



Architectures reconfigurables à base d'opérateur CORDIC pour le traitement du signal: Applications aux récepteurs MIMO

Hongzhi Wang

► To cite this version:

Hongzhi Wang. Architectures reconfigurables à base d'opérateur CORDIC pour le traitement du signal: Applications aux récepteurs MIMO. Traitement du signal et de l'image. Université Rennes 1, 2009. Français. <tel-00446156>

HAL Id: tel-00446156

<https://tel.archives-ouvertes.fr/tel-00446156>

Submitted on 12 Jan 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 3903

Thèse
présentée devant
l'UNIVERSITÉ DE RENNES I
pour obtenir le grade de
Docteur de l'Université de Rennes I
Mention : *Electronique*
par
Hongzhi WANG

Équipe d'accueil : Institut d'Electronique et de Télécommunications de Rennes

École doctorale : Matisse

Composante universitaire : Université de Rennes I

**Architectures reconfigurables à base d'opérateur
CORDIC pour le traitement du signal:
Applications aux récepteurs MIMO**

Soutenue le 28 avril 2009 devant la commission d'examen

Composition du jury

Rapporteurs

M. Emmanuel Boutillon	Professeur des Universités Université Bretagne Sud
M. Lionel Torres	Professeur des Universités Polytech'Montpellier

Examinateurs

M. François Verdier	Maître de conférence Université de Cergy-Pontoise
M. Emmanuel Casseau	Professeur des Universités Université de Rennes1
M. Pierre Leray	Professeur associé SUPELEC
M. Jacques Palicot	Professeur SUPELEC

Remerciements

Je tiens tout d'abord à remercier Monsieur Pierre Leray pour m'avoir proposé ce sujet de thèse et pour les conseil, le soutien dont il m'a fait bénéficier durant ce travail, ainsi que la relecture orthographique intégrale de ce manuscrit . Je tiens à remercier Monsieur Jacques Palicot, responsable du laboratoire Signal Communication et Electronique Embarquée (SCEE) à Supélec Rennes, m'a très bien accueillie dans son équipe de recherche et la confiance qu'il a su m'accorder et son implication tout au long de cette thèse.

Je tiens à remercier l'ensemble des membres de mon jury avoir accepté de juger et évaluer ces travaux de thèse. Je remercie tout particulièrement Monsieur Emmanuel Boutillon, Professeur des Universités à l'Université Bretagne Sud, aussi que Monsieur Lionel Torres, Professeur des Universités à Polytech'Montpellier qui ont accepté d'être les rapporteurs de cette thèse. Je remercie également Monsieur François Verdier, Maître de conférence à l'Université de Cergy-Pontoise, ainsi que Monsieur Emmanuel Casseau, Professeur des Universités à l'Université de Rennes1 pour avoir accepté d'être membre de ce jury.

Je remercie aussi Christophe Moy pour m'avoir guidé et aidé sur le projet Smart Radio Challenge de SDR Forum, ainsi tous les membres de l'équipe MENHIR.

Je remercie l'ensemble des membres de l'équipe SCEE pour leur bonne humeur au quotidien et à la bonne ambiance générale au sein de l'équipe. Ainsi, j'aimerais remercier paticulièrement Steredenn pour sa disponibilité, les échanges d'idées pendant la rédaction de cette thèse. A Loig, Juien, Amor, avec qui j'ai eu la joie de travailler sur de nombreux aspects des architectures reconfigurables.

Ces remerciements s'adressent également à mes parents, ma famme et mes amis qui m'ont supporté, encouragé durant ces trois années de thèse.

Table des matières

Table des matières	iii
Introduction	1
1 Radio logicielle	5
1.1 Définitions et généralités	6
1.2 La paramétrisation dans le contexte de la radio logicielle	8
1.2.1 Fonctions communes	8
1.2.2 Opérateurs communs	8
1.3 Contraintes et besoins pour les architectures radio logicielle	9
1.4 Conclusion	9
2 Système MIMO	11
2.1 Introduction	12
2.2 Modèle de canal MIMO à évanouissements	12
2.3 Capacité des canaux MIMO	13
2.4 Présentation du système MIMO	14
2.4.1 Système MIMO à base de codes spatio-temporels	15
2.4.1.1 Codage spatio-temporel en treillis	15
2.4.1.2 Codage spatio-temporel par blocs	16
2.4.2 Système MIMO à base de Multiplexage spatial	17
2.4.2.1 D-BLAST	17
2.4.2.2 V-BLAST	18
2.4.2.3 H-BLAST	19
2.4.2.4 Turbo-BLAST	20
2.5 Algorithmes de réception MIMO : matrice du canal H connue	20
2.5.1 Algorithmes adaptés au multiplexage spatial	20
2.5.2 Maximum de Vraisemblance	21
2.5.3 Branch Bound	22
2.5.4 Décodage par sphères	22
2.5.5 Récepteur linéaire du Forçage à Zéro (ZF)	22
2.5.6 Récepteur linéaire MMSE	22
2.5.7 Récepteur à retour de décision V-BLAST	23
2.5.8 V-BLAST Square Root	23
2.5.9 Structure SIC de type QR	23
2.5.10 Structure SIC de type GDFE	24

2.5.11	Comparaison en terme de performance/complexité	24
2.6	Algorithmes de réception MIMO : matrice du canal H inconnue	27
2.6.1	La séparation aveugle de sources	27
2.6.2	Le critère CM	28
2.6.2.1	L'algorithme CMA à base de gradient stochastique (SG-CMA)	29
2.7	Description de l'algorithme "V-BLAST Square Root"	30
2.7.1	Principe du récepteur V-BLAST	30
2.7.2	Base de l'algorithme "V-BLAST Square Root"	31
2.7.3	Algorithme "V-BLAST Square Root"	32
2.7.4	Optimisation de l'algorithme "V-BLAST Square Root"	34
2.8	Description fonctionnelle de l'algorithme "V-BLAST Square Root"	35
2.9	Conclusion	36
3	Architecture reconfigurable à base d'opérateur CORDIC	37
3.1	Introduction	38
3.2	Opérateur CORDIC	39
3.2.1	Algorithme CORDIC	39
3.2.1.1	Principe de l'algorithme CORDIC	40
3.2.1.2	Les différents modes de fonctionnement de l'algorithme	43
3.2.1.3	Mode vecteur	43
3.2.1.4	Mode rotation	44
3.2.2	Architecture de l'opérateur CORDIC	44
3.2.2.1	Architecture à opération série et itération série	45
3.2.2.2	Architecture à opération parallèle et itération série	45
3.2.2.3	Architecture à opération parallèle et itération parallèle	45
3.2.2.4	Autre architectures et améliorations	46
3.2.3	Applications	47
3.2.3.1	Modulation numérique	47
3.2.3.2	Transformée de Fourier Discrète	47
3.2.3.3	Décomposition en valeur singulière (SVD)	48
3.2.3.4	Algorithme des moindres carrés récursif	48
3.2.3.5	Rotations de Givens	49
3.3	Proposition d'architecture CBDRA	49
3.3.1	Contrainte de notre problème	49
3.3.2	Idée de base	50
3.3.3	Relation théorique entre le nombre d'opérateur CORDIC et les caractéristiques des applications	50
3.3.4	Architecture CBDRA	51
3.4	Architecture reconfigurable CBDRA	52
3.4.1	Structure de FPGA	53
3.4.2	Reconfiguration dynamique de FPGA	54
3.4.3	Flots de conception du FPGA	56
3.4.4	Connexion entre la partie statique et la partie reconfigurable	59
3.4.5	Gestion de reconfiguration	60
3.4.6	Architecture CBDRA et reconfiguration statique	60

3.4.7	Architecture CBDRA et reconfiguration dynamique des interconnexions	61
3.5	Application de l'architecture CBDRA pour un décodeur MIMO V-BLAST Square Root	62
3.5.1	Stationnarité du canal	63
3.5.2	Implémentation parallèle de CORDIC (Un exemple : calcul de $P^{1/2}$ et de Q_α)	63
3.5.3	Relation théorique entre le débit et le nombre d'opérateurs CORDIC utilisé	66
3.5.4	Exemple de résolution des équations et d'implémentation : calcul de $P^{1/2}$ et de Q_α	68
3.5.5	Utilisation de l'architecture CBDRA avec la reconfiguration dynamique	70
3.5.6	Implémentation de l'ensemble de l'algorithme	72
3.5.7	Simulation	74
3.6	Architecture CBDRA pour d'autres algorithmes MIMO	77
3.6.1	L'algorithme MIMO MMSE	77
3.6.1.1	Simulation	78
3.6.2	L'algorithme CMA	78
3.6.2.1	Simulation	80
3.7	Conclusion	81
4	Implantation d'algorithmes MIMO	85
4.1	Plate-forme Radio Logicielle hétérogène	85
4.1.1	Le fonctionnement d'un composant de traitement	87
4.1.2	Gestion de reconfiguration sur FPGA	88
4.2	Implantation de l'algorithme "V-BLAST Square Root"	88
4.2.1	Réalisation matérielle de l'opérateur CORDIC	88
4.2.2	Architecture générale du décodeur "V-BLAST Square Root"	91
4.2.3	Le module de décomposition QR	91
4.2.4	Le module de l'ordonnancement	93
4.2.5	Le module de calcul des vecteurs "nulling"	94
4.2.6	Le module de décodage des données	96
4.2.7	Résultats de implémentation de l'ensemble de l'algorithme	98
4.2.8	Test du décodeur	99
4.3	Conclusion	100
Conclusion		103
Annexe		107
A Transformée de Fourier rapide (Fast Fourier transform - FFT)		109
B Décomposition en valeur singulière (SVD)		111

C Smart Radio Challenge	119
C.1 Description de la plate-forme de prototypage	119
C.2 Description du système	121
C.3 Réalisation numérique de modulations analogique	121
C.3.1 Modulation AM	121
C.3.2 Démodulation AM	122
C.3.3 Modulation FM	122
C.3.4 Démodulation FM	123
C.4 Chaînes d'émission et de réception modulaires	123
C.4.1 Modulation AM	123
C.4.2 Démodulation AM	124
C.4.3 Modulation FM	125
C.4.4 Démodulation FM	125
C.5 Intégration à l'outil SynDEX	126
C.6 Utilisation de la reconfiguration dynamique	127
D Simulation Modelsim des modules de l'algorithme SRA	131
E Utilisation de l'opérateur CORDIC dans les modules M_4, M_5, M_6 de l'algorithme SRA	135
E.0.1 Le module de l'ordonnancement	135
E.1 Le module de calcul des vecteurs "nulling"	135
E.2 Le module de décodage des données	137
Table des figures	143
Liste des tableaux	147
Bibliographie	149

Liste des acronymes et abréviations

ASIC	Application Specific Integrated Circuit
BPSK	Binary Phase Shift Keying
CAN	Convertisseur Analogique Numérique
CLB	Configurable Logic Bloc
CDMA	Code Division Multiple Access
CMA	Constant Module Algorithm
CNA	Convertisseur Numérique Analogique
CORDIC	Coordinate Rotation Digital Computer
CRC	Cyclic Redundancy Check
DECT	Digital Enhanced Cordless Telephone
DFE	Decision Feedback Equalizer
DMR	Digital Modular Radio
DSP	Digital Signal Processor
DVB	Digital Video Broadcasting
DFT	Discrete Fourier Transform
EDGE	Enhanced Data rate for GSM Evolution
EQM	Erreur Quadratique Moyenne ou en anglais Mean Square Error MSE
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Arrays
GDFE	Generalised Feedback Decision Equalizer
GPP	General Purpose Processor
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
ICA	Independent Component Analysis
IF	Intermediate Frequency ou Fréquence Intermédiaire
IOB	Input Output Bloc
IP	Intellect Property
JTRS	Joint Tactical Radio System
LUT	Look Up Table
MAQ	Modulation d'Amplitude en Quadrature
MC-CDMA	Multi-Carriers Code Division Multiple Access
MENHIR	Multi-standard ENHanced Interoperable Radio
MDP	Modulation par Déplacement de Phase
MIMO	Multiple-Input Multiple-Output
MISO	Multiple Input Single Output
ML	Maximum d'Likelihood
MMSE	Minimum Mean Square Error

MUK	MULTiuser Kurtosis maximization
OFDM	Orthogonal Frequency Division Multiplexing
OSIC	Ordered Successif Interference Cancellation
O-STBC	Orthogonal Space-Time Block Coding
PMCS	Programmable Modular Communications System
QPSK	Quaternary Phase Shift Keying
RF	Radio Frequency ou Radio Fréquence en français
RSB	Rapport Signal à Bruit
RS	Reed-Solomon
SNR	Signal Noise Ratio
SDR	Software Difined Radio
SWR	SoftWare Radio
SIC	Successif Interference Cancellation
SISO	Single Input Single Output
SIMO	Single Input Multiple Output
SoPC	System on Programmable Chip
STC	Space-Time Coding
STTC	Space-Time Trellis Coding
STBC	Space-Time Block Coding
SVD	Singular Value Decomposition
TEB	Taux d'Erreur Binaire
TNS	Traitemet Numerical System
UMTS	Universal Mobile Telecommunications
V-BLAST	Vertical Bell Labs Layered Space-Time)
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WLAN	Wireless Local Area Networks
ZF	Zero Forcing

Introduction

Les systèmes numériques prennent une place de plus en plus importante dans le domaine plus large de l'électronique. Bien qu'au début de leur apparition dans les années 50, la puissance de calcul et la capacité de mémorisation étaient limitées. Mais les évolutions technologiques survenues ont accéléré cette tendance.

Les concepteurs ont longtemps disposé de deux solutions : mettre en oeuvre un système programmé via un microprocesseur ou alors se lancer dans la réalisation complexe d'un circuit spécifique (ASIC, Application Specific Integrated Circuit). L'apparition des ASIC structurés est à mettre en relation avec l'évolution importante des circuits logiques programmables.

Depuis la mise sur le marché du premier ASIC, la complexité des circuits intégrés n'a pas cessé de croître. L'intégration progressive des composants électroniques dans les appareils de consommation grand public a permis des développements de solution architecturales de plus en plus souples et performantes. Cependant, depuis quelques années une autre possibilité s'offre au concepteurs, les circuits reconfigurables.

Parallèlement la diversité et la complexité des applications de communication numérique ne cessent de croître avec l'évolution perpétuelle du marché des télécommunications. De plus, il faut assurer la compatibilité avec les normes les plus anciennes. Dans les réseaux de communication actuels, les traitements critiques et réguliers sont exécutés sur des accélérateurs matériels dédiés (ASIC). Les calculs de faible complexité sont assurés par des processeurs spécialisés dans la mise en oeuvre d'applications de traitement du signal (DSP). Mais pour les télécommunications de future génération, la diversité des traitements implantés dans les accélérateurs matériels impose l'intégration d'un très grand nombre de blocs dédiés et conduit à un taux d'utilisation extrêmement réduit de ces ressources.

En effet, l'utilisation de différentes technologies pour supporter divers standards de communication mobile est un des défis des réseaux de télécommunications actuels. Ils s'appuient essentiellement sur les technologies matérielles, dont les mises à jour et le temps de déploiement sont toujours élevés. La multiplication des standards de communication (WIFI, WIMAX, DVB 2, téléphone 3G, 4G...) nécessite d'avoir des plate-formes matérielles flexibles. L'introduction de la technologie radio logicielle représente un changement important pour l'industrie de la radio transmission. La radio logicielle peut servir à remplacer tout système à radio fréquence existante et devrait permettre de réduire les coûts de développement.

En 1995, J.Mitola a défini dans [1] les principaux concepts de la radio logicielle. Elle a pour objectif de résoudre l'existence simultanée de protocoles multiples en proposant des architectures reconfigurables dynamiquement pour les terminaux dans les environnements radio sans fil. Cette architecture est constituée d'une antenne large bande directement suivie par un convertisseur analogique numérique à haute fréquence d'échantillonnage et large

bande permettant à un circuit de type processeur généraliste de réaliser les fonctions radio de divers standards et des interfaces associées. Un des avantages de la Radio logicielle est la capacité d'ajouter, de mettre à jour et de renforcer les performances des fonctionnalités du système radio par logiciel grâce à la reprogrammabilité des processeurs sans ajouter les coûts de changement du matériel.

Toutefois, de nombreuses limitations technologiques tendent à repousser la maturité de ce concept, dont le manque de puissance de calcul, la forte consommation des processeurs, les limites de performances des convertisseurs et la limitation en terme de bande passante des amplificateurs, mélangeurs ainsi que des antennes. Face aux contraintes technologiques et à la diversité des traitements à exécuter, l'adoption d'architectures pragmatiques semble être la seule solution possible [2]. La différence est donc faite entre la radio logicielle "idéale" et la radio logicielle "restreinte" (SDR, Software Defined Radio). Dans ce cas, les architectures reconfigurable dynamiquement apportent donc une réponse pour relever le défi de la multiplicité des standards de communication.

Parmi les architectures reconfigurables, l'émergence de la technologie FPGA laisse la place à une solution intermédiaire où on allie flexibilité de la programmation et puissance de calcul des architectures spécialisées. Les FPGA de larges capacités offrent la possibilité de concevoir des architectures hétérogènes SoPC intégrant des processeurs embarqués, de large capacité mémoire, des accélérateurs dédiés offrant les différents types de flexibilité : logicielle, matérielle, à la conception ou à l'exécution. Cette grande flexibilité de conception permet de répondre aux besoins des applications radio logicielle tout en offrant des capacités de calculs élevées. Le compromis performance/flexibilité apporté par les FPGA en fait donc un bon candidat pour les applications radio logicielle par rapport aux ASIC et DSP [3]. Nous nous sommes particulièrement intéressés à ce type de circuits et leur mise en oeuvre dans les applications radio logicielle.

La paramétrisation est une des techniques clés dans la radio logicielle. L'objectif général des études de paramétrisation est de mettre en évidence dans un premier temps, les traitements communs ("Common Processing") réalisés dans les chaînes de transmission de différents standards. On peut distinguer deux approches en paramétrisation : l'approche fonction commune et l'approche opérateur commun. Nous montrerons en détails ces approches dans le chapitre 1. Parmi les opérateurs communs les plus utilisés, nous nous intéressons particulièrement à l'opérateur CORDIC. Il fait l'objet d'un regain d'intérêt depuis plusieurs années car il peut être considéré comme un opérateur générique pouvant être utilisé dans un grand nombre d'algorithme classiques de traitement du signal. Nous montrons l'intérêt de ce choix en détail dans le chapitre 3.

L'émergence de techniques de réception en diversité (Space-Time, MIMO) ouvre de nouvelles perspectives pour approcher la capacité maximale du canal.

Les systèmes multi-antennes à l'émission et à la réception (Multi input multi output MIMO) permettent théoriquement d'accroître la capacité des liens de communications sans fil par rapport aux systèmes composés d'une seule antenne à l'émission et à la réception (Single input Single output SISO). En faisant l'hypothèse que les trajets entre chaque antenne d'émission et de réception sont indépendants, Foshini [4] et Telatar [5] ont démontré que la capacité théorique du canal MIMO croît linéairement avec le nombre d'antennes. Les systèmes MIMO sont l'un des principaux axes de développement pour augmenter les débits des communications sans fil et bien que les premiers travaux publiés sur ce sujet ne datent que de quelques années, nous assistons à un très rapide développement de

cette technologie avec des applications déjà envisagées dans les réseaux locaux sans fil et les réseaux de communication de 3ème génération. Les systèmes MIMO sont par exemple proposés pour le futur standard de réseau local sans fil IEEE 802.11n où l'objectif est d'atteindre des débits de 100 mégabits par seconde pour les applications vidéo. Les systèmes MIMO seront donc présents dans le futur système radio logicielle. Nous nous intéressons donc particulièrement à ces systèmes. Nous montrerons les possibilités de les réaliser pour différents cas d'utilisation par mise en oeuvre d'architectures reconfigurables.

Problématique de l'étude

La problématique de l'étude présentée dans ce document est la suivante :

1. Face à la multitude des standards de communication, est-il possible d'offrir une architecture reconfigurable à base d'opérateur commun CORDIC pour un système MIMO en supportant différents nombres d'antennes, différents types de modulations et de propagation ?
2. Afin d'optimiser les ressources matérielles, est-il possible de concevoir une architecture reconfigurable dynamiquement en exploitant la reconfigurabilité offerte par une plate-forme hétérogène ?
3. Cette architecture peut-elle être utilisée pour les différents blocs d'algorithmes MIMO ?
Plus largement, peut-elle être appliquée à d'autres algorithmes de traitement du signal ?

Concernant la reconfiguration dynamique dans l'implémentation d'algorithmes de réception MIMO, il existe peu de travaux publiés actuellement sur ce sujet. Dans le même temps, le domaine des architectures reconfigurables évoluent rapidement. Parmi les différentes architectures reconfigurables, nous nous appuyons essentiellement sur la technologie FPGA pour répondre à notre problème de reconfiguration. Ces circuits sont largement utilisés en traitement du signal grâce à la possibilité de reconfiguration et au haut niveau de parallélisme.

Le premier objectif de cette thèse est donc de concevoir une architecture reconfigurable pour les systèmes MIMO sur FPGA. Nous cherchons ensuite à optimiser les ressources matérielles par la reconfiguration dynamique. Nous étendrons cette architecture vers d'autres algorithmes du traitement du signal.

Contexte de l'étude

Les travaux présentés dans ce document ont été réalisés au sein de l'équipe de recherche SCEE à Supélec campus de Rennes. L'équipe de recherche s'intéresse au domaine de la radio logicielle et de la radio intelligente à travers l'étude des systèmes de traitement du signal et des systèmes de communication numérique.

Plan du mémoire

Ce document se décompose en quatre chapitres qui se répartissent comme suit :

Le **premier chapitre** donne au lecteur un aperçu du domaine de la radio logicielle, permettant de situer les motivations de notre démarche. Le concept de la radio logicielle, quelques architectures de l'état de l'art et les techniques de la paramétrisation seront présentés dans ce chapitre.

Le **second chapitre** présente brièvement un état de l'art des systèmes MIMO. Nous décrivons les différents algorithmes de réception et illustrons notre choix de l'algorithme V-BLAST Square Root en terme de compromis Performances/Complexité. Nous détaillerons l'algorithme " V-BLAST Square Root" que nous avons retenu et analysons l'architecture fonctionnelle de l'algorithme. Il faut noter que c'est une thèse en électronique. Nous avons donc comparé les algorithmes les plus connus, sans aller jusqu'à une recherche exhaustive des algorithmes MIMO.

Le **troisième chapitre** présente une proposition d'architecture reconfigurable à base d'opérateur CORDIC, nommé CBDRA(CORDIC Based Dynamically Reconfigurable Architecture) pour les systèmes MIMO. Nous décrivons d'abord le principe de l'algorithme CORDIC et choisisrons l'architecture la plus adaptée pour notre application parmi les différentes architctures de l'opérateur CORDIC. Ensuite, nous montrons l'intérêt de considérer CORDIC comme un opérateur commun en prenant des exemples de son application pour le traitement du signal. Puis nous analysons la réalisation de l'architecture reconfigurable en utilisant différents nombres d'opérateur CORDIC pour le décodeur MIMO V-BLAST Square Root. Nous analysons également l'application de l'architecture à d'autres algorithmes de décodage MIMO tels que les algorithmes MMSE et CMA.

Le **quatrième chapitre** est consacré à la mise en oeuvre d'une plate-forme matérielle hétérogène et à une analyse de l'application de notre architecture CBDRA pour l'implémentation de l'algorithme MIMO V-BLAST Square Root. Puis nous présentons les résultats de la synthèse des différentes architectures sur FPGA.

Finalement la conclusion propose une synthèse des travaux présentés pour amener le lecteur à une discussion sur les perspectives de ces travaux. Les annexes concernent les rappels sur les autres algorithmes réalisés par l'opérateur CORDIC et présentent les projets nationaux et européen auquels nous avons participé durant cette thèse et notamment le projet MEHNIR pour le Smart Radio Challenge du SDR Forum.

Chapitre 1

Radio logicielle

Sommaire

1.1	Définitions et généralités	6
1.2	La paramétrisation dans le contexte de la radio logicielle	8
1.2.1	Fonctions communes	8
1.2.2	Opérateurs communs	8
1.3	Contraintes et besoins pour les architectures radio logicielle	9
1.4	Conclusion	9

L'approche radio logicielle a pour idéal la numérisation du signal radio immédiatement après l'antenne de réception par conversion A/N⁽¹⁾, et réciproquement une conversion N/A⁽²⁾ avant l'antenne à l'émission. Comme nous l'avons évoqué dans l'introduction générale, l'architecture radio logicielle "idéale" (Software Radio- SWR) nécessite un circuit de type processeur généraliste exécutant toutes les fonctions radio. Les traitements depuis les fonctions radio fréquence jusqu'aux couches applicatives en passant par les traitements en bande de base sont réalisés de façon numérique. Encore aujourd'hui, de nombreuses limitations technologiques ne rendent pas envisageable ce concept, notamment la performance des convertisseurs A/N et N/A. Le manque de puissance de calcul des circuits numériques actuels ne permet pas d'atteindre les capacités de calcul requises pour effectuer les traitements dans le domaine radio fréquence pour des standards déjà existants tels que UMTS et GSM/EDGE [6]. Face aux contraintes technologiques et à la diversité des traitements à exécuter, un nouveau concept nommé radio logicielle "restreinte" (Software Defined Radio, SDR) est apparu. Dans ce concept, les récepteurs s'appuient sur des architectures matérielles étagées en plusieurs segments technologiques. De façon schématique, un émetteur/récepteur radio contient un segment analogique pour la partie RF et FI, des circuits numériques pour les traitements de bande de base et enfin des solutions logicielles pour les couches applicatives. Chaque étage suit une évolution tendant à remplacer progressivement les circuits spécialisés par des composants plus généralistes et programmables. La numérisation se rapproche de l'antenne et du front-end. L'étage analogique d'entrée se réduit. Les circuits ASIC de l'étage numérique vont être remplacés par des circuits programmables (par exemple FPGA) et les formes d'ondes deviennent des applications logicielles.

⁽¹⁾Analogique/Numérique

⁽²⁾Numérique/Analogique

Les premiers projets sont à l'initiative du domaine militaire américain. Le projet SpeakEasy phase I de 1992 à 1995 avait pour objectif d'offrir l'interopérabilité entre les différents systèmes de communications des armées américaines. La phase II de SpeakEasy [7] a montré une architecture PMCS (Programmable Modular Communications System) comme une architecture programmable. Il est à noter que la technologie FPGA a été utilisée pour les traitements numériques des radiocommunications dans ce projet. Ensuite, un vaste programme radio logicielle a été lancé par le DoD américain et une architecture système SDR de référence est donnée par ce programme en 1998. Vanu G. Bose et al. [8] proposent une première implantation radio logicielle sur PC, d'un système de radio communication FSK et du standard de radiocommunication GSM [9]. Dans l'industrie, Motorola, un des principaux participants du projet JTRS, propose une architecture nommée Digital Modular Radio (DMR).

L'évolution de la radio logicielle amène à considérer des applications multi-standards nécessitant de traiter de multiples couches physiques sur la même plate-forme afin d'offrir différents types de services. Les importantes capacités de calcul requises par les traitements en bande de base rendent indispensables l'utilisation de circuits divers afin d'optimiser l'efficacité de calcul. Les architectures d'exécution sont donc hétérogènes pour répondre à la diversité de ces traitements. Ces architectures doivent être flexibles afin d'éviter de multiplier le nombre de circuits d'un système embarqué.

1.1 Définitions et généralités

La figure 1.1 montre un récepteur de type superhétérodyne dans une approche classique, utilisant un ensemble de composants spécialisés à un type de traitement. Les chaînes de transmission permettant l'accès aux standards existants sont conçues à base de composants ASIC afin d'obtenir le maximum de performance pour le minimum de consommation électrique. Ce type de concept n'est pas évolutif et n'autorise pas la prise en compte de nouveaux standards. Le support de plusieurs standards se fait par duplication physique des récepteurs. Ceci nécessite le changement d'équipement à chaque saut technologique pour l'utilisateur et la mise à jour d'équipement au sein des stations de base pour les fournisseurs.

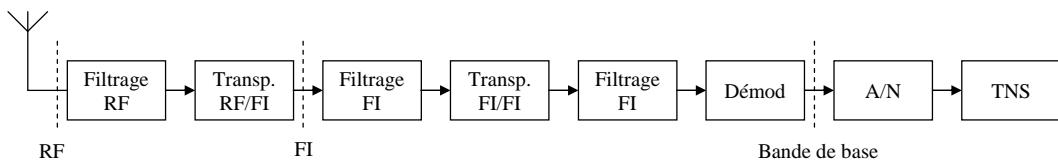


FIG. 1.1 – Architecture superhétérodyne

Contrairement à l'approche classique, l'idée de la radio logicielle est de réaliser de façon numérique tous les traitements depuis les fonctions radio fréquence jusqu'à aux couches applicatives en passant par les traitements en bande de base, ce qui permet d'exécuter les traitements en logiciel. La figure 1.2 présente l'architecture correspondant à la radio logicielle idéale. Cette évolution permet de se libérer de la contrainte actuelle qui est : à chaque standard un circuit dédié.

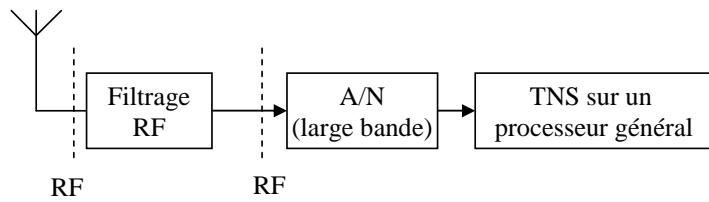


FIG. 1.2 – Architecture de radio logicielle idéale

Les verrous technologiques de la radio logicielle "idéale" peuvent être levés par l'adoption d'une architecture pragmatique [6]. C'est une étape intermédiaire d'une radio logicielle dite "restreinte" (Software Defined Radio, SDR) [2]. Un exemple d'architecture SDR est présenté par la figure 1.3. Ce concept correspond aux caractéristiques d'une numérisation par bande radio restreinte réalisée en fréquence intermédiaire ou en bande de base. La numérisation se rapproche donc de plus en plus de l'antenne afin de converger vers la radio logicielle idéale. Les circuits spécialisés sont remplacés par des circuits programmables permettant l'exécution d'applications logicielles.

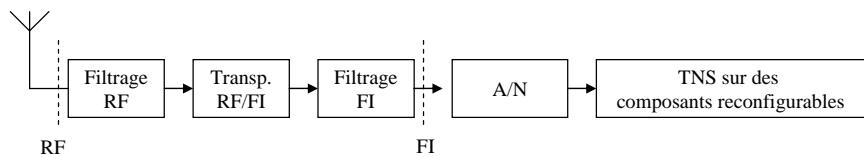


FIG. 1.3 – Architecture de radio logicielle restreinte

Le SDR Forum, principal consortium de promotion de la radio logicielle, a défini plusieurs niveaux de SDR désignant les capacités et la flexibilité atteignables de la radio logicielle [10]. Après les premières actions menées dans le cadre des programmes ACTS (Advanced Communications Technologies and Services) et IST, les projets européens se sont multipliés sur de très divers axes de recherches pour la radio logicielle. Lors du lancement

par l'IST du FP5 ("Framework Programme") , un point de recherche essentiel a été la reconfigurabilité des systèmes radio. Ceci a donné lieu à de nombreux projets dont ceux de l'IST comme CAST, TRUST, Pastoral, WIND-FLEX. Le projet End-to-End Reconfigurability (E^2R) [11] regroupe tous les principaux acteurs du domaine.

Au niveau de la recherche universitaire, on peut citer les travaux du professeur Ryuji Kohno [12] de la Yokohama National University au Japon et ceux du professeur Jeffrey Reed [13] de Viginia Tech aux USA. On peut citer également l'Université de Karlsruhe avec le professeur F.Jondral et le Trinity Collège de Dublin avec le professeur L.Doyle. Au niveau national, l'institut Eurécom a dévoloppé une plate-forme ouverte [14] dans le cadre de projet PLATON et RHODOS. Des SoC et NoC sont également conçus pour la radio logicielle par le CEA. Ils sont nommés FAUST et MAGALI. Notre équipe SCEE de SUPELEC est aussi reconnue au niveau internationnal sur la thématique radio logicielle.

1.2 La paramétrisation dans le contexte de la radio logicielle

Le principe de la paramétrisation est de rechercher dans un premier temps les caractères communs entre les traitements de différents standards puis d'en proposer des architectures communes et flexibles, à l'opposé d'une approche de type Velcro. Ce type de recherche au niveau algorithmique est nécessaire pour exploiter les concepts sur les similarités et les communautés entre traitements dans la conception d'application multi-standards. Ces approches ont pour objectif de mettre en évidence les similitudes entre les traitements à réaliser. Deux approches peuvent être distinguées en paramétrisation : l'approche fonction commune et l'approche opérateur commun.

1.2.1 Fonctions communes

Le premier type de paramétrisation se fait au niveau fonction. Le niveau de granularité est élevé. Le but de ce type d'approche est de définir des fonctions paramétrables communes à plusieurs standards. Une liste de paramètres est associée à la fonction et détermine son mode de fonctionnement.

A.Wiesler et al. proposent des fonctions paramétrables pour les fonctions de modulation et d'égalisation communes aux standards GSM, DECT, UMTS et IS136. Dans ce type d'approche, tous les informations sur les standards doivent être connues avant la conception de la structure. L'inconvénient de ce type d'approche est qu'elle n'exploite pas le principe de reconfiguration et qu'elle ne supporte pas de nouveaux standards.

1.2.2 Opérateurs communs

La granularité de cette appoche est moins élevée que celle des fonctions communes. L'idée est de définir des structures de calcul pour les appliquer à un grand nombre de fonction. La réutilisation devient alors optimale. Par exemple, un opérateur commun de type filtrage a été identifié dans [15] permettant de réaliser des opérations de codage de canal ou de calcul de CRC. Palicot et al. [16] introduisent cette notion d'opérateur commun en radio logicielle en donnant l'exemple de l'opérateur FFT4. A.Al Ghouwayel [17] propose une structure de l'opérateur FFT dans le champ de Galois pour un encodeur Reed-Solomon (RS). Cet encodeur est réalisé dans le domaine fréquenciel pour bénéficier de l'architecture matérielle performante du papillon FFT.

C'est cette approche qui a été privilégiée dans ce travail de thèse. Nous nous intéressons particulièrement à l'opérateur CORDIC comme opérateur commun [18], car il peut être considéré comme un opérateur générique pouvant être utilisé dans un grand nombre d'algorithmes classiques de traitement du signal. Nous donnerons des exemples d'applications réalisées par l'opérateur CORDIC dans le chapitre 3.

1.3 Contraintes et besoins pour les architectures radio logicielle

Face à des verrous technologiques, les contraintes liées aux architecture radio logicielle sont nombreuses. La radio logicielle "idéale" nécessite l'utilisation des processeurs généralistes pour obtenir une flexibilité maximale du système. Ceci demande un coût très élevé en consommation. Dans les systèmes embarqués cette contrainte de consommation est forte. Or les systèmes actuels basés sur des ASIC, ne donnent pas cette flexibilité à cause de sa paramétrisation réduite. Les plateformes les plus adaptées à ce type de traitement sont hétérogènes pour s'adapter à la diversité apportée par la radio logicielle, car ces plateformes sont constituées de divers éléments ayant des capacités et des avantages complémentaires : les processeurs généraux (GPP), les processeurs de traitement numérique (DSP), les circuits spécialisés (ASIC), les circuits programmables (FPGA). Une plateforme hétérogène peut donc traiter, avec une bonne efficacité énergétique, la diversité des fonctions en bande de base issues des différents standards.

1.4 Conclusion

La radio logicielle est une technique prometteuse pour répondre aux besoins de la future génération de la radio communication. Le principe de la radio logicielle a été défini dans ce chapitre. Les différents concepts de la radio logicielle sont présentés, ainsi que les techniques de paramétrisation de la radio logicielle permettant d'identifier des fonctions ou opérateurs communs. Les contraintes de l'architecture radio logicielle nécessite d'utiliser une plateforme hétérogène disposant de composants reconfigurables.

Ayant une très bonne efficacité spectrale et adoptées dans plusieurs standards de systèmes de communications sans fil, les techniques MIMO seront présentes dans le futur système radio logiciel. Nous allons maintenant présenter le principe des systèmes MIMO et les divers algorithmes de réception en les comparant en terme de performance et de complexité.

Chapitre 2

Système MIMO

Sommaire

2.1	Introduction	12
2.2	Modèle de canal MIMO à évanouissements	12
2.3	Capacité des canaux MIMO	13
2.4	Présentation du système MIMO	14
2.4.1	Système MIMO à base de codes spatio-temporels	15
2.4.2	Système MIMO à base de Multiplexage spatial	17
2.5	Algorithmes de réception MIMO : matrice du canal H connue	20
2.5.1	Algorithmes adaptés au multiplexage spatial	20
2.5.2	Maximum de Vraisemblance	21
2.5.3	Branch Bound	22
2.5.4	Décodage par sphères	22
2.5.5	Récepteur linéaire du Forçage à Zéro (ZF)	22
2.5.6	Récepteur linéaire MMSE	22
2.5.7	Récepteur à retour de décision V-BLAST	23
2.5.8	V-BLAST Square Root	23
2.5.9	Structure SIC de type QR	23
2.5.10	Structure SIC de type GDFE	24
2.5.11	Comparaison en terme de performance/complexité	24
2.6	Algorithmes de réception MIMO : matrice du canal H inconnue	27
2.6.1	La séparation aveugle de sources	27
2.6.2	Le critère CM	28
2.7	Description de l'algorithme "V-BLAST Square Root"	30
2.7.1	Principe du récepteur V-BLAST	30
2.7.2	Base de l'algorithme "V-BLAST Square Root"	31
2.7.3	Algorithme "V-BLAST Square Root"	32
2.7.4	Optimisation de l'algorithme "V-BLAST Square Root"	34
2.8	Description fonctionnelle de l'algorithme "V-BLAST Square Root"	35
2.9	Conclusion	36

2.1 Introduction

Avec l'intégration de l'Internet et de nouvelles applications multimédia dans les systèmes de communications sans fil, la demande en terme de débit ne cesse d'augmenter. Plusieurs techniques ont été développées pour répondre à ce besoin. La technique MIMO découverte en 1996 par les chercheurs de Bell Labs [19] reste la plus prometteuse, elle peut augmenter d'une manière substantielle l'efficacité spectrale [20] [21] [22]. Par exemple, les standards de réseaux locaux sans fil à haut débit, tels que IEEE 802.11n, Hiperlan2 [23] et IEEE 802.16 [24], vont adopter les systèmes MIMO dans leurs futures normes. Le débit des transmissions de HSDPA (High Speed Downlink Packet Access) peut atteindre 21.6Mb/s en utilisant les systèmes MIMO [23]. Contrairement aux systèmes classiques (SISO, MISO et SIMO), un système MIMO est un système de communication qui utilise plusieurs antennes à l'émission aussi bien qu'à la réception. Cette technique a reçu beaucoup d'intérêt ces dernières années et a donné lieu à de nombreux travaux. Parmi les difficultés engendrées par cette technique, l'implémentation des algorithmes de démodulation des signaux MIMO est un sujet d'actualité.

Il est à noter que c'est une thèse en électronique non pas en traitement du signal. Nous voulons simplement expliciter les bases de contexte sur lequel nous appuyons notre étude. Nous décrivons les diverses approches algorithmiques afin de déterminer l'algorithme le plus adapté à une réalisation matérielle. Nous présentons donc dans ce chapitre l'aspect théorie du système MIMO. Nous aborderons l'aspect implémentation dans le chapitre suivant.

2.2 Modèle de canal MIMO à évanouissements

Nous utilisons dans cette étude un modèle mono-utilisateur dans un canal non sélectif en fréquence. De plus, le modèle est sans mémoire .C'est le modèle de canal MIMO le plus utilisé, il consiste en une matrice H , dont chaque coefficient complexe h_{ij} représente la fonction de transfert entre la i^e antenne réceptrice et la j^e antenne émettrice. Pour un système avec M antennes à l'émission et N antennes en réception, le vecteur reçu r peut s'écrire :

$$r = Hs + v . \quad (2.1)$$

Dans cette équation, $s = [s_1, s_2, \dots, s_M]^T$ est le vecteur de symboles émis, H est la matrice de canal de dimension $M \times N$ et $v = [v_1, v_2, \dots, v_N]^T$ est le vecteur de bruit additif gaussien en réception. Nous supposons que $E[ss^*] = I_M$, $E[vv^*] = \sigma^2 I_N$ et $E[sv^*] = 0$.

Les éléments de H ont une phase uniformément distribuée et une amplitude qui suit une loi de Rayleigh. Ce modèle est typique d'un environnement avec de nombreux échos et un écart suffisant entre les antennes. Nous supposons aussi que le canal reste constant durant la transmission d'un bloc de données et que le récepteur connaît parfaitement la matrice de canal H . Cette connaissance peut s'obtenir soit par des symboles d'apprentissage soit par estimation aveugle du canal [25], [26] et [27].

2.3 Capacité des canaux MIMO

Pour les formules de capacité, les notations suivantes sont utilisées :

- La puissance du bruit est identique sur chaque antenne de réception et est notée σ^2 .
- P_R est défini comme la puissance moyenne qui serait reçue sur chaque antenne si un seul émetteur utilisait toute la puissance P_0 (donc un canal SIMO).
- Le rapport signal sur bruit moyen (SNR) sur chaque antenne de réception est $\rho_R = P_R/\sigma^2$, et est indépendant de N .

La capacité des systèmes MIMO est présenté dans l'article [28], dont nous ne présentons ici que les éléments de base pour montrer l'intérêt des transmissions MIMO. Nous supposons que la puissance totale moyenne émise P_0 reste constante afin de pouvoir comparer les différentes capacités. Lorsque le nombre d'antennes varie à l'émission, la puissance est répartie entre les M antennes de façon que leur somme reste égale à P_0 . Si aucune connaissance du canal n'est disponible à l'émetteur, la répartition de puissance uniforme est optimale en terme de capacité (chaque antenne émet une puissance P_0/M) [4].

Canal SISO

La capacité d'un canal SISO est [29] :

$$C = \log_2(1 + \rho_R) \text{ bps/Hz.}$$

Elle augmente lentement, en fonction du logarithme de $1 + \rho_R$. Lorsque le SNR est élevé, un gain de 3 dB sur ρ_R ne fournira une augmentation de capacité que d'un bit par seconde par hertz (bps/Hz).

Canal SIMO

La capacité d'un canal SIMO (Single Input, Multiple Output) est donnée par [22] :

$$C = \log_2(1 + \rho_R M^2) \text{ bps/Hz.}$$

Sa capacité augmente en fonction du logarithme de $1 + \rho_R M^2$, soit un peu plus rapidement que dans le cas SISO. Elle reste toutefois petite devant celle du canal MIMO, car la dimension spatiale du système n'est que partiellement exploitée.

Canal MIMO

Pour un canal MIMO, avec une puissance de ρ_0/M sur chaque antenne d'émission, la capacité est [4],[22] :

$$C = \log_2(\det[I_M + \rho_R/NHH^*]) \text{ bps/Hz.}$$

En particulier lorsque M et N sont grands, l'espérance de la capacité pour un canal de Rayleigh croît proportionnellement à N :

$$C \approx N \log_2(1 + \rho_R) \text{ bps/Hz.}$$

La capacité augmente donc beaucoup plus vite que dans les cas SISO et SIMO.

L'avantage en capacité des systèmes MIMO est principalement dû à l'exploitation des trajets multiples. Ils permettent d'émettre plusieurs symboles simultanément. Le prix à payer pour cette augmentation de la capacité est d'abord matériel, avec la multiplication des antennes et de leur électronique associée, mais aussi logiciel, avec des récepteurs plus complexes qui demandent plus de puissance de calcul.

2.4 Présentation du système MIMO

L'apparition des systèmes MIMO a été motivée par le besoin accru en terme de débit par l'arrivée de nouveaux services tels que l'accès à Internet et la transmission d'images via les systèmes de communications sans fil, ainsi que la saturation des ressources en canaux de transmission, en particulier dans la bande de la téléphonie mobile. Les systèmes MIMO consistent à utiliser plusieurs antennes à l'émission et à la réception. En bénéficiant des traitements spatio-temporels associés, ces systèmes ont montré une augmentation considérable de l'efficacité spectrale (proportionnelle au nombre d'antennes utilisées) [30].

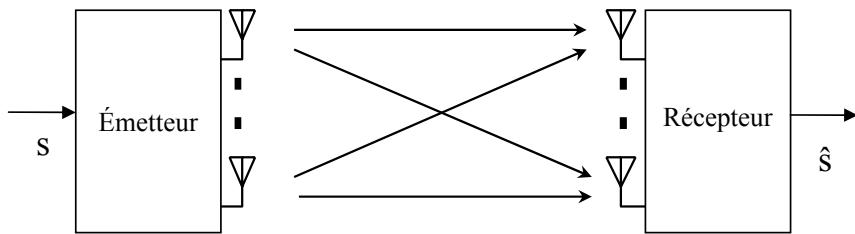


FIG. 2.1 – Principe d'un système MIMO

Partant du point de vue de la théorie de l'information, deux chercheurs des laboratoires Bell, Foschini [4] et Teletar [22] ont indépendamment montré que la capacité des systèmes multi-antennes augmentait linéairement avec le nombre d'antennes émettrices. Ces découvertes sont à l'origine des systèmes MIMO (voir figure 2.1) qui visent essentiellement à résoudre les problèmes d'encombrement et de limitation de capacité des réseaux sans fil large bande. L'idée de base dans les systèmes MIMO est le traitement spatio-temporel, où le temps (dimension naturelle) est complété par une dimension spatiale inhérente à l'utilisation de plusieurs antennes. Un tel système peut être vu comme l'extension des antennes intelligentes. La propriété clef d'un système MIMO est sa capacité à tourner la propagation multi-trajets (traditionnellement un inconvénient) en un avantage, en d'autres termes les systèmes MIMO exploitent les trajets multiples plutôt que de les supprimer.

Les techniques conventionnelles utilisées à la réception pour annuler la distorsion apportée par le canal MIMO nécessitent souvent, soit la connaissance du canal, soit l'utilisation d'une séquence de symboles connue au niveau du récepteur. Dans la pratique le canal est inconnu, donc une estimation de ce dernier est nécessaire. Souvent l'estimation du canal est basée sur l'utilisation des séquences d'apprentissage multiplexées avec les données utiles,

ce qui diminue bien évidemment le débit utile. Pour des canaux invariants dans le temps, la perte n'est pas significative car un seul cycle d'apprentissage est nécessaire.

On distingue principalement trois types de techniques pour la transmission sur les systèmes MIMO la première est basée sur les codes spatio-temporels, la deuxième sur le multiplexage spatial [30],[31] et la troisième est le précodage. De plus ces techniques peuvent être combinées avec de l'OFDM. Cette combinaison permet d'utiliser les trois premiers groupes sur des canaux sélectifs en fréquence. Nous allons détailler les décodeurs utilisés dans ces trois types de techniques.

2.4.1 Système MIMO à base de codes spatio-temporels

Afin d'améliorer la qualité de la transmission, Alamouti [32] et Tarokh [33] ont conçu des systèmes basés essentiellement sur la diversité, proposant un codage et un étiquetage conjoints. Ce codage spatio-temporel (CST ou en anglais Space-Time Coding, STC) permet également des communications plus sûres, il consiste à ajouter de la redondance aux données binaires émises afin d'augmenter la diversité spatiale et éviter les évanouissements propres au canal MIMO. Pour plus de détails sur les codes spatio-temporels, se référer à [34], [32], [33].

2.4.1.1 Codage spatio-temporel en treillis

Les codes spatio-temporels ont été créés par Tarokh pour les systèmes MISO, proposant ainsi la première famille de STC, les STC en treillis (STTC) [20].

Ils combinent le codage de canal avec la modulation sur les antennes émettrices, et peuvent être considérés comme une extension des codes en treillis classiques [20] au cas des antennes multiples à l'émission et à la réception. Si le code est bien construit, on peut ajouter à l'avantage évident de diversité un gain de codage loin d'être négligeable.

Le STTC crée des relations entre les signaux à la fois dans l'espace (plusieurs antennes émettrices) et dans le temps (symboles consécutifs). Le codeur est composé de M polynômes générateurs qui déterminent les symboles émis simultanément. La figure 2.2 propose le diagramme de treillis d'un STTC à 4 états utilisant une modulation simple MDP-4, avec un nombre d'antennes émettrices $M = 2$.

Le fonctionnement du codeur est relativement simple, et peut être résumé comme suit :

- η_k représente l'état du treillis à l'instant k et par conséquent l'état suivant est noté η_{k+1} .
- Considérons que le treillis est à l'état initial $\eta_k = 0$.
- L'état suivant du treillis dépend des bits d'information à coder. Ainsi, si les deux bits à coder sont 11, alors l'état suivant prend la valeur décimale équivalente c'est-à-dire $\eta_{k+1} = 3$.
- Les symboles à droite du treillis sont les codes associés à chaque doublet d'éléments binaires entrants. Dans notre cas ($\eta_k = 0$ et $\eta_{k+1} = 3$) le doublet à la sortie du codeur est donc 30 (3 sur la première antenne et 0 sur la seconde).
- Ces symboles sont alors mis en forme par la MDP-4 avant l'émission par leur antenne respective.

Il est intéressant de noter les similitudes et les différences entre les modulations codées à treillis multiple (MCTM) [35] et les STTC. Dans les STTC, les symboles associés à une branche du treillis sont répartis dans l'espace (les antennes), alors qu'il sont répartis dans le

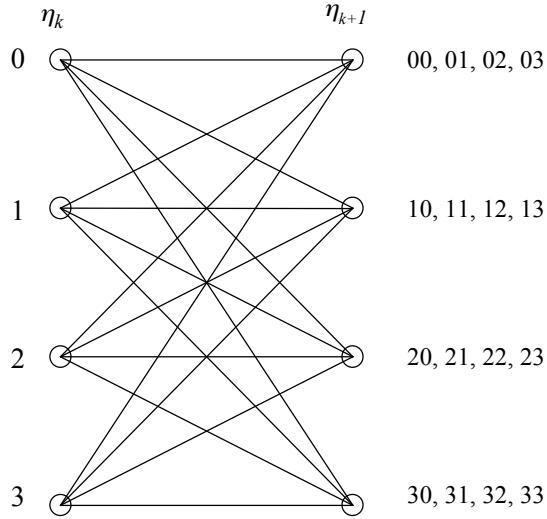


FIG. 2.2 – Diagramme de treillis pour un STTC à 4 états utilisant $M = 2$ émetteurs et une modulation MDP-4

temps pour les MCTM. En considérant le même alphabet, l'efficacité spectrale des STTC est donc N fois plus grande que celle des MCTM. De plus, contrairement aux MCTM qui nécessitent un entrelaceur pour créer un canal sans mémoire, les symboles transmis par les STTC sont naturellement décorrélés (ou très légèrement corrélés) grâce à la séparation physique des antennes.

La réception est basée sur l'estimation des coefficients d'évanouissement du canal et un algorithme de décodage parmi ceux que nous présentons dans la section 2.5.2. Etant donnée la structure des treillis, la complexité de décodage augmente toutefois très rapidement.

2.4.1.2 Codage spatio-temporel par blocs

En espérant réduire la complexité exponentielle du décodeur STTC, Alamouti a proposé un schéma simple de diversité d'émission [32], pour former une nouvelle classe de codes spatio-temporels, les codes spatio-temporels en blocs (STBC), possédant le même avantage de diversité que les techniques de combinaisons des répliques à gain maximal.

Les STBC sont définis comme une opération de modulation d'un bloc de symboles à la fois dans l'espace et dans le temps, créant ainsi des séquences orthogonales transmises par des antennes émettrices différentes.

Le schéma original d'Alamouti comportait deux antennes à l'émission pour atteindre un ordre de diversité égal à 2 et une seule à la réception, le tout sans aucune connaissance du canal à l'émission. La structure de codage proposée peut être représentée matriciellement sous la forme suivante :

$$C = \frac{1}{\sqrt{2}} \begin{pmatrix} s_0 & -s_1^* \\ s_1 & s_0^* \end{pmatrix} \quad (2.2)$$

L'objectif de ce schéma de codage est la diversité pure, et il n'est pas question ici d'augmentation du débit de données. Les lignes de la matrice C représentent les antennes alors que les colonnes sont les poids attribués à chaque période symbole. Comme le bloc de symboles formé par s_0 et s_1 est codé à la fois dans l'espace et dans le temps, le schéma a tout naturellement pris le nom de code spatio-temporel par blocs.

Des études récentes visant à étendre ce travail à plus de 2 antennes émettrices ont montré qu'il est impossible dans ce cas d'obtenir un code parfaitement orthogonal sauf pour des modulations à valeurs strictement réelles, telles les modulations à impulsions d'amplitude (MIA). De nombreux "codes algébriques" ont toutefois vu le jour, les uns sacrifiant le débit pour préserver une structure simple à décoder, les autres augmentant le débit au prix de l'orthogonalité des codes.

Le récepteur est lui composé d'une estimation de canal et d'une détection des symboles. Le temps de cohérence du canal est supposé plus grand que la longueur d'un bloc. Le décodeur STBC, malgré une structure formidablement simple, est donc capable du même gain de diversité que les combinaisons des répliques à gain maximal.

Il faut noter qu'il n'y a aucune mémoire entre les blocs consécutifs et que la longueur typique d'un bloc est très courte, ce qui restreint fortement le gain de codage que l'on peut espérer. Cependant, grâce à la faible complexité du décodeur, une association avec un code correcteur d'erreur est tout à fait envisageable. Les turbo-codes semblent représenter le codage correcteur le plus performant à l'heure actuelle et leur utilisation est déjà préconisée dans de nombreuses normes de télécommunications (UMTS, DVB-RCS...) ; leur insertion dans une chaîne MIMO est donc tout naturellement étudiée dans [36].

Si le récepteur n'a aucune information sur le canal, les systèmes utilisent des codes non-cohérents, comme les codes différentiels introduits dans [35]. Ces codes ont été améliorés par des techniques itératives [37] et de concaténation avec des codes convolutifs traditionnels [38] ou encore par l'usage de la transformée de Cayley [39].

Pour la conception de systèmes de transmission MIMO, les codes spatio-temporels ne sont donc pas les solutions les plus adaptées. En revanche les systèmes basés sur le multiplexage spatial permettent d'augmenter significativement le débit.

2.4.2 Système MIMO à base de Multiplexage spatial

Le principe du multiplexage spatial est l'organisation en espace sans redondance d'une série d'informations. Le système transmet alors N_t fois plus de symboles utiles à chaque instant. Les antennes émettrices utilisent la même modulation et la même fréquence porteuse pour transmettre les symboles différents et indépendants sur les différentes antennes. L'efficacité du système augmente donc en continuant à utiliser la même bande passante qu'un système classique. Les principales techniques de multiplexage spatial ont été développées par les laboratoires Bell. On peut distinguer différents types du système MIMO par multiplexage spatial, tels que D-BLAST, H-BLAST, V-BLAST ou Turbo-BLAST. Ce type de système MIMO est réalisé dans le but d'augmenter le débit de transmission.

2.4.2.1 D-BLAST

Foschini a proposé dans [19] une architecture(figure 2.3) à multiplexage spatial de manière à exploiter de façon optimale la diversité et la capacité. Dans cette architecture, les symboles sont détectés successivement antenne d'émission par antenne d'émission. C'est

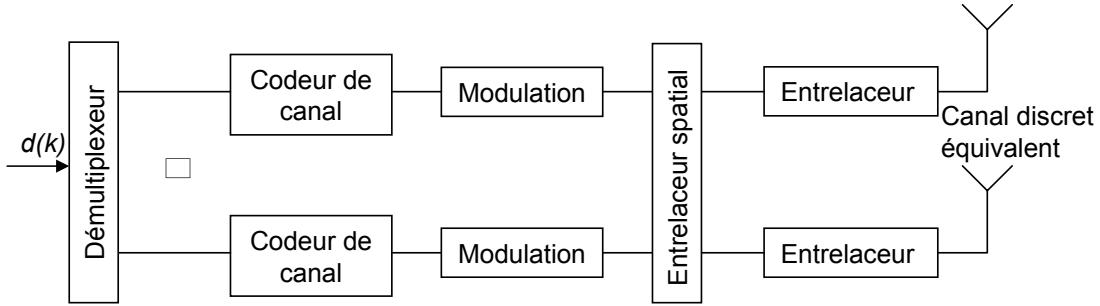


FIG. 2.3 – Architecture d'un transmetteur D-BLAST

une architecture diagonale nommée D-BLAST. La technique D-BLAST est décrite de manière plus théorique par Foschini en 1998 en considérant par la suite des cas plus réalistes [4]. Le flux de données est démultiplexé en entrée en N sous-flux ayant le même débit. Chaque sous-flux est ensuite codé (de la redondance peut être introduite suivant le codeur canal utilisé), mais il n'y a pas d'échanges d'informations entre ces codeurs. Puis les sous-flux sont modulés. L'association flux/antenne est périodiquement cyclique pour ne pas transmettre les N sous-flux vers la même antenne. Le système est plus résistant aux effets d'évanouissements du canal grâce au fait de transmettre un sous-flux en utilisant toutes les antennes possibles. Cette architecture permet d'obtenir un débit proche de la capacité mais elle possède une structure de codeur et décodeur plus complexe. Pour réduire la complexité, Foschini et Wolniansky proposent un système qu'ils nomment V-BLAST ou H-BLAST et qui a la particularité d'être plus simple à mettre en oeuvre que D-BLAST.

2.4.2.2 V-BLAST

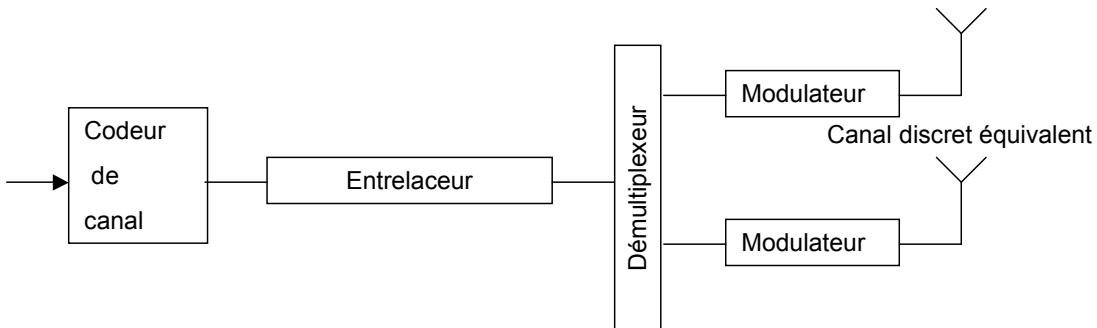


FIG. 2.4 – Architecture d'un transmetteur V-BLAST

V-BLAST (Vertical Bell Labs Layered Space-Time) [40], [41] peut être vu comme une classe spéciale des codes de multiplexage spatial, son principe consiste à diviser le flux de données à l'entrée en plusieurs sous-flux, ces derniers sont transmis sur des antennes différentes. Cette architecture vise principalement à augmenter la capacité du système. Une description simple est donnée sur la figure 2.4.

Les antennes d'émission transmettent chacune un symbole différent, indépendant de celui des autres antennes, mais en utilisant la même modulation et la même fréquence porteuse. La bande passante utilisée reste identique à celle d'un système classique, mais comme plusieurs symboles sont émis, l'efficacité spectrale augmente.

La figure 2.5 illustre le démultiplexage et la modulation du code V-BLAST.

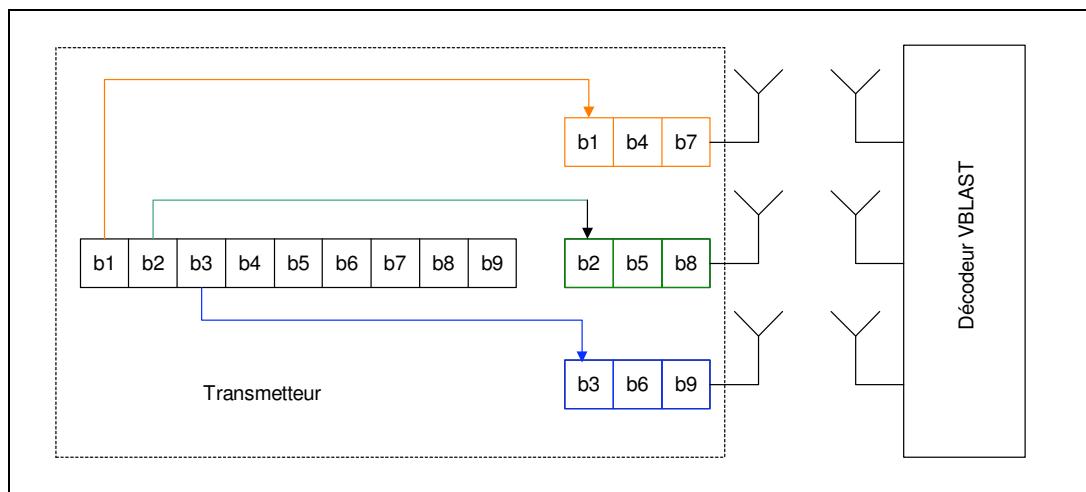


FIG. 2.5 – Principe de l'architecture de modulation V-BLAST

Les bits d'informations sont tout d'abord codés par un codeur de canal. Ensuite ces bits issus du codage sont alors entrelacés et le flux résultant est démultiplexé en N_t sous-flux attaquant chacun un modulateur.

Nous nous intéresseront dans la suite de cette thèse uniquement au multiplexage spatial et plus particulièrement à l'architecture V-BLAST.

2.4.2.3 H-BLAST

L'architecture d'un transmetteur H-BLAST est très proche de celle du V-BLAST, comme illustré sur la figure 2.6. Elle utilise les mêmes blocs que V-BLAST sauf l'ordre de rangement de ces blocs.

La figure 2.7 illustre le démultiplexage et la modulation du code H-BLAST.

Les bits d'information sont démultiplexés en N_t sous-flux. Puis ils sont codés par un codeur. Les bits issus du codage sont alors modulés et entrelacés.

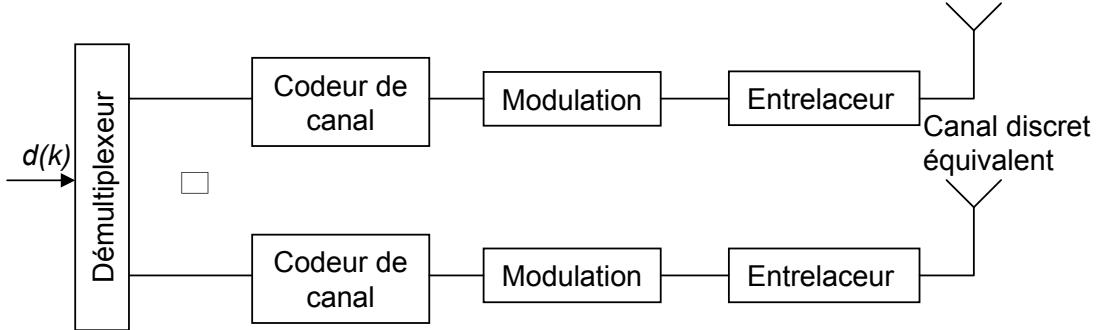


FIG. 2.6 – Architecture d'un transmetteur H-BLAST

2.4.2.4 Turbo-BLAST

Le principe turbo appliqué au multiplexage spatial a été proposé dans [42] et [43]. La structure du transmetteur est exactement la même que celle du D-BLAST sauf que l'entrelaceur spatial n'ajoute pas de partie nulle dans la matrice.

2.5 Algorithmes de réception MIMO : matrice du canal H connue

Il existe de nombreux algorithmes de réception envisageables pour récupérer les symboles dans les systèmes MIMO sous hypothèse que la matrice du canal est connue. Les moins complexes sont les récepteurs linéaires basés sur le critère de forçage à zéro (ZF) ou la minimisation de l'erreur quadratique moyenne (MMSE). L'algorithme de réception proposé pour le système V-BLAST [19] essaie d'éliminer successivement les interférences dues aux émetteurs autres que celui dont on estime les symboles. Ils sont sous-optimaux en termes de TEB. Le détecteur optimal est basé sur le maximum de vraisemblance (ML) qui demande une importante charge de calcul lorsque le nombre d'antennes et la taille de la constellation sont grands. Il existe de nombreux algorithmes sous-optimaux basés sur le ML (par exemple décodage sphère). L'étude de nouveaux algorithmes performants ne relève pas de ce travail. Il existe de très nombreux algorithmes et notre objectif n'est pas d'en faire une recherche exhaustive. Nous souhaitons seulement dans cette section identifier, parmi les algorithmes les plus connus, les meilleurs algorithmes en terme de rapport complexité/performance que nous pourrons planter sur FPGA.

2.5.1 Algorithmes adaptés au multiplexage spatial

Dans un premier temps nous allons présenter les algorithmes adaptés au multiplexage spatial. Le SIC (Successive Interference Cancellation) et l'OSIC (Order SIC) sont les deux principaux algorithmes de réception dans ce contexte.

Algorithme SIC

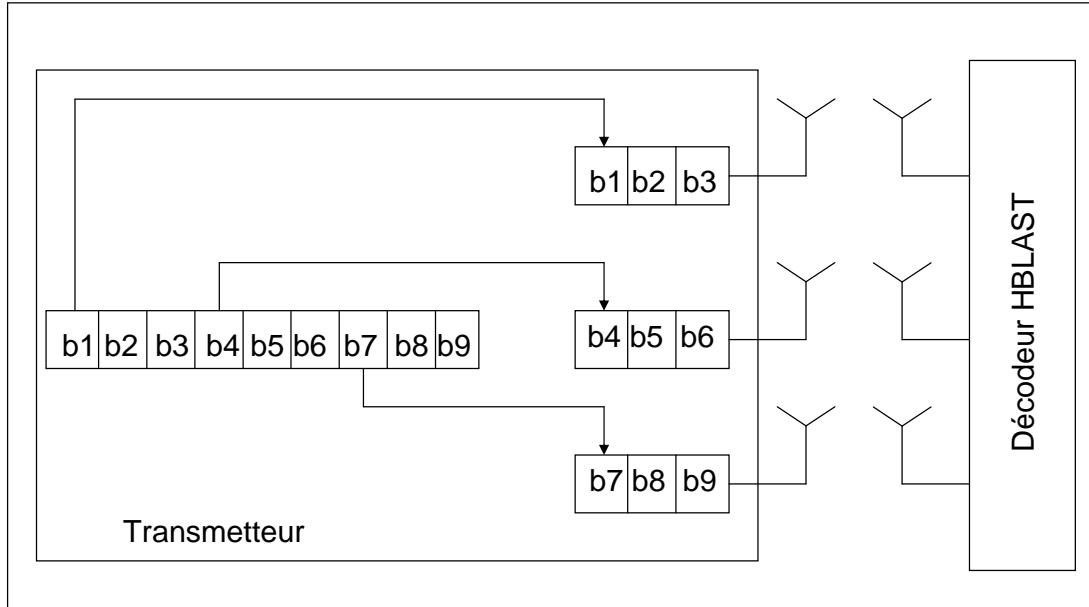


FIG. 2.7 – Principe de l'architecture de modulation H-BLAST

Il s'agit d'un égaliseur à retour de décision adapté à la structure des systèmes MIMO. Ce type de structure consiste à soustraire les interférences de la valeur du signal reçu pour assurer la restauration du message émis. A chaque itération, la contribution des éléments de symbole déjà détectés est retranchée du vecteur reçu ce qui donne un vecteur reçu contenant moins d'interférences à condition que la décision prise soit bonne. L'annulation peut utiliser soit le critère du forçage à zéro (Zero Forcing - ZF), soit le critère minimisant l'erreur quadratique moyenne (MMSE).

Algorithme OSIC

En utilisant la méthode précédente, l'ordre dans lequel les éléments de vecteurs reçus sont traitées devient important pour améliorer les performances du système. L'ordre optimal consiste à démoduler en premier à chaque itération le symbole reçu ayant le rapport signal sur bruit le plus fort. Sa contribution à la réception est ensuite annulée, ce qui permet de minimiser la propagation d'erreurs à chaque bonne décision.

2.5.2 Maximum de Vraisemblance

Le récepteur du maximum de vraisemblance (ML) offre les meilleures performances en Taux d'Erreur Binaire(TEB). En effet, il est optimal si les vecteurs émis s sont équiprobables, ce qui est le cas puisque les symboles s_i sont équiprobables et que les N voies émises en parallèles sont indépendantes. Il s'exprime classiquement de la façon suivante :

$$S = \operatorname{argmin} \|r - Hs\|^2.$$

Toutefois sa charge de calcul devient rapidement très importante car elle augmente exponentiellement avec le nombre d'antennes à l'émission M et linéairement avec le nombre d'antennes en réception N. Le problème est NP complet et il n'existe pas d'algorithme permettant de trouver une solution optimale en temps polynomial. Il est donc naturel de rechercher des algorithmes plus simples ayant des performances proches. Dans cette catégorie nous trouvons l'algorithme du décodage par sphères (SD) [44], [45], ainsi que l'algorithme "Branch Bound". Nous ne listerons pas ici l'ensemble des récepteurs sous-optimaux, la littérature étant très importante sur ce sujet. Nous nous limitons aux plus connus.

2.5.3 Branch Bound

Le principe consiste à construire un arbre dont chaque branche devient un sous-problème plus simple à résoudre [46], [47]. La conservation d'une branche dans l'algorithme est régit par 2 bornes. Si la borne calculée dans la branche considérée est supérieure à celle de la branche conservée précédemment alors cette branche est abandonnée.

2.5.4 Décodage par sphères

Il est cependant possible d'approcher les performances du maximum de vraisemblance en gardant une complexité raisonnable, [44], [45]. Le principe de cet algorithme est de se placer au niveau du signal reçu et de chercher le point du réseau le plus près à l'intérieur d'une sphère de rayon C . Cela diminue fortement le domaine de recherche par maximum de vraisemblance, puisque seuls les points du réseau situés à une distance inférieure à C du signal reçu sont considérés lors de la minimisation de la métrique. Le choix de C est donc crucial pour la vitesse de convergence de l'algorithme et pour la précision des résultats.

Le décodage par sphère fait partie des méthodes optimales mais sa structure n'est pas régulière et ne permet pas une implantation matérielle optimisée. Un nouveau décodeur MIMO utilisant une approche géométrique nommé HISD (Hyperplane Intersection and Selection Decoder) [48] est proposé pour trouver des solutions proches de la solution optimale.

2.5.5 Récepteur linéaire du Forçage à Zéro (ZF)

Ce récepteur est le plus simple et aussi le moins performant. Il cherche à annuler les contributions des autres émetteurs sur chaque symbole. Ceci revient à inverser la matrice de transfert du canal :

$$S = (H^* H)^{-1} H^* r.$$

Lorsque H est mal conditionnée, son inversion multiplie le bruit et dégrade alors sérieusement les performances à faible SNR.

2.5.6 Récepteur linéaire MMSE

Ce critère minimise l'erreur moyenne quadratique due à la fois au bruit et aux interférences entre symboles, contrairement au récepteur ZF qui ne s'occupe que des interférences entre

symboles. Son expression est :

$$S = (H^*H + \sigma^2 I)^{-1}H^*r.$$

Ce récepteur résiste mieux au bruit que le récepteur ZF. A haut SNR, le récepteur MMSE tend vers le récepteur ZF car σ^2 tend vers 0.

2.5.7 Récepteur à retour de décision V-BLAST

Le principe de l'algorithme du récepteur V-BLAST a été présenté dans [40], [19]. Il s'agit d'un égaliseur à retour de décision, adapté à la structure des systèmes MIMO. L'égaliseur peut utiliser soit le critère du forçage à zéro, soit le critère du MMSE. Son principe est le suivant : le symbole le plus favorisé (possédant le meilleur TEB suivant le critère considéré) est démodulé en premier. Sa contribution au vecteur reçu r est ensuite annulée, ce qui augmente le SNR sur les autres symboles (à chaque bonne décision). Cette étape est répétée jusqu'au dernier symbole, le moins favorisé. Ce récepteur est aussi noté dans la littérature OSIC (Ordered Successive Interference Cancellation). Cet algorithme améliore nettement les performances de l'algorithme V-BLAST sans ordonnancement. En effet au lieu d'effectuer le décodage dans l'ordre croissant ou décroissant, nous privilégions ici le rang de la matrice de coefficients de l'égaliseur en choisissant la norme de la rangée la plus fiable, où l'estimation de l'erreur est la plus faible. Comme pour tous les égaliseurs à retour de décision, son principal inconvénient est la propagation des erreurs. Une fois qu'une mauvaise décision a été prise sur la valeur d'un symbole, une mauvaise contribution est retirée au vecteur r , ce qui entraîne que les symboles suivants seront presque certainement mal estimés. C'est la raison pour laquelle l'ordonnancement est utilisé pour minimiser la propagation des erreurs. Nous allons aborder ce problème dans la section suivante.

2.5.8 V-BLAST Square Root

B.Hassibi présente en 2000 l'algorithme "V-BLAST Square Root" [49] qui permet de diminuer la complexité de $O(M^4)$ à $O(M^3)$ sans dégrader le TEB, dans le cas où $M=N$. Parmi les différents algorithmes de détection d'un signal MIMO l'algorithme " V-BLAST Square Root " réalise un bon compromis entre les performances attendues et la faible complexité de l'algorithme.

2.5.9 Structure SIC de type QR

C'est une méthode de structure SIC. A la différence de la structure V-BLAST, elle s'appuie sur une triangulation des coefficients de la matrice de canal H . Les symboles émis sont retrouvés par une méthode de résolution d'équations linéaires. La contribution du symbole déjà estimé est soustrait avant d'estimer le suivant. L'algorithme QR se présente dans les équations suivantes :

$$\begin{aligned} H &= QR \\ \tilde{y} &= Q^T y \\ i &= N \text{ à } 1 \\ \tilde{z}_i &= \tilde{y}_i - \sum_{j=i+1}^N r_{ij} \times \hat{s}_j \end{aligned}$$

$\hat{s}_i = \text{décision}(\tilde{z}_i/r_{ij})$

fin

Comme le montre dans ces équations, le dernier symbole \hat{s}_N est estimé en premier, puis sa contribution est retranchée dans l'équation suivante afin de déterminer le symbole suivant \hat{s}_{N-1} et ainsi de suite.

2.5.10 Structure SIC de type GDFE

L'approche GDFE (Generalised Decision Feedback Equalizer) apporte une amélioration qui s'appuie sur l'algorithme SIC de type QR. Cette structure utilise un filtre transversal de type linéaire. Dans l'objectif de décorréler partiellement les symboles, la structure de GDFE est construite à partir d'une matrice triangulaire supérieure dont les éléments de sa diagonale sont égaux à un. Le critère ZF ou MMSE peut être utilisé indifféremment dans cette structure.

L'algorithme GDFE se résume dans les équations suivantes :

$$H = QR$$

$$W_{GDFE} = \text{diag}^{-1}(R)Q^*$$

$$\tilde{y} = W_{GDFE} \times y$$

i = N à 1

$$\tilde{y}_i = \tilde{y}_i - \sum_{j=i+1}^N g_{ij} \times \hat{s}_j$$

$$\hat{s}_i = \text{décision}(\tilde{y}_i)$$

fin

Comme pour l'algorithme SIC de type QR, les symboles sont estimés en commençant par le dernier \hat{s}_N , puis leur contribution est alors retranchée dans l'équation suivante, ceci afin de déterminer le symbole suivant \hat{s}_{N-1} et ainsi de suite.

2.5.11 Comparaison en terme de performance/complexité

Dans ce paragraphe nous avons présenté les divers algorithmes de détection classique, tels que les structures linéaires et non linéaires ou les structures à maximum de vraisemblance. La figure 2.8 montre une évaluation du TEB pour les récepteurs que nous venons de présenter. Ces résultats sont obtenus dans un système MIMO muni de 2x2 antennes avec une modulation QPSK. Comme le montre la figure 2.8 le récepteur à maximum de vraisemblance est le plus performant, mais son inconvénient est que sa complexité algorithmique devient la plus élevée quand le nombre d'antennes augmente. Les performances des décodeurs de type linéaire semblent correctes mais pas suffisantes au regard des structures itératives. On voit bien à travers les simulations que le V-BLAST permet d'améliorer d'une manière notable les performances des systèmes linéaires.

Par opposition aux algorithmes à structures linéaires, nous pouvons constater que les algorithmes à structures itératives améliorent nettement les performances du récepteur, sans trop augmenter leur charge de calcul. La figure 2.8 montre les performances en terme de taux d'erreur binaire des décodeurs de type itératif à forçage à zéro et de type minimisant l'erreur quadratique moyenne.

Les différentes approches algorithmiques envisagées impliquent un accroissement important des traitements et de la cadence de calcul. Le tableau 2.1 présente pour un système MIMO 4x4 la complexité en termes de nombre d'opérations. Sur la figure 2.9 nous

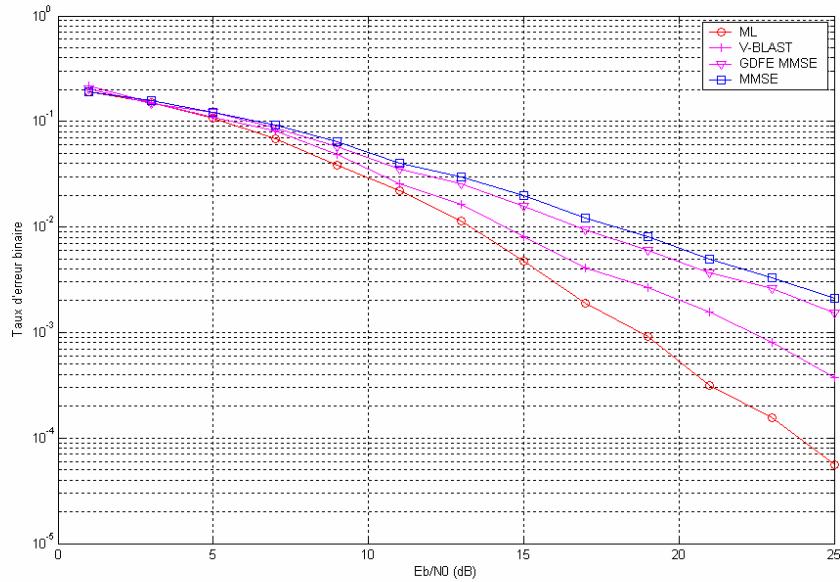


FIG. 2.8 – Comparaison des performances en TEB

avons représenté cette même complexité d'implémentation exprimée en nombre d'opérations (multiplications et additions) par symbole en fonction du nombre d'antennes.

Il est important de noter que pour l'algorithme du maximum de vraisemblance, la taille de la constellation joue un rôle important, parce que le récepteur doit calculer les distances pour un ensemble de vecteurs possibles dans la constellation. Le temps de calcul devient vite excessif pour les constellations de grandes tailles, même pour un faible nombre d'antennes. Par exemple, dans la table 2.1, un système QAM 16 avec 4 antennes nous donne $16^4 = 65536$ possibilités, donc trop complexe pour être retenu. Les performances des architectures de types linéaires semblent correctes mais pas suffisantes au regard des structures non linéaires.

Le table 2.2 montre la comparaison de l'implantation ASIC entre l'algorithme V-BLAST " Square Root " et d'autres détecteurs. Le détecteur exhaustive-search ML est

TAB. 2.1 – Comparaison de complexité des divers algorithmes ($M \times N$)

Algorithmes	Nombre d'opérations/symbole	Total pour 4×4 antennes
ZF	$ 5N^3/3-2N/3 $	103
MMSE	$ 3MN^2+N^3/3+MN/2-N^2/2 $	107
SIC V-BLAST	$ 2M^2N+2MN^2+11M^2 $	960
OSIC V-BLAST	$ M^2N^2+2NM^3+15M^4/4 $	1728
V-BLAST SR	$ 5M^3/3+8M^2N+2MN^2+2MN-M^2 $	763

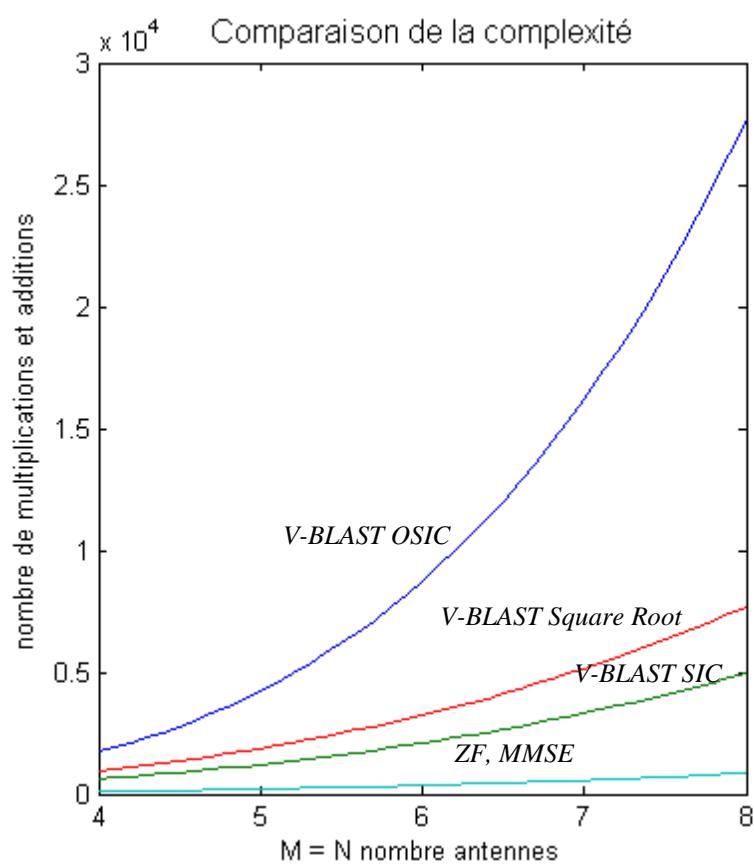


FIG. 2.9 – Comparaison de la complexité en fonction de nombre d'antennes

TAB. 2.2 – Comparaison de l'implantation ASIC des décodeur MIMO

Référence	[51]	[52]	[53]	[50]
Nb Antennes	4×4	4×4	4×4	4×4
Modulation	QPSK	16-QAM	QPSK	QPSK
Décodeur	OSIC V-BLAST SRA	K-best Sphère	ML-APP	ML
BER	sousoptimal	proche de ML	ML	ML
Technologie	$0.35 \mu m$	$0.35 \mu m$	$0.18 \mu m$	$0.25 \mu m$
Surface(GE)	190K	91K+prepoc.	140K	40K
Max. horloge	82MHz	100MHz	122MHz	100MHz
débit	160Mb/s	53.3Mb/s	28.8Mb/s	50Mb/s

pour l'instant un choix intéressant pour les systèmes à bas débit [50]. Concernant l'algorithme de décodage par sphère, il faut bien noter que les évaluations de complexité présentées dans la table 2.2 ne prennent pas en compte les précalculs comme l'inversion de matrice et la décomposition QR ou la factorisation Cholesky, son implantation reste donc coûteuse en surface. Il est critique pour le système de large bande MIMO-OFDM [50]. L'algorithme V-BLAST " Square Root " est très intéressant pour la modulation de grande taille, car la complexité de l'algorithme est presque indépendante de la taille de la constellation.

A la lecture des figures précédentes , nous avons sélectionné l'algorithme " Square Root " comme un des meilleurs compromis Complexité/Performance. Afin de bien comprendre l'implantation sur FPGA, nous décrivons d'abord en détail cet algorithme dans la section 2.7.

2.6 Algorithmes de réception MIMO : matrice du canal H inconnue

Depuis le papier fondateur de Foschini [19] sur le système MIMO, de nombreux auteurs se sont intéressés à la séparation aveugle de ce type de système dès 1997. La détection MIMO dans le cas où l'on considère que la matrice du canal est inconnue correspond à des algorithmes dits aveugles. Parmi ces algorithmes la séparation aveugle de sources est bien adaptée à la démodulation d'un mélange MIMO.

2.6.1 La séparation aveugle de sources

Les premiers travaux sur la séparation aveugle de sources ont été initiés par C.Jutten et J.Hérault [54] en 1985, dans le cas d'un mélange instantané. Depuis une multitude d'algorithmes a été proposée. Nous faisons ici un rapide tour d'horizon des méthodes de séparation de sources proposées dans la littérature. Les algorithmes de séparation de sources peuvent être classés selon différents critères : mélange instantané ou mélange convolutif,

linéaire ou non linéaire, batch ou adaptatif, statistiques d'ordre deux ou d'ordre supérieur. Ici nous considérons un classement selon le type de mélange étudié.

Le problème de la séparation de sources dans le cas d'un canal instantané, connu aussi sous le nom d'analyse en composantes indépendantes ACI (ou ICA, acronyme de l'anglais Independent Component Analysis) a donné lieu à d'importants travaux. En général pour réaliser la séparation, le recours aux statistiques d'ordre supérieur est souvent nécessaire. Dans [55] Cardoso a proposé de considérer les propriétés algébriques des cumulants d'ordre quatre en utilisant l'information d'ordre deux pour rendre unitaire la matrice de mélange puis de l'identifier par des techniques de diagonalisation. En revanche, il montre dans [56] que les cumulants d'ordre quatre sont suffisants pour permettre, à eux seuls, d'estimer la matrice de mélange. En 1993, il publie l'algorithme, très connu, JADE [57].

En plus des méthodes développées spécialement pour la séparation de sources, plusieurs méthodes sont issues de l'extension des algorithmes d'égalisation aveugle. C'est notamment dans ce contexte, que des travaux ont été menés dans l'équipe SCEE.

Parmi ces méthodes, les deux algorithmes les plus connus sont l'algorithme MUK (Multiuser Kurtosis maximization) [58] et le CMA (Constant Modulus Algorithm) [59] [60]. L'algorithme MUK est basé sur la maximisation du kurtosis sous contrainte d'orthonormalité.

L'algorithme CMA est basé sur le critère du module constant qui est adapté au cas des signaux PSK [61], [62], [63]. Une version modifiée du CMA appelée MMA (pour Multimodulus Algorithm) a été proposée pour améliorer l'égalisation des signaux utilisant des constellations QAM denses [64] ; son extension au cas MIMO a été proposé dans [65]. A.IKHLEF [25] de l'équipe SCEE a proposé l'utilisation des algorithmes à norme constante (CNA) [66] dans les techniques de séparation aveugle de sources à la démodulation numérique MIMO. Dans la suite de cette section, l'algorithme CMA sera précisément décrit car celui-ci sera implanté dans notre architecture présentée au chapitre 3.

2.6.2 Le critère CM

Le critère du module constant (CM) est probablement le critère le plus étudié dans la littérature. Il a reçu beaucoup d'intérêt dans la communauté des communications numériques car beaucoup de signaux de communications possèdent la propriété de module constant. L'origine du CM remonte aux travaux de Godard [61] et Treichler et al. [62]. La première application était l'égalisation aveugle. Ensuite les auteurs dans [63] l'ont utilisé dans le beamforming et plus récemment il a été appliqué à la séparation de sources [59] [60]. Une multitude d'algorithmes pour implémenter ce critère ont été proposés et jusqu'à maintenant on voit de nouveaux travaux sur ce critère et ses multiples applications. La raison du succès du critère CM est essentiellement liée à sa simplicité et sa robustesse mais aussi à son efficacité même dans le cas des signaux à modules non constants.

Nous allons présenter brièvement le critère CM et étudier ensuite l'utilisation de l'opérateur CORDIC sur l'algorithme CMA à base de gradient stochastique (SG-CMA) dans ce paragraphe.

Pour un système MIMO présenté dans le paragraphe précédent, l'idée du critère CM est de trouver un séparateur \mathbf{W} tel que les signaux à ses sorties du filtre BSS (Blind Source Separation) $\mathbf{z}(n) = \hat{\mathbf{s}}(n) = \mathbf{W}^T \mathbf{y}(n)$ aient un module constant, typiquement égale à un. Si c'est le cas, le signal récupéré à une des sorties est égal à un des signaux sources avec un déphasage arbitraire inconnu.

Pour calculer le séparateur \mathbf{W} , la propriété du module constant est exploitée dans le but de construire un critère dont la minimisation résulte en ce séparateur. Le critère CM est donné par :

$$\min_W \Gamma_{cma} = \sum_i^M E [(|z_i(n)|^2 - R)^2] \quad (2.3)$$

où $z_i(n)$ est la i-ème sortie du séparateur et R est la constante de dispersion, elle est calculée en supposant une séparation parfaite par rapport à la solution ZF, et est définie comme :

$$R = \frac{E[|s(n)|^4]}{E[|s(n)|^2]} \quad (2.4)$$

Au lieu d'utiliser l'intercorrélation, pour avoir des sorties différentes, Papadias [67] a proposé d'utiliser une contrainte d'orthonormalité. Le critère s'écrit ci-dessous :

$$G^H G = I. \quad (2.5)$$

où $G^T = W^T H$ est la matrice globale du système qui représente la cascade canal séparateur. La contrainte d'orthonormalité est issue de :

$$E[z(n)z(n)^H] = I. \quad (2.6)$$

où on force la matrice d'autocovariance de $z(n)$ à être une matrice identité. Ceci signifie que les éléments en dehors de la diagonale, qui représentent les intercorrélations entre les sorties du séparateur, doivent être nuls. Ce qui permet d'avoir un signal source différent à chaque sortie du séparateur. En l'absence de bruit, 2.6 se réduit à $G^H G = I$. En plus, si on considère un préblanchiment ou on suppose que la matrice de canal est unitaire la contrainte se simplifie encore à : $W^H W = I$. Dans la suite, c'est le critère CM avec une contrainte d'orthonormalisation qui sera considéré.

2.6.2.1 L'algorithme CMA à base de gradient stochastique (SG-CMA)

Depuis les travaux de Larimore et Treichler [62], plusieurs algorithmes basés sur la minimisation du critère CM ont été proposés. Nous nous intéressons dans cette thèse à l'algorithme CMA adaptatif à base de gradient stochastique.

Le premier algorithme CMA [62], dans le contexte SISO, était implémenté en adaptatif via l'algorithme du gradient stochastique. La minimisation du critère CM dans l'équation 2.3 en utilisant l'algorithme du gradient stochastique résulte en l'équation de mise à jour suivante :

$$W(n+1) = W(n) - \nabla_W \Gamma_{cma} = W(n) - \mu [\Delta_1(n), \dots, \Delta_M(n)] y(n)^* \quad (2.7)$$

où μ est le pas d'adaptation et

$$\nabla_W \Gamma_{cma} = (|z_i(n)|^2 - R) z_i(n) y(n)^* = \Delta_i(n) y(n)^*. \quad (2.8)$$

La contrainte dans l'équation 2.5 est satisfaite à chaque itération de l'algorithme adaptatif via l'orthonormalisation rapide [68]. Mais pour pouvoir l'appliquer à W , la matrice de

canal doit être unitaire ou bien le signal reçu doit être blanchi. Dans ce cas la contrainte se simplifie à : $W^H W = \mathbf{I}$. Alors, l'orthonormalisation rapide peut s'appliquer à W.

2.7 Description de l'algorithme "V-BLAST Square Root"

2.7.1 Principe du récepteur V-BLAST

Le principe de l'algorithme du récepteur V-BLAST a été présenté dans [19]. Il s'agit d'un égaliseur à retour de décision, adapté à la structure des systèmes MIMO. L'égaliseur peut utiliser soit le critère du forçage à zéro, soit le critère du MMSE. L'objectif est de réduire la complexité de l'algorithme en garantissant un niveau de performance acceptable. Son principe est le suivant : le symbole de l'émetteur le plus favorisé (possédant le meilleur TEB suivant le critère considéré) est démodulé en premier. Sa contribution au vecteur reçu r est ensuite annulée, ce qui augmente le SNR sur les autres émetteurs (à chaque bonne décision). Cette étape est répétée jusqu'au dernier symbole, le moins favorisé. Les étapes de la réception d'un vecteur sont les suivantes :

- Choix de l'antenne i correspondant au meilleur SNR.
- Le symbole s_i de l'émetteur i choisi est démodulé. Puisque le SNR de l'émetteur i est le plus élevé, la probabilité d'erreur de s_i est la plus faible, donc il va être démodulé en premier.
- En multipliant la $i^{\text{ème}}$ colonne de H avec s_i , on obtient la contribution de s_i sur chaque composante du vecteur reçu r .
- Cette contribution est soustraite du vecteur r , afin d'obtenir ce qui aurait été reçu en l'absence du symbole s_i . Cette opération améliore le SNR des autres symboles, à condition que la bonne valeur de s_i ait été choisie [41].
- La i^{e} colonne de la matrice H est forcée à zéro, ce qui consiste à définir un nouveau canal entre toutes les antennes sauf le i^{e} émetteur. Cette matrice devient la nouvelle matrice de canal pour l'itération suivante, le i^{e} symbole ayant déjà été démodulé.
- Les étapes au-dessus sont répétées N_r fois, jusqu'à ce que tous les symboles soient démodulés.

L'algorithme de détection peut être décrit comme suit sous forme récursive dans le cas où $N_r = N$: [40] :

```

Initialisation  $W^T = H^+$ 
Itération Pour i = 1 à N
 $k_i = \operatorname{argmin}_{j \leq N, j \notin k_1 \dots k_{i-1}} \|(W_i^T)\|^2$ 
 $y_{ki} = w_{ki}^T r_i$ 
 $s_{ki} = \text{décision}(y_{ki})$ 
 $r_{i+1} = r_i - s_{ki} H_{ki}$ 
 $W_{i+1} = (H_{ki}^+)^-$ 
Fin

```

On considère dans cet algorithme les notations suivantes :

- Le vecteur $k = k_1, k_2, \dots, k_N$ détermine l'ordre de détection selon les éléments de la matrice W^T . Des symboles \hat{s}_i et des colonnes de la matrice H permutent de la même façon.
- $(H_{ki}^+)^-$ indique que l'on a annulé la contribution des i premiers émetteurs de telle sorte que les colonnes k_1 à k_i ont été mises à 0.
- H^+ dénote la matrice pseudo inverse (Moore-Penrose pseudo inverse) du canal au sens du MMSE.
- $j \notin k_1 \dots k_j$ signifie que k_i n'appartient pas à la rangée utilisée ultérieurement.
- s_{ki} caractérise le $k_i^{ème}$ symbole estimé.
- r désigne le signal reçu et y_{ki} représente un scalaire sur lequel s'effectue la décision.

Cet algorithme améliore nettement les performances du récepteur, sans trop augmenter sa charge de calcul. Toutefois cette charge est alourdie par le calcul répété du pseudo-inverse de la matrice du canal lorsqu'un grand nombre d'antennes est employé.

Dans l'objectif de chercher le compromis parfait entre la meilleure détection possible et la faible complexité de l'algorithme de décodage, de nombreux auteurs ont proposé des améliorations de l'algorithme, les uns sont basés sur une décomposition QR itérative comme M.DAMEN [44] ou D.Wubben. Les autres utilisent des méthodes de substitution comme W.Zha et D.Blostein [69]. Mais le TEB est dégradé dans ces méthodes.

2.7.2 Base de l'algorithme "V-BLAST Square Root"

Dans le but de diminuer la complexité de l'algorithme sans dégrader le TEB, B.Hassibi présente l'algorithme "V-BLAST Square Root". Cet algorithme permet de diminuer la charge de calcul de $O(M^4)$ à $O(M^3)$ dans le cas où $M = N$. La stabilité numérique des transformations orthogonales utilisées dans cet algorithme le rend plus adapté à une implémentation en virgule fixe.

Considérons la matrice de covariance P qui est une estimation de l'erreur $s - \hat{s}$ et qui s'exprime par la relation :

$$E(s - \hat{s})(s - \hat{s})^* = (\sigma^2 I + H^* H)^{-1} = P.$$

Le symbole correspondant au signal le plus fort doit être placé en dernière position, car la covariance P_{ij} est la plus faible. La soustraction de sa contribution se fait par l'équation suivante :

$$\begin{aligned} r - h_M S_M &= H^{M-1} s^{M-1} + v \\ \text{où } H^{M-1} &= [H_1 \dots H_{M-1}], \\ s^{M-1} &= [s_1 \dots s_{M-1}]^T \end{aligned}$$

L'algorithme peut s'écrire de la façon suivante :

Pour i = 0 à M-1

- $P^{M-i \times M-i} = (\sigma^2 I + H^{*i \times i} H^{i \times i})^{-1}$
- $W^T = (\sigma^2 I + H^{*i \times i})^{-1} H^*$
- $k_i = \arg \min \text{diag}(P)$
- permutation $h_k(i), h_i$
- permutation $\hat{s}_{k(i)}, \hat{s}_i$
- $\tilde{y}_{k(i)} = W_{k(i)}^T r$
- $\hat{s}_{k(i)} = \text{décision}(\tilde{y}_{(i)})$

– $r = r - h_{(i)} \hat{s}_{(i)}$.

Fin.

Les notations utilisées dans cet algorithme sont définies comme suit :

- Le vecteur $k = k_1, k_2, \dots, k_N$ détermine l'ordre de détection des symboles \hat{s}_i des colonnes de la matrice H .
- k_i représente l'élément d'indice i du vecteur k .
- $\arg \min \text{diag}(P)$ détermine le plus petit élément de la diagonale de la matrice de covariance P .
- $W_{k(i)}^T$ représente la $k_i^{\text{ème}}$ ligne de la matrice W^T .
- $s_{k(i)}$ caractérise le $k_i^{\text{ème}}$ symbole estimé.
- permutation. désigne la permutation des colonnes de la matrice H , et des indices des symboles \hat{s} correspondant.

Le principe de cet algorithme réside dans une approche plus matérielle que celle décrite par Golden en ce qui concerne la recherche du nulling vector et de son ordonnancement. Sa complexité s'explique par la nécessité du calcul du pseudo inverse de la matrice H . Si on utilise une décomposition SVD la complexité correspond à : $N_t^2 N_r^2 + 2N_r N_t^3 + 15/4N_t^4$. Celle-ci représente près de 86% de la part des calculs de cette architecture. L'objectif est donc de diminuer considérablement la complexité des calculs liée à la recherche du nulling vector et de l'ordonnancement tout en conservant la performance. La section suivante va expliquer comment atteindre cet objectif.

2.7.3 Algorithme "V-BLAST Square Root"

Dans le but d'utiliser une transformation unitaire, il faut donc calculer $P^{1/2}$, nous faisons d'abord la décomposition QR de la matrice H augmentée de la quantité de bruit notée H_α .

$$H_\alpha = \begin{bmatrix} H^{N \times M} \\ \sqrt{\sigma} I^{M \times M} \end{bmatrix} = QR = \begin{bmatrix} Q_\alpha^{N \times M} \\ \times \end{bmatrix} R^{N \times M} \quad (2.9)$$

Où \times dénote les coefficients de la matrice H non utilisés.

R est une matrice triangulaire supérieure et Q une matrice unitaire.

Le pseudo inverse de la matrice H augmenté de la quantité de bruit s'écrit :

$$H_\alpha^+ = (\sigma^2 I + H^* H)^{-1} H^*. \quad (2.10)$$

Nous pouvons maintenant calculer P sans utiliser la décomposition de Choleski :

$$P = (\sigma^2 I + H^* H)^{-1} = (R^* R)^{-1} = R^{-1} R^{-*}. \quad (2.11)$$

Si nous multiplions H_α par sa matrice transposée nous obtenons une décomposition de Choleski.

$$H_\alpha^+ \times H_\alpha = R^* Q^* Q R = R^* R = \begin{bmatrix} H^* \sqrt{\sigma^2 I} \\ \sqrt{\sigma^2 I} \end{bmatrix} \begin{bmatrix} H \\ \sqrt{\sigma^2 I} \end{bmatrix} = (\sigma^2 I + H^* H)^{-1} H^*. \quad (2.12)$$

On peut identifier R^{-1} à la racine carrée de covariance P par la relation suivante :

$$R^{-1} = P^{1/2}. \quad (2.13)$$

En utilisant les relations (3.2.3.5) et (2.11) dans l'équation (2.10) il devient :

$$H_\alpha^+ = R^{-1} R^{-*} (QR)^* = R^{-1} R^{-*} R^* Q^* = R^{-1} Q^*. \quad (2.14)$$

On déduit le pseudo inverse de la matrice en se reportant à la relation (2.13) :

$$H_\alpha^+ = \left[\frac{H}{\sqrt{\sigma^2 I}} \right]^+ = R^{-1} Q^* = P^{1/2} Q^*. \quad (2.15)$$

Le vecteur des symboles estimés peut donc se déduire de l'expression suivante :

$$\hat{s} = H_\alpha^+ r = P^{1/2} Q^* r. \quad (2.16)$$

Comme nous venons de démontrer que la matrice de covariance et la matrice pseudo inverse peuvent être déduites des matrices $P^{1/2}$ et Q , il reste donc à déterminer l'ordre de décodage des symboles et calculer les coefficients de l'égaliseur optimal.

L'ordre de décodage des symboles est déterminé par la norme minimale des lignes de la matrice $P^{1/2}$ ce qui correspond au plus petit élément de la diagonale de la matrice de variance P . Les indices des symboles estimés sont arrangés de telle façon que le plus petit élément de la diagonale de la matrice P se trouve en dernière ligne. Une transformation unitaire notée \sum est ensuite utilisée pour trouver l'élément $p_i^{1/2}$ à partir de $P^{1/2}$. La transformation peut être exécutée par une séquence de rotation de Givens qui est bien adaptée à l'implémentation matérielle du point de vue complexité. Cette transformation effectue une triangulation supérieure par bloc. Nous aborderons dans le prochain chapitre le détail de la transformation unitaire. La séquence de rotation Givens s'écrit :

$$P_i^{1/2} \Sigma_i = \begin{bmatrix} P_{i-1}^{1/2} & \times^{i-1 \times 1} \\ 0 & p_i \end{bmatrix} \quad (2.17)$$

- \sum représente une transformation unitaire,
- $P_i^{1/2}$ désigne le $i^{\text{ème}}$ élément de la diagonale de la matrice $P^{1/2}$,
- p_i désigne un nombre scalaire.

Le vecteur des coefficients optimaux est défini par l'expression suivante :

$$H_{\alpha,i}^+ = p_i^{1/2} q_i^*. \quad (2.18)$$

Nous venons de démontrer comment calculer $p_i^{1/2}$ par l'expression 4.2.5. Il reste q_i^* à calculer. Pour cela une triangulation supérieure de la matrice Q est effectuée pour éviter le calcul de la pseudo inverse.

L'algorithme précédent se présente maintenant sous la forme suivante :

1. Calculer des matrices $P^{1/2}$ et Q .
2. Calculer la norme minimale des lignes de $P^{1/2}$ et permutez cette ligne pour qu'elle soit la dernière. Ainsi que l'indice du symbole reçu et de la matrice H .

3. Appliquer une transformation unitaire pour satisfaire la relation 4.2.5 telle que :

$$P_i^{1/2} \Sigma_i = \begin{bmatrix} P_{i-1}^{1/2} & \times^{i-1 \times 1} \\ 0 & p_i \end{bmatrix} \quad (2.19)$$

4. Actualiser de la matrice Q_α à partir de la même transformation unitaire que l'étape précédent notée Σ telle que :

$$Q_\alpha = Q_\alpha \times \Sigma. \quad (2.20)$$

5. Calculer le vecteur des coefficients optimums tel que :

$$W_i^T = H_{\alpha,i}^+ = p_i^{1/2} q_i^*. \quad (2.21)$$

6. Décoder le symbole estimé et retirer sa contribution.

7. Retourner à l'étape (3) avec $P_{i-1}^{1/2}$ et $Q_{\alpha,i-1}$.

Comme nous pouvons le voir l'algorithme décrit ci-dessus réduit la complexité et augmente l'efficacité de calcul. Il évite le calcul répétitif du pseudo inverse en utilisant les transformations unitaires. Mais dans le calcul de la matrice $P^{1/2}$, il nécessite d'effectuer une inversion de la matrice R après la décomposition QR de la matrice H_α . Ceci induit des difficultés pour l'implémentation matérielle.

2.7.4 Optimisation de l'algorithme "V-BLAST Square Root"

Dans le but de calculer $P^{1/2}$ sans pour cela inverser la matrice R , B.Hassibi utilise une relation de récurrence connue en filtrage adaptatif RLS (Recursive least squares). Il démontre que si on applique une séquence de rotation de Givens à la relation de récurrence, on obtient $P^{1/2}$ après i itérations.

$$\begin{bmatrix} 1 & (H)_i P_{i-1}^{M \times M} \\ 0^{M \times 1} & P_{i-1}^{M \times M} \end{bmatrix} \Theta_i = \begin{bmatrix} \times 0^{1 \times M} \\ \times P_i^{M \times M} \end{bmatrix} \quad (2.22)$$

- $P_i^{1/2}$ représente la matrice $P^{1/2}$ à la $i^{\text{ème}}$ itération.
- Θ_i représente la $i^{\text{ème}}$ séquence de la rotation de Givens et Θ correspond à la matrice unitaire.
- $P^{1/2}$ est initialisée à $\frac{1}{\sqrt{\sigma^2}} I$ où I est la matrice identité.
- \times dénote les entrées non utilisées.

Toutefois il reste alors à calculer Q . La meilleure solution sera d'appliquer une relation qui fournit $P^{1/2}$ et Q_α en même temps. C'est la raison pour laquelle B.Hassibi met en évidence une nouvelle relation de récurrence à partir du bloc matrice précédent auquel il ajoute un bloc vecteur.

$$\begin{bmatrix} 1 & (H)_i P_{i-1}^{M \times M} \\ 0^{M \times 1} & P_{i-1}^{M \times M} \\ -e_i^{N \times 1} & Q_{i-1}^{N \times M} \end{bmatrix} \Theta_i = \begin{bmatrix} \times 0^{1 \times M} \\ \times P_i^{M \times M} \\ \times Q_i^{N \times M} \end{bmatrix} \quad (2.23)$$

En conclusion, l'algorithme "V-BLAST Square Root" proposé par B.Hassibi [49], évite le calcul répété du pseudo-inverse de la matrice du canal et l'inversion matricielle en

utilisant les transformations unitaires. Ceci permet de diminuer la charge de calcul de $O(M^4)$ à $O(M^3)$ sans dégrader le TEB.

L'algorithme est résumé ci-dessous :

Etape 1 : Calcul de $P^{1/2}$ et Q_a : pour $i = 1, 2, \dots, N$:

$$\begin{bmatrix} 1 & (H)_i P_{i-1}^{M \times M} \\ 0^{M \times 1} & P_{i-1}^{M \times M} \\ -e_i^{N \times 1} & Q_{i-1}^{N \times M} \end{bmatrix} \Theta_i = \begin{bmatrix} \times 0^{1 \times M} \\ \times P_i^{M \times M} \\ \times Q_i^{N \times M} \end{bmatrix} \quad (2.24)$$

Dans cette équation, $P_0^{1/2} = \beta I$, $Q_0 = 0^{N \times M}$, e_i est la $i^{\text{ème}}$ colonne de la matrice identité, Θ_i correspond à une transformation unitaire et \times dénote les entrées non utilisées. Après N itérations on obtient $P_N^{1/2} = P^{1/2}$ et $Q_\alpha = Q_N$.

Etape 2 : Déterminer la norme minimale des lignes de $P^{1/2}$ et permuter cette ligne pour qu'elle soit la dernière. Permuter également l'indice du symbole reçu en rapport. Effectuer une transformation unitaire qui satisfasse la relation 2.25. Pour $i = M, M-1, \dots, 1$:

$$P_i^{1/2} \Sigma_i = \begin{bmatrix} P_{i-1}^{1/2} & \times^{i-1 \times 1} \\ 0 & p_i \end{bmatrix} \quad (2.25)$$

Etape 3 : Actualiser Q_a) à partir de $Q_a \Sigma_i$,

Etape 4 : Calculer le nulling vector :

$$w_i = p_i q_{\alpha,i}^* \quad (2.26)$$

Etape 5 : Calculer $y_i = w_i r$

Etape 6 : Calculer les symboles estimés $\hat{s}_i = \text{décision}(y_i)$

Etape 7 : Annuler les interférences :

$$r = r - s_i (H)_i \quad (2.27)$$

2.8 Description fonctionnelle de l'algorithme "V-BLAST Square Root"

La description fonctionnelle du décodeur MIMO " V-BLAST square-root " est illustrée par la figure 2.10. Elle est composée de 6 modules de traitement. Les entrées sont constituées des messages reçus r et des valeurs de la matrice de canal H . Les trois premiers modules (M_1, M_2, M_3) effectuent la décomposition de la matrice H en utilisant des transformations unitaires.

Ces modules calculent les grandeurs $P^{1/2}$ (Etape 1), Q_α (Etape 1), p_i (Etape 2) et $q_{\alpha,i}^*$ (Etape 3). Le module suivant M_4 détermine l'ordre optimal de décodage et calcule le vecteur w_i (nulling vector) (Etape 4). Le module M_5 décide du vecteur symbole transmis (Etape 6) et le dernier module M_6 réalise l'annulation d'interférences entre les symboles (Etape 7).

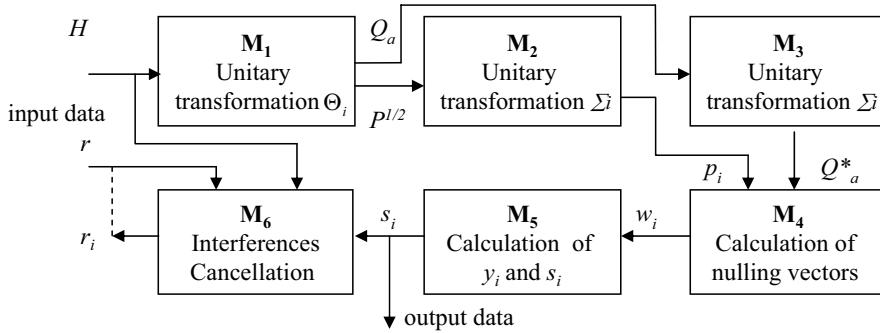


FIG. 2.10 – Architecture fonctionnelle du décodeur MIMO "V-Blast square-root "

2.9 Conclusion

Après avoir introduit les architectures reconfigurables pour les systèmes MIMO, nous avons d'abord effectué un état de l'art du domaine. Les différents types de systèmes MIMO sont présentés dans ce chapitre. Les divers algorithmes de réception sont décrits et comparés en terme de TEB et de complexité. Nous avons distingué deux types d'algorithmes de réception en fonction de la connaissance du canal. L'algorithme V-BLAST SRA est présenté dans ce chapitre comme assurant le meilleur compromis entre la performance en TEB et la complexité. En effet, l'algorithme V-BLAST SRA permet de diminuer la charge de calcul de l'algorithme V-BLAST classique : les calculs du pseudo-inverse de la matrice du canal sont évités par l'utilisation des transformations unitaires de la matrice. L'algorithme V-BLAST Square Root est donc décrit en détail et comparé à l'algorithme V-BLAST classique.

Afin de montrer que notre architecture reconfigurable est capable de supporter différents algorithmes MIMO, nous avons aussi décrit le principe des algorithmes MIMO MMSE SRA et MMA. Nous montrerons la mise en oeuvre de ces algorithmes sur notre architecture reconfigurable dans le chapitre 3.

L'utilisation des transformations unitaires dans les structures de ces algorithmes MIMO permet de mettre en place une architecture à base d'opérateur CORDIC. Nous nous intéressons maintenant à l'opérateur CORDIC utilisé dans de nombreuses fonctions de traitement du signal et notamment dans le cadre des applications MIMO. Nous allons aussi étudier l'utilisation de cette architecture pour d'autres algorithmes MIMO.

Chapitre 3

Architecture reconfigurable à base d'opérateur CORDIC

Sommaire

3.1	Introduction	38
3.2	Opérateur CORDIC	39
3.2.1	Algorithme CORDIC	39
3.2.2	Architecture de l'opérateur CORDIC	44
3.2.3	Applications	47
3.3	Proposition d'architecture CBDRA	49
3.3.1	Contrainte de notre problème	49
3.3.2	Idée de base	50
3.3.3	Relation théorique entre le nombre d'opérateur CORDIC et les caractéristiques des applications	50
3.3.4	Architecture CBDRA	51
3.4	Architecture reconfigurable CBDRA	52
3.4.1	Structure de FPGA	53
3.4.2	Reconfiguration dynamique de FPGA	54
3.4.3	Flots de conception du FPGA	56
3.4.4	Connexion entre la partie statique et la partie reconfigurable	59
3.4.5	Gestion de reconfiguration	60
3.4.6	Architecture CBDRA et reconfiguration statique	60
3.4.7	Architecture CBDRA et reconfiguration dynamique des interconnexions	61
3.5	Application de l'architecture CBDRA pour un décodeur MIMO V-BLAST Square Root	62
3.5.1	Stationnarité du canal	63
3.5.2	Implémentation parallèle de CORDIC (Un exemple : calcul de $P^{1/2}$ et de Q_α)	63
3.5.3	Relation théorique entre le débit et le nombre d'opérateurs CORDIC utilisé	66
3.5.4	Exemple de résolution des équations et d'implémentation : calcul de $P^{1/2}$ et de Q_α	68

3.5.5	Utilisation de l'architecture CBDRA avec la reconfiguration dynamique	70
3.5.6	Implémentation de l'ensemble de l'algorithme	72
3.5.7	Simulation	74
3.6	Architecture CBDRA pour d'autres algorithmes MIMO	77
3.6.1	L'algorithme MIMO MMSE	77
3.6.2	L'algorithme CMA	78
3.7	Conclusion	81

3.1 Introduction

L'évolution des systèmes de communication dans le domaine du traitement du signal et de l'image, demande des algorithmes de plus en plus complexes, nécessitant des puissances de calculs au delà des performances des processeurs actuels. Les circuits spécifiques (ASIC) ont des avantages de rapidité de traitement et de faible coût, mais manquent de souplesse d'adaptation des algorithmes. Pour la réalisation des systèmes MIMO, les algorithmes sont généralement implémentés sur DSP (Digital Signal Processor), il est donc difficile d'atteindre des performances élevées en débit. La solution traditionnelle ASIC, quant à elle, ne supporte pas les évolutivités des futurs systèmes. Des solutions sont proposées comme les architectures multi-processeurs qui bénéficient de la souplesse de la programmation au prix d'un coût et d'un encombrement plus élevés. Aujourd'hui l'émergence d'un nouveau type d'architecture, l'architecture reconfigurable, offre une solution intermédiaire où la flexibilité de la programmation et la puissance de calcul des architectures spécialisées sont alliées.

Les premiers concepts des architectures reconfigurables datent du début des années 1980 avec la naissance de la technologie FPGA. Depuis, les travaux en matière d'architecture reconfigurable sont très nombreux. Grâce à l'amélioration des techniques d'intégration, la notion d'architecture reconfigurable a dépassé le cadre unique des circuits de type FPGA et correspond maintenant à un ensemble de cibles hétérogènes. En effet, l'espace entre ASIC et processeurs laisse place à d'autres paradigmes architecturaux qui offrent donc de nouvelles alternatives entre la grande flexibilité des solutions programmables et les hautes performances des circuits spécifiques. Ces paradigmes architecturaux sont nommés architecture reconfigurable ou "*Configurable Computing*".

La notion d'architecture reconfigurable se prête à des définitions multiples qui varient d'un article à l'autre. Il faut donc dans un premier temps définir ces termes. Les architectures reconfigurables peuvent être classées suivant cinq caractéristiques essentielles : la granularité des ressources de traitement, la granularité des ressources de communications, la topologie du réseau de communication, la liaison avec un éventuel processeur et le type de reconfiguration.

En 2001, R.Hartenstein a montré une étude rétrospective sur des projets concernant les architectures reconfigurables à gros grain [70]. La classification des architectures reconfigurables est définie par la topologie du réseau de routage : topologie matricielle, linéaire ou hiérarchique. D'autres études présentent aussi d'autres façons de classer ces architectures. Par exemple : les caractéristiques de granularité, de couplage avec un processeur et de reconfigurabilité sont proposées par Ben Chehida [71]. Plus récemment, une classification des architectures reconfigurables suivant les caractéristiques de granularité des ressources

de traitement est proposée par L. Bossuet [72]. Trois types peuvent être distingués suivant trois branches : grain fin, gros grain et multi-grains.

Nous pouvons citer par exemple les architectures de type gros grain pour les applications radio communication. Les architectures DReAM [73] et Morphosys [74], RaPiD [75] ou le KressArray [76] utilisent un réseau d'interconnexion à deux dimensions, éventuellement hiérarchique. Un récepteur *rake* pour l'UMTS a été réalisé sur DReAM [77] et un récepteur OFDM sur Morphosys est décrit dans l'article [78]. L'architecture SystolicRing [79], [80] reconfigurable dynamiquement pour les systèmes sur puce a été conçue par le laboratoire LIRMM de l'université de Montpellier ; l'architecture DART [81] [82] du laboratoire R2D2 de IRISA/Enssat de Lannion a été développée pour les applications multimédias.

Cependant, dans le cas de structures reconfigurables à gros grain les outils sont moins avancés. Une majorité des architectures commerciales sont réalisées autour d'une matrice d'éléments reconfigurables à grain fin de type FPGA. Effectivement, les FPGA sont les circuits les plus répandus et disponibles sur le marché. Ils offrent la possibilité de concevoir des architectures hétérogènes SoC/SoPC intégrant des processeurs embarqués, de large capacité mémoire et des accélérateurs dédiés. Nous nous sommes donc particulièrement intéressés à ce type de circuit et à leur mise en oeuvre pour l'implémentation des systèmes MIMO.

Il est à noter qu'une augmentation de la granularité peut être constatée : par exemple, des multiplieurs câblés 18×18 sur des Xilinx Virtex-II et des blocs DSP48 sur Virtex-4, ou encore un ou plusieurs coeurs de microprocesseur embarqués au plus près de la matrice reconfigurable sur Virtex-II Pro, Altera Excalibur ou Atmel FPSIC.

Dans ce chapitre, nous décrivons dans un premier temps l'opérateur CORDIC. Nous présentons l'architecture reconfigurable à base d'opérateurs CORDIC nommée CBDRA (CORDIC Based Dynamically Reconfigurable Architecture). Nous donnons en exemple des applications de l'architecture proposée, en particulier l'implémentation d'un décodeur MIMO V-BLAST Square Root.

3.2 Opérateur CORDIC

3.2.1 Algorithme CORDIC

L'algorithme CORDIC, (acronyme de COordinate ROTation DIgital Computing) a été créé à l'origine par J.E.Volder [83] pour effectuer des calculs tels que les rotations de vecteurs ou les changements de coordonnées cartésiennes-polaires et polaires-cartésiennes dans le plan euclidien, dans des applications de navigation aérienne. C'est une méthode simple et efficace pour le calcul d'une gamme de fonctions complexes qui s'appuie sur une technique d'additions et de décalage entre vecteurs. L'algorithme calcule par approximation la plupart des fonctions basées sur la trigonométrie. Il exécute des rotations sans utiliser d'opérations de multiplication. Un autre avantage de cet algorithme est qu'il permet d'obtenir une précision déterminée à l'avance en effectuant un nombre d'itération donné.

Nous pouvons citer par exemple des applications où l'algorithme CORDIC est utilisé : la modulation en bande latérale unique, la transformée de Fourier discrète, directe et rapide [84] [85], le filtrage fréquentiel (treillis de Gray-Marke, filtres orthogonaux et filtres d'onde [86]), la modélisation adaptative de processus non stationnaires (filtrage récursif optimal, filtre de Kalman [87]). Il intervient aussi dans la résolution d'un grand nombre de

problèmes d'algèbre linéaire comme les algorithmes orthogonaux de Givens [88], Fadeeva, décomposition en valeur singulières [89], décomposition QR et de Cholesky.

L'algorithme CORDIC a donc été conçu initialement comme un outil pour le calcul numérique. Nous nous intéressons particulièrement ici à l'implantation d'architectures VLSI spécialisées de traitement numérique du signal. En effet, il fait l'objet d'un regain d'intérêt depuis plusieurs années car il peut être considéré comme un opérateur générique pouvant être utilisé dans un grand nombre d'algorithmes classiques de traitement du signal.

3.2.1.1 Principe de l'algorithme CORDIC

L'algorithme CORDIC est basé sur des calculs de fonctions trigonométriques, son principe est d'effectuer des rotations sur un vecteur de base pour un angle donné. La transformation élémentaire des rotations s'exprime sous forme matricielle comme suit :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\sqrt{m}\theta & -\sqrt{m}\sin\sqrt{m}\theta \\ 1/\sqrt{m}\sin\sqrt{m}\theta & \cos\sqrt{m}\theta \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.1)$$

Le paramètre m peut prendre les trois valeurs suivantes :

- $m = 1$: cas circulaire, il permet de calculer différentes fonctions trigonométriques (sinus, cosinus, arctangente) ainsi que des racines carrées,
- $m = -1$: cas hyperbolique, il permet de calculer les fonctions sinus, cosinus et tangente hyperboliques ainsi que l'exponentielle,
- $m = 0$: cas linéaire il permet également le calcul direct de multiplications et de divisions.

Il est à noter qu'en combinant ces différentes fonctions on peut calculer de manière indirecte les tangentes circulaire et hyperbolique, ainsi que les logarithmes. La résolution de fonctions telles que sinus ou cosinus par l'algorithme de CORDIC s'appuie sur une méthode de rotation de vecteur dans le plan cartésien. Supposons la rotation du vecteur $V(x, y)$ d'un angle φ telle qu'illustrée à la figure 3.1.

Les coordonnées du vecteur V' sont exprimées selon les équations :

$$x' = x\cos(\varphi) - y\sin(\varphi) \quad (3.2)$$

$$y' = y\cos(\varphi) + x\sin(\varphi) \quad (3.3)$$

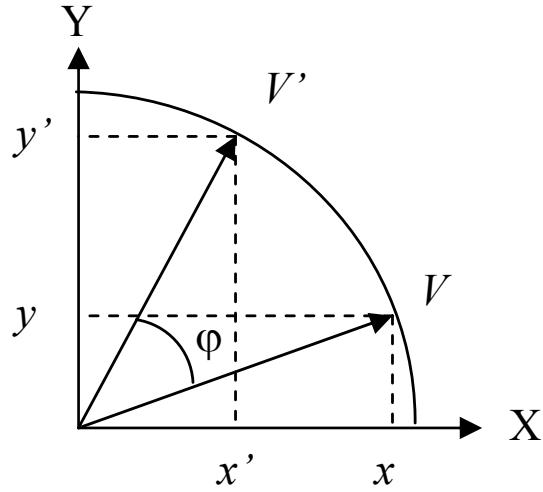
Si on restreint les angles de rotation à $\tan^{-1}(\pm 2^{-i})$ où $i = 0, 1, 2, 3, \dots$, on obtient alors φ par une série de rotations élémentaires successives de l'ordre de :

$$\theta_{i+1} = \theta_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (3.4)$$

où

$$d_i = \pm 1. \quad (3.5)$$

L'indice d_i indique le sens de rotation de l'angle pour chaque itération. Cet indice est déterminé à chaque itération selon le résultat d'une comparaison. Le mode d'utilisation

FIG. 3.1 – Rotation du vecteur V dans le plan cartésien

de l'indice sera expliqué un peu plus loin. Chaque vecteur itératif $V_{i+1}(x_{i+1}, y_{i+1})$ est représenté par :

$$x_{i+1} = K_i[x_i - y_i d_i(2^{-i})] \quad (3.6)$$

$$y_{i+1} = K_i[y_i - x_i d_i(2^{-i})] \quad (3.7)$$

où

$$K_i = \cos(\tan^{-1}2^{-i}) = (1 + 2^{-2i})^{-1/2} \quad (3.8)$$

Puisque pour un nombre relativement élevé d'itérations, le produit tend vers un résultat constant, il nous est possible d'appliquer ce dernier plus loin dans l'algorithme. En effet, pour une séquence donnée de rotations élémentaires, les facteurs K_i peuvent être regroupés et appliqués en une fois. Ainsi on obtient un ensemble d'équations simplifiées et spécifiques au calcul des opérations mathématiques recherchées :

$$x_{i+1} = x_i - d_i dy_i \quad (3.9)$$

$$y_{i+1} = y_i - d_i dx_i \quad (3.10)$$

où

$$dy_i = y_i 2^{-i} \quad (3.11)$$

$$dx_i = x_i 2^{-i} \quad (3.12)$$

$$d\theta_i = \tan^{-1}(2^{-i}) \quad (3.13)$$

$$d_i = \pm 1 \quad (3.14)$$

selon le signe de θ_i ou y_i .

TAB. 3.1 – Constante A_n en fonction du nombre d'étages

Etage	A_n
0	0.707106781
1	0.632455532
2	0.613571991
3	0.608833912
4	0.607648256
5	0.607351770
6	0.607277644
7	0.607259112
8	0.607254479
9	0.607253321
10	0.607253031
11	0.607252959
12	0.607252941
13	0.607252936
14	0.607252935
15	0.607252935

Puis nous calculons :

$$x' = A_n x \quad (3.15)$$

où la constante A_n ne dépend que de la séquence de rotations élémentaires donnée par :

$$A_n = \prod_{i=0}^n K_i, \quad (3.16)$$

Nous remarquons que le principal intérêt de cette constante est qu'elle ne dépend pas de θ mais uniquement du nombre d'étages. La valeur de la constante A_n en fonction du nombre d'étage est résumée dans le tableau 3.1. Pour un nombre croissant d'étages, cette constante tend vers la valeur égale à 0.607252935.

Ces équations servent donc au calcul d'opérations. Le principe général de l'algorithme de CORDIC consiste à faire pivoter dans le sens approprié le vecteur de rotation par un angle de plus en plus petit jusqu'à ce que l'angle θ ou les valeurs x et y soient approximativement égales à 0.

3.2.1.2 Les différents modes de fonctionnement de l'algorithme

On considère deux modes d'utilisation : le mode vecteur et le mode rotation. Le mode vecteur calcule simplement le module et la phase du vecteur à partir des coordonnées x et y. Le mode rotation résout, simultanément à partir d'un angle initial, le sinus et le cosinus de ce dernier.

3.2.1.3 Mode vecteur

De façon réciproque le mode vecteur résout simultanément, à partir des coordonnées x et y données, les fonctions $\text{atan}(y/x)$ et A_n . On initialise respectivement les variables x_0 , y_0 et θ_0 à x, y et 0. Dans ce cas, on effectue des rotations jusqu'à ce que la variable y s'annule. Le mode vecteur est défini par les équations suivantes :

$$\theta_{out} = \arctan(y_{in}/x_{in}) \quad (3.17)$$

$$\begin{bmatrix} x_{out} \\ y_{out} \end{bmatrix} = \begin{bmatrix} \text{sgn}(x_{in}) \bullet \sqrt{x_{in}^2 + y_{in}^2} \\ 0 \end{bmatrix} \quad (3.18)$$

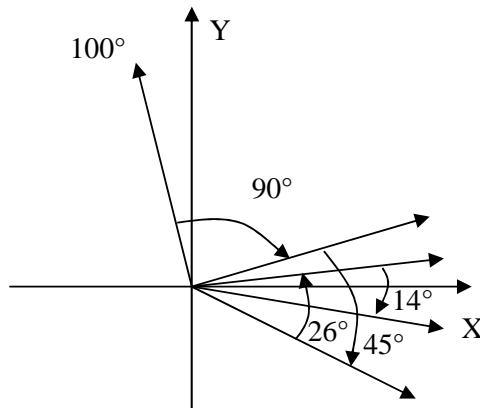


FIG. 3.2 – CORDIC en mode vecteur

En effet, nous disposons d'un vecteur dont nous cherchons à annuler la partie imaginaire par de multiples rotations. Des rotations par approximations successives sont appliquées au vecteur initial jusqu'à ce que ce dernier se confonde avec l'axe des abscisses. Le principe du mode vecteur est illustré sur la figure 3.2 en utilisant un vecteur initial d'angle égale à 100° . Une rotation de 90° est effectuée à la première itération suivie de rotations de 45° , 26° , 14° , 7° et ainsi de suite jusqu'à ce que le vecteur soit confondu avec l'axe des abscisses. Le tableau 3.2 résume le séquencement des rotations successives.

Il est à noter que la valeur absolue de la somme algébrique des rotations effectuées est égal exactement à l'angle du vecteur initial 100° .

TAB. 3.2 – Exemple de rotations successives en mode vecteur

Angles successifs	100°	+10°	-35°	-9°	5°	-2°	+1°	0°
Rotation successives	-90°	-45°	+26°	+14°	-7°	+3°	-1°	0°

TAB. 3.3 – Exemple de rotations successives en mode rotation

Angles successifs	+36°	+126°	81°	+107°	93°	+86°	89°	90°
Rotation successives	+90°	-45°	+26°	-14°	-7°	+3°	+1°	0°

3.2.1.4 Mode rotation

Dans ce mode de base, l'algorithme converge en faisant tendre la variable auxiliaire θ_0 , initialisée au départ à θ (argument de la rotation) vers zéro. Cette mise à jour est effectuée en parallèle avec les itérations de calcul des rotations, sous la forme :

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \bullet \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (3.19)$$

Prenons un exemple d'un vecteur de module 1 et d'angle initial de 36° et un angle de rotation de 54°. Les nouvelles coordonnées du vecteur seront de 1 en abscisse et 0 en ordonnée après avoir effectué quelques rotations. L'angle final sera :

$$36^\circ + 54^\circ = 90^\circ$$

La tableau 3.3 et la figure 3.3 résument les rotations successives.

Le signe de y est alors défini par celui de θ , ce qui correspond à une commande par rétroaction asservissant les itérations de manière à faire tendre θ vers 0. Les coordonnées du vecteur sont à chaque fois réactualisées et la somme des angles de rotation totalise une rotation de 54°. Dans notre système, l'angle de rotation est tout d'abord déterminé par le mode vecteur puis exécuté par le mode rotation.

Les résultats de convergence pour les deux modes de fonctionnement et les trois cas (valeurs de m) d'utilisation de l'algorithme sont résumés dans le tableau 3.4.

3.2.2 Architecture de l'opérateur CORDIC

Les différents choix pour implémenter l'algorithme CORDIC se différencient par les degrés de parallélisme, et les profondeurs de pipeline des additionneurs/soustracteurs utilisés dans l'algorithme [90], [91]. Le choix est guidé par un compromis entre la précision souhaitée, la vitesse de calcul nécessaire et le niveau de complexité demandé par l'application. On peut distinguer trois architectures en fonction du parallélisme sur les opérations et les itérations. En effet, la mise en parallèle de plusieurs additionneurs permet d'augmenter la vitesse de calcul d'un facteur fixe, mais multiplie d'autant le nombre d'opérateurs élémentaires nécessaires à la réalisation de la fonction.

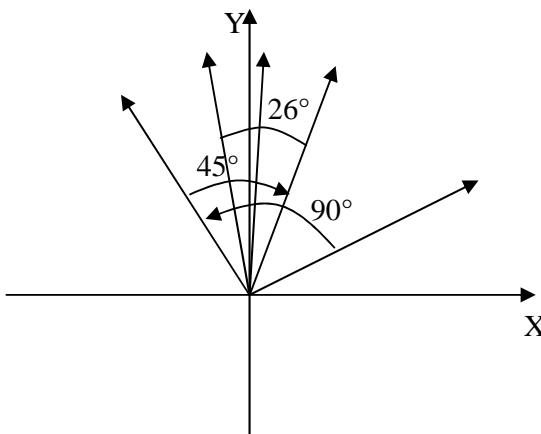


FIG. 3.3 – CORDIC en mode rotation

3.2.2.1 Architecture à opération série et itération série

Cette architecture n'utilise qu'un seul additionneur/soustracteur, comme illustrée par la figure 3.4, elle effectue donc des opérations séries et des itérations séries. C'est une architecture la plus simple en terme de complexité mais la moins performante. Les coordonnées x, y, et z sont calculées en séquentiel. Elle nécessite d'utiliser un séquenceur plus complexe par rapport aux autres architectures. Le séquencement nécessite un nombre important de multiplexeurs. Le principal avantage de cette architecture est sa complexité limitée en terme de ressource matérielle.

3.2.2.2 Architecture à opération parallèle et itération série

On peut mettre en oeuvre des additionneurs/soustracteurs parallèles pour réaliser le calcul des trois équations de l'algorithme, mais avec un calcul séquentiel des itérations de l'algorithme. La figure 3.5 illustre ce type d'architecture. Il faut noter qu'il est nécessaire de stocker les valeurs angulaires utilisées dans le calcul sur la variable auxiliaire z dans une mémoire ROM ou un registre.

3.2.2.3 Architecture à opération parallèle et itération parallèle

La figure 3.6 montre l'architecture à opération parallèle et itération parallèle. L'architecture à opération parallèle et itération parallèle présentée ici est très intéressante, car elle permet d'atteindre des débits de calcul beaucoup plus élevés. C'est la structure la plus complexe sur le plan matériel, mais elle autorise une mise en oeuvre pipeline, limitant ainsi la longueur des chemins critiques de façon à accroître la vitesse de fonctionnement. Dans cette structure, les décalages doivent être réalisés à l'aide de registres à chaque étage, comme illustrés sur la figure 3.6. Il faut aussi noter la nécessité de générer les signaux d'initialisation ou de remise à zéro de ces registres. Par ailleurs, cette architecture est particulièrement adaptée au mode de fonctionnement rotation. Nous utilisons une struc-

TAB. 3.4 – Mode de fonctionnement de CORDIC

Cas	Mode rotation	Mode vecteur
Circulaire	$x_n \rightarrow K(x_0 \cos z_0 - y_0 \sin z_0)$ $y_n \rightarrow K(x_0 \sin z_0 + y_0 \cos z_0)$ $z_n \rightarrow 0$	$x_n \rightarrow K \sqrt{x_0^2 + y_0^2}$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + \arctan(y_0/x_0)$
Hyperbolique	$x_n \rightarrow K(x_0 \cosh z_0 - y_0 \sinh z_0)$ $y_n \rightarrow K(x_0 \sinh z_0 + y_0 \cosh z_0)$ $z_n \rightarrow 0$	$x_n \rightarrow K \sqrt{x_0^2 - y_0^2}$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + \tanh^{-1}(y_0/x_0)$
Linéaire	$x_n \rightarrow x_0$ $y_n \rightarrow y_0 + z_0 x_0$ $z_n \rightarrow 0$	$x_n \rightarrow x_0$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + y_0/x_0$

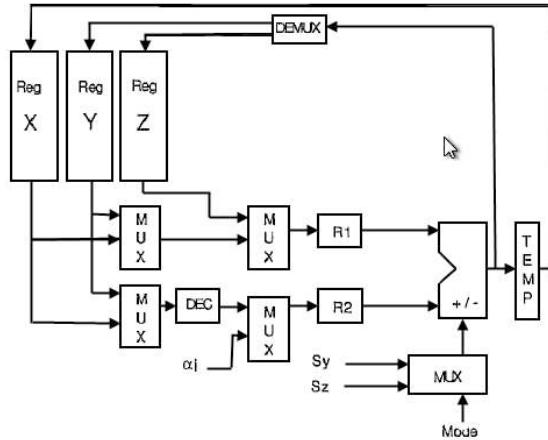


FIG. 3.4 – Architecture à opération série et itération série

ture pipeline de plusieurs étages dans notre implémentation. Nous montrerons l'impact du nombre d'étages sur le résultat de l'application dans le paragraphe 3.6.2.1.

3.2.2.4 Autre architectures et améliorations

Les architectures précédentes implémentent des additionneurs de type bit-parallèles effectuant les opérations sur des mots. Il est donc nécessaire d'utiliser des bus d'une largeur suffisante pour pouvoir présenter en entrée des additionneurs les mots entiers (respectivement x, y et z). Les architectures bit-série [90] sont proches de celles précédemment étudiées ; elles sont plus lentes dans l'absolu, mais plus faciles à optimiser par la mise en cascade notamment en calculant en premier le bit de signe pour initialiser l'étage suivant en additionneur ou soustracteur. Il est aussi possible d'améliorer le fonctionnement des architectures parallèles.

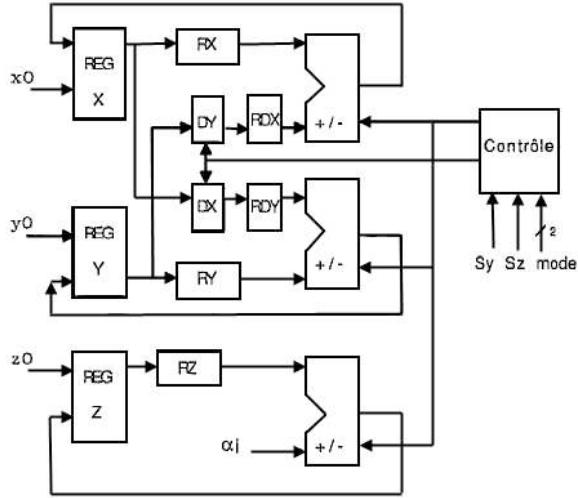


FIG. 3.5 – Architecture à opération parallèle et itération série

3.2.3 Applications

L'algorithme CORDIC peut être utilisé dans de nombreux algorithmes classiques de traitement du signal, ainsi que dans la radio logicielle [92] selon les différents modes.

3.2.3.1 Modulation numérique

Le principe des modulations numériques consiste en une modulation en bande de base sur deux voies, où l'on fait une multiplication par la porteuse complexe $e^{j2\pi f_0 t}$ pour aboutir à un signal modulé. Cette multiplication peut être calculée par l'opérateur CORDIC en mode rotation dans le cas circulaire. L'angle de rotation à effectuer est $2\pi f_0 t$. Il est à noter que le calcul de cet angle peut aussi être réalisé par un second opérateur CORDIC fonctionnant en mode linéaire.

3.2.3.2 Transformée de Fourier Discrète

La Transformée de Fourier Discrète d'un signal $x[n]$ s'exprime par :

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, k \in 0 \dots N \quad (3.20)$$

Il est possible d'utiliser une structure entièrement parallèle pour calculer les N^2 multiplications de la forme $x[n]e^{-j2\pi kn/N}$. Chaque CORDIC effectue une rotation d'un angle $2\pi kn/N$. Il est cependant difficile d'envisager l'utilisation des N^2 opérateurs CORDIC en parallèle pour des raisons de complexité. En pratique une structure de N opérateurs CORDIC en parallèle peut donc être utilisée pour diminuer la complexité. Le calcul est effectué en N itérations. De manière simple, un opérateur est associé à chacune des valeurs d'entrée, et l'on accumule les valeurs du terme $x[n]e^{j2\pi kn/N}$ pour chacun des $X[k]$ dans des

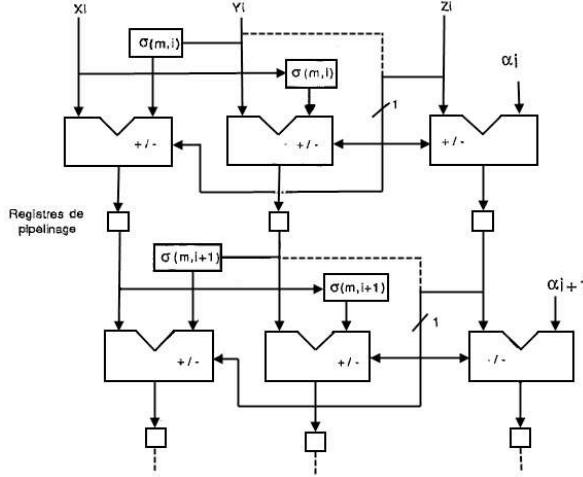


FIG. 3.6 – Architecture à opération parallèle et itération parallèle

registres. Il sera alors nécessaire de pipeliner le calcul par un décalage dans le temps des valeurs d'entrée pour obtenir le même débit. Dans l'article [85], une structure à base de transformées élémentaires d'ordre 4 est utilisée pour montrer l'utilisation de l'opérateur CORDIC. Nous avons analysé cette mise en oeuvre dans [93] en prenant deux exemples portant sur la FFT et la décomposition en valeur singulière (SVD). Nous les décrivons en détail dans les annexes A et B.

3.2.3.3 Décomposition en valeur singulière (SVD)

La décomposition en valeur singulière (SVD) d'une matrice $M \in C^{m \times n}$ est exprimée par l'équation suivante :

$$M = U \sum V^H, \quad (3.21)$$

où $U \in C^{m \times m}$ and $V \in C^{n \times n}$ sont des matrices unitaires et $\sum \in R^{m \times n}$ est une matrice diagonale.

La décomposition SVD peut être effectuée par l'algorithme CORDIC qui calcule l'arc tangente et effectue des rotations des vecteurs. Un processeur SVD a été réalisé par Z.Liu en utilisant l'opérateur CORDIC en virgule flottant [94]. Delosme [95], Yang et Bohme [96] ont démontré l'utilisation de CORDIC dans une transformation de la matrice 2×2 qui est nécessaire dans les calculs de la décomposition SVD d'une matrice réelle. Une approche basée sur l'opérateur CORDIC redondant et on-line est proposée récemment par Ercegovac et Lang [97]. Nous montrerons ces études en détails dans l'annexe B.

3.2.3.4 Algorithme des moindres carrés récursif

Les opérateurs CORDIC s'adaptent bien à l'implémentation de l'algorithme des moindres carrés récursif [98] et ses variantes, algorithmes utilisés dans le filtrage adaptatif [99] et la prédiction linéaire. Par exemple, dans le cas de l'algorithme de Levinson-Durbin, le

filtre de prédiction linéaire peut se reformuler récursivement en calculant conjointement les coefficients du filtre et l'erreur en sortie du filtre :

$$\begin{cases} e_{i+1} = e_i - k_i b_i = (\sqrt{1 - \theta_i^2})(e_i \cosh \theta_i + b_i \sinh \theta_i) \\ b_{i+1} = b_i - k_i e_i = (\sqrt{1 - \theta_i^2})(e_i \sinh \theta_i + b_i \cosh \theta_i) \end{cases}$$

en notant $\sinh \theta_i = \frac{-\theta_i}{\sqrt{1 - \theta_i^2}}$. Nous pouvons voir immédiatement l'implantation possible en utilisant l'algorithme CORDIC dans le cas hyperbolique.

De manière générale, les structures en treillis peuvent être réalisées au moyen d'opérateurs CORDIC, car il s'agit de transformations orthogonales sur des vecteurs de dimension 2. Le facteur de normalisation introduit en plus par CORDIC n'est pas un problème car nous pouvons le corriger et il apporte l'avantage de conserver la dynamique des variables lors de la transformation, ce qui permet de simplifier l'implémentation.

3.2.3.5 Rotations de Givens

La rotation de Givens [100] [88] est essentiellement utilisée afin d'obtenir une transformation unitaire visant à annuler les coefficients situés sur la partie triangulaire inférieure ou supérieure de la matrice. Nous allons décrire son principe. Considérons une matrice $G(i,j, \theta)$ telle que :

$$G(i,j, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & \sin \theta & 0 \\ 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tous les coefficients de cette matrice sont à zéro exceptés les coefficients (i,j) , (j,i) , (i,i) , et (j,j) qui sont respectivement à $\sin(\theta)$, $-\sin(\theta)$, $\cos(\theta)$, $\cos(\theta)$ et les autres éléments de la diagonale sont à 1. Lorsqu'un vecteur est multiplié par $G(i,j, \theta)$ alors le plan défini par les coordonnées (i,j) effectue une rotation d'un angle θ dans le sens trigonométrique. Si l'origine des coefficients du vecteur est représentée par (i,j) alors $\theta = \tan^{-1}(x_j/x_i)$. L'idée générale est illustrée dans l'exemple ci-contre sur une matrice 3×3 :

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} G(1,3, \theta) = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

Les rotations de Givens peuvent donc être réalisées par l'opérateur CORDIC en mode rotation. Les rotations de Givens sont utilisées dans l'algorithme V-BLAST Square Root pour effectuer les transformations unitaires.

3.3 Proposition d'architecture CBDRA

3.3.1 Contrainte de notre problème

La problématique de l'étude présentée dans ce document est la suivante :

1. Face à la multitude des standards de communication, est-il possible d'offrir une architecture reconfigurable à base d'opérateur commun CORDIC pour un système MIMO en supportant différents nombres d'antennes, différents types de modulations et de propagation ?
2. Afin d'optimiser les ressources matérielles, est-il possible de concevoir une architecture reconfigurable dynamiquement en exploitant la reconfigurabilité offerte par une plate-forme hétérogène ?
3. Cette architecture peut-elle être utilisée pour les différents blocs de l'algorithme MIMO ? Plus largement, peut-elle être appliquée à d'autres algorithmes de traitement du signal ?

3.3.2 Idée de base

La puissance et la générnicité de l'opérateur CORDIC sont illustrées dans le paragraphe précédent par des exemples de nombreux algorithmes de traitement du signal. L'utilisation de l'opérateur CORDIC présente également de nombreux avantages pour l'intégration au niveau purement architectural par son excellente adaptation aux contraintes des VLSI. Le partage des ressources et l'utilisation des propriétés des éléments de l'architecture mise en oeuvre permettent une meilleure exploitation des ressources d'un circuit. Les architectures reconfigurables ont des propriétés qui permettent la définition d'une solution à la fois flexible et efficace, tant au niveau de la surface que de la puissance de calcul. Elles permettent de partager des primitives de calcul entre plusieurs applications et peuvent donc théoriquement bénéficier de l'ensemble de niveau de granularité entre les applications citées précédemment. Cela nous amène à proposer une architecture appelée CBDRA (CORDIC Based Dynamically Reconfigurable Architecture) matérielle reconfigurable à base d'opérateurs CORDIC pour réaliser ces applications.

Nous proposons une structure de traitement où le nombre d'opérateurs CORDIC s'adapte aux caractéristiques des applications pour une implémentation optimale. Une architecture entièrement parallèle offre bien sûr une capacité de calcul importante mais au prix d'une très grande complexité. Au contraire une structure itérative construite autour d'un seul opérateur réduit fortement la complexité mais limite la cadence de traitement. C'est en combinant ces 2 approches que l'on obtiendra une architecture optimale, en étant guidé par les relations qui doivent liées les caractéristiques de la chaîne de traitement et l'architecture du décodeur (nombre d'opérateurs CORDIC et leur organisation).

3.3.3 Relation théorique entre le nombre d'opérateur CORDIC et les caractéristiques des applications

La complexité du traitement peut être exprimée par le nombre d'opérations CORDIC NO_{total} à effectuer sur le bloc de données. Pour garantir le traitement continu, le temps maximal T_C nécessaire pour réaliser ce traitement doit rester inférieur au temps d'acquisition T_{acq} du bloc de données. Si l'on dispose de $NC_{utilisé}$ opérateurs CORDIC en parallèle pour effectuer toutes les opérations, le temps de calcul global T_C sera égal à

$$T_C = (NO_{total}/NC_{utilisé}) \times (T_{CORDIC} + T_{reconfiguration}) \quad (3.22)$$

où $T_{reconfiguration}$ correspond au temps de reconfiguration entre chaque itération de calcul. Il faut noter que le temps de reconfiguration correspond au temps de changement

TAB. 3.5 – Notations utilisées dans les équations générales

T_{CORDIC}	Temps maximal pour réaliser le traitement par CORDIC
$NC_{utilisé}$	Nombre d'opérateurs CORDIC utilisé
NO_{total}	Nombre total d'opérations CORDIC
T_{acq}	Temps d'acquisition du bloc de données à traité

des interconnexions entre les opérateurs CORDIC. Il peut être dû au temps de sélection par des multiplexeurs ou au temps de reconfiguration dynamique de la partie reconfigurable du FPGA.

Pour un algorithme donné caractérisé par un NO_{total} d'opérations CORDIC à effectuer sur un bloc de données de temps d'acquisition T_{acq} , l'inégalité suivante permet d'atteindre le nombre de CORDIC optimal garantissant un traitement temps réel du flot de données.

$$T_C \leq T_{acq}. \quad (3.23)$$

En appliquant l'équation 3.22, l'équation 3.23 s'exprime de la façon suivante :

$$NC_{utilisé} \geq NO_{total} \times (T_{CORDIC} + T_{Reconfiguration}) / (T_{acq}) \quad (3.24)$$

Et dans une implémentation de l'opérateur CORDIC utilisant une structure pipeline à N_p étages, le temps T_{CORDIC} pourra être divisé par ce nombre d'étages, d'où une nouvelle inéquation :

$$NC_{utilisé} \geq NO_{total} \times (T_{CORDIC}/N_p + T_{Reconfiguration}) / (T_{acq}) \quad (3.25)$$

3.3.4 Architecture CBDRA

L'architecture CBDRA peut être schématisée par la figure 3.7. L'architecture se décompose en trois niveaux : le gestionnaire de l'architecture, le contrôleur de configuration et le bloc de traitement. Le gestionnaire de l'architecture est chargé de résoudre les équations établies dans le paragraphe précédent en fonction des caractéristiques du système, donc de décider du nombre d'opérateurs CORDIC à utiliser. Ensuite les paramètres obtenus sont envoyés vers le contrôleur de configuration qui est l'entité responsable du chargement des fichiers de configuration (bitstreams) dans le plan mémoire du FPGA via une interface de configuration. Le bloc de traitement est bâti autour d'un contrôleur de traitement, de ressources de mémorisation, d'un ensemble d'opérateurs CORDIC et des interconnexions. Le contrôleur des tâches est chargé de la gestion de l'architecture. Il a pour but d'assurer le bon ordonnancement des traitements. Pour cela, il est chargé de décider quelles sont les opérations qui doivent être exécutées à un instant donné. Cette décision se base sur des séquencements des applications. Enfin l'entité d'entrée/sortie gère les signaux d'interface avec l'extérieur. Cette architecture permet d'offrir une meilleure efficacité énergétique tout en conservant les performances requises par ces applications et un haut niveau de flexibilité.

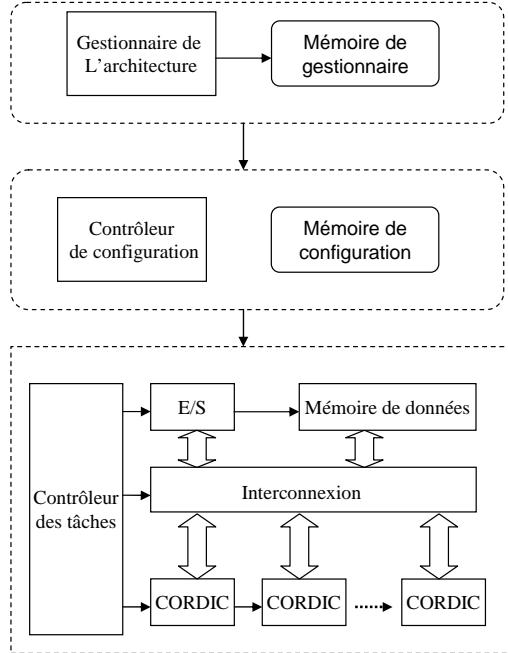


FIG. 3.7 – Architecture de CBDRA

Un modèle de gestion de configuration "HDCM" (Hierachical Distributed Configuration Management) [101] a été étudié dans notre équipe. Il s'appuie sur l'analyse algorithmique des applications multi-standards. Il permet de répondre aux besoins de reconfigurabilité des applications en vue des différents types de changements de contexte. Il possède trois niveaux de hiérarchie comme présenté sur la figure 3.8. Les gestions de l'architecture CBDRA peuvent être retrouvées dans ce modèle. Le gestionnaire de l'architecture se trouve au niveau *L2_CMU* du modèle. Les paramètres envoyés par l'application sont interprétés par *L2_CMU*. Il s'agit ici de résoudre les équations pour le nombre d'opérateurs CORDIC à implémenter et l'organisation des interconnexions entre les opérateurs CORDIC. Le contrôleur de configuration se trouve au niveau *L3_CMU*. Son premier rôle est d'instancier les blocs de traitement paramétrés. Dans notre cas, il s'agit des opérateurs CORDIC à implémenter. Il est ensuite chargé de configurer les ressources matérielles en vue de l'exécution de chaque élément de traitement.

Afin de mettre en oeuvre différentes structures en utilisant la reconfiguration, deux approches sont utilisées dans cette thèse : architecture reconfigurable avec la reconfiguration statique et architecture reconfigurable avec la reconfiguration dynamique. Nous allons présenter ces architectures dans les paragraphes suivants.

3.4 Architecture reconfigurable CBDRA

Les FPGAs sont des architectures reconfigurables de type grain fin. Leur structure permet de réaliser des systèmes reconfigurables. Notre architecture CBDRA se base principale-

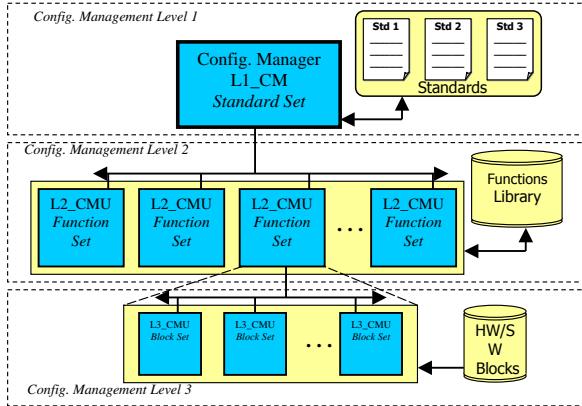


FIG. 3.8 – Gestion de configuration hiérarchique

ment sur FPGA. La première partie de ce chapitre sera donc consacrée à la description des caractéristiques du FPGA et à la démarche de conception. La seconde portera sur l'architecture générale de l'architecture reconfigurable CBDRA avec reconfiguration statique et avec reconfiguration dynamique.

3.4.1 Structure de FPGA

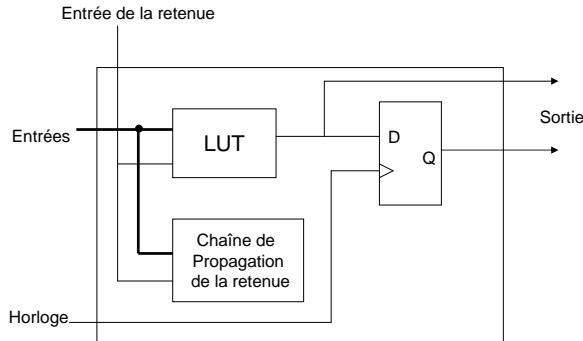


FIG. 3.9 – Elément configurable de base des FPGA

De manière générale, un circuit programmable de type FPGA se compose de blocs logiques reliés entre eux via un réseau d'interconnexion. Les deux leaders du marché des FPGA, Xilinx et Altera, proposent des éléments configurables de base relativement similaires qui se composent d'une LUT à 4 entrées, d'une chaîne de propagation rapide de la retenue et d'un registre de sortie afin d'assurer la synchronisation des signaux comme illustré sur la figure 3.9.

La structure FPGA est composée d'une matrice de blocs logiques CLB programmables, de blocs logiques d'entrée sortie IOB programmable, de buffer trois états, de bloc RAM et

du réseau d'interconnexion entre ces éléments. La configuration d'un CLB est présentée sur la figure 3.10.

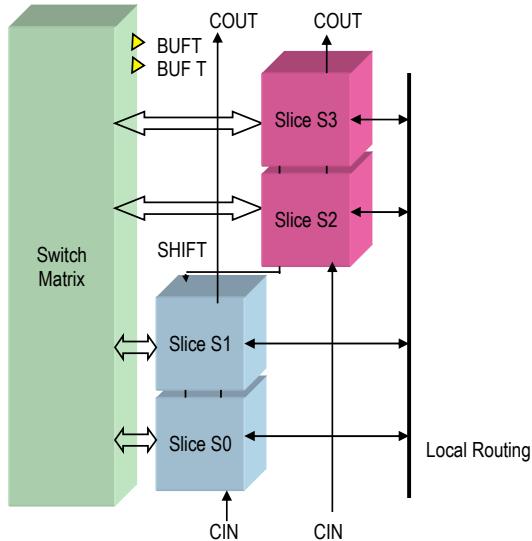


FIG. 3.10 – Configuration d'un CLB

Un FPGA se caractérise par son réseau de routage. Les éléments configurables contiennent des E/S (Entrées/Sorties), des modules logiques (LUT, multiplexeur...), des mémoires (RAM simple ou double port) et des modules arithmétiques plus complexes (Multiplieur, additionneur/soustracteur...). La connexion entre les éléments configurables est assurée par des blocs programmables tandis que les matrices relient ces blocs entre eux.

Les Slices sont composés de deux LUT (look up table) de quatre variables, de deux éléments de stockage de type FLIP-FLOP ou bascule D, de deux multiplexeurs et de logique dédiée. La structure d'un Slice est illustrée sur la figure 3.11.

Un FPGA est capable de réaliser toutes applications grâce à la flexibilité des ressources de routage. Un ensemble de lignes de routage de différentes longueurs est disponible. Afin de répondre aux besoins croissants en terme de mémoire, les FPGA se dotent d'éléments de mémorisation configurables. De plus, l'adjonction de fonction pré-câblées et unités arithmétiques de type MAC rend le FPGA attractif pour les applications de traitement du signal.

Les cibles reconfigurables de type FPGA présentent un attrait certain par leur caractère grain fin. Il est devenu une solution de plus en plus retenue grâce à sa forte puissance de calcul et son court cycle de mise sur le marché des applications. Contrairement à ses concurrents, Xilinx propose une technologie FPGA permettant la reconfiguration partielle du composant.

3.4.2 Reconfiguration dynamique de FPGA

La reconfiguration dynamique est définie par le fait de reconfigurer une partie d'un système pendant que le reste de l'application continue à tourner, contrairement à la notion

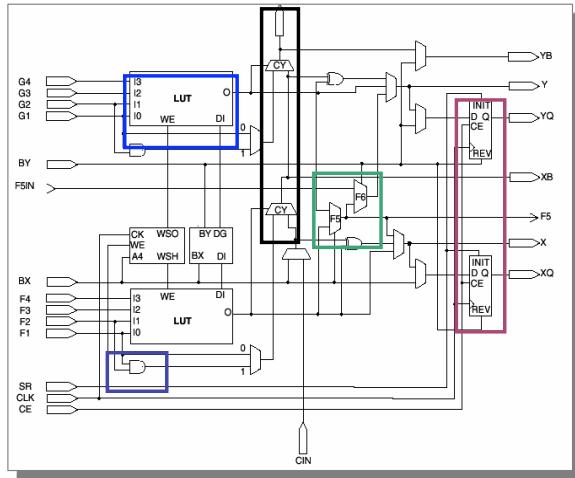
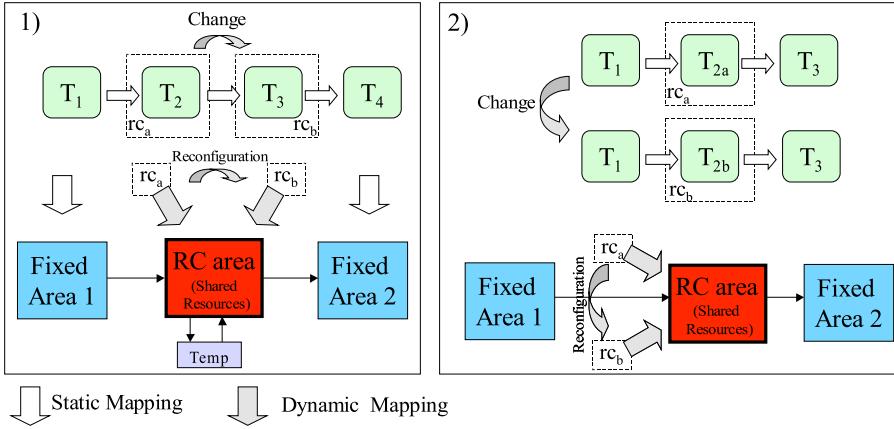


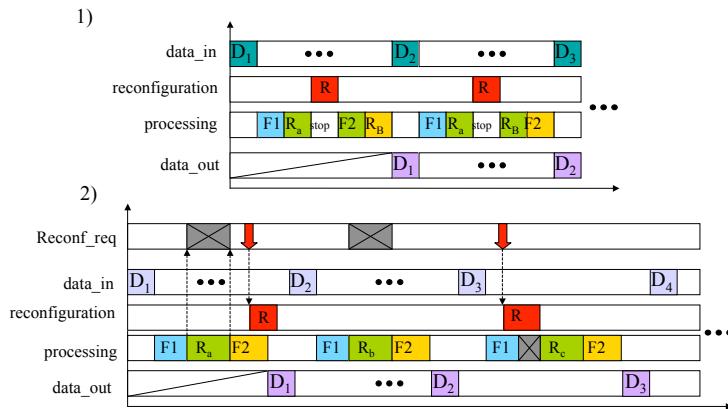
FIG. 3.11 – Illustration d'un Slice

de reconfiguration statique qui implique l'arrêt complet de l'application. La notion de reconfiguration dynamique est liée à la caractéristique de reconfiguration partielle du FPGA dans le cas où l'application s'effectue sur un seul FPGA. En effet, le principe de la reconfiguration partielle consiste à modifier seulement une partie du circuit FPGA ce qui permet de réduire le temps de configuration tout en donnant une souplesse accrue au système. Elle peut aussi être utilisée pour effectuer des opérations de changement de paramètres dans le but de réduire ou supprimer des parties de contrôle d'une application dédiées au changement de paramètre. La reconfiguration partielle peut intervenir pendant le fonctionnement du reste du circuit. La reconfiguration dynamique peut être une tâche exécutée concurremment aux tâches de traitement de l'application [72]. La reconfiguration dynamique est considérée comme un moyen de changer les tâches de traitement de l'application. Nous avons distingué deux cas de mise en oeuvre de la reconfiguration dynamique présentée par la figure 3.12.

La figure 3.12.1 montre le premier cas de reconfiguration dynamique avec une dépendance de données entre les deux fonctions R1 et R2 qui utilisent la même ressource reconfigurable. Dans ce cas F2 doit attendre la fin de la reconfiguration R1, comme montré dans la figure 3.13.1. Il est donc nécessaire d'effectuer une sauvegarde du contexte d'exécution entre chaque reconfiguration, afin de sauvegarder les données en cours de traitement. Dans le deuxième cas (figure 3.12.2, 3.13.2), la reconfiguration peut être exécutée à tout moment en dehors de la phase d'exécution du bloc à reconfigurer. Mais il faut aussi assurer que le temps de la reconfiguration ajouté au temps de traitement ne dépasse pas la période d'échantillonnage des données d'entrées dans des applications de type flot de donnée synchrone. Dans notre application, nous utilisons le premier type de reconfiguration et la reconfiguration est exécutée comme une partie du processus du décodeur MIMO. La reconfiguration est donc utilisée pour changer les interconnexions entre les opérateurs CORDIC. Nous voyons ici que le temps de reconfiguration est une caractéristique qui joue un rôle important sur les performances des traitements de l'application, surtout pour le



deuxième type de reconfiguration où le temps de reconfiguration entre les tâches T_2 et T_3 est ajouté au temps d'exécution de l'application.



3.4.3 Flots de conception du FPGA

Les méthodologies utilisées pour concevoir des modules reconfigurables sont décrites dans ce paragraphe. Il existe de nombreux outils et flots de conception permettant de mettre en oeuvre la reconfiguration partielle sur FPGA. Le flux de conception utilisé au cours de cette thèse est tout d'abord présenté. Le FPGA Virtex-II de Xilinx est le circuit ciblé. Xilinx a proposé deux flots de conception dans [102] pour mettre en oeuvre la reconfiguration partielle. Le premier flux est nommé "Module-Based Partial Reconfiguration". Il est basé sur une méthodologie de conception modulaire proposée par Xilinx, appelée "Modular Design" [103].

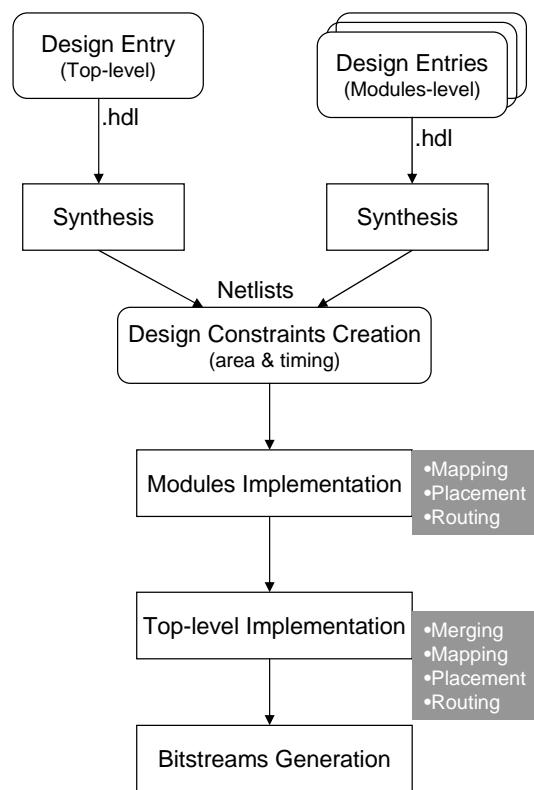


FIG. 3.14 – Flot de conception modulaire sur FPGA

Le deuxième flot est nommé "Difference-Based Partial Reconfiguration". Il permet de générer des bitstreams partiels par différence entre deux conceptions complètement placées/routées. Contrairement au flot "*Module-based Partial Reconfiguration*", aucune zone du FPGA n'est désignée comme reconfigurable ce qui permet de suivre le flot de conception classique jusqu'au placement/routage final.

Nous présentons ici notre flot de conception utilisé au cours de cette thèse. Il est basé sur le flot de conception proposé plus récemment par Xilinx. Nous avons développé nos architectures avec un flot de conception différent pour deux générations de FPGA de Xilinx. Le premier est le flot "*Module-Based Partial Reconfiguration*" utilisé sur le FPGA Virtex-II de Xilinx. Le deuxième est le flot "*Partial Reconfiguration*" utilisé sur le FPGA Virtex-4 de Xilinx.

Le flot "*Module-Based Partial Reconfiguration*" permet de placer et router chaque module indépendamment, dans des zones prédéfinies à l'avance par des contraintes de surface. Les modules reconfigurables sont identifiés au moment de la création de ces contraintes et sont ensuite placés et routés. Les bitstreams des modules sont générés, comme illustrés sur la figure 3.14.

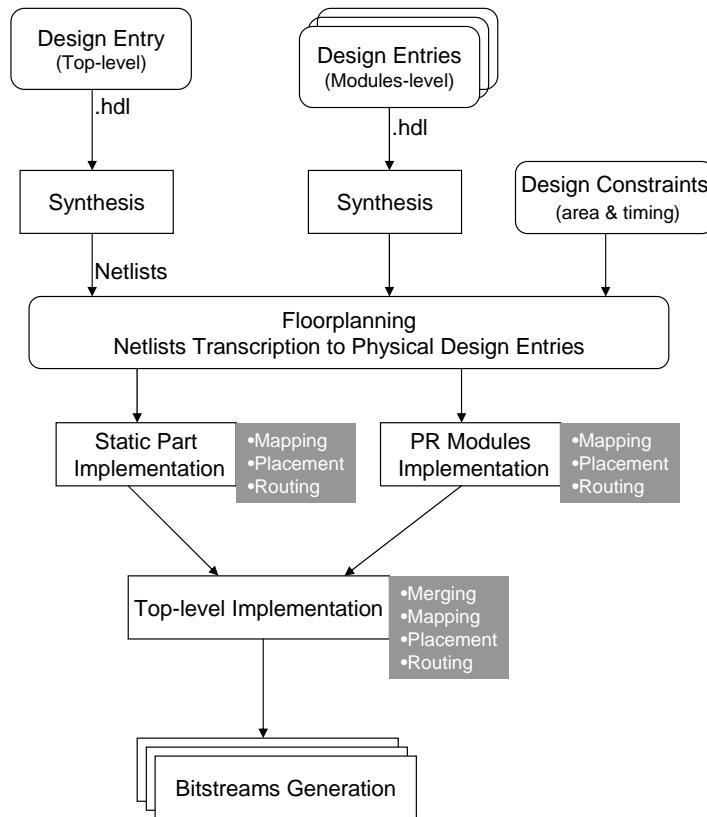


FIG. 3.15 – Flot de conception reconfigurable sur FPGA

Le flot "*Partial Reconfiguration*" permet de réaliser des bitstreams de configuration de modules en augmentant la flexibilité de conception par rapport au flot "*Module-based*

"Partial Reconfiguration", comme illustré sur la figure 3.15. La génération de bitstreams partiels s'effectue par la méthode par différence. Ces bitstreams sont créés par différence entre une conception totale et la conception contenant uniquement la partie statique. Ce flot est réalisé à l'aide de l'outil de "floorplanning" hiérachique PlanAhead. Ceci permet de dissocier complètement la description de haut niveau du système réalisé en VHDL et de réaliser le placement routage sur le FPGA. Les descriptions du niveau le plus élevé et des modules sont synthétisées avec l'outil de synthèse XST (Xilinx Synthesis Tool) avant d'être transcrives par PlanAhead. Pendant la transcription, les parties reconfigurables des modules sont donc extraites afin d'être placées dans des zones reconfigurables du FPGA. Il est à noter que PlanAhead garde les niveaux de hiérarchie des descriptions VHDL et permet d'automatiser le flot de conception à partir des fichiers de synthèse.

3.4.4 Connexion entre la partie statique et la partie reconfigurable

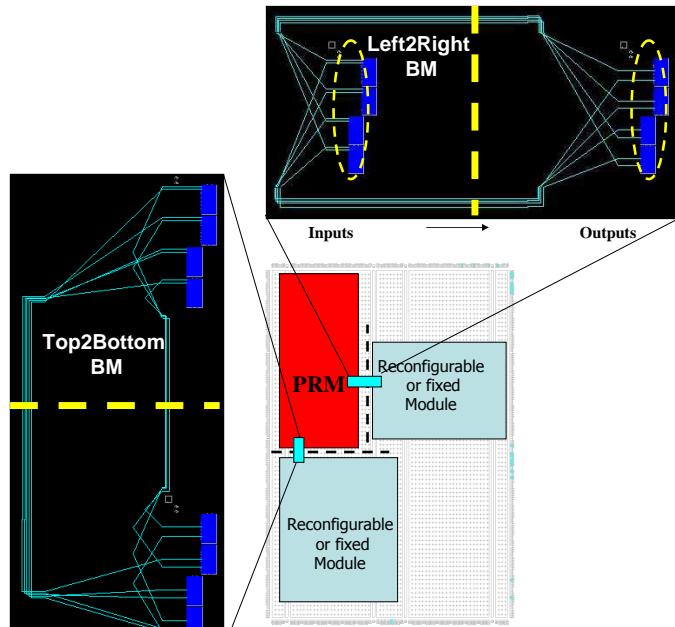


FIG. 3.16 – LUT based *Bus Macro* sur FPGA Virtex-II

Dans le but de gérer les communications avec des modules placés dynamiquement sur FPGA, les interfaces appelées *Bus Macro* sont utilisées pour passer les signaux d'interconnexion. Ces *Bus Macro* sont des composants pré-routés et placés entre les parties reconfigurables et la partie statique. Dans ce cas, le placement/routage des signaux d'entrées sorties d'un module reconfigurable doit être identique pour toutes les configurations. La figure 3.16 illustre en un exemple une vue après placement/routage d'un *Bus Macro* dans un FPGA Virtex-II de Xilinx. Ces *Bus Macro* sont basés sur des LUT. Les versions LUT-based *Bus Macro* améliorent la flexibilité de la reconfiguration, par la densité d'intégration plus forte offerte par les LUT comparées au BUFT et par la disponibilité de déclinaisons horizontales et verticales de ces interfaces.

3.4.5 Gestion de reconfiguration

La gestion de reconfiguration est primordiale dans l'efficacité de la prise en charge des traitements. Deux niveaux peuvent être distingués pour l'architecture reconfigurable : l'étendue de la reconfiguration et la rapidité de configuration.

Si la modification de l'architecture intervient en dehors de l'exécution de l'application alors il s'agit d'une reconfiguration statique. L'architecture matérielle doit être suffisamment souple pour que diverses applications puissent être implantées. En revanche, si l'architecture doit être modifiée en cours de traitement afin de réaliser des tâches particulières dans le temps, la reconfiguration est considérée comme dynamique. L'objectif est de déplacer spatialement certaines parties de l'architecture, de réaliser un traitement plus rapidement en utilisant tous les ressources disponibles. Dans tous les cas, le procédé vise une forte flexibilité et une optimisation en surface sur la cible. Il faut alors stocker en mémoire les configurations à effectuer sur le composant et prévoir des mécanismes de gestion. De manière générale, le processus de reconfiguration se déroule en deux étapes. La première étape est le chargement de la configuration initiale qui s'effectue via un composant externe de type EEPROM ou un microcontrôleur. La deuxième étape est la reconfiguration dynamique partielle qui s'effectue en interne à partir d'un processeur embarqué.

Nous pouvons considérer que la reconfiguration dynamique permet un multiplexage temporel des contextes. On peut distinguer deux types de reconfiguration dynamique : la reconfiguration dynamique globale et la reconfiguration dynamique partielle. La reconfiguration dynamique globale est définie par l'exploitation de la reconfigurabilité du système au sein d'une même application. Elle nécessite un partitionnement de l'application en plusieurs étapes temporellement indépendantes et ensuite les réalise par les configurations successives. Une autre approche consiste à ne modifier qu'une partie du circuit à la fois ce qui permet de réduire le temps de configuration et donner une souplesse accrue au système. La reconfiguration partielle peut intervenir pendant le fonctionnement du reste du circuit.

3.4.6 Architecture CBDRA et reconfiguration statique

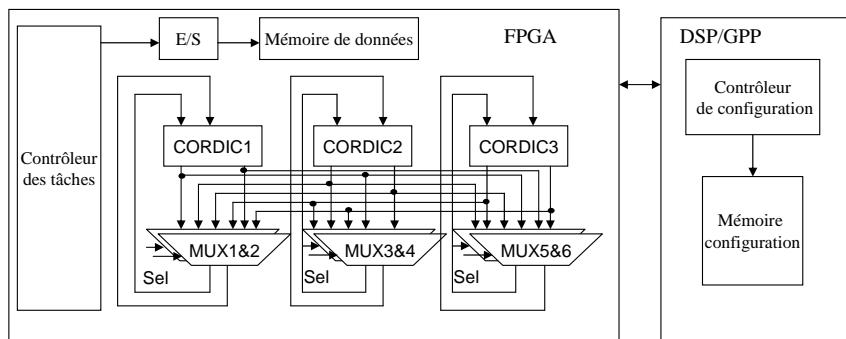


FIG. 3.17 – Architecture CBDRA avec la reconfiguration statique

Dans l'objectif de réduire les ressources matérielles, nous proposons l'architecture CBDRA en utilisant la reconfiguration statique. Elle peut s'adapter au changement des caractéristiques de la chaîne communication. Quand on change la structure du récepteur MIMO ou le débit, la reconfiguration statique est utilisée pour modifier le nombre d'opérateurs CORDIC. L'architecture CBDRA statique est composée des éléments de base CORDIC et des multiplexeurs qui permettent de changer l'état des connexions, comme illustré sur la figure 3.17 en prenant un exemple de 3 opérateurs CORDIC en parallèle. Chaque multiplexeur contrôle le chemin des données entre les entrées et les sorties des opérateurs CORDIC. Les données intermédiaires sont stockées dans des registres. L'architecture CBDRA est implémentée sur la totalité du FPGA. La reconfiguration se fait par la configuration totale du FPGA. L'équation 3.25 donne le nombre d'opérateurs CORDIC nécessaire pour s'adapter aux caractéristiques données.

Il faut souligner que l'architecture reconfigurable CBDRA avec la reconfiguration statique est une solution attractive pour les futurs systèmes de communication numérique, car le nombre d'opérateur CORDIC peut être variable pour s'adapter aux besoins des différentes standards. Nous décrivons cette architecture en détails dans les articles [?] et [104] et expliquons dans le paragraphe suivant en prenant comme exemple l'application d'un décodeur MIMO.

L'opérateur CORDIC exploite une structure pipeline de 15 étages. L'implémentation statique des interconnexions se fait par l'utilisation d'un grand nombre de multiplexeurs occupant une surface importante dans un FPGA et consommant beaucoup. En revanche, les multiplexeurs conservent leur état pendant les 15 étapes des opérations CORDIC. Cela nous amène à proposer une implémentation reconfigurable des interconnexions entre les opérateurs CORDIC en utilisant la reconfiguration partielle du FPGA afin de minimiser les ressources matérielles.

3.4.7 Architecture CBDRA et reconfiguration dynamique des interconnexions

L'architecture CBDRA utilisant la reconfiguration dynamique est illustrée par la figure 3.18. Elle peut se décomposer en deux parties. Nous avons montré les résultats des implémentations CBDRA pour les systèmes MIMO utilisant la reconfiguration dynamique dans [105] et [106].

La première partie est la partie fixe où les opérateurs CORDIC, les multiplieurs et les additionneurs sont implantés. La deuxième partie est définie dans des régions reconfigurables d'un FPGA permettant d'accueillir des modules reconfigurables. Dans notre cas, les modules reconfigurables réalisent les interconnexions des opérateurs CORDIC. Les deux parties sont reliées par un Bus Macro (BM) pour assurer que les signaux entre partie fixe et partie reconfigurable seront bien interconnectés après l'opération de reconfiguration partielle.

Les données de reconfiguration sont stockées dans la mémoire du Host. Puis les bitstreams sont préchargés dans la mémoire cache de configuration dont la vitesse d'accès permet de réduire le temps de reconfiguration. Les chargements des bitstreams sont gérés par le Host. Une première étape d'initialisation pendant laquelle le Host envoie le bitstream total au FPGA permet d'implémenter dans le composant le système global (partie fixe et contexte initial de la partie reconfigurable) ainsi que le contrôleur de reconfiguration constitué notamment d'un module à base de MicroBlaze et de la primitive ICAP (Internal

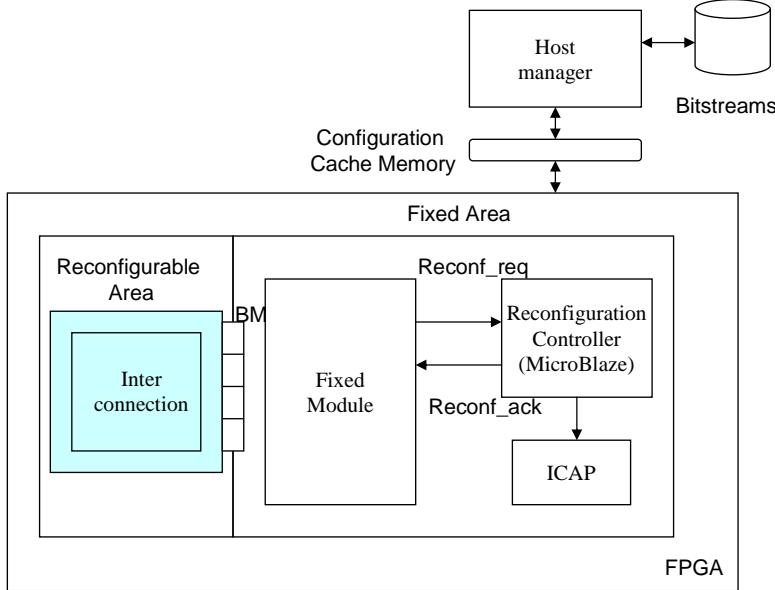


FIG. 3.18 – Architecture CBDRA avec la reconfiguration dynamique

Configuration Port Access). Ensuite les contextes des interconnexions sont changés par les reconfigurations. Lors d'une reconfiguration, seul le bitstream partiel correspondant au contexte du module reconfigurable est rechargé. La taille du bitstream partiel est fortement réduite par rapport à celle du bitstream total permettant de diminuer le temps de reconfiguration.

Ainsi, dans cette approche, les multiplexeurs sont remplacés par des interconnexions câblées reconfigurables, constituées de simples fils de liaisons modifiés à chaque reconfiguration, ce qui permet de diminuer l'encombrement et la consommation. Les signaux de contrôle de configuration de type request/ackowlege (*Reconf_req* et *Reconf_ack* dans la figure 3.18) permettent de contrôler la configuration du composant et de maintenir l'état des interfaces de transfert de données au cours d'une reconfiguration.

3.5 Application de l'architecture CBDRA pour un décodeur MIMO V-BLAST Square Root

Rappelons que nous avons donné la description fonctionnelle du décodeur MIMO V-BLAST Square Root dans le chapitre 1 et qu'elle est composée de 6 modules de traitement.

Les trois modules (M_1, M_2, M_3) présentent une architecture similaire. Ces modules sont conçus en utilisant des opérateurs CORDIC (voir l'exemple ci-dessous pour les calculs de $P^{1/2}$ et de Q_α dans le module M_1). Au lieu d'exécuter la décomposition QR par un réseau triangulaire, nous utilisons une séquence de rotation de Givens basée sur CORDIC.

Les trois derniers modules (M_4, M_5, M_6) sont basés sur les processeurs élémentaires (PE) constitués d'un multiplicateur-accumulateur, un soustracteur et un buffer. L'utilisation de PE est optimale pour réaliser les calculs dans ces trois modules, mais nous étendons l'utilisation de l'opérateur CORDIC dans ces trois derniers module. Ce qui nous permet d'avoir encore plus de flexibilité vis à vis de l'utilisation de l'opérateur CORDIC dans l'ensemble des modules. Nous expliquerons cette approche et comparerons ces deux approches en terme de complexité dans l'annexe E.

Une implémentation de l'algorithme "Square Root" a été réalisée par Z.Guo dans [34], mais cette réalisation ne remplit pas les contraintes de notre problème évoquées dans le paragraphe 3.3. C'est pour y répondre que nous proposons dans cette thèse une implémentation d'un décodeur MIMO V-BLAST Square Root en appliquant l'architecture CBDRA.

3.5.1 Stationnarité du canal

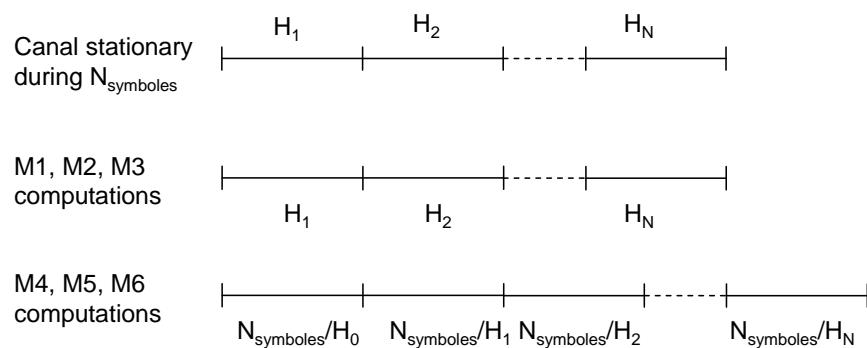


FIG. 3.19 – Chronogramme du flux de données

La plupart des standards de communications mobiles actuels incluent une séquence d'apprentissage pour estimer le canal. Le canal est considéré comme stationnaire pendant cette période d'apprentissage. La stationnarité du canal peut être exprimée par le nombre de symboles utilisé pour l'apprentissage. Les décompositions de matrice du canal peuvent donc s'effectuer pendant ce temps-là. Ces décompositions sont principalement exécutées par les opérateurs CORDIC. La figure 3.19 illustre l'enchaînement des opérations des CORDIC et des PE. Pendant que les opérateurs CORDIC traitent les données du premier canal, les PE utilisent les données du canal précédent.

3.5.2 Implémentation parallèle de CORDIC (Un exemple : calcul de $P^{1/2}$ et de Q_α)

Dans le chapitre 1, nous avons indiqué que le calcul de $P^{1/2}$ et de Q_α est effectué dans le premier module. C'est la première étape pour décomposer la matrice du canal H . Pour des

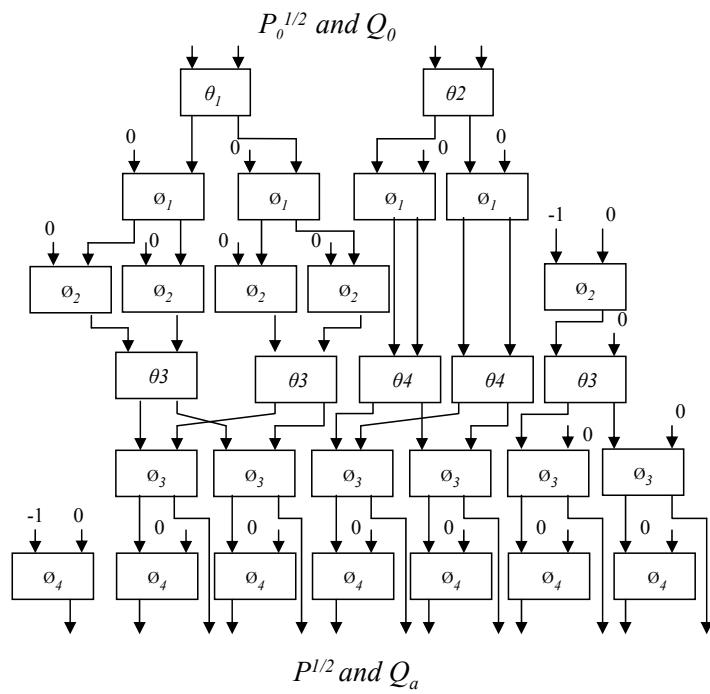


FIG. 3.20 – Structure parallèle des CORDIC pour calculer $P^{1/2}$ et Q_α

raisons de stabilité numérique et de facilité d'implémentation nous utilisons une séquence de rotation de Givens basée sur des opérateurs CORDIC.

Considérons maintenant, dans le reste de cette thèse, un modèle MIMO 2×2 antennes. Alors le premier terme de la seconde colonne de l'équation 2.24 est représentée ci-dessous :

$$H_i^{1 \times M} P_{i-1}^{1/2M \times M} = \begin{bmatrix} HP_{11} & HP_{12} \\ HP_{21} & HP_{22} \end{bmatrix} \quad (3.26)$$

$$\begin{aligned} \theta_1 &= \tan^{-1} \frac{\Im(HP_{11})}{\Re(HP_{11})} \\ \theta_2 &= \tan^{-1} \frac{\Im(HP_{12})}{\Re(HP_{12})} \\ \phi_1 &= \tan^{-1} \frac{|HP_{11}|}{|HP_{12}|} \\ \phi_2 &= \tan^{-1} \frac{1}{|(HP_{11})^2 + |HP_{12}|^2} \\ \theta_3 &= \tan^{-1} \frac{\Im(HP_{21})}{\Re(HP_{21})} \\ \theta_4 &= \tan^{-1} \frac{\Im(HP_{22})}{\Re(HP_{22})} \\ \phi_3 &= \tan^{-1} \frac{|HP_{21}|}{|HP_{22}|} \\ \phi_4 &= \tan^{-1} \frac{1}{|(HP_{21})^2 + |HP_{22}|^2} \end{aligned}$$

Une fois les angles pré-calculés, nous pouvons exécuter les transformations unitaires à l'aide des opérateurs CORDIC. Le calcul de l'équation 2.24 peut alors être décomposé en 29 rotations.

La figure 3.20 illustre une implémentation totalement parallèle des calculs utilisant 29 opérateurs CORDIC. La structure parallèle est conçue pour obtenir un très haut débit, mais elle est coûteuse en ressources matérielles. Pour les applications à plus faible débit, toute la puissance de calcul disponible n'est pas utilisée. Elle est aussi conçue pour s'adapter à un changement du canal très rapide. Mais dans la pratique, souvent le canal reste stationnaire pendant un certain nombre de cycles d'horloge du matériel. En plus, la stationnarité du canal varie d'un système à un autre.

Pour tendre vers une implémentation optimale du décodeur MIMO, nous avons proposé une architecture reconfigurable dont la structure est fonction des caractéristiques de la transmission. Nous utilisons dans un premier temps l'architecture CBDRA avec la reconfiguration statique pour implémenter un décodeur MIMO. Cette architecture est construite autour d'un certain nombre d'opérateurs CORDIC avec leurs interconnexions et des mémoires pour stocker les données intermédiaires. Dans cette structure, un contrôleur est nécessaire pour réaliser le séquencement de l'ensemble de la structure qui est déterminé par le processus de l'algorithme MIMO. Nous étudions d'abord la relation entre le nombre d'opérateurs CORDIC utilisé et le débit symbole en appliquant les équations générales obtenues dans la paragraphe 3.3.3. Cela permet de déterminer le nombre optimal d'opérateurs CORDIC à planter.

3.5.3 Relation théorique entre le débit et le nombre d'opérateurs CORDIC utilisé

Rappelons que dans le paragraphe 3.3.3, nous avons établi une relation générale entre le nombre d'opérateurs CORDIC utilisé en parallèle et le temps d'acquisition du système comme suit :

$$NC_{utilisé} \geq NO_{total} \times (T_{CORDIC}/N_p + T_{Reconfiguration})/(T_{acq}) \quad (3.27)$$

Dans l'application du système MIMO, le temps d'acquisition T_{acq} est fonction du débit symbole et de la durée de la stationnarité du canal. La longueur du temps de stationnarité correspond au nombre de symboles transmis pendant lequel le canal est stationnaire. Le temps d'acquisition T_{acq} peut donc être écrit la forme suivante :

$$T_{acq} = N_{symboles/H}/F_s. \quad (3.28)$$

Avec $F_s = \text{Débit}_{symbole}/(\text{M} \times \text{b})$

A la réception, le temps de calcul des CORDIC est exprimé par la fréquence horloge déterminée par la cadence de l'opérateur CORDIC comme suit :

$$T_{CORDIC}/N_p = 1/Freq. \quad (3.29)$$

En appliquant les équations 3.28 et 3.29 dans l'équation 3.30, nous obtenons une nouvelle inéquation dans le cas du système MIMO :

$$NC_{utilisé} \geq (NO_{total} \times F_s) \times (1/freq + T_{Reconfiguration})/(N_{symboles/H}) \quad (3.30)$$

Dans cette équation, NO_{total} représente le nombre total d'opérations CORDIC à utiliser pour un système MIMO donné. Il est obtenu en fonction du nombre d'antennes à l'émission et à la réception. Cette inéquation permet d'obtenir le nombre optimal d'opérateurs CORDIC à implanter pour s'adapter au débit de la chaîne de communication. Le débit correspond au débit symbole binaire à l'émission ($\text{Débit}_{symbole}$).

Ce même débit peut être obtenu avec un nombre de CORDIC inférieur, jusqu'à obtenir le nombre optimal qui est décrit dans l'équation suivante :

$$NC_{optimal} = (NO_{total} \times F_s) \times (1/freq + T_{Reconfiguration})/(N_{symboles/H}). \quad (3.31)$$

Il faut noter que dans une implémentation statique, le temps de reconfiguration qui correspond au temps de sélection par le multiplexeur peut être ignoré par rapport au temps de calcul CORDIC.

La relation entre le nombre d'opérateurs CORDIC et le débit est illustrée dans la figure 3.21. Les autres facteurs dans l'équation 3.31 sont considérés comme des constantes dans un même système MIMO. Le nombre d'opérateurs CORDIC minimal pour atteindre le débit souhaité est $NC_{optimal}$. Si on diminue le nombre d'opérateurs CORDIC, le débit va chuter linéairement. En revanche, le débit n'augmente pas linéairement avec le nombre d'opérateurs dans la figure 3.21. Il reste constant comme indiqué parce qu'il calcule la même matrice du canal. Nous avons décrit en détail cette équation dans [107].

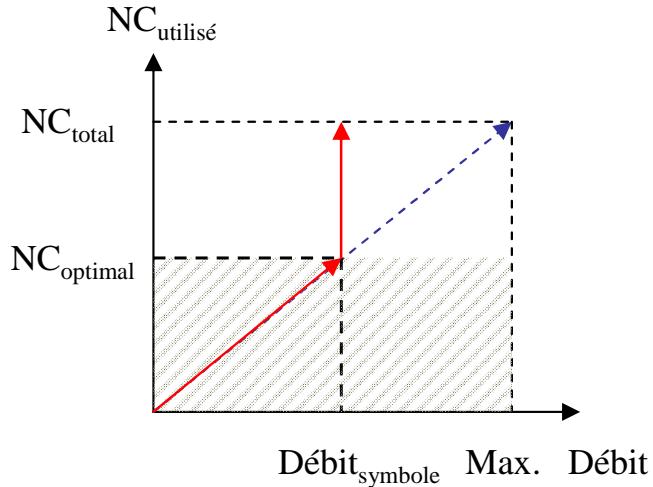


FIG. 3.21 – Relation entre le débit symbole et le nombre d'opérateurs CORDIC utilisés

TAB. 3.6 – Notations utilisées dans les équations

Freq	Fréquence horloge déterminée par la cadence de calcul CORDIC
b	Bits par symbole
$NC_{utilisé}$	Nombre d'opérateurs CORDIC utilisé
$NC_{optimal}$	Nombre optimal d'opérateurs CORDIC
NO_{total}	Nombre total d'opérations CORDIC
F_s	Fréquence symbole
$NC_{symbols/H}$	Nombre de symbole par trame
$Débit_{symbole}$	Débit binaire à l'émission

Nous pouvons remarquer dans l'équation 3.31 que les autres facteurs peuvent être aussi variables. En effet, le nombre d'antennes à l'émission et à la réception, le modulation du signal sont des paramètres qui peuvent changer d'un standard à un autre. La cadence de calcul de l'opérateur CORDIC peut varier selon les différentes cibles matérielles. Cela montre que cette équation est applicable dans différentes conditions liées aux caractéristiques de la chaîne de communication.

3.5.4 Exemple de résolution des équations et d'implémentation : calcul de $P^{1/2}$ et de Q_α

Avec l'équation 3.31, nous pouvons calculer exactement le nombre d'opérateurs CORDIC nécessaire pour s'adapter aux débits donnés en diminuant la complexité. Lorsque le débit demandé change, nous pouvons modifier le nombre d'opérateurs CORDIC ainsi que leurs interconnexions. Par exemple, si nous définissons une trame comme un paquet de 7 symboles ($N_{\text{symboles}}/H = 7$), alors, pour atteindre 600 Mbits/s, le nombre optimal d'opérateurs CORDIC est 5, en supposant que la fréquence de fonctionnement d'un opérateur CORDIC est de 150 MHz sur un système MIMO de 2 antennes avec modulation QPSK. Les calculs de $P^{1/2}$ et de Q_α peuvent être alors exécutés par 5 opérateurs CORDIC en parallèle en 7 cycles (Fig.3.22 et 3.23).

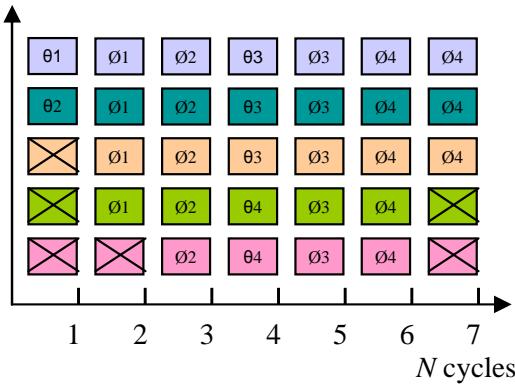


FIG. 3.22 – Organisation des différentes structures à base de 5 CORDIC

Les 5 CORDIC sont utilisés de façon itérative. L'organisation des calculs est illustrée sur la figure 3.22. Les 29 opérations de la structure parallèle sont séparées et exécutées dans les différents cycles. Les 5 CORDIC en parallèle ne sont pas toujours utilisés en raison de problème de dépendance de données entre les opérations CORDIC. Par exemple, l'opération CORDIC du deuxième cycle avec l'angle ϕ_1 ne peut pas commencer tant que les deux opérations du premier cycle ne sont pas finies. En revanche, il est nécessaire de mémoriser des données intermédiaires si l'une des données de sorties d'un CORDIC au cycle précédent n'est pas connectée à une entrée des CORDIC. Par exemple, c'est le cas entre le troisième cycle et le quatrième cycle. La sortie du cinquième CORDIC doit attendre deux cycles pour être utilisée. Il est donc nécessaire de stocker les données pour éviter la perte des données.

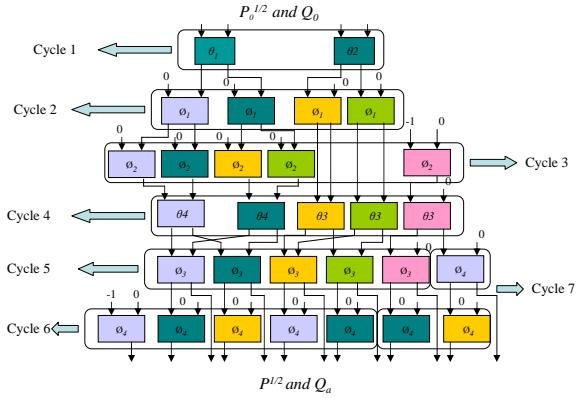


FIG. 3.23 – utilisation itérative de 5 CORDIC

Prenons maintenant le cas d'un débit plus faible, par exemple de 400 Mbits/s nous changeons le nombre d'opérateurs CORDIC. L'application de l'équation 3.30 nous permet de définir le nombre d'opérateurs à implanter, qui doit être 3. Les calculs de $P^{1/2}$ et de Q_α peuvent alors être exécutés en 10 cycles. L'organisation de l'architecture des opérateurs CORDIC est présentée dans les figures 3.24 et 3.25. Nous avons analysé l'utilisation de la reconfiguration statique dans [108].

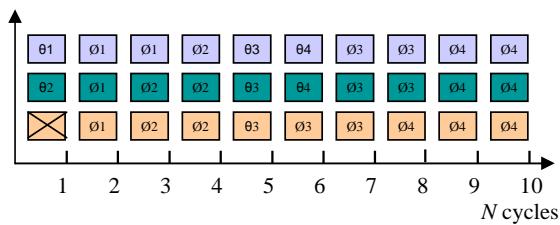


FIG. 3.24 – Organisation des différentes structures à base de 3 CORDIC

Nous analysons maintenant la contrainte d'utilisation de l'équation en raison de dépendance de données entre les opérateurs CORDIC à différents cycles. Le nombre d'opérateurs CORDIC doit être inférieur au nombre d'opérations CORDIC en parallèle dans le même cycle. Si on considère que le nombre maximum d'opérations CORDIC en parallèle dans un cycle est NO_{para} , le nombre maximum d'opérateurs CORDIC à utiliser doit satisfaire la condition suivante pour que l'équation soit applicable :

$$NC_{utilisé} \leq NO_{para}. \quad (3.32)$$

Dans cet exemple de calcul de $P^{1/2}$ et de Q_α , le nombre maximum d'opérations CORDIC qui peut être exécuté en parallèle est 7. Le nombre d'opérateurs doit être inférieur à 7.

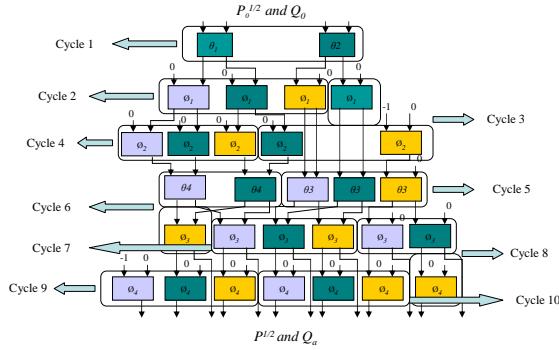


FIG. 3.25 – utilisation itérative de 3 CORDIC

Il faut noter que pour bénéficier de l'efficacité d'une structure pipeline CORDIC, le délai de traitement est de $N_p \times \text{Nsymboles/H}$.

Il est à noter que nous pouvons utiliser également une structure paramétrable pour réaliser les différents débits demandés. Au lieu d'utiliser la reconfiguration statique, les différentes configurations peuvent être réalisées par le changement des paramètres. Dans ce type d'approche, toutes les informations sur les applications visées doivent être connues au moment de la conception de la structure commune. L'avantage de cette approche est de définir un nombre limité de paramètres pour décrire plusieurs modes de fonctionnement avec un temps de changement de mode négligeable. Mais cette solution nécessite de garder le maximum de ressources matérielles pour assurer le débit le plus élevé et d'utiliser des structures de sélection telles que des multiplexeurs.

Dans cette approche, l'opérateur CORDIC exploite une structure pipeline de 15 étages. L'implémentation statique des interconnexions se fait par l'utilisation d'un grand nombre de multiplexeurs qui changent le contexte d'interconnexions l'un après l'autre. Les multiplexeurs occupent une surface importante dans un FPGA et consomment beaucoup. En revanche, les multiplexeurs conservent leur état pendant les 15 étapes des opérations CORDIC. Cela nous amène à étudier une implémentation reconfigurable des interconnexions entre les opérateurs CORDIC.

3.5.5 Utilisation de l'architecture CBDRA avec la reconfiguration dynamique

Nous utilisons l'architecture CBDRA avec la reconfiguration dynamique pour optimiser les ressources matérielles du décodeur MIMO. Les calculs sont séparés en deux parties : la partie statique qui se compose des éléments de base du décodeur CORDIC et la partie reconfigurable où les interconnexions entre les éléments de base sont implémentées. Les multiplexeurs utilisés dans la structure précédente sont remplacés par certain nombre de reconfigurations qui sont déterminées par les calculs du décodeur. Chaque reconfiguration représente donc un état du multiplexeur. Le calcul de $P^{1/2}$ et Q_α est pris comme exemple sur la figure 3.26 pour montrer l'utilisation de la reconfiguration dynamique dans les calculs de l'algorithme MIMO.

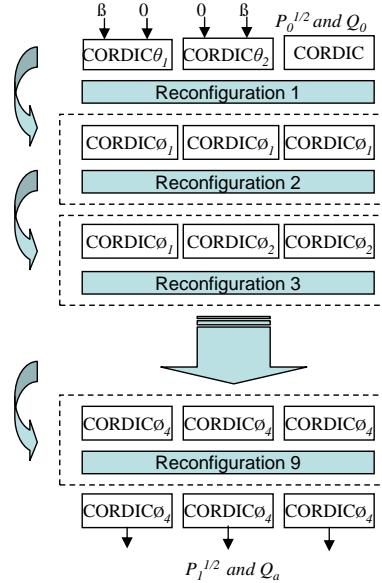


FIG. 3.26 – Utilisaiton de la reconfiguration dynamique pour calculer $P^{1/2}$ et Q_α

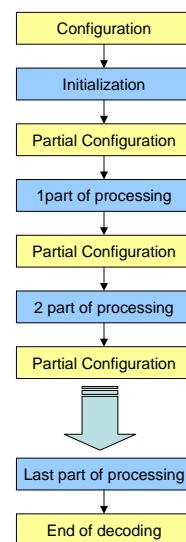


FIG. 3.27 – Séquencement de reconfigurations partielles

Pendant que la partie reconfigurable est modifiée, la partie fixe du FPGA continue à fonctionner. Les tâches des reconfigurations sont considérées comme une partie du traitement du décodeur. Les séquencements de la reconfiguration sont pré-définis par les traitements du décodeur, comme présenté sur la figure 3.27.

Chaque reconfiguration est insérée dans les traitements lorsqu'un changement d'interconnexion est demandé. La partie fixe envoie un signal $reconf_{req}$ au contrôleur de configuration MicroBlaze pour demander le changement de la configuration. Ce dernier est chargé d'envoyer les bitstreams présents dans la mémoire cache de configuration dans le plan mémoire du FPGA via une interface de configuration ICAP.

3.5.6 Implémentation de l'ensemble de l'algorithme

Les modules M1, M2, M3 ont été implantés comme vu précédemment et les modules M4, M5, M6 sont implantés de la manière suivante :

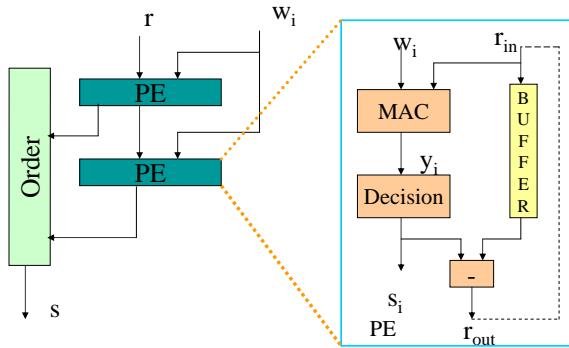


FIG. 3.28 – Architecture des modules M_4 , M_5 , M_6

Ces modules sont constitués de plusieurs étages de PE pour atteindre une haute vitesse de traitement des messages reçus (Fig.3.28). Chaque étage PE se compose d'un multiplicateur-accumulateur, un comparateur, un soustracteur et une mémoire. Le débit peut être encore amélioré en parallélisant plusieurs étages PE. Si le débit n'est pas un critère exigeant, on peut utiliser un seul PE de façon itérative afin de diminuer la complexité du module.

A l'aide de tous les modules de base, nous pouvons construire l'architecture globale. Celle-ci est illustrée par la figure 3.29. Nous utilisons un contrôleur pour organiser les interconnexions entre ces modules. Les données sont stockées dans une mémoire FIFO. La taille de la mémoire dépend du nombre d'opérateurs CORDIC et de PE utilisés.

Nous avons montré que pour un ensemble de paramètres fixés caractérisant la transmission, nous pouvons en déduire l'architecture optimale de réalisation garantissant les performances de l'algorithme pour une complexité matérielle minimale. En utilisant la technologie FPGA, nous pouvons reconfigurer le système de traitement de façon à implémenter cette architecture optimale mais pour une combinaison des paramètres donnée et en ayant préalablement réalisé les différentes implémentations pour différentes combinaisons.

Dans un système MIMO où l'on envisage une adaptation dynamique des paramètres de la transmission (débit variable, structure d'antennes évolutive), il faut concevoir une plateforme matérielle de réalisation du décodeur capable de modifier son architecture en

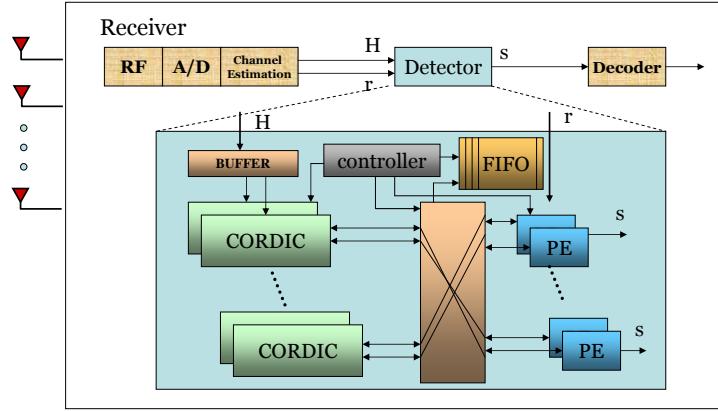


FIG. 3.29 – Architecture de l'ensemble du décodeur MIMO V-BLAST Square Root

fonction de ces changements de contexte. La technologie FPGA avec reconfiguration dynamique autorise cette approche. L'architecture reconfigurable dynamiquement que nous proposons se compose d'une part de fonctions de gestion de reconfiguration et d'autre part des fonctions de traitement. Cette architecture est illustrée par la figure 3.30.

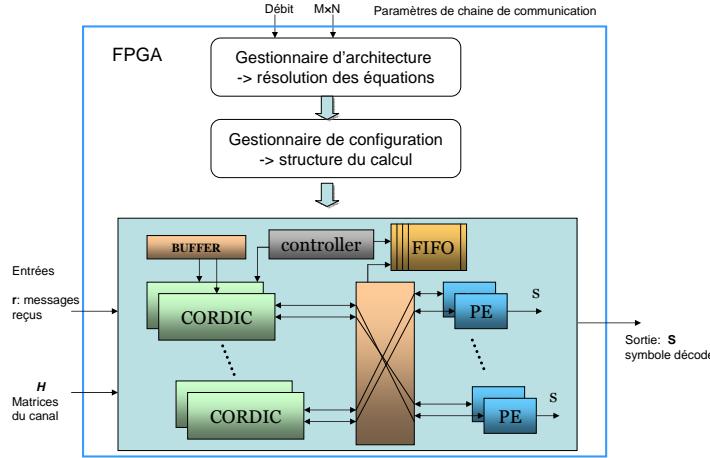


FIG. 3.30 – Flot de gestion de l'architecture et de la reconfiguration

Le gestionnaire d'architecture s'applique à résoudre les équations reliant les caractéristiques de la transmission aux paramètres d'architectures énoncés précédemment. Les données issues de cette analyse sont transmises au gestionnaire de configuration dont la tâche est la construction de l'architecture optimale de traitement [109]. Pour cela le gestionnaire de configuration dispose de ressources matérielles reconfigurables, la reconfiguration matérielle portant sur le nombre d'opérateurs CORDIC implémentés et les chemins de données, ainsi que sur la logique de contrôle.

3.5.7 Simulation

Dans cette partie, nous allons montrer les simulations MATLAB afin de valider notre architecture. Les signaux sources sont indépendants, uniformément distribués. Nous considérons M antennes émettrices et N antennes réceptrices supposées non corrélées. La matrice de canal H est modélisée par une matrice ($N \times M$) avec des éléments indépendants et identiquement distribués (i.i.d.), complexes, de moyenne nulle et gaussiens. Le bruit est complexe blanc gaussien de moyenne nulle et de variance déterminée à partir du rapport signal sur bruit défini. Ces conditions sont identiques pour toutes les simulations de ce chapitre.

La figure 3.31 présente la comparaison de TEB entre l'algorithme V-BLAST classique et V-BLAST Square Root utilisant l'opérateur CORDIC pour MIMO 2×2 avec une modulation 4-QAM. Leurs performances en terme de TEB sont identiques, ce qui montre que notre architecture est applicable à l'algorithme V-BLAST Square Root.

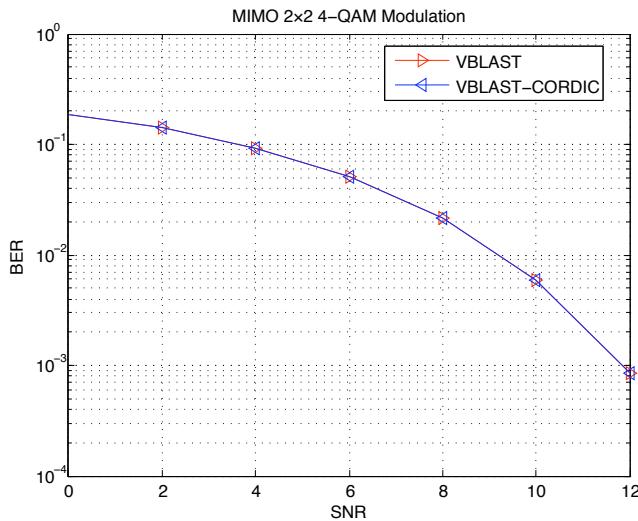


FIG. 3.31 – Le TEB en fonction du SNR.

Nous avons étudié aussi l'impact du nombre d'étages de l'opérateur CORDIC sur le TEB du décodeur. L'évolution du TEB par rapport au nombre d'étages de l'opérateur CORDIC pour les différents SNR est illustrée sur la figure 3.32. Sur cette figure, les courbes TEB pour les rapports signal sur bruit à 2dB et 6dB devient stables après 4^i étages de calculs de l'opérateur CORDIC. Par contre, le courbe TEB pour un rapport signal sur bruit à 10dB devient stable après 6^i étages. Nous avons constaté que le nombre d'étages nécessaire de l'opérateur CORDIC est variable en fonction du rapport signal sur bruit. Plus le rapport signal sur bruit est élevé, plus le nombre d'étages augmente.

Pour un SNR faible, l'impact du nombre d'étages est moins important et on a donc besoin moins d'étages. Quand le SNR devient grand, on a besoin de plus de précision. Il faut donc augmenter le pas de quantification pour avoir plus de précision, c'est à dire le nombre d'étages de l'opérateur CORDIC.

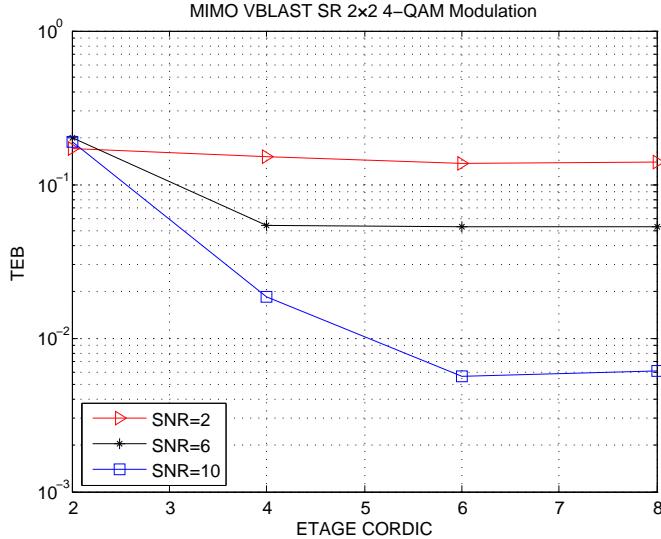


FIG. 3.32 – Le TEB en fonction du nombre d'étages de l'opérateur CORDIC.

Nous comparons maintenant les performances en terme de TEB en utilisant différents nombres d'étages de l'opérateur CORDIC. La figure 3.33 présente le TEB entre V-BLAST classique, V-BLAST utilisant l'opérateur CORDIC de 15 étages et V-BLAST utilisant l'opérateur CORDIC de 6 étages. Les TEB de deux premières réalisations sont identiques. En diminuant le nombre d'étages de CORDIC à partir de 6 étages, le TEB commence à se dégrader.

Il faut noter que la démodulation joue un rôle important dans cette performance. Elle offre une marge pour compenser l'erreur introduite par la réduction du nombre d'étages de l'opérateur CORDIC.

L'erreur quadratique moyenne (EQM) (ou en anglais Mean Square Error (MSE)) comme son nom l'indique permet de mesurer la moyenne du carré de l'erreur entre le signal émis et le signal estimé. Si $\hat{s} = [\hat{s}_1, \dots, \hat{s}_M]^T$ est le vecteur source récupéré, elle est donnée par :

$$EQM_i = \frac{1}{K} \sum_{k=1}^K |s_i(k) - \hat{s}_i(k)|^2. \quad (3.33)$$

où K représente le nombre de symboles utilisés.

La valeur moyenne du EQM sur toutes les sorties est :

$$EQM = \frac{1}{M} \sum_{k=1}^M EQM_k. \quad (3.34)$$

Nous avons étudié l'impact du nombre d'étages de l'opérateur sur EQM.

La figure 3.34 montre les performances en terme d'erreur quadratique moyenne (EQM) en fonction du nombre d'étages de l'opérateur CORDIC. Nous constatons que le nombre

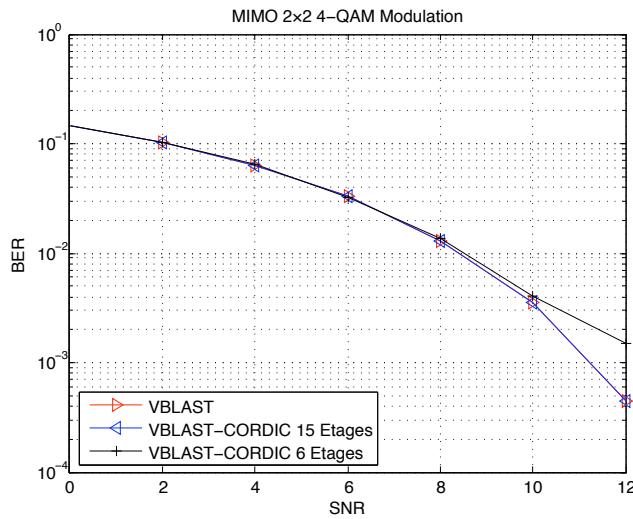


FIG. 3.33 – Le TEB en fonction du SNR.

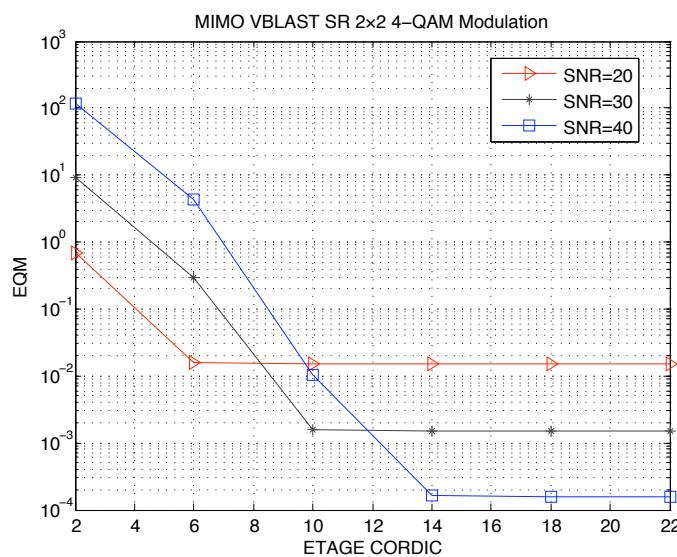


FIG. 3.34 – Le EQM en fonction du nombre d'étages de l'opérateur CORDIC.

d'étages nécessaire est variable pour différents SNR. Cette figure montre que les nombres d'étages nécessaires pour les SNR à 20dB, 30dB, 40dB sont 6, 10, 14 respectivement pour que les courbes EQM deviennent stables. On remarque sur cette figure que plus le SNR est élevé, plus le nombre d'étages nécessaire augmente. C'est à dire que l'erreur de quantification devient importante quand le bruit est faible (SNR élevé). Il faut donc augmenter le pas de quantification pour avoir plus de précision. Dans le cas de l'opérateur CORDIC, le pas de quantification est d'un étage.

Dans l'architecture reconfigurable CBDRA, nous pouvons changer le nombre d'étages de l'opérateur CORDIC pour s'adapter aux différents SNR. Nous pouvons réduire les ressources matérielles en diminuant le nombre d'étages pour les faibles SNR.

3.6 Architecture CBDRA pour d'autres algorithmes MIMO

Comme nous l'avons présenté dans le paragraphe 3.2, l'opérateur CORDIC peut être utilisé dans un grand nombre d'algorithmes du traitement du signal. Nous expliquons ici en détail l'utilisation de l'opérateur CORDIC pour remplacer les méthodes classiques en prenant deux exemples, l'algorithme MIMO MMSE et l'algorithme CMA.

L'étude du nombre optimal d'opérateur CORDIC, comme cela a été fait pour l'algorithme V-BLAST Square Root dans le paragraphe 3.5.4, n'a pas été réalisée pour l'algorithme MMSE et l'algorithme CMA. Les opérateurs CORDIC sont implantés en parallèle pour effectuer les calculs de ces algorithmes.

3.6.1 L'algorithme MIMO MMSE

L'algorithme MIMO MMSE (minimise l'erreur moyenne quadratique) est présenté dans le chapitre 1. Son expression s'écrit :

$$S = (H^*H + \sigma^2 I)^{-1}H^*r. \quad (3.35)$$

Nous allons donc présenter l'utilisation de l'opérateur CORDIC sur l'algorithme MIMO MMSE. Rappelons que dans le chapitre 1 l'algorithme V-BLAST Square Root utilise le critère MMSE. Ces résultats peuvent être appliqués sur l'algorithme MIMO MMSE. L'équation 3.35 prend la forme suivante :

$$S = H_\alpha^+ r = P^{1/2} Q^* r. \quad (3.36)$$

Nous présentons maintenant la déroulement de cet algorithme.

Etape 1 : Calcul de $P^{1/2}$ et Q_α : pour $i = 1, 2, \dots, N$:

$$\begin{bmatrix} 1 & (H)_i P_{i-1}^{M \times M} \\ 0^{M \times 1} & P_{i-1}^{M \times M} \\ -e_i^{N \times 1} & Q_{i-1}^{N \times M} \end{bmatrix} \Theta_i = \begin{bmatrix} \times 0^{I \times M} \\ \times P_i^{M \times M} \\ \times Q_i^{N \times M} \end{bmatrix} \quad (3.37)$$

Dans cette équation, $P_0^{1/2} = \beta I$, $Q_0 = 0^{N \times M}$, e_i est la $i^{\text{ème}}$ colonne de la matrice identité, Θ_i correspond à une transformation unitaire et \times dénote les entrées non utilisées. Après N itérations on obtient $P^{1/2} = P_N^{1/2}$ et $Q_\alpha = Q_N$.

Etape 2 : Calculer le nulling vector :

$$w_{mmse} = P^{1/2} \times Q_\alpha^* \quad (3.38)$$

Etape 3 : Calculer $y = w_{mmse} r$

Etape 4 : Calculer les symboles estimés $\hat{s} = \text{décision}(y)$

3.6.1.1 Simulation

Nous présentons les résultats de simulation pour confirmer l'étude théorique donnée dans les paragraphes précédents. La figures 3.35 montrent que la performance de MMSE classique et celle utilisant l'opérateur CORDIC sont identiques. On a considéré : Modulation 4-QAM, $M = 2$, $N = 2$. Ce qui prouve que notre architecture CBDRA est applicable à l'algorithme MMSE.

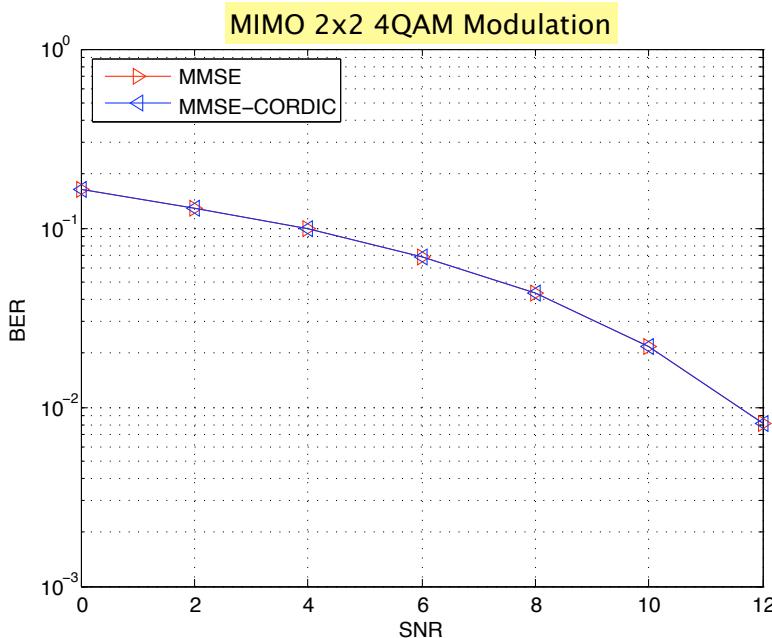


FIG. 3.35 – Constellation 4-QAM. Le taux d'erreur binaire

3.6.2 L'algorithme CMA

Dans le but d'appliquer notre architecture CBDRA pour l'algorithme CMA, nous analysons plus précisément les équations de l'algorithme CMA présenté dans le paragraghe précédent.

Avec la définition de la sortie du filtre BSS et l'équation 2.8, nous pouvons obtenir l'équation suivante :

$$\nabla W_i \Gamma_{cma} = (|z_i(n)|^2 - R) z_i(n) y(n)^* = (|W_i^T y(n)|^2 - R) W_i(n)^T y(n) y(n)^*. \quad (3.39)$$

Prenons par exemple un système MIMO 2×2 , l'équation 3.39 à chaque itération se décrit ci-dessous :

$$\begin{aligned}\nabla W_1(1)\Gamma_{cma} &= ((|W_{11}y(1) + W_1(2)y(2)|^2 - R)(W_{11}y(1) + W_1(2)y(2))y(1)^* \\ &= ((|W_{11}y(1) + W_1(2)y(2)|^2 - R)(W_{11}|y(1)|^2 + W_1(2)y(2)y(1)^*).\end{aligned}$$

$$\begin{aligned}\nabla W_1(2)\Gamma_{cma} &= ((|W_2(1)y(1) + W_2(2)y(2)|^2 - R)(W_2(1)y(1) + W_2(2)y(2))y(1)^* \\ &= ((|W_2(1)y(1) + W_2(2)y(2)|^2 - R)(W_2(1)|y(1)|^2 + W_2(2)y(2)y(1)^*).\end{aligned}$$

$$\begin{aligned}\nabla W_2(1)\Gamma_{cma} &= ((|W_{11}y(1) + W_1(2)y(2)|^2 - R)(W_{11}y(1) + W_1(2)y(2))y(2)^* \\ &= ((|W_{11}y(1) + W_1(2)y(2)|^2 - R)(W_{11}y(1)y(2)^* + W_1(2)|y(2)|^2).\end{aligned}$$

$$\begin{aligned}\nabla W_2(2)\Gamma_{cma} &= ((|W_2(1)y(1) + W_2(2)y(2)|^2 - R)(W_2(1)y(1) + W_2(2)y(2))y(2)^* \\ &= ((|W_2(1)y(1) + W_2(2)y(2)|^2 - R)(W_2(1)y(1)y(2)^* + W_2(2)|y(2)|^2)).\end{aligned}$$

Dans ces équations, la matrice W est multipliée par les vecteurs y(n). L'opérateur CORDIC en mode rotation peut donc être utilisé pour effectuer les mêmes opérations. Cette rotation nécessite d'effectuer une opération CORDIC en mode vecteur pour connaître les angles de rotation. Les vecteurs y(n) peuvent donc être exprimés sous les formes suivantes :

$$y(1) = |y(1)|(cos\theta_1 + jsin\theta_1), \quad (3.40)$$

$$y(2) = |y(2)|(cos\theta_2 + jsin\theta_2). \quad (3.41)$$

$$\text{où } \theta_1 = \tan^{-1} \frac{\Im(y(1))}{\Re(y(1))}, \theta_2 = \tan^{-1} \frac{\Im(y(2))}{\Re(y(2))}.$$

Une fois que les angles de rotation sont calculés, les opérations CORDIC en mode rotation peuvent être effectuées pour calculer les vecteurs de l'algorithme du gradient stochastique. Ces vecteurs dans les équations précédentes deviennent :

$$\begin{aligned}\nabla W_1(1)\Gamma_{cma} &= ((|W_1(1)|y(1)|(cos\theta_1 + jsin\theta_1) + W_1(2)|y(2)|(cos\theta_2 + jsin\theta_2))|^2 - R) \\ &\quad (W_1(1)|y(1)|^2 + W_1(2)|y(2)|(cos\theta_2 + jsin\theta_2)|y(1)|(cos(-\theta_1) + jsin(-\theta_1))).\end{aligned}$$

$$\begin{aligned}\nabla W_1(2)\Gamma_{cma} &= ((|W_2(1)|y(1)|(cos\theta_1 + jsin\theta_1) + W_2(2)|y(2)|(cos\theta_2 + jsin\theta_2))|^2 - R) \\ &\quad (W_2(1)|y(1)|^2 + W_2(2)|y(2)|(cos\theta_2 + jsin\theta_2)|y(1)|(cos(-\theta_1) + jsin(-\theta_1))).\end{aligned}$$

$$\begin{aligned}\nabla W_2(1)\Gamma_{cma} &= ((|W_1(1)|y(1)|(cos\theta_1 + jsin\theta_1) + W_1(2)|y(2)|(cos\theta_2 + jsin\theta_2))|^2 - R) \\ &\quad (W_1(1)|y(1)|(cos\theta_1 + jsin\theta_1)|y(2)|(cos(-\theta_2) + jsin(-\theta_2)) + W_1(2)|y(1)|^2)\end{aligned}$$

TAB. 3.7 – Résumé des calculs remplacés par l'opérateur CORDIC

Méthode	Type d'opération	Nb d'opération
Classique	Multiplication	90
	Addition/soustraction	12
CORDIC	Opération CORDIC	14
	Addition/soustraction	12
	Multiplication	40

$$\nabla W_2(2)\Gamma_{cma} = ((|W_2(1)|y(1)|(cos\theta_1 + jsin\theta_1) + W_2(2)|y(2)|(cos\theta_2 + jsin\theta_2))|^2 - R) \\ (W_2(1)|y(1)|(cos\theta_1 + jsin\theta_1)|y(2)|(cos(-\theta_2) + jsin(-\theta_2)) + W_2(2)|y(1)|^2).$$

Il faut noter que les angles de rotation effectués pour les vecteurs conjugués $y(n)$ sont les inversions des angles des vecteurs $y(n)$. Cela permet d'éviter de recalculer les angles par l'opération CORDIC en mode vecteur.

Le tableau 3.7 compare en résumé le nombre des différents opérateurs utilisés entre l'algorithme SG-CMA et SG-CMA utilisant CORDIC. Nous remarquons que 56% des multiplicateurs sont remplacés par l'opérateur CORDIC.

3.6.2.1 Simulation

Dans un premier temps, nous présentons les résultats de simulation pour confirmer l'étude théorique donnée dans les paragraphes précédents. La figures 3.36 montrent les constellations des signaux reçus, signaux blanchis, et signaux récupérés en utilisant l'algorithme SG-CMA pour des signaux modulés en 4-QAM. On a considéré : SNR = 30dB, M = 2, N = 2 et le pas d'adaptation $\mu = 0.005$. Il faut noter que les constellations sont données avant que l'ambiguité de phase soit enlevée.

Nous présentons aussi les résultats de simulation pour comparer les performances de l'algorithme CMA et de l'algorithme CMA utilisant l'opérateur CORDIC présentés dans ce chapitre. Le bruit est complexe blanc gaussien de moyenne nulle et de variance déterminée à partir du rapport signal sur bruit défini (SNR). Pour chacune des simulations, le signal reçu est préblanchi avant d'appliquer les algorithmes. Les performances sont données en moyennant toutes les sorties. Les courbes pour le SINR sont obtenues à travers 100 réalisations.

Le rapport signal sur bruit plus interférences, plus connu sous l'acronyme SINR pour Signal to Interference plus Noise Ratio, est la mesure de performance la plus répandue dans la séparation de sources. Il permet de mesurer la contribution des autres sources et du bruit sur une des sorties du séparateur. Etant donné le modèle du signal dans 2.1, le SINR est donné par :

$$SINR_k = \frac{|g_{kk}|}{\sum_{l, l \neq k} |g_{lk}|^2 + w_k^T R_b w_k^*}. \quad (3.42)$$

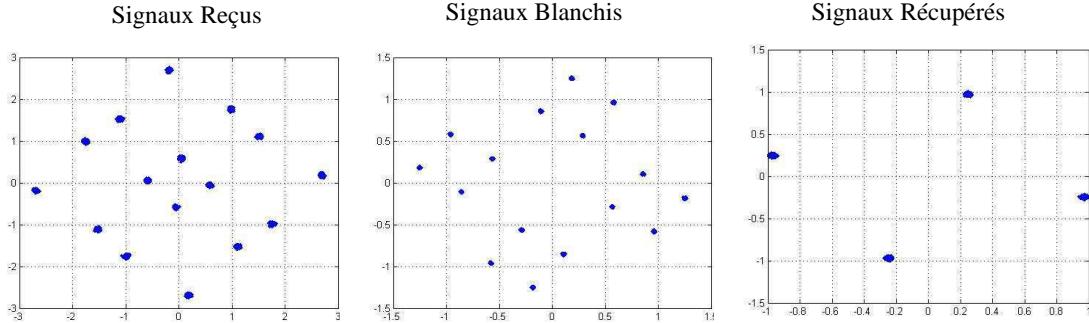


FIG. 3.36 – Constellation 4-QAM.

où : $SINR_k$ est le SINR du signal source présent à la k-ième sortie, $G^T = W^T H$ et $g_k^T = w_k^T H$ est le k-ième filtre canal-séparateur G (g_k est le k-ième vecteur colonne de G). $R_b = E[bb^H]$ est la matrice de covariance du bruit. Dans les simulations et pour éviter l'encombrement, on considère plutôt le SINR moyen sur toutes les sorties. Il est donné par :

$$SINR = \frac{1}{M} \sum_{k=1}^M SINR_k. \quad (3.43)$$

La figure 3.37 représente l'évolution du SINR en fonction des itérations pour l'algorithme adaptatif SG-CMA et celui utilisant l'opérateur CORDIC. Le pas d'adaptation pour l'algorithme SG-CMA est 5×10^{-3} . On observe que la performance de l'algorithme SG-CMA est supérieur que celui utilisant CORDIC en terme de valeur du SINR, parce que l'erreur introduite par l'opérateur CORDIC est accumulée sur le nombre de réalisation. Mais l'algorithme SG-CMA utilisant CORDIC converge à la même vitesse que l'algorithme SG-CMA : à partir de 2500 symboles, la courbe devient stable.

3.7 Conclusion

Nous avons proposé dans ce chapitre l'architecture reconfigurable CBDRA à base d'opérateurs CORDIC pour le traitement du signal, en particulier les systèmes MIMO. L'architecture CBDRA va être implémentée sur FPGA. La structure FPGA et les différents flots de conception sont présentés, ainsi que les gestions de reconfiguration du FPGA et la mise en place des connexions entre la partie fixe et la partie reconfigurable. Nous avons distingué deux types de reconfiguration partielle en fonction de la dépendance des données entre les reconfigurations.

Une relation théorique est proposée pour déterminer le nombre optimal d'opérateurs CORDIC à implémenter pour s'adapter aux caractéristiques des applications. Nous l'avons

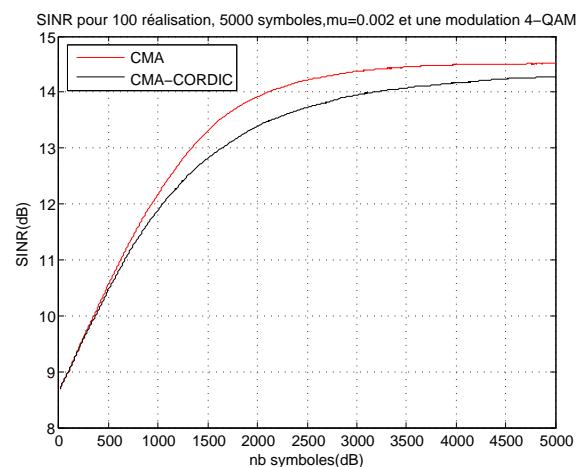


FIG. 3.37 – Comparaison des performances de l'algorithme SG-CMA et SG-CMA utilisant CORDIC. Le SINR en fonction des itérations. 4-QAM, $M = 2$, $N = 2$, SNR = 15dB.

ensuite développée pour calculer le nombre optimal d'opérateurs CORDIC pour un débit donné et un nombre d'antennes donné dans le cas du système MIMO. La charge de calcul importante dans un système MIMO est le calcul de la décomposition de la matrice du canal. Nous avons optimisé ces calculs sur la période de stationnarité du canal. Le nombre d'opérateurs CORDIC peut s'adapter à différentes durées de la stationnarité du canal. Deux exemples ont été pris pour montrer que des débits variables peuvent être obtenus avec différents nombres d'opérateurs CORDIC en parallèle.

Dans cette architecture, la reconfiguration statique est utilisée pour changer le nombre d'opérateurs CORDIC. Les interconnexions des opérateurs CORDIC sont donc réalisées par la sélection des multiplexeurs. La reconfiguration dynamique est utilisée pour changer les connexions entre les opérateurs CORDIC.

Différentes comparaisons en terme de TEB et EQM sont réalisées. L'influence du nombre d'étages CORDIC sur la performance TEB a aussi été analysée dans ce chapitre. Cette étude montre que la complexité peut être réduite en diminuant le nombre d'étages, mais en gardant la même performance en TEB. Ces résultats de simulation Matlab ont validé notre étude théorique sur l'utilisation de l'opérateur CORDIC dans ces algorithmes. Ce qui permet aussi de les comparer ensuite avec les résultats d'implémentation.

Afin de valider cette architecture CBDRA, nous allons implémenter le décodeur MIMO V-BLAST SRA sur une plate-forme hétérogène.

Chapitre 4

Implantation d'algorithmes MIMO

Sommaire

4.1	Plate-forme Radio Logicielle hétérogène	85
4.1.1	Le fonctionnement d'un composant de traitement	87
4.1.2	Gestion de reconfiguration sur FPGA	88
4.2	Implantation de l'algorithme "V-BLAST Square Root"	88
4.2.1	Réalisation matérielle de l'opérateur CORDIC	88
4.2.2	Architecture générale du décodeur "V-BLAST Square Root"	91
4.2.3	Le module de décomposition QR	91
4.2.4	Le module de l'ordonnancement	93
4.2.5	Le module de calcul des vecteurs "nulling"	94
4.2.6	Le module de décodage des données	96
4.2.7	Résultats de implémentation de l'ensemble de l'algorithme	98
4.2.8	Test du décodeur	99
4.3	Conclusion	100

4.1 Plate-forme Radio Logicielle hétérogène

Nous décrirons d'abord dans ce chapitre la plate-forme radio logicielle que nous avons mise en oeuvre dans notre équipe. C'est une plate-forme hétérogène qui comporte un processeur GPP, un DSP et un FPGA. Les deux dernières ressources sont disposées sur une carte de prototypage Sundance. La figure 4.1 représente cette plate-forme. La structure de gestion de configuration hiérachique HDCM présentée dans le chapitre 2 est validée sur cette plate-forme. Cette approche permet de gérer divers types de données de configuration et diverses granularités de configuration [109]. Elle dispose d'une architecture distribuée qui permet de répartir les tâches de gestion de reconfiguration sur les ressources disponibles de la plate-forme.

Le L1_CM est implanté sur le GPP. Il joue le rôle de superviseur général de configuration pour l'ensemble de la plate-forme pendant son fonctionnement. Les L2_CMU sont chargés de récupérer les ordres de configuration et les paramètres des fonctions pour mettre en place des fonctions de traitement. Un L2_CMU est implanté sur chaque module matériel. Il existe donc trois instances de L2_CMU, sur le GPP, sur le DSP et sur le processeur embarqué du FPGA. Le L3_CMU se charge de la reconfiguration ou du changement de

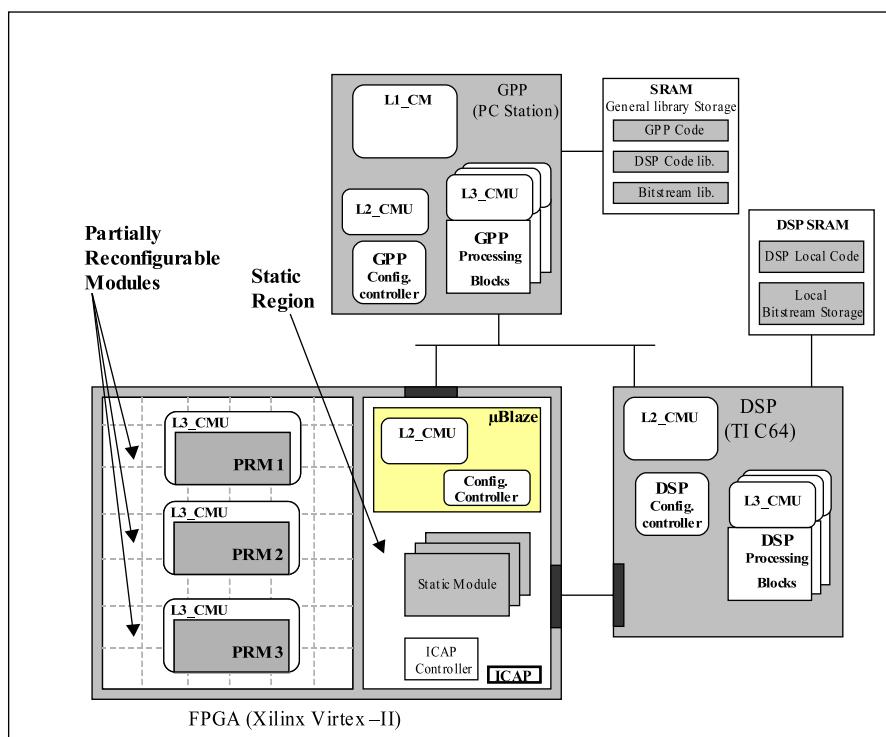


FIG. 4.1 – Architecture de la plate-forme matérielle

paramètres d'une fonction de traitement. Un L3_CM est associé à chaque bloc de traitement. Contrairement au GPP et au DSP, les fonctions de traitement dans un FPGA sous forme d'accélérateur matériel câblé sont de deux types : les traitements implantés dans les régions fixes et ceux implantés dans les régions reconfigurables du FPGA. Les L3_CMU des modules fixes sont inclus dans la fonction et ceux des modules reconfigurables sont réalisés dans le processeur embarqué MicroBlaze. Les L3_CMU sont reliés à un contrôleur de configuration. Dans notre application, un L2_CMU est implanté sur le GPP et deux L3_CMU sont construits séparément sur le GPP et dans le MicroBlaze pour contrôler les modules fixes et reconfigurables.

4.1.1 Le fonctionnement d'un composant de traitement

Le chemin de traitement entre GPP et FPGA est réalisé par une seule liaison sur la plate-forme, appelée *ComPort*. Ce bus est partagé entre la voie de traitement et la voie de configuration pour la transmission des données entre le GPP et le FPGA.

Les différents composants de traitements tels que GPP, DSP et FPGA doivent pouvoir communiquer entre eux dans un schéma flot de données. Un mécanisme de transmission asynchrone des données de type GALS est utilisé comme présenté sur la figure 4.2 afin de résoudre le problème de synchronisation dû aux différences de cadence de fonctionnement entre circuit. Chaque traitement synchrone est encapsulé dans un composant. La production et la consommation des données sont contrôlées par les modules "*ProcessingControl*" du composant. Le rythme de traitement de la chaîne est automatiquement régulé par ces mécanismes.

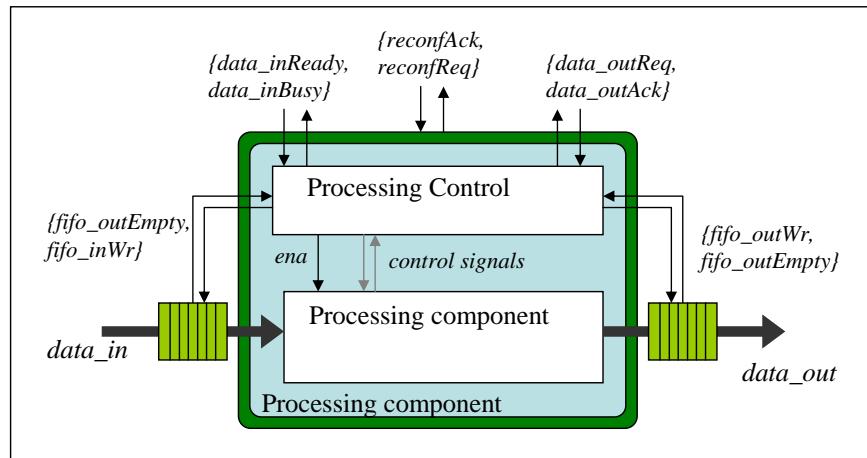


FIG. 4.2 – Composant d'adaptation pour un fonctionnement GALS

Le mécanisme GALS est encapsulé dans tous les composants de la plate-forme. Cette encapsulation permet de développer les fonctionnalités indépendamment les unes des autres que ce soit des composants logiciels ou matériels. Un mécanisme de type "acknowledge/request" permet de contrôler le flot de données.

4.1.2 Gestion de reconfiguration sur FPGA

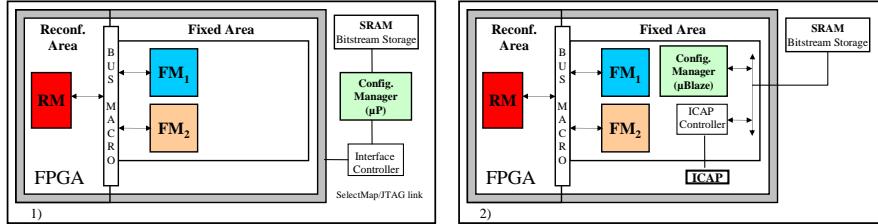


FIG. 4.3 – Deux solutions de gestion de reconfiguration sur FPGA

Un contrôleur de configuration est conçu pour le chargement des fichiers de configuration (bitstreams) dans les registres de configuration du FPGA via une interface de configuration. Deux solutions sont proposées pour réaliser la reconfiguration sur FPGA : un contrôleur externe qui nécessite une interface externe ou un contrôleur interne via une interface interne, comme illustré sur la figure 4.3. Un contrôleur externe peut être un processeur GPP ou un DSP relié aux interfaces externes telles que le port SelectMap ou le BoundaryScan, l'interface SelecMap étant plus rapide. Elle peut fournir une cadence de chargement des bitstreams sur 8 bits à 50 MHz. L'interface de configuration interne est nommée ICAP (Internal Configuration Access Port) [103] disponible sur les Virtex-II/-II Pro, V4 et V5. Elle a les mêmes spécifications que l'interface SelectMap. Elle est contrôlée par un processeur embarqué de type MicroBlaze/PPC ou par une simple machine d'état. La solution que nous avons retenue est d'utiliser un processeur MicroBlaze avec une interface ICAP, ce qui permet d'effectuer la reconfiguration partielle et l'auto-reconfiguration [110] [111]. Ils sont connectés par l'intermédiaire d'un module de contrôle d'ICAP. Le débit du module ICAP dans un FPGA de Xilinx V4 atteint 100MHz à 32 bits, soit 400Mbit/s.

4.2 Implantation de l'algorithme "V-BLAST Square Root"

Notre architecture se trouve principalement sur le FPGA de la plate-forme. Les gestionnaires de l'architecture se situent sur GPP et DSP.

4.2.1 Réalisation matérielle de l'opérateur CORDIC

Afin de répondre à notre spécification d'architecture, nous avons créé notre propre IP (*Intellect Property*) CORDIC qui est flexible et modulaire. Une architecture de type pipeline sera utilisée afin d'augmenter la fréquence de fonctionnement du circuit.

L'algorithme CORDIC est décrit dans le chapitre 2. Nous avons d'abord construit un étage de l'architecture CORDIC à partir des équations 3.6, 3.7, 3.8. Cet étage itératif est représenté sur la figure 4.4. Il est constitué de trois additionneurs/soustracteurs, trois multiplexeurs, deux registres à décalage réalisant une division par 2 pour x et y et une ROM (Read Only Memory) pour stocker les valeurs des angles de rotation. La sélection d_i dépend

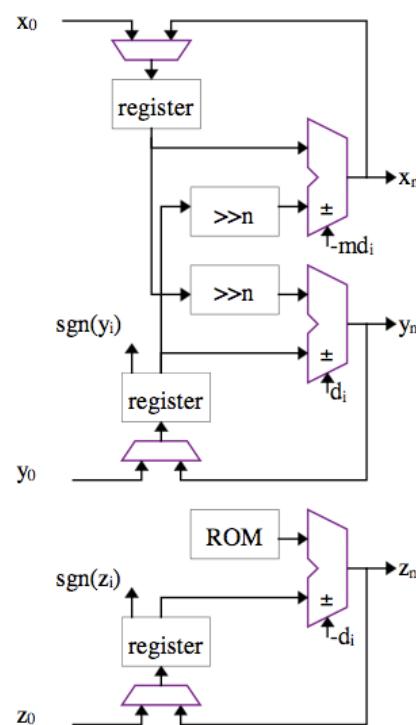


FIG. 4.4 – Etage itératif de l'opérateur CORDIC

du signe de y ou θ en fonction du mode de l'opérateur CORDIC. Les valeurs initiales sont d'abord entrées dans les registres en passant par les multiplexeurs. Au cycle d'horloge suivant, les valeurs des registres sont transmises aux additionneurs/soustracteurs et aux registres à décalage. Les résultats sont enregistrés ensuite dans les registres. L'adresse de la ROM sera incrémentée à chaque coup d'horloge afin d'envoyer les valeurs des angles appropriés à l'additionneur/soustracteur. Les résultats seront obtenus directement en sorties des additionneurs/soustracteurs après N itérations.

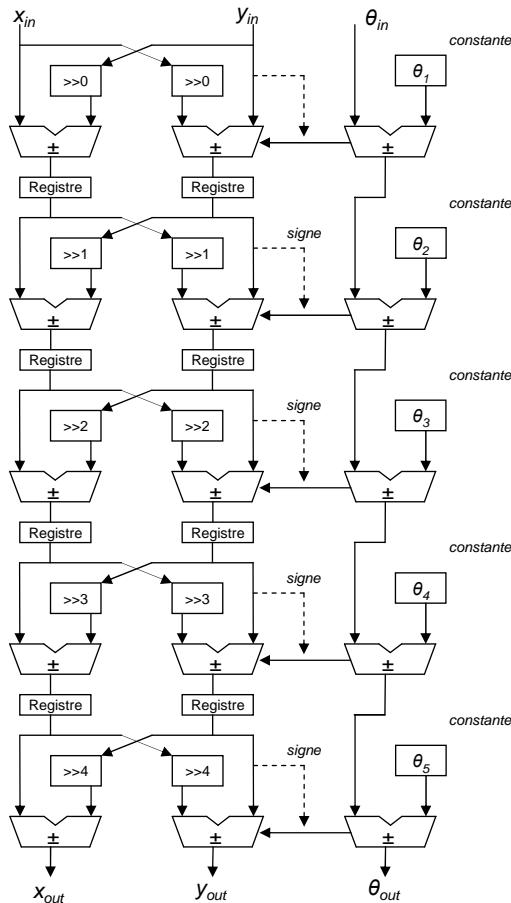


FIG. 4.5 – Schéma de l'opérateur CORDIC en mode rotation

Les traitements itératifs peuvent être exécutés en parallèle par N étages. La figure 4.5 illustre le concept du CORDIC en mode rotation avec ses étages indépendants. Les registres utilisés entre les étages ont pour fonction de construire la structure pipeline, ainsi que resynchroniser les signaux après les opérations effectuées par la logique combinatoire. Cette architecture de type pipeline permet d'augmenter la cadence du circuit.

Le signe de y détermine le sens de rotation. Les valeurs $(x_i, y_i), (x_{i+1}, y_{i+1}), \dots, (x_{i+n}, y_{i+n})$ sont modifiées par rotations élémentaires d'étage en étage.

4.2.2 Architecture générale du décodeur "V-BLAST Square Root"

L'algorithme "V-BLAST Square Root" est présenté dans le chapitre 2 et l'architecture fonctionnelle est illustrée dans le chapitre 3. Nous nous intéressons dans ce chapitre à l'implantation matérielle de l'architecture. Cette architecture est constituée de six modules, comme l'illustre la figure 4.6. Le module d'entrée stocke les composantes du canal H , du rapport $\beta = 1/\sqrt{\sigma_b^2}$ et du message reçu y . Le module de la décomposition de QR calcule les composantes $P^{1/2}$ et Q_α à partir des coefficients représentant le canal H . Le module d'ordonnancement calcule l'ordre optimum et effectue la rotation de Givens pour rendre la matrice $P^{1/2}$ triangulaire et actualise la matrice Q_α avec la même matrice unitaire. Afin d'éviter les calculs de la matrice pseudo inverse et de la matrice inverse, les deux modules utilisent la transformation unitaire. Puis le module suivant calcule les coefficients de l'égaliseur w . Enfin, le dernier module décode les symboles estimés et annule les interférences.

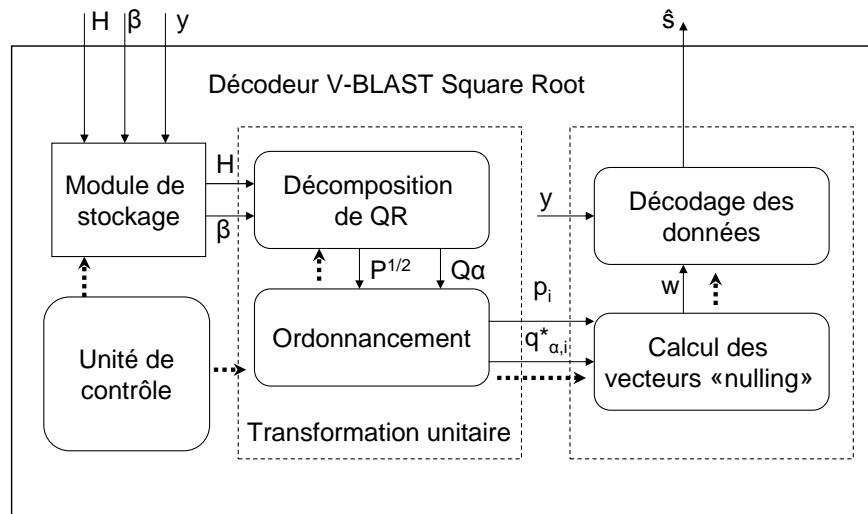


FIG. 4.6 – Architecture générale du décodeur "V-BLAST Square Root"

4.2.3 Le module de décomposition QR

Comme nous l'avons présenté dans le chapitre 3, la principale difficulté de cet algorithme est le calcul des vecteurs "*nulling*". Ce calcul nécessite l'application d'une factorisation de Choleski qui est constituée de la décomposition QR. Cette décomposition peut être exécutée par un réseau triangulaire. Mais le nombre de processeurs élémentaires devient vite trop important pour le système MIMO V-BLAST. Nous utilisons donc une séquence de rotation de Givens présentée par Radar [112] en raison de sa stabilité numérique.

Deux types de transformations unitaires $Q\theta$ et $Q\phi$ sont réalisés par CORDIC pour effectuer la rotation de Givens. La première transformation est utilisée dans le but d'annuler la composante imaginaire sur les nombres complexes. Les autres éléments du vecteur colonne sont déphasés de la même quantité.

$$Q\theta = \begin{bmatrix} e^{-j\theta} & 0 \\ 0 & 1 \end{bmatrix}$$

La seconde agit sur des paires complexes et a pour but d'annuler les parties réelles. Les parties imaginaires sont déphasées de la même quantité.

$$Q\phi = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

Les séquences de rotation de Givens peuvent être implémentées sur une architecture composée d'un simple processeur. Cette architecture basée sur l'opérateur CORDIC (Coordinate Rotation Digital Computing) est proposée par Rader [112].

L'objectif de ce module est de calculer les coefficients des matrice $P^{1/2}$ et Q_α . Il utilise les transformations unitaires pour éviter l'inversion de la matrice $P^{1/2}$. Ces transformations unitaires sont exécutées par des séquences de rotations Givens qui sont basées sur les opérateurs CORDIC. Ce module est décrit théoriquement par B.Hassibi [49] et repris par Z.Guo [51]. Ce dernier emploie un module noté SUPERCELL qui s'appuie sur un concept de réseau systolique proposée par C.M.Rader [112]. Nous utilisons l'architecture plus flexible présentée dans le chapitre 3. Cette architecture utilise un nombre d'opérateurs variable pour s'adapter aux différents besoins et les interconnexions se font soit par multiplexage ou par reconfiguration.

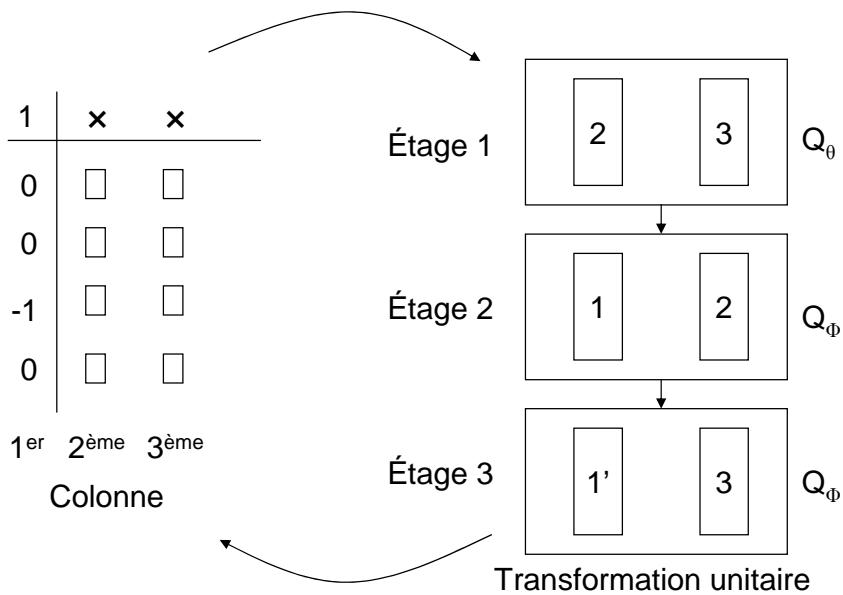


FIG. 4.7 – Principe de fonctionnement du module de la décomposition QR

Le flot de données de trois colonnes issu de l'équation 2.24 traverse trois étages à chaque itération. Chaque étage est basé sur l'opérateur CORDIC. Le premier étage utilise la rotation de Givens de type Q_θ . Les deux autres étages calculent la rotation de Givens

de type Q_ϕ . Les angles de rotation sont pré-calculés par l'opérateur CORDIC en mode vecteur. La succession des calculs des données est illustrée par la figure 4.7. Celle-ci est exécutée en trois cycles. Au premier cycle, les données de la deuxième colonne et celles de la troisième colonne sont exécutées par le premier étage des CORDIC. Au second cycle, les éléments de la deuxième colonne issu du premier cycle et les constantes de la première colonne sont modifiés par le second étage. Au troisième cycle, les opérateurs CORDIC du troisième étage calculent les éléments de la troisième colonne avec ceux de la première colonne.

1	0	0
\times	$P^{1/2}_{11}$	$P^{1/2}_{12}$
\times	$P^{1/2}_{21}$	$P^{1/2}_{22}$
\times	Q_{11}	Q_{12}
\times	Q_{21}	Q_{22}
1 ^{er}	2 ^{ème}	3 ^{ème}
Colonne		

FIG. 4.8 – Présentation des résultats des calculs à chaque itération

Chaque itération répète le même processus jusqu'à obtenir tous éléments des matrices $P^{1/2}$ et Q_α . La figure 4.8 présente les résultats au terme de chaque itération.

Dans le cas d'un système MIMO de 2×2 antennes, le calcul des matrice $P^{1/2}$ et Q_α s'exécute en deux itérations. Chaque itération effectue la même opération avec des coefficients différents.

Les données sont calculées à l'aide des valeurs des coefficients du canal montrées dans l'exemple suivant :

$$\begin{bmatrix} hr11 + hi11 & hr12 + hi12 \\ hr21 + hi21 & hr22 + hi22 \end{bmatrix} \times 2^7 = \begin{bmatrix} -16 - 88i & 16 - 26i \\ 12 + 44i & -6 + 44i \end{bmatrix} \quad (4.1)$$

Les résultats des calculs de la matrice $P^{1/2}$ et Q_α sont résumés dans le tableau 4.1. Les tableaux montrent les différences entre la réalisation en virgule flottante sous Matlab et celle de Modelsim en virgule fixe :

4.2.4 Le module de l'ordonnancement

L'objectif de ce module est de triangulariser la matrice $P^{1/2}$, ainsi d'actualiser la matrice Q_α dans la mesure où un ordonnancement a été effectué. Ce module reprend la même

TAB. 4.1 – Résultat des calculs à la première itération sous Matlab et sous Modelsim

Matlab	Modelsim
$P^{1/2} \times 2^9 \begin{bmatrix} 123,3 - 678,4j & 104,9 - 577,7j \\ 0 & 1017,1 + 1652,8j \end{bmatrix}$	$\begin{bmatrix} 116 - 673j & 96 - 569j \\ 0 & 1005 + 1640j \end{bmatrix}$
$Q_\alpha \times 2^9 \begin{bmatrix} -481,81 + 0j & 52,96 + 0j \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -480 + 0j & 50 + 0j \\ 0 & 0 \end{bmatrix}$

structure que le module précédent. Nous utilisons l'architecture reconfigurable présentée dans le chapitre 3.

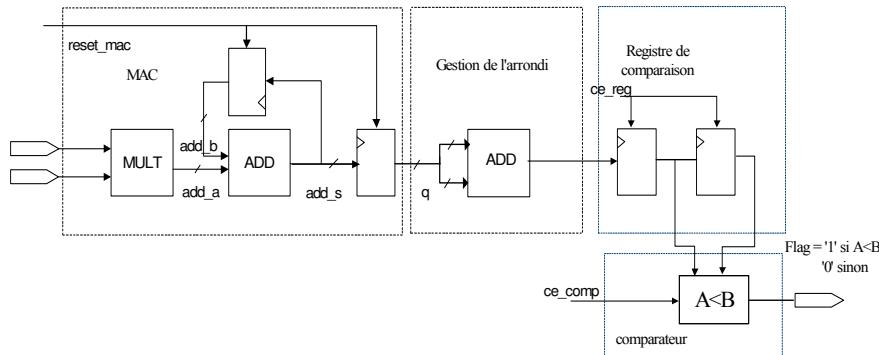


FIG. 4.9 – Schéma du calcul de la norme minimale

Avant d'effectuer la transformation unitaire, la valeur minimale des normes des lignes de la matrice $P^{1/2}$ est calculée. L'objectif du module est de trouver le symbole ayant le meilleur rapport signal sur bruit pour l'estimer en premier, ce qui correspond à la norme minimale de la matrice $P^{1/2}$. Ceci permet de minimiser l'estimation de l'erreur. La figure 4.10 résume l'algorithme réalisé par la structure de calcul de la norme minimale et de sélection. En général, le calcul de la norme minimale peut s'effectuer à l'aide d'un multiplieur accumulateur, d'un registre à décalage et d'un comparateur. Le schéma de cette structure est donné par la figure 4.9.

Dans le but de triangulariser les éléments de la matrice $P^{1/2}$, une transformation unitaire est utilisée. L'enchaînement des calculs des vecteurs est présenté par la figure 4.11.

4.2.5 Le module de calcul des vecteurs "nulling"

Ce module est destiné à calculer les vecteurs "nulling" à partir des coefficients des deux vecteurs complexes $p_i^{1/2}$ et $q_{\alpha,i}^*$. L'équation suivante sera effectuée :

$$w_i = p_i^{1/2} q_{\alpha,i}^*$$

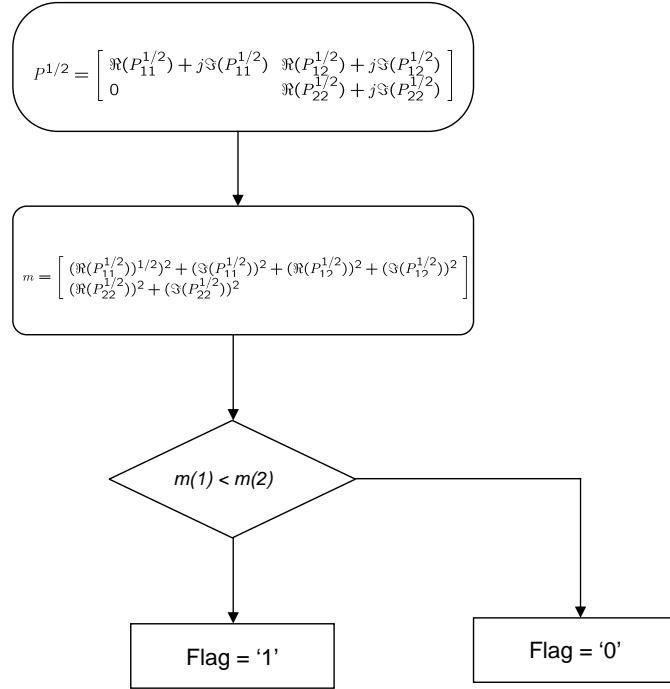


FIG. 4.10 – Algorithme exécuté par le module de calcul de la norme minimale et de sélection

$$\begin{bmatrix} 0 & P_{22}^{1/2} \\ P_{11}^{1/2} & P_{12}^{1/2} \\ Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \times \begin{bmatrix} e^{-j\phi_1} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & P_{22}^{1/2} \\ \bar{P}_{11}^{1/2} & P_{12}^{1/2} \\ Q'_{11} & Q_{12} \\ Q'_{21} & Q_{22} \end{bmatrix}$$

$$\begin{bmatrix} 0 & P_{22}^{1/2} \\ \bar{P}_{11}^{1/2} & P_{12}^{1/2} \\ Q'_{11} & Q_{12} \\ Q'_{21} & Q_{22} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\phi_2} \end{bmatrix} = \begin{bmatrix} 0 & P_{22}^{1/2} \\ \bar{P}_{11}^{1/2} & \bar{P}_{12}^{1/2} \\ Q'_{11} & Q'_{12} \\ Q'_{21} & Q'_{22} \end{bmatrix}$$

$$\begin{bmatrix} 0 & P_{22}^{1/2} \\ \bar{P}_{11}^{1/2} & \bar{P}_{12}^{1/2} \\ Q'_{11} & Q'_{12} \\ Q'_{21} & Q'_{22} \end{bmatrix} \times \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} \cancel{\bar{P}_{21}^{1/2}} & P_{22}^{1/2} \\ 0 & \cancel{\bar{P}_{12}^{1/2}} \\ Q''_{11} & Q''_{12} \\ Q''_{21} & Q''_{22} \end{bmatrix}$$

FIG. 4.11 – Enchaînement du calcul de la triangulation

Le détail de cette équation peut s'écrire comme suivante :

$$w_{BLAST1} = \left[\Re(P_{11}^{1/2}) + j\Im(P_{11}^{1/2}) \right] \times \begin{bmatrix} \Re(Q_{11}) + j\Im(Q_{11}) \\ \Re(Q_{21}) + j\Im(Q_{21}) \end{bmatrix}^*$$

$$w_{BLAST2} = \left[\Re(P_{22}^{1/2}) + j\Im(P_{22}^{1/2}) \right] \times \begin{bmatrix} \Re(Q_{12}) + j\Im(Q_{12}) \\ \Re(Q_{22}) + j\Im(Q_{22}) \end{bmatrix}^*$$

Avec la définition de transposition et conjugaison de la matrice, l'équation devient l'expression suivante :

$$w_1 = \left[\Re(P_{11}^{1/2}) \times \Re(Q_{11}) + \Im(P_{11}^{1/2}) \times \Im(Q_{11}) \right] + j \left[\Im(P_{11}^{1/2}) \times \Re(Q_{11}) - \Re(P_{11}^{1/2}) \times \Im(Q_{11}) \right] \quad (4.2)$$

$$w_2 = \left[\Re(P_{11}^{1/2}) \times \Re(Q_{21}) + \Im(P_{11}^{1/2}) \times \Im(Q_{21}) \right] + j \left[\Im(P_{11}^{1/2}) \times \Re(Q_{21}) - \Re(P_{11}^{1/2}) \times \Im(Q_{21}) \right] \quad (4.3)$$

Ainsi pour les autres coefficients en développant les équations au-dessus.

La figure 4.12 représente le schéma de principe de ce module. Il est basé sur un ensemble de quatre multiplexeurs et de deux multiplieurs accumulateurs. R1, I1 représentent respectivement la partie réelle et la partie imaginaire des coefficients de la matrice $P^{1/2}$, R2, I2 la partie réelle et imaginaire des coefficients de la matrice Q_α . Les opérations effectuées sont résumées par les expressions suivantes :

$$W_r = [R1 \times R2 + I1 \times I2] \quad (4.4)$$

$$W_i = [I1 \times R2 + I2 \times R1] \quad (4.5)$$

Ces calculs seront répétés quatre fois pour obtenir les quatre coefficients de l'égaliseur.

4.2.6 Le module de décodage des données

Le but de ce module est de décoder les données en sortie du module précédent. Ce module est basé sur des processeurs PE. Le nombre de PE utilisé est adapté en fonction du débit de décodage du système V-BLAST.

Ce module utilise un multiplieur accumulateur complexe pour calculer les symboles estimés. C'est un produit vectoriel comme le montre l'équation suivante :

$$\Re(\tilde{y}_1) + j\Im(\tilde{y}_1) = \begin{bmatrix} \Re(w'_1) + j\Im(w'_1) & \Re(w'_2) + j\Im(w'_2) \end{bmatrix} \times \begin{bmatrix} \Re(y_1) + j\Im(y_1) \\ \Re(y_2) + j\Im(y_2) \end{bmatrix} \quad (4.6)$$

Quatre multiplieurs accumulateurs sont utilisés dans ce module afin de maximiser le débit de décodage. Du fait que ces multiplieurs accumulateurs sont en parallèle, les parties réelles et imaginaires sont traitées en même temps.

Ensuite, une opération de multiplication sera effectuée entre le vecteur des coefficients représentant le canal et le symbole estimé. Le résultat de cette opération permet de retrancher la quantité du produit au signal y et devient le nouveau signal reçu. Dans notre

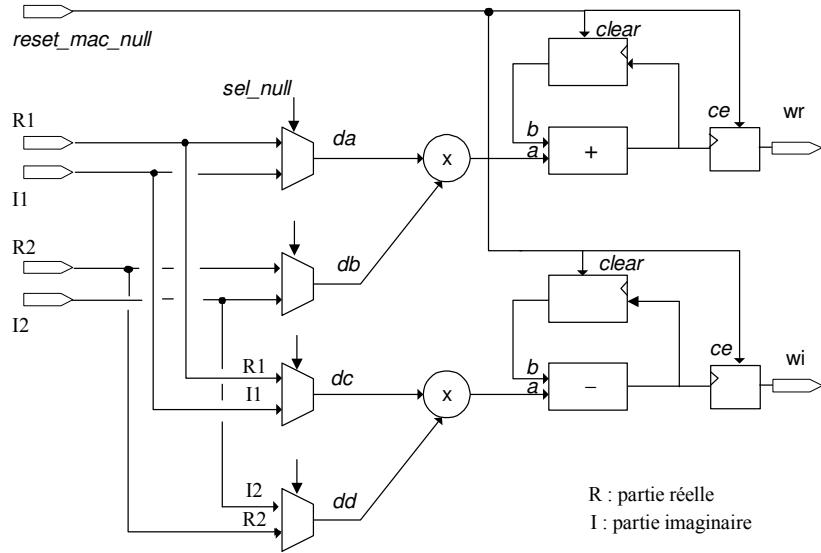
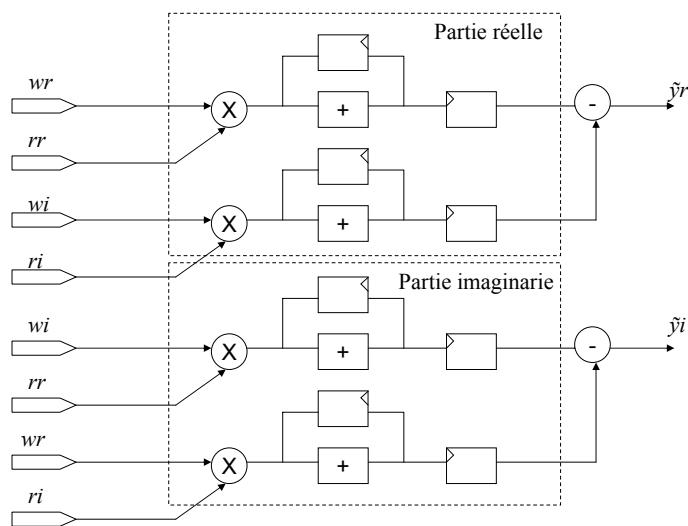
FIG. 4.12 – Schéma du module de calcul du vecteur *nulling*

FIG. 4.13 – Multiplieur accumulateur complexe

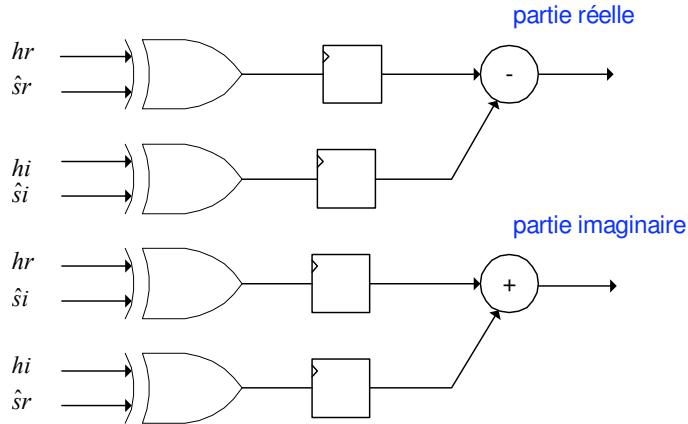


FIG. 4.14 – Opérateur de multiplication complexe

application, l'opération de multiplication complexe de y et les symboles QPSK sont réalisée à l'aide d'une simple fonction ou exclusif, ainsi qu'un additionneur et un soustracteur. Cette opération est présentée sur la figure 4.14.

4.2.7 Résultats de implémentation de l'ensemble de l'algorithme

Les messages reçus sont traités sur 16 bits. Les symboles estimés sont calculés en parallèle. A chaque cycle nous pouvons obtenir 2 symboles en parallèle. La modulation utilisée est une QPSK. Le tableau 4.2 résume les résultats obtenus en fonction du nombre d'opérateurs CORDIC utilisés. Dans ce tableau, les trois derniers modules sont basés sur PE.

Nous comparons les implémentations de plusieurs architectures dans un circuit FPGA Virtex-II et Virtex-4 de Xilinx afin de déterminer l'architecture la plus adaptée à la réalisation matérielle. La première est une architecture parallèle et pipeline qui donne la meilleure performance en débit mais avec un coût en surface important. Puis nous l'optimisons en surface en diminuant le nombre d'opérateurs CORDIC. Le débit reste identique avec la première implémentation en supposant la stationnarité du canal à $1\mu s$. Une autre optimisation utilise 3 CORDIC. Elle peut être utilisée quand le débit n'est pas essentiel ; elle permet de diminuer la complexité de l'architecture. La durée du chemin critique est de 6.729 ns, donc la fréquence maximum est de 148.6 MHz. Enfin les deux dernières réalisations utilisent la reconfiguration dynamique. La stationnarité du canal est donc plus longue que les réalisations avec la reconfiguration statique, parce que le temps de reconfiguration est très long. En utilisant un FPGA de Xilinx Virtex-4 pour effectuer la reconfiguration dynamique, la performance du décodeur reste correcte.

Comme nous l'avons montré dans le chapitre 3, le nombre de d'étages nécessaire est variable pour différents SNR. Par exemple, la performance en terme de EQM est correcte en utilisant l'opérateur CORDIC de 6 étages. Nous avons donc réalisé une implantation du décodeur V-BLAST Square Root avec l'opérateur CORDIC de 6 étages. La table 4.3 montre que le nombre de Slices de FPGA est réduit à 2220, soit 22% de moins par rapport à une réalisation avec l'opérateur CORDIC de 15 étages.

TAB. 4.2 – Résultats des synthèses du décodeur V-BLAST Square Root (avec l'opérateur CORDIC en 15 étages)

FPGA Xilinx	Type reconf	Nb de CORDIC	Slices	% surface	Fréq (MHz)	Débit (Mb/s)	Canal
Virtex-II x2v-2000		50 (29+8+8+5V)	29036	Non app.	148.6	600	1μ s
		16(5+4+4+3V)	14380	Non app.		600	
	statique	8(3+2+2+1V)	9936	90%		360	
		4(3+1V)	4505	40%		135	
x2v-2000	dynamique	4(3+1V)	2857	26%		10	1ms
Virtex-4		4(3+1V)	2857	4%	159	150	

TAB. 4.3 – Résultats des synthèses du décodeur V-BLAST Square Root (avec l'opérateur CORDIC en 6 étages)

FPGA Xilinx	Type reconf	Nb de CORDIC	Slices	% surface	Fréq (MHz)	Débit (Mb/s)	Canal
x2v-2000	dynamique	4(3+1V)	2220	21%		10	1ms
Virtex-4		4(3+1V)	2220	3%	159	150	

Les modules de l'architecture sont synthétisés individuellement par l'outil de synthèse de Xilinx ISE 6.3 pour Virtex-II et ISE 9.1 pour Virtex-4. Avec l'aide de PlanAhead de Xilinx, nous pouvons définir les zones de chaque module de l'architecture dans un FPGA Virtex. Ensuite nous utilisons ISE 6.3 et ISE 9.1 pour placer-router ces modules. La figure 4.15 montre une vue de l'architecture du détecteur V-BLAST Square Root après placement-routage dans un FPGA de Xilinx Virtex II 2000 et Virtex-4.

4.2.8 Test du décodeur

Nous avons développé une plate-forme dont le principe est présenté dans le paragraphe 4.1 afin de valider le fonctionnement des circuits proposés. Dans un premier temps nous avons réalisé un circuit permettant l'interface entre la carte et le PC. Les données sont stockées en mémoire, avant d'être traitées par le FPGA. Les simulations permettent de valider l'architecture de l'opérateur CORDIC et les modules du décodeur MIMO V-BLAST. Les fonctions Matlab sont remplacées matériellement par les fonctions correspondantes implémentées au sein du FPGA, puis les résultats obtenus en virgule fixe sont comparés avec la courbe de référence tracée à partir des résultats de simulations calculés en virgule flottante par Matlab.

Ce banc de test permet de relever le taux d'erreur binaire en temps réel et valider la fiabilité de l'ensemble du décodeur. La figure 4.16 donne le taux d'erreurs binaire en fonction de E_b/N_0 .

Les performances en terme de TEB sont identiques pour l'algorithme V-BLAST classique, utilisant l'opérateur CORDIC sous Matlab et celui sous VHDL. Cela signifie que les décodeurs utilisant l'opérateur CORDIC en virgule fixe et en virgule flottante ont la

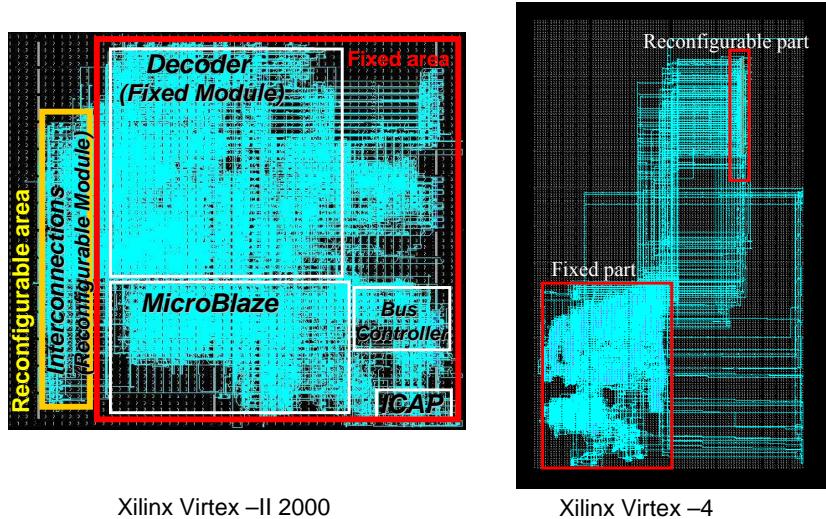


FIG. 4.15 – Vue du décodeur après placement/routage

même performance en terme de TEB. Les données en virgule flottante sous MATLAB sont en 64 bits et celles en virgule fixe sous VHDL sont en 16 bits avec une mantisse de 10 bits. Cela est expliqué par la précision de l'opérateur CORDIC donnée non seulement par le nombre de bits mais aussi par le nombre d'étages. Cette courbe valide notre architecture reconfigurable CBDRA pour l'application du système MIMO.

4.3 Conclusion

La plate-forme hétérogène a d'abord été présentée dans ce chapitre, ainsi que la technique de synchronisation GALS simplifiant la communication entre différents types de processeurs, tels que le GPP, le DSP et le FPGA. La structure pipeline de l'opérateur CORDIC a été construite. Une implémentation du décodeur V-BLAST SRA a été ensuite réalisée sur la plateforme hétérogène. Le bloc de traitement est implanté sur FPGA et les gestionnaires du décodeur sont réalisées sur le GPP et le DSP.

Les différents modules fonctionnels du décodeur V-BLAST SRA sont décrits ainsi que leur réalisation sur FPGA. Plusieurs implémentations avec différents nombres d'opérateurs CORDIC sont comparées en taux d'occupation du FPGA et en débit. Nous vérifions que les TEB du décodeur obtenus d'une part sous Matlab et d'autre part avec sa réalisation matérielle sont identiques. Cela montre que la précision de l'opérateur CORDIC dépend à la fois du nombre de bits utilisé pour la représentation des nombres mais aussi du nombre d'étages de calcul de l'opérateur. Ces résultats ont validé les études théoriques du chapitre précédent mais aussi validé les descriptions VHDL du décodeur. Les résultats de synthèse ont montré les gains en surface du FPGA.

Dans ce chapitre, le temps de reconfiguration a été pris en compte dans le calcul de la performance du décodeur en terme de débit. Nous avons remarqué que la performance du

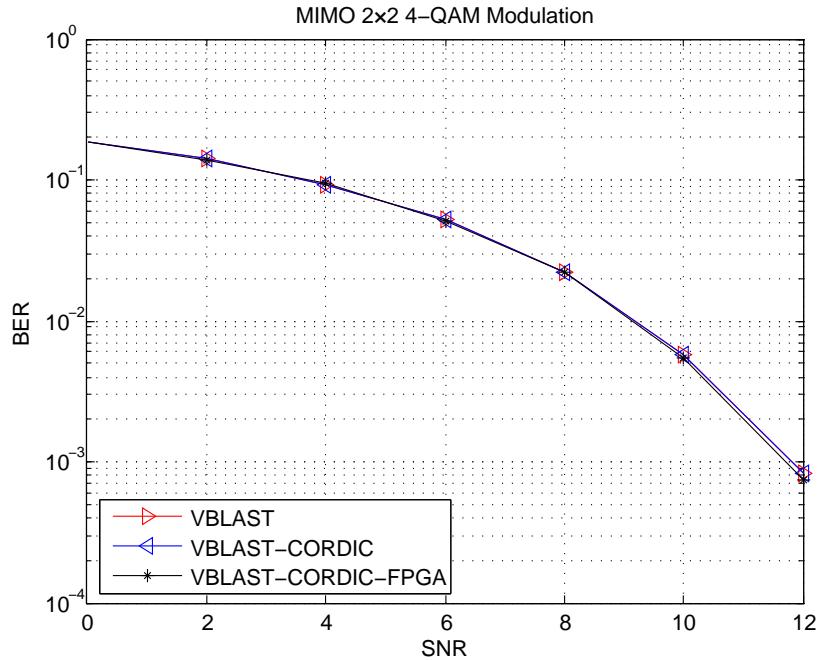


FIG. 4.16 – Taux d’erreurs binaire en fonction de SNR

décodeur utilisant la reconfiguration dynamique est pénalisée par le temps de reconfiguration qui est très long par rapport à la cadence de calcul du traitement. Mais l’amélioration de la technologie des composants reconfigurables, comme les familles Virtex-4, Virtex-5 de Xilinx, permettra d’atteindre des débits plus élevés.

Conclusion

Les travaux présentés dans ce document s'intéressent à l'architecture reconfigurable à base d'opérateur CORDIC permettant l'exécution d'applications de traitement de signal, notamment la radio logicielle. Dans ce mémoire, nous avons étudié particulièrement les architectures reconfigurables pour les systèmes MIMO et des possibilité d'appliquer cette architecture aux autres algorithmes de traitement du signal. Dans un premier temps, nous avons brièvement comparé les algorithmes MIMO les plus connus et avons choisi l'algorithme V-BLAST Square Root comme meilleur compromis en terme de complexité et de performance. Nous avons décrit en détail cet algorithme.

Les architectures reconfigurables offrent de nouvelles alternatives entre la grande flexibilité des processeurs programmables DSP et les hautes performances des circuits spécifiques ASIC. Après avoir analysé et comparé brièvement les architectures reconfigurables, nous nous appuyons sur la technologie FPGA pour répondre à notre besoin de reconfiguration.

Au cours de cette étude, nous avons constaté que l'architecture reconfigurable est indispensable dans les applications des communications numériques et de la radio logicielle. A travers des études sur les systèmes MIMO et les architectures reconfigurables, nous avons présenté une architecture reconfigurable à base d'opérateur CORDIC, nommée CBDRA.

Nous avons ensuite présenté le principe de l'opérateur CORDIC et montré la puissance et la généralité de l'opérateur CORDIC à travers ses applications dans de nombreux algorithmes classiques de traitement du signal. L'utilisation de l'opérateur CORDIC présente également de nombreux avantages pour l'intégration au niveau architectural grâce à son excellente adaptation aux contraintes des VLSI.

Nous avons établi une relation théorique entre le nombre d'opérateurs CORDIC et les caractéristiques des applications. Une architecture optimale peut être obtenue pour les systèmes MIMO en appliquant cette relation. Cette architecture offre une grande flexibilité vis à vis du nombre d'antennes et des différents débits données. Il est adapté aux différents besoins en utilisant différents nombres d'opérateurs CORDIC. Nous avons aussi étendu l'utilisation de l'opérateur CORDIC sur l'ensemble de l'algorithme V-BLAST Square Root ce qui permet d'avoir plus de flexibilité vis à vis de l'ordonnancement de l'opérateur CORDIC.

Afin d'étudier l'utilisation de l'architecture CBDRA pour d'autres algorithmes de décodage MIMO, nous avons analysé aussi l'algorithme MMSE et l'algorithme de détection MIMO aveugle CMA avec des simulations sous Matlab. Les résultats de Matlab ont montré que notre architecture peut s'adapter aux différents algorithmes de décodage MIMO.

Finalement, nous avons réalisé l'implantation du détecteur MIMO basé sur l'algorithme V-BLAST "Square Root" afin de valider l'architecture CBDRA. Cette architecture nous permet d'obtenir des débits considérables, grâce au parallélisme de l'architecture. L'ar-

chitecture est définie par bloc permettant une reconfiguration partielle des ressources matérielles afin de minimiser le coût de la reconfiguration. Cette architecture est adaptée aux besoins des futurs systèmes de radiocommunication par la reconfiguration dynamique.

En résumé, les principales contributions de ces travaux de thèse sont :

1. Nous avons proposé une architecture reconfigurable à base d'opérateur commun CORDIC, nommée CBDRA, pour répondre aux besoins de la multitude des standards de communication. Le système MIMO V-BLAST a été étudié en appliquant cette architecture CBDRA. Cette architecture est reconfigurable en supportant différents nombres d'antennes, différents types de modulations et de propagation.
2. La reconfiguration dynamique est utilisée dans l'architecture du décodeur MIMO V-BLAST Square Root, afin d'optimiser les ressources matérielles.
3. Deux autres algorithmes MIMO sont analysés afin de montrer la possibilité d'utiliser cette architecture pour différents algorithmes.

Les perspectives de cette étude sont multiples. A court terme, il serait intéressant d'implémenter les trois derniers modules du décodeur V-BLAST Square Root en utilisant l'opérateur CORDIC sur FPGA et de comparer la complexité aux autres versions du décodeur. Ainsi l'algorithme de la séparation de sources SG-MMA permettrait de montrer plus d'intérêts dans l'utilisation de notre architecture reconfiguration CBDRA. Une piste d'étude possible, est de chercher un ordonnancement optimal, de manière à ne jamais laissé inactif un opérateur CORDIC.

A long terme, un autre prolongement de ce travail est de continuer à étudier l'opérateur CORDIC comme un opérateur commun dans la paramétrisation de la radio logicielle. Comme le montre cette thèse, on peut utiliser l'opérateur CORDIC pour la FFT, la décomposition en valeur sigulière et les modulations, etc, l'opérateur CORDIC jouera un rôle important dans l'équipement de la radio logicielle et apportera plus de solutions que les opérateurs classiques tels que le multiplieur, l'additionneur.

Publications personnelles

Revues

- Hongzhi Wang and Pierre Leray and Jacques Palicot, "*A Reconfigurable Architecture for MIMO Square Root Decoder*", Reconfigurable Computing : Architectures and Applications,Lecture Notes in Computer Science,pp. 317-322, 2006
- Hongzhi Wang and Pierre Leray and Jacques Palicot, "*Reconfigurable Architecture for MIMO systems based on CORDIC operators*", Elsevier, Comptes rendus Physique, vol. 7, pp. 735-750, 2006

Conférences internationales

- Hongzhi Wang, Pierre Leray and Jacques Palicot, "*A CORDIC-based dynamically reconfigurable FPGA architecture for signal processing algorithms*", The XXIX General Assembly of the International Union of Radio Science(URSI08), August 07-16, 2008, Chicago, Illinois, USA.
- Loig Godard, Hongzhi Wang, Christophe Moy, Pierre Leray, "*Common Operators Design on Dynamically Reconfigurable Hardware for SDR Systems*", SDR Forum Technical Conference'07, 5-9 November 2007, Denver, Colorado, USA.
- Hongzhi Wang and Pierre Leray and Jacques Palicot, "*An Efficient MIMO V-BLAST Decoder Based on a Dynamically Reconfigurable FPGA Including its Reconfiguration Management*", ICC '08. IEEE International Conference on Communications, 2008, Beijing, China, May 2008.
- Hongzhi Wang and Jean-Philippe Delahaye and Pierre Leray and Jacques Palicot, "*Managing dynamic reconfiguration on MIMO Decoder*", 21th International Parallel and Distributed Processing Symposium (IPDPS 2007), 26-30 March 2007, Long Beach, California, USA.
- Hongzhi Wang and Pierre Leray and Jacques Palicot, "*A reconfigurable architecture for MIMO detection using CORDIC operator*", 4th Karlsruhe Workshop on Software Radios (wrs06), Karlsruhe, German, March 2006.
- Hongzhi Wang and Pierre Leray and Jacques Palicot, "*A Reconfigurable Architecture for MIMO Square Root Decoder*", International Workshop on Applied Reconfigurable Computing (ARC) 2006, Delft, The Netherlands, March 1-3, 2006.

Conférences nationales

- Hongzhi Wang, Pierre Leray, Jacques Palicot, "Architecture reconfigurable CBDRA basée sur l'opérateur CORDIC pour le traitement du signal : Applications aux récepteurs MIMO", (article soumis au) Gretsi 09, Dijon, France, Septembre 2009.
- Hongzhi Wang, Pierre Leray, Jacques Palicot et Jean-Philippe Delahaye, "Utilisation de la reconfiguration dynamique partielle dans l'implémentation d'un décodeur MIMO V-BLAST", Gretsi 07, Troyes, France, Septembre 2007.

Communication orales

- Hongzhi Wang, Pierre Leray, Jacques Palicot, "FPGA Reconfigurable architecture for MIMO V-BLAST detector , CODEST Telecommunications workshop", Xi'an, Chine, November 23-25, 2006.
- Hongzhi Wang, Pierre Leray, Jacques Palicot, "A reconfigurable architecture for MIMO Square Root Algorithm detector based on CORDIC operator", IST Mobile Summit'06, MEWCOM Workshop, 9 june 2006, Mykonos, Greece.
- Hongzhi Wang, Pierre Leray, Jacques Palicot, "Architecture reconfigurable pour les systèmes MIMO à base d'opérateurs CORDIC Journées Scientifiques", "Vers des radiocommunications reconfigurables et cognitives" du CNFRS, 28 et 29 mars 2006, PARIS.

Divers

- Hongzhi Wang, Pierre Leray, Jacques Palicot, "A CORDIC based dynamically reconfigurable FPGA architecture for signal processing algorithms in the Software Radio context : the MIMO SRA algorithm case study.", Commission Télécommunication du Conseil Scientifique, SUPELEC, 22 Octobre 2008.
- Hongzhi Wang, Pierre Leray, Jacques Palicot, "Utilisation de la reconfiguration dynamique partielle dans l'implémentation d'un décodeur MIMO", Journée des doctorants en électronique de Bretagne, IETR, Rennes, France, 20 Juin 2007.
- Hongzhi Wang, "Radio logicielle et reconfiguration dynamique", Doctoriales de Bretagne 2007, Vannes, France, 18-23 Novembre 2007.
- Hongzhi Wang, Pierre Leray, Jacques Palicot, "Architecture reconfigurable pour les systèmes MIMO à base d'opérateurs CORDIC", Séminaire SCEE du 20 Avril 2006, Supélec, Campus de Rennes.

Annexe

Annexe A

Transformée de Fourier rapide (Fast Fourier transform - FFT)

La définition de la DFT est donnée dans l'équation 3.20. Park a proposé une méthode pour générer les coefficients *twiddle* de la DFT in [113]. La FFT (Fast Fourier Transform) est une méthode rapide pour calculer la DFT. Elle est directement issue d'une réorganisation du calcul des matrices de la DFT.

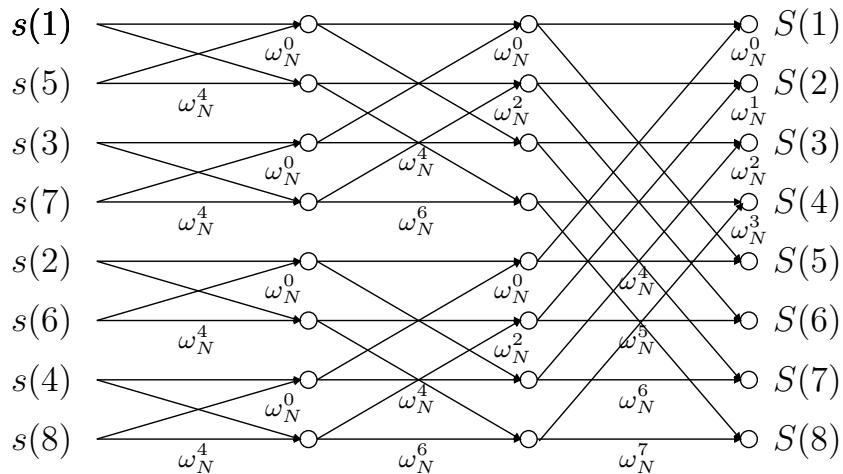


FIG. A.1 – Graphe d'une FFT sur 8 points

La figure A.1 illustre l'exemple d'un graphe d'une FFT sur 8 points reposant sur l'utilisation de l'opérateur papillon.

Heyne [114] présente une méthode pour implémenter la FFT en utilisant l'opérateur CORDIC. Deux types d'éléments de base sont proposés dans cette structure. Le premier type, nommé type I, est composé de deux opérateur CORDIC en mode rotation d'angle $\pi/4$, comme le montre la figure A.2. Il calcule séparément les parties réelles et les parties imaginaires de deux valeurs complexes. Le deuxième type, type II, est illustré par la figure

A.3. Il est construit de la même façon que le premier mais avec un angle de rotation différent de $3\pi/4$.

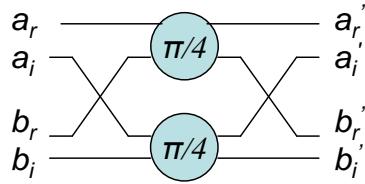


FIG. A.2 – Structure de type I

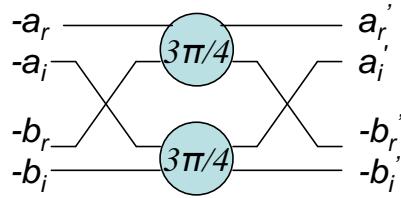


FIG. A.3 – Structure de type II

La structure de calcul de la FFT à base d'opérateurs CORDIC est très voisine de celle de la FFT papillons classique. Il est à noter qu'une correction et une réorganisation de l'ordre sont nécessaires pour retrouver les mêmes résultats que la FFT standard.

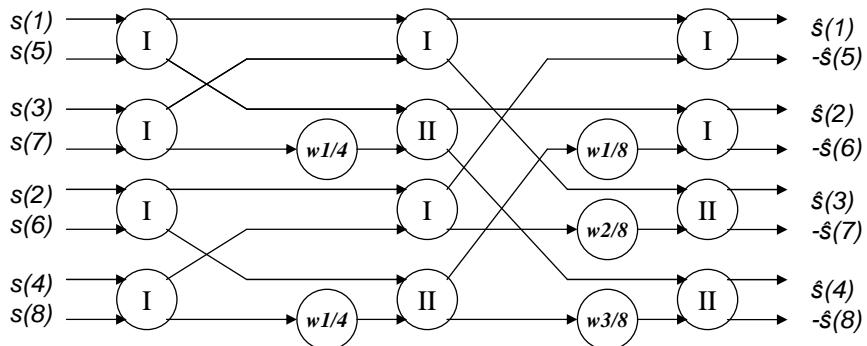


FIG. A.4 – Graphe d'une FFT en utilisant les deux types d'opérateur CORDIC

Annexe B

Décomposition en valeur singulière (SVD)

Beaucoup d'applications du traitement de signal nécessitent des calculs de factorisation de matrice comme de Eigenvalue décomposition en valeur singulière (SVD). Une structure de Jacobi-like est utilisée pour les calculs de ces factorisations grâce à son adaptation à l'implantation de la structure parallèle. L'algorithme CORDIC calcule l'arc tangente et effectue des rotations des vecteurs qui sont les opérations primitives exigées par la décomposition SVD. Delosme [95], Yang et Bohme [96] ont démontré l'utilisation de CORDIC dans une transformation de la matrice 2×2 ce qui est nécessaire dans les calculs de la décomposition SVD d'une matrice réelle. Une approche basée sur l'opérateur CORDIC redondant et on-line est proposée récemment par Ercegovac et Lang [97].

La décomposition en valeur singulière (SVD) d'une matrice $M \in C^{m \times n}$ est exprimée par l'équation suivante :

$$M = U \sum V^H, \quad (\text{B.1})$$

où $U \in C^{m \times m}$ and $V \in C^{n \times n}$ sont des matrices unitaires et $\sum \in R^{m \times n}$ est une matrice diagonale.

La méthode de Jacobi pour les calculs d'Eigenvalue peut être étendue aux calculs de la décomposition SVD d'une matrice carrée $M \in C^{m \times m}$. La matrice devient une matrice diagonale en appliquant une séquence de SVD.

La décomposition SVD d'une matrice réelle 2×2 est décrite par l'équation suivante :

$$R(\theta_r)^T \begin{bmatrix} a & b \\ c & d \end{bmatrix} R(\theta_l) = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} \quad (\text{B.2})$$

où

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad (\text{B.3})$$

est une matrice de rotation et la matrice entrée est

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (\text{B.4})$$

Plusieurs méthodes pour déterminer les angles θ_l et θ_r sont présentés dans [115]. Une approche [116] propose de déterminer d'abord θ_{sum} et θ_{diff} comme décrit ci-dessous :

$$\theta_{sum} = (\theta_r + \theta_l) = \tan^{-1}\left(\frac{c+b}{d-a}\right) \quad (\text{B.5})$$

$$\theta_{diff} = (\theta_r - \theta_l) = \tan^{-1}\left(\frac{c-b}{d+a}\right) \quad (\text{B.6})$$

Les angles θ_l et θ_r peuvent être déduits à partir de ces équations.

La décomposition SVD complexe peut être exécutée en utilisant l'algorithme Jacobi. La matrice de base 2×2 à éléments complexes est transformée en une matrice diagonale dans l'étape de base de l'algorithme Jacobi-SVD complexe.

Plusieurs solutions sont proposées dans la littérature [117] pour rendre une matrice complexe diagonale en utilisant la transformation unitaire. Forsythe [116] utilise une transformation unitaire à deux côtés d'une matrice initiale, comme il décrit ci-dessous :

$$\begin{bmatrix} c_\phi e^{i\theta_\alpha} & s_\phi e^{i\theta_\beta} \\ s_\phi e^{i\theta_\gamma} & c_\phi e^{i\theta_\delta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} c_\varphi e^{i\theta_\xi} & s_\varphi e^{i\theta_\eta} \\ s_\varphi e^{i\theta_\zeta} & c_\varphi e^{i\theta_\omega} \end{bmatrix} = \begin{bmatrix} We^{i\theta_w} & 0 \\ 0 & Ze^{i\theta_z} \end{bmatrix} \quad (\text{B.7})$$

où

$$\tan(\theta_\alpha - \theta_\beta) = -\frac{AC \sin(\theta_a - \theta_c) + BD \sin(\theta_b - \theta_d)}{AC \cos(\theta_a - \theta_c) + BD \cos(\theta_b - \theta_d)} \quad (\text{B.8})$$

$$\tan(\theta_\eta - \theta_\omega) = -\frac{AB \sin(\theta_a - \theta_b) + CD \sin(\theta_c - \theta_d)}{AB \cos(\theta_a - \theta_b) + CD \cos(\theta_c - \theta_d)} \quad (\text{B.9})$$

$$\tan(\theta_\phi - \theta_\varphi) = -\frac{Be^{i(\theta_\alpha + \theta_\omega + \theta_b)} + Ce^{i(\theta_\beta + \theta_\eta + \theta_c)}}{De^{i(\theta_\beta + \theta_\omega + \theta_d)} - Ae^{i(\theta_\alpha + \theta_\eta + \theta_a)}} \quad (\text{B.10})$$

et

$$\tan(\theta_\phi + \theta_\varphi) = -\frac{Be^{i(\theta_\alpha + \theta_\omega + \theta_b)} - Ce^{i(\theta_\beta + \theta_\eta + \theta_c)}}{De^{i(\theta_\beta + \theta_\omega + \theta_d)} + Ae^{i(\theta_\alpha + \theta_\eta + \theta_a)}} \quad (\text{B.11})$$

Quand la matrice M est réelle, les conditions B.8, B.9, B.10 et B.11 se réduisent à :

$$\theta_\alpha = \theta_\beta = \theta_\gamma = \theta_\omega = 0, \quad (\text{B.12})$$

et

$$\tan(\theta_\phi - \theta_\varphi) = -\frac{B+C}{D-A}, \quad (\text{B.13})$$

$$\tan(\theta_\phi + \theta_\varphi) = -\frac{B-C}{D+A}, \quad (\text{B.14})$$

Ces résultats sont identiques à ceux des rotations de deux angles utilisés sur une matrice réelle B.5 et B.6.

Toutefois cette méthode est très lourde en utilisant les calculs arithmétiques traditionnels et il est difficile d'employer l'opérateur CORDIC. C'est la raison pour laquelle Hemkumar propose une structure où les transformations s'effectuent par l'opérateur CORDIC.

Dans cette structure, la décomposition SVD d'une matrice complexe est réalisée en deux étapes. La première étape utilise une transformation pour calculer la décomposition QR de la matrice M. La deuxième étape a pour but de diagonaliser la matrice.

Dans la première étape, deux sous-transformations sont employées sur les deux côtés de la matrice initiale. La première rend les éléments de la deuxième ligne de la matrice en réel. Elle est suivie par une rotation de Givens pour mettre l'élément de gauche à zéro. La transformation complète est définie par l'équation suivante :

$$\begin{bmatrix} c_\phi e^{i\theta_\alpha} & s_\phi e^{i\theta_\beta} \\ s_\phi e^{i\theta_\alpha} & c_\phi e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} c_\varphi e^{i\theta_\gamma} & s_\varphi e^{i\theta_\gamma} \\ s_\varphi e^{i\theta_\delta} & c_\varphi e^{i\theta_\delta} \end{bmatrix} = \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix} \quad (\text{B.15})$$

où les valeurs des angles sont données par les équations suivantes :

$$\theta_\alpha = \theta_\beta = \frac{-(\theta_d + \theta_c)}{2}, \quad (\text{B.16})$$

$$\theta_\gamma = -\theta_\delta = \frac{(\theta_d - \theta_c)}{2}, \quad (\text{B.17})$$

et

$$\theta_\pi = 0, \theta_\varphi = \tan^{-1} \frac{C}{D}. \quad (\text{B.18})$$

La deuxième étape est composée également de deux sous-transformations sur deux côtés de la matrice obtenue par la première étape. La première sous-transformation converge les éléments diagonaux en valeur réelle. La deuxième profite du fait que l'élément à gauche en bas de la matrice est nulle pour rendre l'élément à droit en haut en valeur réelle. Quand tous les éléments de la matrice sont nuls, la solution utilisée dans la décomposition SVD pour la matrice réelle peut être appliquée afin de diagonaliser la matrice. La deuxième transformation se décrit sous la forme suivante :

$$\begin{bmatrix} c_\lambda e^{i\theta_\xi} & s_\lambda e^{i\theta_\eta} \\ s_\lambda e^{i\theta_\xi} & c_\lambda e^{i\theta_\eta} \end{bmatrix} \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix} \begin{bmatrix} c_\rho e^{i\theta_\varsigma} & s_\rho e^{i\theta_\varsigma} \\ s_\rho e^{i\theta_\omega} & c_\rho e^{i\theta_\omega} \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix} \quad (\text{B.19})$$

Il existe deux possibilités de mettre les valeurs initiales des angles de la deuxième transformation. C'est dû au fait que les angles de gauche et de droite de la transformation sont interchangeables. Les choix sont entre :

$$\theta_\xi = \frac{-(\theta_x + \theta_w)}{2}, \theta_\eta = \frac{(\theta_x - \theta_w)}{2}, \quad (\text{B.20})$$

$$\theta_\varsigma = \frac{(\theta_x - \theta_w)}{2}, \theta_\omega = \frac{(\theta_x + \theta_w)}{2}, \quad (\text{B.21})$$

et

$$\theta_\xi = -\left(\frac{\theta_x}{2}\right), \theta_\eta = \left(\frac{\theta_x}{2}\right) 2, \quad (\text{B.22})$$

$$\theta_\varsigma = \left(\frac{\theta_x}{2}\right) - \theta_w, \theta_\omega = -\left(\frac{\theta_x}{2}\right), \quad (\text{B.23})$$

Les angles de rotation de la deuxième sous-transformation sont définis par les équations suivantes :

$$\tan(\theta_\lambda - \theta_\rho) = -\left(\frac{X}{Z - W}\right), \quad (\text{B.24})$$

$$\tan(\theta_\lambda + \theta_\rho) = -\left(\frac{X}{Z + W}\right), \quad (\text{B.25})$$

Nous proposons d'utiliser l'opérateur CORDIC en mode vecteur pour calculer les angles de rotation et en mode rotation pour effectuer les transformations. Les figures B.1 et B.2 illustrent les calculs d'un étage de la décomposition SVD d'une matrice complexe en utilisant l'opérateur CORDIC.

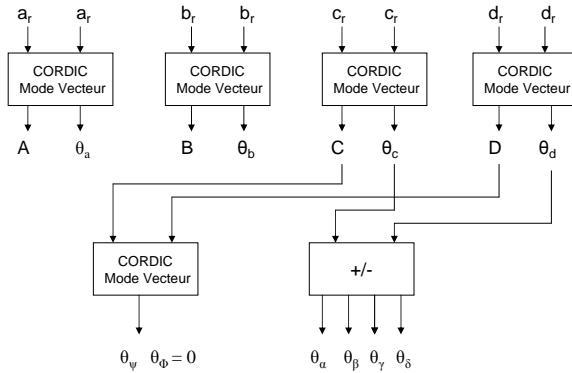


FIG. B.1 – SVD d'une matrice complexe - calculs des angles

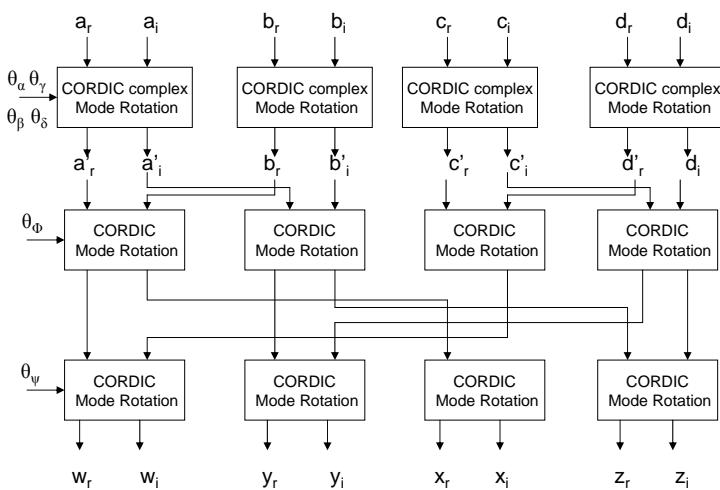


FIG. B.2 – SVD d'une matrice complexe - calculs des rotations

Pour profiter du maximum de parallélisme, Cavallaro [118] utilise une structure constituée de quatre opérateurs CORDIC. Dans le but d'augmenter la cadence de l'architecture, une structure pipeline peut être employée en ajoutant les registres entre les opérateurs CORDIC. Ces structures différentes peuvent être réalisées dans notre architecture reconfigurable CBDRA.

Pour les structures linéaire et systolique, l'algorithme SVD le plus efficace est l'algorithme Jacobi-like [119]. Brent, Luk et Vanloan [115] ont proposé une structure systolique pour implémenter le méthode Jacobi-SVD sur les matrices réelles. Cette structure est la plus efficace pour les calculs de l'algorithme Jacobi SVD.

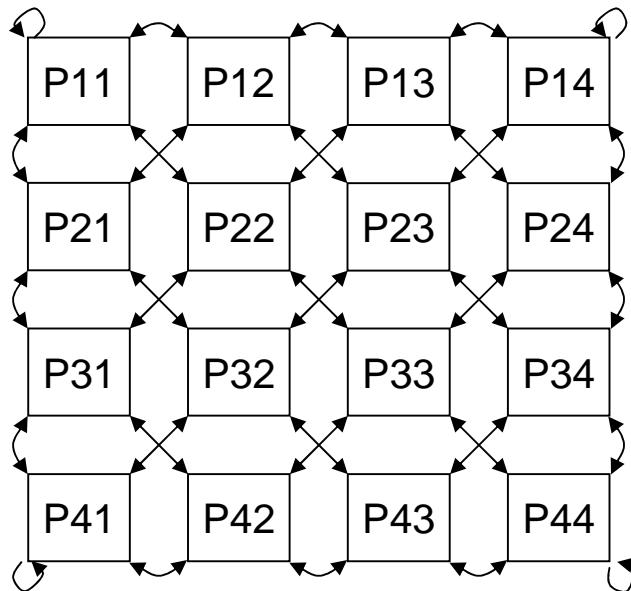


FIG. B.3 – structure systolique de l'algorithme Jacobi-SVD

Cette structure s'appuie sur un concept de réseau systolique, réseau de processeurs qui calculent et échangent des données régulièrement. Chaque processeur contient une sous-matrice 2×2 de la matrice M . La figure B.3 illustre un réseau de processeurs de 16 pour une matrice 8×8 . Les liens de communications pour échanger les données entre les processeurs sont montrés sur la figure B.4.

En général, chaque processeur est composé de quatre éléments réel :

$$\begin{bmatrix} \alpha_{ii} & \beta_{ii} \\ \gamma_{ii} & \delta_{ii} \end{bmatrix}$$

Dans chaque étape, deux rotations (c_i^L, s_i^L) et (c_i^R, s_i^R) sont effectuées sur deux côtés du processeur diagonal P_{ij} pour annuler les éléments β_{ii} et γ_{ii} . Ce sont les mêmes calculs que le SVD d'une matrice réel 2×2 présentés dans le paragraphe précédent. La même méthode sera utilisée pour effectuer ces rotations.

Ces calculs sont exprimés par les équations suivantes :

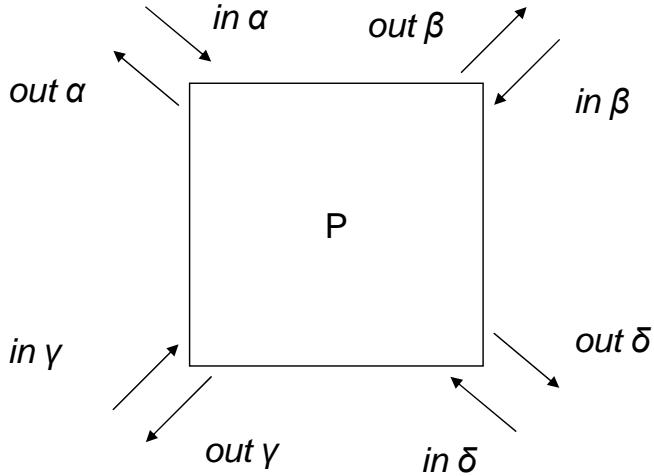


FIG. B.4 – liens de communications entre les processeurs

$$\begin{bmatrix} \alpha'_{ii} & 0 \\ 0 & \delta'_{ii} \end{bmatrix} = \begin{bmatrix} c_i^L & s_i^L \\ -s_i^L & c_i^L \end{bmatrix} \begin{bmatrix} \alpha_{ii} \beta_{ii} \\ \gamma_{ii} \delta_{ii} \end{bmatrix} \begin{bmatrix} c_i^R & s_i^R \\ -s_i^R & c_i^R \end{bmatrix} \quad (\text{B.26})$$

Les rotations sont également effectuées sur les processeurs non-diagonaux en appliquant les angles générés par les processeurs diagonaux. Les calculs se décrivent sous la forme suivante :

$$\begin{bmatrix} \alpha'_{ij} & \beta'_{ij} \\ \gamma_{ij} & \delta'_{ij} \end{bmatrix} = \begin{bmatrix} c_i^L & s_i^L \\ -s_i^L & c_i^L \end{bmatrix} \begin{bmatrix} \alpha_{ij} \beta_{ij} \\ \gamma_{ij} \delta_{ij} \end{bmatrix} \begin{bmatrix} c_j^R & s_j^R \\ -s_j^R & c_j^R \end{bmatrix} \quad (\text{B.27})$$

Les liens de communications (B.4) sont connectés comme présenté sur la figure B.3 afin de faciliter les échanges des données. Les interconnexions entre les processeurs sont spécifiées par les conditions indiquées dans la figure B.5.

La décomposition SVD est utilisée dans un nouvel algorithme sous optimal appelé HISD proposé par A.Nafkha dans sa thèse [120]. Cet algorithme a pour but résoudre de manière efficace le problème du décodage MIMO. Il repose sur une approche géométrique. Comparé aux algorithmes existants, le HISD possède trois caractéristiques très attrayantes pour une implantation dans des systèmes réels. La première concerne ses performances en terme de TEB qui sont proches de celles de ML. Sa faible complexité en terme de calcul et sa structure intrinsèque rendent d'autant plus aisée son implantation matérielle.

if $i = 1$ and $j = 1$ then $\begin{bmatrix} \text{out } \alpha \leftarrow \alpha; & \text{out } \beta \leftarrow \beta \\ \text{out } \gamma \leftarrow \gamma; & \text{out } \delta \leftarrow \delta \end{bmatrix}$
 else if $i = 1$ then $\begin{bmatrix} \text{out } \alpha \leftarrow \beta; & \text{out } \beta \leftarrow \alpha \\ \text{out } \gamma \leftarrow \delta; & \text{out } \delta \leftarrow \gamma \end{bmatrix}$
 else if $j = 1$ then $\begin{bmatrix} \text{out } \alpha \leftarrow \gamma; & \text{out } \beta \leftarrow \delta \\ \text{out } \gamma \leftarrow \alpha; & \text{out } \delta \leftarrow \beta \end{bmatrix}$
 else then $\begin{bmatrix} \text{out } \alpha \leftarrow \delta; & \text{out } \beta \leftarrow \gamma \\ \text{out } \gamma \leftarrow \beta; & \text{out } \delta \leftarrow \alpha \end{bmatrix}$
 {wait for outputs to propagate to inputs of adjacent processors}
 then $\begin{bmatrix} \text{in } \alpha \leftarrow \alpha; & \text{in } \beta \leftarrow \beta \\ \text{in } \gamma \leftarrow \gamma; & \text{in } \delta \leftarrow \delta \end{bmatrix}$

FIG. B.5 – Conditions d'échanges entre les processeurs

Annexe C

Smart Radio Challenge

Le SDR-Forum a organisé en 2007 un concours auprès d'universités du monde entier, le Smart Radio Challenge, concernant la radio logicielle. Le projet MENHIR (Multi-standard ENHanced Interoperable Radio) présenté par l'équipe SCEE a été retenu pour participer à la suite du concours. Il tente d'apporter une solution au scénario suivant : les secours(pompiers, police, ambulance) arrivent sur les lieux d'un incendie par exemple, et les infrastructures de communication ont été détruites ; chacun des services utilise un standard de communication différent des deux autres, le but du projet est de construire un terminal capable d'émettre et d'écouter sur les différents standards mis en jeux, de détecter et de choisir celui sur lequel il trouve un autre terminal avec qui communiquer. Nous avons donc développé un système de communication supportant la modulation et la démodulation AM et FM, et capable de se reconfigurer de l'un à l'autre pour chercher un autre terminal avec qui communiquer.

C.1 Description de la plate-forme de prototypage

Les cartes de prototypage utilisées à Supélec sont les solutions commercialisées par la société Sundance. Les réalisations effectuées dans le cadre de ce projet ont été portées sur la plate-forme "Software Defined Radio (SDR) Development Kit SMT8036". Cette plate-forme de Sundance se compose d'une carte mère compatible PCI sur laquelle sont insérés une carte fille DSP SMT365 et un module de conversion Analogique/Numérique SMT370.

La carte mère utilisée est la SMT310Q (figure C.1), elle permet d'accueillir 4 modules au format TIM (Texas Instrument Module).

Les principales interfaces de communications entre la carte mère et le Host (PC) sont le "*Global Bus*" sur 32 bits (un débit max. de 100 Mbytes/s @33MHz PCI clock) et le "*Host CommPort*" sur 8 bits d'un débit maximal théorique de 10 Mbytes/s.

La carte fille SMT365 est une carte DSP TMS320C6416T à virgule fixe cadencée à 1 GHz. L'architecture de cette carte fille est illustrée sur la figure C.2. Cette carte dispose de capacité mémoire de type ZBTRAM cadencé à 133 MHz d'une capacité de 256 Mbytes. Nous pouvons aussi remarquer que cette carte comporte un FPGA Xilinx Virtex-II XC2VP30-6 en package FF896. Sur cette carte le FPGA est dédié à fournir les interfaces de communications entre le DSP et le Host.

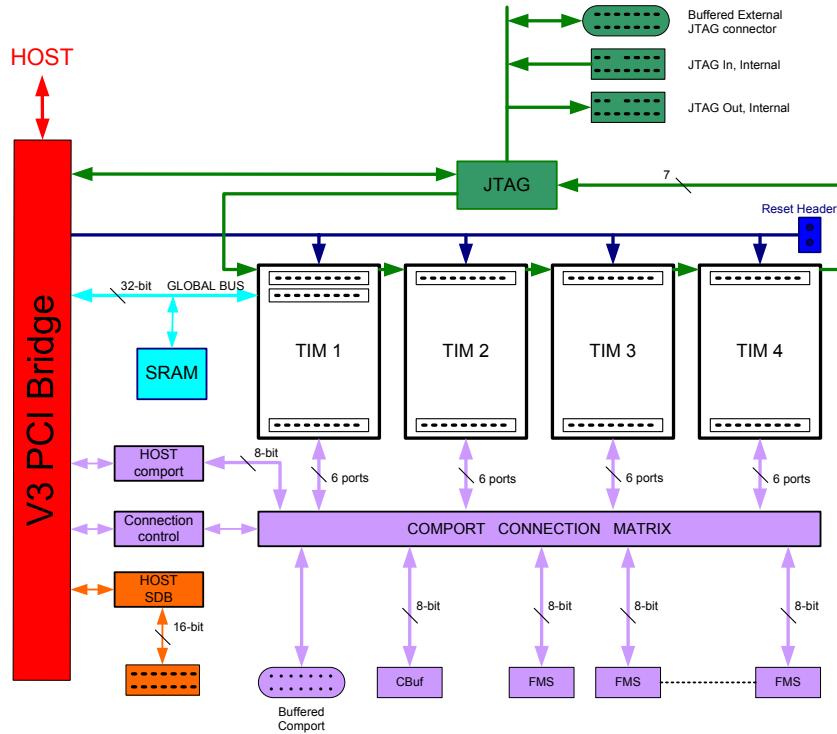


FIG. C.1 – Architecture de la carte mère (Source DS Sundance)

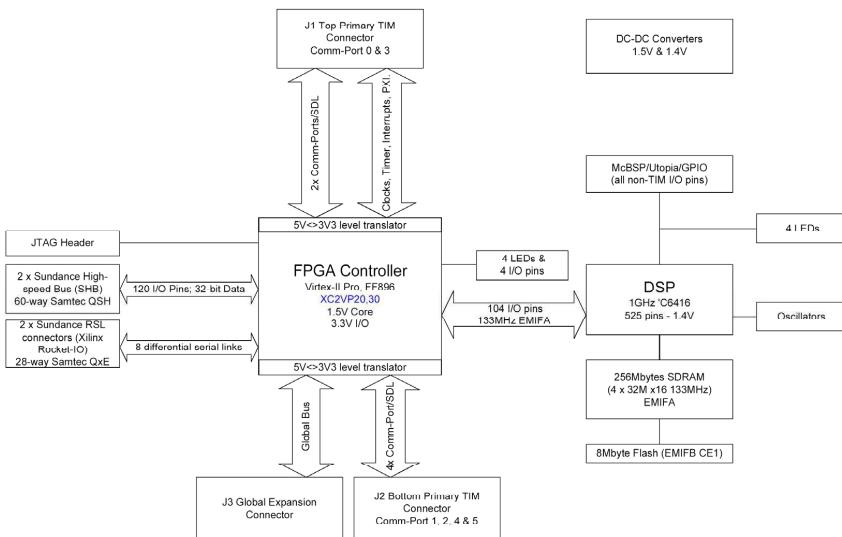


FIG. C.2 – Architecture du module DSP (Source DS Sundance)

C.2 Description du système

Nous avons utilisé une plate-forme produite par Sundance contenant essentiellement :

- un GPP,
- un DSP à virgule fixe,
- un FPGA Xilinx,
- un ADC et un DAC, avec un FPGA dédié à l'interface entre ces deux convertisseurs et le reste de la plate-forme.

La figure C.3 montre un schéma simplifié de la plate-forme :

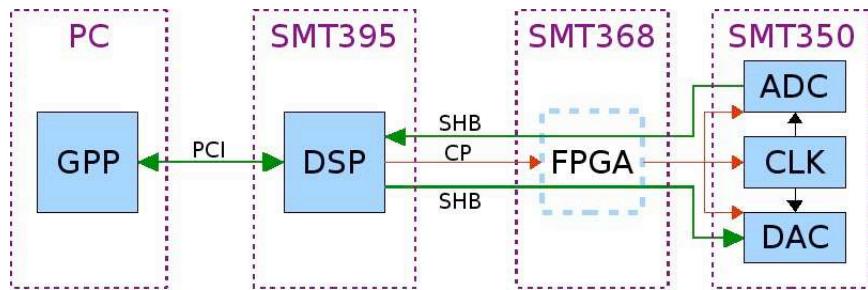


FIG. C.3 – Schéma simplifié de la plat-forme Sundance

les liens PCI et SHB sont les liens principaux, destinés aux communications des transmissions de données ;

les liens CP sont utilisés uniquement lors de la configuration des composants ; le FPGA de la carte SMT368 est le FPGA dédié à l'interface entre les convertisseurs et le DSP (comme indiqué plus haut), il n'est pas utilisable en temps que tel, d'où la représentation en pointillés, car il est plus ou moins transparent du point de vue du DSP ; notons la présence d'un composant horloge (CLK) sur la carte contenant les convertisseurs (SMT350) : il s'agit du circuit générant les horloges qui cadencent les convertisseurs, il a donc une importance particulière. Nous nous intéressons ensuite plus en détail à la carte SMT350 (contenant les convertisseurs), elle est l'élément le plus complexe du système complet mais la compréhension de son fonctionnement précis est cruciale dans des projets de radio logicielle.

C.3 Réalisation numérique de modulations analogique

Il s'agit de numériser les formules de modulation analogique, afin de simuler numériquement une modulation analogique.

C.3.1 Modulation AM

Comme son nom l'indique, le principe est que le signal à transmettre va moduler l'amplitude de la porteuse. Cela s'effectue simplement en multipliant le signal modulant et la porteuse, et la formule classique $y(t) = (A + x(t))\cos(\omega_c t)$ s'adapte très facilement au signal échantillonné :

$$y_n = (A + x_n) \cos(\omega_c n T_s).$$

ω_c étant la pulsation de la porteuse, T_s la période d'échantillonnage, et en notant $s_n = s(nT_s)$ pour tout signal s .

C.3.2 Démodulation AM

La démodulation s'effectue en multipliant à nouveau le signal modulé avec la porteuse. Pour être plus précis on calcule le signal en phase i en multipliant par la porteuse, et le signal en quadrature q en multipliant par la porteuse déphasée de $\pi/2$, car cela permet d'éliminer un inévitable déphasage entre l'émetteur et le récepteur. Notons qu'un filtre passe-bas est ensuite appliqué à ces deux signaux pour éliminer la composante à $2\omega_c$: D'où les formules échantillonnées :

$$\begin{aligned} i(t) &= y(t) \cos(\omega_c t + \phi_0) \frac{A + x(t)}{2} (\cos(2\omega_c t + \phi_0) + \cos(\phi_0)) \cong \frac{A + x(t)}{2} \cos(\phi_0) \\ q(t) &= y(t) \sin(\omega_c t + \phi_0) \frac{A + x(t)}{2} (\sin(2\omega_c t + \phi_0) + \sin(\phi_0)) \cong \frac{A + x(t)}{2} \sin(\phi_0). \end{aligned}$$

d'où les formules échantillonnées :

$$\begin{aligned} i_n &= y_n \cos(\omega_c n T_s + \phi_0) \\ q_n &= y_n \sin(\omega_c n T_s + \phi_0). \end{aligned}$$

Il ne reste alors plus qu'à estimer le module du complexe $i(t) + jq(t) = \frac{A+x(t)}{2} e^{j\phi_0}$, ce qui donne en numérique :

$$z_n = \sqrt{i_n^2 + q_n^2} = \frac{A + x(t)}{2}.$$

C.3.3 Modulation FM

Cette fois le signal à transmettre va modifier la fréquence de la porteuse (ou la pulsation, cela revient au même) selon la formule : $\omega(t) = \omega_c + \omega_d x(t)$ (ω_d étant la pulsation de déviation par rapport à la porteuse). Le signal modulé est donc donné par la formule suivante :

$$y(t) = A \cos\left(\int_0^t \omega(\tau) d\tau\right) = A \cos\left(\omega_c t + \omega_d \int_0^t x(\tau) d\tau\right).$$

La première formule échantillonnée qui vient à l'esprit est la suivante :

$$y_n = A \cos\left(\omega_c n + \omega_d \sum_{k=0}^{n-1} T_s\right).$$

Cela revient en fait à utiliser la méthode "des rectangles" pour le calcul de l'intégrale, et il pourra sans doute être avantageux d'utiliser une méthode plus précise comme la méthode de Simpson ou celle de Gauss-Legendre. Nous gardons cependant cette formule dans un premier temps.

C.3.4 Démodulation FM

La méthode que nous présentons ici s'appelle le Slope algorithm. On commence par calculer les signaux en phase et en quadrature exactement de la même manière que dans la démodulation AM (en appliquant à nouveau un filtre passe-bas une fois la multiplication avec la porteuse effectuée) :

$$\begin{aligned} i(t) &= y(t)\cos(\omega_c t + \phi_0) \cong \frac{A}{2}\cos(\phi(t)) \\ q(t) &= y(t)\sin(\omega_c t + \phi_0) \cong \frac{A}{2}\sin(\phi(t)). \end{aligned}$$

en notant $\phi(t) = \omega_d \int_0^t x(\tau)d\tau - \phi_0$.

On multiplie alors les signaux entre eux, en introduisant un retard d et on calcule :

$$z(t) = i(t)q(t-d) - i(t-d)q(t) = \frac{A^2}{2}\sin(\omega_d \int_{t-d}^t x(\tau)d\tau) \cong \frac{A^2}{2}\sin(\omega_d \int_{t-d}^t x(\tau)d\tau).$$

Si $\omega_d \int_{t-d}^t x(\tau)d\tau$ est suffisamment petit.

Pour calculer les formules échantillonées correspondantes on choisit un retard $d = T_s$, ce qui simplifie les formules :

$$\begin{aligned} i_n &= y_n \cos(\omega_c n T_s + \phi_0) \\ q_n &= y_n \sin(\omega_c n T_s + \phi_0) z_n = i_n q_{n-1} - i_{n-1} q_n \\ &= \frac{A^2}{2} \sin(\omega_d x_{n-1} T_s) \\ &\cong \frac{A^2}{2} \omega_d x_{n-1} T_s \end{aligned}$$

C.4 Chaînes d'émission et de réception modulaires

L'étude des différentes chaînes de modulation nous a permis d'identifier les modules de bases qui reviennent systématiquement et à partir desquels on peut construire toutes les chaînes de modulation. L'idée est très connue dans le génie logiciel et consiste à découper le système en composants réutilisables. La radio logicielle tentant d'implémenter la plupart des traitements sur le signal de manière logicielle, il est normal que ces techniques de programmation soient reprises. Bien entendu, plus le code utilisé est générique, moins il sera performant, il ne faut donc pas perdre de vue le fait qu'un système de modulation ou de démodulation doit fonctionner en temps réel. Il s'agit de trouver un bon compromis entre réutilisabilité du code et performance. Les architectures des modulations et démodulations AM et FM seront présentées dans ce paragraphe.

C.4.1 Modulation AM

- la boîte MIC représente le microphone, c'est-à-dire l'acquisition du son ;
- l'ensemble PFR+IIR réalise le suréchantillonnage du signal, pour l'adapter à la fréquence d'échantillonnage du convertisseur numérique-analogique : il s'agit essentiellement d'intercaler des zéro entre les échantillons et d'appliquer un filtre passe-bas ;

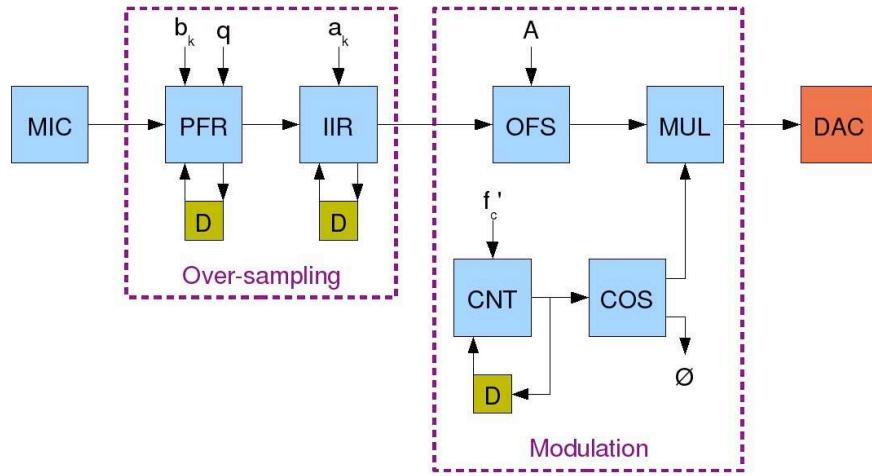


FIG. C.4 – Diagramme de la modulation AM

- l'ensemble CNT+COS génère un cosinus à la fréquence porteuse (la sortie non utilisée est le sinus correspondant) ;
- la boîte MUL multiplie cette porteuse avec le signal suréchantilloné auquel on a ajouté un offset (boîte OFS) ;
- le signal est enfin envoyé au convertisseur numérique-analogique : DAC.

Notons que les boîtes D sont des boîtes spéciales retardant simplement le signal qu'elles ont en entrée : elles sont utilisées par les modules ayant besoin de mémoriser certains données.

C.4.2 Démodulation AM

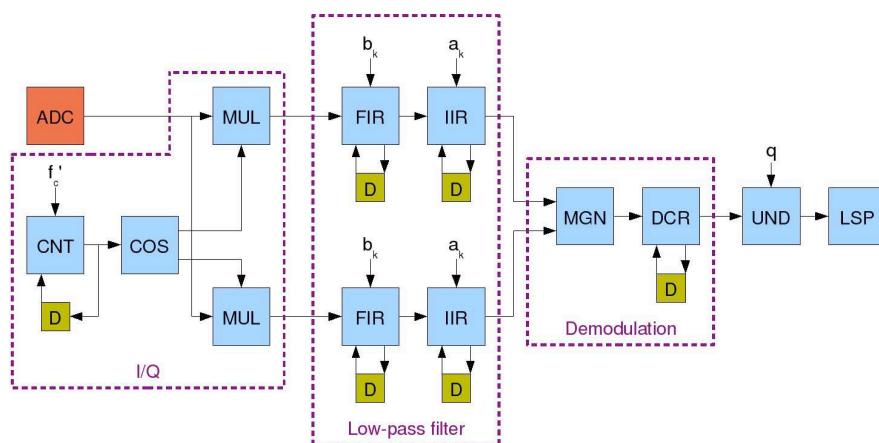


FIG. C.5 – Diagramme de la démodulation AM

- le signal arrive en provenance du convertisseur analogique-numérique : ADC ;
- l'ensemble "I/Q", constitué des boîtes CNT, COS et MUL déjà vues, réalise la séparation des parties en phase et en quadrature du signal ;
- l'ensemble FIR+IIR constitue un filtrage passe-bas (la même chose que pour le suréchantillonnage, mais sans l'intercalage de zéros) ;
- le démodulation effective est réalisée au niveau des boîtes MGN (détection d'amplitude) et DCR (suppression de la composante continue) ;
- le signal est ensuite sous-échantillonné (UND) pour revenir au taux d'échantillonnage initial, et envoyé au haut-parleur LSP.

C.4.3 Modulation FM

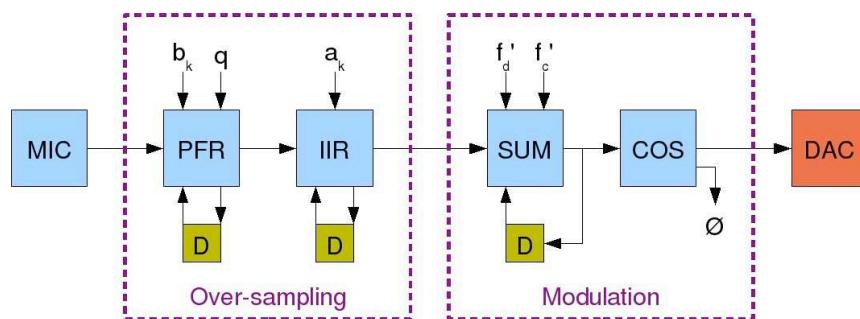


FIG. C.6 – Diagramme de la modulation FM

- une fois acquis au niveau du micro MIC, le signal est suréchantillonné comme pour la modulation d'amplitude (PFR+IIR) ;
- au lieu de le multiplier à une porteuse, le signal sert à générer la phase de la porteuse ;
- le signal modulé est enfin envoyé au convertisseur numérique-analogique : DAC.

Notons que les boîtes D sont des boîtes spéciales retardant simplement le signal qu'elle ont en entrée : elles sont utilisées par les modules ayant besoin de mémoriser certains données.

C.4.4 Démodulation FM

- le signal arrive en provenance du convertisseur analogique-numérique : ADC ;
- l'ensemble "I/Q", constitué des boîtes CNT, COS et MUL déjà vues, réalise la séparation des parties en phase et en quadrature du signal ;
- l'ensemble FIR+IIR constitue un filtrage passe-bas (la même chose que pour le suréchantillonnage, mais sans l'intercalage de zéros) ;
- le démodulation effective est réalisée au niveau des boîtes MGN (détection d'amplitude) et DCR (suppression de la composante continue) ;
- le signal est ensuite sous-échantillonné (UND) pour revenir au taux d'échantillonnage initial, et envoyé au haut-parleur LSP.

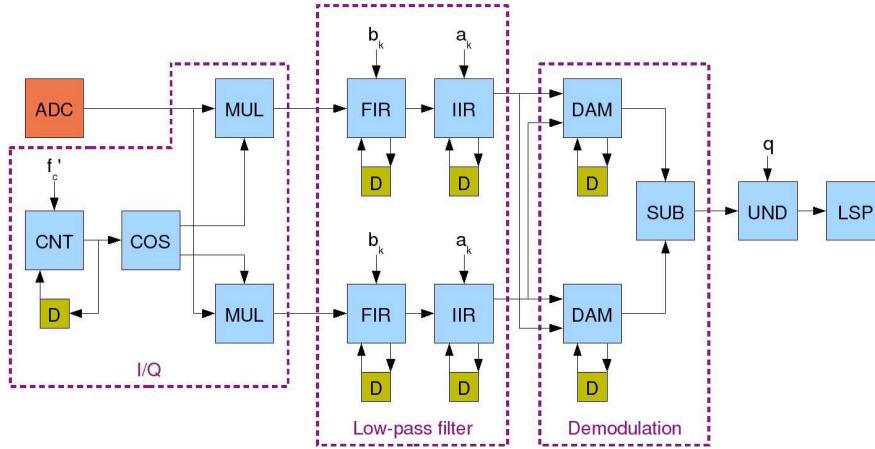


FIG. C.7 – Diagramme de la démodulation FM

C.5 Intégration à l'outil SynDEX

Nous utilisons l'outil SynDEX développé par l'INRIA qui permet de développer facilement des chaînes de traitements sur la plate-forme hétérogène. Il suffit de développer chacun des modules de traitement séparément, de dessiner la chaîne globale sur l'interface graphique du logiciel et SynDEX se charge de répartir les différents modules sur les différents processeurs et de gérer les communications entre les différents modules. C'est un exemple d'outil permettant de faire abstraction de la complexité du programme global et du système distribué. Il offre la possibilité de développer et tester rapidement de nouveaux programmes de radio logicielle.

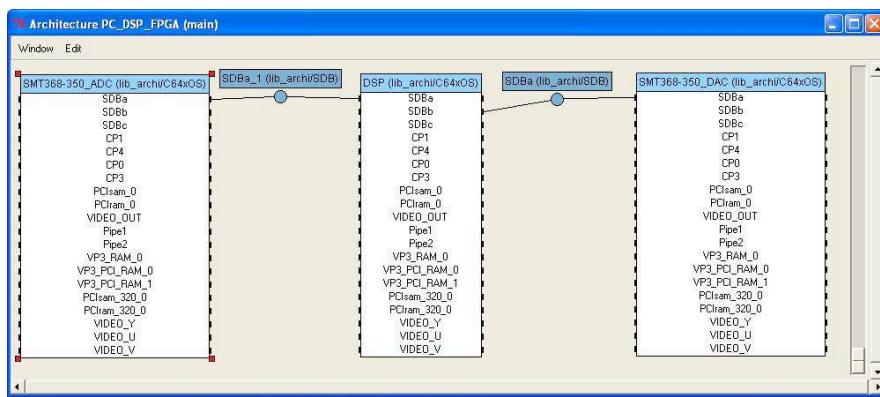


FIG. C.8 – Exemple d'architecture décrite sous SynDEX

La figure C.8 montre une architecture constituée d'un convertisseur analogique-numérique (à gauche), d'un DSP (au milieu) et d'un convertisseur numérique-analogique (à droite) telle qu'elle peut être décrite sous SynDEx.

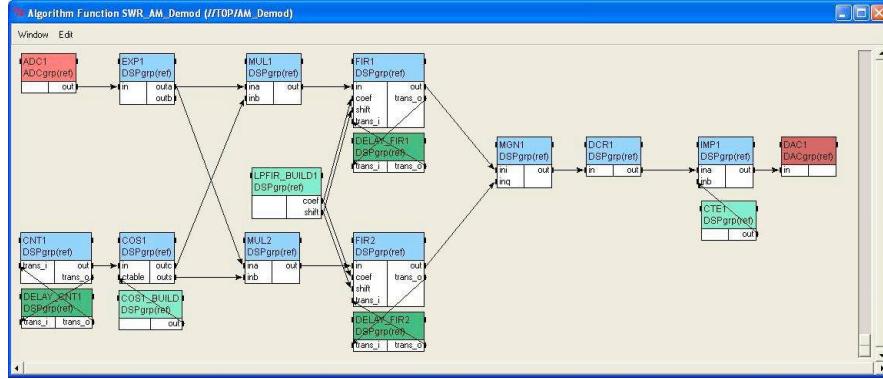


FIG. C.9 – Exemple d'algorithme (démodulation AM) décrite sous SynDEx

La figure C.9 montre comment on peut décrire un algorithme de démodulation AM (modulaire) sous SynDEx.

Une fois l'architecture matérielle et l'algorithme décrit sous SynDEx, celui-ci gère lui-même la répartition des modules sur les différents processeurs ainsi que la synchronisation des communications entre les différents modules. Il génère alors un macro-code décrivant le programme dans le langage de macro m4 et on peut utiliser l'outil GNU m4 pour produire du code C correspondant à la description de l'algorithme.

C.6 Utilisation de la reconfiguration dynamique

Pour bénéficier des capacités de la reconfiguration partielle de FPGA, nous avons réalisé certains modules de la modulation AM et FM dans le FPGA. L'architecture reconfigurable du système est illustrée sur la figure C.10.

Les traitements de deux standards avec les modulations AM et FM respectivement peuvent être synthétisés dans la partie reconfigurable du FPGA. Ces modules partagent les mêmes ressources matérielles du FPGA. Les fichiers de configuration (Bitstreams) de ces modules sont stockés dans la mémoire du GPP. Quand un standard est demandé, le module correspondant peut être chargé dans le FPGA. L'architecture FPGA s'établit autour d'un processeur embarqué (MicroBlaze). Il est aussi chargé de contrôler les paramètres des modules et de transmettre des données entre le GPP et le FPGA. Dans ce projet, nous avons réalisé quelques modules des modulations AM et FM dans le FPGA afin d'illustrer les avantages de l'utilisation de la reconfiguration partielle, par exemple le filtre FIR.

Il faut noter que l'outil SynDEx ne permet pas de générer les codes VHDL, mais nous pouvons toujours l'utiliser pour gérer la répartition des modules entre le FPGA et les processeurs tels que GPP et DSP à condition que les codes VHDL des modules soient générés séparément et que les interfaces des modules soient identiques à celles des GPP et DSP.

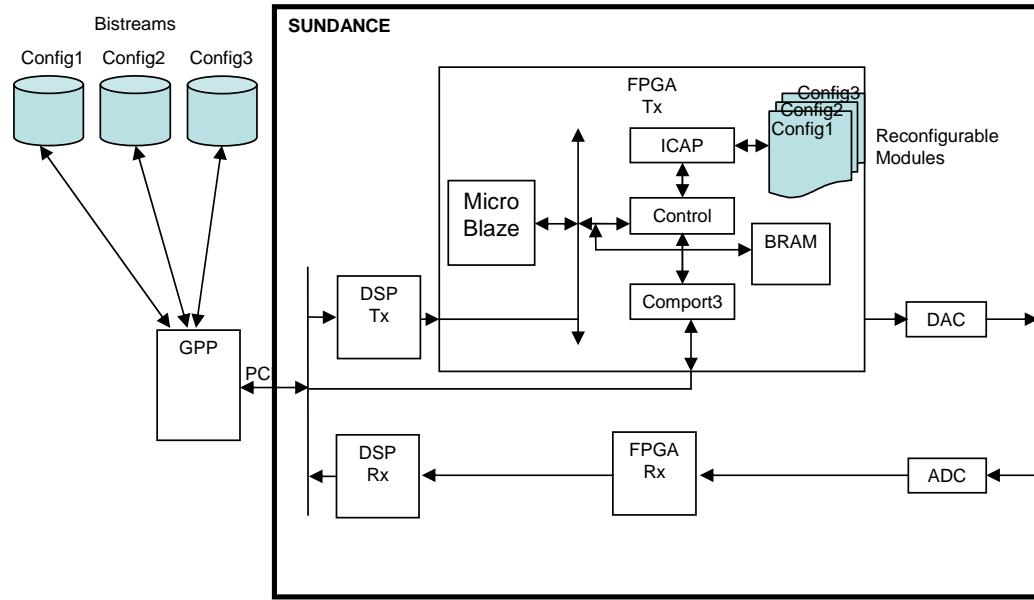


FIG. C.10 – Architecture reconfigurable du système

La figure C.11 montre le poster présenté dans le concours Smart Radio Challenge de SDR Forum en 2007 à Denver aux USA. L'équipe MENHIR a obtenu le second prix dans le même type de projet.

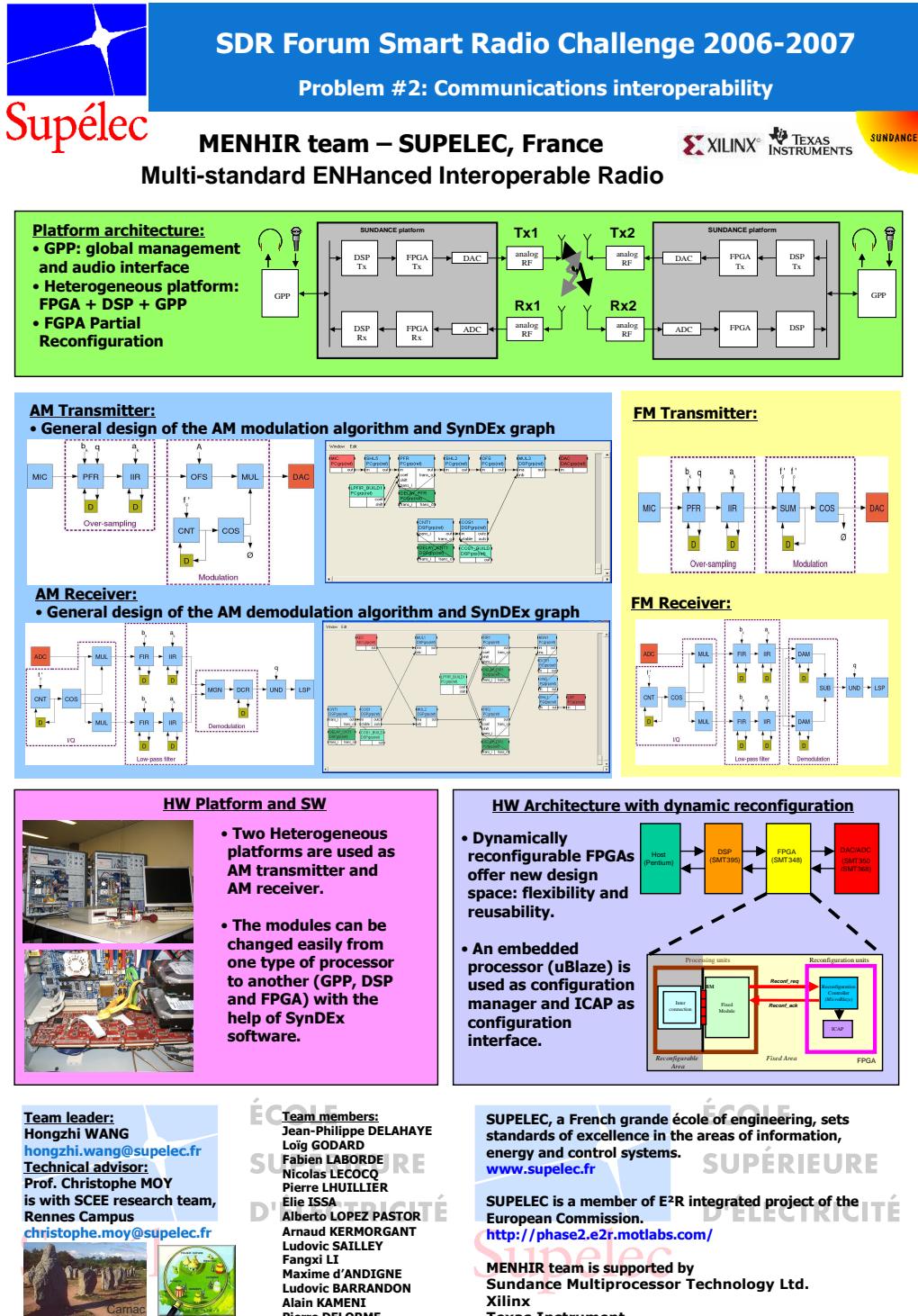


FIG. C.11 – Poster présenté au SDR Forum 2007

Annexe D

Simulation Modelsim des modules de l'algorithme SRA

Nous avons simulé les modules de l'algorithme V-BLAST Square Root pour vérifier le bon fonctionnement de l'architecture

Le fonctionnement de l'ensemble de l'architecture est illustré par les chronogrammes présenté dans les figures D.1 et D.2. Les données sont calculés issue des valeurs des coefficients du canal montrées dans l'exemple suivant :

$$\begin{bmatrix} hr11 + hi11 & hr12 + hi12 \\ hr21 + hi21 & hr22 + hi22 \end{bmatrix} \times 2^7 = \begin{bmatrix} -16 - 88i & 16 - 26i \\ 12 + 44i & -6 + 44i \end{bmatrix} \quad (\text{D.1})$$

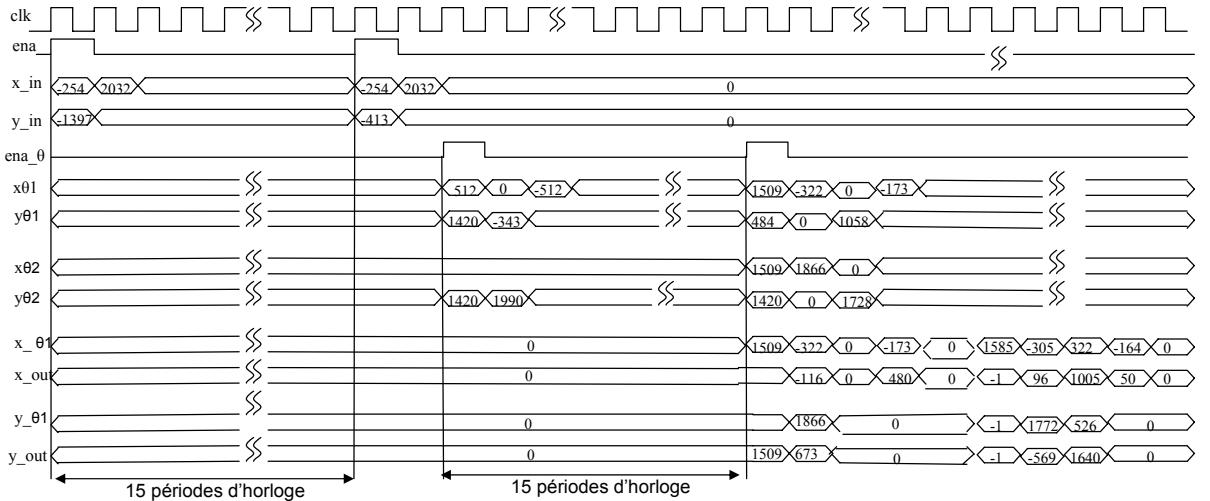


FIG. D.1 – Chronogramme de la première itération du module de la décomposition QR

La figure D.3 montre le chronogramme de fonctionnement du processeur PE1.

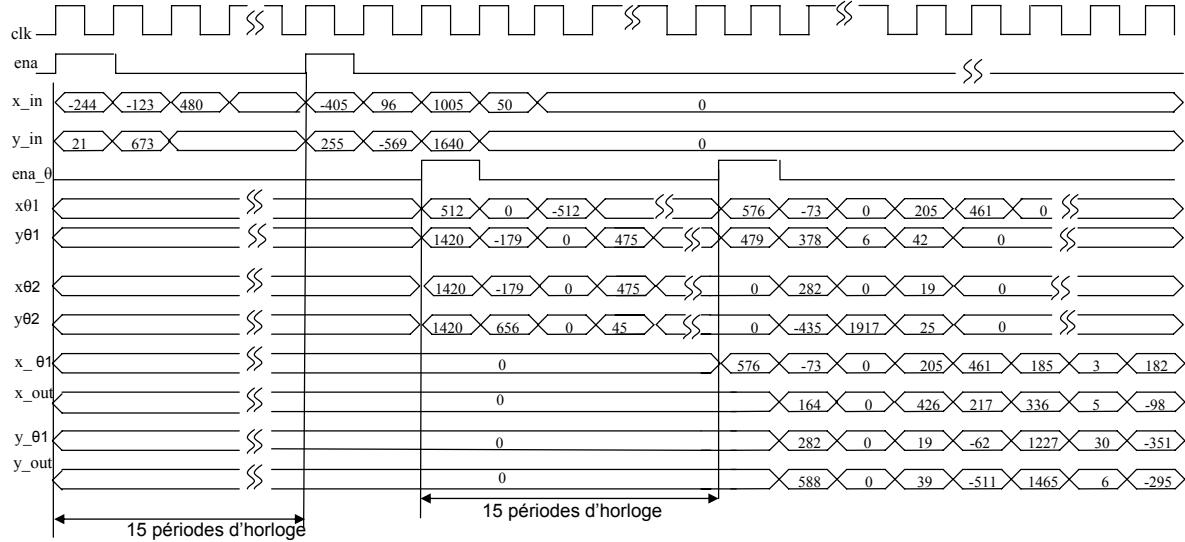


FIG. D.2 – Chronogramme de la deuxième itération du module de la décomposition QR

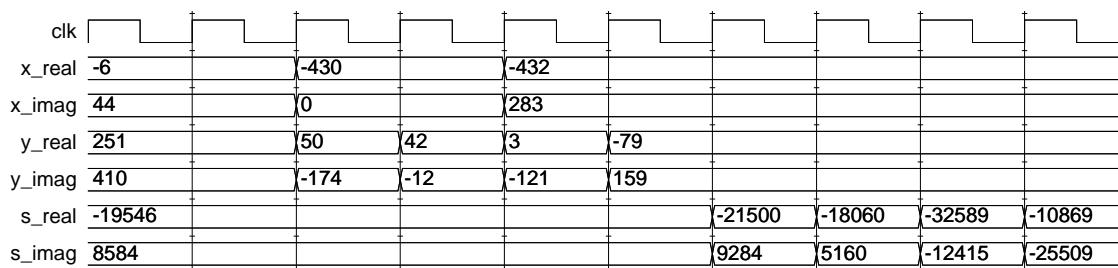


FIG. D.3 – Chronogramme des calculs du processeur PE1

Les figures D.4 et D.5 présentent les chronogrammes des données en entrée et en sortie du décodeur. Comme le montre la figure D.4, le signal de début de traitement *ena* permet de valider la mémorisation des paramètres en entrée du décodeur.

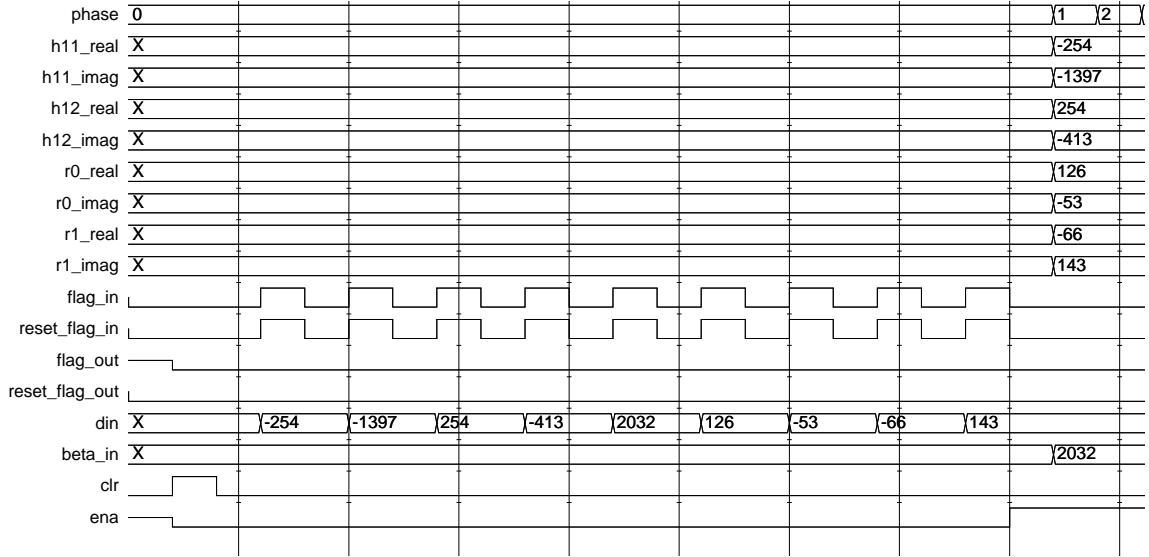


FIG. D.4 – Chronogramme des signaux en entrée du décodeur V-BLAST

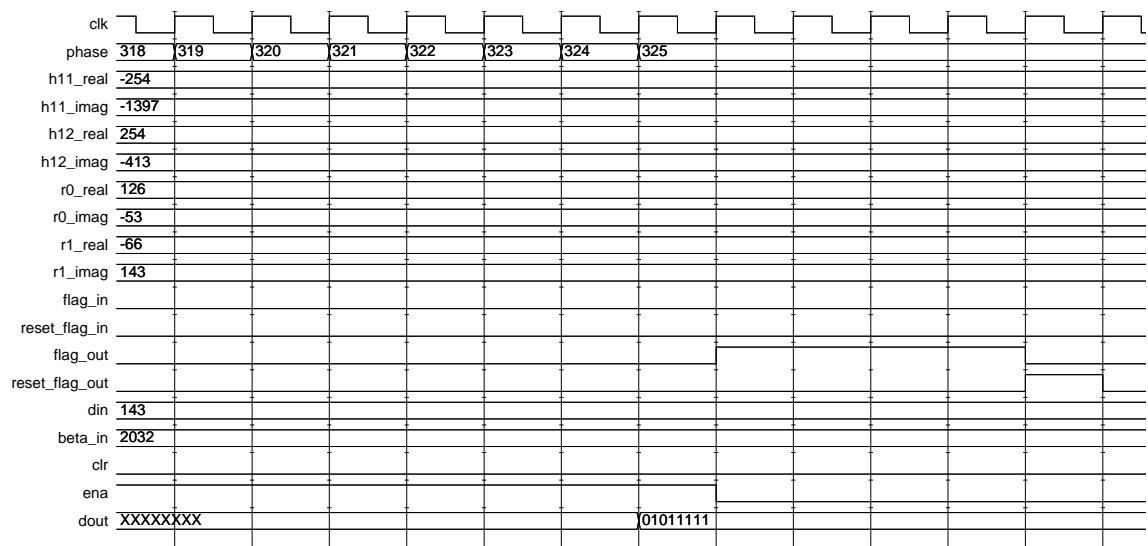


FIG. D.5 – Chronogramme des signaux en sortie du décodeur V-BLAST

Annexe E

Utilisation de l'opérateur CORDIC dans les modules M_4 , M_5 , M_6 de l'algorithme SRA

Nous avons étendu l'utilisation de l'opérateur CORDIC dans les modules M_4 , M_5 , M_6 de l'algorithme V-BLAST Square Root. Nous avons implémenté ces trois modules en utilisant les PEs dans le chapitre 4.

E.0.1 Le module de l'ordonnancement

Nous proposons une structure à base d'opérateurs CORDIC pour effectuer les mêmes calculs. En effet, le multiplicateur est utilisé pour calculer le module du vecteur complexe. Comme il est présenté dans le paragraphe 2, le module et la phase peuvent être calculés à l'aide de l'opérateur CORDIC en mode vecteur. Au total, six multiplications et quatre additionneurs sont remplacés par quatre opérations CORDIC. Cette structure est illustrée par la figure E.1.

E.1 Le module de calcul des vecteurs "nulling"

Afin d'utiliser l'opérateur CORDIC dans ce module, nous réformons les équations effectuées le calcul des vecteur "nulling". L'équation 4.2 et 4.3 peuvent être aussi exprimée sous forme suivante :

$$w_1 = M(P_{11}^{1/2}) [(cos\beta_1 \times \Re(Q_{11}) + sin\beta_1 \times \Im(Q_{11})) + j(sin\beta_1 \times \Re(Q_{11}) - cos\beta_1 \times \Im(Q_{11}))] \quad (\text{E.1})$$

$$w_2 = M(P_{11}^{1/2}) [(cos\beta_1 \times \Re(Q_{21}) + sin\beta_1 \times \Im(Q_{21})) + j(sin\beta_1 \times \Re(Q_{21}) - cos\beta_1 \times \Im(Q_{21}))] \quad (\text{E.2})$$

$$\text{où } \beta_1 = \tan^{-1} \frac{\Im(P_{11}^{1/2})}{\Re(P_{11}^{1/2})}.$$

Avec les équations E.4 et E.2, nous pouvons constater que ces calculs peuvent être effectués par une structure à base d'opérateur CORDIC. Elle est composée de trois opérateurs CORDIC, dont un en mode vecteur et deux en mode rotation. La figure E.2 illustre

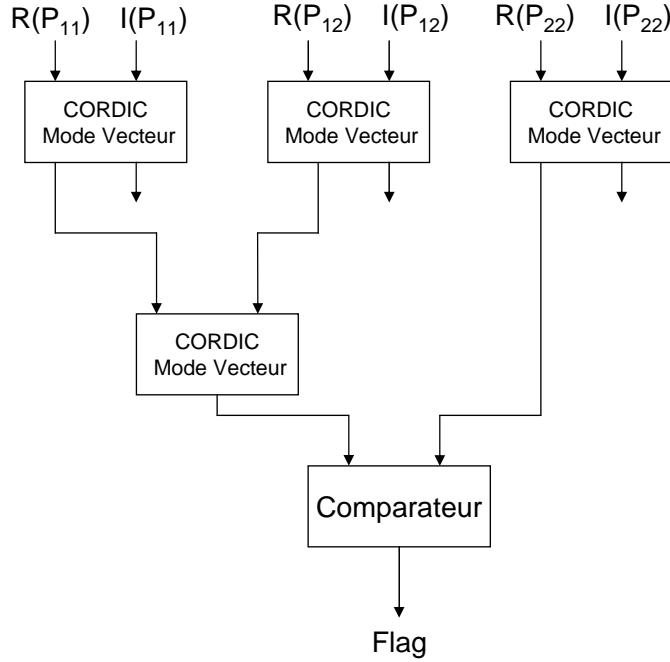


FIG. E.1 – Structure de calcul et de sélection à base de CORDIC

le schéma de principe de ces calculs. Elle peut remplacer quatre multiplications et quatre additions/soustractions utilisées dans la structure précédente. Il faut noter que l'opération en mode vecteur peut être évitée si aucun ordonnancement est effectué dans les calculs précédents.

Puisque les valeurs des vecteurs "nulling" w'_1 et w'_2 sont multipliés par le même facteur $M(P_{11}^{1/2})$, ces quatre multiplications peuvent être évitées dans le cas d'une modulation QPSK. Les résultats obtenus par la structure CORDIC se décrivent dans les équations suivantes :

$$w'_1 = [(\cos\beta_1 \times \Re(Q_{11}) + \sin\beta_1 \times \Im(Q_{11})) + j(\sin\beta_1 \times \Re(Q_{11}) - \cos\beta_1 \times \Im(Q_{11}))] \quad (\text{E.3})$$

$$w'_2 = [(\cos\beta_1 \times \Re(Q_{21}) + \sin\beta_1 \times \Im(Q_{21})) + j(\sin\beta_1 \times \Re(Q_{21}) - \cos\beta_1 \times \Im(Q_{21}))] \quad (\text{E.4})$$

Les autres vecteurs seront calculés de la même façon. A l'issue du traitement lié à l'ordonnancement, le signal généré par le contrôleur valide les valeurs des vecteurs "nulling" en sortie.

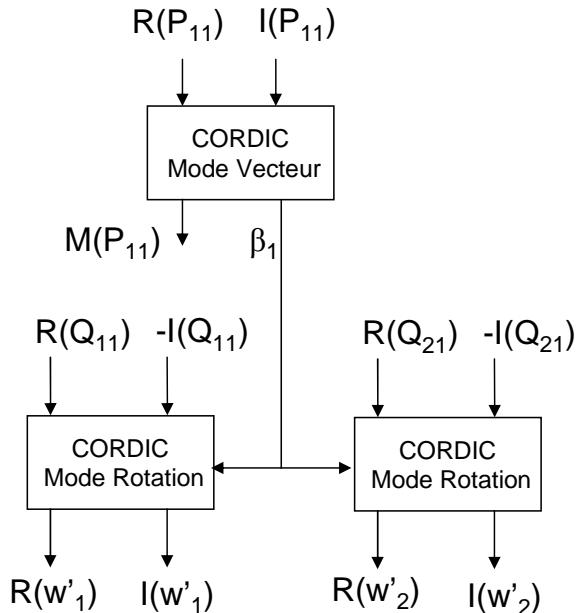


FIG. E.2 – Calculs des vecteur "nulling" par l'opérateur CORDIC

E.2 Le module de décodage des données

Nous proposons une structure à base d'opérateurs CORDIC pour effectuer les décodages. Dans le cas de la démodulation QPSK, il suffit de trouver le signe de la partie réelle et celui de la partie imaginaire et ignorer le module du vecteur complexe. L'opération CORDIC en mode rotation peut effectuer une rotation sans multiplication du module. L'équation s'écrit sous forme suivante en utilisant l'opérateur CORDIC :

$$\Re(\tilde{y}_1) + j\Im(\tilde{y}_1) = \begin{bmatrix} \Re(w_1) + j\Im(w_1) & \Re(w_2) + j\Im(w_2) \end{bmatrix} \times \begin{bmatrix} Y_1(\cos\theta_1 + j\sin\theta_1) \\ Y_2(\cos\theta_2 + j\sin\theta_2) \end{bmatrix} \quad (\text{E.5})$$

$$\text{où } \theta_1 = \tan^{-1} \frac{\Im(y_1)}{\Re(y_1)}, \theta_2 = \tan^{-1} \frac{\Im(y_2)}{\Re(y_2)}.$$

Les deux multiplications complexes, c'est à dire huit multiplications sont remplacées par quatre opérations CORDIC dont deux en mode vecteur et deux en mode rotation, ainsi que un diviseur et deux multiplicateurs. Le schéma de la figure E.3 illustre le principe de cette opération.

Le produit de l'opérateur CORDIC donne un nombre scalaire sur lequel est effectuée une décision. Le résultat de cette décision détermine alors le symbole complexe estimé s_1 . Le symbole est qualifié de quatre bit, dont deux pour la partie réel et deux pour la partie imaginaire.

Deux opérations CORDIC sont effectuées pour le produit vectoriel par le processeur PE2 à partir du second vecteur de l'égaliseur et du nouveau signal y obtenu en sortie du

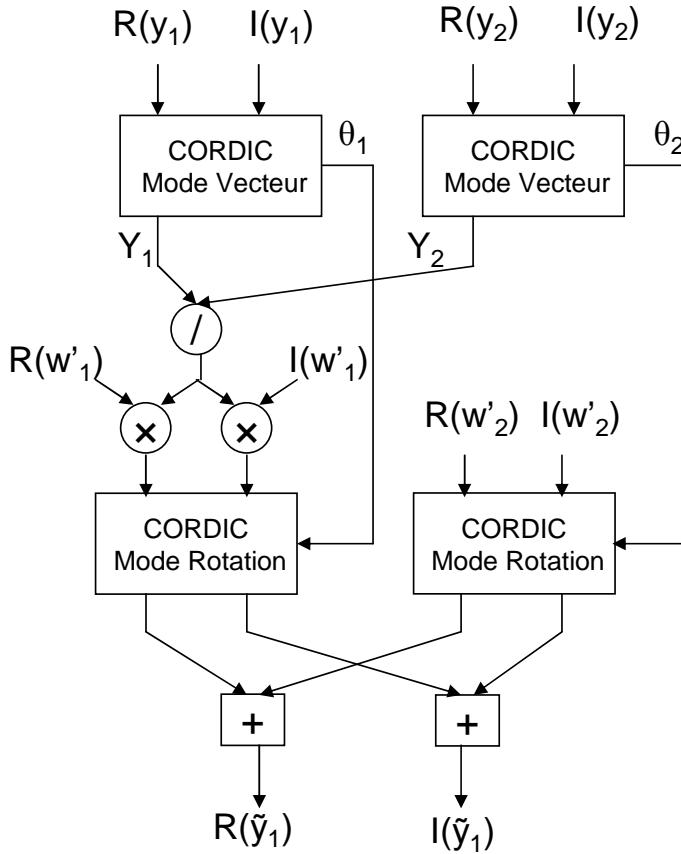


FIG. E.3 – Calculs des symboles estimés par l'opérateur CORDIC

processeur PE1. Comme précédemment, le second symbole estimé s_2 est déterminé par le signe du produit.

Le tableau E.1 compare en résumé les nombres de différents opérateurs utilisés entre la méthode classique et la méthode utilisant CORDIC.

Les nombres d'opération représentés dans le tableau E.1 sont les nombres totaux d'opération effectués dans le cas de MIMO 2×2 . Ce tableau résume les trois modules où les mutiplieurs sont remplacés par les opérateurs CORDIC. Il s'agit du calcul de la norme minimale dans le module de l'ordonnancement, des vecteurs "nulling" et du décodage des données. Comme l'illustre le tableau, 90% des multiplieurs sont remplacés par l'opérateurs CORDIC. Ceci permet d'avoir plus de flexibilité vis à vis de l'utilisation de l'opérateur dans l'ensemble de modules. Il à noter que l'utilisation de l'additionneur/soustracteur est aussi fortement diminuée (75%) en utilisant l'opérateur CORDIC.

TAB. E.1 – Résumé des calculs remplacés par l'opérateur CORDIC

Méthode	Type d'opération	Norme minimale	Vecteur "nulling"	Décodage	Total
Classique	Multiplication	6	16	16	38
	Addition/soustraction	4	8	8	20
CORDIC	Opération CORDIC	4	6	8	18
	Addition/soustraction			4	4
	Multiplication			4	4
	Dévision			2	2

Notations mathématiques

Variables utilisées

v	Bruit additif Gaussien
$Débit_{symbole}$	Débit symboles binaire
H	Matrice de canal MIMO de dimension $M \times N$
i	indice d'itération
I	Matrice identité
M	Nombre d'antennes d'émission
N	Nombre d'antennes de réception
P	Matrice de covariance de l'erreur estimé
Q	Matrice unitaire
R	Matrice triangulaire supérieur
\hat{s}	Vecteur des symboles estimé
σ^2	Variance du bruit blanc additif gaussien
Σ	Transformation unitaire

Notations mathématiques

x	Scalaire
x^*	Conjugué de x
A^+	Pseudo inverse de la matrice A
\mathbf{x}^T	Vecteur transposé
$E\{\mathbf{x}\}$	Espérance mathématique de la variable aléatoire \mathbf{x}
$ \mathbf{v} $	Module du vecteur \mathbf{v}
$\ \mathbf{v}\ $	Norme1 du vecteur \mathbf{v}

Table des figures

1.1	Architecture superhétérodyne	6
1.2	Architecture de radio logicielle idéale	7
1.3	Architecture de radio logicielle restreinte	7
2.1	Principe d'un système MIMO	14
2.2	Diagramme de treillis pour un STTC à 4 états utilisant $M = 2$ émetteurs et une modulation MDP-4	16
2.3	Architecture d'un transmetteur D-BLAST	18
2.4	Architecture d'un transmetteur V-BLAST	18
2.5	Principe de l'architecture de modulation V-BLAST	19
2.6	Architecture d'un transmetteur H-BLAST	20
2.7	Principe de l'architecture de modulation H-BLAST	21
2.8	Comparaison des performances en TEB	25
2.9	Comparaison de la complexité en fonction de nombre d'antennes	26
2.10	Architecture fonctionnelle du décodeur MIMO "V-Blast square-root"	36
3.1	Rotation du vecteur V dans le plan cartésien	41
3.2	CORDIC en mode vecteur	43
3.3	CORDIC en mode rotation	45
3.4	Architecture à opération série et itération série	46
3.5	Architecture à opération parallèle et itération série	47
3.6	Architecture à opération parallèle et itération parallèle	48
3.7	Architecture de CBDRA	52
3.8	Gestion de configuration hiérachique	53
3.9	Elément configurable de base des FPGA	53
3.10	Configuration d'un CLB	54
3.11	Illustration d'un Slice	55
3.12	Différents types de reconfiguration	56
3.13	Représentation temporelle de la reconfiguration	56
3.14	Flot de conception modulaire sur FPGA	57
3.15	Flot de conception reconfigurable sur FPGA	58
3.16	LUT based <i>Bus Macro</i> sur FPGA Virtex-II	59
3.17	Architecture CBDRA avec la reconfiguration statique	60
3.18	Architecture CBDRA avec la reconfiguration dynamique	62
3.19	Chronogramme du flot de données	63
3.20	Structure parallèle des CORDIC pour calculer $P^{1/2}$ et Q_α	64

3.21	Relation entre le débit symbole et le nombre d'opérateurs CORDIC utilisés	67
3.22	Organisation des différentes structures à base de 5 CORDIC	68
3.23	utilisation itérative de 5 CORDIC	69
3.24	Organisation des différentes structures à base de 3 CORDIC	69
3.25	utilisation itérative de 3 CORDIC	70
3.26	Utilisaiton de la reconfiguration dynamique pour calculer $P^{1/2}$ et Q_α	71
3.27	Séquencement de reconfigurations partielles	71
3.28	Architecture des modules M_4 , M_5 , M_6	72
3.29	Architecure de l'ensemble du décodeur MIMO V-BLAST Square Root	73
3.30	Flot de gestion de l'architecture et de la reconfiguration	73
3.31	Le TEB en fonction du SNR.	74
3.32	Le TEB en fonction du nombre d'étages de l'opérateur CORDIC	75
3.33	Le TEB en fonction du SNR.	76
3.34	Le EQM en fonction du nombre d'étages de l'opérateur CORDIC.	76
3.35	Constellation 4-QAM. Le taux d'erreur binaire	78
3.36	Constellation 4-QAM.	81
3.37	Comparaison des performances de l'algorithme SG-CMA et SG-CMA utilisant CORDIC. Le SINR en fonction des itérations. 4-QAM, M = 2, N = 2, SNR= 15dB.	82
4.1	Architecture de la plate-forme matérielle	86
4.2	Composant d'adaptation pour un fonctionnement GALS	87
4.3	Deux solutions de gestion de reconfiguration sur FPGA	88
4.4	Etage itératif de l'opérateur CORDIC	89
4.5	Schéma de l'opérateur CORDIC en mode rotation	90
4.6	Architecture générale du décodeur "V-BLAST Square Root"	91
4.7	Principe de fonctionnement du module de la décomposition QR	92
4.8	Présentation des résultats des calculs à chaque itération	93
4.9	Schéma du calcul de la norme minimale	94
4.10	Algorithme exécuté par le module de calcul de la norme minimale et de sélection	95
4.11	Enchaînement du calcul de la triangularisation	95
4.12	Schéma du module de calcul du vecteur <i>nulling</i>	97
4.13	Multiplieur accumulateur complexe	97
4.14	Opérateur de multiplication complexe	98
4.15	Vue du décodeur après placement/routage	100
4.16	Taux d'erreurs binaire en fonction de SNR	101
A.1	Graphe d'une FFT sur 8 points	109
A.2	Structure de type I	110
A.3	Structure de type II	110
A.4	Graphe d'une FFT en utilisant les deux types d'opérateur CORDIC	110
B.1	SVD d'une matrice complexe - calculs des angles	114
B.2	SVD d'une matrice complexe - calculs des rotations	114
B.3	structure systolique de l'algorithme Jacobi-SVD	115
B.4	liens de communications entre les processeurs	116

B.5	Conditions d'échanges entre les processeurs	117
C.1	Architecture de la carte mère (Source DS Sundance)	120
C.2	Architecture du module DSP (Source DS Sundance)	120
C.3	Schéma simplifié de la plat-forme Sundance	121
C.4	Diagramme de la modulation AM	124
C.5	Diagramme de la démodulation AM	124
C.6	Diagramme de la modulation FM	125
C.7	Diagramme de la démodulation FM	126
C.8	Exemple d'architecture décrite sous SynDEx	126
C.9	Exemple d'algorithme (démodulation AM) décrite sous SynDEx	127
C.10	Architecture reconfigurable du système	128
C.11	Poster présenté au SDR Forum 2007	129
D.1	Chronogramme de la première itération du module de la décomposition QR	131
D.2	Chronogramme de la deuxième itération du module de la décomposition QR	132
D.3	Chronogramme des calculs du processeur PE1	132
D.4	Chronogramme des signaux en entrée du décodeur V-BLAST	133
D.5	Chronogramme des signaux en sortie du décodeur V-BLAST	134
E.1	Structure de calcul et de sélection à base de CORDIC	136
E.2	Calculs des vecteur " <i>nulling</i> " par l'opérateur CORDIC	137
E.3	Calculs des symboles estimés par l'opérateur CORDIC	138

Liste des tableaux

2.1	Comparaison de complexité des divers algorithmes ($M \times N$)	25
2.2	Comparaison de l'implantation ASIC des décodeur MIMO	27
3.1	Constante A_n en fonction du nombre d'étages	42
3.2	Exemple de rotations successives en mode vecteur	44
3.3	Exemple de rotations successives en mode rotation	44
3.4	Mode de fonctionnement de CORDIC	46
3.5	Notations utilisées dans les équations générales	51
3.6	Notations utilisées dans les équations	67
3.7	Résumé des calculs remplacés par l'opérateur CORDIC	80
4.1	Résultat des calculs à la première itération sous Matlab et sous Modelsim .	94
4.2	Résultats des synthèses du décodeur V-BLAST Square Root (avec l'opérateur CORDIC en 15 étages)	99
4.3	Résultats des synthèses du décodeur V-BLAST Square Root (avec l'opérateur CORDIC en 6 étages)	99
E.1	Résumé des calculs remplacés par l'opérateur CORDIC	139

Bibliographie

- [1] J.MITOLA, « The software Radio Architecture ». *IEEE Communications Magazine*, pages 26–38, May 95.
- [2] Jacques PALICOT et Chistian ROLAND, « La radio logicielle : enjeux, contraintes et perspectives ». *Revue de l'Électricité et de l'Électronique*, vol. 7, décembre 2001.
- [3] M. CUMMINGS et S. HARUYAMA, « FPGA in the software radio ». *Communications Magazine, IEEE*, vol. 37, n°2, pages 108–112, Feb 1999.
- [4] G. FOSCHINI et M. GANS, « On Limits of Wireless Communication in a Fading Environment when using Multiple Antennas ».
- [5] I. Emre TELATAR, « Capacity of multi-antenna Gaussian channels ». *European Transactions on Telecommunications*, vol. 10, pages 585–595, 1999.
- [6] D. GREIFENDORF, J. STAMMEN, S. SAPPOK, M. van ACKEREN et P. JUNG, « A novel hardware design paradigm for mobile 'software defined radio' terminals ». *Spread Spectrum Techniques and Applications, 2002 IEEE Seventh International Symposium on*, vol. 3, pages 828–831 vol.3, 2002.
- [7] P.G. COOK et W. BONSER, « Architectural overview of the SPEAKEasy system ». *Selected Areas in Communications, IEEE Journal on*, vol. 17, n°4, pages 650–661, Apr 1999.
- [8] V. G. BOSE et A. B. SHAH, « Software Radios for Wireless Networking ». *Infocomm '98, San Fransisco*, vol. 3, pages 828–831 vol.3, April 1998.
- [9] T. TURLETTI, H.J. BENTZEN et D. TENNENHOUSE, « Toward the software realization of a GSM base station ». *Selected Areas in Communications, IEEE Journal on*, vol. 17, n°4, pages 603–612, April 1999.
- [10] « SDR Forum, Accelerating the proliferation of software defined radio technologies ». <http://www.sdrforum.org>.
- [11] « Informations sur End-to-End Reconfigurabilité Project ». *IST IP E²R Project phase 2*, <http://e2r2.motlabs.com>.
- [12] H. HARADA, « Software defined radio prototype for W-CDMA and IEEE802.11a wireless LAN ». *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, 2004.
- [13] Jeffrey H. REED, « Software Radio : A Modern Approach to Radio Engineering ». *Prentice Hall, NJ, USA, 2002*.
- [14] ENOUT Alain HUMBLET Pierre A. MONTALBANO Giuseppe NORDIO Alessandro CHRISTIAN BONNET, CAIRE Giuseppe, « An open software-radio architecture

- supporting advanced 3G+ systems ». *Annales des télécommunications*, vol. vol. 56, no.5-6, pp. 332-343, 2001.
- [15] M.GHOZZI, « Intérêts des techniques de paramétrization pour des architectures radio logicielle reconfigurables ». *Rapport de stage Master, Université de Rennes1/SUPELEC*, Juin 2004.
- [16] J. PALICOT et C. ROLAND, « FFT : a basic function for a reconfigurable receiver ». *Telecommunications, 2003. ICT 2003. 10th International Conference on*, vol. 1, pages 898–902 vol.1, Feb.-1 March 2003.
- [17] A. AL GHOUWAYEL, Y. LOUET et J. PALICOT, « A reconfigurable architecture for the FFT operator in a software radio context ». *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–, 0-0 2006.
- [18] L.GODARD, H.WANG, C.MOY et P.LERAY, « Common Operators Design on Dynamically Reconfigurable Hardware for SDR Systems ». 5-9 November 2007.
- [19] G. J. FOSHINI, « Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas ». *Bell Labs Technical Journal*, pages 41–57, Autumn 1996.
- [20] V. TAROKH, N. SESHADRI et A.R. CALDERBANK, « Space-time codes for high data rate wireless communications : performance criteria and code construction ». *IEEE Transactions on Information Theory*, vol. 44, pages 744–765, Mars 1998.
- [21] V. TAROKH, H. JAFARKHANI et A.R. CALDERBANK, « Space-time block codes from orthogonal designs ». *Information Theory, IEEE Transactions on*, vol. 45, n°5, pages 1456–1467, Jul 1999.
- [22] E. TELESTAR, « Capacity of multi-antenna gaussian channels ». *Bell Labs, Tech. Rep.*, June 1995.
- [23] IEEE Std 802.11, « <http://www.ieee802.org/11/index.shtml> ».
- [24] IEEE Std 802.16, « <http://www.ieee802.org/groups/802/16/> ».
- [25] A. IKHLEF, K. ABED-MERAIM et D. LE GUENNEC, « A fast blind adaptive separation algorithm using multiuser kurtosis maximization criterion ». *Signal Processing Advances in Wireless Communications, 2007. SPAWC 2007. IEEE 8th Workshop on*, pages 1–5, June 2007.
- [26] A. IKHLEF, D. LE GUENNEC et J. PALICOT, « A new algorithm to blind separate QAM signals in narrowband MIMO communication systems ». *Signal Processing and Its Applications, 2005. Proceedings of the Eighth International Symposium on*, vol. 2, pages 459–462, 28-31, 2005.
- [27] S. DAUMONT et D. LE GUENNEC, « Blind source separation with order recovery for MIMO system and an Alamouti or Tarokh space-time block coding scheme ». *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 431–436, Dec. 2007.
- [28] H. BOLCSKEI, D. GESBERT et A.J. PAULRAJ, « On the capacity of OFDM-based spatial multiplexing systems ». *Communications, IEEE Transactions on*, vol. 50, n°2, pages 225–234, Feb. 2002.
- [29] J. WINTERS, « On the Capacity of Radio Communication Systems with Diversity in a Rayleigh Fading Environment ». *Selected Areas in Communications, IEEE Journal on*, vol. 5, n°5, pages 871–878, Jun 1987.

- [30] D.GESBERT, M. SHAFI, D. S. SHIU, P. SMITH et A. NAGUIB, « From theory to practice : An overview of MIMO space-Time coded wireless systems ». *IEEE Journal on Selected Areas in Communications*, vol. 21, n°3, pages 281–302, April 2003.
- [31] D.GESBERT et J. AKHTAR, « Breaking the barriers of Shannon’s capacity : An overview of MIMO wireless systems ». *Telektronikk Telenor Journal*, Jan. 2002.
- [32] S.M.ALAMOUTI, « A simple transmit diversity technique for wireless communications ». *Selected Areas in Communications, IEEE Journal on*, vol. 16, n°8, pages 1451–1458, Oct 1998.
- [33] Vahid TAROKH, Nambi SESHADRI et A. Robert CALDERBANK, « Space-Time Codes for High Data Rate Wireless Communications : Performance criterion and Code Construction ». *IEEE Transactions on Information Theory*, vol. 44, n°2, pages 744–765, 1998.
- [34] Zhan GUO et Peter NILSSON, « VLSI implementation issues of lattice decoders for MIMO systems ». In *ISCAS (4)*, pages 477–480, 2004.
- [35] B.L. HUGHES, « Differential space-time modulation ». *IEEE Transactions on Information Theory*,, vol. 46, n°7, pages 2567–2578, Nov 2000.
- [36] A. STEFANOV et T.M. DUMAN, « Turbo-coded modulation for systems with transmit and receive antenna diversity over block fading channels : system model, decoding approaches, and practical considerations ». *Selected Areas in Communications, IEEE Journal on*, vol. 19, n°5, pages 958–968, May 2001.
- [37] C. SCHLEGEL et A. GRANT, « Differential space-time turbo codes ». *IEEE Transactions on Information Theory*,, vol. 49, n°9, pages 2298–2306, Sept. 2003.
- [38] C. SCHLEGEL et A. GRANT, « Concatenated space-time coding ». *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2001*, vol. 1, pages C–139–C–143 vol.1, Sep 2001.
- [39] B. HASSIBI et B.M. HOCHWALD, « Cayley differential unitary space-time codes ». *IEEE Transactions on Information Theory*,, vol. 48, n°6, pages 1485–1503, Jun 2002.
- [40] G. GOLDEN, G. FOSCHINI, R. VALENZUELA et P. WOLNIASKY, « Detection algorithm and initial laboratory results using the V-BLAST space-time communication architecture ». *Electronics Letters*, vol. 35, pages 14–15, January 1999.
- [41] P.W.WOLNIANSKY, G.J.FOSCHINI, G.D.GOLDEN et R.A.VALENZUELA, « V-BLAST : an architecture for realizing very high data rates over the rich-scattering wireless channel ». *1998 URSI International Symposium on Signals, Systems, and Electronics, ISSSE 98.*, pages 295–300, 29 Sep-2 Oct 1998.
- [42] Sirikiat Lek ARIYAVISITAKUL, « Turbo Space-Time Processing to Improve Wireless Channel Capacity ». In *ICC (3)*, pages 1238–1242, 2000.
- [43] M. SELLATHURAI et S. HAYKIN, « Turbo-BLAST for wireless communications : theory and experiments ». *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 50, n°10, pages 2538–2546, Oct 2002.
- [44] M.O.DAMEN, K.ABED-MERAIM et J.-C.BELFIORE, « A generalized lattice decoder for asymmetrical space-time communication architecture ». *Acoustics, Speech, and Signal Processing, 2000. ICASSP ’00. Proceedings. 2000 IEEE International Conference on*, vol. 5, pages 2581–2584 vol.5, 2000.

- [45] M.O.DAMEN, A.CHKEIF et J.-C.BELFIORE, « Lattice code decoder for space-time codes ». *Communications Letters, IEEE*, vol. 4, n°5, pages 161–163, May 2000.
- [46] J.LUO, K.PATTIPATI, P.WILLETT et L.BRUNEL, « Branch-and-bound-based fast optimal algorithm for multiuser detection in synchronous CDMA ». *IEEE International Conference on Communications, 2003. ICC '03.*, vol. 5, pages 3336–3340 vol.5, 11-15 May 2003.
- [47] A. H. LAND et A. G. DOIG, « An automatic method for solving discrete programming problems ». *Econometrica*, vol. 28, n°3, pages 497–520, 1960.
- [48] A. NAFKHA, C. ROLAND et E. BOUTILLON, « A near-optimal multiuser detector for MC-CDMA systems using geometrical approach ». *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 3, pages iii/877–iii/880 Vol. 3, March 2005.
- [49] B. HASSIBI, « An efficient square-root algorithm for BLAST ». In *Proceedings. 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing : ICASSP '00, 5–9 June 2000*, vol. 2, pages II737–II740, 2000.
- [50] A. BURG, N. FELBER et W. FICHTNER, « A 50 Mbps 4×4 maximum likelihood decoder for multiple-input multiple-output systems with QPSK modulation ». *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, vol. 1, pages 332–335 Vol.1, Dec. 2003.
- [51] Z. GUO et P. NILSSON, « A VLSI architecture of MIMO detection for future wireless communications ». *Proc. IEEE PIMRC'03*, vol. 3, pages 2852–2856, 2003.
- [52] R. Cheng K. WONG, C. Tsui et W. Mow, « A VLSI architecture of a. K-best lattice decoding algorithm for MIMO channels ». *Proc. IEEE ISCAS'02*, vol. 3, pages 273–276, 2002.
- [53] D. GARRETT, G.K. WOODWARD, L. DAVIS et C. NICOL, « A 28.8 Mb/s 4×4 MIMO 3G CDMA receiver for frequency selective channels ». *Solid-State Circuits, IEEE Journal of*, vol. 40, n°1, pages 320–330, Jan. 2005.
- [54] ANS B. HERAULT J., JUTTEN C., « Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé ». *10e Colloque GRETSI*, vol. pp. 1017-1022, Nice, mai 1985.
- [55] J.-F. CARDOSO, « Source separation using higher order moments ». *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 2109–2112 vol.4, May 1989.
- [56] J.-F. CARDOSO, « Super-symmetric decomposition of the fourth-order cumulant tensor. Blind identification of more sources than sensors ». *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 3109–3112 vol.5, Apr 1991.
- [57] J.F. CARDOSO et A. SOULOUMIAC, « Blind beamforming for non-Gaussian signals ». *Radar and Signal Processing, IEE Proceedings F*, vol. 140, n°6, pages 362–370, Dec 1993.
- [58] C.B. PAPADIAS, « Globally convergent blind source separation based on a multiuser kurtosis maximization criterion ». *Signal Processing, IEEE Transactions on*, vol. 48, n°12, pages 3508–3519, Dec 2000.

- [59] L. CASTEDO, C.J. ESCUDERO et A. DAPENA, « A blind signal separation method for multiuser communications ». *Signal Processing, IEEE Transactions on*, vol. 45, n°5, pages 1343–1348, May 1997.
- [60] A.-J. van der VEEN et A. PAULRAJ, « An analytical constant modulus algorithm ». *Signal Processing, IEEE Transactions on*, vol. 44, n°5, pages 1136–1155, May 1996.
- [61] D. GODARD, « Self-Recovering Equalization and Carrier Tracking in Two-Dimensional Data Communication Systems ». *Communications, IEEE Transactions on*, vol. 28, n°11, pages 1867–1875, Nov 1980.
- [62] J. TREICHLER et B. AGEE, « A new approach to multipath correction of constant modulus signals ». *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 31, n°2, pages 459–472, Apr 1983.
- [63] R. GOOCH et J. LUNDELL, « The CM array : An adaptive beamformer for constant modulus signals ». *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, vol. 11, pages 2523–2526, Apr 1986.
- [64] Jian YANG, J.-J. WERNER et G.A. DUMONT, « The multimodulus blind equalization and its generalized algorithms ». *Selected Areas in Communications, IEEE Journal on*, vol. 20, n°5, pages 997–1015, Jun 2002.
- [65] P. SANSRIMAHACHAI, D.B. WARD et A.G. CONSTANTINIDES, « Blind source separation for BLAST ». *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, vol. 1, pages 139–142 vol.1, 2002.
- [66] A. GOUPIL et J. PALICOT, « New Algorithms for Blind Equalization : The Constant Norm Algorithm Family ». *Signal Processing, IEEE Transactions on*, vol. 55, n°4, pages 1436–1444, April 2007.
- [67] C.B. PAPADIAS, « Unsupervised receiver processing techniques for linear space-time equalization of wideband multiple input / multiple output channels ». *Signal Processing, IEEE Transactions on*, vol. 52, n°2, pages 472–482, Feb. 2004.
- [68] E. MOREAU et J.-C. PESQUET, « Generalized contrasts for multichannel blind deconvolution of linear systems ». *Signal Processing Letters, IEEE*, vol. 4, n°6, pages 182–183, Jun 1997.
- [69] Wei ZHA et S.D.BLOSTEIN, « Multiuser receivers that are robust to delay mismatch ». *IEEE Transactions on Communications*, vol. 50, n°12, pages 2072–2081, Dec 2002.
- [70] Reiner W. HARTENSTEIN, « Coarse grain reconfigurable architecture (embedded tutorial) ». pages 564–570, ACM, 2001.
- [71] Karim Ben CHEHIDA, « Méthodologie de partitionnement Logiciel/Matériel pour Plateformes Reconfigurables Dynamiquement ». *Thèse de Doctorat*, Université catholique de Louvain, Faculté des sciences appliquées- Laboratoire de microélectronique, 2002.
- [72] Lilian BOSSUET, « Exploration de l'espace de conception des architectures reconfigurables ». *Thèse de Doctorat*, Université Bretagne Sud, LESTER, 2004.
- [73] T. Pionteck J. BECKER et M. GLESNER, « DReAM : A Dynamically Reconfigurable Architecture for Future Mobile Communication Applications ». *International Workshop on Field Programmable Logic and Applications*, vol. Lecture notes in Computer Science 1896/2000, pages 312–321, August 2000.

- [74] H. SINGH, Ming-Hau LEE, Guangming LU, F.J. KURDAHI, N. BAGHERZADEH et E.M. CHAVES FILHO, « MorphoSys : an integrated reconfigurable system for data-parallel and computation-intensive applications ». *IEEE Transactions on Computers*, vol. 49, n°5, pages 465–481, May 2000.
- [75] Darren C. CRONQUIST, Chris FISHER, Miguel FIGUEROA, Paul FRANKLIN et Carl EBELING, « Architecture Design of Reconfigurable Pipelined Datapaths ». *ARVLSI '99 : Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, page 23, 1999.
- [76] Reiner W. HARTENSTEIN et Rainer KRESS, « A datapath synthesis system for the reconfigurable datapath architecture ». *ASP-DAC '95 : Proceedings of the 1995 conference on Asia Pacific design automation (CD-ROM)*, page 77, 1995.
- [77] J. BECKER, T. PIONTECK et M. GLENNER, « An Application-Tailored Dynamically Reconfigurable Hardware Architecture for Digital Baseband Processing ». *SBCCI '00 : Proceedings of the 13th symposium on Integrated circuits and systems design*, page 341, 2000.
- [78] Amir H. KAMALIZAD, Chengzhi PAN et Nader BAGHERZADEH, « Fast Parallel FFT on a Reconfigurable Computation Platform ». *SBAC-PAD '03 : Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, page 254, 2003.
- [79] G. SASSATELLI, « Architectures reconfigurables dynamiquement pour les systèmes sur puc ». *Thèse de Doctorat*, Université de Montpellier, France, 2002.
- [80] Gilles SASSATELLI, Lionel TORRES, Jérôme GALY, Gaston CAMBON et Camille DIOU, « The Systolic Ring : A Dynamically Reconfigurable Architecture for Embedded Systems ». *Field-Programmable Logic and Applications, 11th International Conference, FPL 2001, Belfast, Northern Ireland, UK, August 27-29, 2001, Proceedings*, vol. 2147, pages 409–419, 2001.
- [81] R. DAVID, « Architectures reconfigurables dynamiquement pour les applications mobiles ». *Thèse de Doctorat*, Université de Rennes 1, France, 2003.
- [82] R. DAVID, D. CHILLET, S. PILLEMENT et O. SENTIEYS, « DART : a dynamically reconfigurable architecture dealing with future mobile telecommunications constraints ». *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 156–163, 2002.
- [83] J.E. VOLDER, « The CORDIC Trigonometric Computing Technique ». *IRE Transactions on Electronic Computers*, vol. EC-8, n°3, pages 330–334, 1959.
- [84] A. M. DESPAIN, « Very Fast Fourier Transform Algorithms Hardware for Implementation ». *IEEE Transactions on Computers*, vol. C-28, n°5, pages 333–341, 1979.
- [85] A. M. DESPAIN, « Fourier Transform Computers using CORDIC Iterations ». *IEEE Transactions on Computers*, vol. C-23, n°10, pages 993–1001, October, 1974.
- [86] S. K. RAO, « Orthogonal digital filters for VLSI implementation ». *IEEE Transactions on circuits and systems*, vol. CAS-31, n°11, November, 1984.
- [87] Tze-Yun SUNG et Yu-Hen HU, « VLSI Implementation of real-time Kalman filter ». *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, vol. 11, pages 2223–2226, Apr 1986.

- [88] H.M. AHMED, J.-M. DELOSME et M. MORF, « Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing ». *Computer*, vol. 15, n°1, pages 65–82, Jan 1982.
- [89] Joseph R. CAVALLARO et Franklin T. LUK, « CORDIC Arithmetic for an SVD Processor ». *J. Parallel Distrib. Comput.*, vol. 5, n°3, pages 271–290, 1988.
- [90] G.Privat et M.RENAUDIN, « L'algorithme CORDIC dans les architectures spécialisées de traitement numérique du signal ». *Traitemet du Signal Journal*, vol. 5, n°6, 1988.
- [91] R. ANDRAKA, « A Survey of CORDIC Algorithms for FPGAs ». *FPGA '98. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, vol. 34, n°2, pages 191–200, Feb. 22-24 1998.
- [92] J. VALLS et et AL., « The use of CORDIC in software defined radios : A Tutorial ». *IEEE Communications Magazine*, vol. 44, n°9, pages 46–50, October 2006.
- [93] Hongzhi WANG, Pierre LERAY et Jacques PALICOT, « A CORDIC-based dynamically reconfigurable FPGA architecture for signal processing algorithms ». *The XXIX General Assembly of the International Union of Radio Science(URSI08)*, August 2008.
- [94] Z. LIU, K. DICKSON et J. V. McCANNY, « A floating-point CORDIC based SVD processor ». In *ASAP*, 2003.
- [95] J.M. DELOSME, « A processor for two-dimensional symmetric eigenvalue and singular value arrays ». *IEEE 21th Asilomar conf. Circuits, Syst. and Comput.*, pages 217–221, 1987.
- [96] B. YANG et J.F. BOHME, « Reducing the computations of the SVD array given by Brent and Luk ». *SIAM J.Matrix Anal. Appl.*, vol. 12, n°4, pages 713–725, 1991.
- [97] M.D. ERCEGOVAC et T. LANG, « Redundant and on-line CORDIC : Application to matrix triangularization and SVD ». *IEEE transaction on Computer*, vol. 39, n°6, pages 725–740, 1990.
- [98] M. Terré C. VIROULAUD, « Implantation à base d'opérateurs CORDIC d'un algorithme des moindres carrés rapides par décomposition QE ». *Quinzième colloque GRETSI*, Sept. 1995.
- [99] Pierre LERAY, « Filtrage numérique - Filtrage adaptatif ». *École Supérieure d'Électricité*, 1998.
- [100] Gene H. ; Charles F. Van Loan GOLUB, « Matrix Computations (3/e ed.) ». *Baltimore : John Hopkins University Press. ISBN 978-08018-5414-9*, 1996.
- [101] Jean-Philippe DELAHAYE, « Plate-forme hétérogène reconfigurable : application à la radio logicielle ». *Thèse de Doctorat*, Université Rennes1, SCEE, 2007.
- [102] D.LIM et M.PEATTIE, « Two *Flows for Partial Reconfiguration : Module Based or Small Bit Manipulations ». *Xilinx, Inc., 2100 Logic Drive*,.
- [103] 2100 Logic Drive XILINX, Inc., « Development system Reference Guide, <http://www.xilinx.com> ».
- [104] Hongzhi WANG, Pierre LERAY et Jacques PALICOT, « A Reconfigurable Architecture for MIMO Square Root Decoder ». *Reconfigurable Computing : Architectures and Applications*, pages 317–322, 2006.

- [105] Hongzhi WANG, Jean-Philippe DELAHAYE, Pierre LERAY et Jacques PALICOT, « Managing dynamic reconfiguration on MIMO Decoder ». *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–8, 2007.
- [106] Hongzhi WANG, Pierre LERAY et Jacques PALICOT, « An Efficient MIMO V-BLAST Decoder Based on a Dynamically Reconfigurable FPGA Including its Reconfiguration Management ». *Communications, 2008. ICC '08. IEEE International Conference on*, pages 746–750, May 2008.
- [107] Hongzhi WANG, Pierre LERAY et Jacques PALICOT, « A reconfigurable architecture for MIMO detection using CORDIC operator ». *4th Karlsruhe Workshop on Software Radios (wrs06)*, March 2006.
- [108] Hongzhi WANG, Pierre LERAY et Jacques PALICOT, « A Reconfigurable Architecture for MIMO Square Root Decoder ». *Reconfigurable Computing : Architectures and Applications, Second International Workshop, ARC 2006, Delft, The Netherlands, March 1-3, 2006, Revised Selected Papers*, pages 317–322, 2006.
- [109] J.P. DELAHAYE, J. PALICOT et P.LERAY, « A Hierarchical Modeling Approach in Software Defined Radio System Design ». *SIPS 2005*, vol. Athens ,Greece, 2005.
- [110] Brandon BLODGET, Scott McMILLAN et Patrick LYSAGHT, « A Lightweight Approach for Embedded Reconfiguration of FPGAs ». In *DATE*, pages 10399–10401, IEEE Computer Society, 2003.
- [111] Brandon BLODGET, Philip JAMES-ROXBY, Eric KELLER, Scott McMILLAN et Prasanna SUNDARARAJAN, « A Self-reconfiguring Platform ». In *FPL* (Peter Y. K. CHEUNG, George A. CONSTANTINIDES et José T. de SOUSA (eds.)), vol. 2778 of *Lecture Notes in Computer Science*, pages 565–574, Springer, 2003.
- [112] C. M. RADER, « VLSI Systolic Arrays for daptive Nulling ». vol. 13, pages 29–49, 1996.
- [113] Sang Yoon PARK, Nam Ik CHO, Sang Uk LEE, Kichul KIM et Jisung OH, « Design of 2K/4K/8K-point FFT processor based on CORDIC algorithm in OFDM receiver ». *Communications, Computers and signal Processing, 2001. PACRIM. 2001 IEEE Pacific Rim Conference on*, vol. 2, pages 457–460 vol.2, 2001.
- [114] B. HEYNE et J. GOETZE, « A pure CORDIC based FFT for reconfigurable digital signal processing ». *12th European Signal Processing Conference (Eusipco2004)*, vol. 7, September 2004.
- [115] F. T. Luk R. P. BRENT et C. Van LOAN, « Computation of the singular value decomposition using mesh-connected processors ». *Journal of VLSI Computer Systems*, vol. 1, n°3, pages 242–270, 1985.
- [116] G. E. FORSYTHE et P. HENRICI, « The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix ». *Transactions of the American Mathematical Society*, vol. 94, n°1, pages 1–23, Jan. 1960.
- [117] E.G. KOGBETLIANTZ, « Solution of Linear Systems by Diagonalization of Coefficients Matrix ». *Quarterly of Applied Math.*, vol. 13, pages 123–132, 1955.
- [118] J.R. CAVALLARO et F.T LUK, « Architecture for a CORDIC SVD Processor ». *Proceeding SPIE Real-Time Signal Processing*, vol. 5(3), pages 271–290, 1986.

- [119] R. P. BRENT et F. T. LUK, « The solution of the Singular-Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays ». *SIAM Journal of Scientific and Statistical Computing*, vol. 6, n°1, pages 69–84, 1985.
- [120] Amor NAFKHA, « A geometrical approach detector for solving the combinatorial optimisation problem : Application in wireless communication systems ». *Thèse de Doctorat*, Université Bretagne Sud, LESTER, 2006.