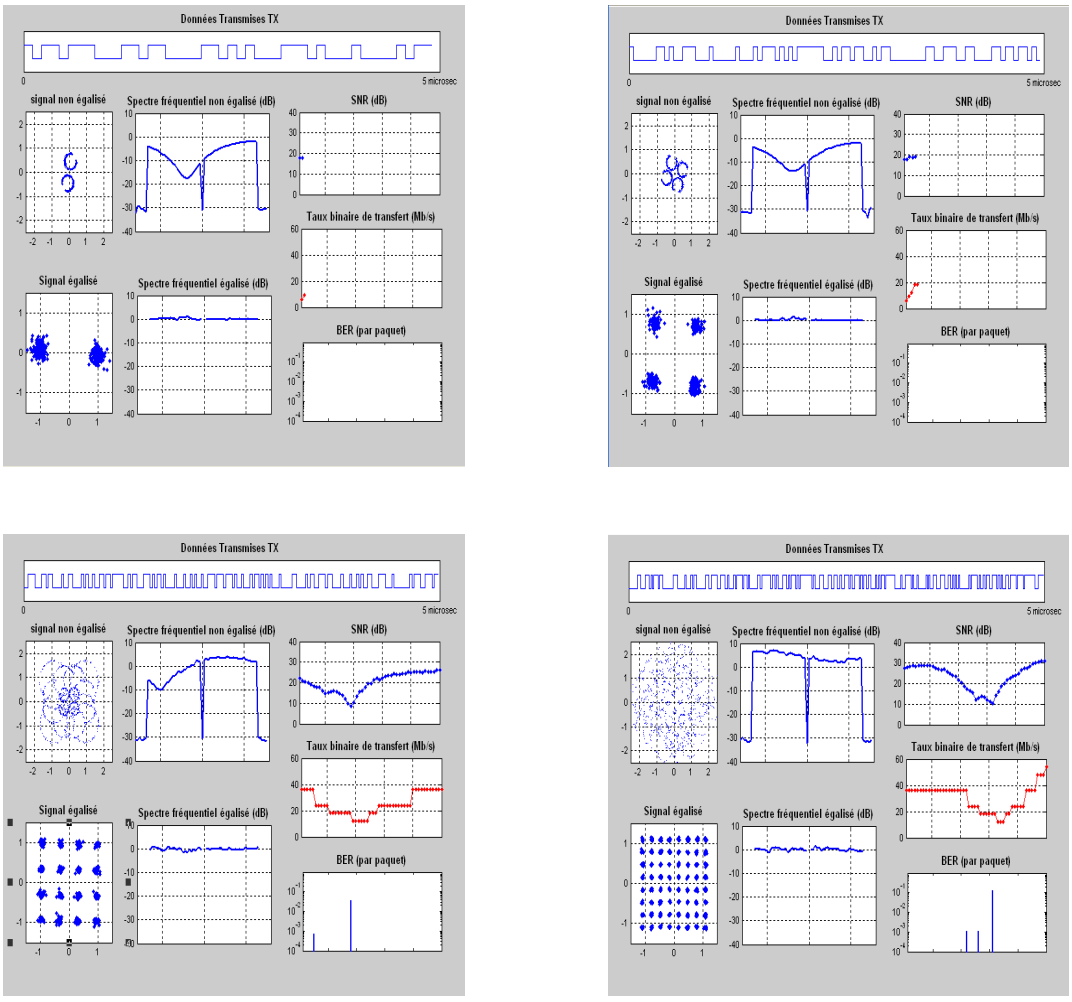


Figure 4.3 Visualisation des signaux reçus pour une modulation adaptative selon BPSK, QPSK, 16-QAM, 64-QAM pour un canal bruité, dispersif et attenuant

parcourus, on constate que le TEB diminue lorsque l'on augmente le débit du canal c'est-à-dire la modulation du signal à transmettre. En effet, afin que le rapport

signal à bruit (Eb/No) à la réception ne se dégrade pas de plus de 0.5dB, la norme prévoit un seuil de tolérance du Eb/No selon les modulations. Pour les modulations BPSK , QPSK-1/2, QPSK-3/4, 16QAM-1/2, 16QAM-3/4, 64QAM-2/3 et 64QAM-3/4 les seuils correspondants sont respectivement 13dB, 16dB, 18.5dB, 21.5dB, 25dB, 28dB et 31dB. En observant les figures on constate que le TEB n'évolue plus au-delà d'un certain seuil correspondant à ceux requis. C'est pour cela qu'une modulation adaptative est généralement appliquée afin de réguler la modulation et de ce fait ce rapport signal à bruit.

Ces résultats concordent à ceux spécifiés dans la norme. Le modèle en bande de base est ainsi validé en remplissant les exigeances de la norme. Toutefois, ce modèle peut être étendu par l'ajout de la partie radio-fréquence (RF) de la transmission en intégrant des antennes d'émission et de reception ainsi qu'un bloc de synchronisation en entrée du récepteur pour une propagation réelle à travers l'air.

4.2 Processus de co-simulation

La co-simulation a été réalisée afin de valider à haut niveau le modèle système de l'application. Afin de valider notre modèle au niveau système, nous lui avons appliqué les mêmes tests qu'au modèle de base *Simulink* prenant ainsi en compte tous les cas possibles. On observe les mêmes valeurs des données envoyées respectivement depuis *Simulink* vers l'espace de travail *Matlab* (183, 27, 13) puis vers *Visual Studio* (B7, 1B, D) et enfin vers *SpaceStudio* (B7, 1B, D). Dans *Visual Studio* et *SpaceStudio*, les valeurs sont représentées en hexadécimal alors qu'elles sont en décimal dans *Matlab*.

La figure 4.4 nous montre une co-simulation entre les différentes plateformes.

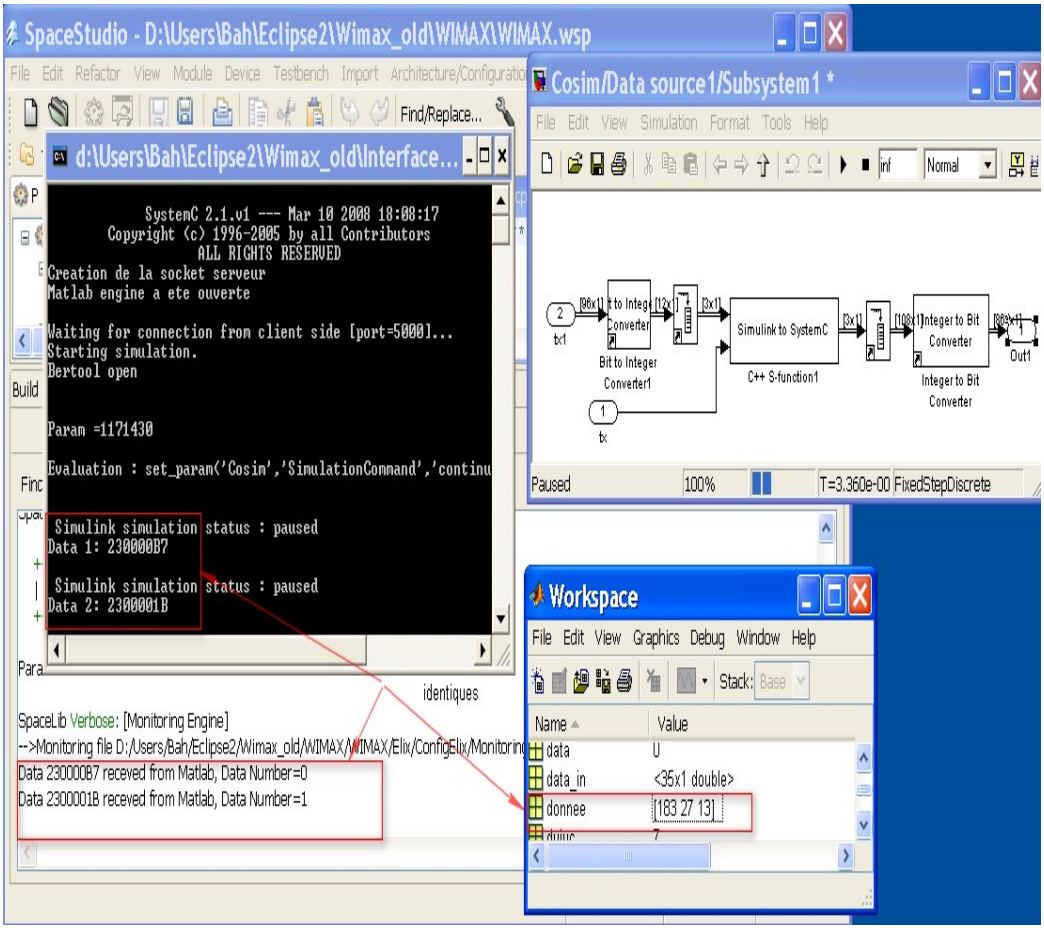


Figure 4.4 Vue d'une co-simulation entre Space Studio-Visual Studio-Matlab-Simulink

4.3 Exploration architecturale et synthèse

Dans cette section nous présentons à la fois les résultats de l'exploration architecturale et ceux de la synthèse matérielle. Nous avons choisi de les présenter ensemble car ils sont complémentaires. Le niveau Elix n'offrant d'intérêt que pour une validation globale à haut niveau de l'application, les résultats obtenus ont été réalisés au niveau Simtek. Nous pouvons toutefois affirmer que la simulation Elix est très rapide étant donné le niveau TLM considéré (voir chapitre 1, section 1.2.3.2). D'autre part, la partie étudiée étant la chaîne de codage, seuls les blocs correspondant ont été

synthétisés au niveau RTL. En d'autres termes, les blocs Send, Param, Receive et Matlab n'ont pas été synthétisés au niveau RTL. La simulation a été effectuée pour différentes configurations du partitionnement logiciel/matériel mais également selon le type de modulation et le nombre de sous canaux utilisés. Les bancs d'essai utilisés sont présentés à l'annexe 4, section III.4.

4.3.1 Résultats de simulation selon le type de modulation et le nombre de sous-canaux

Comme nous l'avons présenté au niveau des résultats de performance au niveau algorithmique, les paramètres du Wimax (c'est-à-dire la modulation, le taux de transfert et le nombre de sous-canaux) influencent le débit de transfert des données. Nous évaluons ici, si ces paramètres ont une influence sur le partitionnement choisi.

Le tableau 4.1 présente les résultats de simulation pour une configuration entièrement matérielle au niveau Simtek; il s'agit du niveau BCA (au cycle près dans la modélisation transactionnelle TLM). Étant donné que le nombre d'octets envoyés dépend de la modulation et du nombre de sous-canaux, nous avons effectués les tests conformément aux valeurs définies dans la norme (Voir Annexe 1, section nombre d'octets obligatoires par modulation).
À des fins de cohérence, nous avons utilisé le même nombre de données par modulation, ce qui fait que par exemple : pour BPSK-1/2 et pour 1 sous-canal on envoie 16 symboles OFDM, pour BPSK-1/2 et 16 sous-canaux on envoie 1 symbole OFDM et ainsi de suite. Selon les sous-canaux on retrouve le même nombre de symboles envoyés. Par exemple, pour 16 sous-canaux utilisés, quelque soit la modulation, on envoie 1 seul symbole, pour 8 sous-canaux on envoie 2 symboles et ainsi de suite. Les bancs d'essai appliqués sont présentés dans l'annexe 4.

Tableau 4.1 Temps d'exécution en ms selon le nombre de sous-canaux et le type de modulation

Modulation	Nombre de Sous Canaux				
	1	2	4	8	16
BPSK-1/2	42.7	41.4	40.8	40.5	40.4
QPSK-1/2	42.7	41.5	40.9	40.6	40.5
QPSK-3/4	42.8	41.5	40.9	40.6	40.5
16QAM-1/2	42.9	41.7	41.1	40.8	40.7
16QAM-3/4	43.0	41.8	41.2	40.9	40.8
64QAM-2/3	43.2	42	41.3	41.0	41.0
64QAM-3/4	43.2	42.0	41.4	41.1	41.1

On remarque que plus le nombre de sous-canaux augmente, plus le temps de calcul diminue. Cela est dû au fait que le nombre de symboles envoyés diminue. Par ailleurs, les différences au niveau des temps de simulation sont très minimes : la plus faible est de 40.4 ms et la plus grande de l'ordre de 43.2 ms ce qui fait une différence d'environ 2.8 ms. On peut en conclure que la modulation et le nombre de sous-canaux n'influencent pas le temps de calcul. Par la suite, nos tests ont été effectués pour un seul des cas, c'est à dire pour 16 Canaux utilisés avec une modulation QPSK-1/2.

4.3.2 Profilage du modèle système

Le but de cette étape était de déterminer l'architecture la plus optimale en terme de temps de calcul et de communication d'une part puis d'occupation spatiale (nombre de slices[2] sur le FPGA) de l'autre. De plus, étant donné la nature de notre application qui est de type flot de données, elle nécessite des débits de transfert très élevés. L'architecture optimale en terme de temps serait une configuration entièrement matérielle alors que si l'on ne tient compte que de la surface, il s'agirait

[2]une slice est une unité logique d'un FPGA

de mettre le plus de blocs possibles en logicel. Nos tests ont ainsi visé à déterminer l'équilibre entre ces deux contraintes.

La plateforme dispose de différentes composantes architecturales au niveau TLM. Dans notre cas nous avons utilisé principalement le processeur Microblaze et son bus OPB. Toutefois, il existe des liens directes entre modules matériels permettant ainsi d'accélérer les transactions. De cette façon, le bus est allégé et on évite les arbitrages entre modules. Les figures 4.5 et 4.6 offrent une vue graphique et analyse statistique des résultats de profilage pour la configuration 3 présentée au tableau 4.3 avec les modules Send, Receive et Param en logiciel et les autres en matériel.

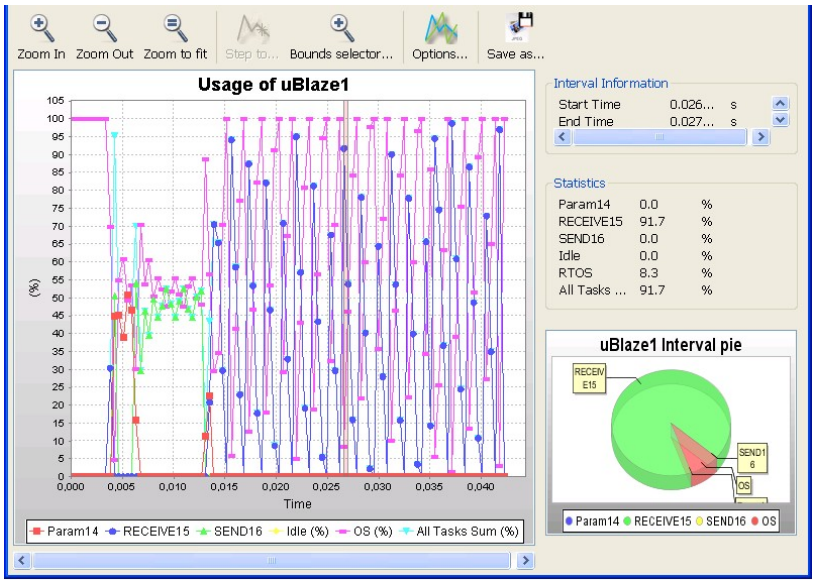


Figure 4.5 Utilisation du processeur

Dans un premier temps, nous avons cherché à déterminer le temps de calcul des différents blocs selon le partitionnement matériel ou logiciel. Le tableau 4.2 présente les temps d'exécution simulés que nous avons obtenus pour chacun des blocs. Étant donné que la modulation et le nombre de sous-canaux n'influence pas le temps de simulation, nous avons effectué nos tests pour le banc d'essai n°1 requis par la norme;

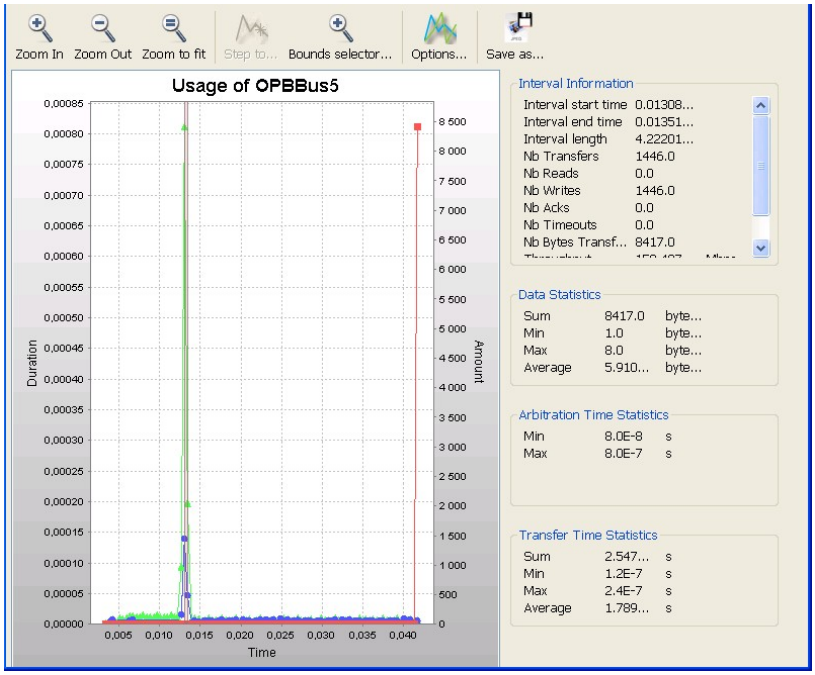


Figure 4.6 Utilisation du bus

donc pour 16 sous-canaux utilisés, 1 symbole envoyé et une modulation QPSK-3/4. Notre système a une fréquence de fonctionnement de 25MHz comme pour le modèle en *Simulink* ce qui fait qu'un cycle est égal à 40 ns et que les tests ne concernent que la chaîne de codage.

Etant donné qu'en matériel, il est possible de paralléliser les données, cela donne des temps de simulation moindres. Il s'agit donc de déterminer quels modules logiciels ont une durée très au-dessus de la moyenne qui est approximativement de 13.31 Mcycles. On constate que les blocs les plus lents sont le De-Entrelaceur et le Viterbi qui nécessitent une mémorisation des données avant calcul.
Le tableau 4.3 nous présente les résultats de simulation pour différentes configurations de partitionnement : Matériel-Matériel via l'OPB (1), Materiel-Materiel à travers des liens directs (2), Matériel-Logiciel avec le Microblaze (2 et 3).

On constate que la configuration 2 est la plus rapide ce qui est normal car les modules communiquent à travers des liens sans avoir à passer par le bus. Cela évite les délais

Tableau 4.2 Cycles de Simulation par module et pour un Symbole OFDM

Modules	HWLatence(cycles)	SW(Mcycles)
SEND	N/A	0.0113
RECEIVE	N/A	0.0633
Param	N/A	0.0251
Melangeur	358	0.2762
Encodeur Reed-Solomon	417	0.3515
Convolution	470	0.3773
Perforation	2050	0.5942
Entrelaceur	472,582	12.4396
Assignation	7736	0.6488
Formation Symbole OFDM	9180	0.9606
De-Assignation	9875	0.8721
De-Entrelaceur	646,885	12.7388
Décodeur Viterbi	2,161,716	169.0
Décodeur RS	37285	0.3843
De-Melangeur	356	0.2011

dû aux mécanismes d'arbitrage du bus. On remarque aussi que le fait de placer les modules en logiciel augmente le temps d'exécution du modèle ce qui est normal étant donné que les données sont exécutées de façon séquentielle en logiciel et parallèle en matériel.

D'Autre part, lors de la compilation, nous avons constaté que la mémoire LMBRAM du processeur de la carte utilisée[3] ne permet pas de supporter une application aussi lourde que celle réalisée; en effet il faudrait au moins 3 processeurs Microblaze pour supporter l'application complète. Ainsi après plusieurs essais nous avons constaté que le microblaze ne pouvait supporter que peu de modules logiciels. Nous avons ainsi choisi la configuration 3 représentant les modules Send, Receive et Param en logiciel puisqu'il ne font pas réellement partie de la chaîne de codage et avons laissé les blocs de la chaîne de codage en matériel avec une communication par liens directes.

[3]voir résultats de synthèse plus loin

Tableau 4.3 Résultats de temps d'exécution par configuration

No	Configuration	Temps d'exécution simulé (ms)	Temps de simulation (s)
1	SW = {} HW = {*Tous les modules*}	130.8	99
2	SW = {} HW-LD = {*Tous les modules*}	61.7	26
3	SW = {*Send*, *Param*, *Receive*} HW-LD = {*Modules restants*}	422	372
4	SW = {*Melangeur*} HW-LD = {*Modules restants*}	448.3	393

4.3.3 Résultats de synthèse

Les différents blocs de la chaîne de communication on été synthétisé sous EDK pour une carte FPGA Virtex-II. Les éléments de l'architecture ont été générés automatiquement par GenX (voir chapitre précédent). Le tableau 4.4 nous présente les résulats de synthèse par l'outil *ForteCynthesizer* des différents blocs matériels de l'architecture obtenue à partir du profilage dans Space Codesign. Etant donné que les modules Send, Receive, Param,sont placés en logiciel, ils ne font pas partie du tableau. Dans ce tableau nous présentons également les résultats de synthèse des blocs de *Xilinx* ou de ceux codés à la main, car le bloc Mélangeur n'étant pas fourni par la librairie *Coregen* de *Xilinx*. Nous l'avons donc codé manuellement.

Il est possible d'estimer le nombre total de slices au niveau de Space Codesign en faisant une addition du nombre de slices des blocs synthétisés. Nous obtenons ainsi 18279 slices pour les blocs de Coregen et 29675 pour ceux de Forte sachant que nous avons réalisé les blocs Mélangeur et dé-melangeur à la main. Cela nous donne une augmentation d'environ 38% sans avoir rajouté les mémoires externes des différents blocs de Coregen (4 pour le Viterbi, 2 pour le Reed-solomon, 1 pour l'entrelaceur, 1

Tableau 4.4 Nombre de Slices par module

Modules	Blocs de Coregen	Blocs de Forte
Melangeur	69 (0%)	418(3%)
Encodeur Reed-Solomon	956(8%)	1085(10%)
Convolution	186(1%)	186(1%)
Perforation	29(0%)	3293 (30%)
Entrelaceur	144(1%)	3447(32%)
De-Entrelaceur	130(1%)	1246(12%)
Décodeur Viterbi	1080(10%)	2264(22%)
Décodeur RS	4306(40%)	6104(56%)
De-Melangeur	69 (0%)	326(3%)
MicroBlaze	1188(11%)	1188(11%)

pour le dé-entrelaceur).

Nous pouvons remarquer que nos blocs synthétisés occupent une plus grande part de slices que ceux de *Xilinx*, cela étant dû au fait que nous n'utilisons pas de mémoire externe contrairement aux blocs Viterbi, décodeur Reed-Solomon et Entrelaceur/Dé-Entrelaceur.

La différence significative entre les blocs de coregen et les blocs de Forte s'explique par le fait que les blocs de Coregen sont spécifiquement optimisés pour les différents modules alors que les blocs générés automatiquement par Forte suivent un processus de rafinement générique. De plus, l'orthogonalisation des communications et des calculs dans l'approche TLM ainsi que l'automatisation du partitionnement peuvent dégrader la génération d'une implémentation optimale de ces modules.

Toutefois, dans le cas de blocs pré-définis (IPs) tel que les blocs Coregen, une grande partie de la surface est occupée par les processeurs, mémoires, bus et non pas par les modules matériels eux-mêmes. Dans ce cas, l'utilisation d'une méthodologie d'automatisation du raffinement des communications et de la synthèse matérielle compense largement cet inconvénient car on obtient au final un code facilement modifiable et optimisable car à haut niveau que l'on peut de façon automatique synthétiser vers le matériel. De plus, on obtient une application modularisée ce qui

la rend très extensible. Ceci est un avantage dans la conception de réseaux sans fil où les normes évoluent très rapidement.

Nous n'avons malheureusement pas pu effectuer une exécution complète du modèle sur le FPGA étant donné la taille restreinte de la carte dont nous disposions. Mais nous pouvons tout de même supposer que pour une implémentation finale, nous obtiendrons une architecture assez performante étant donné qu'une simulation complète sous *Modelsim* a été effectuée. Nous supposons que la différence entre les deux modèles RTL se trouvera réduite étant donné l'absence de mémoires contrairement à Coregen qui verra ses performances se dégrader à cause des accès mémoire.

CONCLUSION ET TRAVAUX FUTURS

Les systèmes sur puce occupent une place très importante dans l'implémentation de réseaux sans fil car ils constituent la base de la conception de systèmes embarqués que sont les téléphones mobiles, les cartes réseaux, etc. D'après la prédiction de Gordon Moore, le nombre de transistors disponibles sur une puce de silicium doublerait tous les deux ans. Cette prédiction, qui s'avère exacte, entraîne un écart de productivité entre le nombre de transistors disponibles et ceux réellement utilisés. La grande évolution des réseaux sans fil et la compétitivité liée au domaine pose le problème de conception de ce type de systèmes.
Les techniques de co-design, qui sont des méthodes de conception, réduisent la complexité des systèmes et leur temps de réaliation en vue d'atteindre les objectifs liés à la productivité (coûts) en diminuant le temps de mise sur le marché du produit(time-to-market).

Ce mémoire s'articule sous deux thèmes principaux : la conception de la couche physique OFDM de la norme IEEE 802.16 grâce à la plateforme ESL SpaceStudio et la cosimulation que nous avons exploitée afin de valider notre système.

La conception d'une application s'effectuant selon un schéma bien défini :

1. étude des spécifications
2. modélisation, réalisation du système
3. validation à partir des spécifications

Nous avons effectué une étude approfondie du standard représentant notre application ainsi que des travaux reliés tel que le Wifi afin de déterminer la meilleure façon de la réaliser. Cette étude a représenté l'une des plus importantes parties de notre travail

car d'elle dépendait la validité du résultat final.

Notre travail s'effectuant dans le cadre du projet technologique *Space Codesign*, à travers notre application, nous avons réussi à exploiter la plupart des possibilités actuelles qu'offre cet outil. Cet outil nous a permis de réaliser assez rapidement une des plus essentielles parties du Wimax faite en général par des compagnies spécialisées avec de nombreuses ressources humaines et matérielles. En partant d'un modèle algorithmique que nous avons réalisé sous *Simulink* et auquel nous avons soumis les exigences requises par la norme IEEE 802.16, nous avons réalisé son équivalent en SystemC au niveau de la plateforme *Space Codesign*. Puis ce modèle a été partitionné puis raffiné automatiquement et différentes configurations architecturales lui ont été appliquées. Grâce aux outils de profilage de la plateforme, nous avons déterminé le partitionnement optimal de notre système permettant ainsi de remplir les contraintes matérielles et temporelles liées à la conception d'une application de réseau sans fil. Elle nous a également fourni via son outil *GenX*, tous les éléments architecturaux (bus, processeurs, mémoires, timeurs,...) nécessaires à une implémentation d'un système sur puce en générant à partir de la plateforme un projet *XPS* de la plateforme *Xilinx* de synthèse et d'implémentation sur FPGA. Ainsi, 3 modèles de conception à 3 niveaux d'abstraction utilisant trois platformes différentes ont été réalisés. De plus, afin de valider les requis de la norme au niveau du modèle système, nous avons réalisé une interface de co-simulation à haut niveau entre le modèle de *Simulink* et le modèle de *Space Codesign*.

Ainsi, à travers ce projet, nous avons abordé deux domaines d'études essentiels : les télécommunications et les systèmes embarqués temps réels qui se rejoignent à travers des applications telles que le Wimax.

Bien que notre travail se termine là, il existe de nombreuses perspectives permettant

d'enrichir le modèle que nous avons réalisé. L'utilisation de la cosimulation pour la validation de notre modèle a permis d'ouvrir de nouvelles perspectives à Space Codesign concernant une possible intégration de l'outil *Matlab-Simulink* à la plateforme. Pour cela, il existerait deux solutions : soit utiliser la technique de cosimulation choisie pour la réalisation de ce projet; soit de l'autre réaliser l'intégration de l'outil CODIS (présenté au chapitre 1) à la plateforme Space Codesign.

Nous pensons également, qu'il serait intéressant d'étudier la couche Mac du standard et de la réaliser sous Space Codesign complétant ainsi l'application du Wimax. De plus, notre modèle système pourrait être optimisé en essayant de paralléliser certains blocs tels que le Reed-Solomon, le Viterbi et l'Entrelaceur en effectuant un partitionnement interne de ces éléments et en y rajoutant des mémoires. Un dernier champ d'exploration serait la possibilité d'utiliser l'outil de cosimulation *VHDL/Simulink* au niveau RTL, le *SystemGenerator* de *Xilinx*, afin d'exploiter toute la partie analogique laissée de côté lors de notre implémentation.

RÉFÉRENCES

[1] Altera. Accelerating WiMAX System Design with FPGAs. Consulté le 2007-11-29, tiré de http://www.altera.com/literature/wp/wp_wimax.pdf.

[2] Baek, J. H., Kang, J. Y. et Sunwoo, M. H. Design of a High Speed Reed-Solomon Decoder. Consulté le 2008-08-10, tiré de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1010823.

[3] Bernstein, A., Burton, M. et Ghenassia, F. (2004). How to bridge the abstraction gap in system level modeling and design. *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on.* (pp. 910–914).

[4] Boland, J.-F., Thibeault, C. et Zilic, Z. (2004). Using Matlab and Simulink in a SystemC verification environment. *Nascug, DVCon 2004.*

[5] Bonnerud, T. E., Hemes, B. et Ytterdal, T. (2001). A mixed-signal, functional level simulation framework based on SystemC for system-on-a-chip applications. *Custom Integrated Circuits, 2001, IEEE Conference on.* (pp. 541–544).

[6] Bouchima, F., Bière, M., Nicolescu, G., Abid, M. et Aboulhamid, E. (2006). A systemC/Simulink Co-SImulation Framework for Continuous/Discrete-Events Simulation. *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International.* (pp. 1–6).

[7] Cai, L. et Gadjski, D. (2003). Transaction level Modeling: an overview. *Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference on.* (pp. 19–24).

[8] Cai, Z., Hao, J., Sun, S. et Chin, F. P. (2006). A High-Speed Reed-Solomon Decoder for Correction of Both Errors and Erasures. *Circuits and Systems,*

2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on. (p. 4pp).

[9] Chang, H.-C. et Shung, C. B. (1999). New Serial Architecture for the Berlekamp-Massey Algorithm. *Communications, IEEE Transactions on.* (pp. 481–483 vol.47 issue.4).

[10] Chen, H.-W., Wu, J.-C., Huang, G.-S., Lee, J.-C. et Chang, S.-S. . A new VLSI architecture of Reed-Solomon decoder with erasure function.

[11] Clark, M. IEEE 802.11a WLAN model. Consulté le 2007-09-20, tiré de http://www.mathworks.com/matlabcentral/fileexchange/3540.

[12] Deltoso, C., Joanblanq, C., Cand, M. et Senn, P. (1996). Fast prototyping based on generic and synthesizable VHDL models. A case study: punctured Viterbi decoders. *Rapid System Prototyping, 1996. Proceedings., Seventh IEEE International Workshop on.* (pp. 158–163).

[13] Dietler, S. Implémentation de codes de Reed-Solomon sur FPGA pour communications spatiales. Consulté le 2008-08-10, tiré de http://www.sweegy.ch/documents/reports/Projet_Diplome_2005%20v1.1-dietler%20.pdf.

[14] Doyle, A. J., Han, K., Nadkarni, S., Seshadrinathan, K., Simha, R. et Wong, I. C., (2003). Performance evaluation of the ieee 802.16a physical layer using simulation. Rapport technique, University of Texas at Austin.

[15] Dubois, M. et Aboulhamid, E. (2005). Techniques to improve cosimulation and interoperability of heterogeneous models. *Electronics, Circuits and Systems, 2005. ICECS 2005. 12th IEEE International Conference on.* (pp. 1–4).

[16] Filion, L., Cantin, M.-A., Moss, L., Aboulhamid, M. et Bois, G. (2007). Space Codesign : A SystemC Framework for Fast Exploration of Hardware/Software Systems. *DVCon 2007.*

[17] Forum, W. About Wimax Forum overview. Consulté le 2008-10-15, tiré de http://www.wimaxforum.org/about/about-wimax-forum-overview.

[18] Fujitsu Microelectronics America, I. WiMAX Subscriber Station Design Using the Fujitsu 802.16-2004 SoC Reference Kit. Consulté le 2007-11-29, tiré de http://www.fujitsu.com/us/services/edevices/microelectronics/wsubstation.html.

[19] Garcia, J. et Cumplido, R. (2005). On the design of an FPGA-Based OFDM modulator for IEEE 802.16-2004. *Reconfigurable Computing and FPGAs, 2005. ReConFig 2005. International Conference on.* (p. 4 pp).

[20] Gruyer, P. et Paillard, S. Modélisation d'un modulateur et démodulateur OFDM. Consulté le 2008-08-10, tiré de http://symoon.free.fr/scs/ofdm/RapportOFDM.pdf.

[21] Hasan, M. A. Performance Evaluation of WiMAX/IEEE 802.16 OFDM Physical Layer. Consulté le 2007-11-29, tiré de http://lib.tkk.fi/Dipl/2007/urn009599.pdf.

[22] Hu, Q., Wang, Z., Zhang, J. et Xiao, J. (2005). Low Complexity Parallel Chien Search Architecture for RS Decoder. *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on.* (pp. 340–343 vol.1).

[23] Huang, K., Han, S.-i., Popovici, K., Brisolara, L., Guerin, X., Li, L., Yan, X., Chae, S.-I., Carro, L. et Jerraya, A. A. (2007). Simulink-Based MPSoC Design Flow: Case Study of Motion-JPEG and H.264. *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE.* (pp. 39–42).

[24] Initiative, O. S. SystemC 2.2. Consulté le 2008-03-15, tiré de http://www.systemc.org/.

[25] Intel. Scalable OFDMA Physical Layer in IEEE 802.16 WirelessMAN. Consulté le 2007-11-29, tiré de ftp://download.intel.com/technology/itj/2004/volume08issue03/art03_scalableofdma/vol8_art03.pdf.

[26] Jeng, J.-H. et Truong, T.-K. (1999). On Decoding of Both Errors and Erasures of Reed-Solomon Code Using an Inverse-Free Berlekamp-Massey Algorithm. *Communications, IEEE Transactions on.* (pp. 1488–1494 vol.47 issue.10).

[27] Keutzer, K., Malik, S., Newton, A., Rabaey, J. et Sangiovanni-Vincentelli, A. (2000). System Level Design: Orthogonalization of Concerns and Platform-Based Design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on.* (pp. 1523–1543).

[28] Khan, M. N. WiMAX 802.16e PHY and MAC (beta). Consulté le 2007-09-20, tiré de http://www.mathworks.com/matlabcentral/fileexchange/21039-wimax-802-16e-phy-and-mac-beta.

[29] Lou, H. (1995). Implementing the Viterbi algorithm. *Signal Processing Magazine, IEEE.* (pp. 42–52 vol.12 issue.5).

[30] Mahmood, A. et Hammami, O. (2005). Hardware-Software Co-simulation of Adaptive MC-CDMA Physical layer for Wireless Communications. *Microelectronics, 2005. ICM 2005. The 17th International Conference on.* (pp. 305–309).

[31] Moore, G. E. (1998). Cramming More Components Onto Integrated Circuits. *Proceedings of the IEEE, vol. 86, no.1, January 1998.* (pp. 82–96).

[32] Moss, L., Cantin, M.-A. et Bois, G. (2008). Automation of Communication Refinement and Hardware Synthesis within a System-Level Design Methodology. *Rapid System Prototyping, 2008. RSP '08. The 19th IEEE/IFIP International Symposium on.* (pp. 75–81).

[33] Moss, L., De Nanclas, M., Filion, L., Fontaine, S., Bois, G. et Aboulhamid, M. (2007). Seamless Hardware/Software Performance Co-Monitoring in a Codesign Simulation Environment with RTOS Support. *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07.* (pp. 1–6).

[34] Nicopolitidis, P., Obaidat, M., Papadimitriou, G. et Pomportsis, A., (2003). *Wireless Networks.* Wiley.

[35] Perrier, V. (2004). A look inside electronic system level (ESL) design. *Exclusive features of EEDesign News of EEtimes.*

[36] Saleem, A., Ikram, A. A., Bazuin, B., Shah, S. I. et Saleem, Z. (2008). Design and Implementation of a Software Radio based WiMAX Communication System using LabVIEW. *Networking and Communications Conference, 2008. INCC 2008. IEEE International.* (pp. 39–43).

[37] Swan, S. (2006). SystemC transaction level models and RTL verification. *Design Automation Conference, 2006 43rd ACM/IEEE.* (pp. 90–92).

[38] Systems, F. D. Cynthesizer 3.0. Consulté le 2008-11-15, tiré de http://www.forteds.com/highleveldesign.

[39] Szeker, B. Design and development of basic physical layer WiMAX network simulation models. Consulté le 2007-11-29, tiré de http://pubs.drdc.gc.ca/PDFS/unc80/p530808.pdf.

[40] Xi, C., Ningyi, X. et Zucheng, Z. (2003). A methodology for SystemC algorithmic model verification applying MATLAB. *ASIC, 2003. Proceedings. 5th International Conference on.* (pp. 294–297 vol.1).

[41] Xilinx, I. Wimax Solutions. Consulté le 2007-11-29, tiré de http://www.xilinx.com/company/press/kits/wimax/WiMAX_editorpres.pdf.

ANNEXE I

RÉSEAUX SANS FIL

Au Canada l'organisme chargé de réglementer les bandes fréquentielles et les spectres électromagnétiques est le ministère de l'industrie du gouvernement Canadien.

I.1 Des réseaux sans fil au Wimax

Un réseau sans fil est un réseau informatique (numérique) qui connecte différents postes ou systèmes entre eux par ondes radio. Il peut-être associé à un réseau de télécommunications (téléphonie, télévision) pour réaliser des interconnexions entre noeuds. Il est découpé en catégories (PAN, LAN, MAN, WAN)[1] selon le périmètre géographique désservi (voir figure I.1):

Les réseaux WMAN quant à eux se différencient au niveau de leur couche physique par leur fréquence d'émission (fréquence centrale) et leur débit binaire:

- WMAN-SC (WMAN à porteuse unique): ce type de réseau fonctionne à des fréquences allant de 10 à 66GHz. A ces fréquences, les opérations à vue(LOS) sont nécessaires à causes des propriétés électromgnétiques des ondes ce qui rend la mobilité impossible. Il utilise une modulation(QPSK, QAM) adaptée pour chaque SS.
- WMAN-SCa (WMAN à porteuse unique): fonctionne à des fréquences en-dessous de 11 GHz. Il a les mêmes spécifications que le WMAN-SC avec de

[1] voir Acronymes

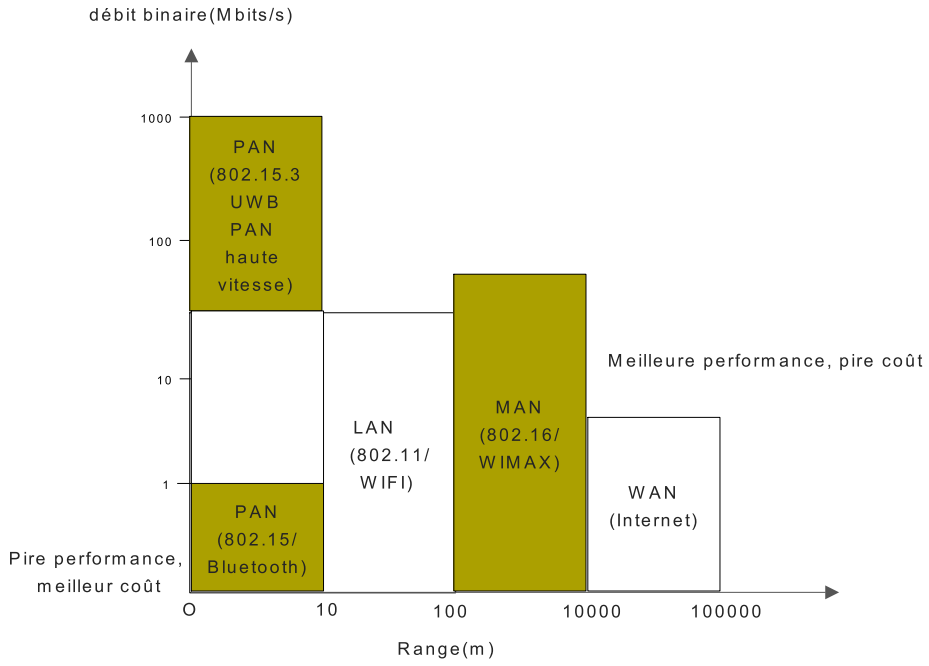


Figure I.1 Catégories de réseaux sans fil

légères différences au niveau de l'encodage des données. Il n'est cependant pas utilisé dans l'industrie.

- WMAN-OFDM (WMAN à porteuses multiples): fonctionne à des fréquences en-dessous de 11GHz et utilise une modulation OFDM.

- WMAN-OFDMA (WMAN à porteuses et accès multiples): fonctionne également à des fréquences inférieures à 11GHz et utilise une modulation OFDMA qui est une extension de l'OFDM avec un accès multiple.

Le WMAN-SCa ne trouvant pas d'application dans l'industrie et l'OFDMA basée étant encore en cours de recherche, l'OFDM est à ce jour la plus appliquée dans l'industrie.

I.2 Définitions

La norme 802.16 est un système assez complexe; seules les parties nécessaires à la compréhension de sa couche physique seront développées. Cette partie présente donc quelques conceptes de bases qui sont utilisés tout au long de ce document.

1. Symbole: c'est l'élément de base d'une transmission. Dans les systèmes numériques, il s'agit selon les cas de bits, d'octets, de groupes de bits, etc.....

2. Burst: c'est un paquet transféré dans le canal, il est constitué de plusieurs symboles (dans notre cas OFDM). Avant modulation, ce sont les informations binaires transmises qui sont regroupées en burst.

3. Trame: il s'agit d'un paquet transféré par une station. En général il est constitué d'un ou plusieurs bursts.

4. LOS: ce sont les conditions pour lesquelles le parcourt du signal est visible à 60% dans la zone de Fresnel; c'est à dire la région du parcourt dans laquelle un signal réfléchi aura un déphasage inférieur à 90°. Il s'agit de la ligne de vue.

5. NLOS: dans ces conditions le signal n'a pas de ligne de vue. Ces deux propriétés sont définies pour des radiofréquences.

6. Bande Fréquentielle: cet élément est très important car il constitue une mesure de la capacité de transmission d'un canal. Le Wimax étant un système de réseau sans fil, le canal de transmission est l'Air car les informations sont transmises sous forme d'ondes à l'aide d'antennes de transmission et de réception. Il faut donc pouvoir déterminer la quantité d'informations transmises à un instant donné d'où la notion de bande fréquentielle ou bande passante. Dans le cas du Wimax, la bande passante est exprimée en Megahertzs (MHz) ou Gigahertzs (GHz).

7. Canal de Transmission: la bande fréquentielle est souvent divisée en plusieurs canaux permettant ainsi de réaliser différentes communications au même instant. En effet, le Wimax faisant du Point-to -Multipoint, à un instant donné un élément du réseau peut communiquer avec un ou plusieurs autres éléments.

8. AAS(Adaptative antenna systems): les antennes sont les éléments responsables de la conversion des signaux modulés en ondes électromagnétiques lors de la transmission ou de la démodulation à la réception. Puisque le Wimax fonctionne sur un réseau MAN c'est-à-dire sur de grandes distances il faut que le signal soit amplifie au cours de la transmission d'où la nécessité d'avoir des antennes adaptables au type de transmission et au type de données.

9. Porteuses uniques (single carrier) et multiples: la transmission des données sur une longue distance entraine des problèmes de délai de propagation et d'interférence car les signaux sont transmis sous forme d'ondes électromagnétiques. Pour les résoudre on peut utiliser des techniques d'égalisation de l'enveloppe spectrale de l'onde dans une transmission à porteuse unique cependant la complexité de cette technique augmente en fonction du taux de transfert des données et de la variation du canal dans le domaine temporel. Lorsqu'on utilise une modulation à porteuses multiples, le fait de diviser le canal en sous porteuses permet de diminuer le débit de transfert des données rendant ainsi la technique d'égalisation plus efficace.

10. Duplexing,multiplexing et multiple access: le duplexage (multiplexage)est une technique consistant a transmettre dans une même tranche de temps ou une même bande fréquentielle différentes informations. Il vise à réaliser une communication bidirectionnelle entre deux composants du réseau. Par exemple entre une Station de base (BS) et une sous station (SS) tout en ayant des données protégées en écriture et en lecture. Il existe deux types de duplexage (dans le temps et en fréquence) :

- TDD (time division duplexing): il s'agit d'une méthode half-duplex c'est-à-dire qu'il s'agit d'une communication bidirectionnelle qui ne s'effectue pas en même temps. Lorsqu'une station émet/reçoit vers une autre elle ne peut pas recevoir/émettre pendant ce temps, il faut qu'elle ait finit d'émettre/recevoir avant de commencer à recevoir/émettre. Dans ce cas l'émission et la réception peuvent se faire dans les mêmes bandes de fréquence (voir figure ci-après). Le temps alloué dans chaque direction peut être fixe ou adaptable.
- FDD (frequency division duplexing): dans ce cas l'émission et la réception se font dans des bandes de fréquence différentes et peuvent donc ainsi se faire en même temps. Il s'agit donc d'une méthode full-duplex. Il existe toutefois un half-duplexing fréquentiel qui se trouve être une combinaison des deux (TDD et FDD). La technique FDD requièrt une bande fréquentielle plus grande que la TDD pour un même système ce qui entraine une plus grande possibilité d'interférence surtout dans les réseaux ne requérant pas de licence. Toutefois les fonctions de qualité de service et d'accès au canal de communication sont moins complexes pour un système FDD que pour le TDD.

11. Multiplexing: c'est comme pour le duplexage sauf que le TDD est remplace par un TDM et le FDD par un FDM pour permettre a un composant de communiquer avec plus de 2 autres composants en même temps.

12. Multiple access: cette section explique les différentes techniques disponibles dans le Wimax pour qu'un composant réseau accède au canal de communication (les ondes).

13. TDMA : plusieurs stations ont des tranches de temps prédéterminées pour transmettre et leur nombre ainsi que leur localisation peuvent être statiquement ou dynamiquement assignés. S'ils sont assignés dynamiquement, une certaine

coordination ou distribution est nécessaire. Il est aussi possible de fournir un accès déterministe au canal. Par exemple dans le Wimax lors d'une communication UL, la BS alloue les tranches de temps. Lors d'une communication DL elle ordonnance les transmissions utilisant les bursts TDM (flots continus de données) adressés à des SS spécifiques.

14. FDMA: plusieurs stations utilisent différentes fréquences pour accéder au medium. Toutefois la quantité de bande passante requise et les transmissions sur des fréquences adjacentes diminuent l'efficacité de cette technique.

15. CDMA: chaque utilisateur a un code unique avec lequel il se connecte au réseau. Au lieu de transmettre une simple onde renfermant les données, l'émetteur envoie la convolution temporelle de l'onde et du code assigné ce qui augmente considérablement le taux de transmission.

16. OFDM: modulation à porteuses multiples utilisant TDMA/FDMA/CSMA. Le Wimax utilise la modulation OFDM avec la technique TDMA/FDMA. Ce n'est donc pas une méthode d'accès au canal.

17. OFDMA: extension de OFDM comme méthode d'accès dans laquelle un ensemble de sous porteuses peut être alloué a un composant pour l'accès au réseau.

I.3 Wimax versus Wifi

Cette section donne quelques comparaisons entre la norme 802.16 du Wimax et la norme 802.11 du Wifi. Contrairement à certaines idées préconçues ces deux normes ne sont pas concurrents mais plutôt complémentaires. Il existe plusieurs technologies de réseaux sans fil conçues pour différents domaines:

- Personal area networks (PANs): un exemple de technologie PAN est le bluetooth
- Local area networks (LANs): un exemple de LAN est le Wifi

- Metropolitan area networks (MANs): un exemple est le Wimax
- Wide area networks (WANs): un exemple est Internet

Ces différents réseaux diffèrent par:

* bande fréquentielle occupée
* la distance de couverture
* la puissance requise
* les services offerts

La norme 802.16-2004 est construite pour des modèles d'accès fixes (le 802.16-2005 rajoute la possibilité d'accès mobiles).

Il existe donc quelques différences entre le Wimax et le Wifi (voir tableau I.1) qui sont en fait des améliorations apportées au Wifi. La principale amélioration apportée se situant au niveau de la couche physique, certaines différences se trouvent être:

I.4 Principe de fonctionnement du Wimax

Le Wimax(worldwide interoperability for microwave access) est un système embarqué complet(c'est à dire logiciel-matériel) car résolvant des tâches spécifiques tout en étant intégré à un élément tel qu'un téléphone, pda, gps, etc... Il regroupe des normes IEEE de réseaux sans fil étendus dont le 802.16 et comprend toutes les spécifications de bases nécéssaires à l'implantation de ce type de réseau. Du fait de l'expansion des populations surtout dans les pays en voie de développement, les réseaux filaires existants demeurent insuffisants et leur extension coûteuse. Il devient donc nécéssaire de trouver de nouvelles techniques de transmission rapides et à faible coût de réalisation. Le Wimax tel que spécifié dans la norme promet des débits de plusieurs dizaines de mégabits/seconde sur des rayons de couverture

Tableau I.1 Comparaison Wifi-Wimax

	Wimax	Wifi
débit de transfert maximal(Mbits/s) au niveau de la couche physique	72	54
tolérance aux délais de propagation	10μs	900ns
Maximum MAC overhead	6 octets	28 octets
Qos (qualité de service)	Le Wimax fournit une meilleure qualité de service que le Wifi car elle est intégrée très tôt dans la conception du système. Par exemple il peut augmenter la bande passante lorsqu'il s'agit d'un domaine commercial ou ralentir la transmission des données pour une communication téléphonique ou pour des applications video. Et s'il s'agit d'une communication standard il peut fournir une qualité moyenne de service	

de quelques dizaines de kilomètres.Toutefois dans la réalité, quand le signal est stable, le Wimax permet des débits relativement importants, de l'ordre de 1 à 10 Mbits/s. Contrairement à l'ADSL, le spectre radio est partagé entre les utilisateurs et contrairement au Wi-Fi, qui utilise des bandes de fréquence radio non soumises à licence, l'opérateur contrôle entièrement le spectre et peut ainsi garantir des débits minimum. Il permet l'interopérabilité avec les réseaux existants grâce à des techniques de multiplexage des données telles que l'OFDM(orthogonal frequency division multiplexing).

Pour s'identifier au réseau il faut les deux plus basses couches du modèle OSI ou la première couche du modèle TCP/IP. Ces couches représentent la couche MAC et la couche Physique appelée PHY et sont les seules qui différentient un type de réseau d'un autre (figure I.2). Les éléments de la couche Mac de la norme IEEE 802.16

Quantification	Le Wimax utilise le protocole de Grant-request access pour l'envoi des données qui ne permet pas la collision des données et par conséquent rend la bande passante plus efficace grâce à sa technique de retransmission des données. Grâce à des canaux fréquentiels flexibles et à une modulation adaptative, il permet une plus grande connectivité des usagers. De plus, cette flexibilité des canaux donne des bandes passantes plus étroites, ce qui permet des taux de transferts très bas sans perte de bande passante. En présence de bruit ou lorsque les signaux transmis faiblissent en intensité, la modulation adaptative permet aux usagers de rester connectés quand ils devraient être déconnectés
Accès au Canal	Une différence entre le Wifi et le Wimax se situe au niveau de l'accès au medium de communication. En effet si dans le premier l'accès se fait par une technique CSMA/CA c'est à dire par un mécanisme d'envoi et de réception par acknowledge. Le Wimax, quant à lui est basé sur une technique TDM/TDMA/OFDMA qui est un multiplexage dans le temps ou en fréquence. Dans ce cas le medium est divise entre les différents usagers ce qui évite les interférences de données
Taille des données	Du fait de la possibilité de division du canal de communication, la taille du canal n'est pas la même pour le Wifi (22 MHz) que pour le Wimax (1.25 à 28MHz)

présentés sont communs à toutes les couches physiques supportées.

I.5 Processus de communication

La communication se fait point à point entre une station de base(BS) qui est la tour de contrôle et des stations clientes (SS) qui sont des stations de relais ou l'utilisateur final(figure I.3).Les différenciations des réseaux se situent au niveau des couches MAC et PHYSIQUE. Par exemple un réseau LAN se différencie d'un réseau MAN par sa couche MAC et sa couche physique tandis que deux réseaux MAN se distinguent par leur couche Physique.

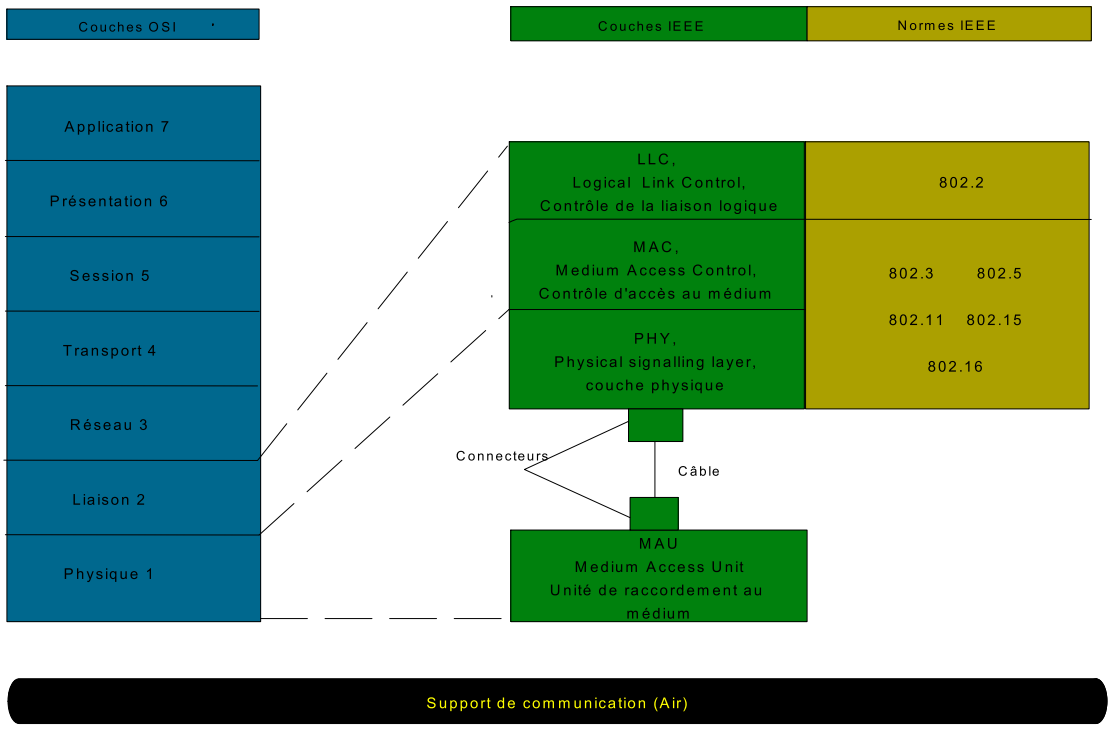


Figure I.2 Correspondance OSI-IEEE

Ainsi lors d'une communication entre une BS et une SS, les couches MAC des deux stations se connectent et s'identifient(c'est à dire que la SS se synchronise à la BS). Ensuite les données sont transmises à la couche inférieure (qui se trouve être la couche Physique) puis envoyées à travers le canal de communication.

Il existe deux types de communication entre une BS et une SS : DL(downlink : de la BS vers la SS) et UL(uplink : de la SS vers la BS).

La BS fixe les paramètres de base des couches MAC et PHY tels que la taille des trames[2], le duplexage des données et la configuration du système; elle alloue les bandes fréquentielles pour les communications DL/UL, ordonnance la communication par trame de toutes les SS, transmet/reçoit les données de contrôle et d'information

[2]Rq: les paquets échangés sont des trames(frames en anglais), ils sont découpés en burst eux-mêmes subdivisés en symboles qui sont des ensembles de bits(éléments binaires valant 0 ou 1)

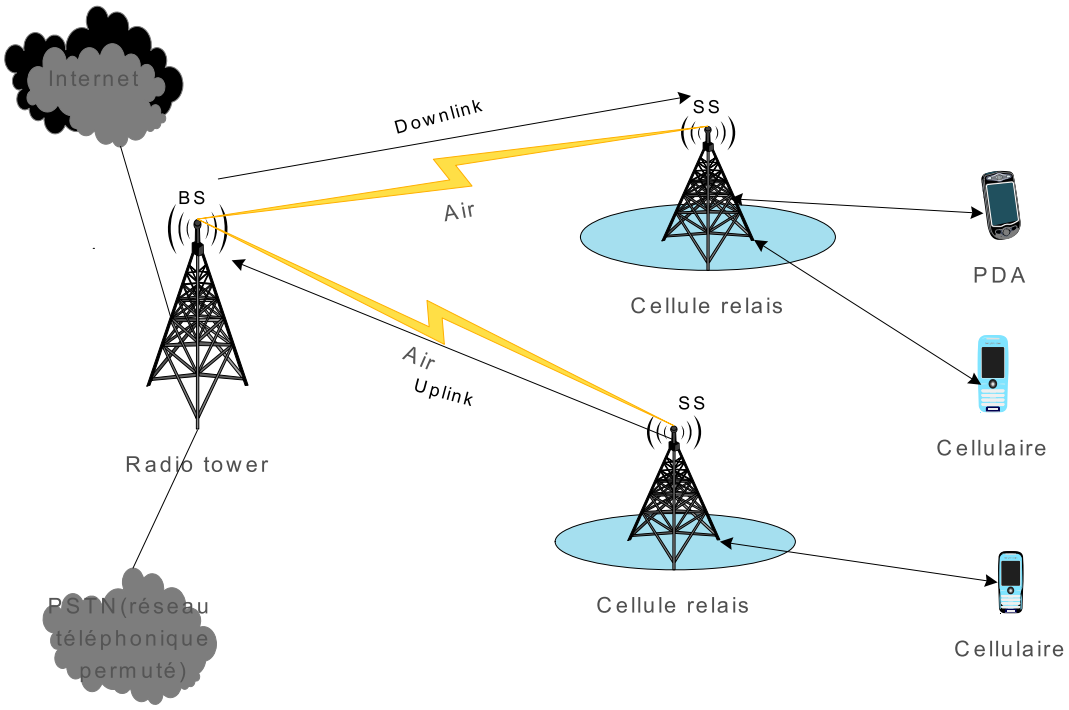


Figure I.3 Communication Wimax

pour chaque trame.

La SS quant à elle identifie la BS et se synchronise au niveau de la couche PHY et reçoit les PDUs[3] diffusés(par broadcast/multicast) et les renvoie vers les utilisateurs (voir figure I.4).

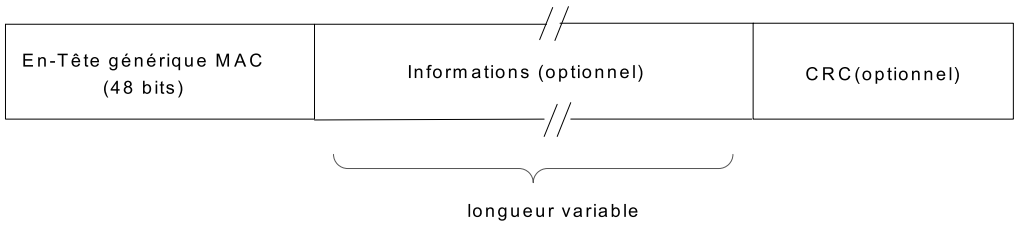


Figure I.4 Structure d'un PDU MAC

[3]les PDUs sont les paquets échangés entre deux niveaux de couche IEEE

I.6 Processus d'identification d'une BS et d'une SS

La couche MAC est chargée du contrôle de l'accès au canal(médium). Elle utilise des techniques d'accès multiples(Point-Multipoint) ou centralisé(Mesh). Elle est en outre responsable de l'encapsulation et de la fragmentation des données venant/allant de/vers la couche physique(packets PDU). Les données de base échangées entre deux couches ou sous-couches du protocole réseau sont appelés PDU et SDUs selon les cas. Les SDU sont les données échangées entre deux couches/sous-couches de protocole(OSI,IEEE,TCP/IP) adjacentes d'un même matériel. Les PDU sont les données échangées entre deux couches de protocole identiques de matériels différents. Quant aux SDUs, ils agissent sur les PDUs reçus que la couche MAC transmet aux autres couches du protocole utilisé à l'aide d'une interface de point d'accès(SAP: service access point). Elle est divisée en trois sous-couches : CS(convergence sublayer), CPS(couche fonctionnelle Mac), (security sublayer).

Ainsi une fois la SS activée, le processus d'identification et d'accès au réseau débute. C'est ce processus de synchronisation qui permettra à la SS d'obtenir tous les paramètres nécessaires à la communication.

Une fois la synchronisation effectuée, la connection au réseau débute(c'est la période d'initialisation). La durée des symboles transmis par la SS est ajustée de telle sorte à correspondre à celle de la BS. Ensuite la puissance du signal de la SS est ajustée pour un niveau de réception optimal(synchronisation fine) et on alloue à la SS des codes d'identification permettant d'effectuer les opérations. Le fréquence des SS peut également être ajustée. La procédure d'initialisation diffère d'une couche physique à une autre. Une fois l'initialisation terminée la SS indique à la BS quelles options elle supporte c'est à dire les spécificités des couches MAC et physique telles que la puissance maximale du signal que la SS est capable d'émettre,

le type de modulation supportée, les codes de détection et correction supportés ainsi que la bande fréquentielle à allouer. Cette étape est nommée (SS basic capability negotiation). Enfin, une fois la période d'initialisation terminée, la couche Mac transfère avec les paramètres spécifiés les données qui seront encodées puis modulées avant d'être transmises à travers le canal de communication(voir chapitre 3).

I.7 Structure des données échangées

La couche physique OFDM (voir chapitre 2) supporte deux structures de trames incompatibles: une architecture point à multipoint (PMP) obligatoire et une architecture mesh optionnelle. Chacune des architectures supportant des trames de durée comprises entre 2.5ms et 20 ms. Dans une architecture PMP, la communication entre une BS et une SS peut être soit unidirectionnelle ou bidirectionnelle d'où la notion de duplexage. Le duplexage peut être entier ou partiel selon qu'on peut envoyer et recevoir simultanément ou non. Il existe donc deux types de duplexages: temporel et fréquentiel. Le duplexage temporel(TDD: time division duplexing) signifie que l'on peut envoyer des données durant une plage de temps et recevoir durant une autre. Le duplexage fréquentiel(FDD: frequency division duplexing) c'est lorsqu'on reçoit pendant qu'on envoie des données. Le multiplexage quant à lui signifie qu'un élément du réseau peut envoyer des informations[4] à plusieurs autres éléments(ordinateurs,cellulaires,etc...) de ce réseau dans un même canal. Les multiplexages temporels et fréquentiels sont appelés(TDM et FDM[5]). Cette couche utilise les techniques de duplexage TDD et FDD(DL et UL simultanés) dans le cas d'une architecture PMP et uniquement TDD dans le cas d'une architecure Mesh.

Les figures I.5 et I.6 présentent une structure de trame PMP-TDD et Mesh. Les

[4]les termes données ou informations font références aux éléments binaires échangés
[5]voir acronymes

PDUs étant les informations échangées entre la couche physique et la couche Mac.

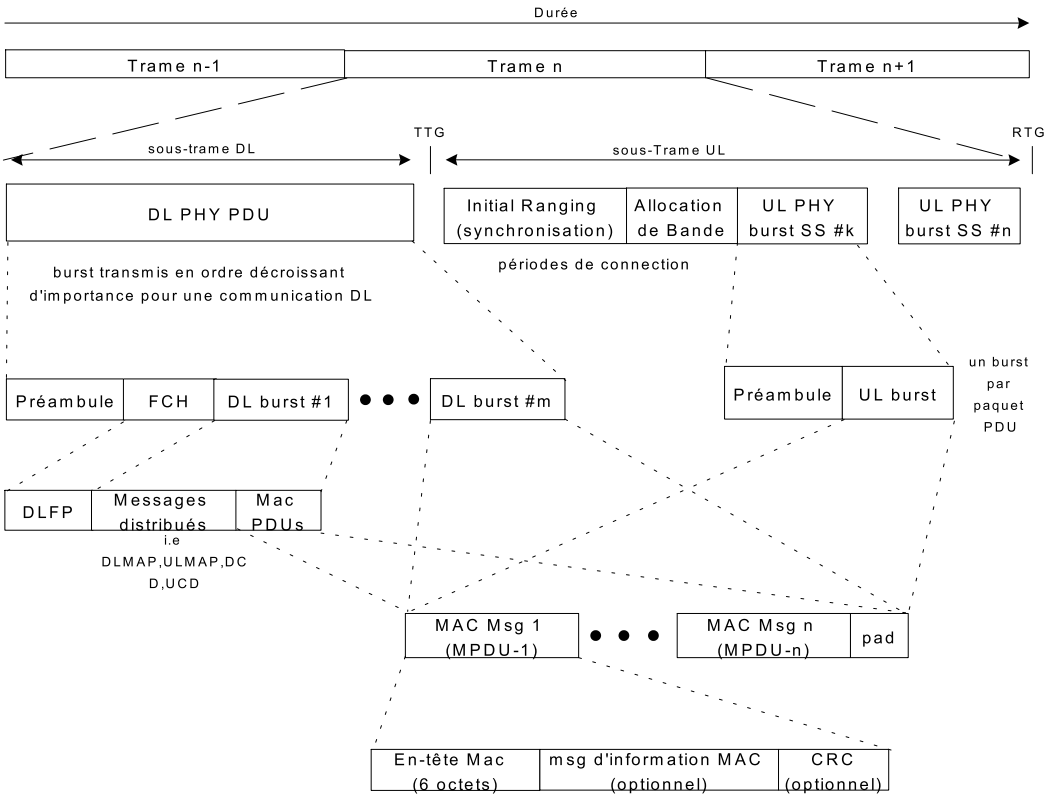


Figure I.5 Structure de trame TDD pour architecture PMP

Ainsi donc pour une architecture PMP en TDD une trame est constituée d'une sous-trame DL et d'une autre UL. La sous-trame DL(c'est à dire les informations venant de la BS) n'est constituée que d'un seul paquet PDU tandis que la sous-trame UL comprend plusieurs paquets PDU en plus des paquets d'initialisation(à la connection).Un PHY-PDU est constitué d'un ou plusieurs burst eux-mêmes comprenant un ou plusieurs symboles. Un DL-PDU comprend un préambule(rajouté au niveau de la couche physique),un burst FCH comprenant un unique symbole

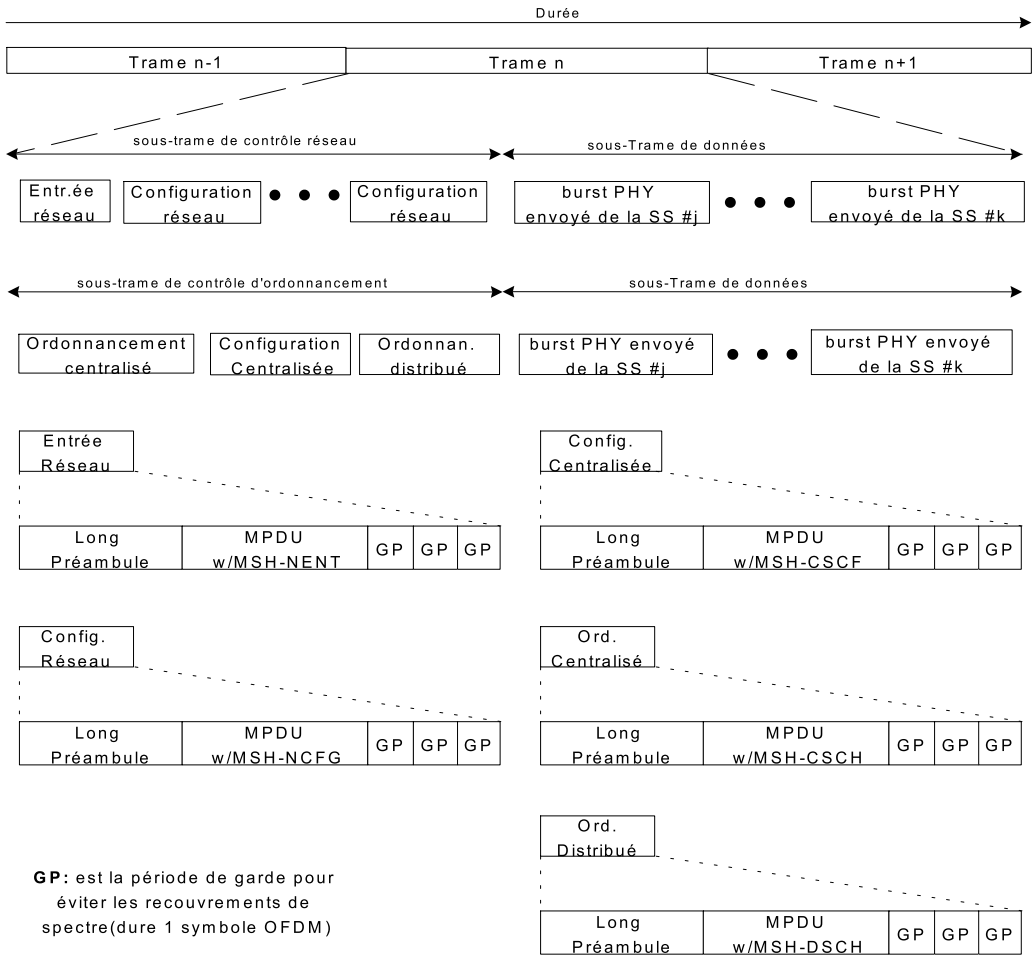


Figure I.6 Structure de trame TDD pour architecture Mesh

OFDM utilisant une modulation BPSK-1/2[6] et d'un ou plusieurs autres burst(venant de la couche MAC). Le burst FCH contient toutes les informations concernant le profil du burst(taille des burst suivants, identifiant de la station de base, numéro de la trame, nombre de changements de configuration, nombre de symboles OFDM, etc..). Les messsages tels que DL-MAP,UL-MAP,DCD,UCD contiennent les paramètres des symboles (taille des burst, DIUC/UIUC:type de burst associé à l' intervalle de temps, le début de la trame, les mesures du canal: moyenne et déviation standard du signal, etc...).

[6]voir chapitre 2

L'architecture Mesh est quant à elle constituée d'une partie contrôle et d'une partie données. La partie contrôle sert d'une part à la création et à la maintenance de la cohésion entre les différents systèmes (Contrôle réseau utilisant une modulation QPSK-1/2 [7]) et la seconde s'occupe de l'ordonnancement des transfert de données entre les différents systèmes(Ordonnancement réseau).Ces deux contrôles occupent des bandes de fréquences différentes et les données sont envoyées à la fois dans l'une et l'autre (voir figure I.6). Comme il s'agit d'un réseau distribué à travers l'air, la possibilité d'interférence est accrue ce qui fait que le Wimax prévoit l'envoi d'un long préambule afin de prévenir ces interférences.

La structure des préambules est la suivante:

P_{ALL} = 1-j,1-j,-1-j,1+j,1-j,1-j,-1+j,1-j,1-j,1-j,1+j,-1-j,1+j,1+j,-1-j,1+j,-1-j,-1-j,1-j,-1+j,1-j,1-j,-1-
j,1+j,1-j,1-j,-1+j,1-j,1-j,1-j,1+j,-1-j,1+j,1+j,-1-j,1+j,-1-j,-1-j,1-j,-1+j,1-j,1-j,-1-j,1+j,1-j,1-j,-1+j,1-
j,1-j,1-j,1+j,-1-j,1+j,1+j,-1-j,1+j,-1-j,-1-j,1-j,-1+j,1+j,1+j,1-j,-1+j,1+j,1+j,-1-j,1+j,1+j,1+j,-1+j,1-
j,-1+j,-1+j,1-j,-1+j,1-j,1-j,1+j,-1-j,-1-j,-1-j,-1+j,1-j,-1-j,-1-j,1+j,-1-j,-1-j,-1-j,1-j,-1+j,1-j,1-j,-1+j,1-
j,-1+j,-1+j,-1-j,1+j,0,-1-j,1+j,-1+j,-1+j,-1-j,1+j,1+j,1+j,-1-j,1+j,1-j,1-j,1-j,-1+j,-1+j,-1+j,-1+j,1-
j,-1-j,-1-j,-1+j,1-j,1+j,1+j,-1+j,1-j,1-j,1-j,-1+j,1-j,-1-j,-1-j,-1-j,1+j,1+j,1+j,1+j,-1-j,-1+j,-1+j,1+j,-
1-j,1-j,1-j,1+j,-1-j,-1-j,-1-j,1+j,-1-j,-1+j,-1+j,-1+j,1-j,1-j,1-j,1-j,-1+j,1+j,1+j,-1-j,1+j,-1+j,-1+j,-
1-j,1+j,1+j,1+j,-1-j,1+j,1-j,1-j,1-j,-1+j,-1+j,-1+j,-1+j,1-j,-1-j,-1-j,1-j,-1+j,-1-j,-1-j,1-j,-1+j,-1+j,-
1+j,1-j,-1+j,1+j,1+j,1+j,-1-j,-1-j,-1-j,-1-j,1+j,1-j,1-j;

P_{SUB} = 1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-1-j,1-j,1+j,1-j,1+j,1+j,-1-j,-1-j,1+j,1-j,1+j,-
1-j,1+j,1+j,1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-1-j,-1-j,1+j,1-j,1+j,1+j,-1-j,-1-j,1+j,1-
j,1+j,-1-j,1+j,1+j,1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-1-j,1-j,-1-j,-1+j,-1-j,-1-j,1+j,1+j,-
1-j,-1+j,-1-j,1+j,-1-j,-1-j,1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-1-j,1-j,-1-j,-1+j,-1-j,-1-
j,1+j,1+j,-1-j,-1+j,-1-j,1+j,-1-j,-1-j,0,1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-1-j,1-j,1+j,1-
j,1+j,1+j,-1-j,-1-j,1+j,1-j,1+j,-1-j,1+j,1+j,-1-j,-1-j,1+j,-1-j,1-j,-1-j,-1-j,-1-j,1+j,1+j,-1+j,1+j,1-j,-1-
j,-1+j,-1-j,-1-j,1+j,1+j,-1-j,-1+j,-1-j,1+j,-1-j,-1-j,-1-j,-1-j,1+j,-1-j,1-j,-1-j,-1-j,-1-j,1+j,1+j,-1+j,1+j,-
1+j,1+j,1-j,1+j,1+j,-1-j,-1-j,1+j,1-j,1+j,-1-j,1+j,1+j,1+j,1+j,-1-j,1+j,-1+j,1+j,1+j,1+j,-1-j,-1-j,1-j,-
1-j,-1-j,-1-j,-1+j,-1-j,-1-j,1+j,1+j,-1-j,-1+j,-1-j,1+j,-1-j,-1-j;

[7] voir chapitre 2

On peut donc résumer la communication DL entre une BS et une SS de la sorte: l'unité de mesure de la trame est le symbole OFDM[8]).Une trame est divisée en burst, les burst sont divisés en symboles, chaque symbole ayant un nombre fixe de bit. La couche Mac de la BS envoie des trames à sa couche physique contenant tous les paramètres et spécificités permettant l'échange d'information avec la SS. La couche physique s'occupe de la protection(codage) et la mise en forme(modulation) des MAC PDU grâce aux informations contenues dans les premiers symboles(préambule, FCH,DL-MAP,UL-MAP,DCD,UCD) et envoie les données à travers le canal(air) à l'aide d'antennes d'émission. La couche physique de la SS reçoit les symboles modulés venant de la BS par des antennes de réception et effectue les opérations inverses(démodulation,décodage) puis récupère les PDUs et les envoie à la couche Mac. La communication UL s'effectue de la même façon.

I.8 Paramètres OFDM

Les valeurs suivantes correspondent aux paramètres de transmission d'un signal à travers la couche physique OFDM de la norme IEEE-802.16. Soit BW la bande fréquentielle du canal de transmission, N_u le nombre de porteuses utilisées, n le facteur d'échantillonnage de la bande fréquentielle, G le ratio entre la période de CP et celle du symbole OFDM. On a:

- N_u égal à 200 est le nombre de porteuses constituées par les données et les pilotes.
- N_{FFT} est la plus petite puissance de 2 supérieure à N_u donc est égale à 256. C'est le nombre de points de la transformée de fourier.
- n vaut 8/7, 86/75, 144/125, 316/275 et 57/50 pour des valeurs de BW multiples

[8]voir chapitre 2

de 1.75MHz, 1.5MHZ, 1.25MHz, 2.75MHz et 2MHz respectivement et vaut 8/7 pour le reste. Il s'agit d'un facteur multiplcatif de fréquence.

- $F_s = \lfloor N * BW/8000 \rfloor * 8000$ est la fréquence d'échantillonnage.
- $T_b = 1/\Delta f$ est la période du symbole OFDM avant ajout de CP
- T_g est la durée de CP et $T_s = T_g + T_b$ est la période totale du symbole OFDM
- $G = T_g/Tb$ vaut 1/4,1/8,1/16 ou 1/32 est la durée de CP, la copie de la fin du symbole OFDM.
- T_b/N_{FFT} est la période d'échantillonnage du signal.

Le tableau I.2 représente les allocations fréquentielles en fonction du découpage en sous-canaux de la bande fréquentielle.

Tableau I.2 Tableau des paramètres des symboles OFDM

Parametres	**Valeur**
Indices des positions des fréquences de garde	-128, -127,...,-101 +101, +102, ..., +127
Indices des positions des pilotes	-88, -63, -38, -13, 13, 38, 63, 88
Indice des Sous-Canaux :	Indices des fréquences allouées :
$0b10000 : \{0b01000 : \{0b00100 : \{0b00010 : \{0b00001 : \{-100:-98,-37:-35,1:3,64:66\}$ $0b00011 : \{-97:-95,-34:-32,4:6,67:69\}\}$ $0b00110 : \{0b00101 : \{-94:-92,-31:-29,7:9,70:72\}$ $0b00111 : \{-91:-89,-28:-26,10:12,73:75\}\}\}$ $0b01100 : \{0b01010 : \{0b01001 : \{-87:-85,-50:-48,14:16,51:53\}$ $0b01011 : \{-84:-82,-47:-45,17:19,54:56\}\}$ $0b01110 : \{0b01101 : \{-81:-79,-44:-42,20:22,57:59\}$ $0b01111 : \{-78:-76,-41:-39,23:25,60:62\}\}\}\}$ $0b11000 : \{0b10100 : \{0b10010 : \{0b10001 : \{-75:-73,-12:-10,26:28,89:91\}$ $0b10011 : \{-72:-70,-9:-7,29:31,92:94\}\}$ $0b10110 : \{0b10101 : \{-69:-67,-6:-4,32:34,95:97\}$ $0b10111 : \{-66:-64,-3:-1,35:37,98:100\}\}\}$ $0b11100 : \{0b11010 : \{0b11001 : \{-62:-60,-25:-23,39:41,76:78\}$ $0b11011 : \{-59:-57,-22:-20,42:44,79:81\}\}$ $0b11110 : \{0b11101 : \{-56:-54,-19:-17,45:47,82:84\}$ $0b11111 : \{-53:-51,-16:-14,48:50,85:87\}\}\}\}$	

I.9 Seuils d'erreurs SNR

Pour s'assurer que le rapport signal à bruit SNR à la réception n'excède pas 0.5dB par rapport au rapport signal à bruit côté transmission, la norme admet des erreurs relatives de constellation selon les modulations utilisées. Le tableau I.3 présente les seuils d'erreurs admis en fonction des modulations supportées :

Tableau I.3 Seuils d'erreurs de constellation

Type de Modulation	Erreur de constellation (dB)
BPSK-1/2	-13.0
QPSK-1/2	-16.0
QPSK-3/4	-18.5
16-QAM-1/2	-21.5
16-QAM-3/4	-25.0
64-QAM-2/3	-28.5
64-QAM-3/4	-31.0

I.10 Format des vecteurs d'initalisation pour une communication descendante

Pour une communication DL (DownLink) il y a 2 vecteurs d'initialisation: le premier au début de la trame[9] et le second au début de chaque burst excepté le premier qui n'est pas re-initialisé. La figure I.7 présente le format de trame utilisé en mode DL :

MSB | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | LSB

Figure I.7 Initialisation de la trame descendante

La figure I.8 illustre une rafale DL inclut dans une trame DL:

[9] une trame est consituée de plusieurs bursts eux-mêmes constitués de symboles OFDM

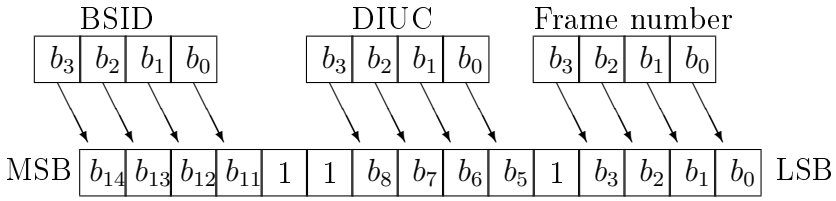


Figure I.8 Initialisation des rafales en mode descendant

I.11 Nombre d'octets obligatoire par modulation

Selon la norme IEEE 802.16, le nombre d'octets à envoyer selon la modulation est spécifié dans le tableau I.4:

Tableau I.4 Nombre d'octets requis par modulation

Modulation	Nombre d'octets avant FEC	Nombre d'octets en sortie de RS (N',K',T')	Nombre d'octets après FEC
BPSK-1/2	12	(12,12,0)	24
QPSK-1/2	24	(32,24,4)	48
QPSK-3/4	36	(40,36,2)	48
16QAM-1/2	48	(64,48,8)	96
16QAM-3/4	72	(80,72,4)	96
64QAM-2/3	96	(108,96,6)	144
64QAM-3/4	108	(120,108,6)	144

Cela assure que tous les octets envoyés sont codés et assignés sur les porteuses. Ces valeurs sont valables dans le cas où tous les 16 sous-canaux sont utilisés. Dans le cas d'un découpage en sous-canaux (1, 2, 4 ou 8), on multiplie ces valeurs par le nombre de sous-canaux puis on divise par 16.

ANNEXE II

CODES CORRECTEURS D'ERREURS

II.1 Codes systématiques

Les codes convolutifs, introduits en 1955 par Elias, peuvent être considérés comme un cas particulier des codes en bloc linéaires, mais d'un point de vue plus large la structure convolutive additionnelle munit le code linéaire de propriétés favorables qui facilitent à la fois son codage et améliorent ses performances.

Les codes systématiques forment une classe de codes convolutifs et comprennnt les éléments suivants:

- Poids de Hamming : C'est nombre de « 1 » dans un mot de code. Exemple : P(01101)= 3
  Distance de Hamming d entre deux mots de code: Nombre de positions où les deux mots de code sont différents.

- Pouvoir détecteur d'un code : un pouvoir détecteur de N signifie que toute combinaison de N erreurs ou moins dans un bloc pourra assurément être détectée. Il est égal à d_{min}.

- Pouvoir correcteur d'un code : Un pouvoir correcteur de N signifie que toute combinaison de N erreurs ou moins dans un bloc est corrigible. Il est égal à $\frac{d_{min}-1}{2}$ si d_{min} est impaire et $\frac{d_{min}}{2}-1$ sinon.

II.2 Théorie de Galois

Il s'agit de la théorie sur laquelle sont basés les codes de Reed-Solomon [?].

- Groupe, Anneau, Corps ou Champs

  Un groupe G est un ensemble d'éléments finis défini par l'opération binaire $\bullet$. Chaque paire d'éléments a et b a un seul élément c défini par l'opération binaire $\bullet$ dans G :

  $c = a \bullet b$. Ainsi on applique à un Groupe les propriétés suivantes:

  - Un groupe doit être fermé :

    $\forall$ a et $b \in$ G $\Rightarrow$ $c = a \bullet b \in$ G .

  - Associativité applicable :

    $a(bc) = (ab)c \; \forall \; a,b,c \in$ G.

  - Elément identité :

    $\forall$ $a \in$ G, il $\exists$ e élément identité de G / $a \bullet e = e \bullet a = a$.

  - *Elément inverse :*

    $\forall$ $a \in$ G, il $\exists$ a' / $a \bullet a' = e$.

  Un anneau R est un ensemble d'éléments défini par deux opérations binaires appelées *addition* et *multiplication* . On lui applique les propriétés suivantes :

  - Commutativité :

    $\forall$ $a,b \in$ R, $a+b = b+a$ *et* $a \bullet b = b \bullet a$.

  - *Associativité :*

    $\forall$ $a,b,c \in$ R, $a(bc) = (ab)c$

  - *Distributivité de la multiplication par rapport à l'addition:*

    $\forall$ $a,b,c \in$ R, (a+b)c = ac+bd.

- Intégrité :

  $\forall$ $a \in$ G, il $\exists$ e élément identité de G / $a \bullet e = e \bullet a = a$.

  $\forall$ a,$b \in$ R, $a \bullet b \Rightarrow$ *a=0 ou b=0.*

- Champs de Galois :

  Il s'agit d'une branche des mathématiques modélisant des fonctions numériques d'un alphabet donné. Les champs de Galois (*du mathématicien français Evariste Galois)* sont des ensembles d'éléments fermés sur eux-même. Un corps ou champ C est un domaine d'intégrité dans lequel tous les éléments non nuls sont inversibles : $\forall$ $a \in$ C, il $\exists$ a^{-1} *dans C /*
  $a \bullet a^{-1} = 1$. Il possède les propriétés des Groupes et Anneaux.

  - Elements des champs de Galois

    Un «champ de galois» est ensemble de nombres constitués à partir de l'élément de base α tels que :

    0,1,α,α^2,α^3,....,α^{N-1}

    En posant N=2^m-1, on forme un ensemble de 2^m éléments. Le champ est alors noté : GF(2^m).

    GF(2^m) est formé à partir du champ de base GF(2) et contient des multiples des éléments simples de GF(2).

  - Polynome Primitif

    Il s'agit du polynôme permettant de construire le champ de Galiois souhaité. En effet toutes les opérations d'addition/soustraction et de multiplication/division sont des opérations de congruence modulo ce polynôme.

    Dans le cas du Reed-Solomon on pose :

    m=8 et $P(X) = 1 + X^2 + X^3 + X^4 + X^8$

II.3 Polynône primitif

Les éléments de GF(2^8) sont dans le tableau **??**.

$\otimes$ est l'opérateur de la multiplication

$\oplus$ est l'opérateur de l'addition.

II.4 Multiplication dans GF(2^8)

Soit les polynômes A et B de GF(2^8), on pose:

$$A = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0; \quad \text{(II.1)}$$

$$B = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0; \quad \text{(II.2)}$$

$$\text{(II.3)}$$

Alors la multiplication de A et B (II.1 $\otimes$ II.2) donne :

$$C = c_7x^7 + c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x^1 + c_0; \quad \text{(II.4)}$$

Avec :

$$
\begin{cases}
c_0 = b_0 \otimes a_0 \oplus b_7 \otimes a_7 \oplus b_7 \otimes a_6 \oplus b_6 \otimes a_7 \oplus b_7 \otimes a_5 \oplus b_6 \otimes a_6 \oplus b_5 \otimes a_7 \oplus \\
b_7 \otimes a_1 \oplus b_6 \otimes a_2 \oplus b_5 \otimes a_3 \oplus b_4 \otimes a_4 \oplus b_3 \otimes a_5 \oplus b_2 \otimes a_6 \oplus b_1 \otimes a_7; \\
c_1 = b_1 \otimes a_0 \oplus b_0 \otimes a_1 \oplus b_7 \otimes a_7 \oplus b_7 \otimes a_6 \oplus b_6 \otimes a_7 \oplus b_7 \otimes a_2 \oplus b_6 \otimes a_3 \oplus \\
b_5 \otimes a_4 \oplus b_4 \otimes a_5 \otimes b_3 \otimes a_6 \oplus b_2 \otimes a_7; \\
c_2 = b_2 \otimes a_0 \oplus b_1 \otimes a_1 \oplus b_0 \otimes a_2 \oplus b_7 \otimes a_6 \oplus b_6 \otimes a_7 \oplus b_7 \otimes a_5 \oplus b_6 \otimes a_6 \oplus \\
b_5 \otimes a_7 \oplus b_7 \otimes a_3 \oplus b_6 \otimes a_4 \oplus b_5 \otimes a_5 \oplus b_4 \otimes a_6 \oplus b_3 \otimes a_7 \oplus b_7 \otimes a_1 \oplus \\
b_6 \otimes a_2 \oplus b_5 \otimes a_3 \oplus b_4 \otimes a_4 \oplus b_3 \otimes a_5 \oplus b_2 \otimes a_6 \oplus b_1 \otimes a_7; \\
c_3 = b_3 \otimes a_0 \oplus b_2 \otimes a_1 \oplus b_1 \otimes a_2 \oplus b_0 \otimes a_3 \oplus b_7 \otimes a_5 \oplus b_6 \otimes a_6 \oplus b_5 \otimes a_7 \oplus \\
b_7 \otimes a_4 \oplus b_6 \otimes a_5 \oplus b_5 \otimes a_6 \oplus b_4 \otimes a_7 \oplus b_7 \otimes a_2 \oplus b_6 \otimes a_3 \oplus b_5 \otimes a_4 \oplus \\
b_4 \otimes a_5 \oplus b_3 \otimes a_6 \oplus b_2 \otimes a_7 \oplus b_7 \otimes a_1 \oplus b_6 \otimes a_2 \oplus b_5 \otimes a_3 \oplus b_4 \otimes a_4 \oplus \\
b_3 \otimes a_5 \oplus b_2 \otimes a_6 \oplus b_1 \otimes a_7; \\
c_4 = b_4 \otimes a_0 \oplus b_3 \otimes a_1 \oplus b_2 \otimes a_2 \oplus b_1 \otimes a_3 \oplus b_0 \otimes a_4 \oplus b_7 \otimes a_7 \oplus b_7 \otimes a_3 \oplus \\
b_6 \otimes a_4 \oplus b_5 \otimes a_5 \oplus b_4 \otimes a_6 \oplus b_3 \otimes a_7 \oplus b_7 \otimes a_2 \oplus b_6 \otimes a_3 \oplus b_5 \otimes a_4 \oplus \\
b_4 \otimes a_5 \oplus b_3 \otimes a_6 \oplus b_2 \otimes a_7 \oplus b_7 \otimes a_1 \oplus b_6 \otimes a_2 \oplus b_5 \otimes a_3 \oplus b_4 \otimes a_4 \oplus \\
b_3 \otimes a_5 \oplus b_2 \otimes a_6 \oplus b_1 \otimes a_7; \\
c_5 = b_5 \otimes a_0 \oplus b_4 \otimes a_1 \oplus b_3 \otimes a_2 \oplus b_2 \otimes a_3 \oplus b_1 \otimes a_4 \oplus b_0 \otimes a_5 \oplus b_7 \otimes a_4 \oplus \\
b_6 \otimes a_5 \oplus b_5 \otimes a_6 \oplus b_4 \otimes a_7 \oplus b_7 \otimes a_3 \oplus b_6 \otimes a_4 \oplus b_5 \otimes a_5 \oplus b_4 \otimes a_6 \oplus \\
b_3 \otimes a_7 \oplus b_7 \otimes a_2 \oplus b_6 \otimes a_3 \oplus b_5 \otimes a_4 \oplus b_4 \otimes a_5 \oplus b_3 \otimes a_6 \oplus b_2 \otimes a_7; \\
c_6 = b_6 \otimes a_0 \oplus b_5 \otimes a_1 \oplus b_4 \otimes a_2 \oplus b_3 \otimes a_3 \oplus b_2 \otimes a_4 \oplus b_1 \otimes a_5 \oplus b_0 \otimes a_6 \oplus \\
b_7 \otimes a_5 \oplus b_6 \otimes a_6 \oplus b_5 \otimes a_7 \oplus b_7 \otimes a_4 \oplus b_6 \otimes a_5 \oplus b_5 \otimes a_6 \oplus b_4 \otimes a_7 \oplus \\
b_7 \otimes a_3 \oplus b_6 \otimes a_4 \oplus b_5 \otimes a_5 \oplus b_4 \otimes a_6 \oplus b_3 \otimes a_7; \\
c_7 = b_7 \otimes a_0 \oplus b_6 \otimes a_1 \oplus b_5 \otimes a_2 \oplus b_4 \otimes a_3 \oplus b_3 \otimes a_4 \oplus b_2 \otimes a_5 \oplus b_1 \otimes a_6 \oplus \\
b_0 \otimes a_7 \oplus b_7 \otimes a_6 \oplus b_6 \otimes a_7 \oplus b_7 \otimes a_5 \oplus b_6 \otimes a_6 \oplus b_5 \otimes a_7 \oplus b_7 \otimes a_4 \oplus \\
b_6 \otimes a_5 \oplus b_5 \otimes a_6 \oplus b_4 \otimes a_7;
\end{cases}
$$

Le tableau II.1 contient les éléments primitifs de GF(2^8), α: élément de base.

Tableau II.1 Eléments de GF(2^8)

Eléments	Formes Polynomiales	Formes binaires	Formes décimales
0	0	00000000	0
α^0	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00000001	1
α^1	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00000010	2
α^2	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00000100	4
α^3	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00001000	8
α^4	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00010000	16
α^5	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00100000	32
α^6	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01000000	64
α^7	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10000000	128
α^8	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00011101	29
α^9	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00111010	58
α^{10}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01110100	116
α^{11}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11101000	232
α^{12}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11001101	205
α^{13}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10000111	135
α^{14}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00010011	19
α^{15}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00100110	38
α^{16}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01001100	76
α^{17}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10011000	152
α^{18}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00101101	45
α^{19}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01011010	90
α^{20}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10110100	180
α^{21}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01110101	117
α^{22}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11101010	234
α^{23}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11001001	201
α^{24}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10001111	143
α^{25}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00000011	3
α^{26}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00000110	6
α^{27}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00001100	12
α^{28}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00011000	24
α^{29}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00110000	48
α^{30}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01100000	96
α^{31}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11000000	192
α^{32}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10011101	157
α^{33}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00100111	39

α^{34}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01001110	78
α^{35}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10011100	156
α^{36}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00100101	37
α^{37}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01001010	74
α^{38}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10010100	148
α^{39}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00110101	53
α^{40}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01101010	106
α^{41}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11010100	212
α^{42}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10110101	181
α^{43}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01110111	119
α^{44}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11101110	238
α^{45}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11000001	193
α^{46}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10011111	159
α^{47}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00100011	35
α^{48}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01000110	70
α^{49}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10001100	140
α^{50}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00000101	5
α^{51}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00001010	10
α^{52}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00010100	20
α^{53}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00101000	40
α^{54}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01010000	80
α^{55}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10100000	160
α^{56}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01011101	93
α^{57}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10111010	186
α^{58}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01101001	105
α^{59}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11010010	210
α^{60}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10111001	185
α^{61}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01101111	111
α^{62}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11011110	222
α^{63}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10100001	161
α^{64}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01011111	95
α^{65}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10111110	190
α^{66}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01100001	97
α^{67}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11000010	194
α^{78}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10011001	153
α^{69}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00101111	47
α^{70}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01011110	94
α^{71}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10111100	188
α^{72}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01100101	101

α^{73}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11001010	202
α^{74}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10001001	137
α^{75}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00001111	15
α^{76}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00011110	30
α^{77}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00111100	60
α^{78}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01111000	120
α^{79}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11110000	240
α^{80}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11111101	253
α^{81}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11100111	231
α^{82}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11010011	211
α^{83}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10111011	187
α^{84}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01101011	107
α^{85}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11010110	214
α^{86}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10110001	177
α^{87}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01111111	127
α^{88}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11111110	254
α^{89}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11100001	225
α^{90}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11011111	223
α^{91}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10100011	163
α^{92}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01011011	91
α^{93}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10110110	182
α^{94}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01110001	113
α^{95}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11100010	226
α^{96}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11011001	217
α^{97}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10101111	175
α^{98}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01000011	67
α^{99}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10000110	134
α^{100}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00010001	17
α^{101}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00100010	34
α^{102}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01000100	68
α^{103}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10001000	136
α^{104}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00001101	13
α^{105}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00011010	26
α^{106}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00110100	52
α^{107}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01101000	104
α^{108}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11010000	208
α^{109}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10111101	189
α^{110}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01100111	103
α^{111}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11001110	206
α^{112}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10000001	129

α^{113}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00011111	31
α^{114}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00111110	62
α^{115}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01111100	124
α^{116}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11111000	248
α^{117}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11101101	237
α^{118}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11000111	199
α^{119}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10010011	147
α^{120}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00111011	59
α^{121}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01110110	118
α^{122}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11101100	236
α^{123}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11000101	197
α^{124}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10010111	151
α^{125}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00110011	51
α^{126}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01100110	102
α^{127}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11001100	204
α^{128}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10000101	133
α^{129}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00010111	23
α^{130}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00101110	46
α^{131}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01011100	92
α^{132}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10111000	184
α^{133}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01101101	109
α^{134}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11011010	218
α^{135}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10101001	169
α^{136}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01001111	79
α^{137}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10011110	158
α^{138}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00100001	33
α^{139}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01000010	66
α^{140}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10000100	132
α^{141}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00010101	21
α^{142}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00101010	42
α^{143}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01010100	84
α^{144}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10101000	168
α^{145}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01001101	77
α^{146}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10011010	154
α^{147}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00101001	41
α^{148}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01010010	82
α^{149}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10100100	164
α^{150}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01010101	85
α^{151}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10101010	170
α^{152}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01001001	73

α^{153}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10010010	146
α^{154}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00111001	57
α^{155}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01110010	114
α^{156}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11100100	228
α^{157}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11010101	213
α^{158}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10110111	183
α^{159}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01110011	115
α^{160}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11100110	230
α^{161}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11010001	209
α^{162}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10111111	191
α^{163}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01100011	99
α^{164}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11000110	198
α^{165}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	10010001	145
α^{166}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00111111	63
α^{167}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01111110	126
α^{168}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11111100	252
α^{169}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11100101	229
α^{170}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11010111	215
α^{171}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10110011	179
α^{172}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01111011	123
α^{173}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	11110110	246
α^{174}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11110001	241
α^{175}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11111111	255
α^{176}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11100011	227
α^{177}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11011011	219
α^{178}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10101011	171
α^{179}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01001011	75
α^{180}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10010110	150
α^{181}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00110001	49
α^{182}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01100010	98
α^{1853}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11000100	196
α^{184}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10010101	149
α^{185}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00110111	55
α^{186}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01101110	110
α^{187}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11011100	220
α^{188}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10100101	165
α^{189}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01010111	87
α^{190}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10101110	174
α^{191}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01000001	65

α^{192}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10000010	130
α^{193}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00011001	25
α^{194}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00110010	50
α^{195}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01100100	100
α^{196}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11001000	200
α^{197}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10001101	141
α^{198}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	00000111	7
α^{199}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00001110	14
α^{200}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00011100	28
α^{201}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	00111000	56
α^{202}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01110000	112
α^{203}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11100000	224
α^{204}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11011101	221
α^{205}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	10100111	167
α^{206}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	01010011	83
α^{207}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10100110	166
α^{208}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01010001	81
α^{209}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10100010	162
α^{210}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01011001	89
α^{211}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10110010	178
α^{212}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	01111001	121
α^{213}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11110010	242
α^{214}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11111001	249
α^{215}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11101111	239
α^{216}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11000011	195
α^{217}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10011011	155
α^{218}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00101011	43
α^{219}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	01010110	86
α^{220}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	10101100	172
α^{221}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01000101	69
α^{222}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	10001010	138
α^{223}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00001001	9
α^{224}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	00010010	18
α^{225}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00100100	36
α^{226}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01001000	72
α^{227}	$1\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10010000	144
α^{228}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	00111101	61
α^{229}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	01111010	122
α^{230}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	11110100	244

α^{231}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	11110101	245
α^{232}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11110111	247
α^{233}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11110011	243
α^{234}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11111011	251
α^{235}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11101011	235
α^{236}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	11001011	203
α^{237}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10001011	139
α^{238}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00001011	11
α^{239}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00010110	22
α^{240}	$0\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	00101100	44
α^{241}	$0\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	01011000	88
α^{242}	$1\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	10110000	176
α^{243}	$0\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	01111101	125
α^{244}	$1\alpha^7+1\alpha^6+1\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0$	11111010	250
α^{245}	$1\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	11101001	233
α^{246}	$1\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	11001111	207
α^{247}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	10000011	131
α^{248}	$0\alpha^7+0\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0$	00011011	27
α^{249}	$0\alpha^7+0\alpha^6+1\alpha^5+1\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	00110110	54
α^{250}	$0\alpha^7+1\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0$	01101100	108
α^{251}	$1\alpha^7+1\alpha^6+0\alpha^5+1\alpha^4+1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0$	11011000	216
α^{252}	$1\alpha^7+0\alpha^6+1\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0$	10101101	173
α^{253}	$0\alpha^7+1\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0$	01000111	71
α^{254}	$1\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0$	10001110	142
α^{255}	$0\alpha^7+0\alpha^6+0\alpha^5+0\alpha^4+0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0$	00000001	1

II.5 constellations

Le tablau II.2 présente les constellations correspondant aux différentes modulations supportées par la norme : BPSK, QPSK, 16-QAM, 64-QAM. Ces constellations ont les facteurs de normalisation respectifs : 1 pour BPSK, $1/\sqrt{2}$ pour QPSK, $1/\sqrt{10}$ pour 16-QAM, $1/\sqrt{42}$ pour 64-QAM

Tableau II.2 Tableau des constellations BPSK,QPSK, 16QAM,64QAM

bits à moduler	modulations							
$b_5b_4b_3b_2b_1b_0$	BPSK (b_0)		QPSK (b_1b_0)		16-QAM ($b_3b_2b_1b_0$)		64-QAM ($b_5b_4b_3b_2b_1b_0$)	
constantes normalisation	c=1		c=$1/\sqrt{2}$		c=$1/\sqrt{10}$		c=$1/\sqrt{42}$	
	I	Q	I	Q	I	Q	I	Q
000 000	1	0	1	1	1	1	3	3
000 001	0	0	1	-1	1	3	3	1
000 010	x	x	-1	1	1	-1	3	5
000 011	x	x	-1	-1	1	-3	3	7
000 100	x	x	x	x	3	1	3	-3
000 101	x	x	x	x	3	3	3	-1
000 110	x	x	x	x	3	-1	3	-5
000 111	x	x	x	x	3	-3	3	-7
001 000	x	x	x	x	-1	1	1	3
001 001	x	x	x	x	-1	3	1	1
001 010	x	x	x	x	-1	-1	1	5
001 011	x	x	x	x	-1	-3	1	7
001 100	x	x	x	x	-3	1	1	-3
001 101	x	x	x	x	-3	3	1	-1
001 110	x	x	x	x	-3	-1	1	-5
001 111	x	x	x	x	-3	-3	1	-7
010 000	x	x	x	x	x	x	5	3
010 001	x	x	x	x	x	x	5	1
010 010	x	x	x	x	x	x	5	5
010 011	x	x	x	x	x	x	5	7
010 100	x	x	x	x	x	x	5	-3
010 101	x	x	x	x	x	x	5	-1

010 110	x	x	x	x	x	x	5	-5
010 111	x	x	x	x	x	x	5	-7
011 000	x	x	x	x	x	x	7	3
011 001	x	x	x	x	x	x	7	1
011 010	x	x	x	x	x	x	7	5
011 011	x	x	x	x	x	x	7	7
011 100	x	x	x	x	x	x	7	-3
011 101	x	x	x	x	x	x	7	-1
011 110	x	x	x	x	x	x	7	-5
011 111	x	x	x	x	x	x	7	-7
100 000	x	x	x	x	x	x	-3	3
100 001	x	x	x	x	x	x	-3	1
100 010	x	x	x	x	x	x	-3	5
100 011	x	x	x	x	x	x	-3	7
100 100	x	x	x	x	x	x	-3	-3
100 101	x	x	x	x	x	x	-3	-1
100 110	x	x	x	x	x	x	-3	-5
100 111	x	x	x	x	x	x	-3	-7
101 000	x	x	x	x	x	x	-1	3
101 001	x	x	x	x	x	x	-1	1
101 010	x	x	x	x	x	x	-1	5
101 011	x	x	x	x	x	x	-1	7
101 100	x	x	x	x	x	x	-1	-3
101 101	x	x	x	x	x	x	-1	-1
101 110	x	x	x	x	x	x	-1	-5
101 111	x	x	x	x	x	x	-1	-7
110 000	x	x	x	x	x	x	-5	3
110 001	x	x	x	x	x	x	-5	1
110 010	x	x	x	x	x	x	-5	5
110 011	x	x	x	x	x	x	-5	7
110 100	x	x	x	x	x	x	-5	-3
110 101	x	x	x	x	x	x	-5	-1
110 110	x	x	x	x	x	x	-5	-5
110 111	x	x	x	x	x	x	-5	-7
111 000	x	x	x	x	x	x	-7	3
111 001	x	x	x	x	x	x	-7	1
111 010	x	x	x	x	x	x	-7	5
111 011	x	x	x	x	x	x	-7	7
111 100	x	x	x	x	x	x	-7	-3
111 101	x	x	x	x	x	x	-7	-1
111 110	x	x	x	x	x	x	-7	-5
111 111	x	x	x	x	x	x	-7	-7

II.6 Durée des Trames

Les durées des trames admises sont déterminées par les périodicités des débuts des préambules. Elles sont données dans le tableau II.3.

Tableau II.3 Durée de Trames admises

Durée de Trame(en ms)	Nombre de trames par secondes
2.5	400
4	250
5	200
8	125
10	100
12.5	80
20	50

ANNEXE III

CONCEPTION DANS SPACE CODESIGN

La technologie SpaceCodesign à travers sa plateforme Space Codesign permet donc d'effectuer des applications temps réelles complètes et de bâtir des systèmes numériques synthétisables sur FPGA. Elle est basée sur l'environnemtn Eclipse et après installation des packages de SpaceCodesign dans cet environnement, on peut accéder à SpaceStudio en lançant une application.

III.1 Présentation de Space Codesign

La plateforme de SpaceStudio [?] offre diverses fonctionalités. A partir des librairies de Space elle permet de bâtir une application, de la simuler et de générer les composantes matérielles nécessaires à une implémentation FPGA (voir figure III.1).

Tout d'abord on crée un projet dans Space Codesign. Ensuite à partir du menu, l'utilisateur peut créer des modules ou des composantes(uart, mémoire, controleur, bus, microprocesseur,..). Chaque module représentant une partie de l'application à concevoir. Lorsque l'on crée un projet, un fichier «ApplicationDefinition» est créé. C'est dans ce fichier que l'on place toutes les variables communes aux différents modules de l'application:

```
#ifndef APPLICATION_DEFINITIONS_H
#define APPLICATION_DEFINITIONS_H
//Définir les variables du système à cet endroit.
#endif //APPLICATION_DEFINITIONS_H
```

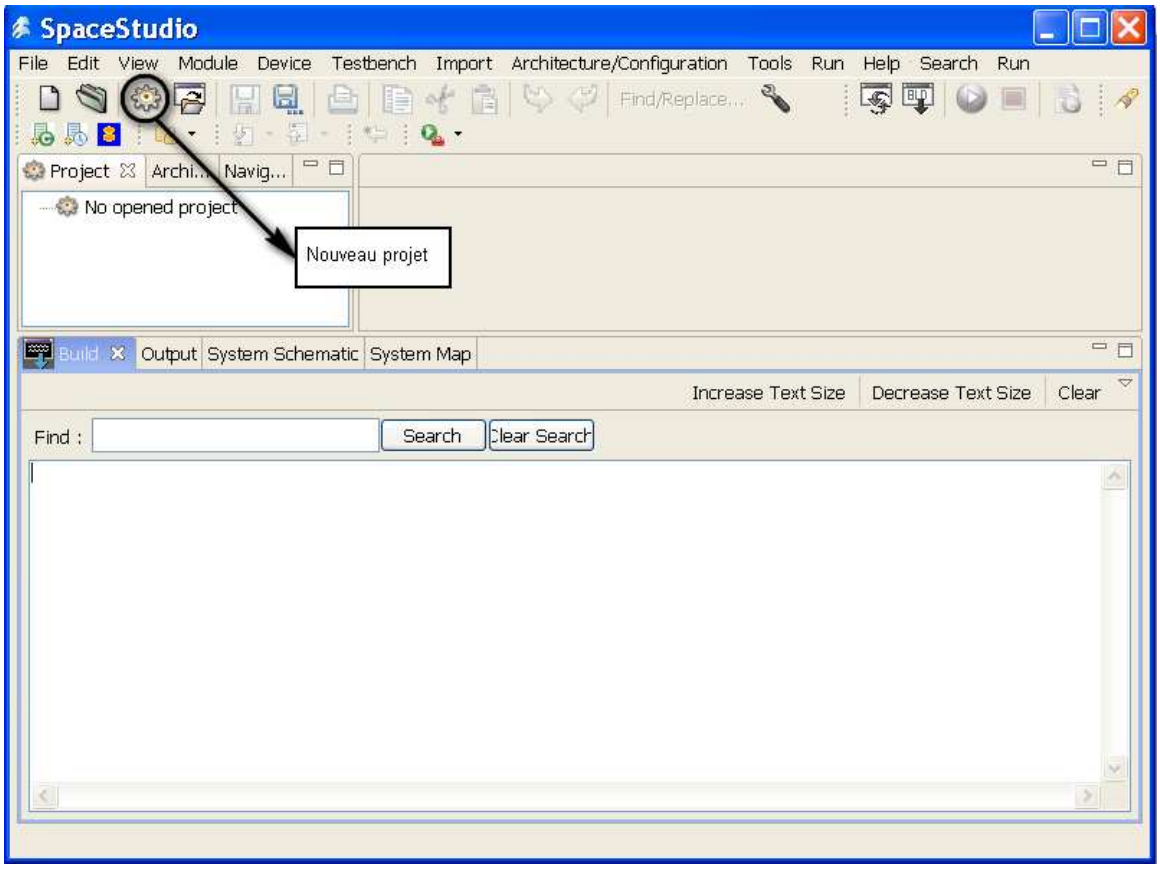


Figure III.1 Vue de la plateforme SpaceStudio

A partir de l'onglet «Module» on crée un nouveau module: Deux fichiers sont automatiquement créés: un fichier .h et un .cpp. Le fichier .h contient la déclaration de la classe SpacebaseModule qui est un template systemC d'un module Space. Cette classe renferme toutes les interfaces et ports permettant de communiquer avec un adaptateur ou un autre module(dans le cas de Lien Direct). Le fichier .cpp contient les instantiations du module défini dans le fichier.h.

```
//Fichier .h
#ifndef NOUVEAU_MODULE_H
#define NOUVEAU_MODULE_H
#include <systemc.h>
#include <SpaceBaseModule.h>
class nouveau_module : public SpaceBaseModule
{
public:
    SC_HAS_PROCESS(nouveau_module);

        // Default constructor
        nouveau_module(sc_module_name zName,double dClockPeriod,
                sc_time_unit Unit,unsigned char ucID,
                unsigned char ucPriority,bool bVerbose);
        // Thread
    void thread(void);

private:
        //Add your variables here...

};
#endif
```

```
//Fichier .cpp
#include "nouveau_module.h"

// Automatically-generated includes
#include <PlatformDefinitions.h>

// User-generated includes
#include <ApplicationDefinitions.h>

// Space library includes
#include <SpaceDisplay.h>
#include <SpaceTypes.h>

// Default constructor
nouveau_module::nouveau_module( sc_module_name zName,
        double dClockPeriod,
        sc_time_unit ClockPeriodUnit,
        unsigned char ucID,
        unsigned char ucPriority,
```

```
        bool bVerbose )
: SpaceBaseModule( zName, dClockPeriod, ClockPeriodUnit,
ucID, ucPriority, bVerbose )
{
        //This thread is sensitive to the positive edge of the clock signal
        SC_THREAD(thread);
}

void nouveau_module::thread(void)
{
        while(1)
        {
                //Add your code here...

                SpacePrint( "nouveau_module executing...\n" );
                //This wait(1) is not mandatory.
                //It should be removed when you have
                //implemented what this module should do.
                wait(1); //Remove this line
        }
}
```

Une fois l'application créée, on compile les différents fichiers puis on les exécute à partir du simulateur de SpaceCodesign(ISS). Pour cela on crée une configuration (Elix ou Simtek) à partir de l'onglet «Architecture/Configuration» puis on choisit les composantes de l'architecture à l'aide de l'option> Architecure/Configuration > Configuration Manager (voir figure III.2)

Une fois l'architecture configurée, on génère et on compile les fichiers de compilation à l'aide des options > tools > generate et > tools > build : Makefile, fichier de définition des composants, main(création/instanciation des signaux(horloge,reset),ports de communication,etc..) puis on execute avec l'onglet > Run > execute. Lors de la génération, plusieurs modes sont disponibles(release, debug, monitoring) permettant ainsi à l'utilisateur de débugger et d'observer(monitoring) les résultats de simulation de son application.

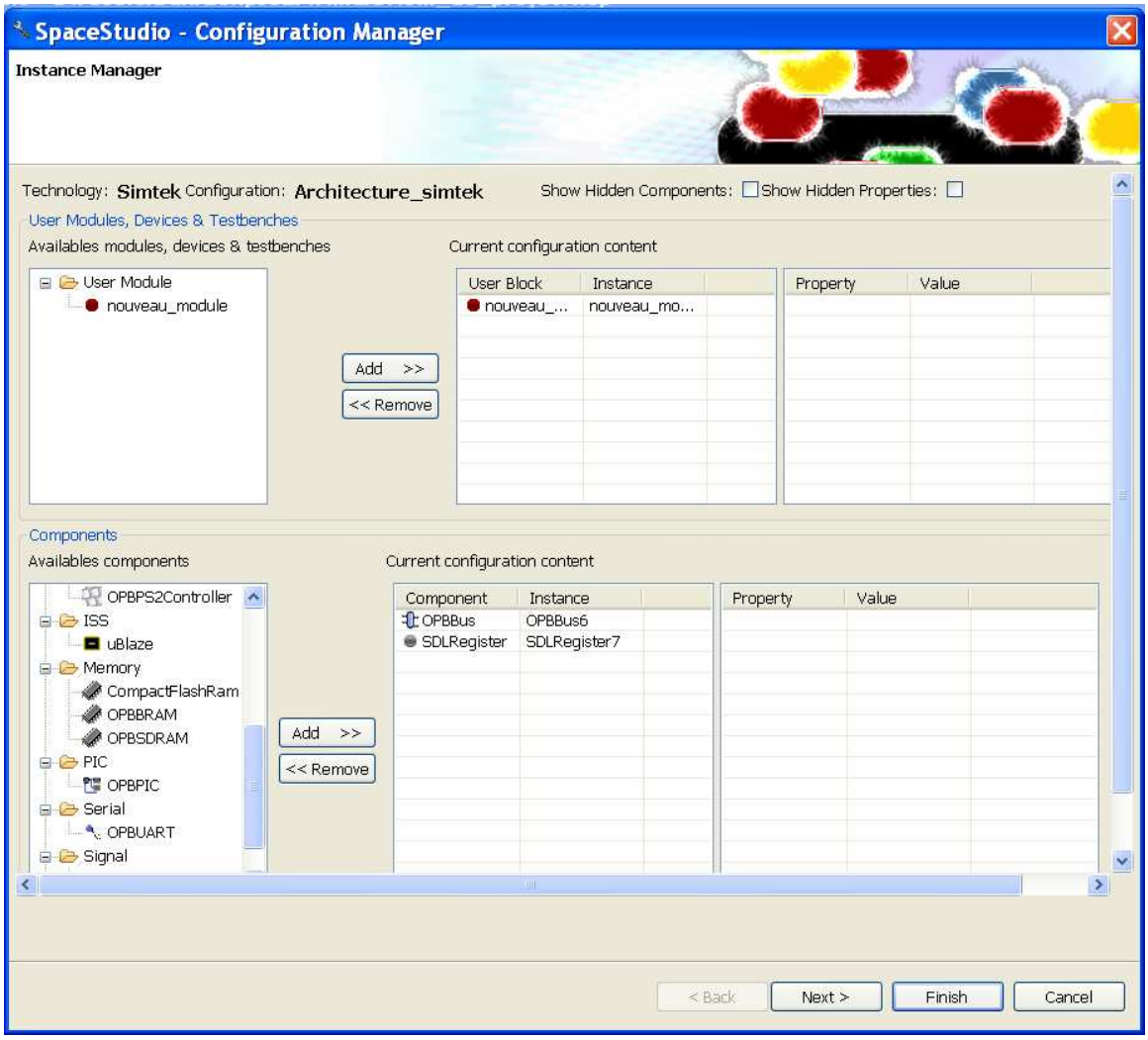


Figure III.2 Configuration de l'architecture

III.2 SpaceMonitoring

Le SpaceMonitoring [?] est un outil disponible dans Space Codesign permettant d'observer les résultats de simulation tels que le temps d'occupation du bus, le temps d'execution des différents modules et composantes du système,la distribution des tâches sur le processeur,les accès mémoire, etc..Grâce à ces résultats on peut prendre des décisions concernant l'implémentation ou la modification de l'architecture en cours. On peut accéder à cet outil d'abord en spécifiant l'option «Monitoring» lors de la génération des fichiers nécéssaires à la compilation. Ensuite dans l'onglet > tools > Preferences on spécifie les options du monitoring(analyse des statistiques des résultats, analyse temps réelle des informations reçues par socket).

Des fichiers binaires sont générés au moment de la simulation de l'application puis visualisés à partir de l'onglet > tools > SpaceMonitor de Space Codesign.

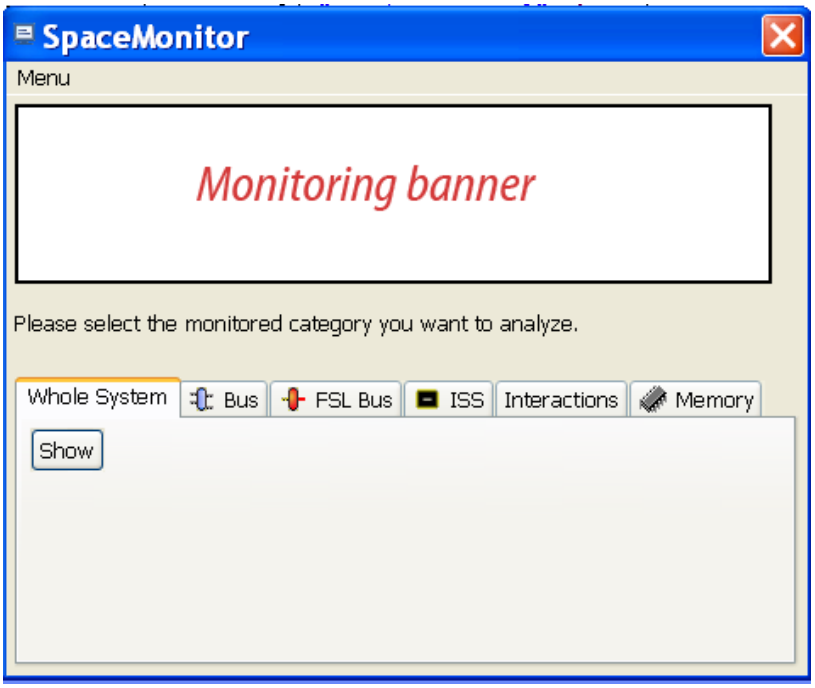


Figure III.3 Vue de SpaceMonitor

III.3 Section GenX

Une fois que les résultats de simulations satisfaisants et le choix d'une architecture établit, la génération des fichiers Xilinx se fait à l'aide de l'outil GenX de Space Codesign. En effet Xilinx est une plateforme complète de développement de systèmes numériques sur cartes FPGA. Cette option n'est accessible que pour le niveau Simtek car le niveau Elix n'est pas synthétisable. Les composantes créées par GenX sont toutes disponibles sous Xilinx EDK. Quant aux modules, GenX génère les interfaces de communication du système et c'est à l'utilisateur de remplacer les modules par leur code RTL(VHDL,Vérilog) correspondant. Pour effectuer la génération par GenX certaines opérations doivent être effectuées au moment de la configuration de l'architecture:

- s'assurer que le nombre de mémoire externes et leur taille (SDRAM,DDR) n'excède pas celui/celle disponible sur la carte FPGA sur laquelle l'applciation s'exécutera.
- le nombre d'interface externe ne doit pas excéder celui disponible sur la carte FPGA.
- la capacité de la mémoire interne instantiée dans l'architecture ne doit pas dépasser celle des BRAMs(mémoires internes) disponibles sur la carte.
- la fréquence de focntionnement du système global doit être supportées par la carte.

III.4 Bancs d'Essai

Dans cette section nous présentons les bancs d'essai utilisés pour valider notre modèle système au niveau de la plateforme Space Codesign.

Les trois premiers bancs d'essai sont ceux requis par la norme. Les autres servent à tester tous les cas de modulation.

```
****************** Banc d'essai 1**********************************
Modulation: QPSK, Taux : 3/4, Nombre de symboles OFDM dans le burst: 1,
 UIUC: 7, BSID: 1, Numéro de Trame : 1, Numero de Sous-canal: 0b10000
Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80 50
45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D
******************* Banc d'essai 2 **********************************
Modulation : 16QAM, Taux : 3/4, Nombre de symboles OFDM dans le burst: 3,
UIUC : 7, BSID : 1, Numéro de Trame : 1, Numero de Sous-canal: 0b00010
Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80 50
45 1B 9F D9 2A 88 95 EB AE B5
******************** Banc d'essai 3 **********************************
Modulation : QPSK, Taux : 3/4, Nombre de symboles OFDM dans le burst: 5,
UIUC : 7, BSID : 1, Numéro de Trame : 1, Numero de Sous-canal: 0b00001
Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87
******************************************************************************
```

```
****************** Banc d'essai 1**********************************
Modulation: BPSK, Taux : 1/2, Nombre de symboles OFDM dans le burst:16,
8,4,2,1 -UIUC:7, BSID:1, Numéro de Trame : 1, Nombres de sous-canaux:1,
2,4,8,16 - Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5
******************** Banc d'essai 2 **********************************
Modulation: QPSK, Taux: 1/2, Nombre de symboles OFDM dans le burst: 16,
8,4,2,1 -UIUC:7, BSID:1, Numéro de Trame: 1, Numero de Sous-canal:1,2,
4,8,16 -Données A transmettre: 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95
******************** Banc d'essai 3 **********************************
Modulation: QPSK, Taux: 3/4, Nombre de symboles OFDM dans le burst: 16,
8,4,2,1 -UIUC:7, BSID: 1, Numéro de Trame: 1, Numero de Sous-canal:1,2,
4,8,16 -Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D
******************************************************************************
```

```
******************** Banc d'essai 4 ********************************
Modulation:16QAM, Taux: 1/2, Nombre de symboles OFDM dans le burst: 16,
8,4,2,1 -UIUC:7, BSID: 1, Numéro de Trame: 1, Numero de Sous-canal:1,2,
4,8,16 -Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D 45 29
C4 79 AD 0F 55 28 AD 87 B5 76
******************** Banc d'essai 5 ********************************
Modulation:16QAM, Taux : 3/4, Nombre de symboles OFDM dans le burst:16,
8,4,2,1-UIUC:7, BSID: 1, Numéro de Trame: 1, Numero de Sous-canal: 1,2,
4,8,16 -Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D 45 29
C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5
2E 03 4F 09 14 69 58 0A 5D 45
******************** Banc d'essai 6 ********************************
Modulation:64QAM, Taux : 2/3, Nombre de symboles OFDM dans le burst:16,
8,4,2,1 -UIUC:7, BSID: 1, Numéro de Trame: 1, Numero de Sous-canal:1,2,
4,8,16 -Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D 45 29
C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5
2E 03 4F 09 14 69 58 0A 5D 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80
50 45 1B 9F D9 2A 88 95 EB AE
******************** Banc d'essai 7 ********************************
Modulation: 64QAM, Taux: 3/4, Nombre de symboles OFDM dans le burst: 16,
8,4,2,1-UIUC:7, BSID: 1, Numéro de Trame: 1, Numero de Sous-canal: 1,2,
4,8,16 -Données A transmettre : 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A
9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D 45 29
C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80 50 45 1B 9F D9 2A 88 95 EB AE B5
2E 03 4F 09 14 69 58 0A 5D 45 29 C4 79 AD 0F 55 28 AD 87 B5 76 1A 9C 80
50 45 1B 9F D9 2A 88 95 EB AE B5 2E 03 4F 09 14 69 58 0A 5D 45 29
**********************************************************************
```

ANNEXE IV

UTILISATION DE SIMULINK

Pour réaliser un système sous Simulink, on lance Simulink à partir de Matlab puis on crée un nouveau fichier simulink. Les librairies de Simulink sont très avancées dans les systèmes de communication. Elles disposent de tous les éléments nécessaires à la réalisation d'un réseau de télécommunication. Il existe de nombreux outils de visualisation dans Simulink que l'on peut trouver dans la librairie Simulink contenant tous les blocs de base de l'outil. Les blocs simulink communiquent à l'aide de signaux. Toutefois bien que les librairies Simulink soient bien conçues, il est parfois nécessaire d'utiliser des fonction Matlab dont les librairies sont plus élaborées et destinées à tous les domaines techniques. Ainsi des blocs spéciaux appelés S-Function permettent à un modèle Simulink d'accéder aux variables Matlab et de placer des paramètres ou signaux dans l'espace de travail Matlab(Workspace) créant ainsi une interface Matlab/Simulink.

IV.1 Simulation

Pour lancer une simulation on va à: File > New > Model. Puis on construit le système en insérant les blocs Simulink par la technique de drag and drop. Lors d'une simulation d'un modèle Simulink on peut configurer les paramètres de simulation à partir de l'onglet > simulink > Configuration Parameters. Pour tous les détails de création d'un modèle Simulink ou de fonctions Matlab, une aide précise est fournie et accessible à parir de l'onglet > Help de Matlab ou simulink (voir Figure **??**). On peut également appliquer un masque à un certain nombre de blocs possédant des paramètres communs(cette fonction sera très utile lors de l'implémentation du

modèle).

Le tableau IV.1 présente les routines de la librairie Matlab Engine, permettant à un code SystemC d'envoyer/récupérer des variables dans l'espace de travail Matlab.

Tableau IV.1 Fonctions de communication SystemC/Matlab/Simulink

Fonctions	Description
Engine *engOpen(const char *startcmd)	fonction C++ qui ouvre et ferme Matlab Engine
int eng{Get/Set}Visible(Engine *ep, bool value)	fonctions C++ qui permettent de rendre la session Engine visible
int engEvalString(Engine *ep, const char *string)	fonction C++ qui évalue des fonctions Matlab
mxArray *eng{Put/Get}Variable(Engine *ep, const char *name)	fonction C++ qui place/enlève une variable dans le Workspace Matlab
{get/set}_param('obj', 'parameter1',value1, 'parameter2', value2, ...)	fonction Matlab qui évalue les paramètres du modèle Simulink. Elle permet d'obtenir/modifier l'état de la simulation à partir du paramètre 'SimulationStatus'.
start(obj), pause(obj), continue(obj), stop(obj)	methodes qui permettent de contrôler la simulation.

IV.2 S-Functions

Les S-Functions sont compilées en fichier MEX(exécutable Matlab) et linkés dynamiquement au moment de l'exécution dans Matlab. Un template de S-Functions est fournit pour chaque langage. On peut accéder à des exemples en tapant la commande :> sfundemos dans la fenêtre Matlab et au template C/C++ en allant à :$(Matlabroot)/simulink/src/sfuntmpl_doc.c. En voici un exemple :

```
//**** Template des fonctions obligatoires d'une C/C++ S-Function
#define S_FUNCTION_NAME  sfuntmpl_basic
#define S_FUNCTION_LEVEL 2
/*
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */
#include "simstruc.h"
/*====================*
 * S-function methods *
 *====================*/

static void mdlInitializeSizes(SimStruct *S)
{
 /* fonction d'initialisation des ports d'entrée sortie (taille, type, nombre)*/
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* fonction d'initalisation du solveur c'est à dire de l'horloge*/
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
   /* fonction de calcul du module.*/
}
static void mdlTerminate(SimStruct *S)
{
 /* suppression des pointeurs*/
}
/*=============================================*
 * See sfuntmpl_doc.c for the optional S-function methods *
 *=============================================*/

/*=============================*
 * Required S-function trailer *
 *=============================*/
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

IV.3 Interfaces de cosimulation

Interface Space Studio vers Visual Studio :

```
#ifndef MATLAB_H
#define MATLAB_H

#include <systemc.h>
#include <SpaceBaseModule.h>
#include <ServerClient.h>

class Matlab : public SpaceBaseModule
{
public:

SC_HAS_PROCESS(Matlab);
// Default constructor
Matlab(          sc_module_name zName,
         double dClockPeriod,
         sc_time_unit Unit,
         unsigned char ucID,
         unsigned char ucPriority,
         bool bVerbose            );

         // Thread
     void mat_sync(void);

private:

         //Add your variables here...
         unsigned char nbsubchannels;
         long dataToSend[256];
```

```
        int nbSend,nbIfft,nbRec;
        int Num,NumRec,Read_bytes,sendRec;
        bool receivedHeader , endHeader,noRead,goIfft,go;
        bool noSend,noRec,send_param,send_data,send_fft;
        long packet,dataToRcv[256],valSend[1];

        // variable de la socket
        socketParamStruct InterfaceSpaceOut;

        //fonctions
        eSpaceStatus status;
        // fonctions de Space Studio vers Visual Studio
        void receiv_data_from_mel(); // envoi des données décodées
        void send_param_to_mel();    // reception des paramètres
        void send_data_to_mel();    // reception des données à coder
        void send_ifft_to_mat();    // envoi des données assignées
        void receiv_fft_to_mat();  // réception des données dé-modulées

        // Ouverture de la socket vers Visual Studio
        InterfaceSpaceOut.hostPort = WRITE_PORT;
        strcpy(InterfaceSpaceOut.hostName,SERVER_NAME);
if(connectToServer(&InterfaceSpaceOut)!=0){
SpacePrint("Impossible d'ouvrir la socket du cote Visual\n");
sc_stop();
}else{
SpacePrint("Connection etablie,
reception des donnees a envoyer a MAtlab\n");
        }

};

#endif
```

Interface Visual Studio vers Space Studio

```
#include "ServerClient.h"
#include <systemc.h>
#include <stdio.h>
#include <iostream>
#include <engine.h>
using namespace std;
#define SERVER_NAME "localhost"
#define MAX 256
#define NB_USERS 1


SC_MODULE(MatlabVisual)
{
        sc_in <bool> clk;
        socketParamStruct MatlabInterfaceIn;
        Engine *ep;
        bool sendparam,sendData,sendFFT;
        bool noSend,noPut,first;
        int recues,compt;
        long dataToRcv[MAX],packet,dataToSend[MAX];
        unsigned char valSM[3];
        char Nombre,Total,NombreRec;

        //fonctions de Visual Studio vers Space Studio

void envoi_data(int m); // envoi des données à coder
void envoi_param();  // envoi des parametres
void receiv_data(long packet,int m); // réception des données à évaluer
void put_ifft(); // reception des données assignées
void get_fft();  // envoi des données démodulées
```

```
//fonctions internes
void receiv_zero();
void mat_sync();

SC_CTOR(MatlabVisual){

        SC_THREAD(mat_sync);
        sensitive << clk.pos();
        MatlabInterfaceIn.hostPort = WRITE_PORT;
        strcpy_s(MatlabInterfaceIn.hostName,SERVER_NAME);
        cout<<"Creation de la socket serveur  "<< endl;
if (!(ep = engOpen("\0"))) {
fprintf(stderr, "\nImpossible d'ouvrir Matlab engine\n");
exit(-1);
}else {
        cout << "Matlab engine a ete ouverte" << endl;
}

                // fonctions de MatlabEngine
                        engSetVisible(ep,1);

                        //re-compilation des S-Functions

engEvalString(ep,"mex SpaceVersMatlabCode.cpp");
engEvalString(ep,"mex SpaceVersMatlabIFFT.cpp");
engEvalString(ep,"mex MatlabVersSpace.cpp");
engEvalString(ep,"mex MatlabVersSpaceFFT.cpp");

engEvalString(ep,"Workspace;");
engEvalString(ep,"clear all;");

//Lancement du modèle Simulink de cosimulation
engEvalString(ep,"Cosim;");
```

```
// Initialisation de la simulation
engEvalString(ep,"load IEEE80216a_init;");
//engEvalString(ep,"set_param('Cosim','SimulationCommand','stop');");

// Ouverture de la socket vers Space Studio

try{
if(connectToClient(&MatlabInterfaceIn)!=0)
        throw "Impossible d'ouvrir la socket du cote SpaceStudio";

}catch(char* str){
        cout << "Exception rencontree" << str << endl;
}
}

};
```