Management d'opérateurs communs dans les architectures de terminaux multistandards.

Malek Naoues

N° d'ordre : 2013-26-TH

THÈSE EN CO-TUTELLE

Pour obtenir le grade de Docteur délivré par :

SUPELEC

Ecole Doctorale MATISSE
DOMAINE : STIC
SPECIALITE : Télécommunications

SUP'Com

Université de Carthage
Ecole Supérieure des Communications de Tunis
Ecole Doctorale en TIC

Présentée par :

Malek NAOUES

Sujet :

Management d'opérateurs communs dans les architectures de terminaux multistandards

Soutenue le 26 Novembre 2013 devant les membres du jury :

Président du jury :	M. Sami TABBANE	Professeur à Sup'Com, Tunis
Rapporteurs :	M. Bertrand GRANADO	Professeur à l'UPMC, Paris
	M. Mohamed ABID	Professeur à l'ENIS, Sfax
Examinateur :	M. Jacques PALICOT	Professeur à Supélec, Rennes
Directeurs de Thèse :	M. Yves LOUET	Professeur à Supélec, Rennes
	M. Adel GHAZEL	Professeur à Sup'Com, Tunis
Membres invités :	M. Dominique NOGUET	Expert senior CEA-LETI, Grenoble
	M. Khaled GRATI	Maître assistant à Sup'Com, Tunis

Résumé

Les équipements de communications numériques intègrent de plus en plus de standards. Cette tendance se confirme et les équipements mettant en oeuvre ces standards se devront d'être reconfigurables pour deux raisons principales. Tout d'abord, la mise rapide sur le marché (time-to-market) impose de concevoir un nouvel équipement en réutilisant autant que possible les blocs de la version précédente. Ensuite, lors de l'utilisation de l'équipement, la commutation d'un standard à l'autre doit pouvoir se faire au prix d'un surcoût matériel modéré, ce qui impose l'utilisation de ressources communes à ces standards dans des instanciations différentes. Cette notion de reconfigurabilité introduite à tous les niveaux (couches physiques, piles protocolaires, applications) des systèmes radio (stations de base, terminaux) est donc fondamentale. La plateforme matérielle nécessaire à l'exécution d'une couche physique multistandard est le segment du système présentant le plus de contraintes par rapport à la reconfiguration : réactivité, consommation et occupation de ressources matérielles. Cependant, la réponse à ces contraintes ne réside pas dans un processeur généraliste puissant supportant l'ensemble des traitements comme l'avait défini les premiers travaux de la radio logicielle. Le défi est important de par le fait que l'utilisation seule de processeurs généralistes ne semble pas être une solution ni à court, ni à long terme. Une approche pragmatique de conception des plateformes reconfigurable est alors nécessaire pour traiter efficacement la diversité des fonctions en bande de base issues des différents standards.
La paramétrisation se situe parmi les techniques de reconfiguration actuelles et vise une implémentation multistandards plus efficace. L'objectif de cette technique est d'identifier des traitements communs entre les standards, voire entre blocs de traitement au sein d'un même standard, afin de définir des blocs génériques pouvant être réutilisés facilement. Dans ce contexte, l'approche des opérateurs communs (OC) a été proposée. Celle-ci consiste à identifier des entités, appelées opérateurs communs, utilisées massivement par les différents modules d'un équipement multistandards. Étant reconfigurables par un simple chargement de paramètres, les opérateurs communs permettent d'obtenir un équipement multistandards reconfigurable tout en limitant le nombre d'éléments physiques à implémenter. Par conséquent, cette approche vise principalement à concevoir des équipements évolutifs pouvant

s'adapter à un large éventail de standards.
Le travail de recherche présenté propose une approche de conception qui a pour objectif de permettre le déploiement d'applications reconfigurables sur une plateforme matérielle hétérogène basée sur des opérateurs communs. Nous définissons dans ce manuscrit le management des opérateurs communs et nous étudions l'implémentation des opérateurs en se basant essentiellement sur des évaluations de complexité pour quelques standards utilisant la modulation OFDM. Nous montrons que les algorithmes de FFT et décodage de canal sont des traitements lourds en terme de calcul. Donc, nous nous concentrons sur la définition d'une technique de gestion efficace des ressources entre ces familles d'algorithmes en définissant une nouvelle structure matérielle d'un opérateur commun FFT/Viterbi. Nous proposons ainsi une architecture d'un processeur commun permettant la gestion efficace des ressources matérielles entre les algorithmes FFT et décodage de Viterbi. L'architecture, que nous avons proposé et implémenté sur FPGA, permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permet l'accélération des traitements.
Les résultats des implémentations montrent que l'utilisation de cette architecture commune offre des gains en complexité pouvant atteindre 30% dans les configurations testées par rapport à une implémentation classique avec une réduction importante de l'occupation mémoire due au partage de la mémoire entre les algorithmes. Ainsi, notre approche permet de profiter du partage des ressources et de parallélisme des traitements, non seulement pour ajouter de la flexibilité au design, mais aussi pour diminuer la complexité et la surface des circuits.

Mots-clés

Radio reconfigurable ; Conception flexible ; Paramétrisation ; Opérateurs communs ; FFT ; Décodage de Viterbi.

Abstract

Today's telecommunication systems require more and more flexibility, and reconfiguration mechanisms are becoming major topics especially when it comes to multistandard designs. In typical hardware designs, the communication standards are implemented separately using dedicated instantiations which are difficult to upgrade for the support of new features. To overcome these issues, Software Defined Radio (SDR) describes a software-reconfigurable multistandard, and multiband radios. The SDR technique is the main way to design flexible and reconfigurable architectures capable of supporting different transmission standards in a single platform. A digital communication chain, when supporting different standards, uses typical signal processing operations such as modulation, channel coding, equalization, etc. These common operations can be identified and then explored to take advantage of the commonalities among tasks in order to enhance power efficiency and area occupation. In this context, parameterization technique has been introduced in the area of digital communications. It consists in identifying the common aspects among the targeted modes and standards in order to define a generic operation capable of handling the required tasks. This generic operation can switch from a configuration to another by a simple change of its parameters.
In this work, we exploit a parameterization approach called the common operator technique that can be considered to build a generic terminal capable of supporting a large range of communication standards. The main principle of the common operator technique is to identify common elements based on smaller structures that could be widely reused across signal processing functions. This technique aims at designing a scalable terminal based on medium granularity operators, larger than basic logic cells and smaller than signal processing functions. Similarly to flip flop or logic gate, a Common Operator (CO) is used regardless of its calling functions. From this point, the common operator technique aims at being less standard dependent than the highest level reuse approaches where the functionality required by a standard is implemented and executed when needed. It is expected that the reduction of the exploration space to telecommunication baseband functions will help defining such medium-grain common operators. The resulted implementation is expected to be more flexible and scalable for a wide range of standards.

Such a regular structure is also well adapted to cope with silicon technology process variability. Indeed, as CMOS technology shrinks, the performance of the operator instances may vary across space (on the silicon wafer) and time. Dealing with regular building blocks helps to map the most demanding algorithms onto the best performing cells, making the design dependable and self-healing. Many previous works focused on defining implementing and managing the Common Operators are presented in this manuscript.
The present work deals with two widely used algorithms in wireless communication systems : Viterbi decoding and Fast Fourier Transform (FFT). These algorithms require a significant computational complexity, and combining them would result in a significant step forward. Implementing the FFT and Viterbi algorithms in a multistandard context through a common architecture poses significant architectural constraints. Indeed, to meet the needs of the Common Operators technique and support multiple standards, a flexible FFT/Viterbi processor architecture allowing variable FFT size and variable Viterbi constraint length is necessary. The idea is to divide the problem into equal parts for each algorithm and share equally the execution time in each physically implemented common butterfly units. In this thesis, we focus on the design of a flexible processor to manage the common operators and take advantage from structural similarities between FFT and Viterbi trellis.
A flexible FFT/Viterbi processor was proposed and implemented on FPGA and compared to dedicated hardware implementations. The results show a considerable gain in flexibility. This gain is achieved with no complexity overhead since the complexity if even decreased by more than 30% in some configurations.

Keywords

Reconfigurable Radio ; Flexible Design ; Parametrization ; Common Operators ; FFT ; Viterbi.

Remerciements

Les travaux de recherche présentés dans cette thèse de doctorat se sont déroulés principalement au sein du Laboratoire Solutions sans fil et Plateformes numériques (LSP) du Service Technologies de la Communication et de la Sécurité (STCS) au CEA-LETI à Grenoble. Je tiens à remercier toutes les personnes qui ont contribué de près ou de loin à la réalisation de ces travaux.

Je voudrais remercier aussi mes deux directeurs de thèse Prof. Yves Louët de Supélec Rennes et Prof. Adel Ghazel de SUP'Com Tunis. Leurs encadrements, les conseils et le soutien dont ils m'ont fait bénéficier durant ce travail se sont révélés précieux.

Je tiens à remercier Dominique Noguet mon encadrant CEA de m'avoir proposé le sujet de thèse pour sa disponibilité, les échanges d'idées et le suivi attentif tout au long de cette thèse. Je tiens à remercier aussi Khaled Grati, mon deuxième encadrant universitaire, pour son implication et conseils pendant mes séjours à SUP'Com.

Mes sincères remerciements s'adressent aussi aux membres du jury qui ont accepté d'évaluer ce travail.

En fin, je remercie les membres du service STCS au CEA-LETI et les membres des laboratoires GRESCOM à SUP'Com et IETR-SCEE à Supélec pour les échanges constructifs, les conseils, et les moments de distractions.

Tunis, Juin 2013
Malek Naoues

Table des matières

Résumé **3**

Table des figures **13**

Liste des tableaux **15**

Liste des acronymes et abréviations **17**

Introduction générale **21**

1 La technique des opérateurs communs **27**
1.1 Introduction 27
1.2 Contexte de la radio reconfigurable 28
1.3 Les plateformes de la radio reconfigurable 30
1.3.1 Les contraintes des standards 31
1.3.2 Plateforme de traitement parallèle 32
1.3.3 Gestion des ressources 34
1.4 Réutilisation des ressources 34
1.5 La paramétrisation 37
1.5.1 Approche par fonctions communes 38
1.5.2 Approche par opérateurs communs 41
1.6 Identification des opérateurs communs 41
1.6.1 Approche théorique 42
1.6.2 Approche pragmatique 46
1.7 Un premier jeu d'opérateurs communs 46
1.7.1 Les opérateurs de faible granularité : LFSR et Cordic 47
1.7.2 Les opérateurs à forte granularité : FFT et FEC . . . 49
1.8 Conclusion 51

2 Management des opérateurs communs **53**
2.1 Introduction 53
2.2 Définition du management des OCs 54
2.2.1 Le management des OCs pendant la conception 55
2.2.2 Le management des OCs pendant l'exécution 55

2.2.3 Définition du problème de management des OCs . . . 58
2.3 Techniques de management des OCs . . . 59
2.3.1 Couches logicielles de reconfiguration . . . 60
2.3.2 Mécanismes de management des OCs . . . 61
2.3.3 La librairie des opérateurs communs . . . 61
2.3.4 Espace d'implémentation du manageur des OCs . . . 63
2.4 Considérations sur l'implémentation des OCs . . . 64
2.4.1 Classification des accélérateurs matériels . . . 65
2.4.2 Sélection des accélérateurs matériels . . . 65
2.4.3 Analyse des coûts en traitement pour les OCs . . . 69
2.5 Modèle d'architecture reconfigurable à base d'OCs . . . 75
2.6 Conclusion . . . 77

3 L'opérateur commun FFT/Viterbi 79
3.1 Introduction . . . 79
3.2 Similarités entre FFT et Viterbi . . . 80
3.2.1 Structure du décodeur de Viterbi . . . 80
3.2.2 Structure de la FFT . . . 83
3.2.3 Vers une structure commune FFT/Viterbi . . . 84
3.3 Premières structures de l'OC FFT/Viterbi . . . 86
3.4 Structure optimisée pour l'OC FFT/Viterbi . . . 89
3.4.1 Calcul du papillon FFT Radix-2 (1er mode) . . . 93
3.4.2 Calcul du ACS (2ème mode) . . . 95
3.4.3 Calcul du BMC (3ème mode) . . . 96
3.5 Comparaison des implémentations proposées . . . 98
3.6 Le jeu d'opérateurs FFT/FEC . . . 100
3.7 Conclusion . . . 100

4 Mécanismes de management de l'operateur FFT/Viterbi 103
4.1 Introduction . . . 103
4.2 Contraintes sur le mgmt. de l'OC FFT/Viterbi . . . 104
4.3 Architectures de mgmt. de l'OC FFT/Viterbi . . . 106
4.3.1 Architecture en pipeline . . . 107
4.3.2 Architecture à colonne partielle . . . 109
4.3.3 Architecture séléctionnée . . . 113
4.4 Mgmt. des interconnexions en colonne partielle . . . 113
4.4.1 Génération des interconnexions . . . 114
4.4.2 Mgmt. des largeurs de chemin de données . . . 114
4.5 Implémentation du processeur FFT/Viterbi . . . 115
4.5.1 Description de l'architecture implémentée . . . 115
4.5.2 Résultats de l'implémentation . . . 117
4.6 Conclusion . . . 117

Conclusion générale 121

Publications de l'auteur **125**

A Terminologie et concepts des systèmes radio avancés **127**
A.1 La radio non-reconfigurable 127
A.2 La radio reconfigurable 128
A.3 La radio contrôlée par logiciel 128
A.4 La radio logicielle 128
A.5 La radio intelligente 129

B L'opérateur commun transformée de Fourier rapide **131**
B.1 Utilisation dans les modulations multi-porteuses 132
B.2 Utilisation dans le codage canal 133
B.3 Utilisation dans la convolution et de corrélation 134
B.4 Utilisation dans d'autres fonctions 135

Bibliographie **137**

Table des figures

1.1 Différents types de « Handover » 29
1.2 Exemple d'une implémentation classique d'un récepteur . . . 29
1.3 Rapport flexibilité/performance des circuits en BB 31
1.4 Le circuit MAGALI 33
1.5 Exemple de partage des ressources entre trois standards . . . 34
1.6 Les fonctions réalisées par un processeur de traitement en BB 36
1.7 Classification des traitements communs proposée par Alaus . 37
1.8 Fonction de modulation paramétrable 39
1.9 Vers une chaîne de transmission multistandards unifiée 40
1.10 Banc d'opérateurs communs 42
1.11 Exemple de décomposition partielle en graphe des OCs 43
1.12 Exemple de décomposition en graphe avec les coûts associés . 44
1.13 Illustration de l'utilisation des opérateurs LFSR 47
1.14 Etage itératif pour les opérateurs LFSR ou Cordic 49
1.15 Illustration de l'utilisation de l'opérateur FFT 50
1.16 Partage des opérateurs communs entre la FFT, RS et Viterbi 51

2.1 Étapes de conception d'un équipement avec la tech. des OCs 56
2.2 Représentation en couches du manager des OCs 57
2.3 Étapes du management des OCs 59
2.4 Les composants du manager des OCs 60
2.5 Mécanismes de management des OCs 62
2.6 Degrés de libertés dans l'implémentation du manager des OCs 64
2.7 Les besoins de calculs pour la (dé)modulation et le décodage 67
2.8 Parallélisations des traitements pour le décodeur de Viterbi . 74
2.9 Modèle de gestion de configuration des OCs 75
2.10 Architecture de management local de l'OC FFT/Viterbi . . . 76
2.11 Architecture de management local de l'OC LFSR et CORDIC 77
2.12 Architecture globale de gestion des OCs 78

3.1 Structure globale d'un décodeur de Viterbi 81
3.2 Le papillon Viterbi et l'actualisation des métriques de chemin 81
3.3 La représentation du treillis Viterbi 83

3.4 Le papillon FFT Radix-2 84
3.5 La représentation du treillis FFT 84
3.6 Reprśentation commune du treillis FFT et Viterbi DIF 85
3.7 Génération des adresses pour les papillons Viterbi et FFT . . 86
3.8 Reprśentation commune du treillis FFT et Viterbi DIT . . . 86
3.9 Opérateur AS3 87
3.10 Opérateur commun FFT/Viterbi parallèle 88
3.11 Architectures du papillion FFT Radix-2 (2 et 3 multiplications) 89
3.12 Architecture du papillon FFT Radix-2 à 2 multiplications . . 90
3.13 Cellule en pipeline proposée pour les algorithmes FFT et Viterbi 92
3.14 Structure du papillon FFT Radix-2 93
3.15 Les trois modes de fonctionnement de la cellule proposée . . . 94
3.16 Implémentation des opérations de calcul de chemins 95
3.17 Entrées/Sorties du papillion de Viterbi 96
3.18 Représentation schématique des opérations BMC et ACS . . . 97
3.19 Jeu d'opérateurs FFT/FEC 101

4.1 Exemple d'architecture parallèle à géométrie constante 105
4.2 Exemple d'architecture parallèle à géométrie variable 105
4.3 Allocation dynamique des ressources entre les algorithmes . . 106
4.4 Architecture en pipeline classique d'une FFT Radix-2 107
4.5 Architecture en pipeline pour le mgmt. de l'OC FFT/Viterbi 108
4.6 Exemple d'adaptation du parallélisme des traitements 111
4.7 Architecture à colonne partielle du processeur FFT/Viterbi . 112
4.8 Exploration des architectures de gestion de l'OC FFT/Viterbi 113
4.9 Exemple de génération d'adresses mémoire 114
4.10 Masquage des bits de poids faible 115
4.11 Implémentation flexible du processeur FFT/Viterbi sur FPGA 116
4.12 Accélération des traitements en fonction des OCs affectés . . 118
4.13 Partage des ressources entre les algorithmes FFT et Viterbi . 119

A.1 Cycle de cognition de Mitola 130

B.1 Opérateur commun FFT ($\mathbb{C}$ et $GF(F_t)$) 134

Liste des tableaux

1.1 Evolution des besoins des standards 32

2.1 Domaines à explorer pour le management des opérateurs communs 55
2.2 Coûts d'implémentation des terminaux W-CDMA et 802.11a sur un GPP 68
2.3 Classification du degré de programmabilité pour quelques fonctions de traitement de signal 69
2.4 Coût en MOCPS pour quelques standards à base de la modulation OFDM 70
2.5 Comparaison des implémentations matérielles et logicielles des FFTs Radix-2 71
2.6 Contraintes des standards considérés pour le décodeur 73
2.7 Contraintes des standards considérés pour le décodeur en cycles par bit décodé 73

3.1 Comparaisons de consommation entre les architectures de papillons à 2 et à 3 multiplieurs 91
3.2 Comparaisons de complexité des opérateurs FFT/Viterbi . . 99

4.1 Comparaison des ressources utilisées entre une implémentation classique et architecture proposée 118
4.2 Comparaison des performances de l'opérateur FFT/Viterbi par rapport à une cellule classique FFT-Radix 2 118

Liste des acronymes et abréviations

2G Mobile de 2nd Generation
3G Mobile de 3rd Generation
3GPP LTE 3rd Generation Partership Project Long Term Evolution
4G Mobile de 4rd Generation
ACS Add-Compare-Select
ADC Analog-to-Digital Converter
ASIC Application Specific Integrated Circuit
ASIP Application Specific Instruction set Processor
BOC Banc d'Opérateurs Communs
BMC Branch Metric Calculation
BRAM Block RAM
CORDIC Coordinate Rotation Digital Computer
CRC Cyclic Redundancy Check
DAC Digital-to-Analog Converter
DECT Digital Enhanced Cordless Telecommunications
DFT Discrete Fourier Transform
DIF Decimation In Frequency
DIT Decimation In Time
DSP Digital Signal Processor
DVB Digital Video Broadcasting
WIFI Wireless Fidelity ou IEEE802.11
EDGE Enhanced Data rates for GSM Evolution
FDD Frequency-Division Duplexing
FDM Frequency Division Multiplexing
FEC Forward Error Correction

FFT Fast Fourier Transform
FIFO First-In-First-Out
FIR Finite Impulse Response
FLMS Frequency Domain Least Mean Square
FPGA Field Programmable Gate Array
GF Galois Field
GIPS Giga Instruction per second
GMSK Gaussian Minimum Shift Keying
GPP General Purpose Processor
GPRS General Packet Radio Service
GSM Global System for Mobile communications
HSPA High Speed Downlink Package Access
HW Hardware
IEEE the Institute of Electrical and Electronics Engineers
IF Intermediate Frequency
IFFT Inverse Fast Fourier Transform
IP Intellect Property
LFSR Linear Feedback Shift Register
LSB Least Significant Bit
LUT LookUp Table
M Multiplier
MAC Media Access Control
MIPS Million Instruction per Seconde
MPSoC Multiprocessor System-on-Chip
MSB Most Significant Bit
MUX Multiplexer
NoC Network-on-Chip
OC Opérateur Commun
OFDM Orthogonal Frequency Division Multiplexing
PE Processing Element
QAM Quadrature Amplitude Modulation
QPSK Quadrature Phase Shift Keying
R Register
RAM Random-Access Memory
RF Radio fréquence

RL Radio Logicielle

RI Radio Intelligente

RS Reed-Solomon

SDR Software Defined Radio

SIMD Single Instruction stream, Multiple Data stream

SMM Survivor Memory Management

SoC System-on-Chip

SDF Single-path Delay Feedback

SW Software

UMTS Universal Mobile Telecommunications System

UWB Ultra-WideBand

VHDL Very High Speed Integrated Circuit Hardware Description Language

VLSI Very-Large-Scale Integration

W-CDMA Wideband Code Division Multiple Access

WIMAX Worldwide éInteroperability For Microwave Access

WWRF Wireless World Research Forum

XOR eXclusive-OR

Introduction

La prolifération des solutions reconfigurables constitue une évolution considérable dans le domaine des radiocommunications qui a déjà été partiellement amorcée avec le développement des équipements multi-modes et multi-bandes. Cette évolution est alimentée par les demandes croissantes des communications sans fil pour un large éventail de besoins. A titre d'exemple, en 2012 il y a déjà plus de 4 milliards d'utilisateurs de téléphones mobiles dans le monde. De plus, des estimations telles que celles du Forum mondial de recherche sans fil (WWRF) prévoient que d'ici 2017 il y aura 7 billions d'appareils sans fil desservant 7 milliards d'utilisateurs [1]. Pour répondre à ces attentes avec la limitation du spectre radio, des méthodes plus souples pour partager les ressources matérielles et spéctrales entre plusieurs services et réseaux de communication sont nécessaires. À cet égard, il y a un intérêt croissant pour les solutions reconfigurables qui permettent la cohabitation de plusieurs systèmes radios dans le même équipement [1] multistandards [2]. Dans ce contexte, le concept de radio logicielle [3] a poussé les travaux de recherches vers les techniques de traitement pouvant s'adapter à plusieurs standards. Une architecture radio logicielle idéale est constituée d'une antenne large bande directement suivie par un convertisseur analogique/numérique à haute fréquence d'échantillonnage permettent à un circuit de type processeur de traiter ensuite un signal numérique contenant l'ensemble des signaux caractérisant une radio multistandards. La reprogrammabilité des processeurs permet l'adaptation des traitements numériques aux différents standards par simple téléchargement des logiciels. Toutefois, il existe à l'heure actuelle de nombreuses limitations technologiques ne permettant pas de concrétiser un tel concept, dont par exemple 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 circuits Radio Fréquence (RF).

La réalisation classique d'équipements multistandards se limite à la juxtaposition au sein d'un même équipement les différentes chaînes de traitement des modes et/ou standards ciblés. Cette solution, dite Velcro [4], permet de

1. Ce terme sera utilisé pour désigner indifféremment un terminal, une station de base ou un modem de télécommunication capable de traiter plusieurs modes ou plusieurs standards de communication.

traiter plusieurs standards par le biais d'un seul équipement. Néanmoins, son évolutivité est limitée aux standards considérés initialement et la complexité de sa mise en œuvre, voire la consommation, ne sont pas optimales. En effet, la complexité de la solution classique correspond à la somme des complexités de chaque standard à laquelle s'ajoute celle des organes de reconfiguration. La paramétrisation [5] [6] se situe parmi les techniques de reconfiguration actuelles et vise une implémentation multistandards plus efficace. L'objectif de cette technique est d'identifier des traitements communs entre les standards, voire entre blocs de traitement au sein d'un même standard, afin de définir des blocs génériques pouvant être implémentés indépendamment des spécifications des normes. Dans ce contexte, l'approche des opérateurs communs (OC) [7] a été proposée. Celle-ci consiste à identifier des entités, appelées opérateurs communs, utilisées massivement par les différents modules d'un équipement multistandards. La granularité de ces opérateurs communs se situe à un niveau intermédiaire entre les fonctions de hauts niveaux et les opérateurs arithmétiques de bas niveaux (opérateurs MAC [2] par exemple). La définition d'un opérateur commun est fondée sur des aspects fonctionnels et structurels et est effectuée indépendamment des standards. Ainsi, un opérateur commun est défini pour effectuer des opérations élémentaires de traitement du signal indépendamment de la fonction qui l'exécute. Étant reconfigurables par un simple chargement de paramètres, ils permettent d'obtenir un équipement multistandards reconfigurable tout en limitant le nombre d'éléments physiques à implémenter. Par conséquent, cette approche vise principalement à concevoir des équipements évolutifs pouvant s'adapter à un large éventail de standards.

Problématique

Plusieurs travaux réalisés au sein de l'équipe SCEE à Supélec et au CEA-LETI ont permis la définition de plusieurs opérateurs communs (thèses d'Al-Ghouwayel [8], Wang [9] et Alaus [10]). D'autres travaux des mêmes équipes ont traité l'identification des opérateurs communs en se basant sur une approche mathématique avec la théorie des graphes (thèses de Gul [11] et Kaiser [12]). Cependant, tous ces travaux se limitent à l'identification et la définition des opérateurs communs et ne traîtent pas à la gestion de ces opérateurs dans un contexte multistandards. En effet, face à la diversité des traitements numériques et à l'hétérogénéité incontournable des ressources pour répondre efficacement aux besoins de traitement, il est nécessaire d'offrir une architecture de management des opérateurs communs répondant à la fois aux besoins de changement de contextes applicatifs et capable d'exploiter la reconfigurabilité offerte par une plateforme hétérogène des opérateurs. Donc nous pouvons résumer la problématique abordée dans cette thèse par la définition

2. Multiply-ACcumulate

des techniques et mécanismes de management des opérateurs communs dans un contexte multistandards et l'amélioration du jeu d'opérateurs existant.

Contributions

Les contributions de ces travaux s'articulent, autour du management des opérateurs communs, sur trois thèmes complémentaires :

- Définition du management des opérateurs communs.
- Proposition d'une nouvelle structure matérielle d'un opérateur pour les algorithmes de FFT et Viterbi.
- Définition des mécanismes de management pour l'opérateur FFT/Viterbi en implémmenant des architectures communes aux deux algorithmes sur FPGA.

Les travaux décrits dans ce manuscrit proposent un modèle fonctionnel qui a pour objectif de permettre le déploiement des applications reconfigurables sur une plateforme matérielle hétérogène basée sur les opérateurs communs. Nous définissons donc le management des opérateurs communs et nous étudions leurs implémentations en se basant essentiellement sur des évaluations de complexité pour quelques standards utilisant la modulation OFDM. Nous montrons que les algorithmes de FFT et décodage de canal sont des traitements lourds en terme de calcul. Donc, nous nous concentrons sur la définition d'une technique de gestion efficace des ressources entre ces familles d'algorithmes en définissant une nouvelle structure matérielle d'un opérateur FFT/Viterbi. Nous proposons ainsi une architecture d'un processeur commun permettant la gestion efficace des ressources matérielles entre les algorithmes FFT et Viterbi. L'architecture que nous proposons permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permet l'optimisation de l'utilisation des ressources matérielles. Il en résulte une meilleure gestion de la consommation énergétique.
Les travaux présentés dans cette thèse ont fait l'objet de plusieurs publications scientifiques et deux brevets listés à la fin de ce manuscrit.

Environnement de recherche

Les travaux de recherche présentés dans cette thèse ont été réalisés dans les locaux du CEA-LETI[3] à Grenoble en collaboration avec deux laboratoires de recherche universitaires : GRESCOM[4] à SUP'Com Tunis et l'IETR-SCEE[5] à Supélec campus de Rennes. Ces travaux, notamment ceux autour de la définition des architectures communes FFT/Viterbi, sont engagés dans

3. http://www-leti.cea.fr
4. http://www.supcom.mincom.tn/
5. http://www.rennes.supelec.fr/ren/rd/scee/

des projets européens dont le réseau d'excellence NEWCOM++ [6] et le projet collaboratif C2POWER [7].

Plan du mémoire

Ce document se décompose en quatre chapitres (un chapitre introductif suivi de trois chapitres qui prsentent nos contributions). Ces chapitres se répartissent comme suit :

Le premier chapitre donne au lecteur un aperçu sur le contexte de notre travail de recherche ce qui permet de situer de manière technique les choix et motivations de nos travaux. Nous présentons d'abord le contexte de la radio reconfigurable avec les plateformes utilisées et les contraintes des standards. Ensuite, nous décrivons l'approche de paramétrisation et la technique des opérateurs communs pour les architectures reconfigurables. A la fin de ce chapitre, nous résumons les travaux antérieurs à nos recherches en cataloguant un premier jeu d'opérateurs communs que nous avons classés par leurs niveaux de granularité.

Le deuxième chapitre définit un modèle fonctionnel abstrait qui a pour objectif de permettre le déploiement des fonctionnalités reconfigurables sur une plateforme matérielle hétérogène basée sur les opérateurs communs présentés dans le premier chapitre. Dans une première section nous définissons d'une façon générale le management des opérateurs communs. Nous détaillons ensuite les techniques de management des opérateurs communs en se basant sur des approches de la littérature traitant de la gestion de ressources matérielles reconfigurables. Ensuite, nous étudions l'implémentation de ces opérateurs en se basant essentiellement sur des évaluations de complexités sur quatre standards considérés. A la fin de ce chapitre, nous proposons des modèles d'architectures permettant le management des opérateurs communs.

Le troisième chapitre propose des cellules de traitement matériel reconfigurables pour les algorithmes FFT et Viterbi, capables de prendre en compte leurs différences fonctionnelles et de s'adapter à l'utilisation qui en est faite. Ces travaux sur l'opérateur FFT/Viterbi viendront compléter le jeu d'opérateurs présenté dans le premier chapitre. Dans ce chapitre, nous commençons par étudier les similarités structurelles entre les algorithmes FFT et Viterbi. Ensuite nous présentons deux premières architectures parallèles du papillon commun pour proposer une architecture en pipeline plus efficace que les deux premières. Nous discutons à la fin de ce chapitre les comparaisons entre les cellules de l'opérateur FFT/Viterbi proposé avec celles de la

6. `http://www.newcom-project.eu/`
7. `http://www.ict-c2power.eu/`

littérature.

Le quatrième chapitre se concentre sur les mécanismes de gestion de l'opérateur FFT/Viterbi défini dans le chapitre précèdent. Dans ce chapitre, nous proposons des architectures de traitement matériel reconfigurables pour les algorithmes FFT et Viterbi, capables de partager efficacement les ressources matérielles entre les deux algorithmes. Notre objectif est d'assurer une affectation efficace des ressources entre les deux algorithmes FFT et de décodage de Viterbi. Ainsi, nous commençons par l'exploitation des similitudes structurelles entre la FFT et algorithme de Viterbi (présentés dans le troisième chapitre) et nous proposons une implémentation efficace et flexible du processeur commun FFT/Viterbi.

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 l'utilisation de l'opérateur FFT dans différentes fonctions de traitement de signal et la définition des termes et concepts utilisés dans ce manuscrit en relation avec les systèmes radio avancés.

Chapitre 1

Technique des opérateurs communs pour la radio reconfigurable

Sommaire

1.1 Introduction 27
1.2 Contexte de la radio reconfigurable 28
1.3 Les plateformes de la radio reconfigurable 30
1.3.1 Les contraintes des standards 31
1.3.2 Plateforme de traitement parallèle 32
1.3.3 Gestion des ressources 34
1.4 Réutilisation des ressources 34
1.5 La paramétrisation 37
1.5.1 Approche par fonctions communes 38
1.5.2 Approche par opérateurs communs 41
1.6 Identification des opérateurs communs 41
1.6.1 Approche théorique 42
1.6.2 Approche pragmatique 46
1.7 Un premier jeu d'opérateurs communs 46
1.7.1 Les opérateurs de faible granularité : LFSR et Cordic 47
1.7.2 Les opérateurs à forte granularité : FFT et FEC . 49
1.8 Conclusion 51

1.1 Introduction

Les techniques radio reconfigurable visent à offrir un accès à un large choix d'applications et de standards de radiocommunications sur une architecture matérielle unique. Une grande flexibilité du système reconfigurable

est nécessaire afin de répondre à la diversité des traitements à exécuter. Les techniques de communications numériques employées dans les standards entraînent des besoins en ressources de traitements hétérogènes. La reconfigurabilité d'une plate-forme d'exécution hétérogène est donc un point clef technologique à l'apparition de systèmes radio reconfigurables. La diversité des traitements induit aussi une variété de configurations de l'architecture d'un équipement multistandard.
Nos travaux de recherche portent sur la définition d'un terminal multistandard reconfigurable permettant une gestion efficace des ressources matérielles. En effet, la reconfigurabilité est dépendante des contraintes (temporelles, de consommation, de coût,...) imposées par les standards de communication et devient donc tributaire des capacités de la plateforme. Dans ce contexte, nous présentons dans ce chapitre les techniques de paramétrisation comme solution à cette problématique. Cette technique offre la possibilité de reconfigurer en temps réel les éléments de la plateforme reconfigurable et de garantir l'aspect évolutif de l'implémentation. Nous orientons nos travaux dans la définition et mise en application des techniques de paramétrisation. Pour atteindre cet objectif, nous considérons l'idée d'identifier des éléments communs entre différents standards. Les travaux antérieurs à nos recherches ont définis un premier jeu d'opérateurs communs que nous présenterons à la fin de ce chapitre.

1.2 Contexte de la radio reconfigurable

Les équipements de communication actuels doivent répondre à des besoins croissants en termes de flexibilité et d'adaptabilité à plusieurs standards. Ces besoins proviennent non seulement de la prolifération des applications mais aussi des différences de standardisation entre zones géographiques. En effet, les terminaux cellulaires actuels doivent pouvoir gérer plusieurs standards à la fois tel que le GSM, EDGE, l'UMTS et la 3GPP LTE. Outre ces réseaux, un terminal cellulaire doit pouvoir se connecter à des réseaux locaux tels que le Wifi et le Bluetooth, des services de localisation, et recevoir la télévision et radio numériques. Ainsi, les différences entre les standards de communication et l'incompatibilité entre nouveaux et anciens standards posent des problèmes de flexibilité et d'adaptabilité dès la conception des équipements. Ce besoin de s'adapter à différents standards et services a été défini sous le nom de « Handover Vertical », par complémentarité avec le « Handover Horizontal » qui qualifie la mobilité géographique d'une cellule à l'autre [13] (Figure 1.1).
La conséquence de la prolifération des standards de communication est le coût croissant des circuits en termes de surface de silicium et de consommation électrique. La complexité des processeurs augmente bien plus vite que la capacité des batteries, fait qu'il devient de plus en plus nécessaire

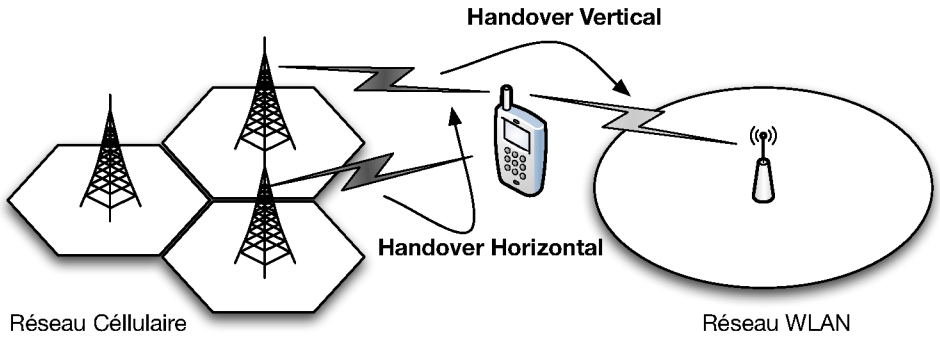


FIGURE 1.1 – Différents types de « Handover »

de changer l'approche de conception des équipements multistandards [14]. L'approche classique pour implémenter les fonctionnalités d'un équipement multistandard est d'instancier plusieurs chaînes d'émetteurs-récepteurs, où chaque chaine instanciée est dédiée à un seul mode ou standard (Figure 1.2). Avec cette approche, une grande partie du matériel doit être repensée pour chaque évolution des standards ou des applications. Cette approche conventionnelle appelée « Velcro » n'exploite pas les points communs entre les différents standards de communication [4]. Ainsi, un équipement radio ne peut réaliser qu'un nombre donné de standards. L'implémentation est limitée par le coût des circuits dédiés, concaténés dans la même architecture. Cette approche classique utilise un ensemble de composants spécialisés à chaque type de traitement, dupliqués pour chaque standard que doit exécuter l'équipement multistandard. Ces architectures de traitement classiques sont figées dès la conception et il est difficile de les faire évoluer pour supporter des nouvelles fonctionnalités.

Le concept de la radio reconfigurable vient contourner ce problème en exploitant les capacités des réseaux et des équipements à l'auto-adaptation à un environnement dynamique, avec l'objectif d'améliorer la qualité du service, l'efficacité de l'utilisation des ressources matérielles et spectrales. La radio reconfigurable se base essentiellement sur l'idée de la Radio Logicielle

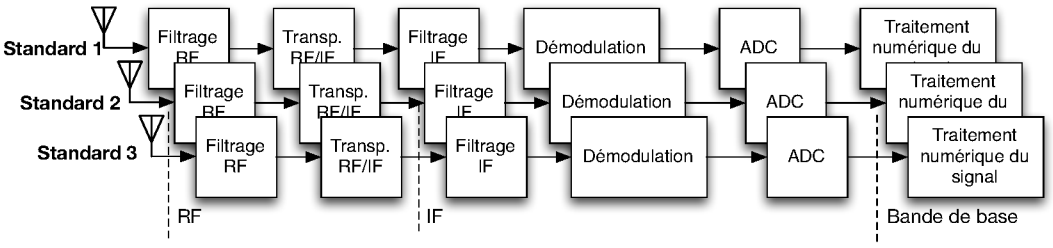


FIGURE 1.2 – Exemple d'une implémentation classique d'un récepteur tri-standards (Approche Velcro)

(RL) introduite par Mitola [3] qui est restée historiquement associée à des implémentations logicielles (voir Annexe A). Le principe de la numérisation du signal au plus près de l'antenne permet le traitement des opérations (filtrage, décimation, décodage etc.) en numérique, ce qui ajoute beaucoup de flexibilité aux architectures de traitement. Ainsi, il serait possible de traiter un grand nombre de standards de communication par simple chargement des paramètres associés au standard.
De plus, la Radio Intelligente (RI), définie par Mitola également [15], répond aux besoins des équipements d'effectuer en permanence un Handover vertical et de passer automatiquement d'un standard à un autre sans interrompre la communication. La radio intelligente est un concept qu'on pourrait définir comme une interface radio avec des capacités d'apprentissage, c'est à dire une radio en mesure de connaitre son environnement et d'ajuster ses paramètres et mécanismes de fonctionnement en conséquence. Il est clair qu'une telle approche offre beaucoup d'avantages en termes d'efficacité spectrale ce qui est très important lorsque les ressources spectrales sont rares, comme c'est le cas aujourd'hui.
Ainsi, il existe un grand intérêt dans les technologies de la radio reconfigurable qui a pour objectif de répondre aux besoins de flexibilité et d'adaptabilité nécessaires pour la RI. L'équipement reconfigurable dispose de ressources matérielles génériques et reconfigurables par un logiciel qui permet de s'adapter aux besoins des traitements de chaque standard. L'évolution et l'ajout de nouvelles fonctionnalités se fait par modification ou ajout de nouveaux logiciels permettant de reconfigurer les paramètres des blocs de traitements.

1.3 Les plateformes de la radio reconfigurable

Historiquement, les circuits utilisés en bande de base sont les circuits spécialisés de type Application-Specific Integrated Circuit (ASIC), les processeurs de type General Purpose Preprocessor (GPP) et Digital Signal Processor (DSP). Les ASIC offrent de hautes performances de traitement (nécessaires pour les besoins des standards récents) mais leur point faible est la flexibilité qui est une caractéristique indispensable dans la définition de la radio reconfigurable. En revanche, la programmabilité des circuits de type processeurs en font les cibles les plus adaptés à la reconfiguration. Mais cette flexibilité a un coût, notamment en consommation, et les performances de ces processeurs sont souvent insuffisantes face aux besoins de traitement du signal des applications récentes. Le manque d'efficacité énergétique par rapport aux performances est par ailleurs un obstacle pour l'intégration de ce type de circuits dans des terminaux mobiles.
Pour réaliser un compromis entre la flexibilité et la performance, il existe des solutions reconfigurables pour les applications de traitement du signal qui offrent les capacités de faire des changements structurels d'éléments de cal-

culs câblés. Comme le montre la figure 1.3, les circuits de type ASIP, FPGA et Coarse Grain peuvent remplir les compromis flexibilité/performance attendus par la radio reconfigurable. Par contre, le choix de ces éléments de calcul n'est pas toujours évident. En fait, le bilan de puissance très serré des équipements actuels fait souvent pencher la balance du coté performance plutôt que celui de la flexibilité.

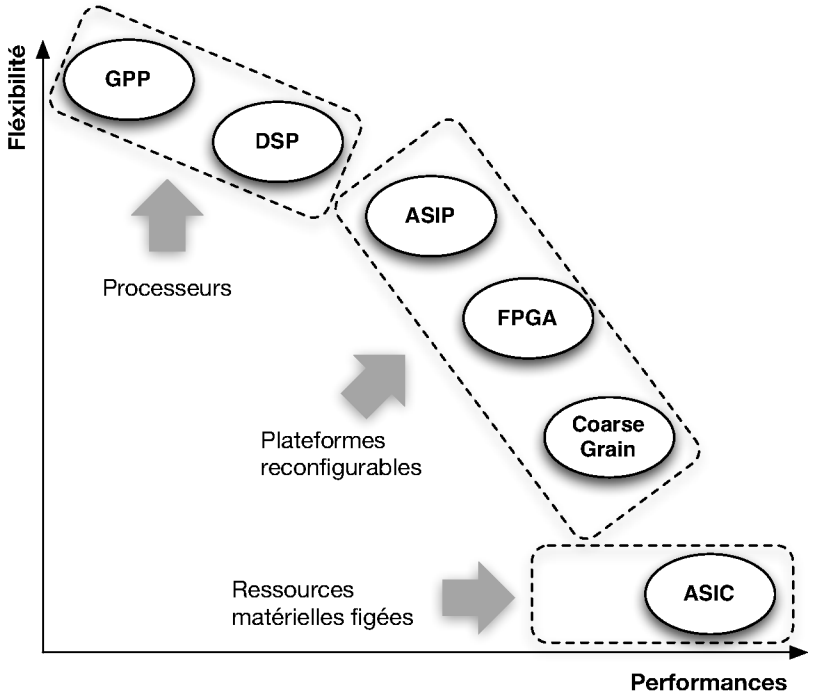


FIGURE 1.3 – Rapport flexibilité/performance des circuits de traitement en bande de base

1.3.1 Les contraintes des standards

Les plateformes matérielles de la radio reconfigurable suivent essentiellement les contraintes des standards et l'évolution de l'offre des circuits. Dans ce paragraphe, nous donnons un aperçu de ces contraintes et nous présentons quelques architectures et circuits adaptés aux traitements numériques en bande de base dans le contexte de la radio reconfigurable.
L'évolution des standards de communication entraine de forts besoins de calculs [16]. Comme le montre le tableau 1.1 l'évolution des besoins des calculs des nouveaux standards est plus rapide que l'évolution de la technologie surtout aux niveaux des puissances de calculs et des capacités des batteries. En effet, les capacités des batteries évoluent lentement (de l'ordre de 1Wh tous les 5 ans) par rapport à l'évolution des charges de calculs ($\times$10 tous les 5

ans). Ces charges de calculs, qui dépassent les 1000 GOPS (Giga Opérations Par Seconde) pour la 4e génération des terminaux cellulaires, seront de plus en plus difficiles à satisfaire surtout pour les terminaux mobiles qui fonctionnent sous des contraintes de consommation très strictes.

TABLE 1.1 – Evolution des besoins des standards (adapté de [16])

Année	1995	2000	2005	2010	2015
Génération	2G	2.5-3G	3.5G	pre-4G	4G
Standard	GSM	GPRS-UMTS	HSPA	HSPA-LTE	LTE/LTE-A
Débit [Mb/s]	0.01	0.1	1	10	100
Capacité des batteries [Wh]	1	2	3	4	5
Charge de calcul [GOPS]	0.1	1	10	100	1000
#Cœurs programmables	1	2	4	8	16

Le tableau 1.1 montre que la tendance actuelle est d'augmenter le nombre des cœurs programmables dans les terminaux mobiles pour faire face aux contraintes de la consommation et des charges de calculs. Dans ce contexte, les technologies de traitement parallèle (Parallel processing) couvrent une multitude d'approches différentes.

1.3.2 Exemple d'une plateforme de traitement parallèle reconfigurable

Il existe une grande variété de technologies de traitement parallèle reconfigurables qui vont des structures à faible granularité (reconfiguration niveau bit comme dans les FPGA) jusqu'aux structures à gros grain (reconfiguration niveau processeur comme dans les MPSoC). Ces structures à gros grain ont émergé ces dernières années grâce aux réseaux sur puce (Network on Chip NoC) qui ont permis le traitement parallèle à travers un réseau de processeurs reconfigurables.
Dans ce contexte, plusieurs projets sur ces technologies reconfigurables ont été réalisés au CEA-LETI dans la conception des NoCs pour les applications télécoms. Parmi ces projets, nous citons le projet MAGALI qui est un circuit

développé pour les terminaux cellulaires de 4e génération [17]. Il est basé sur une architecture innovante de type réseau sur puce asynchrone (NoC) supportant des liaisons à $2.2GB/s$. Le circuit embarque 23 processeurs intégrés dédiés au traitement du signal et au traitement au niveau digit. Il contient un processeur intégré ARM1176 supportant les piles protocolaires et plusieurs accélérateurs matériels dédiés aux traitements en bande de base pour la 4e génération des réseaux sans-fil (Comme des modules OFDM, UWB, Wiflex dans la Figure 1.4) [18].
L'innovation majeure dans ce projet porte sur la reconfiguration. Le circuit bande de base peut être entièrement reconfiguré en moins de $50\mu s$ (temps maximum). Lors de tests applicatifs, on observe une reconfiguration en $4\mu s$ en moyenne. De plus, le circuit est capable de supporter plusieurs applications en parallèle. En effet, la puissance de calcul peut être partagée sur les mêmes unités de calcul entre plusieurs applications radio. Ces deux caractéristiques font de MAGALI un circuit particulièrement adapté aux applications de radio reconfigurable. Grâce à l'architecture réseau sur puce asynchrone, chacun des 23 processeurs est un ilot de fréquence programmable dynamiquement. On obtient ainsi le rapport optimal entre la consommation et les performances. Au final, le circuit MAGALI a une consommation inférieure à $500mW$ pour des performances atteignant les 40 GOPS (Giga Operations Per Second).

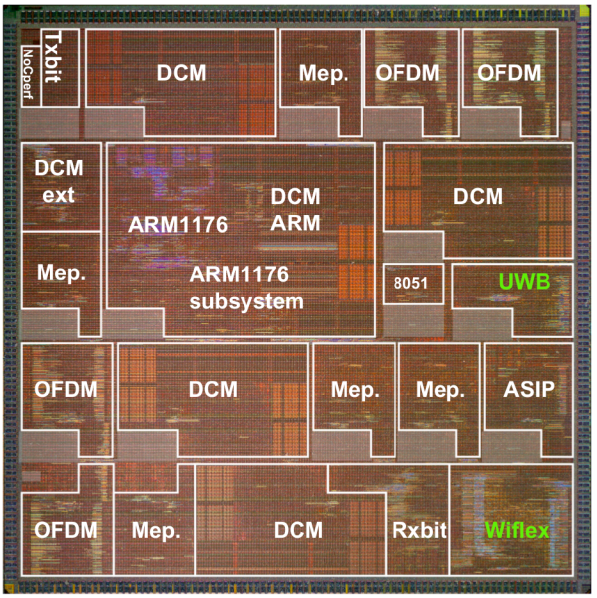


FIGURE 1.4 – Le circuit MAGALI [17]

1.3.3 Gestion des ressources dans les plateformes reconfigurables

Les ressources offertes par les plateformes matérielles nécessitent une gestion efficace afin de respecter les contraintes des standards de points de vue des délais et de la consommation. L'aspect reconfigurable et l'utilisation multistandard ajoutent plus de complexité à la gestion de ressources. En effet, chaque élément de traitement de la plateforme peut être reconfiguré et partagé entre plusieurs standards. Cela nécessite donc une décomposition efficace des fonctions de traitement en bande de base afin d'assurer une bonne réutilisation des ressources matérielles. La figure 1.5 montre un exemple de partage des ressources entre trois standards.
Nous présentons par la suite comment les ressources des plateformes de la radio reconfigurable peuvent être réutilisé efficacement dans les terminaux multistandards.

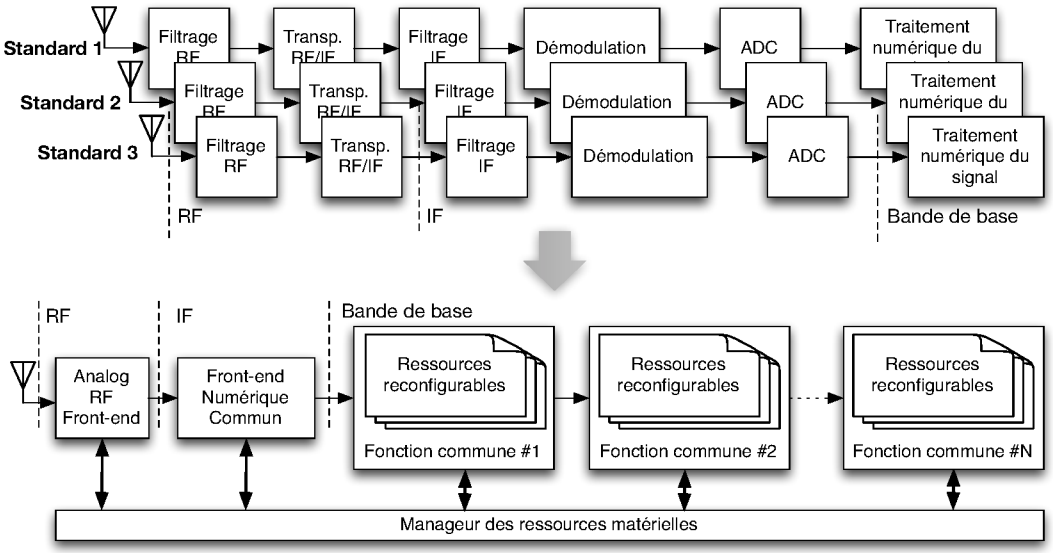


FIGURE 1.5 – Exemple de partage des ressources entre trois standards dans une plateforme radio reconfigurable

1.4 Réutilisation des ressources dans les terminaux multistandards

Avec les limites technologiques actuelles, il est possible de répondre aux besoins des standards en utilisant des architectures à multiprocesseurs comme nous l'avons montré dans les paragraphes précédents. Mais aussi, il est nécessaire d'utiliser une stratégie de gestion des ressources offertes par les processeurs

afin d'optimiser la performance et la consommation du système. Parmi les stratégies à considérer, l'utilisation de l'allocation dynamique des ressources de traitement hétérogènes qui touche directement les caractéristiques de flexibilités (objectif de notre travaux de recherche). L'allocation dynamique des ressources dans ce contexte d'étude peut être appliquée à plusieurs niveaux :

- La réutilisation des ressources matérielles et logicielles (associées) réduit la quantité du matériel physiquement implémenté et donc réduit la surface du circuit,
- La réutilisation des noyaux logiciels (kernels) entre plusieurs standards peut réduire les besoins de mémoire et spécialement la mémoire du programme,
- La réutilisation du matériel et du logiciel developpés entre différents projets réduit considérablement les délais et les coûts de développement.

Concrètement, l'allocation dynamique peut devenir très intéressante dans les équipements multistandard et/ou multimodes puisque le basculement d'un standard/mode à un autre peut libérer des ressources et en occuper d'autres. Par exemple, les variations de l'état du canal de transmission peuvent provoquer un grand changement dans les ressources utilisées. En effet, pendant les conditions de fading sévère, des algorithmes de compensation et de codage canal sophistiqués et gourmands en ressources sont utilisés pour maintenir une communication fiable. En revanche, pendant les bonnes conditions de propagation, plus de ressources sont allouées aux traitements niveau symboles dans le but d'augmenter le débit. Cela suppose donc que les ressources partagées ne sont pas trop spécialisées et sont suffisamment génériques pour s'acquitter de plusieurs tâches différentes.
L'optimisation de la gestion des ressources de la plateforme radio reconfigurable nécessite alors l'étude des chaines de traitement de signal et l'identification des éléments de traitements qui peuvent être partagés entre les standards et les fonctions.
La plupart des systèmes radio contiennent deux chaines de traitement en bande de base, la chaine de réception et la chaîne d'émission. D'une façon générale dans la chaine d'émission, le processeur de traitement Bande de Base reçoit les données de la couche MAC et effectue principalement les fonctions suivantes :

- Codage canal
- Modulation numérique
- Filtrage de mise en forme

A la réception, les données envoyées par l'ADC sont traitées par les fonctions suivantes :

- Filtrage, synchronisation, décimation et contrôle de gain
- Démodulation, estimation du canal et égalisation
- Décodage et correction des erreurs

Ces principales fonctions de traitement de signal en Bande de Base sont illustrées dans la figure 1.6.

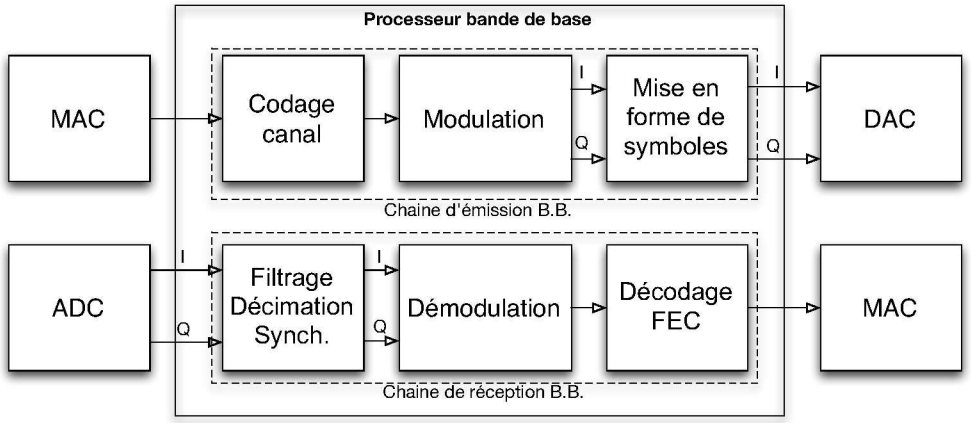


FIGURE 1.6 – Les fonctions réalisées par un processeur de traitement bande de base

Il existe des similarités entre les fonctions bande de base surtout entre fonctions des standards utilisant la modulation multi-porteuse. En se focalisant sur les standards basés sur l'OFDM (Orthogonal Frequency-Division Multiplexing) qui regroupent un grand nombre des standards de télécommunication actuels, on peut constater que les algorithmes utilisés et la dépendance de données à travers les chaines de traitement sont très proches. Nous proposons de tirer profit de ces similarités entre standards et fonctions afin de gagner en flexibilité et en complexité avec la réutilisation des ressources.
Dans l'optique de tirer profit de ces similarités entre les algorithmes, Alaus a proposé [10] une classification des fonctions de traitement de signal en fonction des types de structure qu'elles mettent en œuvre et relativement aux données qu'elles traitent. Cette classification figure 1.7 repose non pas sur le rôle des fonctions dans la chaine de traitement mais sur les structures qui les constituent :

- Les fonctions utilisant les opérations de multiplications et d'additions à l'instar de la FFT, des FIRs, de la corrélation...
- Les fonctions réalisant des opérations de décalages et des opérations logiques telles que le codage convolutionel, le turbo codage...

- Les fonctions réalisant des opérations sur la localisation des bits comme « l'interleaving » ou le « puncturing ».
- Les fonctions réalisant des associations de données comme les différentes modulations.

Cette classification a permis l'identification et le choix des fonction qui peuvent être mises en communs dans les chaines de traitement en bande de base [10]. Dans les sections suivantes nous définissons les approches considérées pour le partage des ressources matérielles dans les terminaux multistandards.

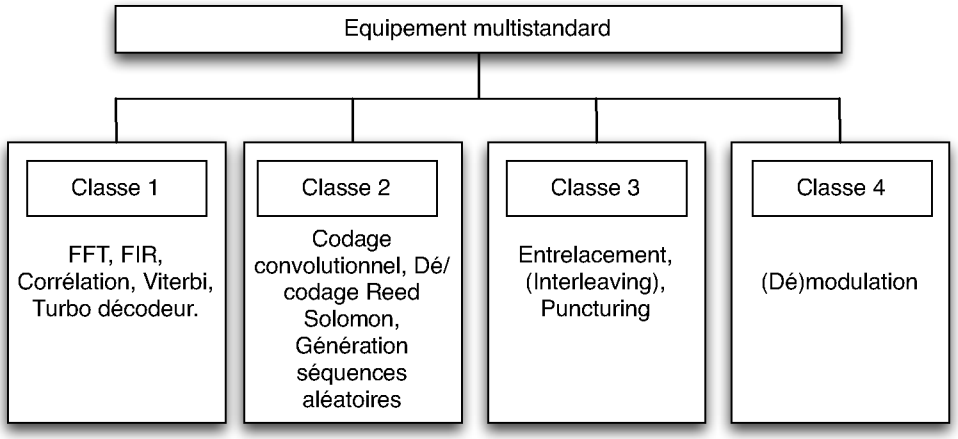


FIGURE 1.7 – Classification des traitements communs proposée par Alaus [10]

1.5 La paramétrisation

Dans cette section, nous présentons une approche appelée paramétrisation utilisée dans le cadre de la radio reconfigurable permettant d'exploiter les similarités entre les algorithmes et les standards comme nous l'avons évoqué dans la section précédente. Dans un premier temps, nous précisons que le terme paramétrisation n'existe pas dans la langue française. En réalité, ce terme est dérivé du terme anglais « Parameterization technique » qui est avant tout un concept mathématique qui a pour objectif d'identifier les degrés de liberté d'un système. De plus dans le domaine de la radio reconfigurable, plusieurs approches de paramétrisation ont été considérées dans la littérature [15] et [5]. Dans nos travaux, nous définissons la paramétrisation comme une technique permettant de mettre en commun les traitements les plus utilisés parmi un nombre déterminé de standards, fonctions de traitement de signal ou opérateurs mathématique. En effet, un équipement multistandard devra

basculer entre différents standards en se basant sur une plateforme composée essentiellement d'éléments de traitements communs. Ainsi, cette technique définit des blocs de traitement communs génériques qui restent figés après la conception mais qui s'adaptent à un ensemble de fonctionnalités grâce à un simple jeu de paramètres, d'ou le nom de paramétrisation. Cette approche permettrait donc de diminuer le nombre d'éléments à implémenter et constitue une méthode de reconfiguration en elle-même.
Nous distinguons deux approches de paramétrisation, émergées de la littérature : l'approche par fonctions communes et l'approche par opérateurs communs.

1.5.1 Approche par fonctions communes

Nous commençons par présenter le premier type de paramétrisation qui est l'approche par fonctions communes. Dans cette approche, l'analyse des similarités se fait au niveau fonction c'est à dire à un niveau de granularité élevé. L'objectif de cette approche est de définir des structures de traitements (macro-fonctions) paramétrables communes à plusieurs standards. Un jeu de paramètres est associé à chaque fonction permettant de déterminer son mode de fonctionnement.
Wiesler [6], [19] a défini des structures paramétrables pour les fonctions de modulation communes aux standards GSM, DECT, UMTS UTRA/FDD et IS-136 comme le montre l'exemple illustré sur la figure 1.8. D'autres travaux réalisés par la même équipe de recherche ont menés à la proposition d'une fonction commune de codage canal [20] ou dans [21] qui a proposé des implémentations multi-modes pour les terminaux 3G.
D'autres exemples de structures de fonctions communes sont décrites dans la littérature, comme l'architecture VITURBO [22] proposant une structure commune de turbo décodage et de Viterbi pour les systèmes 3G.
Delahaye a également étudié dans sa thèse une chaîne de transmission multistandards composée de fonctions communes [23]. Cette étude est basée, de façon volontaire sur des standards dont les caractéristiques sont très différentes. Dans la figure 1.9 il a proposé une factorisation fonctionnelle de trois chaînes de traitements (UTRA/FDD, 802.11g et GSM).
Cependant cette approche a plusieurs limites. Le premier inconvénient est que la structure à base de fonctions communes ne peut pas évoluer facilement et supporter de nouveaux standards. De plus, il y a toujours certains modules de traitement, non requis par les paramètres sélectionnés, qui peuvent rester inactifs au moment de l'exécution. Alors que des mécanismes de coupures d'alimentation sont facilement réalisables sur des fonctions entières, il est plus compliqué de couper l'alimentation sur des sous-blocs d'une même fonction. La structure consomme donc dans sa totalité, il en résulte une perte d'efficacité énergétique. Le coût en surface de telles structures peut aussi être significatif.

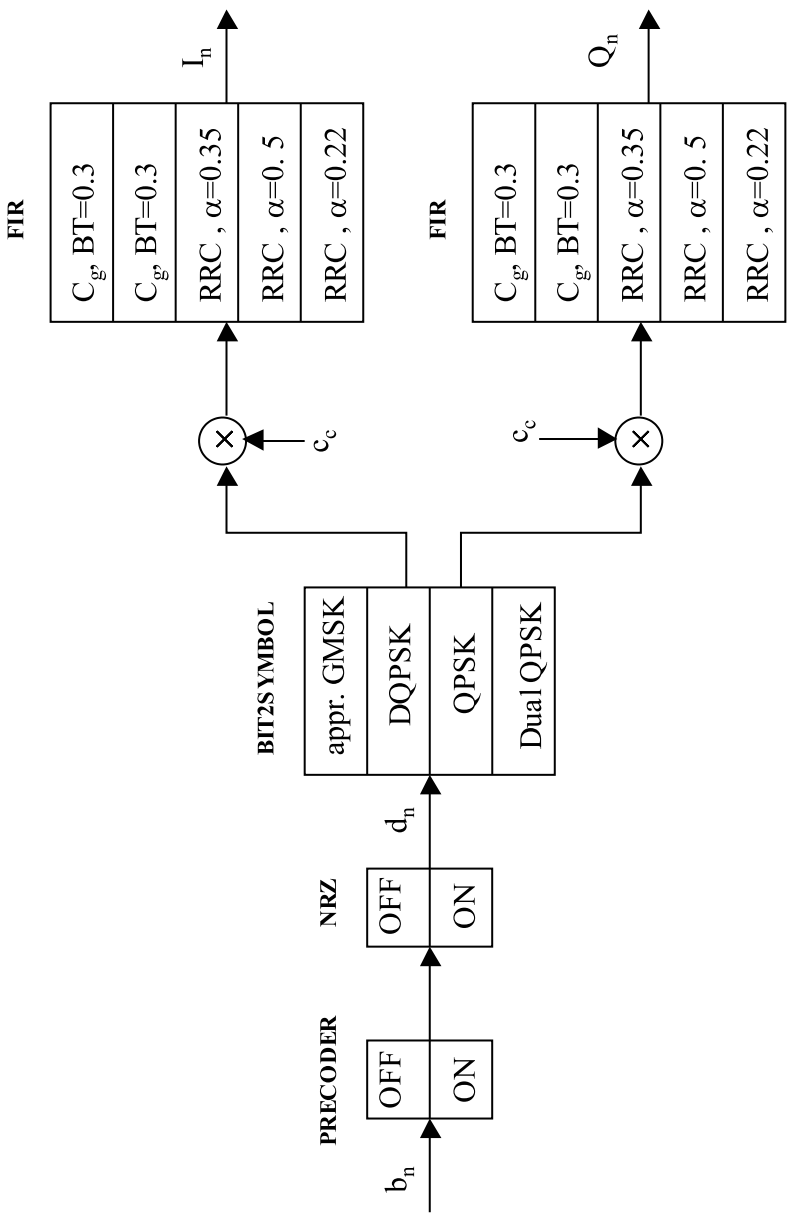


FIGURE 1.8 – Fonction de modulation paramétrable [19]

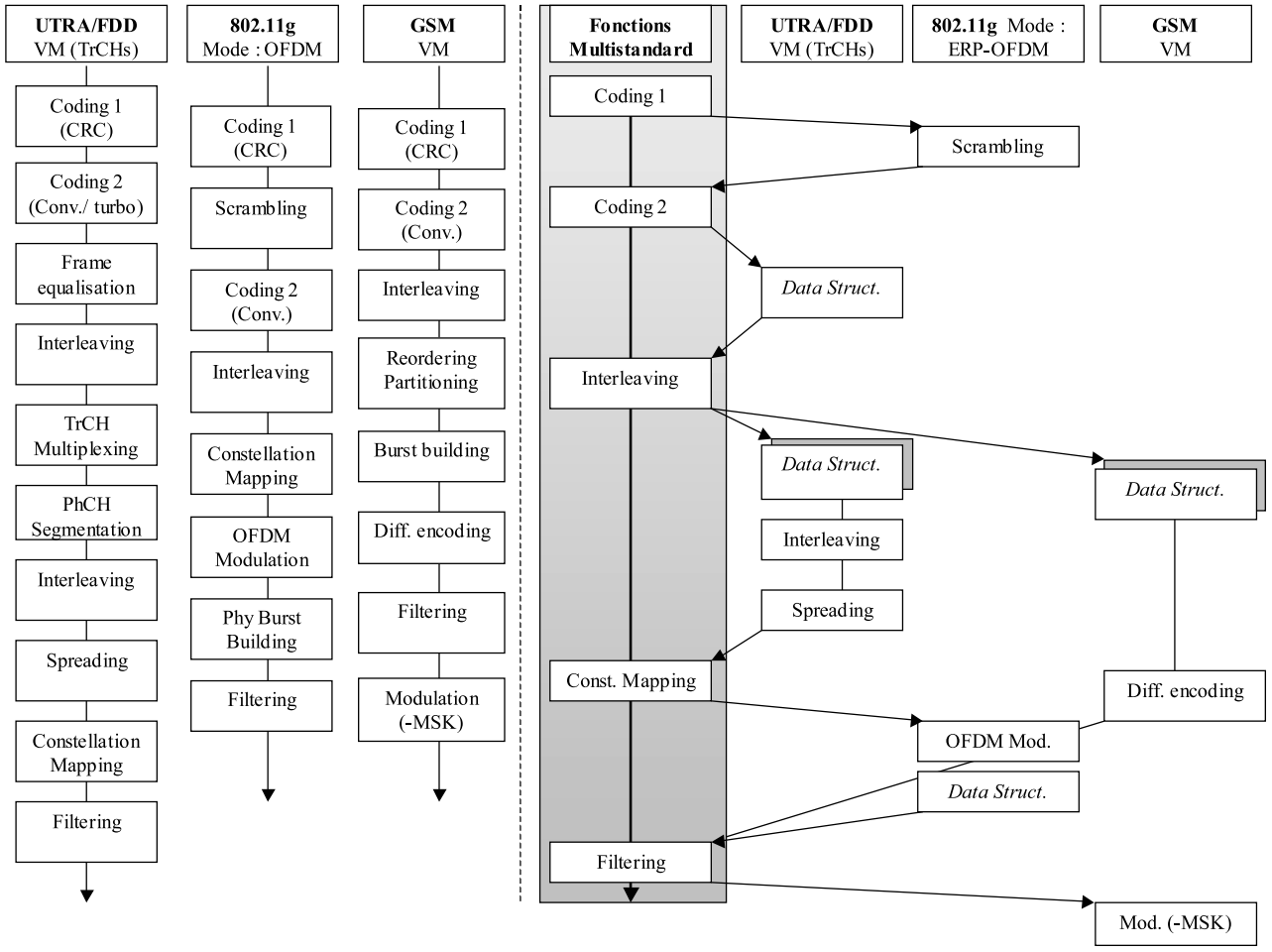


FIGURE 1.9 – Vers une chaîne de transmission multistandards unifiée [23]

De plus, dans ce type d'approches, toutes les informations sur les standards ciblés doivent être connues avant la conception de la structure commune. Il faut donc stocker un nombre limité de paramètres pour décrire plusieurs modes de fonctionnement.

1.5.2 Approche par opérateurs communs

L'approche par Opérateurs Communs (OC) est une deuxième approche de la paramétrisation qui consiste à identifier des éléments communs de granularité inférieure à celle de la fonction commune. Son niveau de granularité se situe à un niveau intermédiaire entre la fonction commune et un opérateur arithmétique de bas niveau (MAC par exemple). La conception d'un opérateur commun est fondée sur des aspects structurels et elle est effectuée indépendamment des normes. Ainsi, un opérateur commun est défini pour effectuer des opérations élémentaires de traitement du signal indépendamment de la fonction qui l'exécute. Par conséquent, cette approche vise principalement à concevoir des équipements évolutifs. En contraste avec la fonction commune, un opérateur commun n'est pas spécifique à un ensemble de standards uniques, il permet une conception plus souple s'adaptant à un large éventail de standards.
En effet, l'analyse de différentes chaînes de traitement de signal a montré que certaines structures de traitement sont utilisées dans plusieurs fonctions (ou plusieurs standards). Palicot [24] a introduit cette notion d'opérateur commun en radio reconfigurable en donnant l'exemple de l'opérateur FFT. Cet opérateur est réutilisé par plusieurs fonctions pouvant être réalisées dans le domaine fréquentiel. Alaus a montré dans [25] qu'il est intéressant d'implémenter les opérateurs communs dans un banc (appelé Banc d'Opérateurs Communs BOC) où les opérateurs peuvent être mappés et utilisé par un ensemble de standards comme le montre la figure 1.10.

1.6 Identification des opérateurs communs

Pour identifier les opérateurs communs, nous commençons par rappeler leur fonction principale qui est d'optimiser un équipement multistandard selon un ensemble de critères donnés (consommation, surface,...). Ainsi, les différentes fonctions de traitement de signal sont explorées afin d'identifier des opérateurs de moyenne granularité (entre les fonctions et les opérateurs arithmétiques) qui pourraient être réutilisées dans l'exécution plusieurs fonctions et/ou standards. L'exploration des fonctions ici se fait de point de vue opérationnel et non fonctionnel. En effet, nous nous intéressons aux ressources qui peuvent êtres partagées ce qui signifie que deux fonctions très différentes de point de vue fonctionnels tels que la modulation et le codage

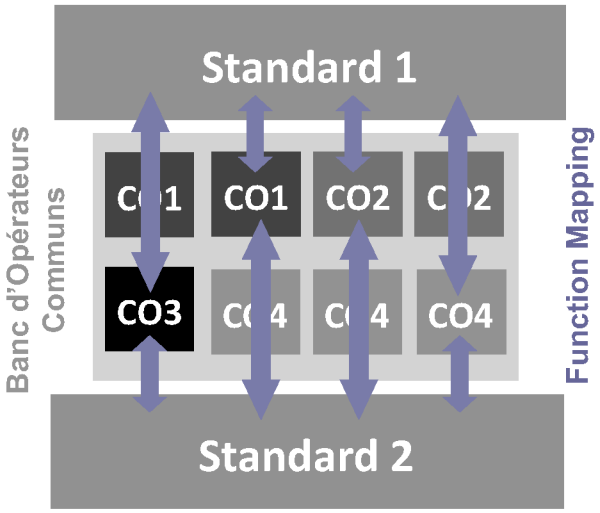


FIGURE 1.10 – Banc d'opérateurs communs

peuvent se partager des ressources de traitement communes.
Comme nous l'avons mentionné auparavant, l'opérateur commun est présenté avant tout dans un but d'optimisation du léquipement multistandard. Dans cette optique, Rodriguez et.al [26] ont illustré le concept de l'opérateur commun par une représentation mathématique graphique (figure 1.11) en définissent une première méthode d'optimisation permettant de déterminer le meilleur jeu d'éléments communs à considérer. Cette approche dite théorique se base donc sur une hiérarchisation en graphe permettant de décomposer des différents standards à considérer en opérations de base. Ce graphe illustré dans la figure 1.11 permet ainsi de faire apparaître même le niveau de granularité des éléments du graphe.
Mise a part cette approche théorique, il existe une approche pragmatique permettant l'identification des opérateurs communs. Nous détaillons ces deux approches dans les paragraphes suivants.

1.6.1 Approche théorique

La première approche de l'identification des opérateurs communs reprend la vision initiale décrite par initialement par Rodriguez en [26]. Cette approche consiste à définir un processus d'optimisation basé sur des coûts (consommation, surface, délais..) qui décompose un équipement multistandard en plusieurs opérateurs de base avec plusieurs niveaux de granularité. En effet, l'approche permet d'identifier des « briques de base » de traitement qui peuvent être factorisés et utilisés par plusieurs fonctions et standards à la fois. Avec une représentation en graphe, il est alors possible de sélectionner un jeu optimal d'opérateurs selon les contraintes considérées par l'optimisation.

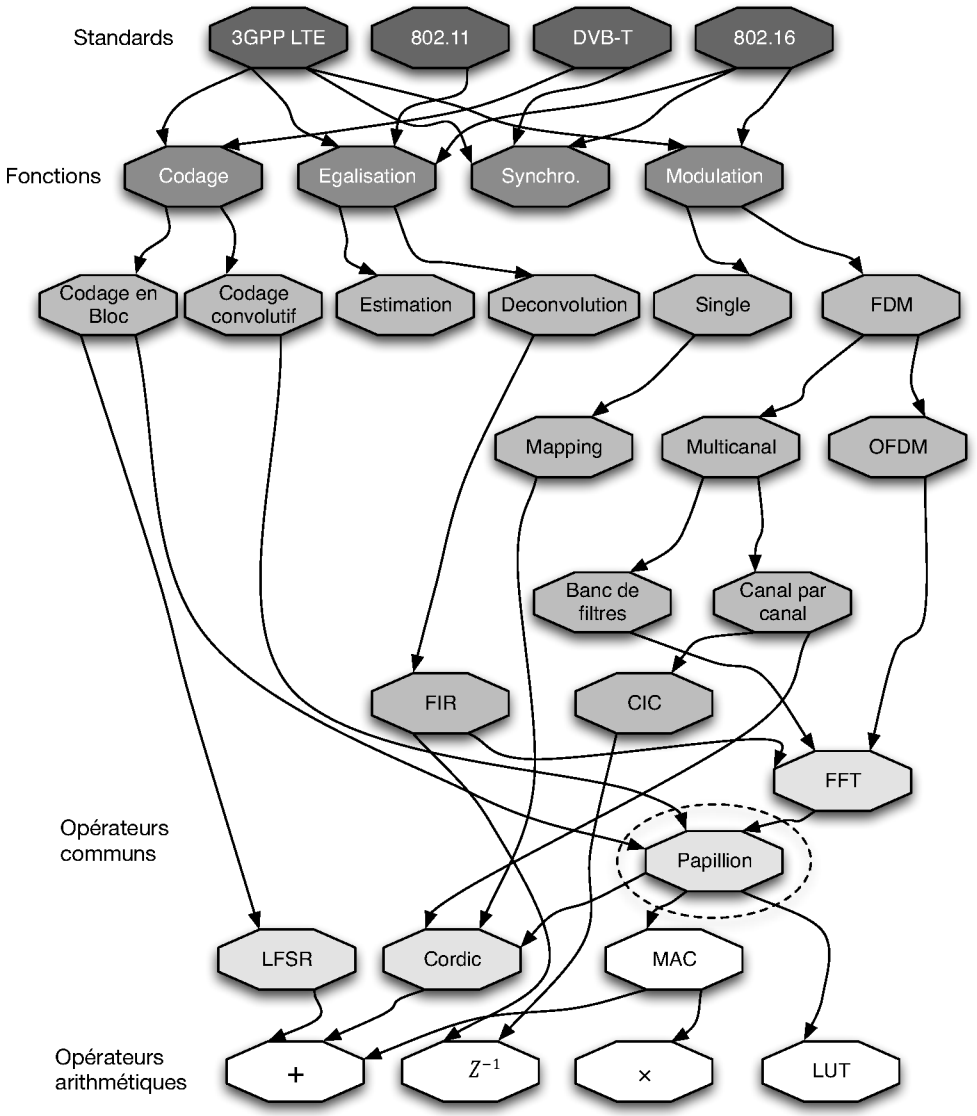


FIGURE 1.11 – Exemple de décomposition partielle en graphe d'un terminal multistandard permettant l'identification des opérateurs communs

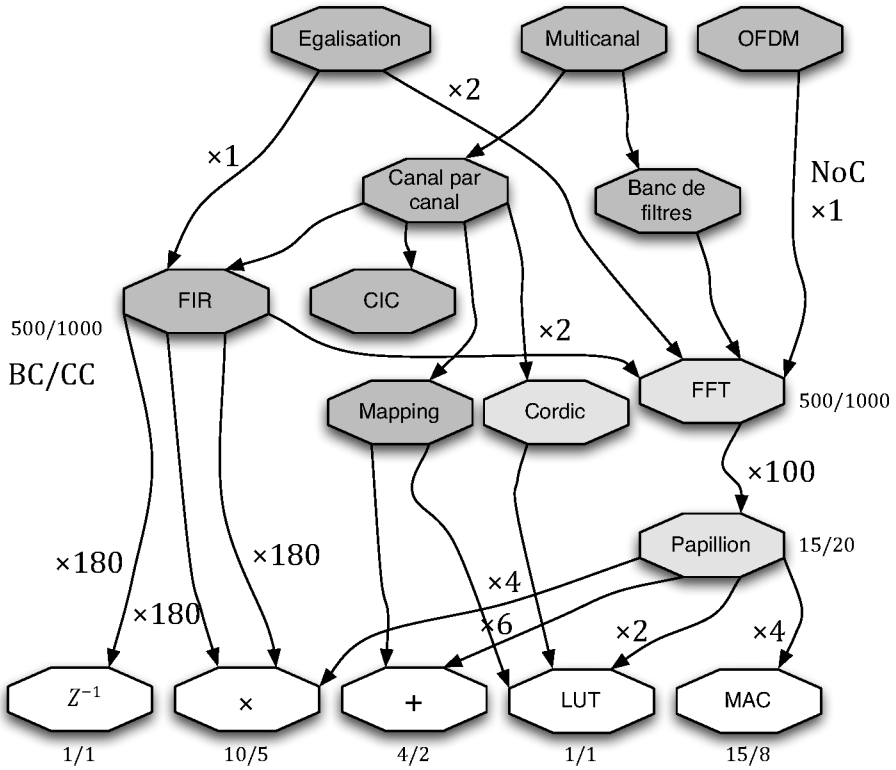


FIGURE 1.12 – Exemple de décomposition partielle en graphe avec les coûts associés (adapté de[11])

Ainsi, l'ensemble des fonctions réalisées par un équipement multistandard est représenté par un graphe comme celui illustré dans la figure 1.11.
Plusieurs exemples de cette approche ont été utilisés dans les travaux de Rodriguez [26] et Gul [11]. Un des exemples présenté est celui de l'utilisation de l'FFT comme opérateur commun pour plusieurs fonctions tels que : la (dé)modulation OFDM, filtrage et égalisation comme le montre la figure 1.12. Chaque sommet dans le graphe d'opérateurs communs représente un élément de traitement fonctionnel à un niveau de granularité donné par l'emplacement du sommet dans le graphe. Chaque élément de traitement peut soit être implémenté directement dans le système, ou décomposé en plusieurs éléments de niveaux de granularité inférieurs. Ainsi, cette approche permet de parcourir toutes les options possibles afin de sélectionner un jeu d'opérateurs pour une fonctionnalité donnée. Ce jeu d'opérateurs doit donc pouvoir offrir le meilleur compromis (décrit par une fonction de coût) entre flexibilité et efficacité .
La fonction de coût est une fonction qui calcule le choix de chaque jeu d'opérateurs. Elle a été initialement proposée par Gul [11] et développé par Kaiser [12]. Cette fonction de coût lorsqu'elle est optimisée permet-

tra de sélectionner le jeu d'opérateurs ayant le coût minimal (défini par le designer de l'équipement multistandard). Les paramètres de cette fonction de coût prennent en considération la notion de flexibilité et d'efficacité de l'implémentation.
Pour définir la fonction coût, Gul [11] a associé à chaque sommet du graphe (élément de traitement) un coût de fabrication (Building Cost, BC), un coût de calcul (Computational Cost, CC), et un nombre d'appels (Number of Calls, NoC) sur chaque branche.
Le premier coût BC correspond au coût payé pour implémenter un élément de traitement (un sommet dans le graphe). Il est « payé » une seule fois et sera donc indépendant du nombre d'appels de cet élément. La réutilisation des éléments de traitement fournit ainsi une réduction de ce coût. Le deuxième coût CC est associé au temps nécessaire pour exécuter une fonction donnée. Ce coût est calculé à chaque fois qu'un élément de traitement est appelé.
Le troisième coût NoC représente le nombre de fois que des les éléments de traitements sont appelés par un étage de granularité supérieur. Il s'agit d'un facteur multiplicatif associé aux arcs (figure 1.12).
Les paramètres introduits précédemment ont permis de définir une fonction de coût ayant deux objectifs : la réduction du coût de fabrication d'un équipement multistandard en mutualisant les traitements grâce aux opérateurs communs, tout en assurant des temps d'exécution conformes aux exigences des standards. Ces deux objectifs de la fonction de coût sont souvent en opposition puisqu'une réduction du coût de fabrication (surface, complexité) entraine une augmentation du temps du calcul et vice versa.
Ainsi le coût de total de fabrication (CTF) est exprimé par l'équation 1.1 :

$$CTF = \sum_{i} BC_i \times N_i \tag{1.1}$$

$N_i \in 0, 1$ indique si le *ieme* élément de traitement est présent dans l'équipement.
Le coût total de calcul (CTC) est évalué par l'équation 1.2 :

$$CTC = \sum_{n} \sum_{k} CC_k((S_n)_{n\in\{1,2,\ldots,N\}} \tag{1.2}$$

CTC représente ainsi le coût total de calcul où la première somme calcule le coût pour chaque standard et la deuxième somme calcule le coût total des N standards S_n.
L'optimisation du graphe d'opérateurs se ramènd alors á un problème d'optimisation multi-objectifs. L'approche de la « somme pondérée » [27] [28], a été considérée par Gul [11] pour définir la fonction de coût en raison de sa simplicité. Elle consiste à regrouper les fonctions d'optimisation CFT et CTC dans une seule fonction, avec des coefficients de pondération associés à chaque fonction. Ainsi, la fonction de coût totale bi-objectifs combine les équations 1.1 et 1.2 et est définie dans 1.3.

$$coût = \bar{\omega} \sum_i BC_i \times N_i + \sum_n \sum_k \omega_n CC_k((S_n)_{n\in\{1,2,\ldots,N\}} \qquad (1.3)$$

où $\bar{\omega}$ est le poids associé au coût de fabrication de l'équipement multistandard et ω_n est le poids est associé au standard S_n. Ce problème d'optimisation a été traité dans la thèse de Kaiser [12] en utilisant la théorie des graphes.
L'opérateur commun est donc vu ici comme une entité idéale qui peut être instanciée une seule fois et appelée à chaque fois qu'une fonction l'utilise. L'idée principale de cette approche est de permettre la définition de nouveaux operateurs à partir d'une combinaison d'operateurs déjà présents dans le graphe considéré et permettant d'optimiser le design. Il est donc nécessaire d'avoir une deuxième approche qui permettre de définir les opérateurs communs.

1.6.2 Approche pragmatique

La deuxième approche que nous considérons dans cette thèse est une approche qualifiée de pragmatique. En effet, elle aborde la technique des opérateurs communs d'une façon plus pratique. Avec cette approche, la définition des opérateurs communs se fait à partir d'une étude spécifique des contraintes des standards de manière à favoriser la réutilisation maximale des éléments de traitements.
Ainsi, l'approche pragmatique consiste d'abord à identifier dans la littérature les traitements similaires qui peuvent être mutualisés ou partager des ressources communes (tant au niveau algorithmique qu'architectural), et ensuite définir un opérateur générique qui devra alors réaliser les traitements ciblés. Cette approche a été l'origine de l'idée des opérateurs communs. En effet Palicot a défini dans [24] avec cette approche une architecture reconfigurable d'une transformée de Fourier rapide (FFT).
En se basant sur cette approche pragmatique, nous présentons un premier jeu d'opérateurs communs en se basant sur les travaux d'Al-Ghouwayel [8], Wang [9] et Alaus [10].

1.7 Un premier jeu d'opérateurs communs

Dans cette section, nous présentons un premier jeu d'opérateurs communs définis au sein de l'équipe SCEE à Supélec et au LETI. Ces opérateurs ont été identifiés avec l'approche pragmatique, et dans certains cas, le choix a été validé par la méthode des graphes et le calcul des coûts (approche théorique).

1.7.1 Les opérateurs de faible granularité : LFSR et Cordic

Le LFSR (LinearFeedback Shift Register) est un registre à décalage dont le bit d'entrée est une combinaison des contenus précédents du registre. Cela présente une caractéristique intéressante qu'on veut exploiter en faisant appel à une sous-structure du LFSR comme opérateur commun et ordonnancer ces appels. La figure 1.13 montre les algorithmes qu'on peut traiter avec le jeu d'opérateurs LFSR définis par Alaus [10]. En effet, l'opérateur LFSR peut être appelé essentiellement par les fonctions : codage/décodage CRC, Dé/Embrouilleur, codeur convolutif et Turbo codeur. La figure 1.13 présente aussi un jeu d'opérateurs LFSR définis par Alaus dans [10]. Quatre architectures de cet opérateur ont été développées afin de construire 16 opérateurs communs LFSR qui permettent de supporter plusieurs fonctions de traitement numérique du signal.
Les quatre architectures proposées sont basées sur les deux familles de LFSR, Fibonacci et Galois :

- RG-LFSR (Reconfigurable Galois LFSR),
- RF-LFSR (Reconfigurable Fibonacci LFSR),
- R-LFSR (Reconfigurable LFSR),
- ER-LFSR (Extended Reconfigurable LFSR).

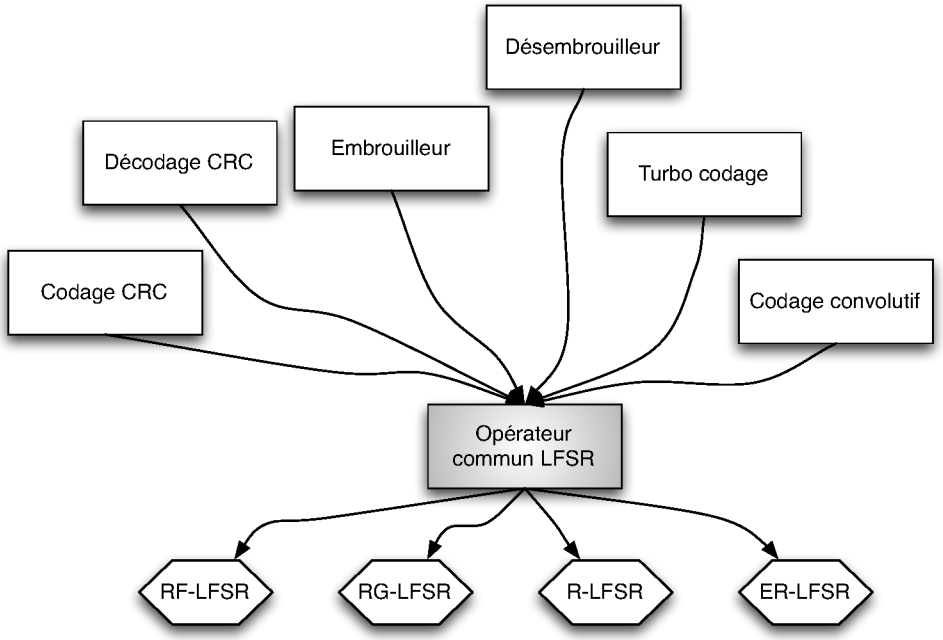


FIGURE 1.13 – Illustration de l'utilisation des opérateurs LFSR dans les fonctions de traitement de signal

Dans ce même contexte des opérateurs communs à faible granularité, Wang [9] a défini dans sa thèse une architecture reconfigurable de l'opérateur CORDIC basée sur des étages itératifs. L'algorithme COordinate Rotation DIgital Computer (CORDIC) a été initialement défini dans [29] afin d'effectuer des traitements tels que les rotations de vecteurs ou les changements de coordonnées cartésiennes/polaires (ou l'inverse) dans le plan euclidien. Cet algorithme fait appel seulement à des opérations arithmétiques en base 2 (additions, soustractions et décalages). Il permet aussi l'implémentation des traitements mathématiques tel que le calcul de racines carrées et les fonctions trigonométriques. Une généralisation de l'algorithme en a été proposée dans [30]. Une extension de l'algorithme à des espaces pseudo-euclidiens dotés d'une métrique hyperbolique ou linéaire a permis le calcul de la multiplication, de la division, des fonctions trigonométriques, ainsi que d'exponentielles. De plus par composition des fonctions précédentes, l'algorithme permet aussi de calculer les tangentes circulaires et hyperboliques et de logarithmes.
Ainsi, l'opérateur CORDIC défini par Wang [9] peut être utilisé par un grand nombre de fonctions avancées du traitement numérique du signal. On peut citer comme exemples de fonctions : La FFT, DFT [31] [32], le filtrage complexe de signaux, la modulation en bande latérale unique, le filtrage fréquentiel sur structures à faible sensibilité en bande passante (treillis de Gray-Markel, filtres orthogonaux et filtres d'onde [33] [34]), la factorisation spectrale de processus stationnaires par l'algorithme de Schur normalisé [35] [36], la modélisation adaptative de processus non stationnaires (filtrage récursif optimal au sens des moindres carrés sur structure en treillis normalisé [37], filtrage de Kalman [38]) . Plus généralement, l'opérateur CORDIC peut être utilisé dans l'implémentation de beaucoup de fonctions d'algèbre linéaire : résolution exacte ou approchée au sens des moindres carrés de systèmes linéaires, équations aux valeurs propres, décomposition en valeurs singulières [39], décomposition QR, de Cholesky [40] [41], le cas des matrices de Toeplitz ou quasi-Toeplitz (à rang de déplacement fini), donnant toujours lieu à des algorithmes spécifiques [42].
Ces opérateurs de faible granularité LFSR et CORDIC offrent de nombreux avantages d'intégration au niveau architectural, par leur excellente adaptation aux contraintes des VLSI. En effet, grâce à sa structure à base d'étages itératifs un module de base LFSR ou CORDIC, offre la possibilité de choisir différents degrés de parallélisme [43] pour une adaptation optimale aux débits. La figure 1.14 illustre un exemple d'étage itératif pour les opérateurs LFSR ou Cordic.
Dans ce contexte, Alaus a défini dans [25] une technique de gestion de ces opérateurs dans les architectures reconfigurables. Par exemple, l'opérateur commun utilisé dans un design donné : une chaîne LFSR de 16 registres, deux chaînes de 32 registres ou trois chaînes de 8 registres présente une difficulté

de partage de ressources. Afin de s'affranchir de ce problème, Alaus a défini un système de Banc d'Opérateurs Communs (BOC) [25] où des opérateurs élémentaires de tailles identiques sont interconnectés. Ces connections sont paramétrables et permettent la mise en oeuvre des chaînes ER-LFSR de tailles variables. Ainsi, le BOC permettra aussi d'éviter les problèmes liés à l'ordonnancement des cellules matérielles partagées entre les fonctions de traitement de signal.
Malgré les avantages liés à la simplicité de l'utilisation de ces opérateurs de faible granularité, le gain en complexité et en flexibilité engendré par cette utilisation reste négligeable. Pour cette raison, d'autres travaux se sont intéressés à la définition des opérateurs de plus forte granularité comme la FFT et algorithmes de décodage de canal.

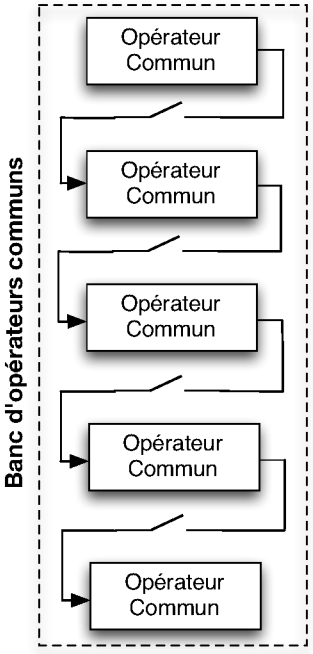


FIGURE 1.14 – Etage itératif pour les opérateurs LFSR ou Cordic

1.7.2 Les opérateurs à forte granularité : FFT et décodage FEC

Les travaux de recherche sur les opérateurs communs été initiés par Palicot dans [24] ont permis d'identifier plusieurs scénarios d'utilisation de la FFT dans plusieurs fonctions de traitement de signal (voir Annexe B). La FFT est un algorithme très connu qui calcule la Transformée de Fourier Discrète (DFT) et son inverse. Il existe plusieurs variantes de algorithme

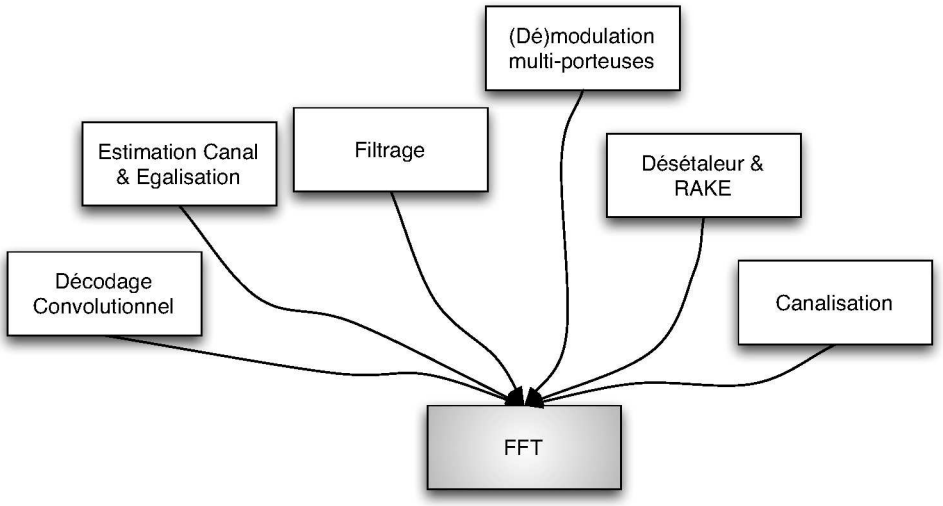


FIGURE 1.15 – Illustration de l'utilisation de l'opérateur FFT dans les fonctions de traitement de signal

FFT dans la littérature mais l'algorithme de Cooley-Tukey reste le plus utilisé en pratique. En effet, la FFT est définie dans [24] comme un opérateur mathématique utilisé dans des fonctions tels que le filtrage, l'estimation du canal, l'égalisation, le désétalement du spectre, la (dé)modulation multiporteuse...Dans [24] et [44], il a été montré que la FFT peut être utilisée comme opérateur commun pour un grand nombre de fonctions traitées dans le domaine fréquentiel comme illustré dans la figure 1.15.

Dans le même contexte de l'utilisation de l'opérateur FFT, il a été montré qu'il est efficace d'effectuer dans le domaine fréquentiel des calculs qui le sont habituellement dans le domaine temporel pour des opérations de décodage de canal de codes cycliques. Dans la thèse de Al-Ghouwayel [8], le cas de l'utilisation de la FFT, a été étudié pour le décodage de canal et plus particulièrement pour le codage de Reed Solomon, a été considéré. Ces travaux ont démontré qu'il est était possible d'utiliser une FFT reconfigurable capable d'effectuer les traitements classique d'une DFT dans le corps $\mathbb{C}$ des complexes, mais aussi ceux du décodage Reed-Solomon (RS) dans le corps finis de Galois (GF). Cela n'est par contre possible uniquement pour une famille des codes de RS, ceux définis dans le corps $GF(F_t)$ [45]. Alaus a proposé également la définition d'un opérateur commun pour les algorithmes de FFT et le décodage de Viterbi basé sur le partage de ressources de type papillon [10]. Nous étudions plus en détail cet opérateur dans les chapitres 3 et 4 de ce manuscrit où nous proposons de nouvelles structures et des architectures de gestion.

La figure 1.16 illustre le partage des opérateurs communs entre la FFT, le décodeur RS et Viterbi.

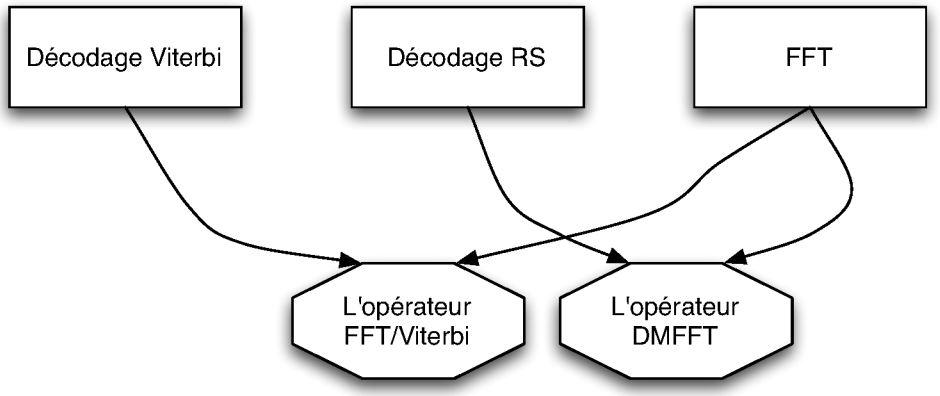


FIGURE 1.16 – Partage des opérateurs communs entre la FFT, RS et Viterbi

1.8 Conclusion

Dans un contexte de plus en plus multistandard, la reconfigurabilité est, à tous les niveaux, un axe central des recherches. Parmi les défis que soulève la radio reconfigurable est celui de la reconfigurabilité des plates-formes matérielles. Ce défi est important de par le fait que l'utilisation seule de processeurs généralistes ne semble pas être une solution ni à court, ni à long terme. Une approche pragmatique de conception des plates-formes reconfigurable est nécessaire pour traiter efficacement la diversité des fonctions en bande de base issues des différents standards. Pour atteindre cet objectif, nous avons proposé dans ce chapitre la technique des opérateurs communs comme une approche à suivre dans ce contexte. Nous avons présenté dans ce chapitre un premier jeu d'opérateurs communs que nous allons élargir dans la suite de ce manuscrit. Un des problèmes à résoudre dans l'utilisation de ces opérateurs communs, c'est comment gérer les ressources de calculs qu'ils offrent dans un contexte multistandard. Pour cela, nous présentons dans le prochain chapitre les techniques de management des opérateurs communs.

Chapitre 2

Management des opérateurs communs

Sommaire

2.1 Introduction 53
2.2 Définition du management des OCs 54
2.2.1 Le management des OCs pendant la conception . . 55
2.2.2 Le management des OCs pendant l'exécution . . . 55
2.2.3 Définition du problème de management des OCs . 58
2.3 Techniques de management des OCs 59
2.3.1 Couches logicielles de reconfiguration 60
2.3.2 Mécanismes de management des OCs 61
2.3.3 La librairie des opérateurs communs 61
2.3.4 Espace d'implémentation du manageur des OCs . 63
2.4 Considérations sur l'implémentation des OCs . . 64
2.4.1 Classification des accélérateurs matériels 65
2.4.2 Sélection des accélérateurs matériels 65
2.4.3 Analyse des coûts en traitement pour les OCs . . . 69
2.5 Modèle d'architecture reconfigurable à base d'OCs 75
2.6 Conclusion 77

2.1 Introduction

Le management de la reconfiguration joue un rôle très important dans le fonctionnement des systèmes radio reconfigurable. Dans ce contexte Nous proposons dans ce chapitre des techniques de management des opérateurs communs présentés dans le chapitre précédent.
Pour ce faire, nous présentons dans ce chapitre un modèle fonctionnel abstrait qui a pour objectif de permettre le déploiement des fonctionnalités reconfigurables sur une plate-forme matérielle hétérogène basée sur les opérateurs

communs. Dans une première section nous définissons d'une façon générale le management des opérateurs communs. Nous détaillons ensuite les techniques de management des opérateurs communs en se basant sur des approches de la littérature traitant la gestion de ressources matérielles reconfigurables. Ensuite, nous étudions l'implémentation de ces opérateurs en se basant essentiellement sur des évaluations de complexités pour des standards donnés. A la fin de ce chapitre, nous décrivons des modèles d'architectures permettant le management des opérateurs communs.

2.2 Définition du management des opérateurs communs

Nous définissons dans cette section le management des opérateurs communs ainsi que la méthodologie d'implémentation d'une plateforme à base des opérateurs présentés dans les chapitres précédents. On peut définir le mot « management » ou « gestion » dans le contexte des opérateurs communs comme :

- Les architectures qui incluent (a minima) l'instanciation d'opérateurs communs et des modules de gestion de ces opérateurs.
- Les méthodes d'utilisation et de partage des ressources offertes par les opérateurs communs dans contexte Multistandard.
- L'ordonnancement des fonctionnalités sur les opérateurs disponibles.
- La configuration effective des opérateurs et des chemins de donnés (bus, réseaux...) qui les lient.

L'objectif du management des opérateurs communs est d'abord de permettre la réutilisation des ressources matérielles et logicielles du design afin d'optimiser leur utilisation (pour limiter la complexité, réduire la consommation énergétique,...). Le deuxième objectif est de permettre une flexibilité qui peut couvrir les standards actuels et futurs. Le troisième objectif est d'offrir une puissance de calcul suffisante pour les traitements des standards actuels tout en garantissant une consommation énergétique raisonnable et un respect des contraintes temps-réel imposées par les standards. Bien entendu, les modules de gestion des opérateurs doivent avoir une complexité raisonnable qui pourra s'estimer en comparant la complexité de l'architecture reconfigurable globale, avec la somme des complexités des instances classiques nécessaires pour assurer les même modes, et/ou standards.
Ces défis, nécessitent l'éxploration de beaucoup de domaines dont ceux listés dans le tableau 2.1.

TABLE 2.1 – Domaines à explorer pour le management des opérateurs communs

Du point de vue méthodologique	Du point de vue architectural
- Le partitionnement matériel/logiciel - Co-design logiciel / matériel - L'ordonnancement - Le ≪ Profiling ≫ sous contraintes (comme par exemple la latence ou la consommation)	- Les platformes hybrides (matériel, logiciel) - Les architectures reconfigurables - La conception des MPSoCs - Les systèmes de communication sur puce (bus, réseau, crossbar...)

2.2.1 Le management des opérateurs communs pendant la conception

Nous proposons dans cette section une première définition de ce qu'on appelle ≪ management ≫ des opérateurs communs. Il s'agit du processus complet du design d'un processeur de traitement bande de base reconfigurable en utilisant des ≪ briques ≫ de base que l'on a appelé opérateurs communs. La figure 2.1 illustre les étapes de conception d'un équipement Multistandard avec la technique des opérateurs communs. Ces étapes commencent par la sélection des opérateurs communs à partir des contraintes des standards, suivie par les définitions des architectures et le partitionnement matériel/logiciel et enfin par l'ordonnancement de ces opérateurs. Pour cette dernière étape, nous avons utilisé deux termes différents pour différentier les cibles d'implémentation des opérateurs communs : l'ordonnancement pour les implémentations en logiciel et le ≪ Hardware multiplexing ≫ pour les implémentations en matériel.
Ces étapes de conception peuvent être formalisées et décrites mathématiquement avec la théorie des graphes (quelques exemples : [46] [47] [48] pour la sélection des opérateurs communs, [49] [50] [51] pour la synthèse automatique des architectures à base de Network-on-Chip NoC et [52] [53] [54] pour l'ordonnancement des systèmes distribués). Dans la suite de ce chapitre, nous proposons une approche plus ≪ pragmatique ≫ pour le management des opérateurs communs en se basant sur l'analyse des besoins des standards et les spécificités des architectures reconfigurables.

2.2.2 Le management des opérateurs communs pendant l'exécution

On définit dans ce paragraphe les techniques de management des opérateurs communs pendant l'exécution. Du point de vue niveau d'abs-

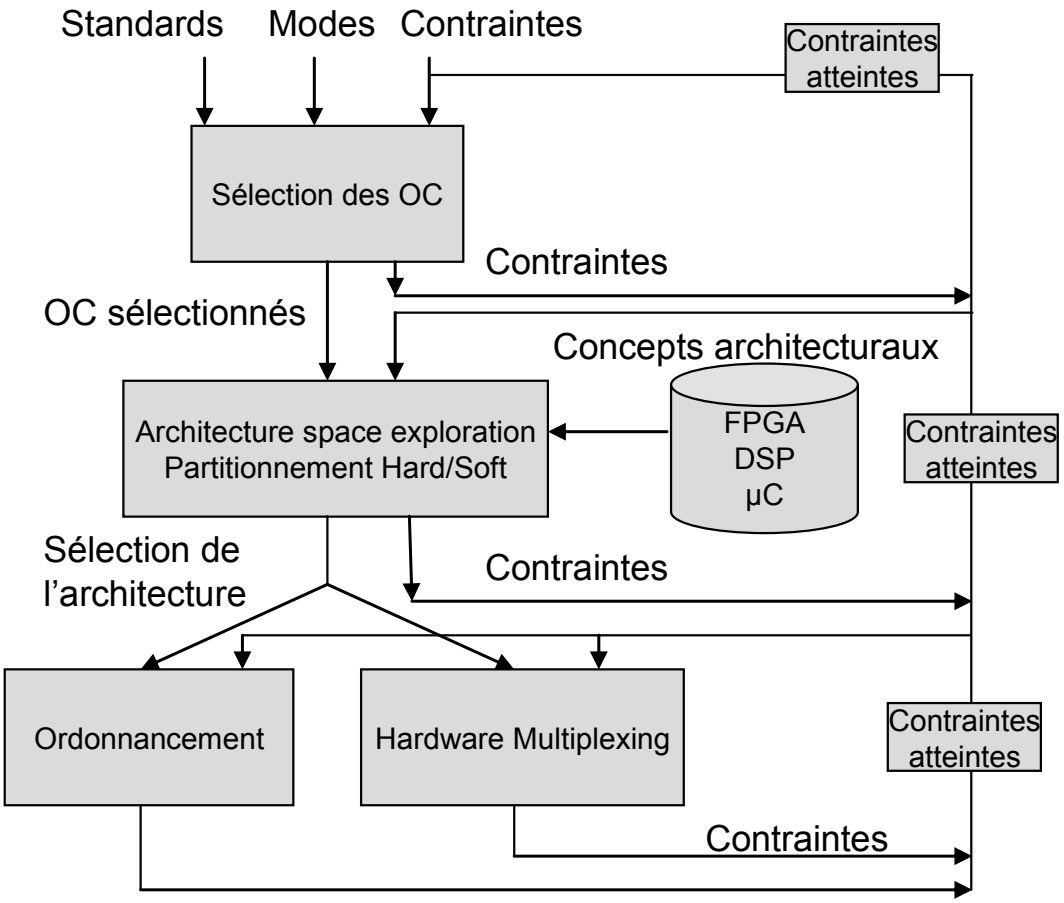


FIGURE 2.1 – Étapes de conception d'un équipement multistandard avec la technique des opérateurs communs

traction, le manager des opérateurs communs se situe entre la plateforme matérielle des opérateurs communs et la couche applicative (standards & fonctions). La figure 2.2 illustre une représentation en couches du positionnement du manager des opérateurs communs.

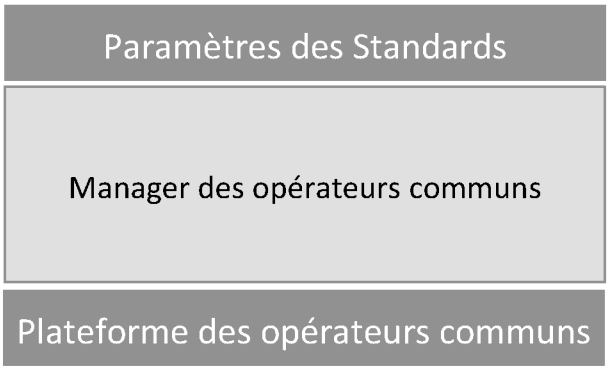


FIGURE 2.2 – Représentation en couches du manager des opérateurs communs

Pour mieux définir le manager des opérateurs communs nous commençons par décrire les besoins des systèmes de communications actuels et comment le manager des opérateurs communs contribue à aborder ces besoins. Les points qui suivent décrivent comment le manager des opérateurs communs contribue à la réalisation des objectifs visés par la technique des opérateurs communs.

Réduction du cout : Une fois bien défini, le manager des opérateurs communs joue un rôle très important dans la réduction du temps de développement. En effet, il permet la définition d'une couche d'abstraction de la plateforme qui permet aux designers de créer plus facilement et plus efficacement des nouvelles applications à base des opérateurs communs.

Evolutivité : Le manager des opérateurs communs exploite la flexibilité offerte par les opérateurs communs pour permettre une implémentation évolutive qui s'adapte facilement aux nouveaux besoins des standards.

Flexibilité : Selon les besoins des standards et les ressources disponibles dans la plateforme, le manager alloue les ressources offertes par les opérateurs communs d'une façon flexible. Cette allocation de ressources doit se faire pendant l'exécution. Donc le manager des opérateurs communs doit se baser sur des algorithmes de décision à la fois rapides et efficaces. Il faut différentier ici le cas où la plateforme se reconfigure pour passer d'un standard à l'autre ou le cas ou elle se reconfigure pour effectuer différents traitements du même

standard. Cette 2ème approche peut mener a un surcroit de complexité et de mémorisation du contexte prohibitif [10]. Une plateforme basée sur des opérateurs communs contient des opérateurs hétérogènes du point de vue implémentation (matérielle ou logicielle) et du point de vue granularité. Les algorithmes d'allocation doivent donc être capables de gérer cette hétérogénéité et supporter différents types d'opérateurs.

Aspect temps réel : Le manager des opérateurs communs doit tenir compte de l'aspect temps réel qui est très important pour les applications de télécommunications. Il doit offrir une couche d'abstraction au-dessus des opérateurs et de la plateforme matérielle qui permet aux développeurs de gérer les timings et la performance du mapping.

Performances en puissance de calcul : Le manager des opérateurs communs se base sur des fonctions couts (puissance de calcul et de la consommation) dans les décisions et l'allocation des ressources. Ce qui veut dire, par exemple, en assignant des ressources opérateurs communs au standard ou à la fonction, le manager des opérateurs communs peut optimiser la consommation tout en respectant les contraintes de performances.

2.2.3 Définition du problème de management des opérateurs communs

Au sens large, le problème principal que l'on cherche à résoudre avec le manager des opérateurs communs est de faire correspondre les besoins et les propriétés des standards et fonctions aux ressources offertes par les opérateurs communs et ce d'une façon flexible, rapide et efficace.
Il est évident que les domaines concernés par cette question sont très vastes et les explorer entièrement dépasserait le cadre de cette thèse. Pour cela, nous définissions l'approche à suivre pour assurer le management des opérateurs communs dans un contexte général multistandard et nous appliquerons les principes développés à des fonctions spécifiques de traitement de signal (comme les algorithmes de FFT et les décodeurs de Viterbi).
Pour satisfaire les besoins des standards en termes d'opérateurs communs, le manager a d'abord besoin de construire à partir des modes et des spécification des standards (BER, délais,..) un graphe de tâches qui définit les opérateurs utilisés, les interconnexions entre ces opérateurs et les besoins en mémoire. Une fois cette opération effectuée, le manager doit ensuite affecter les tâches précédemment définies aux ressources matérielles disponibles (Figure 2.3). Il doit effectuer ces deux opérations (définitions du graphe des tâches et l'affectation des ressources) rapidement sans générer d'interférences inter-fonctions ou inter-standards. De plus, le manager doit offrir une couche d'abstraction qui permet aux développeurs d'exploiter facilement les opérateurs communs

dans les designs.

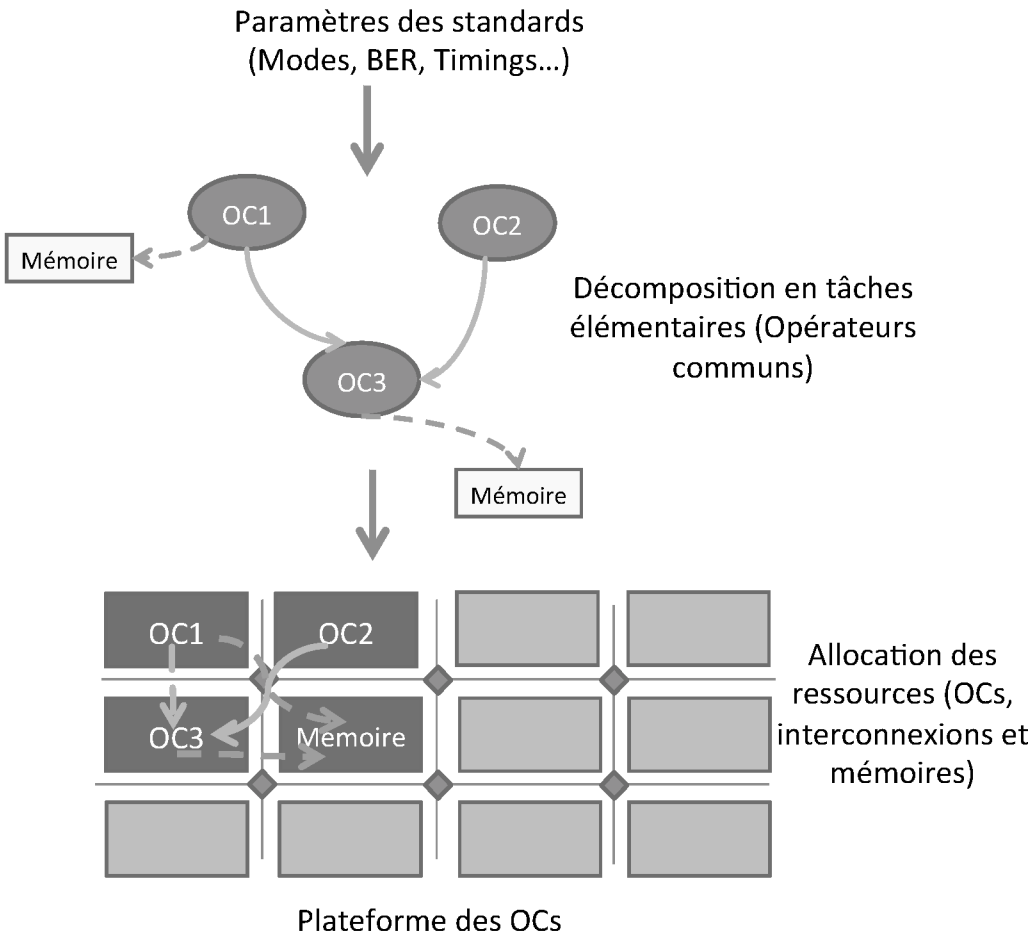


FIGURE 2.3 – Étapes du management des opérateurs communs pendant l'exécution

2.3 Techniques de management des opérateurs communs

Cette partie détaille les composants du manager des opérateurs communs. Concrètement, nous définissons les services que doit assurer le manager des opérateurs communs par rapport aux standards et quelles tâches il doit assurer par rapport à la plateforme. La figure 2.4 détaille les composants du manager des opérateurs communs que l'on va décrire dans les paragraphes suivants.

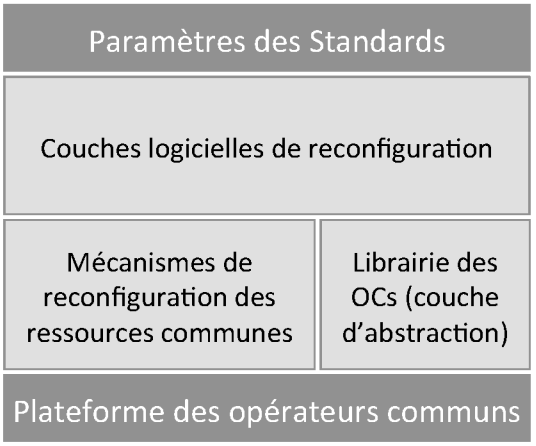


FIGURE 2.4 – Les composants du manager des opérateurs communs

2.3.1 Couches logicielles de reconfiguration

Les couches logicielles de reconfiguration constituent en général un composant indépendant de la plateforme matérielle. Il interagit avec les standards et les algorithmes de traitement de signal. Pour son fonctionnement, il reçoit comme information les paramètres des standards (modes, délais, taux d'erreurs...) et des informations concernant les opérateurs disponibles. Son rôle est de faire correspondre les besoins des standards avec les ressources en opérateurs communs, interconnexions et mémoire. Comme le montre la figure 2.3 la couche logicielle de reconfiguration génère un graphe de tâches qui définit les opérateurs utilisés les interconnexions entre ces opérateurs et les besoins en mémoire. Nous ne traitons pas ce composant dans cette thèse puisqu'il existe beaucoup de travaux dans la littérature qui ont étudié ce problème indépendamment de la plateforme cible en définissant des couches logicielles intermédiaires [55] [56] [57].
Les couches logicielles sont introduites afin de rendre la cible d'exécution transparente par rapport à la partie applicative et ainsi augmenter la portabilité des composants radio logicielle et leur réutilisabilité. L'abstraction de l'architecture matérielle permet de rendre l'application indépendante du matériel sur lequel elle est exécutée. Ainsi le développeur pourra augmenter la réutilisation de son application sur diverses cibles matérielles. La complexité de cette couche d'abstraction dépend des fonctionnalités qu'elle offre. Les couches logicielles intermédiaires les plus complexes ont pour objectif de proposer une architecture logicielle standardisée permettant une interopérabilité et une évolutivité de l'équipement. Cette tendance est portée initialement par le projet SCA (militaire à l'origine) [58] [57]. Nous pouvons aussi citer les

travaux de l'UPC (Universitat Politècnica de Catalunya) [56] [59] [60] qui offrent un niveau d'abstraction inférieur à celui du SCA afin de minimiser le surcoût apporté par les couches intermédiaires.
Dans le cadre des opérateurs communs, les travaux qui ont été fait à Supélec par Gul [11] et Kaiser [12] peuvent adaptés pour générer les graphes des tâches à base des opérateurs communs en ajoutant la composante « temps » à la fonction d'optimisation [61]. Dans l'état actuel de ces travaux, l'ordonancement temporel des traitements n'est pas pris en compte dans les fonctions d'optimisations.

2.3.2 Mécanismes de management des ressources communes

Une fois que le graphe des opérateurs communs a été généré, le manager des ressources alloue les opérations aux ressources disponibles de la plateforme : les éléments de traitement (processing elements PE), les interconnexions et les mémoires). Le problème a déjà été traité dans plusieurs travaux [62] [63] [64]. Il s'agit de faire correspondre (mapping) un graphe des OCs (GO) à un graphe architecture (GA) [62]. Comme le montre la figure 2.5, le manager des opérateurs communs se base sur des contraintes de coûts pour les allocations des PEs, des mémoires et des interconnexions. Le gestionnaire de ressources effectue donc les décisions d'allocation des ressources. Toutefois, pour l'exécution de ces décisions, le gestionnaire de ressources doit s'appuyer sur les mécanismes. Un mécanisme décrit un ensemble d'actions, l'ordre dans lequel elles doivent être effectuées, leurs conditions respectives et les événements déclencheurs de ces actions. Afin de détecter les événements de déclenchement, un mécanisme repose sur un ou plusieurs moniteurs. L'action est effectuée par un ou plusieurs actionneurs. Les mécanismes de gestion des ressources fonctionnent en étroite collaboration avec la bibliothèque des opérateurs communs, afin de réaliser les deux fonctions d'allocation de base, comme l'instanciation de tâches sur les éléments de traitement PEs, l'allocation des blocs mémoire et la mise en place de structures de communication inter-opération, et des fonctions plus complexes comme par exemple la migration des opérations en exécution (réaffectation de opérateurs communs). Nous nous intéressons à la définition détaillée de ces mécanismes dans les prochains chapitres pour le cas de l'opérateur FFT/Viterbi.

2.3.3 La librairie des opérateurs communs

La librairie des opérateurs communs, précédemment illustré sur la figure 2.4, a deux fonctions principales :

1. Fournir des primitives qui permettent d'avoir une couche d'abstraction à la fois des opérateurs communs et de la plateforme utilisée. Elle per-

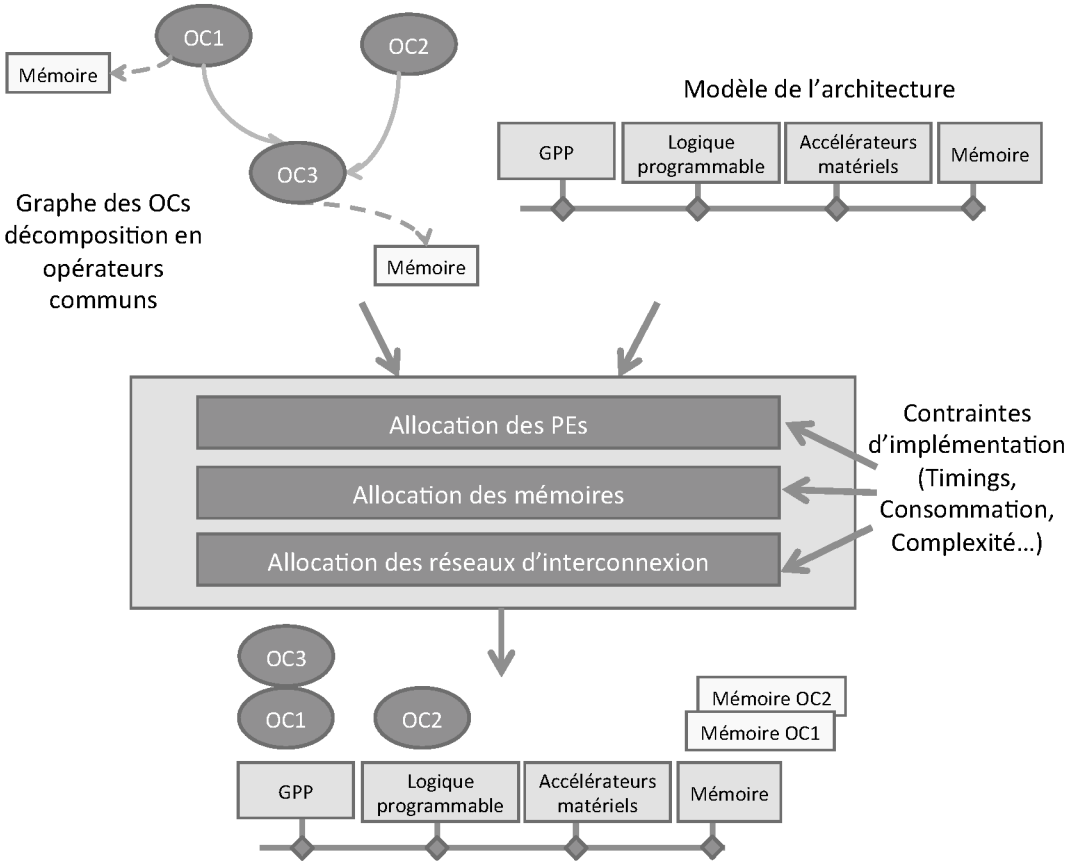


FIGURE 2.5 – Mécanismes de management des ressources communes

met aux concepteurs de développer facilement des applications à base d'opérateurs communs.

2. Collaborer avec les couches logicielles de reconfiguration et les mécanismes de management pour exécuter les décisions en agissant comme interface entre ses différentes couches (générateur du graphe des opérateurs communs et mécanismes de management des ressources communes).

Elle intervient sur trois niveaux :
- Au niveau de la génération du graphe des opérateurs communs : en fournissant des primitives qui lient les standards et les fonctions de traitement de signal avec le générateur du graphe des opérateurs. Cela permet aux standards de renégocier de nouvelles reconfigurations selon l'état du canal de communication par exemple.
- Au niveau de la communication inter-opération : en fournissant des primitives liées au modèle de programmation comme par exemple lorsque les opérateurs communes communiquent avec des messages (primitives typiques *envoyer() recevoir()*).
- Au niveau de l'ordonnancement : en fournissant des primitives pour créer/détruire des opérations dans la plateforme et gérer leurs interactions (avec des sémaphores par exemple) [61].

2.3.4 Espace d'implémentation du manageur des opérateurs communs

Les différentes façons selon lesquelles le graphe des opérateurs communs est décomposé et mappé sur le graphe de l'architecture définissent l'espace de solutions de notre mapping. Différentes politiques de management des opérateurs communs peuvent être appliquées et différents compromis peuvent être identifiés (rapidité de mapping contre qualité ou algorithmes statiques Vs dynamiques...) [65]. La figure 2.6 montre d'autres degrés de libertés dans l'implémentation du manageur des opérateurs communs.

Adaptatif Vs Non-adaptatif : Pour les architectures reconfigurables des terminaux multistandards, notre choix pour la politique de gestion de ressources va aller vers les solutions adaptatives (dynamique) [66]. En effet, une dégradation de l'état du canal peut engendrer un changement des algorithmes de traitement de signal utilisés et donc changement (qui peut être important dans les standards actuels) des ressources matérielles utilisées.

Centralisé Vs Distribué : Vu le nombre d'opérateurs considérés et la variété de leurs niveaux de granularité, nous excluons la solution centralisée afin de minimiser le flux de reconfiguration entre les PEs. Par contre, la solution distribuée ne convient pas puisqu'il faut intégrer un

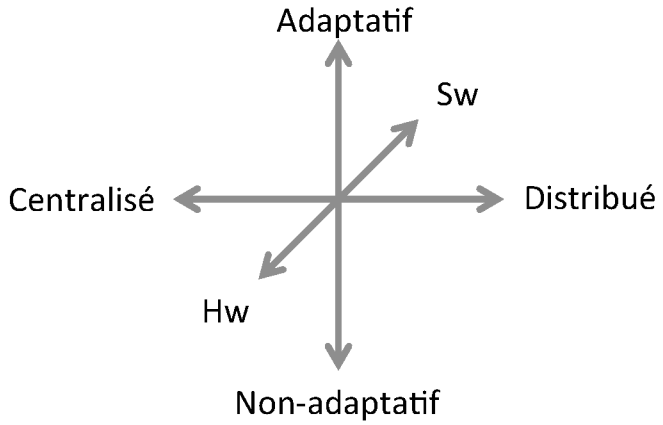


FIGURE 2.6 – Degrés de libertés dans l'implémentation du manager des opérateurs communs

système de management complet avec toutes ses fonctionnalités (trop complexe) dans chaque PE. On optera donc pour une solution mixte où un processeur de contrôle (central) s'occupera de la génération des graphes des tâches et une partie des politiques de gestion des ressources (pour les opérateurs de faible granularité) et laisser les mécanismes de gestion de ressources se gérer d'une façon distribuée au niveau des PEs.

Matériel Vs Logiciel : les systèmes de management et les ordonnanceurs sont typiquement implémentés en logiciel. Depuis quelques années, avec le succès des MPSoCs, beaucoup de fonctionnalités de gestion de ressources commencent à être implémentées en matériel supplémentaire pour des besoins de puissance ou de consommation [67]. Dans le problème que nous traitons, à savoir celui des architectures reconfigurables à base d' opérateurs communs de granularité très variées, il est plus approprié de considérer une implémentation mtérielle localement (aux niveaux des PEs) et logicielle globalement pour rester en cohérence avec le point précèdent.

2.4 Considérations sur l'implémentation des opérateurs communs

Tel qu'il a été défini, un opérateur commun peut être implémenté aussi bien en matériel qu'en logiciel. Le choix de l'implémentation est généralement fait avec le partitionnement matériel/logiciel qui ne reste applicable que pour des implémentations bien spécifiques. Toutefois, nous proposerons dans ce paragraphe un raisonnement qui permettra de faire des choix concernant l'implémentation des opérateurs et l'architecture du processeur bande de base

sur une plateforme hybride matérielle/logicielle d'une façon générale. Nous identifions, dans le paragraphe suivant, les opérateurs qui seront implémentés en matériel.

2.4.1 Classification des accélérateurs matériels

La solution pour augmenter la capacité de calcul en gardant une bonne efficacité de consommation et de surface est de faire le choix des accélérateurs matériels [68]. L'accélérateur est un module matériel supplémentaire ajouté au design pour effectuer une classe de tâches préconfigurées. En revanche, chaque tâche accélérée supplémentaire augmente le coût du matériel. Pour cette raison, il est essentiel de sélectionner les bons accélérateurs pour couvrir efficacement les besoins de calculs des standards ciblés. L'accélération matérielle peut se faire à plusieurs niveaux et selon différentes fonctions :

Accélération niveau fonction : Il s'agit d'une fonction complète, comme par exemple un décodeur de Viterbi, un filtre ou une FFT, implémentée en stand-alone et définie en tant qu'une unité d'exécution indépendante connectée au ≪ réseau ≫ de communication interne du MPSoC et contrôlées à travers des registres de contrôle. L'implémentation des accélérateurs niveaux fonction peut être classifiée en deux catégories. La première catégorie est celle des fonctions nécessitant peu de flexibilité telles que les filtres, FFT, décodeurs...qui sont généralement implémentés en ASIC. La deuxième regroupe les fonctions nécessitant plus de flexibilité (ou une reconfigurabilité de fine granularité) qui est généralement implémentée sur FPGA au détriment de la surface du circuit et des tailles des mémoires.

Accélération niveau instruction : L'accélération niveau instruction est une autre forme d'accélération utilisée au sein même des processeurs pour accélérer une tâche spécifique. Ce type d'accélération peut concerner des opérateurs comme le LFSR, Cordic...

2.4.2 Sélection des accélérateurs matériels

Afin de faire les bon choix dans la sélection des accélérateurs matériels, on propose dans ce paragraphe une approche permettant de donner une nouvelle classification des opérateurs pour proposer nos premières idées d'architectures reconfigurables avec l'approche des opérateurs communs. On liste ci-dessous les critères d'évaluation pour choisir les opérateurs implémentés en matériel.

Le coût en MOPS : C'est le premier critère qui permet de comparer les opérateurs communs en fonction de leurs charges de calcul. Le coût en $MOPS$ correspond au nombre d'instructions par seconde (en millions) avec lequel un GPP standard pourrait traiter une fonction spécifique. Il est évalué comme suit :

$$Coût\ en\ MOPS = \frac{OP \times N}{t} \tag{2.1}$$

Où OP est le nombre de cycles d'horloge nécessaire pour qu'un GPP standard effectue l'opération commune, N est le nombre d'échantillons ou bits à traiter et t est temps maximal alloué pour effectuer l'opération. La figure 2.7 fournit des exemples des besoins de calculs pour la (dé)modulation et le décodage.
A partir du coût en $MOPS$ on peut dans un premier temps identifier les opérateurs qui ne peuvent être implémentés qu'en matériel pour certains standards. De plus, on peut définir à partir du coût en $MOPS$ un autre indicateur important qui peut non seulement être utilisé pour le partitionnement matériel/logiciel des opérateurs mais aussi pour valider le choix des opérateurs communs ou en identifier d'autres. Il s'agit du coût relatif en $MOPS$ d'un opérateur par rapport au coût total de partie numérique de l'équipement.
Le tableau 2.2 représente les coûts d'implémentation sur un GPP des fonctions pour les terminaux W-CDMA (Wideband Code Division Multiple Access) et IEEE 802.11a. On peut constater que pour le 802.11a (standard utilisant la modulation OFDM) les FFT et le décodeur Viterbi (qui ont déjà été identifiés comme opérateurs communs) représentent 80% du coût total de l'équipement multistandard et en ajoutant le filtrage (non identifié comme OC) on peut dépasser les 90%. Pour le W-CDMA n'utilisant pas la modulation OFDM, les opérateurs communs couvrent approximativement 30% du coût total de léquipement multistandard. Sachant que cette évaluation se base sur l'utilisation classique des FFT et décodeurs, une utilisation plus optimisée [24] (par exemple l'utilisation de la FFT dans les filtres FIR, désétalement,...) augmentera le coût relatif en MOPS de ces opérateurs par rapport au coût de l'équipement multistandard. Dans cette analyse les coûts des opérateurs LFSR et Cordic on été négligés puisqu'il s'agit des accélérateurs classés au niveau instruction utilisés par des fonctions nécessitant peu de ressources par rapport au coût total de l'équipement multistandard.
On peut conclure que pour le critère coût en charges de calcul, il est intéressant d'implémenter les opérateurs à forte granularité (comme les FFT, décodeurs de canal,...) en matériel (accélérateurs niveau fonction) plus que les opérateurs à faible granularité et plus spécialement pour les standards à base d'OFDM.

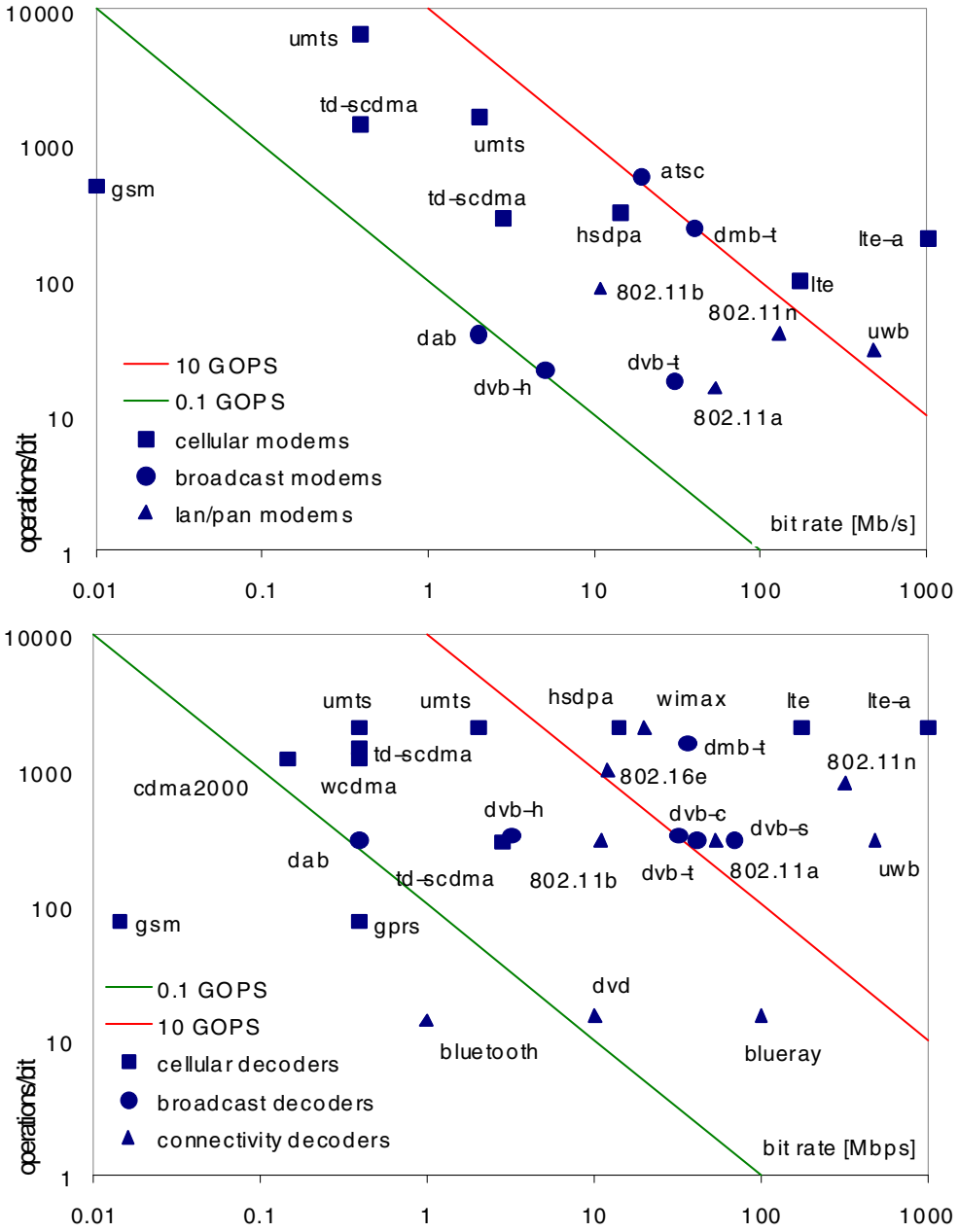


FIGURE 2.7 – Les besoins de calculs pour la (dé)modulation (Haut) et le décodage (Bas) [16]

TABLE 2.2 – Coûts d'implémentation des terminaux W-CDMA et 802.11a sur un GPP (adapté de[69])

Algorithmes	Configurations	GPP (Alpha)			
		Traitement Vectoriel	Taille des Vecteurs	Largeur des données (Bit)	Mcycles par sec.
802.11a (24 Mbps)					
FFT	64 points	Oui	64	16	15,600
IFFT	64 points	Oui	64	16	15,600
Égaliseur	64 points	Oui	54	16	960
QAM	Constéllation 64 points	Non	1	4	1
DQAM	Constéllation 64 points	Non	1	4	3
FIR (Tx)	1 filtre $\times 33tap \times 20MSps \times 2$	Oui	33	16	3,040
FIR (Rx)	1 filtre $\times 33tap \times 40MSps \times 2$	Oui	33	16	6,080
Sync. Fréq.	Défini dans le standard 802.11a	Partiel	16	16	190
Sync. Temp.	Défini dans le standard 802.11a	Partiel	16	16	190
Interpolateur	Structure Farrow 8 taps	Oui	2048	16	4,800
Entrelacement	1 frame	Non	1	1	290
Désentrelacement	1 frame	Non	1	16	290
Enc. Conv.	$K = 7$, Rate = 3/4	Partiel	6	1	100
Dec. Viterbi	$K = 7$, Soft Input	Partiel	64	8	35,000
Embrouilleur	Défini dans le standard 802.11a	Partiel	7	1	340
Débrouilleur	Défini dans le standard 802.11a	Partiel	7	16	340
Total Mcycles par sec.					82,824
W-CDMA (2 Mbps)					
Embrouilleur	Défini dans le standard W-CDMA	Oui	2560	1,1	240
Débrouilleur*	12 fingers, 3 base stations	Oui	2560	1,8	2,600
Étalement	Facteur d'étalement = 4	Oui	512	8	300
Détalement*	12 fingers, 3 base stations	Oui	512	8	3,600
PN Code (Rx)	3 base stations	Non	1	8	30
PN Code (Tx)	Défini dans le standard W-CDMA	Non	1	8	10
Combiner*	2 Mbps Data Rate	Partiel	12	8	100
FIR (Tx)	4 filters $\times 65$ coeff. $\times 3.84MSps$	Oui	64	1,16	7,900
FIR (Rx)	2 filters $\times 65$ coeff. $\times 7.68MSps$	Oui	64	8,8	3,900
Searcher*	3 base stations, 320 windows	Oui	320	1,8	26,500
Entrelacement	1 frame	Non	1	8	10
Désentrelacement	1 frame	Non	1	8	10
Codeur Turbo	K=4	Partiel	3	1,1	100
Décodeur Turbo*	K=4, 2 SOVA, 5 Iterations	Partiel	8	8,8	17,500
Total Mcycles par sec.					62,800
*Algorithmes dont les charges de traitement varient avec l'état du canal de transmission					

La réutilisation : Ce critère a été considéré lors que la sélection des opérateurs communs avec la méthode des graphes. On définit la réutilisation comme un autre critère pour l'identification des accélérateurs matériels. Pour ce critère, ce sont les opérateurs à faible granularité qui l'emportent. En effet, ces opérateurs sont les plus utilisés par les fonctions et les standards. En revanche, une très forte réutilisation peut poser des problèmes architecturaux pour le partage de ces ressources.

La programmabilité : Le troisième critère concerne la programmabilité de l'opérateur Commun. Moins on a besoin de programmabilité et plus il est intéressant de considérer une implémentation en matériel. Le tableau 2.3 montre le degré de programmabilité des fonctions dans les terminaux multi-standard. On peut constater que les filtres au niveau du frontend numérique sont ceux qui nécessitent le moins de programmabilité ; du fait de leurs coûts importants en MOPS, on peut se poser la question sur la possibilité de considérer un nouvel opérateur Filtrage/décimation.

Surface et Consommation : L'implémentation en matériel des opérateurs communs n'est justifiée par rapport à une implémentation logicielle que si on peut gagner considérablement en surface ou/et en consommation. Ce critère est particulièrement difficile à évaluer sans implémentation. En effet, il faut tenir compte des instances de contrôle et des connexions ce qui n'est pas facile à estimer surtout pour les opérateurs communs à faible granularité.

TABLE 2.3 – Classification du degré de programmabilité pour quelques fonctions de traitement de signal (adapté de [16])

Besoins de programmabilité	Fonctions
Très élevé	Couches protocolaires
Elevé	Estimation du canal
Moyen	(Dé)modulation
Faible	Décodage
	(i)FFT
Trés faible	Filtres

2.4.3 Analyse des coûts en traitement pour les opérateurs communs

Afin de compléter la réflexion initiée dans les deux paragraphes précédents sur l'implémentation des opérateurs communs, nous évaluons dans ce paragraphe les coûts des traitements de ces opérateurs (particulièrement pour les opérateurs FFT et décodage FEC).

2.4.3.1 Coût en traitement de la FFT

Afin d'évaluer les besoins des traitements de la FFT, nous partons des spécifications de quelques standards utilisant la modulation OFDM (3GPP LTE, IEEE 802.11a/g, 802.16 et DVB-T), et nous évaluions les besoins de ces opérateurs en puissance de calculs. Nous considérons dans cette étape la période minimale de la FFT spécifiée par les standards comme le paramètre principal déterminant pour les performances nécessaires des implémentations. Nous présentons dans le tableau 2.4 ce dernier paramètre avec les tailles maximales des FFTs pour les standards précédemment cités, le nombre des papillons logiques correspondant aux tailles des FFTs ($\frac{N}{2}\log_2(N)$) et le coût approximatif en opérateurs communs par seconde (MOCPS). Ce coût en MOCPS est calculé conformément à l'équation 2.1 où l'opération élémentaire est l'opérateur commun (papillon Radix-2 définit dans le chapitre précédent). Il faut mentionner que ce coût est approximatif puisqu'il ne prend pas en compte le délai nécessaire pour effectuer les permutations avant et/ou après les opérateurs.

TABLE 2.4 – Coût en MOCPS pour quelques standards à base de la modulation OFDM

Standard	Période min. de la FFT	Taille max. de la FFT	Nombre max. des papillons logiques (Radix-2)	Coût approximatif en MOCPS
3GPP LTE	$67.77\mu s$	2048	11264	166 MOCPS
802.11a/g	$3.2\mu s$	64	192	60 MOCPS
802.16	$102\mu s$	1024	5120	50 MOCPS
DVB-T	$450\mu s$	8192	53248	118 MOCPS

Pour mieux exploiter ces contraintes des standards, nous proposons de faire quelques benchmarks des implémentations de l'opérateur FFT en logiciel et en matériel et vérifier si on peut respecter ces contraintes. La première partie du tableau 2.5 présente une comparaison des performances de l'implémentation d'une FFT Radix-2 que nous avons testé sur un DSP (Analog Devices de la famille ADSP-21xxx SHARC [1]) et sur un FPGA (Xilinx Virtex [2] 4). Pour cette comparaison, nous avons utilisé l'outil Coregen de Xilinx et les exemples de codes d'Analog Devices pour générer les différents FFTs. Les deux implémentations utilisent un seul papillon Radix-2

1. http://www.analog.com/
2. http://www.xilinx.com/

TABLE 2.5 – Comparaison des implémentations matérielles et logicielles des FFTs Radix-2

Taille de la FFT	ADSP-21xxx SHARC				FPGA Xilinx Virtex 4			
	Latence en cycles	Latence en μs @150 MHz	Latence en μs @450 MHz	Mémoire data en $Kbit$	Latence en cycles	Latence en μs @150 MHz	Latence en μs @450 MHz	Mémoire data en $Kbit$
Radix-2								
64	1002	7	2	5	440	3	1	90
128	2088	14	5	10	843	6	2	90
256	4486	30	10	20	1694	11	4	90
512	9764	65	22	41	3505	23	8	90
1024	21314	142	47	81	7364	49	16	90
2048	46432	310	103	164	15575	104	35	162
4096	100734	672	224	328	33002	220	73	324
8192	217504	1450	483	655	69885	466	155	612
Radix-4								
64	920	6	2	6	250	2	1	198
256	4044	27	9	23	864	6	2	198
1024	19245	128	43	90	3446	23	8	198
4096	90702	605	202	360	14476	97	32	396
16384	419434	2796	932	1441	61602	411	137	1512

(l'opérateur commun) avec une taille Input/output de 32 bits. Les résultats montrent que pour les implémentations sur DSP, on ne peut respecter la contrainte de la période minimale de la FFT que pour le standard IEEE 802.16 pour une fréquence de fonctionnement à 450 MHz. En revanche, il est possible de respecter les délais de tous les standards considérés pour une implémentation sur FPGA (et a fortiori sur technologie ASIC). En revanche, même pour une implémentation en matériel, il n'est pas possible de traiter une seule FFT en respectant les délais des standards. Cela ne résout encore pas le problème puisqu'il faut dupliquer la même implémentation autant de fois que nombre des FFTs présents dans l'équipement multistandard.
Nous avons également comparé des implémentations des FFT Radix-4 sur FPGA et sur DSP pour vérifier si l'augmentation de la taille du Radix résout le problème. La deuxième partie du tableau 2.5 présente une comparaison des performances des implémentations FFT Radix-4 testées dans les mêmes conditions que précédemment. Les résultats montrent que les performances sont légèrement améliorées par rapport à celles du Radix-2 au détriment d'une augmentation des besoins en mémoire et d'une limitation des tailles des FFTs à des multiples de 4.

Nous pouvons ainsi conclure que l'utilisation de la FFT comme une ressource partagée entre plusieurs fonctions de traitement de signal nécessite une implémentation plus rapide que les implémentations Radix-2 classiques. Nous avons donc le choix entre d'une part l'utilisation des Radix supérieurs / mixes ce qui va augmenter légèrement les performances ou d'autre part l'exploitation du parallélisme des papillons (opérateurs communs) pour améliorer la vitesse des traitements. Nous avons opté par la suite pour la deuxième solution qui s'adapte bien aux objectifs de la technique des opérateurs communs.

2.4.3.2 Coûts en traitement des décodeurs FEC

Comme pour la FFT, nous évaluons dans ce paragraphe les coûts en traitements des décodeurs FEC pour les mêmes standards considérés que précédemment. Cependant, contrairement à la FFT, la latence du décodeur dépend de la taille de la trame d'entrée et du nombre d'itérations. Elle dépend aussi de l'état du décodeur à un instant donné. En particulier, le temps nécessaire pour décoder un bloc de code peut être important si le décodeur est toujours en itération sur le bloc de code précédent. Pour cette raison nous considérons ici le « nombre de cycles par bit décodé » comme un paramètre principal dans la comparaison des performances nécessaires des implémentations. Le tableau 2.6 montre les contraintes des standards en termes de nombres d'états du décodeur et le débit maximal requis. Dans ce tableau le nombre de papillons logiques est la moitié du nombre d'état du décodeur. Pour mieux exploiter ces contraintes des standards, nous évaluons dans l'équation 2.2 le « nombre maximal de cycles par bit décodé » ce qui

TABLE 2.6 – Contraintes des standards considérés pour le décodeur

Standard	Type du code	Débit	Nombres des papillons logiques
3GPP LTE	8 états Turbo Code (TC)	Jusqu'à 100 Mbps	-
802.11a/g	64 états Code Convolutif (CC)	Jusqu'à 54 Mbps	32
802.16	64 états CC 8 états TC	Jusqu'à 20 Mbps	32
DVB-T	64 états CC	Jusqu'à 32 Mbps	32

permettra de définir la latence à respecter.

$$Nombre\ Max.\ Cycles/bit\ décode = \frac{Fréquence\ de\ fonctionnement}{Debit\ Max.} \quad (2.2)$$

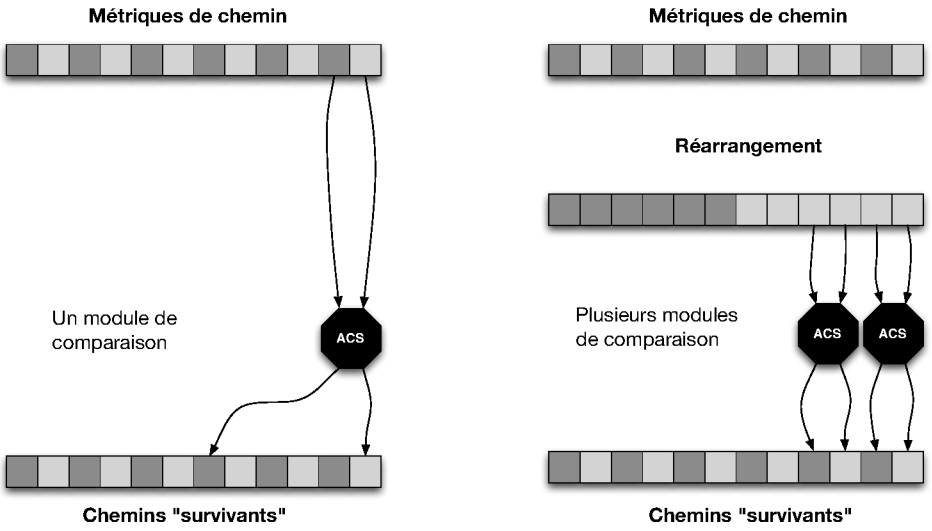
Le tableau 2.7 montre les valeurs du « nombre maximal de cycles par bit décodé » pour les standards sélectionnés et pour les fréquences de 150 MHz et 450 MHz. Nous constatons que les contraintes sont plus fortes par rapport à celles de la FFT. Cela est prévisible puisque la FFT traite des symboles et le décodeur manipule des bits avec des débits plus importants.

TABLE 2.7 – Contraintes des standards considérés pour le décodeur en cycles par bit décodé

		# Max. cycles par bit décodé	
Standard	Débit max.	@150 MHz	@450 MHz
3GPP LTE	100 Mbps	2	5
802.11a/g	54 Mbps	3	9
802.16	20 Mbps	8	23
DVB-T	32 Mbps	5	15

Les contraintes présentées dans le tableau 2.7 montrent qu'il est difficile de répondre aux besoins des standards avec une implémentation en logiciel. En effet un décodeur de Viterbi de faible complexité pour les applications GSM (longueur de contrainte = 6 avec un rendement = 1/2) a une complexité de 30751 Cycles et il a besoin de 58 cycles pour décoder un bit sur une cible DSP ADSP-21xxx SHARC. De plus, même pour les implémentations en matériel, il est nécessaire de paralléliser les traitements en augmentant le nombre des modules Add-Compare-Select (ACS) physiques (l'opérateur commun) comme le montre la figure 2.8.

Les chiffres présentés dans les tableaux 2.5 et 2.7 confirment les premières analyses faites dans le paragraphe 2.4.2 sur la nécessité d'implémenter ces opérateurs FFT/Viterbi en matériel. Ainsi, nous pouvons atteindre un gain en capacité de calcul considérable lorsque ces opérateurs matériels sont partagés entre les algorithmes.



(a) Implémentation série (b) Implémentation parallèle

FIGURE 2.8 – Parallélisations des traitements pour le décodeur de Viterbi

2.4.3.3 Coût en traitement des opérateurs LFSR et CORDIC

Nous discutons dans ce paragraphe le coût en traitement des opérateurs LFSR et CORDIC que nous avons classé dans le paragraphe 2.4 comme des opérations d'accélération niveau instruction. Il s'agit d'opérateurs de faible granularité appelés par des fonctions de faible complexité. En effet, les LFSRssont très rapide et utilisent peu de ressources puisqu'un LFSR typique sur FPGA consomme approximativement 400 « cellules logiques » et sur DSP nécessitent entre 200 et 700 cycles. De même pour l'opérateur CORDIC qui utilise dans une implémentation typique sur FPGA 800 « cellules logiques » et nécessitent rarement plus de 1000 cycles.
Les implémentations de ce type d'opérateurs communs conviendraient bien à des accélérations au niveau instruction qui viendraient se greffer à un processeur de traitement de signal qui effectue des opérations simples tels que les fonction trigonométriques avec l'opérateur CORDIC [9] ou des fonctions telles que le codage/décodage CRC ou le codage convolutif pour le LFSR [10].

2.5 Modèle d'architecture reconfigurable à base d'opérateurs communs

Nous avons présenté précédemment dans ce chapitre les plates-formes matérielles, les couches logicielles, et les architectures de gestion de reconfiguration pour le management des opérateurs communs. Nous avons notamment montré l'importance que la gestion des ressources prend dans le fonctionnement des systèmes reconfigurables à base d'opérateurs communs. Le choix des ressources matérielles, avec les forts besoins de flexibilité qu'engendre la radio reconfigurable est un autre point clef de la définition de ces systèmes. Nous proposons dans cette section des modèles d'architectures traitant de ces deux parties essentielles à la définition d'un système à base d'opérateurs communs. Cette modélisation fonctionnelle permettra de mettre en œuvre la reconfiguration des opérateurs communs implantés sur des ressources matérielles hétérogènes. Nous proposons ici le modèle de gestion de configuration hiérarchique issu de notre analyse de la problématique du management d'opérateurs dans la section 2.3. Le modèle de gestion simplifié tel qu'illustré sur la figure 2.9 possède deux niveaux de hiérarchie. Il permet de répondre aux besoins de reconfigurabilité des équipements en vue des différents types de changements de contexte (basculement de standard, de mode, adaptation à la situation du canal de transmission...). Nous choisissons donc une gestion adaptative des ressources et mixte (entre distribuée et centralisée) comme nous l'avons justifié dans la section 2.3.

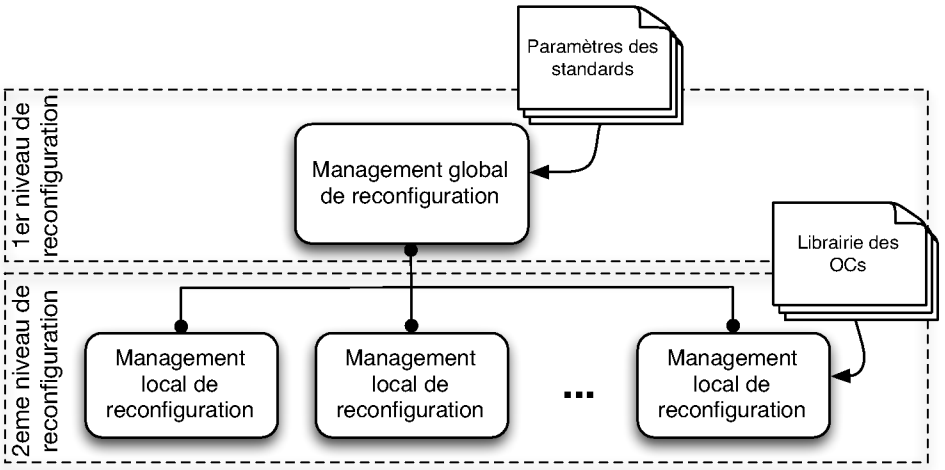


FIGURE 2.9 – Modèle de gestion de configuration des opérateurs communs

Le manager centralisé des opérateurs communs (couche logicielle) s'occupe essentiellement de générer le graphe des opérateurs communs et de définir les politiques de management. Ensuite, les mécanismes de mana-

gement des opérateurs (ressources matérielles) sont traités localement au deuxième niveau de configuration.
La figure 2.10 illustre un exemple d'architecture de management local de l'opérateur FFT/Viterbi. Dans cet exemple, un banc d'opérateurs est configuré pour définir les interconnexions et le partage des opérateurs entre les algorithmes FFT et Viterbi. Puisqu'il s'agit d'opérateurs de moyenne granularité (accélérateurs niveau fonctions), le banc d'opérateurs peut se connecter directement à la chaîne de traitement (via un réseau sur puce comme le montre l'exemple). Le manageur local de reconfiguration peut être implémenté en matériel pour ces opérateurs de haut niveau de granularité et ne nécessitant pas un grand degré de programmabilité (comme illustré sur le tableau 2.3). Nous traîterons en détails les mécanismes de management de l'opérateur FFT/Viterbi dans les chapitres suivants.

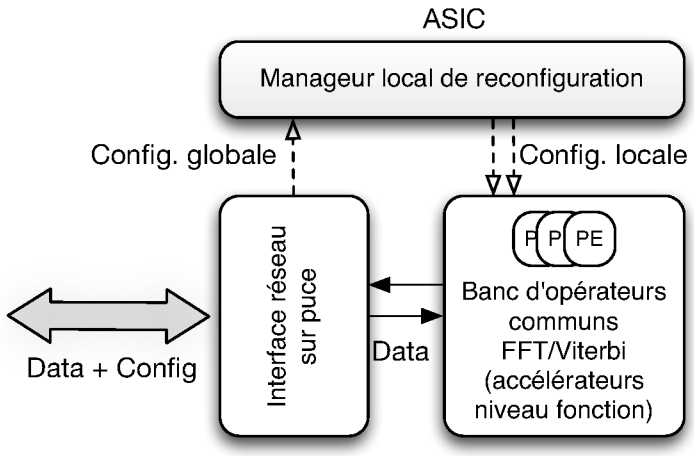


FIGURE 2.10 – Exemple d'architecture de management local de l'opérateur FFT/Viterbi

Pour les opérateurs de faible granularité comme le LFSR et le CORDIC, nous proposons une architecture de type ASIP où le banc d'opérateurs est appelé par les instructions du processeur de traitement de signal (Figure 2.11). Contrairement à l'exemple précèdent, les entrées/sorties du banc d'opérateurs sont connectées au processeur local qui exécute les fonctions qui les appellent.

La figure 2.12 illustre un exemple d'architecture globale de management d'opérateurs communs où les exemples d'architectures précédemment présentées sont connectées via un rséau sur puce au manager centralisé des opérateurs communs.

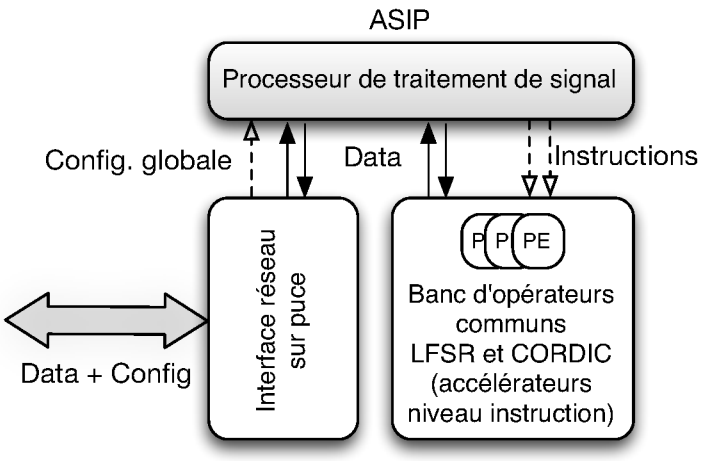


FIGURE 2.11 – Exemple d'architecture de management local des opérateurs LFSR et CORDIC

2.6 Conclusion

Un des éléments indispensables à la flexibilité des systèmes reconfigurable est le contrôle et le management de la configuration. La diversité des traitements en bande de base nécessite l'introduction d'une technique de management des ressources matérielles utilisant des éléments de traitements génériques comme les opérateurs communs.
Nous avons présenté dans ce chapitre une approche qui a pour objectif de permettre le déploiement des applications reconfigurables sur une plateforme matérielle hétérogène basée sur les opérateurs communs. Nous avons défini également le management des opérateurs communs et nous avons étudié l'implémentation de ces opérateurs en se basant essentiellement sur des évaluations de complexités pour quelques standards utilisant la modulation OFDM. Nous avons montré aussi que les algorithmes de FFT et décodage FEC sont des traitements lourds en terme de calcul. Donc, la définition d'une technique de gestion efficace des ressources entre ces familles d'algorithmes constituera un pas significatif en avant.
Dans la suite de ce manuscrit, nous présentons de nouvelles structures reconfigurable de l'opérateur commun FFT/Viterbi et discuterons de la mise en œuvre des mécanismes de gestion de cet opérateur avec des exemples concrets. Nous présenterons aussi des architectures permettant la gestion des ressources entre les deux algorithmes que nous implémenterons sur FPGA.

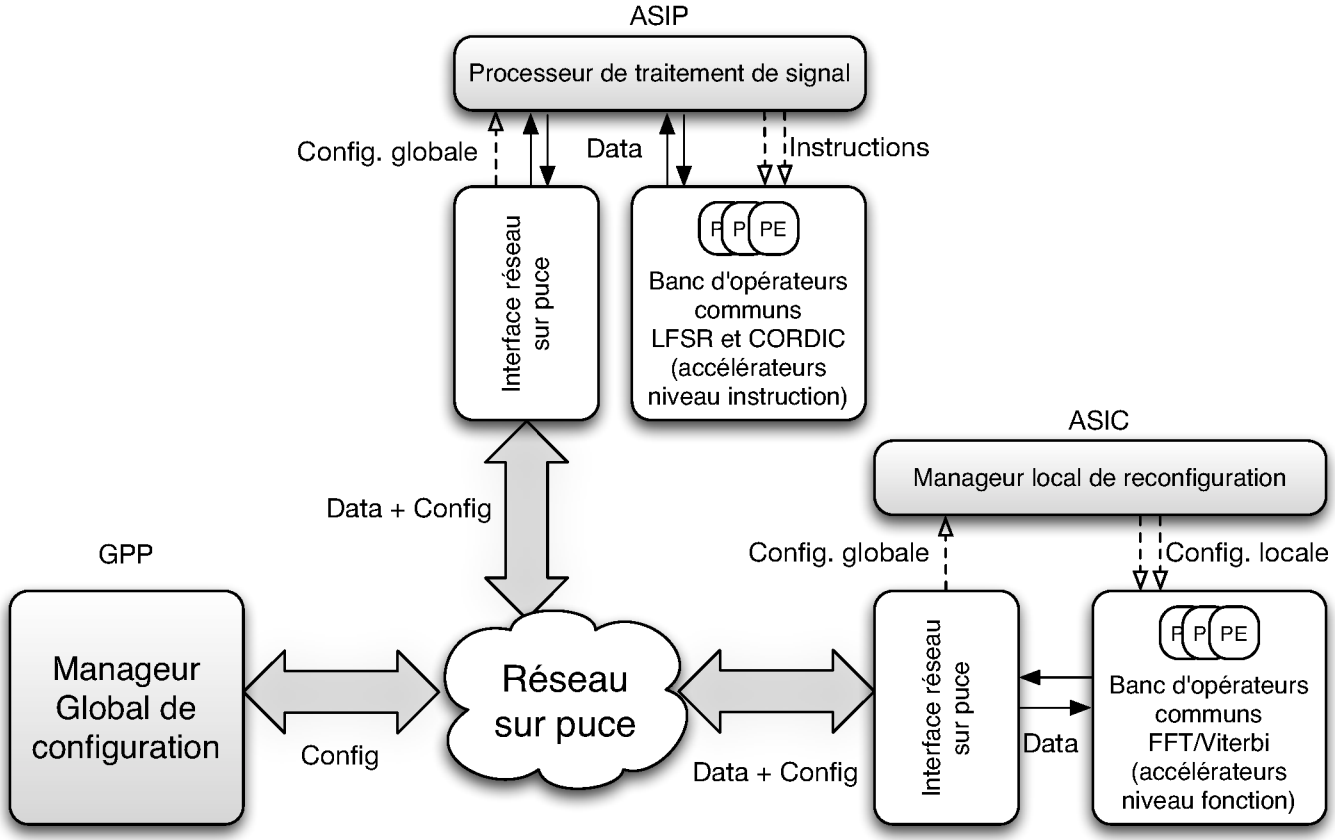


FIGURE 2.12 – Exemple d'architecture globale de gestion des opérateurs communs

Chapitre 3

L'opérateur commun FFT/Viterbi

Sommaire

3.1 Introduction 79
3.2 Similarités entre FFT et Viterbi 80
3.2.1 Structure du décodeur de Viterbi 80
3.2.2 Structure de la FFT 83
3.2.3 Vers une structure commune FFT/Viterbi 84
3.3 Premières structures de l'OC FFT/Viterbi 86
3.4 Structure optimisée pour l'OC FFT/Viterbi . . . 89
3.4.1 Calcul du papillon FFT Radix-2 (1er mode) 93
3.4.2 Calcul du ACS (2ème mode) 95
3.4.3 Calcul du BMC (3ème mode) 96
3.5 Comparaison des implémentations proposées . . . 98
3.6 Le jeu d'opérateurs FFT/FEC 100
3.7 Conclusion 100

3.1 Introduction

Les normes de télécommunication basées sur des modulations multi-porteuses utilisent les Transformées de Fourier Rapides (FFT), notamment dans les algorithmes de (dé)modulation d'égalisation, d'estimation du canal, de filtrage, etc...[24]. Outre la FFT, une grande partie des normes de télécommunication actuelles standardisent des codes convolutifs pour corriger les erreurs induites par le canal de transmission. Le décodage d'un code convolutif peut être effectué par l'algorithme de Viterbi. Ainsi, en exploitant l'approche des opérateurs communs, une mutualisation des implémentations FFT et Viterbi peut être envisagée. Nous avons montré dans le chapitre

précédent que les algorithmes de FFT et de Viterbi sont des traitements lourds en terme de calcul et que nous espérons gagner en complexité en mutualisant leurs implémentations.
Dans ce chapitre, nous proposons des cellules de traitement matériel reconfigurables pour les algorithmes FFT et Viterbi, capables de prendre en compte leurs différences fonctionnelles et de s'adapter à l'utilisation qui en est faite. Ces travaux sur l'opérateur FFT/Viterbi viendront compléter le jeu d'opérateurs présenté dans le premier chapitre.
Nous commençons par étudier les similarités structurelles entre les algorithmes FFT et Viterbi. Ensuite nous présentons deux premières architectures parallèles du papillon commun pour arriver à une architecture en pipeline plus efficace que les deux premières. Nous discutons à la fin de ce chapitre les comparaisons entre les cellules de l'opérateur FFT/Viterbi proposé avec celles de la littérature.

3.2 Similarités entre les algorithmes FFT et Viterbi

Dans cette section, nous étudions les similarités entre les algorithmes FFT et Viterbi. En effet, des similarités structurelles entre les deux algorithmes ont été observées auparavant dans plusieurs travaux de la littérature mais n'ont pas été exploitées pour développer des structures communes de traitement pour les deux algorithmes. Dans [70] certaines similitudes structurelles ont été brièvement soulignées pour implémenter l'algorithme de Viterbi adapté aux implémentations Hypercube. Dans [71], certaines de ces similitudes ont été exploitées afin de développer une architecture en pipeline pour décodeurs de Viterbi comparable à celle de la FFT Single-path Delay Feedback (SDF). Dans [72], les similitudes entre les deux algorithmes ont été utilisées pour simplifier le stockage des métriques de chemin pour le décodeur de Viterbi et de réduire sa taille. Pour étudier plus en détails les similitudes structurelles entre la FFT et algorithme de Viterbi, nous partons d'une topologie classique de type treillis du décodeur de Viterbi et nous montrons comment il est possible d'établir une correspondance entre la connectivité des deux algorithmes.

3.2.1 Structure du décodeur de Viterbi

Les traitements effectués par l'algorithme de Viterbi peuvent être scindés en trois blocs différents (Figure 3.1) [73]. Le premier module BMC (Branch Metric Calculation) effectue le calcul des métriques de branche pour chaque état du treillis. Le deuxième module ACS (Add Compare Select) effectue le calcul des métriques de chemin et la sélection du chemin survivant, pour chaque état du treillis. Il comporte une rétroaction puisque le calcul des

métriques de chemin à un instant t nécessite de connaître les métriques de chemin à l'instant $t-1$. Le troisième module SMM (Survivor Memory Management) mémorise les choix effectués à propos des chemins survivants, pour chaque état du treillis afin de restituer le résultat de décodage en fin de trame.

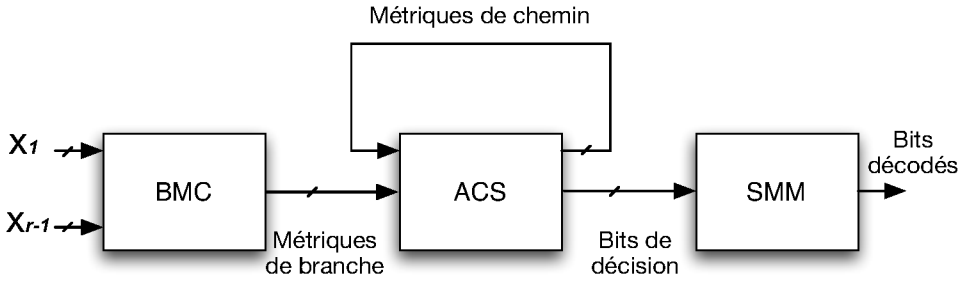


FIGURE 3.1 – Structure globale d'un décodeur de Viterbi

Pour bien illustrer le fonctionnement de ces différents modules, nous considérons un décodeur de Viterbi $(k, 1/r)$ où k est la longueur de contrainte et $1/r$ est le rendement du code. Les principales étapes de l'algorithme peuvent être illustrées par un graphe en treillis comme le montre la Figure 3.2. Les éléments à la verticale du treillis sont nommés états (nombre d'états $= 2^{k-1}$). On leur associe des valeurs numériques pm (métriques de chemin) et les poids des arêtes sont donnés par les métriques de branche bm. Le schéma partiel du treillis de la Figure 3.2 est appelé papillon [74].

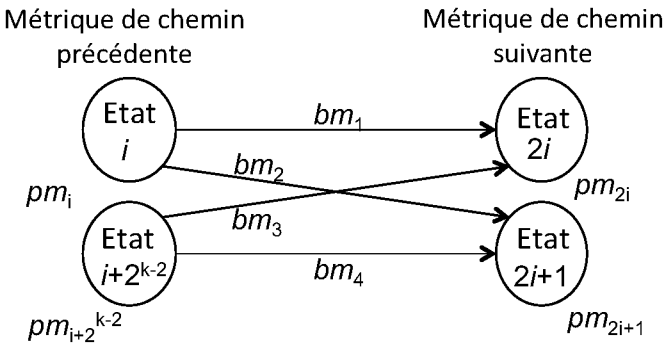


FIGURE 3.2 – Le papillon Viterbi et l'actualisation des métriques de chemin

Les performances d'un code convolutif sont liées à sa distance libre [75]. On la définit à l'aide de la distance de Hamming, qui correspond au nombre de bits qui diffèrent entre deux mots binaires. La distance libre est égale à a plus petite distance de Hamming qui existe entre deux chemins quelconques du treillis. Le choix de cette distance pour effectuer le calcul des métriques

de branche est conditionné par le format des entrées du décodeur. Lorsque les entrées du décodeur sont binaires, on parle dans ce cas d'entrées dures ou de décodage ferme. La distance de Hamming, qui consiste à compter le nombre de différences entre des éléments binaires, est bien adaptée à ce cas de figure. Mais il est également possible de réaliser un décodage pondéré [75]. L'entrée du décodeur est alors constituée par une suite d'échantillons analogiques. Pour une implémentation matérielle, ces échantillons sont en réalité quantifiés sur un certain nombre de bits. On parle alors d'entrées souples. La distance pertinente à utiliser dans ce cas est la distance euclidienne. Le décodage à entrées souples fournit de meilleurs résultats que le décodage à entrées dures. [76]
Nous expliquons dans la suite de ce paragraphe le calcul de ces distances et par la suite le calcul des métriques de branche et de chemin. Soit $({x_1}^t, {x_2}^t, \ldots, {x_r}^t)$ le r-uplet d'échantillons quantifiés à l'entrée du décodeur à l'instant t (où $1/r$ est le rendement du code). Soit $(b_1, b_2, \ldots, b_r)$ le r-uplet d'éléments binaires associé à une branche donnée du treillis. Le carré de la distance euclidienne entre l'échantillon reçu et la valeur théorique associée à la transition est :

$$d^{t2} = \sum_{i=1}^{r} ({x^t}_i - {b^t}_i)^2 = \sum_{i=1}^{r} ({x^t}_i^2 - 2 \times {x^t}_i {b^t}_i + {b^t}_i^2) \tag{3.1}$$

Or, en utilisant une modulation de phase, les valeurs théoriques $(b_1, b_2, \ldots, b_r)$ associées à une transition donnée sont alors à valeur dans $\{-1; +1\}$.
Donc dans l'expression de la distance, les termes au carré sont identiques pour toutes les transitions à un instant donné. Or, dans l'algorithme de Viterbi, le résultat de décodage ne dépend que de la différence entre les métriques. Ainsi, l'expression de la distance se simplifie considérablement en éliminant les termes au carré et le facteur -2 du double produit. Au lieu de chercher à minimiser l'expression précédente, il est donc plus simple de chercher à maximiser l'expression suivante, où $bm([j], [j'])^t$ représente la métrique de branche associée à la transition entre l'état j à l'instant $t-1$ et l'état j' à l'instant t :

$$bm([j], [j'])^t = \sum_{i=1}^{r} {x^t}_i \times {b^t}_i \tag{3.2}$$

On remarque que le calcul des distances ci-dessus se réduit à de simples sommes ou soustractions entre les entrées souples du décodeur, étant donné que les éléments $(b_1, b_2, \ldots, b_r)$ sont à valeur dans $\{-1; +1\}$ [76].
On peut également noter que le choix de la distance à utiliser intervient uniquement dans le calcul des métriques de branche et reste donc totalement indépendant des autres traitements effectues par le décodeur. Ainsi, il est possible de modifier la distance utilisée pour ce calcul sans aucune conséquence sur le reste du décodeur.

Le décodeur de Viterbi met à jour ses métriques de chemins à partir des métriques de branche selon les équations 3.3 et 3.4 pour $i = 0, \ldots, 2^{k-2} - 1$.

$$pm_{2i} = Max(pm_i + bm_1, pm_{i+2^{k-2}} + bm_3) \tag{3.3}$$

$$pm_{2i+1} = Max(pm_i + bm_2, pm_{i+2^{k-2}} + bm_4) \tag{3.4}$$

Les équations équations 3.3 et 3.4 décrivent les opérations effectuées dans la fonction Add-Compare-Select (ACS) du décodeur de Viterbi. La figure 3.3 montre les dépendances entre les états au cours du processus d'actualisation des métriques de chemin.

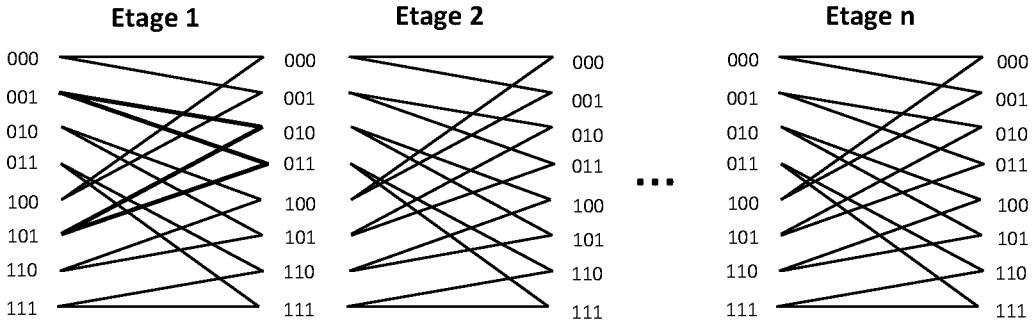


FIGURE 3.3 – La représentation du treillis Viterbi (pour $k = 4$)

3.2.2 Structure de la FFT

La même structure treillis/papillon peut également être identifiée pour l'algorithme de FFT utilisée pour réduire le coût total de calcul de la transformée de Fourier discrète DFT. La représentation habituelle de la FFT radix-2, la forme la plus commune de l'algorithme de Cooley-Tukey [77], est le diagramme en treillis qui partitionne récursivement une DFT en éléments de calculs intermédiaires également nommés papillons. L'algorithme FFT radix-2 met à jour les échantillons intermédiaires selon la Figure 3.4.
Nous illustrons dans l'équation 3.5 les opérations réalisées par un papillon FFT Radix-2 Décimation-en-Temps (DIT) où les $y_k^{[j]}$ sont les échantillions à l'entrée du papillion et W_n^k les « Twiddle factors » de la FFT.

$$\begin{cases} y_{k+1}^{[0]} = y_k^{[0]} + y_k^{[1]} \times W_n^k \\ y_{k+1}^{[1]} = y_k^{[0]} - y_k^{[1]} \times W_n^k \end{cases} \tag{3.5}$$

On propose l'écriture complexe suivante pour les entrées du papillon FFT (Figure 3.4) dans les équations 3.6 et 3.7.

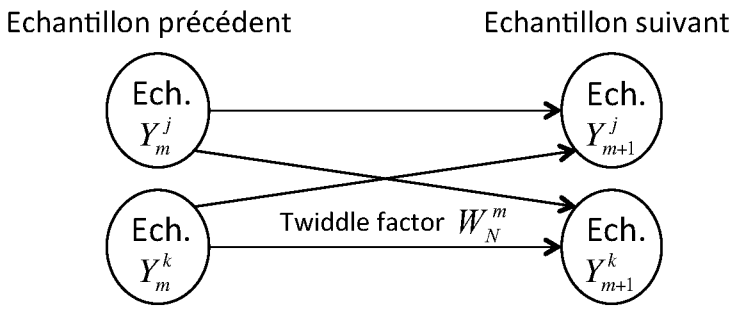


FIGURE 3.4 – Le papillon FFT Radix-2

$$\begin{cases} y_k^{[1]} = a + ib \\ W_n^k = c + id \\ y_k^{[0]} = e + if \end{cases} \tag{3.6}$$

$$\begin{cases} y_k^{[0]} + y_k^{[1]} \times W_n^k = e + (ac - bd) + i.[f + (ad + bc)] \\ y_k^{[0]} - y_k^{[1]} \times W_n^k = e - (ac - bd) + i.[f - (ad + bc)] \end{cases} \tag{3.7}$$

Les dépendances entre les papillons dans la FFT radix-2 sont montrées dans le diagramme en treillis de la Figure 3.5.

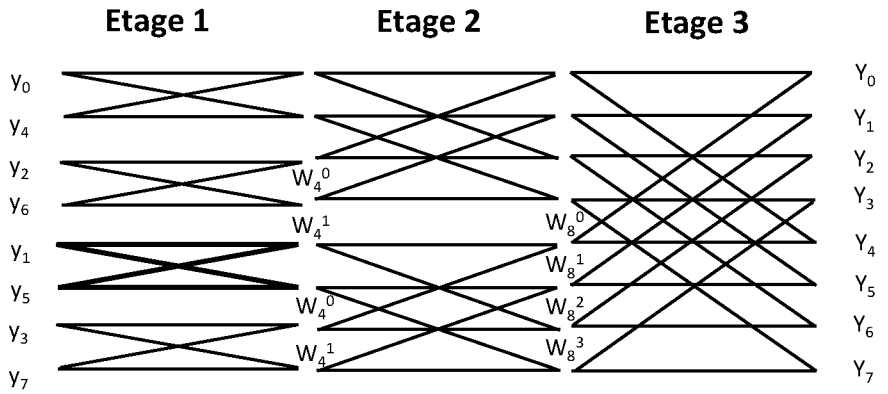


FIGURE 3.5 – La représentation du treillis FFT (pour $N = 8$)

3.2.3 Vers une structure commune FFT/Viterbi

L'idée principale de notre travail est de reconstruire le treillis de Viterbi et de l'adapter au treillis de l'algorithme de FFT. La figure 3.6 illustre une représentation du treillis Viterbi adaptée aux interconnexions des papillons FFT [72]. De la manière similaire que l'algorithme de FFT, cette

représentation du treillis de l'algorithme de Viterbi génère une désorganisation des états du décodeur dans la mémoire pendant un ensemble fini d'étages. Après $k-1$ étages, la disposition initiale des états du décodeur est rétablie [70]. En pratique, le nombre d'étapes (paramètre n dans figure 3.3) nécessaires pour obtenir un rendement suffisant pour décoder est d'environ $n \approx 5(k-1)$ [70].

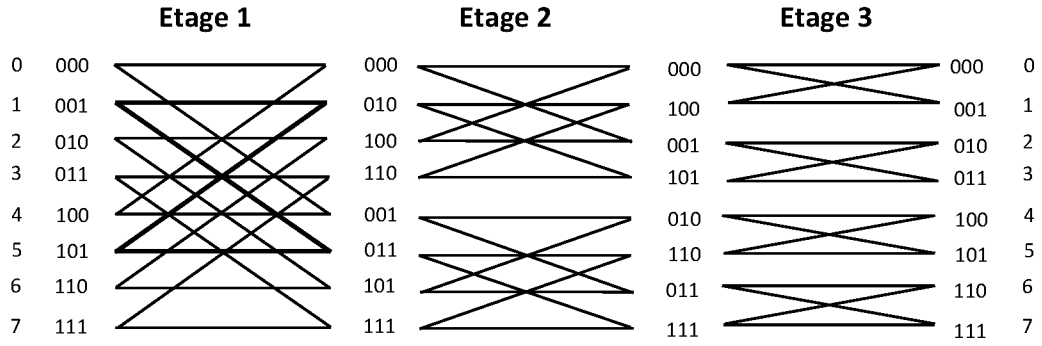


FIGURE 3.6 – Reprśentation commune du treillis FFT et Viterbi (Taille FFT = 8, Langueur de contrainte Viterbi = 4, Décimation en fréquence)

Ces propriétés sont exprimées dans l'équation 3.8 où $a^m = (a^m_{k-2}, a^m_{k-3}, \ldots, a^m_0)$ représente l'emplacement mémoire d'un état dans un étage m et $\pi(j)$ la fonction de permutation.

$$a^{m+1} = a^{m+1}_{\pi(j)}, \quad j = 0, \ldots, k-2 \tag{3.8}$$

La permutation des états du treillis peut être exprimée par une fonction de mapping donnée par 3.9.

$$\pi(j) = j + 1 \mod (k-1) \tag{3.9}$$

L'équation 3.9 montre que la permutation des états du décodeur après chaque étape peut facilement être identifiée comme une opération de rotation à gauche ou sur a^m. L'exemple illustré sur la figure 3.6 confirme cette propriété et montre que ces permutations sont équivalentes à celles de l'algorithme FFT-DIF (Décimation-en-Fréquence). La génération des adresses des papillons à chaque étape est illustrée à la figure 3.7.

À partir de la figure 3.6, une deuxième représentation du treillis de Viterbi basée sur la FFT-DIT (Décimation-en-Temps) est proposée sur la figure 3.8. Avec cette représentation, la première colonne des états du décodeur est référencée dans un ordre des bits inversés pour chaque étage $k-1$. De la même façon que pour la FFT-DIT, la conversion de l'ordre binaire inversé à l'ordre naturel peut être facilement mise en oeuvre en inversant les bits de

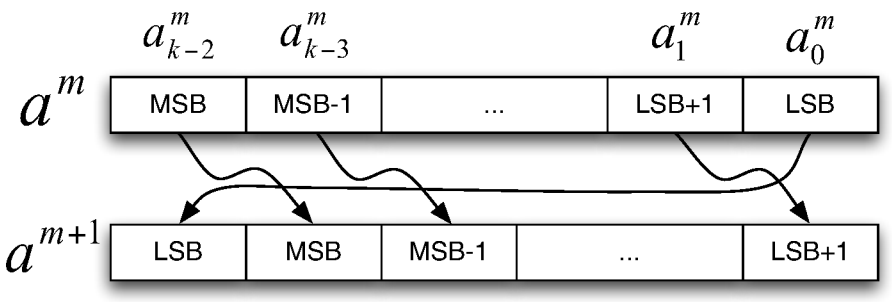


FIGURE 3.7 – Génération des adresses pour les papillons Viterbi et FFT

l'index (0 à $k-2$) de chaque élément. Après le premier étage, les mêmes permutations que pour le treillis DIF sont utilisées. Dans les sections suivantes, seules la représentation en DIT est considérée car elle présente de meilleures propriétés en terme de longueur de mot finis [78].

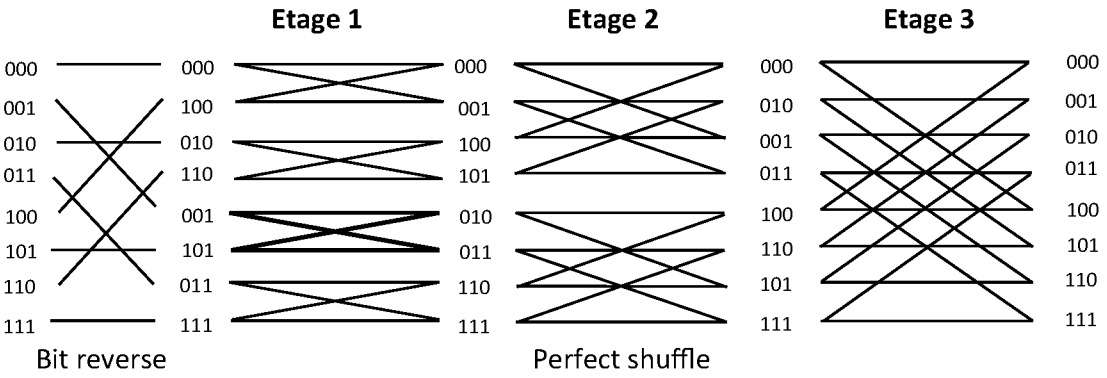


FIGURE 3.8 – Représentation commune du treillis FFT et Viterbi (Taille FFT = 8, Langueur de contrainte Viterbi = 4, Décimation en temps)

3.3 Premières structures de l'opérateur commun FFT/Viterbi

Dans [79] et [80] (Figures 3.9 et 3.10), des solutions avec l'approche Opérateurs Communs pour la mutualisation des algorithmes FFT et Viterbi ont été proposés. Ces solutions, se basant sur une architecture parallélisée de papillon FFT Radix-2 à 3 multiplications, conviennent lorsque l'optimisation en vitesse de traitement est privilégiée par rapport à la complexité et la consommation électrique du design.
En effet, dans les deux solutions proposées, l'utilisation des multiplexeurs pour commuter entre les deux modes de fonctionnement ajoute un coût

matériel non négligeable au circuit (4 multiplexeurs supplémentaires pour [79] et 11 pour [80]). Aussi, le fait que le papillon FFT nécessite beaucoup plus d'opérations d'Addition/Soustraction que celui de l'algorithme de Viterbi a conduit à la proposition de deux opérateurs :

- Le premier, proposé en [79], permet la mutualisation du papillon Viterbi (module ACS pour les calculs des chemins [76]) et une partie du papillon FFT Radix-2 (Figure 3.9).
- Le deuxième, proposé en [80], permet la mutualisation du papillon FFT en entier avec les opérations de calculs de branches BMC et les opérations de calcul des chemins ACS [76] (Figure 3.10).

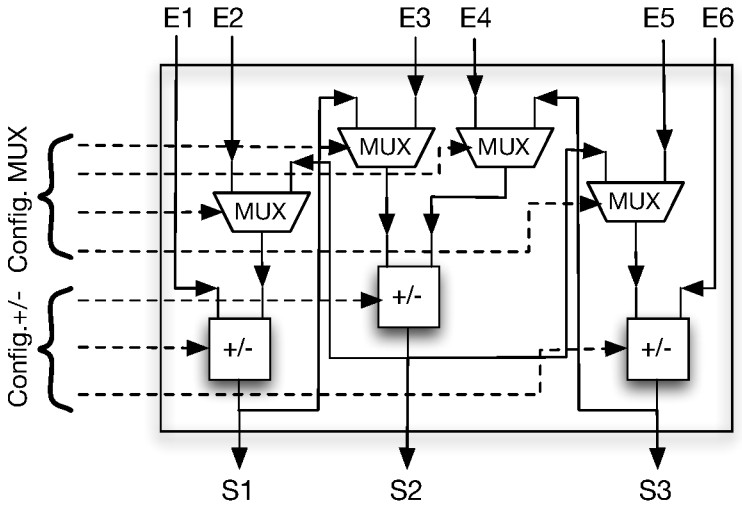


FIGURE 3.9 – Opérateur AS3 présenté dans [79]

En pratique, le nombre des modules nécessaires pour le calcul des métriques de branches est très inférieur au nombre des modules ACS pour l'algorithme de Viterbi. Par conséquent, l'utilisation de l'opérateur proposé dans [80] va engendrer un gaspillage de ressources significatif dû à la duplication du module BMC dans chaque cellule.
D'autre part, l'opérateur commun proposé dans [79] et [80] se base sur une architecture du papillon FFT Radix-2 à trois multiplieurs réels (Figure 3.11). Cette architecture du papillon FFT Radix-2 a été retenue parmi deux architectures (à 3 et 4 multiplieurs) dans le seul but de réduire le nombre de multiplications réelles (Figure 3.11). Cependant, ce choix n'a pas pris en compte d'autres paramètres importants comme la consommation électrique, la surface et la fréquence maximale de fonctionnement pour le papillon FFT indépendamment de l'architecture globale de l'Opérateur Commun. Ceci est d'autant plus dommageable que le papillon est connu pour être l'élément qui

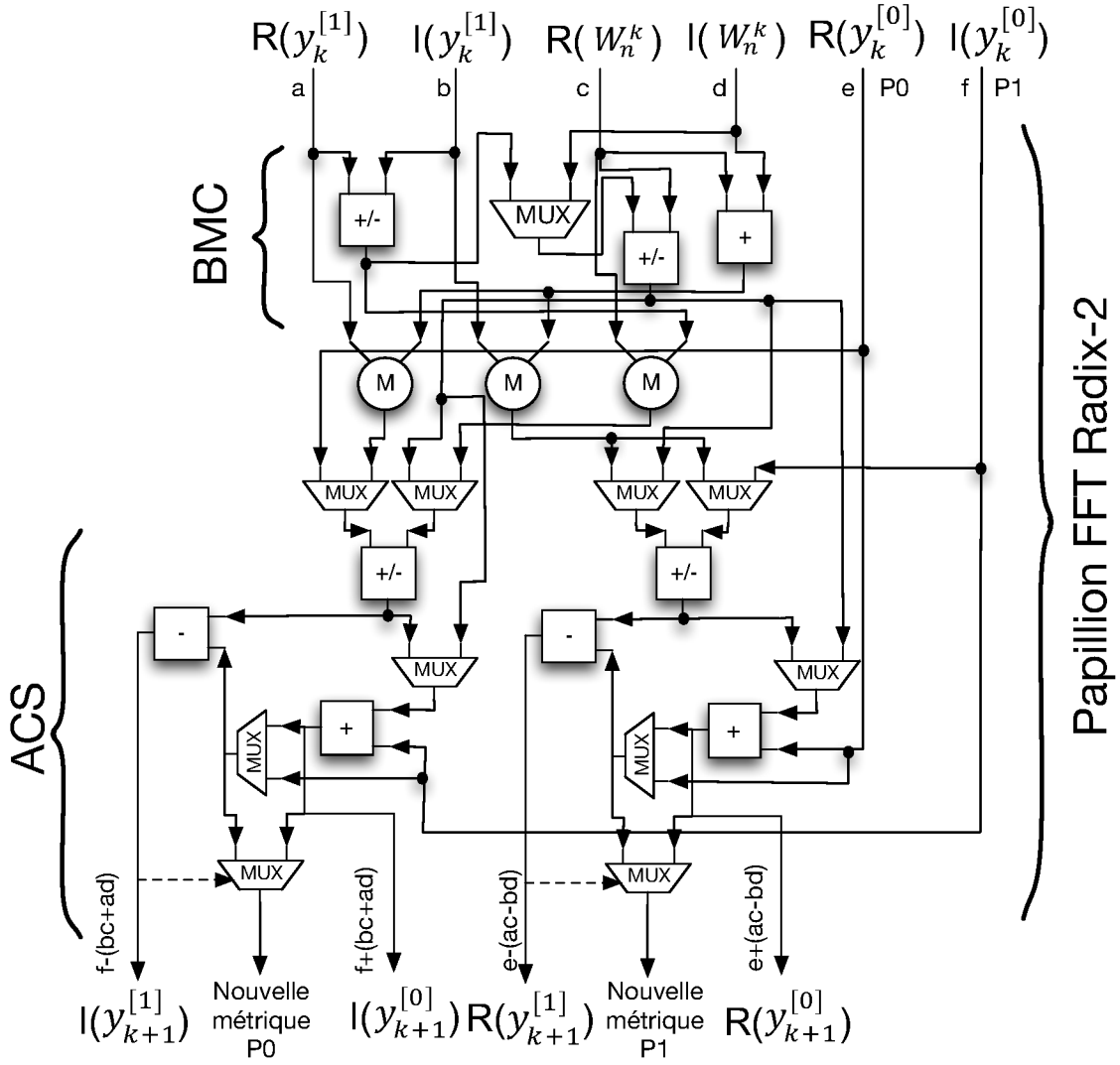
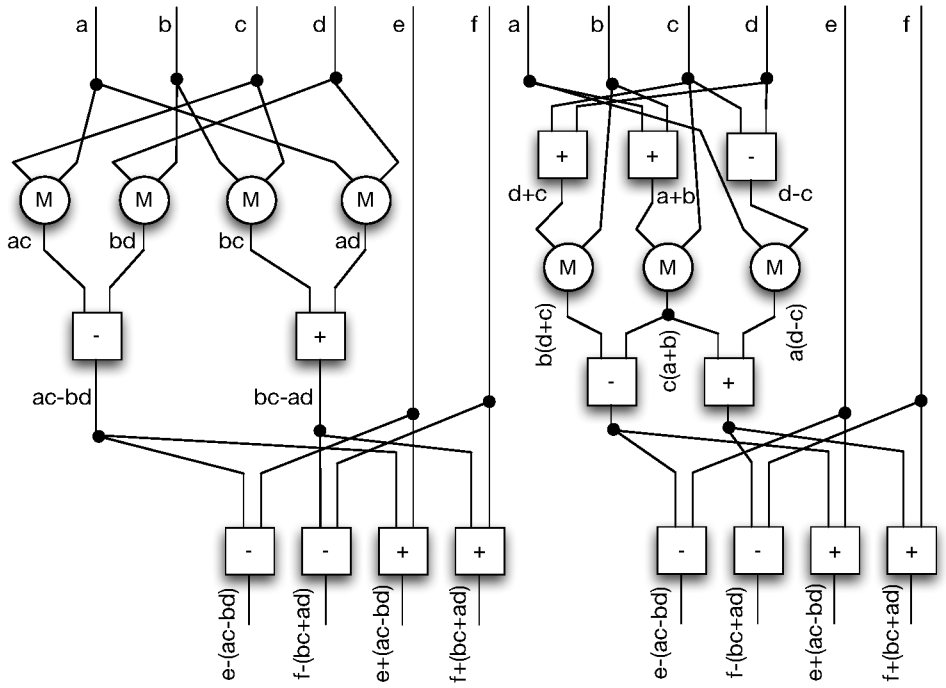


FIGURE 3.10 – Opérateur commun FFT/Viterbi présenté dans [80]

consomme le plus dans les processeurs FFT [81].
Dans [82] et [83] des comparaisons des architectures des papillons de la FFT ont démontré que l'architecture à 3 multiplications n'est pas la mieux adaptée pour les calculs des papillons FFT Radix-2. En effet, l'architecture à 3 multiplications permet de réduire très légèrement la surface et la consommation électrique du circuit par rapport à une architecture à 4 multiplications, mais diminue la fréquence maximale du fonctionnement du papillon. En réalité, la suppression d'un multiplieur ajoute un étage supplémentaire d'additionneurs/soustracteurs, qui induit un gain en surface réduit par rapport à la perte en vitesse liée au délai ajouté par ces éléments (Figure 3.11).



(a) Architecture à 4 multiplications (b) Architecture à 3 multiplications

FIGURE 3.11 – Architectures du papillion FFT Radix-2 (2 et 3 multiplications réelles)

3.4 Proposition d'une structure optimisée pour l'opérateur commun FFT/Viterbi

La référence [83] propose une architecture pipelinée du papillon FFT Radix-2 (figure 3.12 où R est un registre) qui permet de réduire très significativement la surface et la consommation de circuit ($\approx 50\%$) par rapport à

l'architecture à 3 multiplications. Il a été montré également que la réduction de la vitesse de fonctionnement reste limitée ($5ns$ à 50 MHz et $0.01ns$ à 100 MHz sur ASIC 0.11μ). Tableau 3.1 (adapté de [82]) illustre la réduction de la consommation attendue basée sur les réalisations des papillons FFT radix-2 (deux et trois multiplicateurs) sur une technologie ASIC 0.11μ.

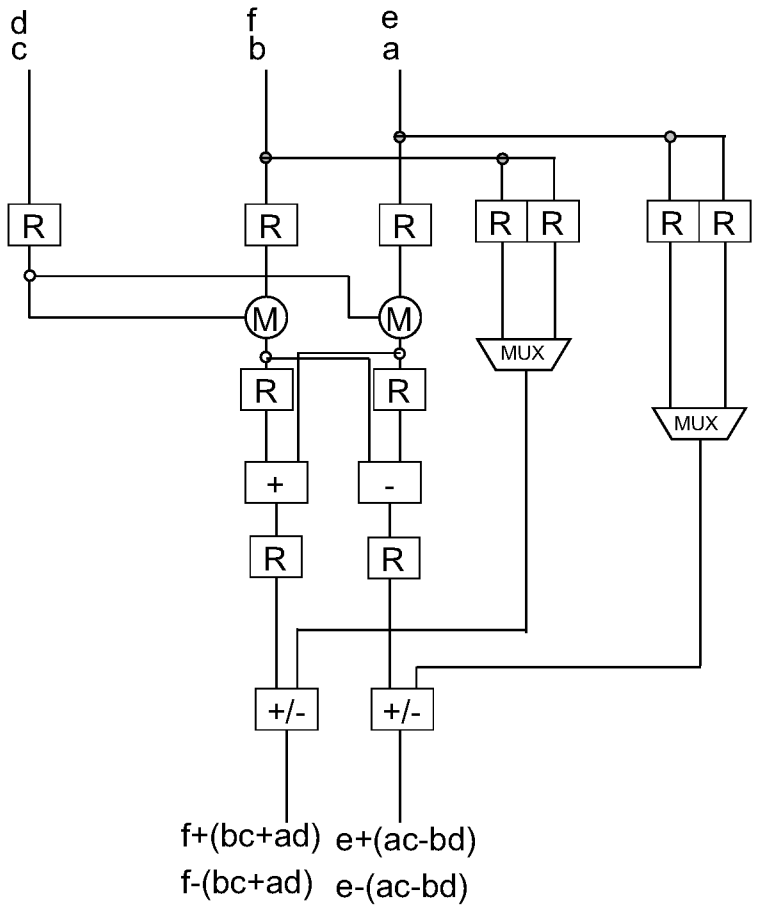


FIGURE 3.12 – Architecture du papillon FFT Radix-2 à 2 multiplications [83]

Le présent paragraphe a pour but d'utiliser l'architecture du papillon FFT à deux multiplications pour construire un nouvel opérateur FFT/Viterbi avec les objectifs suivants :

- Minimiser le gaspillage des ressources en modifiant très légèrement le papillon FFT Radix-2 (architecture à 2 multiplications) pour supporter les opérations de calculs de branches BMC et des calculs de chemins ACS pour l'algorithme de Viterbi.

TABLE 3.1 – Comparaisons de consommation entre les architectures de papillons à 2 et à 3 multiplieurs (adapté de[82])

Papillion FFT Radix-2	Architecture à 3 multiplication		Architecture à 2 multiplication	
Freq. [MHz]	50	100	50	100
Conso. [mW]	2.96	4.22	1.44	2.13

- Séparer les modules de calculs de branches BMC et des calculs de chemins ACS pour l'algorithme de Viterbi afin d'éviter de dupliquer inutilement les ressources matérielles liées au calcul des BMC.
- Conserver une paramétrisation simple pour un basculement rapide entre les deux algorithmes et pour limiter le surcoût matériel nécessaire à la reconfiguration.
- Garder une fréquence maximale du circuit comparable à celle des opérateurs proposés dans [79] et [80].

A partir de l'architecture du papillon FFT Radix-2 présentée dans la figure 3.12, nous proposons dans la figure 3.13 une nouvelle architecture de l'opérateur FFT/Viterbi. Cet opérateur est issu d'une architecture prévue pour le calcul de la FFT. Cette architecture a été modifiée pour étendre ses possibilités afin d'être utilisée par un décodage de Viterbi pour les opérations des calculs de branches et des calculs de chemins. Ce nouvel opérateur permet de basculer entre une implémentation dédiée FFT ou une implémentation spécifique au décodage de Viterbi sans surcout supplémentaire en nombre d'opérateurs par rapport au papillon FFT Radix-2. La seule modification apportée par rapport au papillon FFT Radix-2 (figure 3.12) est colorée dans la figure 3.13.
Le principe consiste à utiliser les ressources déjà implémentées pour les papillons FFT afin d'effectuer les calculs des métriques pour l'algorithme de Viterbi, tout en limitant le surcoût lié à cet ajout en flexibilité. En effet, les ressources globales nécessaires pour les calculs de l'algorithme de FFT sont très importantes par rapport à celles du Viterbi. En se référant aux standards actuels, la taille des FFT varie entre 64 et 2048, ce qui nécessite entre 32 et 1024 papillons « logiques » par étape du treillis. Les degrés des polynômes générateurs des codes convolutionels varient entre 4 et 8, ce qui implique d'utiliser entre 8 et 128 papillons « logiques ». Le nombre limité de papillon de l'algorithme de Viterbi est à l'origine de notre volonté d'utiliser la grande quantité de papillons FFT pour les réaliser. Le surcoût induit par la capacité à traiter l'algorithme de Viterbi est donc largement démultiplié compte tenu du grand nombre de cellules utilisées pour la FFT.
Dans cette nouvelle architecture, nous séparons les calculs des branches et

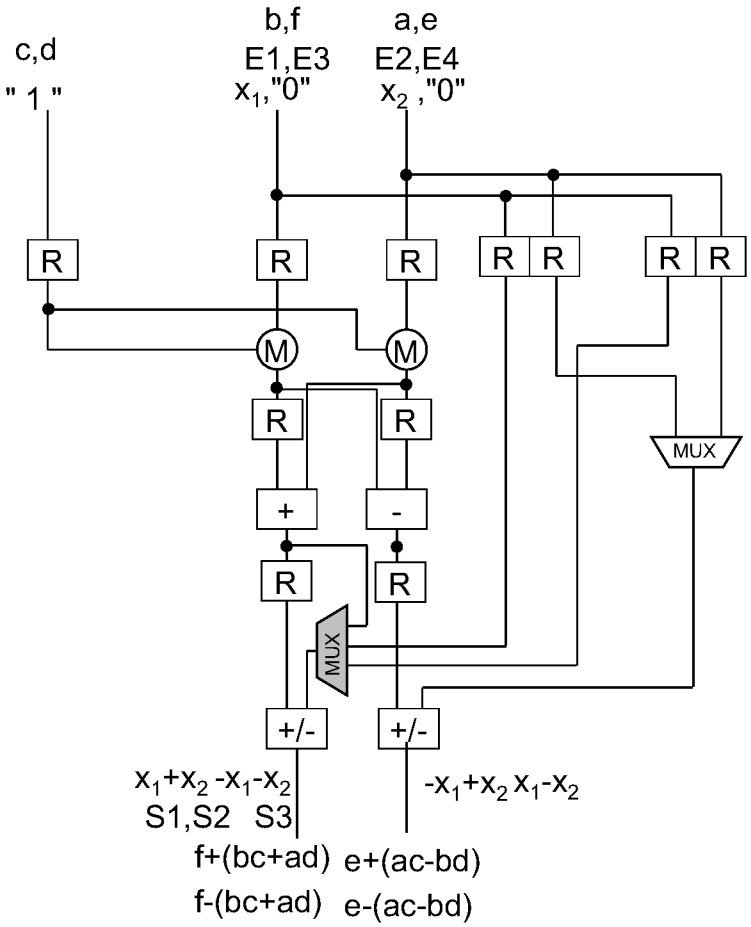


FIGURE 3.13 – Cellule en pipeline proposée pour les algorithmes FFT et Viterbi

les calculs des chemins pour le décodeur Viterbi. En effet, nous proposons de définir trois modes de fonctionnement au niveau de la cellule de traitement commune aux deux algorithmes :

- **1er mode :** Calcul du papillon FFT Radix-2.
- **2ème mode :** Calcul des métriques de chemin (ACS) pour le décodeur de Viterbi.
- **3ème mode :** Calcul des métriques de branches (BMC) pour le décodeur de Viterbi.

La volonté de séparer les traitements BMC et ACS contrairement à l'opérateur commun proposé dans [80] vient du fait qu'en pratique les calculs des chemins nécessitent beaucoup plus d'instances que les calculs des branches. L'objectif est de permettre une utilisation accrue des ressources offertes par les papillons FFT. La nouvelle cellule reconfigurable que nous proposons permet d'effectuer les deux opérations nécessaires au décodage Viterbi séparément, ce qui induit une meilleure répartition des ressources selon les besoins des standards ciblés. Nous décrivons les 3 modes de fonctionnement dans les paragraphes suivants.

3.4.1 Calcul du papillon FFT Radix-2 (1er mode)

La figure 3.14 illustre les opérations complexes nécessaires pour le calcul du papillon FFT Radix-2.

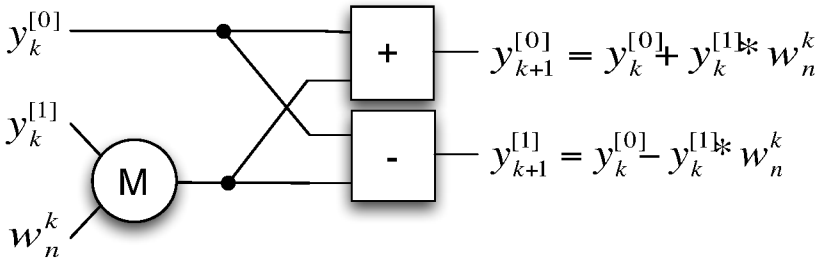
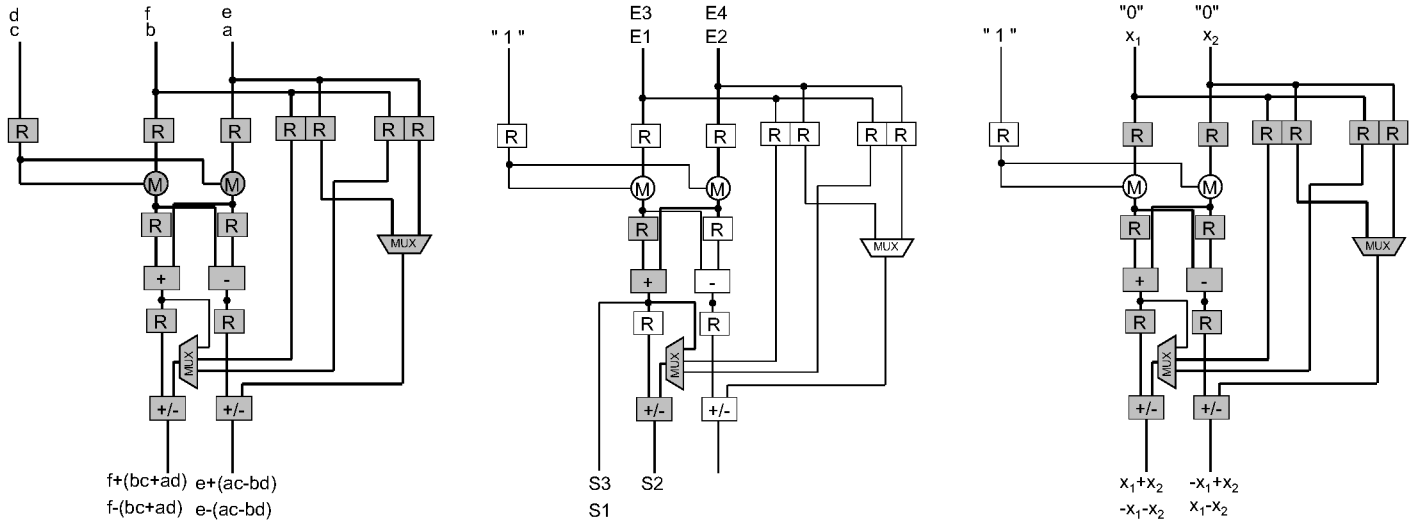


FIGURE 3.14 – Structure du papillon FFT Radix-2

Les opérations illustrées dans la figure 3.14 et l'équation 3.7 peuvent être effectuées d'une façon « pipelinée » en 2 cycles d'horloge selon l'équation 3.10 comme il a été montré dans [82].

$$\begin{array}{c} Cycle\#1 : \ A_1 = ac; \quad B_1 = bc \\ Cycle\#2 : \ A_2 = ad; \quad B_2 = bd \\ \left\{ \begin{array}{l} y_k^{[0]} + y_k^{[1]} \times W_n^k = e + (A_1 - B_2) + i.[f + (A_2 + B_1)] \\ y_k^{[0]} - y_k^{[1]} \times W_n^k = e - (A_1 - B_2) + i.[f - (A_2 + B_1)] \end{array} \right. \end{array} \tag{3.10}$$



(a) Ressources utilisés pour le calcul du papillon FFT Radix-2

(b) Ressources utilisées pour le calcul des métriques des chemins pour le décodeur de Viterbi

(c) Ressources utilisées pour le calcul des métriques des branches pour le décodeur de Viterbi

FIGURE 3.15 – Les trois modes de fonctionnement de la cellule proposée

La figure 3.15a illustre l'implémentation de l'équation 3.10 et les ressources utilisées pour le calcul du papillon FFT Radix-2. On peut voir que ce calcul consomme 100% des ressources de l'opérateur, ce qui répond au premier objectif que nous nous étions fixé , à savoir la réduction du gaspillage des ressources consommées par la paramétrisation du circuit.

3.4.2 Calcul des métriques de chemin pour le décodeur de Viterbi (2ème mode)

En se focalisant maintenant sur le papillon de l'algorithme de Viterbi, nous pouvons dans un premier temps écrire les équations mathématiques qui le régissent. La figure 3.16 illustre l'écriture mathématique des opérations réalisées par le papillon Viterbi.
Ainsi, à partir de la figure 3.16 , nous faisons apparaitre pour chacune des

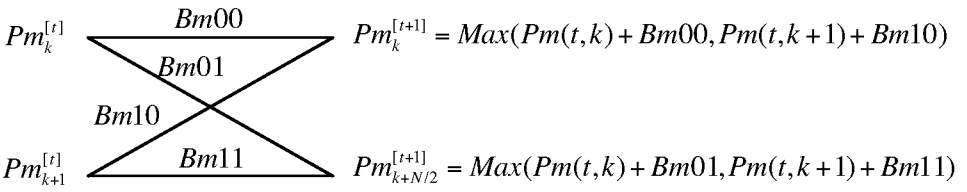


FIGURE 3.16 – Implémentation des opérations de calcul de chemins

métriques de chemins, 2 additionneurs et 1 soustracteur, soit 6 opérations d'Additions/Soustractions pour effectuer les comparaisons et calculer les résultats.
Par la suite, les relations entre les entrées/sorties du papillon de l'algorithme de Viterbi peuvent s'écrire :

$$\begin{cases} S_1 = E_3 + 3_4 \\ S_2 = E_3 + E_4 - E_1 - E_2 \\ S_3 = E_1 + E_2 \end{cases} \tag{3.11}$$

où les S_i et E_i sont les entrées/sorties du modèle du papillon définis sur la figure 3.16 (Étage de séléction final n'est pas représenté sur la figure).

Ainsi, à partir des opérations illustrées sur la figure 3.16, nous pouvons proposer une solution pour réaliser ces opérations par le papillon FFT comme illustré sur la figure 3.15b.
Le calcul des métriques des chemins nécessaires pour le décodeur de Viterbi se fait séquentiellement en utilisant la même architecture que pour le papillon FFT. En effet, les données entrent et sortent du module ACS d'une façon successive via un bus commun pour chacune des deux entrées/sorties. L'étage de séléction final est implémenté à la sortie de l'opérateur commun

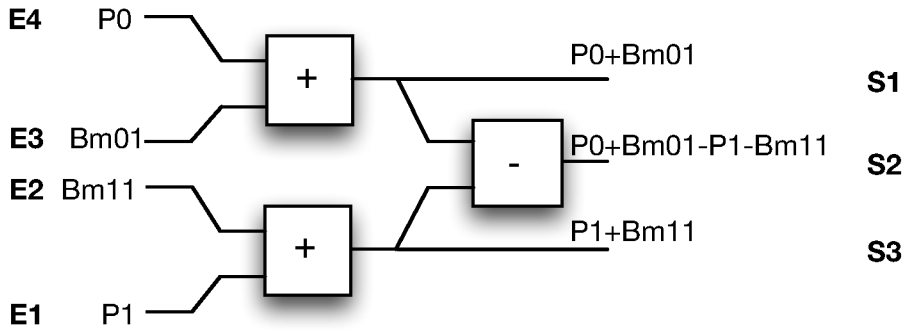


FIGURE 3.17 – Entrées/Sorties du papillion de Viterbi

via un banc de multiplexeurs.
Il est clair qu'au niveau de la cellule, la vitesse du traitement sera inférieure d'environ 45% par rapport aux architectures parallèles proposées dans [79] et [80]. En revanche, nous exploitons l'idée d'une utilisation efficace des ressources de l'algorithme FFT puisque dans les standards de communications actuels, les ressources allouées pour les FFT sont très supérieures à celles du Viterbi. Cela permettra d'utiliser plusieurs cellules en parallèle s'il y a un besoin d'accélérer le décodage Viterbi. La reconfiguration de la cellule pour les opérations des calculs des chemins se fait très facilement en injectant un élément neutre à l'entrée des « twiddle factors » des papillons FFT pour contourner les opérations de multiplications et en configurant le multiplexeur coloré dans la figure 3.15b.

3.4.3 Calcul des branches pour le décodeur Viterbi (3ème mode)

Le papillon du décodeur de Viterbi nécessite outre le calcul des métriques de chemins, un module de calcul de branches (BMC) qui calcule les distances associées à chaque état du treillis, afin d'évaluer l'exactitude des données reçues par rapport à une transition du treillis. Comme nous avons montré précédemment dans l'équation 3.2, le calcul des métriques de branches se réduit à de simples sommes ou soustractions entre les entrées souples du décodeur. Ce module BMC est traditionnellement réalisé comme illustré dans la figure 3.18.
Ces opérations BMC consistent en des opérations simples d'additions et soustractions et leurs opposés comme le montre l'équation 3.12.

$$\begin{cases} Bm_{00} = x_1 + x_2 \\ Bm_{01} = x_1 - x_2 \\ Bm_{10} = -x_1 - x_2 \\ Bm_{11} = -x_1 + x_2 \end{cases} \tag{3.12}$$

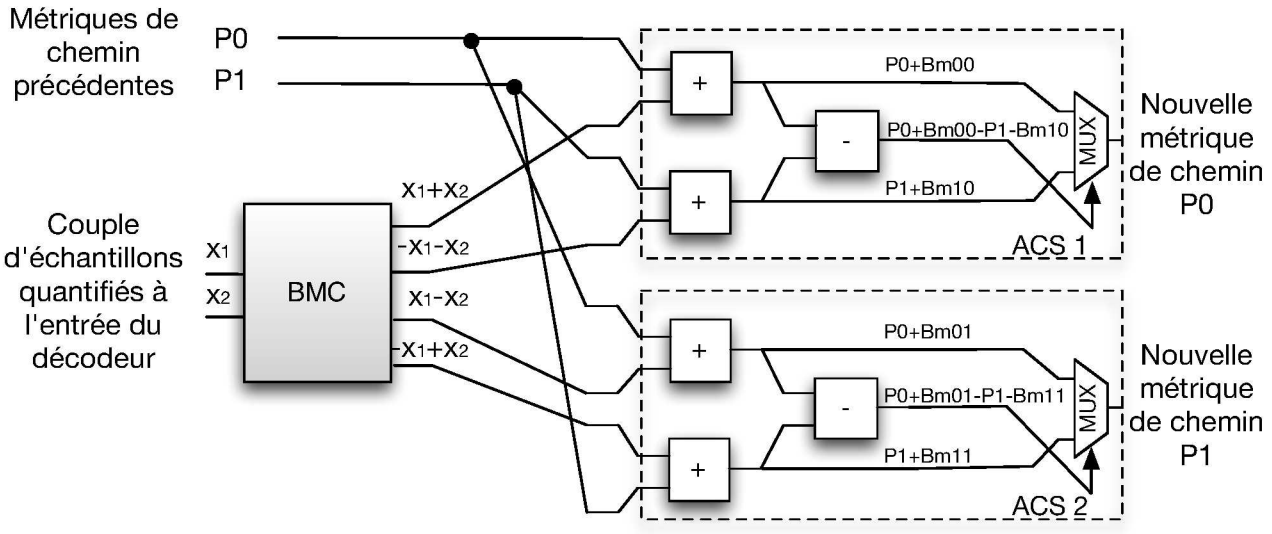


FIGURE 3.18 – Représentation schématique des opérations nécessaires pour le calcul des métriques des branches et des chemins pour le décodeur de Viterbi

Ainsi, nous pouvons proposer une solution pour réaliser ces opérations par le papillon FFT comme illustré sur la figure 3.15c.
Le calcul des métriques des branches nécessaires pour le décodeur de Viterbi se fait aussi séquentiellement en utilisant la même architecture que pour le papillon FFT et le module ACS. Dans ce cas, la reconfiguration est très simple, et consiste simplement à injecter un élément « neutre » à l'entrée des « twiddle factors » des papillons FFT pour contourner les opérations de multiplications et ajouter des éléments « nuls » à l'entrée tous les deux échantillons. En outre, en travaillant sur des données réelles au niveau des entrées de l'opérateur proposé, nous pouvons traiter un algorithme de Viterbi à entrées « souples » ou à entrées « dures ».
Avec ces trois modes de fonctionnements, nous pouvons ainsi proposer une solution pour l'utilisation de cette cellule avec l'approche du Banc d'Opérateurs Communs (BOC) définie en [25]. L'idée consiste à implémenter un réseau de cellules élémentaires que l'on pourra utiliser pour la majorité des standards de communications actuels et qui selon l'application ou le mode ciblé, se reconfigure pour effectuer des calculs de FFT ou Viterbi ou les deux à la fois. Cette solution permet non seulement d'économiser les ressources du système (en complexité et en consommation) mais permet aussi d'obtenir une architecture régulière permettant son utilisation dans la réalisation d'une architecture redondante [14] (tolérante aux fautes ou aux variabilités du processus technologique [84] [85] [86]).

3.5 Comparaison des implémentations proposées

Dans cette section, nous comparons les implémentations de l'opérateur commun proposé (figure 3.13) avec l'opórateur [80] et des cellules reconfigurables de la littérature [87] et [88]. Dans [88], un coprocesseur reconfigurable pour les systèmes de communications (RCC) a été proposé. L'élément de traitement présenté (PE) [88] fournit un débit élevé et une très grande flexibilité pour couvrir une large gamme d'algorithmes. D'autre part, dans [87], la surface du circuit a été réduite pour augmenter le débit et la flexibilité de la cellule, où un élément de traitement basé sur des opérations de multiplication et d'accumulation RMAC-PE a été proposé. Dans notre travail, nous avons suivi la même démarche que dans [87], mais en offrant un meilleur débit et moins de portes au détriment de la flexibilité. Nous avons limité l'utilisation de notre PE pour les algorithmes de FFT et Viterbi.
Dans le tableau 3.2, nous comparons les opérateurs communs avec d'autres cellules reconfigurables [87] [88]. Comme le montre le tableau, la cellule proposée fournit un gain important de point de vue de la complexité [88] avec une bonne performance en termes de nombre d'opérations par cycle. Cette réduction de débit peut être compensée par la réutilisation des opérateurs

physiquement implémentés par multiplexage temporel. En effet, [73] a montré que la limitation du nombre de papillons physiques peut se faire avec une gestion de multiplexage temporel. Ainsi, l'opérateur commun proposé peut être exploré dans le cadre de la technique du Banc d'Opérateurs Communs (BOC) (comme présenté dans [25]). Cette technique peut être prise en compte dans la recherche de l'optimisation de l'utilisation des opérateurs communs.
En plus des améliorations de complexité, la réalisation en pipeline de l'opérateur

TABLE 3.2 – Comparaisons de complexité des opérateurs FFT/Viterbi

		OC parallèle [80]	OC proposé Fig.3.13	RMAC-PE [87]	RCC [88]
Complexité (Gate Count)		Reference	−27%	−26%	+103%
Opérations par Cycle	Papillon FFT Radix-2	1	0,5	0,33	1,5
	Métriques de chemin ACS	2	0,5	0,5	2
	Métriques de chemin BMC	1	0,5	0,33	4

FFT/Viterbi (Figure 3.13) permet d'obtenir une réduction en consommation considérable par rapport à la cellule parallèle (Figure 3.10). En effet, comme nous l'avons mentionné ci-dessus, la définition de la cellule est basée sur une architecture de papillon FFT à « faible-consommation » proposée dans [83] qui a été construite de manière à réduire la surface, les ressources et les activités de reconfiguration. Ainsi, comme illustré dans le tableau 3.1, environ 50% de la réduction de la consommation d'énergie peut être obtenue avec la réalisation en pipeline (basé sur deux multiplicateurs papillon FFT radix-2) par rapport à la réalisation parallèle (basé sur trois multiplicateurs). Cette réduction de la consommation électrique devrait même être observée au niveau de l'architecture globale, puisque les papillons dominent la consommation de l'énergie dans les FFTs et processeurs de Viterbi [81].

3.6 Le jeu d'opérateurs FFT/FEC

Les architectures de l'opérateur FFT/Viterbi présentés dans ce chapitre et celles présentés dans [89], peuvent être examinées ensemble pour former une bibliothèque d'opérateurs FFT/FEC (FEC pour Forward error correction). On peut représenter cette proposition par un graphe illustré dans la figure 3.19. En effet, nous avons rappelé dans le premier chapitre que la FFT peut être appelée par beaucoup de fonctions de traitement de signal et également par certaines opérations de décodage canal de codes en blocs, constituant ainsi le cœur de traitement des décodeurs FEC. Donc la définition d'une bibliothèque d'opérateurs commune aux décodeurs FEC et FFT, constituera un grand pas en avant vers la mutualisation de ces familles d'algorithmes. La définition de ce jeu d'opérateur donnera la possibilité de construire un banc d'opérateurs FFT/FEC comme celui fait pour les opérateurs LFSR [10]. Ce banc apportera plus d'efficacité en surface aux designs ainsi que l'avantage de la régularité de l'architecture. Cependant, la gestion et l'ordonnancement des opérateurs FFT/FEC présentent beaucoup plus de difficultés que celle du LFSR car qu'il s'agit de fonctions très différentes en terme de fonctionnement et d'échantillons à traiter. Le défit principal à confronter dans le prochain chapitre, sera de définir des mécanismes de gestion de ces opérateurs.

3.7 Conclusion

L'opérateur commun FFT/Viterbi trouve son application dans la majorité des standards de communications actuels où le codage canal et les transformées de Fourier sont utilisés. En effet, pour le décodage canal plusieurs travaux ont montré la possibilité de couvrir un large domaine d'utilisation avec les modules BMC et ACS avec des entrées « souples » et « dures » [76]. De même, l'algorithme de Viterbi a lors d'études passées été étendu au décodage des Turbo Codes [22]. D'un autre côté, des travaux récents ont montré que le papillon FFT Radix-2 peut être facilement utilisé comme une cellule de base pour réaliser d'autres types de Radix [83]. Ainsi, la cellule du papillon proposée peut être appliquée pour des opérations diversifiées. En [24], l'auteur montre que la FFT peut s'étendre à la corrélation, la génération de filtre FIR, l'estimation de canal ou la détection de plusieurs utilisateurs. Ainsi, l'architecture que nous proposons peut être utilisée par une majorité des fonctions de l'équipement multistandard et de ne pas se restreindre aux seuls blocs de décodage canal et de (dé)modulation (I)FFT. De plus, lorsqu'elle est utilisée dans le cadre des Banc des opérateurs communs (BOC) [25], la mutualisation des papillons FFT et Viterbi peut apporter les avantages des architectures régulières pour avoir un design plus tolérant aux fautes et moins dépendant des variabilités du processus technologique. Ainsi, les nom-

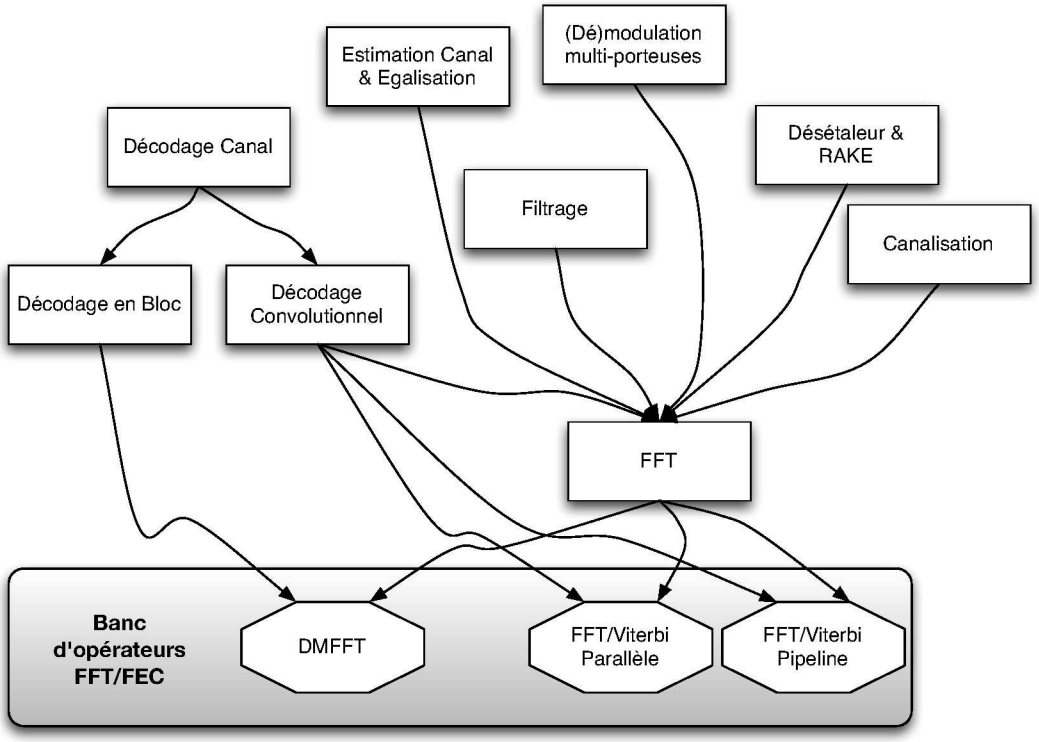


FIGURE 3.19 – Jeu d'opérateurs FFT/FEC

breux travaux concernant les réseaux des papillons FFT pour la tolérance aux fautes [84] [85] [86] peuvent être généralisés à un cercle d'applications plus large avec cette cellule commune pour les algorithmes FFT et Viterbi. Dans le chapitre suivant, nous traitons le management l'opérateur FFT/Viterbi en définissant une architecture générique à base d'un réseau d'interconnexions reconfigurables.

Chapitre 4

Mécanismes de management de l'operateur FFT/Viterbi

Sommaire

4.1 Introduction 103
4.2 Contraintes sur le mgmt. de l'OC FFT/Viterbi . 104
4.3 Architectures de mgmt. de l'OC FFT/Viterbi . . 106
4.3.1 Architecture en pipeline 107
4.3.2 Architecture à colonne partielle 109
4.3.3 Architecture séléctionnée 113
4.4 Mgmt. des interconnexions en colonne partielle . 113
4.4.1 Génération des interconnexions 114
4.4.2 Mgmt. des largeurs de chemin de données 114
4.5 Implémentation du processeur FFT/Viterbi . . . 115
4.5.1 Description de l'architecture implémentée 115
4.5.2 Résultats de l'implémentation 117
4.6 Conclusion 117

4.1 Introduction

Nous avons montré dans les chapitres précédents que l'exploitation de l'approche des opérateurs communs pour les algorithmes FFT et Viterbi peut permettre une mutualisation des implémentations. Nous avons conclu dans le chapitre 2 que les algorithmes de FFT et de Viterbi étaient des traitements lourds en terme de calcul et que nous espérons gagner en complexité en mutualisant leurs implémentations.
Nous avons étudié dans le deuxième chapitre le management des opérateurs communs d'une façon globale. Nous nous concentrons dans ce chapitre sur les mécanismes de gestion de l'opérateur FFT/Viterbi défini dans le chapitre

précèdent. Dans ce chapitre, nous proposons des architectures de traitement matériel reconfigurables pour les algorithmes FFT et Viterbi, capables de partager efficacement les ressources matérielles entre les deux algorithmes.
Notre objectif est d'assurer une affectation efficace des ressources entre les deux algorithmes FFT et Viterbi. Ainsi, nous commençons par l'exploitation des similitudes structurelles entre la FFT et algorithme de Viterbi (présentés dans le chapitre précédent) et nous proposons des architectures flexibles pour le processeur commun FFT/Viterbi.

4.2 Contraintes sur le management de l'opérateur FFT/Viterbi

Les contraintes de chaque standard imposent des besoins spécifiques pour l'architecture des processeurs FFT et des décodeurs de Viterbi. En effet, les tailles des FFT et le nombre d'états des décodeurs Viterbi varient beaucoup en fonction des standards avec des besoins de débits très différents. Cela implique une grande variation par rapport au nombre des opérateurs communs à implémenter physiquement. Le problème réside dans l'affectation et la connexion des cellules communes qui change en fonction des modes et des standards considérés. En effet, l'architecture du processeur doit être suffisamment générique pour supporter une affectation dynamique des opérateurs communs entre les deux algorithmes.
Une grande partie du problème réside dans la définition d'un réseau d'interconnexions reconfigurables entre les cellules [90]. En effet, le processeur commun doit gérer les flux de données entre les papillons qui change selon la taille de la FFT ou le nombre des états du décodeur Viterbi. Les figures 4.1 et 4.2 montrent deux exemples classiques des interconnexions entre les papillons pour une FFT 8 points et un décodeur Viterbi 8 états. En effet, les architectures classiques des décodeurs de Viterbi utilisent principalement des treillis avec des interconnexions à géométrie constante (Figure 4.1) alors que les treillis FFT utilisent généralement $\log_2(N) - 1$ étages d'interconnexions différents (Figure 4.2). Cependant, il existe des travaux dans la littérature où des implémentations du décodeur de Viterbi sont réalisées à base des architectures à géométrie variable [91] [71] et des implémentations FFT à base des architectures à géométrie constante [92] [93] [94]. Nous avons démontré dans le chapitre précèdent que les deux représentations en treillis étaient équivalentes et qu'il était possible de les mutualiser.
Pour gérer ces différentes interconnexions entre les deux algorithmes, il faut aussi considérer la variation des longueurs des chemins des données des deux algorithmes. En effet, dans les standards de communications actuels, l'algorithme de Viterbi fonctionne avec des échantillons codés sur 3 à 8 bits et la FFT avec des valeurs de 8 à 32 bits. Donc, il faut que l'architecture de traitement matériel reconfigurable pour les algorithmes FFT et Viterbi soit

capable de prendre en compte cette différence et de s'adapter à l'utilisation qui en est faite.

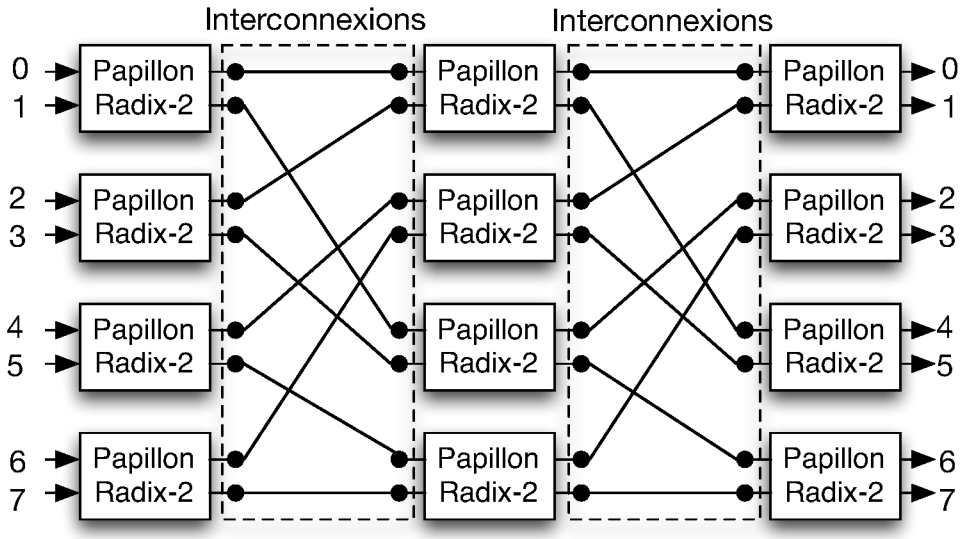


FIGURE 4.1 – Exemple d'architecture parallèle à géométrie constante principalement utilisée dans les décodeurs de Viterbi (8 états)

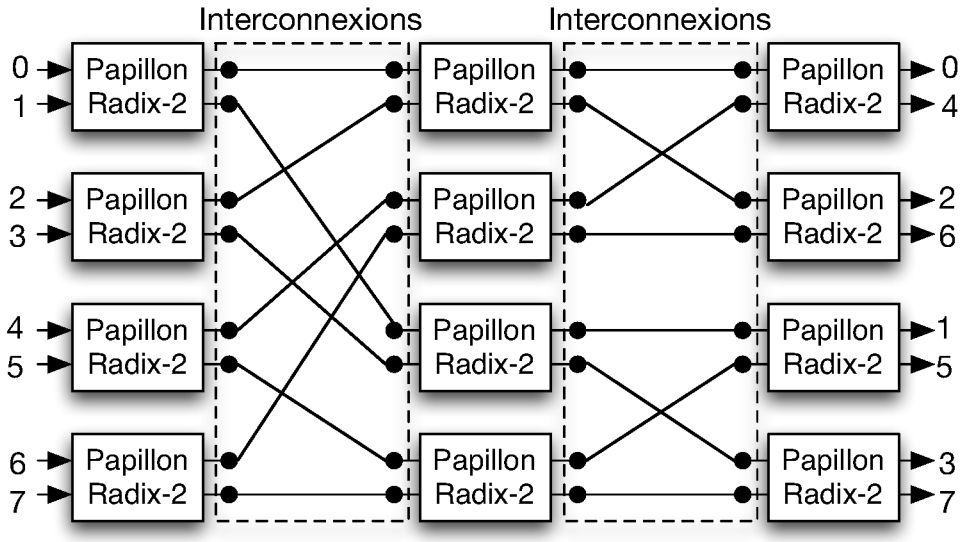


FIGURE 4.2 – Exemple d'architecture parallèle à géométrie variable principalement utilisée dans les algorithmes de FFT (8 points)

En plus du problème de variation de tailles pour les deux algorithmes, le processeur doit être capable d'adapter le débit du décodage de Viterbi ou FFT aux exigences de standard. En pratique le nombre de papillons physiquement implémentés est inférieur au nombre de papillons logiques utilisés par l'algorithme. Donc faire varier le débit à la sortie du processeur se ramène à faire varier le nombre de papillons physiques utilisés. Par conséquent, en plus des différences des permutations entre les deux algorithmes et les variations des tailles, le processeur doit permettre une variation du nombre des papillons physiques affectés à chaque algorithme et agir sur le degré de parallélisme des traitements.
En effet, l'architecture du processeur commun doit être souple et évolutive pour soutenir une affectation dynamique des ressources matérielles entre la FFT et algorithme de Viterbi. La figure 4.3 illustre l'allocation dynamique des ressources entre les algorithmes dans un exemple de design bi-standard.

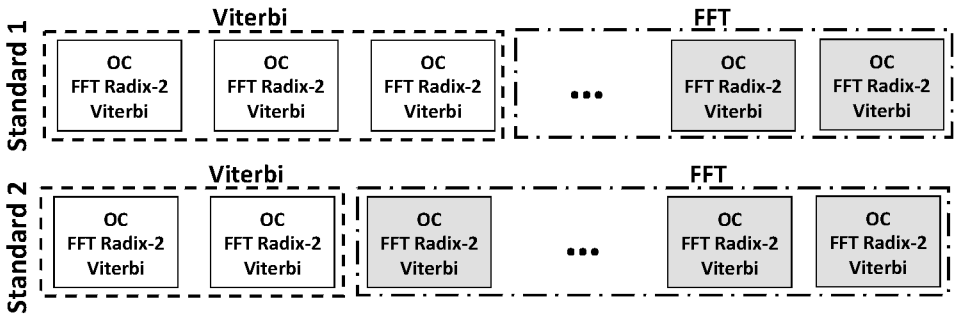


FIGURE 4.3 – Allocation dynamique des ressources entre les algorithmes FFT et Viterbi dans un exemple de design bi-standard

Afin de répondre aux besoins de la technique des opérateurs communs et supporter plusieurs standards, une architecture générique du processeur FFT/Viterbi permettant des tailles de FFT et une vitesse de décodage variable est nécessaire.

4.3 Architectures de management de l'opérateur commun FFT/Viterbi

Dans cette section, nous définissons les mécanismes de management des opérateurs communs FFT/Viterbi en partant des architectures parallèles classiques des algorithmes FFT et Viterbi. En général, les processeurs parallèles (celles utilisant plus d'un papillon physique) peuvent être divisés en quatre classes [95] : entièrement parallèles où tous les papillons sont implémentés physiquement (Figures 4.1 et 4.2), en pipeline (en cascade),

en colonnes ou en colonne partielle [96]. Dans les trois premières classes, le nombre d'unités de papillons implémentés physiquement dépendent de la longueur de la FFT ou du nombre d'états du décodeur, ce qui implique que la performance de la structure de calcul ne peut pas être adaptée selon les besoins de l'application. La flexibilité peut être obtenue en utilisant l'organisation en colonne partielle où le nombre d'unités « papillon » peut être modifié, et donc le coût de mise en œuvre (surface) peut être échangé contre le temps (vitesse de calcul). Dans la suite nous explorons les deux architectures les plus utilisées : l'architecture en pipeline (choisie en raison de sa simplicité) et l'architecture en colonne partielle (choisie en raison de sa flexibilité).

4.3.1 Architecture en pipeline

Les architectures en pipeline offrent un bon compromis entre la surface du circuit et son débit. Elles permettent d'avoir un contrôle simple des interconnexions avec des structures régulières. Plusieurs architectures pipelinées ont été initialement développées pour implémenter l'algorithme de FFT. Nous pouvons citer à titre d'exemple les architectures de types Multi-path Delay Commutator (MDC), Single-Path Delay Feedback (SDF) et Single-Path Delay Commutators [97] [98]. Dans ce paragraphe, nous nous focalisions sur l'architecture SDF Radix-2 la forme la plus utilisée pour les implémentations FFT pipelinées et qui convient aux opérateurs communs présentés dans le chapitre précédent. La figure 4.4 montre une architecture SDF Radix-2 d'une FFT DIF de taille $N = 8$ où les données sont traitées à travers les $\log_2 N$ étages de papillons. La rétroaction est utilisée avec les FIFOs pour stocker les résultats intermédiaires du treillis.

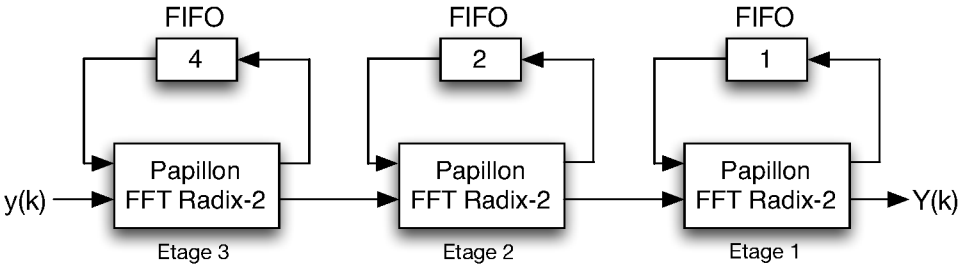


FIGURE 4.4 – Architecture en pipeline classique d'une FFT Radix-2 de taille $N = 8$

Nous avons démontré dans le deuxième chapitre que les algorithmes de FFT et Viterbi ont le même flux de données. Nous proposons donc d'adapter l'architecture SDF Radix-2 de la FFT aux traitements du décodeur de Viterbi. L'idée se base sur le treillis communs proposé dans la figure 3.6 du cha-

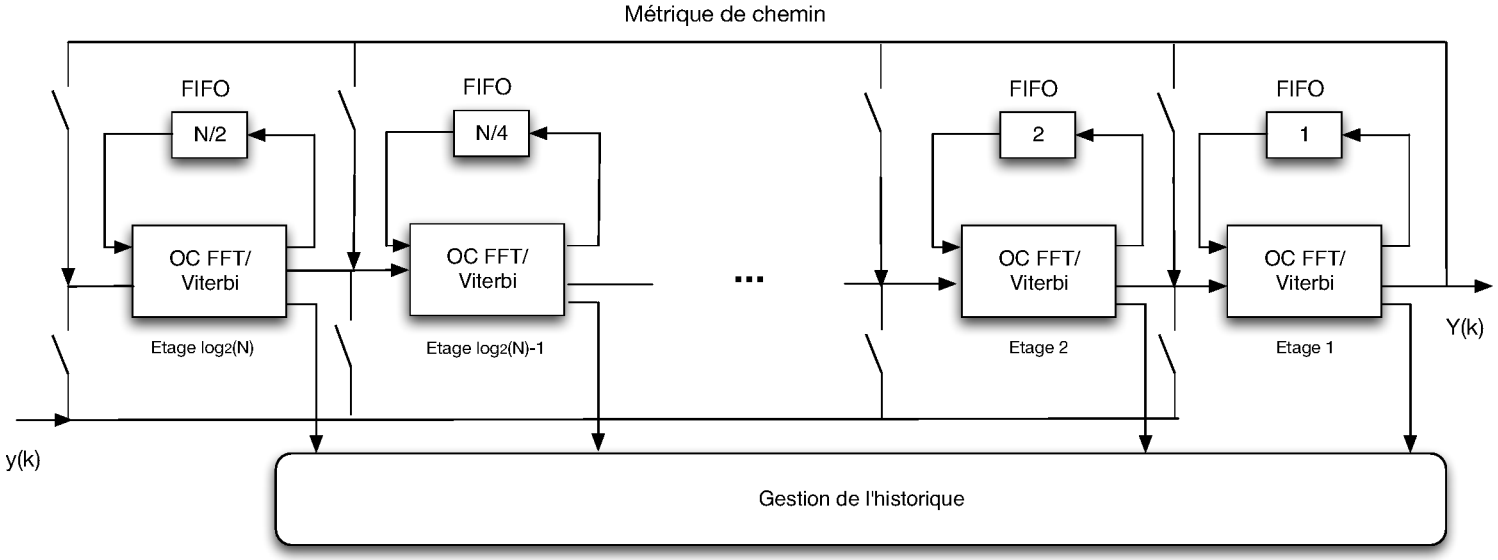


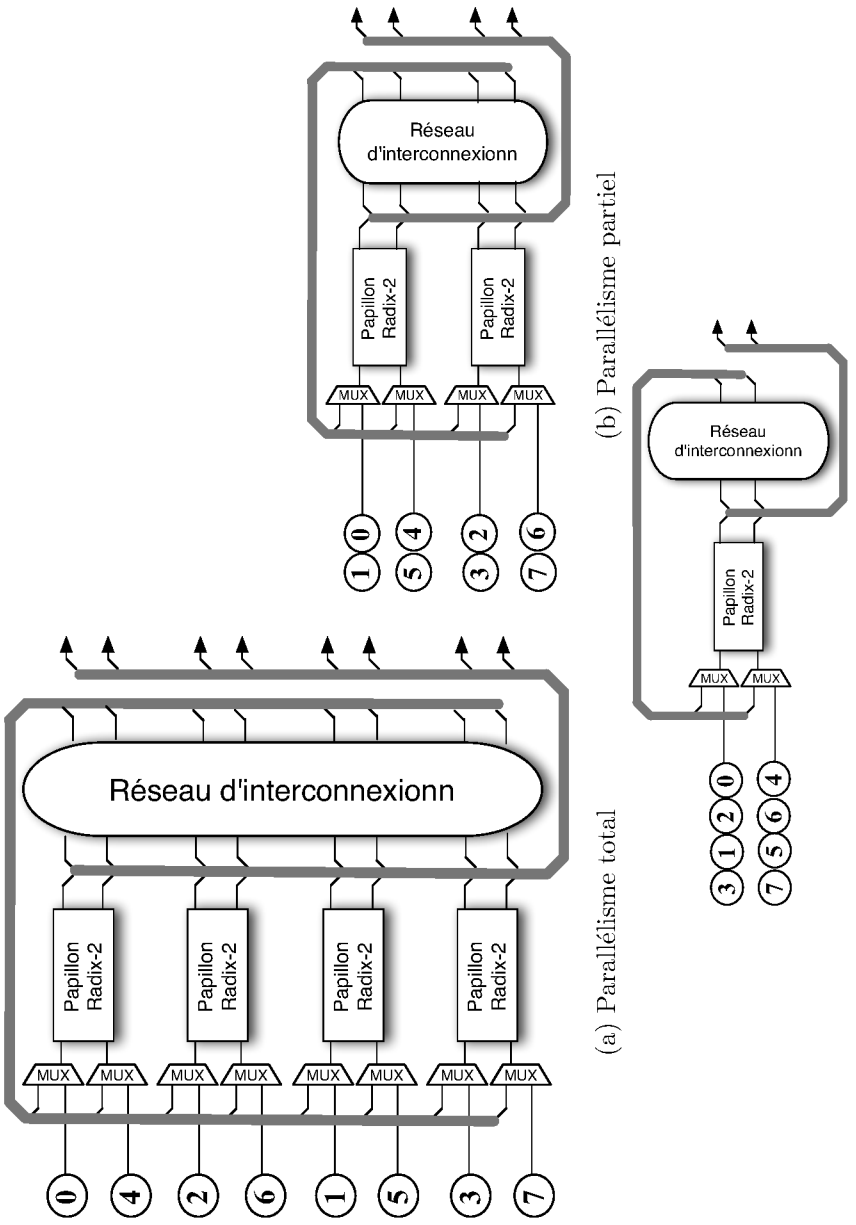
FIGURE 4.5 – Architecture en pipeline proposée pour le management de l'opérateur FFT/Viterbi

pitre précédent qui permet d'obtenir les mêmes interconnexions (géométrie variable) entre les papillons des deux algorithmes. Nous avons montré que le treillis communs permet d'utiliser les mêmes architectures que pour la FFT à condition d'accepter le brouillage des données dans les étages intermédiaires. Nous proposons dans la figure 4.5 une architecture de gestion des opérateurs communs FFT/Viterbi basée sur l'architecture présentée dans la figure 4.4. L'architecture de gestion se base sur l'ajout d'un module mémoire pour la gestion de l'historique des bits des décisions pour les opérations ACS. Une boucle de rétroaction est ajoutée aussi par rapport à l'architecture FFT puisque le calcul de ces métriques de chemin à un instant t nécessite de connaitre les métriques de chemin à l'instant $t-1$. Afin de rendre l'architecture adaptable à plusieurs tailles de FFT et plusieurs longueurs de contrainte pour Viterbi, nous ajoutons switchs pour sélectionner le niveau des entrées et la boucle de retour des métriques de chemin par rapport à l'étage du papillon et les FIFOs. En effet pour traiter une FFT de taille N, il suffit d'activer l'entrée des échantillons au niveau de l'étage $\log_2 N$ de l'architecture. Pour une configuration Viterbi de longueur de contrainte k la boucle de retour des métriques de chemin se fait à l'étage $k-1$. Pour cette architecture, les opérateurs FFT/Viterbi parallèles (Figure 3.10) conviennent mieux que les opérateurs en pipeline proposés (figure 3.13). En effet, le débit de ce type d'architectures est limité par le débit des papillons implémentés [91]. Donc le choix des opérateurs parallèles aura un effet important sur le débit à la sortie du processeur commun. En plus, les opérateurs en pipeline traitent un seul échantillon FFT (éventuellement une seule métrique de chemin) à la fois alors que dans l'architecture proposée nous traitons deux échantillons à la fois. Alors les opérateurs communs parallèles sont les plus adaptés pour l'architecture en pipeline.
L'architecture de gestion des opérateurs FFT/Viterbi illustrée sur la figure 4.5 est une architecture simple ne nécessitant pas beaucoup de mémoire ni un réseau d'interconnexion complexe. Par contre, cette architecture ne permet pas une adaptation du débit en fonction des besoins de l'application puisque le nombre des opérateurs commun est figé et dépond de la taille N de la FFT et la longueur de la contrainte k du décodeur. En plus, le partage des opérateurs communs entre les deux algorithmes ne peut pas se faire en même temps comme nous l'avons décrit dans la figure 4.3. En effet, l'utilisation des FIFOs, qui génèrent des retards différents à chaque étage, rend le mapping des OCs (opérateurs communs) possible que dans un seul sens à partir du premier étage. Pour cela, nous présentons une deuxième architecture de gestion des opérateurs FFT/Viterbi dans le paragraphe suivant.

4.3.2 Architecture à colonne partielle

De nombreuses architectures de processeurs FFT ou de Viterbi peuvent être adaptées afin de concevoir une architecture flexible pour les deux al-

gorithmes. Toutefois, le besoin de flexibilité nous pousse à sélectionner les architectures de traitement à colonne partielle [96] au lieu des architectures en pipeline présentée dans le paragraphe précédent. En effet, l'architecture à colonne partielle permet aux papillons physiques d'être sélectionnés en fonction des exigences des normes à considérer. En d'autres termes, pour une FFT à N points ou d'un déncodeur de Viterbi de 2^{k-1} états, il y a $N/2$ (ou 2^{k-2}) papillon dans chaque colonne comme on le voit sur les figures 4.1 et 4.2. L'idée principale de cette architecture consiste à calculer Q opérations de papillons où $Q < N/2$. Ainsi, cette architecture est évolutive, puisque le nombre des papillons physiques (opérateurs communs) est indépendant de la taille de la FFT et de longueur de contrainte du décodeur Viterbi. Le nombre Q d'OCs instanciés est choisi selon les exigences de l'application donnée, car il définit le degré du parallélisme, et donc la vitesse de traitement. La figure 4.6 montre un exemple d'adaptation du parallélisme dans les architectures à colonne partielle pour une FFT 8 points où le parallélisme du traitement est adapté par changement du nombre des paillions physiques alloués.
L'architecture proposée (figure 4.7) est réalisée de telle sorte que les opérandes nécessaires pour le traitement d'un seul OC sont transférées simultanément à partir de la mémoire. Dans le chapitre précédent, les OCs en pipeline sont définis de telle sorte que deux opérandes sont transférés vers le papillon simultanément, donc chaque liaison de l'OC doit dédiée dans le bus mémoire. D'autre part, pour l'OC pipeliné proposé dans le chapitre précédent, un seul opérande à la fois est transféré à l'unité de papillon, et donc chaque papillon a un bus dédié vers et depuis la mémoire. Une telle disposition augmente légèrement le temps de calcul, mais cela peut être compensé en augmentant le nombre d'unités physiques de l'OC. Dans cette section, nous utilisons l'architecture en pipeline définie dans le chapitre précédent avec un système de mémoire parallèle pour augmenter le taux de transfert de données entre la mémoire et les unités de traitement [99]. En outre, la représentation en « géométrie variable » du treillis de Viterbi permet de minimiser le stockage en mémoire en tenant compte d'un schéma de calcul « in-place » [72] [100] pour la mise à jour des métriques de chemin et des opérandes FFT. Cette propriété, apporté par les caractéristiques du treillis FFT et appliquée au mode Viterbi, permet aux entrées et sorties de chaque étage de partager les mêmes emplacements dans la mémoire. Etant donné que dans le treillis FFT/Viterbi, les nouvelles valeurs écrasent les précédentes, seules deux adresses mémoire sont impliquées dans chaque opération papillon.
La figure 4.7 montre l'architecture proposée pour le processeur FFT/Viterbi, où les opérateurs utilisés sont définies dans le chapitre précédent dans la figure 3.13. Dans cette architecture, les unités OC sont connectées à des modules de mémoire via un réseau de commutation nommé « crossbar ».
L'approche pipeline des unités OC considérées permet la lecture et l'écriture de la mémoire d'un opérande à la fois (deux ports mémoires sont utilisées).



(a) Parallélisme total (b) Parallélisme partiel (c) Traitement série (un seul papillon physique)

FIGURE 4.6 – Exemple d'adaptation du parallélisme dans les architectures à colonne partielle pour une FFT 8 points

La fonction « OC Config. » alloue les unités de commande communes à la FFT ou les algorithme de Viterbi (schémas de configuration détaillées dans la figure 4.5.1). Les permutations FFT / Viterbi treillis sont contrôlés par deux fonctions. Le premier nommé « Switching Control. » connecte les unités d'opérateur commun pour les blocs de mémoire, conformément aux transitions de treillis. La seconde fonction appelée générateur d'adresses sélectionne la position de mémoire associée à l'unité de commande commune. L'adresse et les modules de commande de commutation, algorithmes radix-2 dans le contexte des systèmes de mémoire parallèles, ont déjà été souligné dans de nombreux articles [101] [102] [103]. Ces modules effectuent deux mapping, ils mappent N (ou 2^{k-1} pour Viterbi) adresses bit $a^m = (a^m_{k-2}, a^m_{k-3}, \ldots, a^m_0)$ sur une adresse de longeur $\log_2 Q$ des modules OCs et sur une adresse mémoire (sa longueur dépend de la taille de la mémoire) dans le module mémoire sélectionné [99].

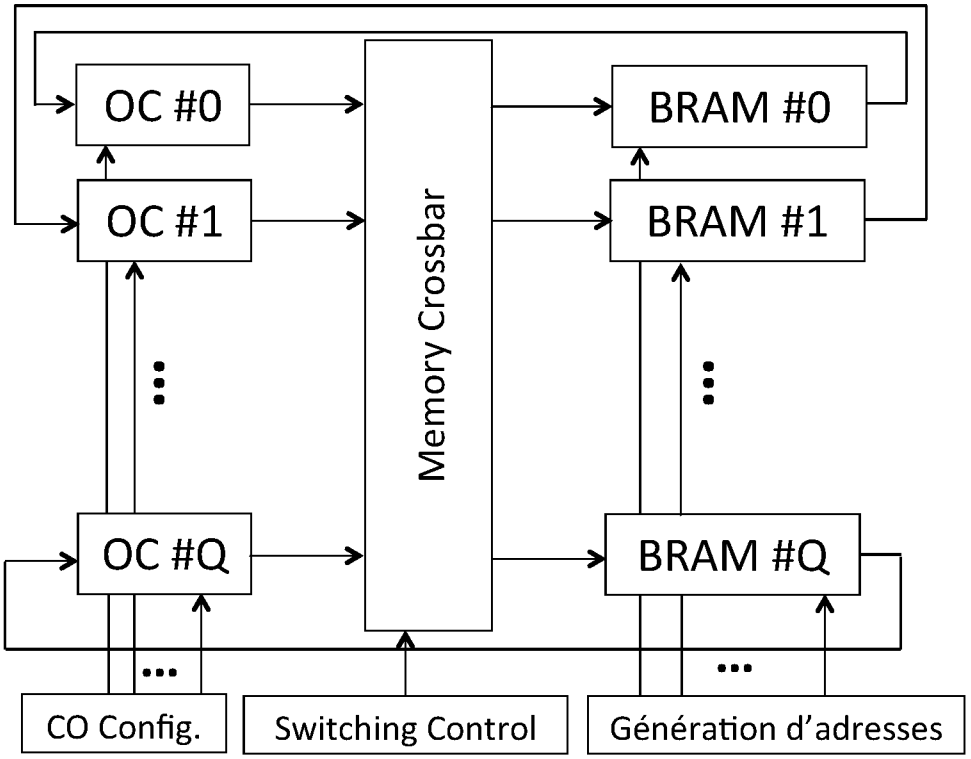


FIGURE 4.7 – Architecture à colonne partielle proposée pour le management de l'opérateur FFT/Viterbi

4.3.3 Architecture séléctionnée

Pour résumer, nous avons exploré dans cette section les architectures de gestion de l'opérateur FFT/Viterbi pour un partage efficace des ressources matérielles entre les deux algorithmes (Figure 4.8). Nous avons d'abord proposé une architecture en pipeline du processeur commun et nous avons montré que l'utilisation de l'architecture parallèle de l'opérateur convient mieux dans ce cas. Cette première solution n'était pas assez flexible par rapport aux objectifs fixés au début de ce chapitre. Nous avons donc proposé une deuxième architecture à colonne partielle plus flexible et nous avons montré que l'opérateur commun en pipeline avec un schéma d'accès « in-place » (treillis à géométrie variable) conviennent le mieux pour cette architecture.

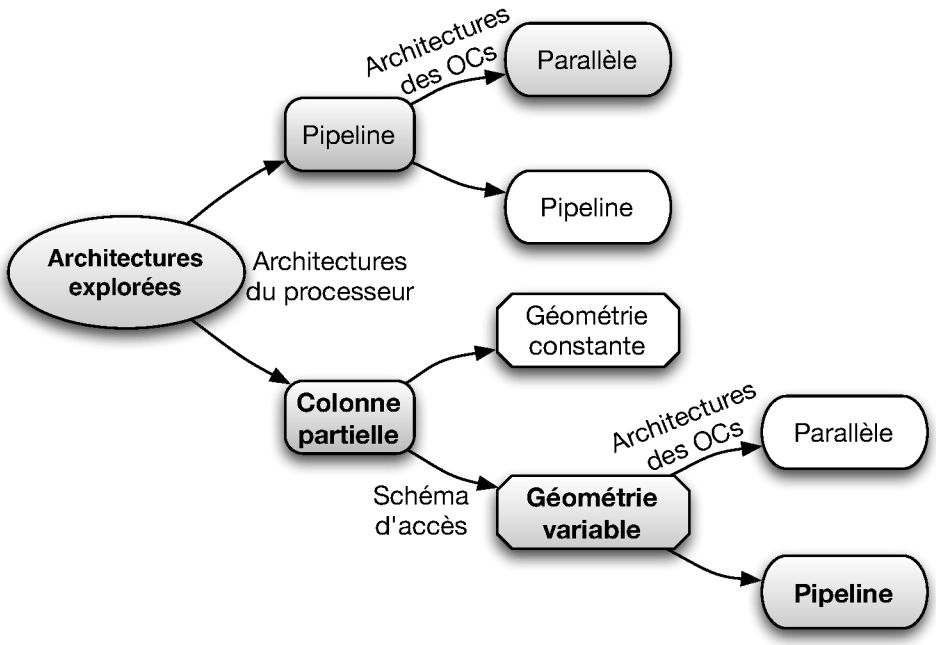


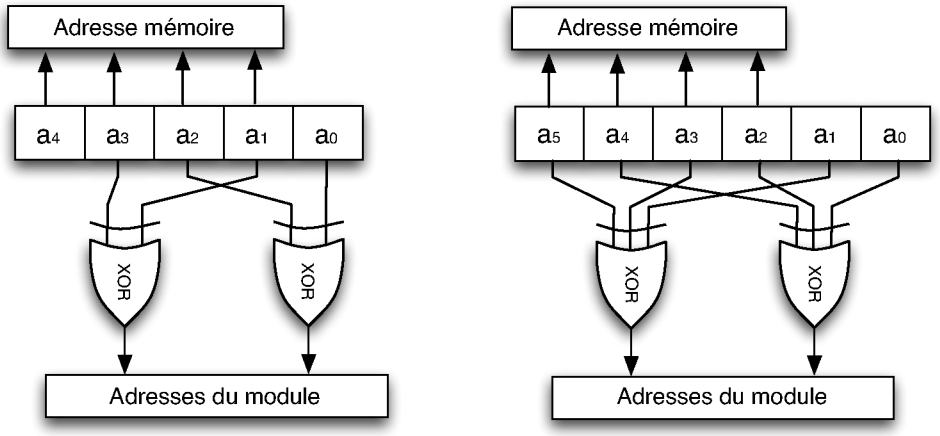
FIGURE 4.8 – Exploration des architectures de gestion de l'opórateur commun FFT/Viterbi

4.4 Management des interconnexions dans l'architecture à colonne partielle

Dans cette section nous présentons comment nous gérons les interconnexions et les largeurs des chemin de de donnés dans l'architecture à colonne partielle proposée dans la section précédente.

4.4.1 Génération des interconnexions

Le générateur d'adresses et les fonctions de commande des interconnexions ont déjà été étudiés pour la FFT [104] [105] et les décodeurs Viterbi [106] [107] [108] séparément dans de nombreux travaux. Toutefois, les similitudes structurelles entre les deux algorithmes montrés dans le chapitre 3 et la définition d'une structure de papillon commune (OC), nous permettent de considérer le même accès/permutations pour le processeur commun FFT/Viterbi. Nous considérons dans l'architecture proposée un schéma d'accès simple et très utilisé dans la littérature, appelé schéma XOR [109] [110], qui permet de gérer facilement la génération d'adresses et les fonctions de commande du réseau d'interconnexion. La figure 4.9 illustre un exemple de génération d'adresses des modules OCs et les adresses mémoires.



(a) $N = 2^{k-1} = 32$ sur 4 OCs physiques

(b) $N = 2^{k-1} = 64$ sur 4 OCs physiques

FIGURE 4.9 – Exemple de génération d'adresses mémoire

4.4.2 Management des largeurs de chemin de données

Nous avons précisé au début de ce chapitre que les algorithmes de FFT et Viterbi requièrent différentes largeurs des données. En effet, dans les standards de communications actuels, l'algorithme de Viterbi fonctionne avec des métriques codées sur 3 à 8 bits alors que la FFT utilise des échantillons codés sur 8 à 32 bits. Donc, il faut que l'architecture de traitement matériel reconfigurable pour les algorithmes FFT et Viterbi soit capable de prendre en compte cette différence et de s'adapter à l'utilisation qui en est faite. En effet, Biard a montré dans [76] que de gain de codage sature rapidement avec l'utilisation de quelques bits de quantification et que le codage des métriques

du décodeur Viterbi sur plus 8 bits n'a pas d'influence sur l'efficacité du décodeur. Donc l'utilisation des mêmes largeurs de chemin de données pour la FFT et le décodeur de Viterbi engendre nécessairement un gaspillage des ressources matérielles et de la consommation [111] [112].
Pour surmonter ce problème, nous considérons une technique de longueur de mot variable [113] [114], qui permet d'adapter la précision de calcul en fonction du mode sélectionné (FFT ou Viterbi) et bien sur les exigences du standard. L'idée consiste à masquer bits les moins significatifs lorsque le bas de la fourchette dynamique est nécessaire. Il a été prouvé que cette technique affecte positivement la consommation électrique (jusque'à 35% entre 2 et 8 bits de précision [114]) et les performances du système [113]. La figure 4.10 montre un exemple de configuration de longueur de mot pour une précision de 16 bits pour la FFT et 8-bits de précision pour Viterbi.

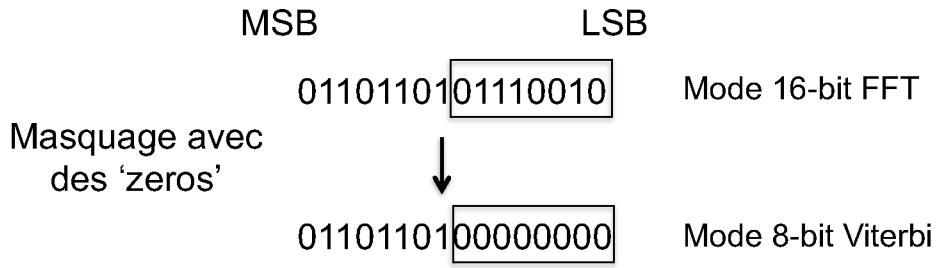


FIGURE 4.10 – Masquage des bits de poids faible

4.5 Implémentation du processeur commun FFT / Viterbi

Dans cette section, nous présentons une implémentation flexible sur FPGA de l'architecture à colonne partielle proposée dans les sections précédentes suivie d'une discussion et interprétation des résultats expérimentaux.

4.5.1 Description de l'architecture implémentée

Comme nous avons expliqué dans le paragraphe précédent, l'objectif principal de la mise en œuvre du processeur commun FFT/Viterbi est de fournir un partage dynamique des ressources entre les algorithmes. Pour atteindre cet objectif, nous proposons une mise en œuvre souple du générateur des adresses des modules et de la case mémoire. L'idée est de laisser la possibilité de varier dynamiquement la longueur de l'adresse du module et la case mémoire pour supporter l'affectation dynamique des ressources. Dans l'implémentation effectuée, les adresses mémoires et de module sont générées

par un logiciel dans chaque étage de treillis pour rendre possible le mapping souple des adresses a^m (définies dans le chapitre 3) dans les modules de permutation. Comme l'illustre la figure 4.11, le processeur FFT / Viterbi est implémenté conformément à la technique SIMD (Single Instruction Multiple Data) sur un FPGA Xilinx Virtex-4 FPGA. L'architecture du processeur est implémenté de façon à ce que le contrôle des permutations FFT/Viterbi en treillis soit commandé par un processeur MicroBlaze via des interfaces GPIO (représente le manager local de reconfiguration présenté dans la section 2.5). Sur la figure 4.11, le MicroBlaze assure le management des opérateurs communs en agissant sur trois registres : REG0 qui configure les OCs (mode FFT ou Viterbi comme illustré dans la figure), REG1 relie les OCs aux modules de mémoire (adresse du module) et REG2 choisit l'emplacement de stockage dans chaque module de mémoire (adresse de ligne). Ainsi, les adresses a^m, sont mappés par le MicroBlaze dans les registres REG1 et REG2 utilisant le schéma XOR pour les systèmes parallèles de mémoire [99]. Ainsi, la longueur de la FFT et la longueur de contrainte du décodeur de Viterbi sont limitées par la taille de ces deux registres.

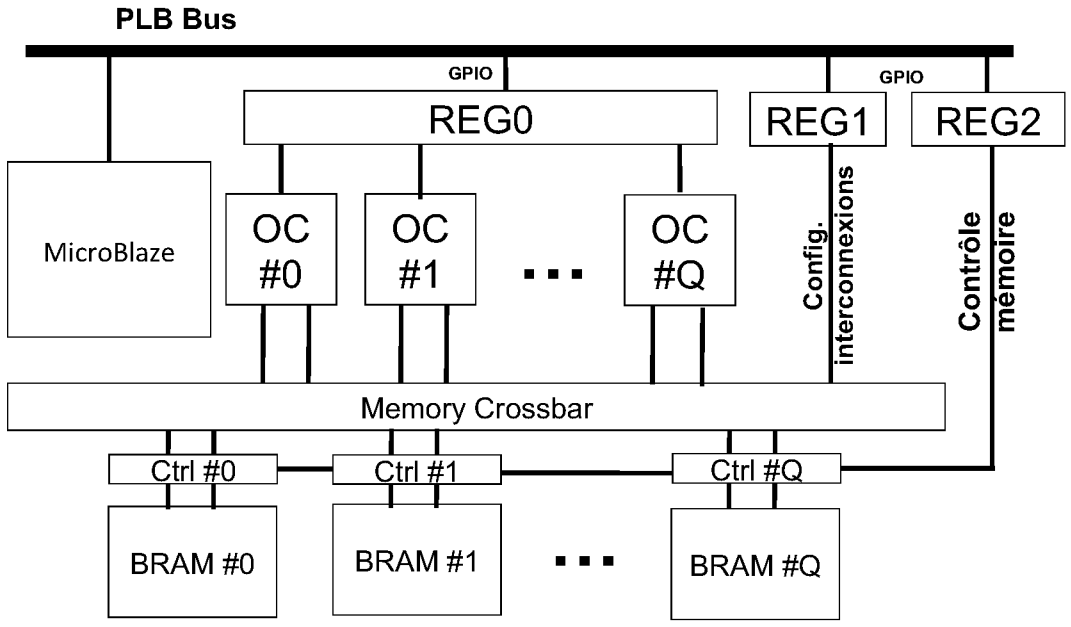


FIGURE 4.11 – Implémentation flexible du processeur FFT/Viterbi sur FPGA

4.5.2 Résultats de l'implémentation

La complexité du processeur commun est comparée à une implémentation classique où nous avons instancié parallélement un cœur Xilinx FFT et un décodeur de Viterbi (généré à partir des travaux [76]) sur la même plateforme FPGA Virtex 4. Comme le montre le tableau 4.1, ce gain en flexibilité offert par le processeur commun est réalisé sans surcoût de complexité car celle-ci a diminué -30% dans les configurations testées grâce au partage des ressources de calcul. Les résultats montrent également une réduction importante de l'occupation mémoire due au partage des BRAMs entre les algorithmes. Cependant, la mise l'architecture proposée est plus lente qu'un matériel dédié. En effet, comme le montre le tableau 4.2, l'opérateur commun considéré est environ 40% moins rapide qu'un papillon FFT Radix-2 classique. Par contre, la vitesse de calcul peut être améliorée en échange d'une augmentation des ressources allouées aux algorithmes (augmenter le parallélisme du calcul avec l'allocation de plus de ressources en termes d'OCs).
En effet, Le plus grand avantage offert par l'architecture implémentée est la flexibilité qui permet d'augmenter le parallélisme du traitement en partageant des ressources communes entre les algorithmes. L'architecture à colonne partielle proposée permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permettre l'accélération des traitements. Afin quantifier cet avantage, nous avons évalué cette accélération des traitements en comparant (pour les deux algorithmes) la latence en nombre de cycles obtenue avec un seul opérateur commun à la latence en cycles obtenue avec 2, 4, 8 et 16 opérateurs. La figure 4.12 illustre l'accélération des traitements en fonction du nombre des opérateurs communs alloué pour chaque algorithme.
Ces résultats montrent ainsi que dans des implémentations multistandard, il est possible de profiter du partage des ressources et le parallélisme des traitements pour diminuer la complexité des designs (comme illustré dans la figure 4.13).
En outre, les opérateurs communs considérés sont basés sur une architecture de efficace en consommation et ont été construits de maniére à réduire la surface des ressources de reconfiguration. Donc a partir de fait que les opérations papillon dominent la consommation d'énergie dans les processeurs Viterbi et FFT [81], une réduction de la consommation peut être obtenue au niveau du processeur surtout avec l'utilisation de la technique de masquage des bits de poids faible présentée précédemment.

4.6 Conclusion

Dans ce chapitre nous avons étudié d'une manière approfondie du management des opérateurs communs FFT/Viterbi dans un contexte multistan-

TABLE 4.1 – Comparaison des ressources utilisées entre une implémentation classique et architecture proposée

	Slices / BRAM	
Nombre de papillons	4	8
Hw FFT + Viterbi	3817 / 9	7854 / 20
Processeur proposé	3251 / 4	5298 / 8

TABLE 4.2 – Comparaison des performances de l'opérateur FFT/Viterbi par rapport à une cellule classique FFT-Radix 2

	Cellule FFT Radix-2	Cellule OC FFT/Viterbi piepeline
Fréquence Max.	117.592 MHz	71.798 MHz
Nombre de slices	100	141

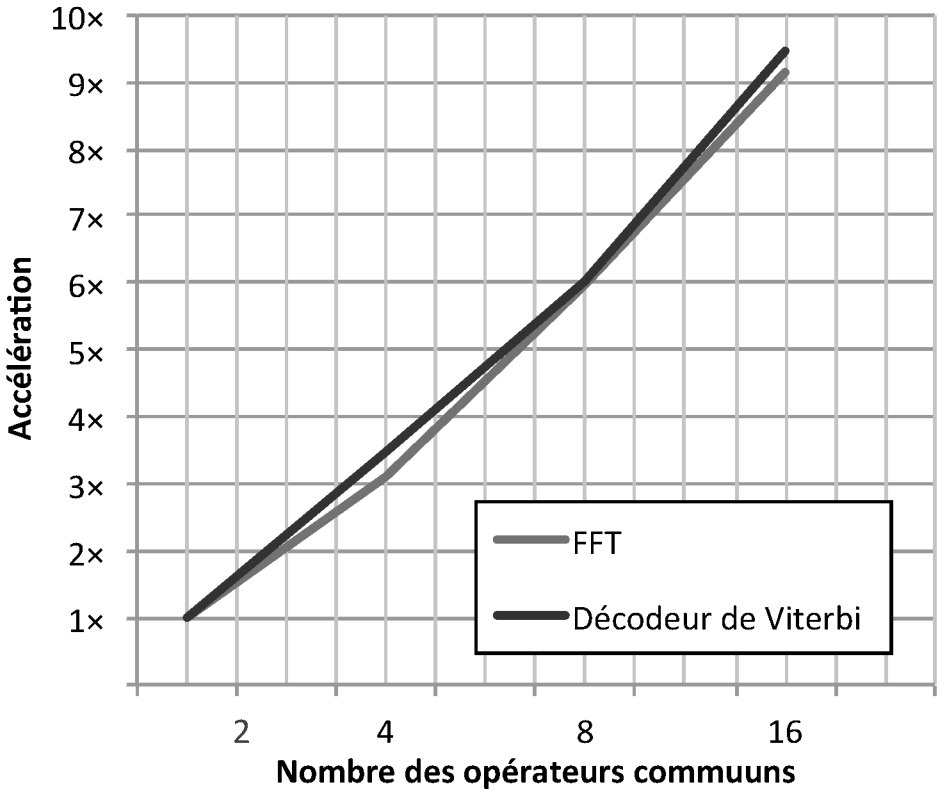


FIGURE 4.12 – Accélération des traitements en fonction des opérateurs communs affectés aux algorithmes FFT et Viterbi

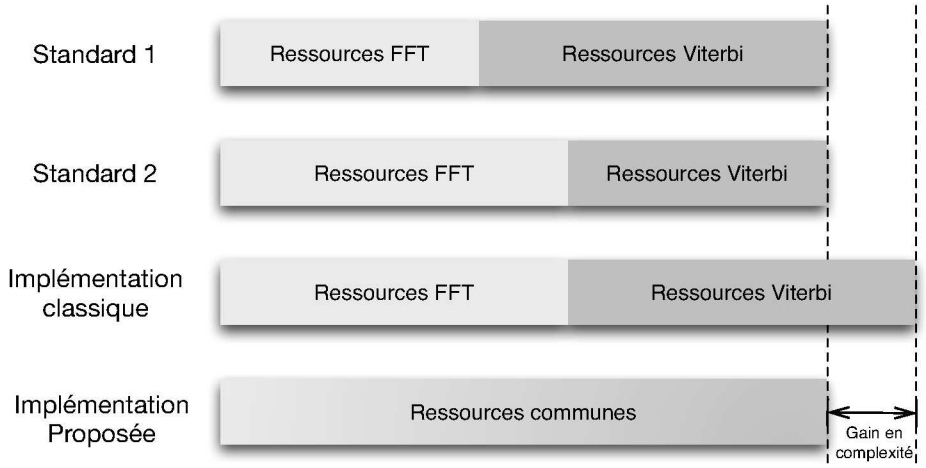


FIGURE 4.13 – Partage des ressources entre les algorithmes FFT et Viterbi dans un exemple de design bi-standard

dard et nous avons proposé plusieurs mécanismes de gestions à plusieurs niveaux. Nous avons également proposé et implémenté une architecture de processeur permettant la gestion efficace des ressources matérielles entre les algorithmes FFT et Viterbi. L'architecture proposée permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permettre l'accélération des traitements. Les résultats des implémentations montrent que l'utilisation de l'opérateur FFT/Viterbi offre des gains en complexité pouvant atteindre 30% par rapport à une implémentation classique. Ainsi que dans des implémentations à base de ces opérateurs communs FFT/Viterbi, il est possible de profiter du partage des ressources et de parallélisme des traitements, non seulement pour ajouter de la flexibilité au design, mais aussi pour diminuer la complexité et la surface des circuits.

Conclusion

Dans un contexte applicatif de plus en plus multistandards, la reconfigurabilité est, à tous les niveaux, un axe central des recherches dans le domaine des radiocommunications. En effet, les techniques et les architectures reconfigurables deviennent de plus en plus indispensables aux différents acteurs du secteur (équipementiers, fournisseurs de services...) pour proposer à l'avenir des systèmes de communications génériques. Le travail présenté dans cette thèse se situe dans ce contexte de la radio reconfigurable et plus précisément dans la définition de structures de traitement communes permettant la réalisation d'équipements multistandards. L'approche suivie pour atteindre cet objectif, appelée paramétrisation, consiste à identifier des opérateurs capables de réaliser plusieurs fonctionnalités offrant ainsi une conception à la fois flexible, évolutive et efficace. En particulier, nous avons exploré dans ce manuscrit la technique des opérateurs communs. Cette technique consiste à identifier des entités, appelées opérateurs communs, utilisées massivement par les différents modules d'un équipement multistandard. Le niveau de granularité de ces opérateurs communs se situe à un niveau intermédiaire entre les fonctions de traitement de signal et les opérateurs arithmétiques de bas niveau.
La diversité des traitements en bande de base nécessite l'introduction d'une technique de management des ressources matérielles utilisant des éléments de traitements génériques comme les opérateurs communs. Nous avons présenté dans ce manuscrit des modèles fonctionnels abstraits qui ont pour objectif de permettre le déploiement des applications reconfigurables sur une plateforme matérielle hétérogène basée sur les opérateurs communs. Nous avons défini également le management des opérateurs communs et nous avons étudié l'implémentation de ces opérateurs en se basant essentiellement sur des évaluations de complexité pour quelques standards utilisant la modulation OFDM. Nous avons montré aussi que les algorithmes de FFT et décodage de canal sont des traitements lourds en terme de calcul et peuvent représenter plus de 80% des charges de traitements d'une radio utilisant la modulation OFDM. Donc, la définition d'une technique de gestion efficace des ressources entre ces familles d'algorithmes constitue un pas significatif en avant. Pour cela, nous avons présenté de nouvelles structures reconfigurables de l'opérateur commun FFT/Viterbi après avoir

étudié les ressemblances structurelles entre les deux algorithmes. Nous avons comparé les stuctures de l'opérateur FFT/Viterbi avec des structures similaires de la littérature. Comme nous l'avons montré dans ce manuscrit, la structure sélectionnée fournit un gain important de point de vue de la complexité avec une bonne performance en termes de nombre d'opérations par cycle. Ensuite, nous avons étudié d'une manière approfondie le management des opérateurs communs FFT/Viterbi dans un contexte multistandard et nous avons proposé plusieurs mécanismes de gestions à différents niveaux. Nous avons également proposé et implémenté une architecture de processeur permettant la gestion efficace des ressources matérielles entre les algorithmes FFT et Viterbi. L'architecture proposée permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permet l'accélération des traitements. Les résultats des implémentations montrent que l'utilisation de l'opérateur FFT/Viterbi offre des gains en complexité pouvant atteindre 30% dans les configurations testées par rapport à une implémentation classique. Les résultats montrent également une réduction importante de l'occupation mémoire due au partage de la mémoire entre les algorithmes. Cependant, la mise l'architecture proposée est plus lente qu'un matériel dédié. En effet, il a été montré que l'opérateur commun considéré est environ 40% moins rapide qu'un papillon FFT Radix-2 classique. Mais nous avons pu montrer que la vitesse de calcul peut être améliorée en échange d'une légère augmentation des ressources alloués en termes d'opérateurs communs. Un autre grand avantage offert par l'architecture implémentée est la flexibilité qui permet d'augmenter le parallélisme du traitement en partageant des ressources communes entre les algorithmes. En effet, l'architecture à colonnes partielles proposée permet d'adapter le nombre d'opérateurs communs alloués à chaque algorithme et donc permettre l'accélération des traitements.
Ces résultats montrent que dans des implémentations à base de ces opérateurs de forte granularité comme l'opérateur commun FFT/Viterbi, il est possible de profiter du partage des ressources et de parallélisme des traitements, non seulement pour ajouter de la flexibilité au design, mais aussi pour diminuer la complexité et la surface des circuits. Ce gain en complexité et en flexibilité a maintenant plus d'impact au niveau système par rapport aux opérateurs de faible granularité (comme le LFSR [10] et le CORDIC [9]).

Perspectives

Les travaux menés dans le cadre de cette thèse ouvrent de nombreuses perspectives. Du point de vue des réalisations, les perspectives à court terme sont dans un premier temps de proposer des cellules d'opérateurs communs supportant différents radix pour la FFT et des rendements de type n_0/r (où $n_0 > 1$) pour le décodage de Viterbi. En effet, les similarités identifiées

entre les deux algorithmes peuvent être généralisées pour supporter des algorithmes plus complexes (radix mixtes par exemple).
A long terme, il est envisageable de définir de nouveaux opérateurs communs pour les fonctions de filtrage au niveau du front-end numérique. En effet, nous avons montré dans le deuxième chapitre que dans les radios utilisant la modulation multiporteuse les fonctions de filtrage, décodage canal et transformés de Fourier représentent plus de 90% des charges de calcul en bande de base. Donc définir un banc d'opérateurs communs pour ces trois familles de traitement de signal constituera un pas significatif vers l'optimisation des traitements numériques.
Un autre prolongement de ce travail serait de poursuivre les travaux de l'équipe SCEE sur l'optimisation des graphes d'opérateurs en introduisant les considérations architecturales identifiés dans cette thèse traduites en termes de fonctions de coûts.

Publications de l'auteur

Revues

[**R1**] M. Naoues, D. Noguet, L. Alaus, Y. Louët, "A common operator for FFT and FEC decoding," Microprocessors and Microsystems, Volume 35, Issue 8, November 2011, Pages 708-715, ISSN 0141-9331, 10.1016/j.micpro.2011.08.007.

[**R2**] M. Naoues, et. al "Common Operator architectures for FFT and Viterbi Decoding," The Journal of Signal Processing Systems, 2013 [*Submitted*].

Brevets

[**B1**] M. Naoues, D. Noguet, "Processor for processing digital data with pipelined butterfly operator for the execution of an FFT/IFFTt and telecommunication device", France and US Patent Application FR2960990, US20130077663A1.

[**B2**] D. Noguet, M. Naoues, R. Michard, "Cellule optimisée de l'opérateur FFT/Viterbi", France Patent Application. [*Pending*]

Conférences à comités de lecture

[**CI1**] M. Naoues, D. Noguet, Y. Louët, K. Grati, A. Ghazel, "A Flexible Processor for FFT and Viterbi Algorithms," IEEE International Symposium on Communications and Information Technology, ISCIT 2012, Gold Coast, Australia.

[**CI2**] M. Naoues, D. Noguet, Y. Louët, K. Grati, A. Ghazel, "An Efficient Flexible Common Operator for FFT and Viterbi algorithms," IEEE 73rd Vehicular Technology Conference, VTC Spring 2011, Budapest, Hungary.

[**CI3**] M. Naoues, L. Alaus, D. Noguet, "A Common Operator for FFT and Viterbi algorithms," The 13th Euromicro Conference on Digital System Design, DSD 2010, Lille, France.

Conférences sans comités de lecture

[**CO1**] M. Naoues, D. Noguet, Y. Louët, K. Grati, A. Ghazel, "Common operator approach for flexible radio design", Joint Workshop on Wireless Communications, NEWCOM++ / COST 2100, JNCW 2011, March 2011, Paris, France.

[**CO2**] M. Naoues, D. Noguet, Y. Louët, C. Moy, J. Palicot, "Common operator approach for flexible radio design", NEWCOM++ Dissemination Day, June 2010, Florence Italy.

[**CO3**] M. Naoues, "Techniques de gestion des Opérateurs Communs pour les algorithmes de FFT et de Viterbi", séminaire SCEE, Mars 2012, Supelec Rennes.

Projets européens

[**PE1**] Projet C2POWER :

- *Deliverable D4.1 intermediate version* : Flexible low power digital techniques.
- *Deliverable D4.4 final version* : Flexible Low-Power Digital Techniques including Hardware Module.

[**PE2**] Projet NEWCOM ++ :

- *Deliverable DRC.3, WPRC* : Performance evaluation and guidelines for future flexible radio architectures.

Annexe A

Terminologie et concepts des systèmes radio avancés

Les systèmes radios avancés utilisent un ensemble de techniques visant à répondre aux besoins de flexibilité et aux évolutions des standards de radiocommunications. Nous donnons dans cette annexe quelques définitions des termes utilisés dans ce manuscrit qui sont liés aux concepts des systèmes radio avancés. Ces définitions ont été adaptées du standard IEEE 1900.1-2008 [115] et du forum d'innovation sans-fil (Wireless innovation forum anciennement SDR forum) [116]. Les termes utilisées dans les systèmes radio avancés se déclinent en plusieurs expressions suivant l'architecture de l'équipement de radiocommunication.

A.1 La radio non-reconfigurable

Ce type de radio est basé uniquement sur des composants matériels ne pouvant pas être modifiés autrement que par une intervention physique (en contraste avec la radio logicielle).
Il faut noter que le remplacement d'un composant matériel avec un composant identique qui contient des données ou des instructions différentes est considéré comme une intervention physique sur l'équipement. Par ailleurs, une radio ayant des paramètres pouvant être changés sans une intervention physique n'est pas considéré comme radio non-reconfigurable [115].
Ce terme représente une abstraction idéalisée utilisée dans la classification des équipements radio (radio matérielle, radio logicielle, radio intelligente...). Le terme est aussi utilisé pour décrire l'évolution générale de la reconfigurabilité des équipements radio.

A.2 La radio reconfigurable

Un type de radio dont la fonctionnalité peut être changée par une reconfiguration manuelle (par intervention physique ou changement des composants matériels) ou via des modules logiciels de reconfiguration qui interviennent sur les paramètres des modules matériels reconfigurables (qui peuvent être téléchargés via le réseau de communication ou générés localement) [115].

A.3 La radio contrôlée par logiciel

Il s'agit d'une radio où seules les fonctions de contrôles sont implantées en logiciel. Un nombre limité de fonctions peut changer, comme les interconnections, les niveaux de puissance, mais pas les bandes de fréquences ou les types de modulations utilisées [115].

A.4 La radio logicielle

Par contraste avec la radio non-reconfigurable (ou matérielle), la radio logicielle est définie comme un type de radio où les fonctions de la couche physique sont implémentées (totalement ou partiellement) en logiciel. Il existe trois niveaux de radio logicielle [116].

La radio logicielle restreinte (Software Defined Radio, SDR) Les systèmes à base de SDR fournissent un contrôle ainsi qu'une partie des fonctions de la couche physique sous forme logicielle. La SDR correspond aux caractéristiques d'une numérisation par bande radio restreinte réalisée en fréquence intermédiaire (IF) ou en bande de base. Les contraintes des systèmes SDR se concentrent sur la couverture des bandes de fréquences car au niveau du « front-end » un multiplexage entre plusieurs chaines RF est nécessaire.

La radio logicielle idéale (Ideal Software Radio, ISR) représente une radio où toutes les fonctions de traitement de signal sont réalisées en logiciel. Donc la différence par rapport aux systèmes SDR est que les étages analogiques de la radio sont supprimés (sans le passage par une fréquence intermédiaire). La programmabilité du système est étendue à l'ensemble du système radio avec une conversion analogique-numérique au niveau de l'antenne.

La radio logicielle « ultime » (Ultimate Software Radio, USR) Cette définition est donnée dans le seul but de pouvoir comparer les systèmes

réalisables à une référence ultime. Il s'agit d'une radio permettant de se reprogrammer pour traiter tout type de traffic et supportant une grande plage de fréquences, d'interfaces air et d'applications.

A.5 La radio intelligente

La radio intelligente est un type de radio où les systèmes de communications sont « conscients » de leur environnement et peuvent faire des décisions concernant leurs comportements radio en se basant sur ces informations (qui peuvent inclure la position géographique ou non) et sur des objectifs prédéfinis. La radio intelligente utilise les techniques de la radio logicielle pour ajuster automatiquement son comportement ou ses opérations pour atteindre les objectifs désirés [115].
Ainsi, un équipement intelligent est défini comme capable non seulement de déterminer le contexte dans lequel il opère mais également à même de modifier son comportement pour s'adapter à ce contexte, la méthode d'émission / réception et par conséquent l'architecture nécessaire à celle-ci. Le changement opéré par le terminal voire le va-et-vient entre différentes configurations devra répondre à des contraintes dynamiques et apparaitre transparent pour l'utilisateur.
Ce type de radio est capable d'apprentissage automatique (Machine learning) et est parfois considérée dépendante de la radio contrôlée par logiciel. Cette dépendance vient de du fait que l'apprentissage automatique nécessite une adaptation des procédures de prise de décision ce qui nécessite une reprogrammation dynamique des fonctions de contrôle de la radio. Par contre, cette adaptation dynamique des procédures de prise de décision ne nécessite pas nécessairement une implémentation purement logicielle (de type radio logicielle par exemple) puisque seules les fonctions de contrôles nécessitent un niveau de flexibilité élevé pour actualiser les paramètres de la couche physique (implémentée indifféremment en logiciel ou en matériel).
Mitola définit le cycle cognitif en six étapes (figure A.1) [15] :

Observer : Prendre conscience de l'environnement.

Orienter : Orienter le traitement selon divers niveaux de priorité (normal, urgent, immédiat).

Planifier : Planifier les meilleures configurations possibles suivant les priorités précédentes.

Décider : Allouer les ressources.

Agir : Effectuer la reconfiguration de l'équipement.

Apprendre : Apprendre des échecs ou des réussites des précédentes reconfigurations.

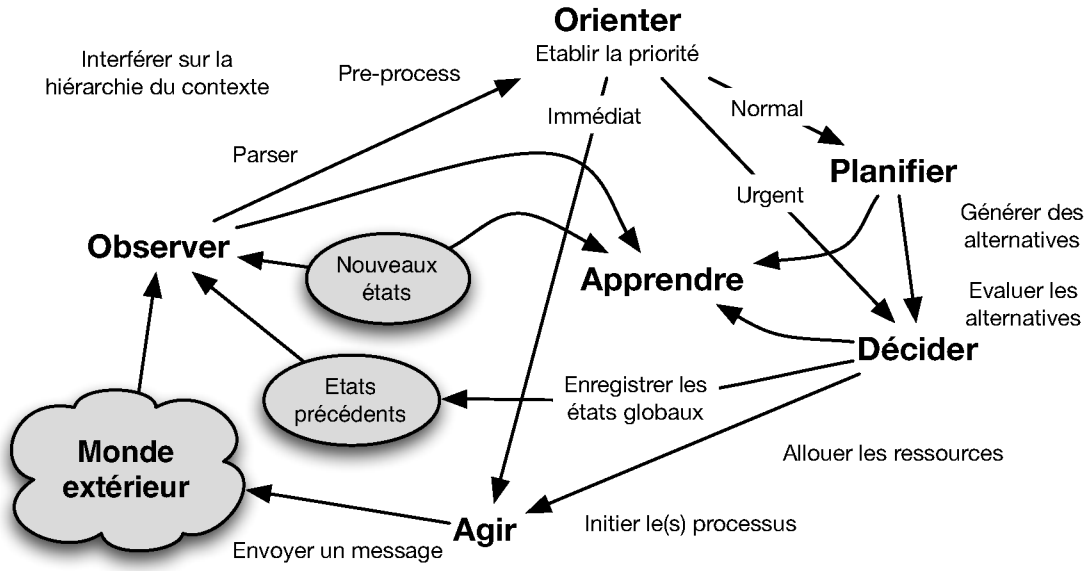


FIGURE A.1 – Cycle de cognition de Mitola (adapté de [15])

Annexe B

L'opérateur commun transformée de Fourier rapide

Dans cette annexe nous rappelons quelques possibilités d'utilisation de l'opérateur Transformée de Fourier Rapide (FFT) à travers les étapes d'une chaine de traitement de signal. La FFT est présentée ici comme un opérateur mathématique, qui est utilisé, seul ou en complément d'autres opérations mathématiques pour effectuer différentes fonctions de traitement de signal. Nous commençons par rappeler brièvement les fondamentaux relatifs à la FFT afin de comprendre les différentes utilisations qui peuvent en être faites. La FFT, est un algorithme de calcul de la transformée de Fourier Discrète [77] qui a l'avantage de réduire le nombre d'itérations de $2N^2$ à $2N\log(N)$ pour une longueur de FFT de taille N. Les algorithmes de FFT se divisent en deux grandes classes : « décimation en temps » et « décimation en fréquence ». L'algorithme de Cooley-Tukey le plus utilisé pour le traitement de la FFT se base sur une technique de calcul par itération successive [77]. L'idée est de diviser successivement par deux la taille des données sur lesquelles les calculs sont effectués. Dans $\mathbb{C}$, la transformée de Fourier discrète d'un vecteur $v = (v_0, v_1, \ldots, v_{N-1})$ de nombres réels ou complexes est un vecteur $V = (V_0, V_1, \ldots, V_{N-1})$ défini par :

$$V_k = \sum_{i=0}^{N-1} e^{-j\frac{2\pi ik}{N}} v_i \qquad k = 0, \ldots, N-1 \tag{B.1}$$

N étant un entier représentant la longueur de la transformée et $j = \sqrt{-1}$. La base de Fourier $exp(\frac{-j2\pi}{N})$ est la N^{eme} racine de l'unité dans $\mathbb{C}$.

B.1 Utilisation de l'opérateur FFT dans les modulations multi-porteuses

Une grande partie des standards actuels (dont ceux considérés dans ce manuscrit) utilisent les techniques de modulation multi-porteuses [117]. Les modulations multi-porteuses comme l'OFDM consistent à repartir les symboles sur plusieurs porteuses à l'opposé des systèmes classiques qui transmettent les symboles en série, chaque symbole occupant alors toute la bande passante disponible. Ainsi dans le cas de l'OFDM, pour un train de symboles de période T_{Si}, les symboles seront répartis en N trains plus lents et auront alors une durée $T_S = NT_{Si}$. Pour répartir les données à transmettre sur les N porteuses, on groupe les symboles c_k par paquets de N. Les c_k sont des nombres complexes définis à partir des éléments binaires par une constellation. La séquence de N symboles $(c_0, c_1, \ldots, c_{N-1})$ constitue un symbole OFDM. Le k^{eme} train de symboles parmi les N trains module un signal de fréquence f_k. Le signal modulé du train k s'écrit sous forme complexe : $c_k e^{2j\pi f_k t}$. Le signal total $s(t)$ correspondant à l'ensemble des N symboles réassemblés en un symbole OFDM :

$$s(t) = \sum_{i=0}^{N-1} c_k e^{2j\pi f_k t} \tag{B.2}$$

Les fréquences f_k sont orthogonales si $f_k = f_0 + \frac{k}{T_s}$. En effet chaque porteuse modulant un symbole pendant une fenêtre temporelle rectangulaire de durée T_s, son spectre en fréquence est un sinus cardinal qui est une fonction qui s'annule tous les multiples de $1/T_s$ Ainsi, il n'y a aucune interférence entre les sous-porteuses lorsque l'échantillonnage est effectué à une fréquence f_k d'une sous-porteuse. C'est ce qui permet de recouvrir les spectres des différentes porteuses et d'obtenir ainsi une occupation optimisée du spectre. Donc, le signal $s(t)$ s'écrit sous la forme :

$$s(t) = e^{2j\pi f_0 t} \sum_{i=0}^{N-1} c_k e^{2j\pi \frac{kt}{T_s}} \tag{B.3}$$

En discrétisant ce signal et en le ramenant en bande de base on obtient une sortie $s(n)$ sous la forme :

$$s(n) = \sum_{i=0}^{N-1} c_k e^{2j\pi \frac{kn}{N}} \tag{B.4}$$

Les $s(n)$ sont donc obtenus par une transformée de Fourier inverse discrète des c_k. En choisissant le nombre de porteuses N tel que $N = 2n$, le calcul de la transformée de Fourier inverse se simplifie et peut se calculer par une simple IFFT. Ainsi la modulation et la démodulation OFDM correspondent respectivement à une IFFT et une FFT. Ainsi, nous avons montré comment la modulation OFDM est réalisée par l'opérateur mathématique FFT.

B.2 Utilisation de l'opérateur FFT dans le codage canal

De la même façon qu'il existe des transformées de Fourier définies sur des ensembles infinis (tel que le domaine des nombres complexes $\mathbb{C}$) utilisées classiquement en traitement du signal, il existe aussi des transformées de Fourier sur des ensembles finis (tels que les corps de Galois $GF(F_t)$ utilisés en particulier dans le traitement des codes cycliques. Dans un corps de Galois $GF(F_t)$, l'élément primitif α d'ordre N est la N^{eme} racine de l'unité. Par analogie entre $exp(\frac{-j2\pi}{N})$ dans $\mathbb{C}$ et α dans $GF(F_t)$, considérons un vecteur $v = (v_0, v_1, \ldots, v_{N-1})$ dans $GF(F_t)$ et α un élément d'ordre N de ce corps. Le vecteur v et sa transformée de Fourier sont reliés par l'équation suivante [118] :

$$V_k = \sum_{i=0}^{N-1} \alpha^{ij} v_i \qquad k = 0, \ldots, N-1 \tag{B.5}$$

Les transformées de Fourier définies dans $GF(F_t)$ ont été introduites dans l'étude des codes cycliques dans un souci de réduction de complexité des décodeurs par Gore [119] et puis par Michelson [45], Chien [120] et Lempel [121]. Plus tard, Blahut [122], afin d'optimiser l'utilisation des transformées de Fourier a traduit le processus de codage (classiquement effectué en temporel) dans le domaine fréquentiel. Il a aussi adapté les différents algorithmes de décodage de façon à être réalisés dans le domaine fréquentiel.
Le principe du codage d'un code $C(n, k)$ proposé par Blahut [122] consiste à former un mot d'information de longueur k dans le domaine fréquentiel dans lequel $2t$ composantes prédéterminées sont fixées à 0 (t : pouvoir de correction du code). Ensuite, le mot de code temporel de longueur n est obtenu à l'aide d'une transformée inverse de Fourier. Pour le décodage, la transformée de Fourier peut être utilisée pour le calcul des deux étapes les plus longues, à savoir le calcul des syndromes et l'algorithme de Chien. Al-Ghouwayel a proposé dans sa thèse un opérateur FFT (figure B.1) basé sur un papillon reconfigurable pouvant ráliser à la fois des opérations dans $\mathbb{C}$ et dans $GF(F_t)$ pour le décodage Reed-Solomon (RS) [8].
Aussi, Tomlinson a proposé dans [123] un décodeur convolutionnel de type Viterbi qui se base l'algorithme FFT et qui peut faire face à un décalage de fréquence, rotation de phase du signal reçu. L'avantage de ce décodeur [123] est qu'il n'a pas besoin d'être précédé par un modem et n'est pas sensible aux problèmes de sauts de cycles. Il peut aussi fonctionner avec un rapport signal-à-bruit (SNR) négatif et il est bien adapté au décodage des codes convolutifs à faible taux.
Ainsi, nous avons montré dans cette section comment l'opérateur FFT est utilisé dans les décodeurs Viterbi et RS.

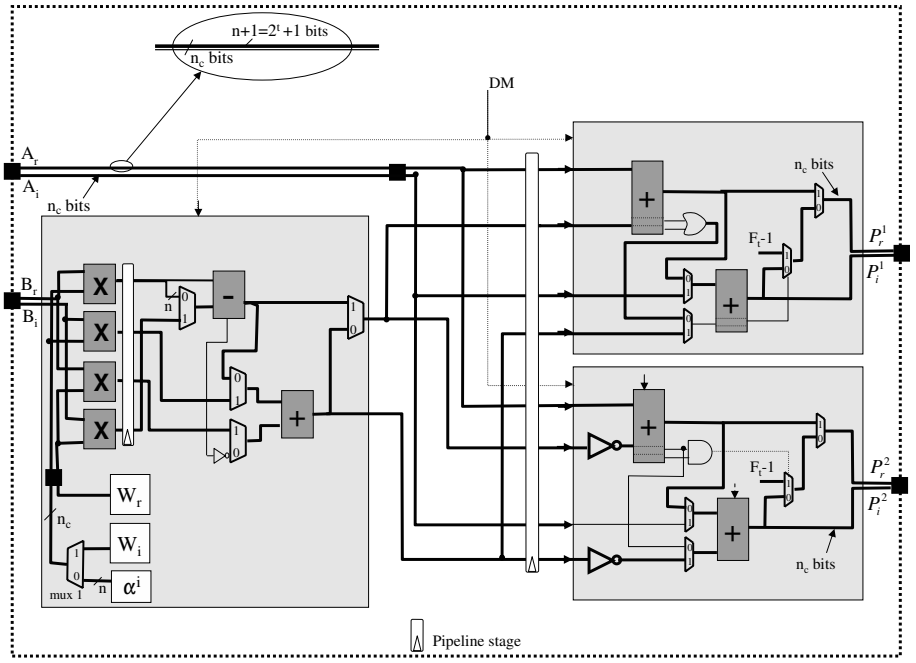


FIGURE B.1 – Opérateur commun FFT ($\mathbb{C}$ et $GF(F_t)$) proposé par Al-Ghouwayel [8]

B.3 Utilisation de la FFT dans les opérations de convolution et de corrélation

Une autre utilisation « indirecte » de la FFT consiste principalement à transposer les signaux dans le domaine fréquentiel afin de simplifier les traitements. Ainsi, contrairement au cas de la première section, la FFT ne réalise pas directement les opérations concernées mais effectue une transposition dans le plan fréquentiel afin de permettre une approche de réalisation différente. Les standards que nous avons considérés dans ce manuscrit permettent la détection du signal par corrélation avec une séquence d'apprentissage connue. Les formules de la corrélation et de la Transformée de Fourier sont très proches et il est possible d'obtenir une corrélation (cross-correlation, plus exactement) à partir de la FFT. Ainsi, la corrélation de deux signaux peut se faire au moyen de la FFT et de son inverse la IFFT. De même, un filtrage numérique suit la même utilisation de la FFT que la corrélation. Un filtrage est obtenu par convolution du signal d'entrée avec la réponse impulsionnelle du filtre dans le domaine temporel.
En suivant le même raisonnement que pour la corrélation, la convolution de deux signaux transposés dans le domaine fréquentiel se ramène à la multiplication de ces signaux. De la même manière, les fonctions de décorrélation [124] nécessaires dans la chaine de réception et qui s'assimilent à des fonctions

de convolutions peuvent être rálisés par une transposition dans le domaine fréquentiel grâce à la FFT.

B.4 Utilisation de l'opérateur FFT dans les fonctions d'égalisation, filtrage et estimation de canal

Outre la convolution et la corrélation qui peuvent être réalisés dans le domaine fréquentiel, une abondante littérature décrit plusieurs possibilités d'implémenter des égaliseurs et des estimateurs de canal en domaine fréquentiel. Ces différentes implémentations sont largement décrites en [125] pour le FLMS (Frequency Domain Least mean square), en [126] pour le FLMS sans contrainte (UFLMS), en [127] pour l'algorithme de Quasi-Newton et pour les égaliseurs adaptatifs à retour décision. Ainsi, plusieurs types d'égaliseurs peuvent être implémentés via l'opérateur FFT par transposition dans le domaine fréquentiel.
En suivant le même procédé de passage en fréquence, en [128] et en [129], il a été démontré que respectivement les calculs d'un RAKE et d'une mise en canaux pouvait se faire à l'aide d'une FFT. Ainsi, celle-ci peut être opérationnelle pour la corrélation, la convolution, le filtrage, l'estimation de canal, l'égalisation, le désétalement de spectre.

La conclusion à retenir de cette annexe est que l'opérateur FFT peut être utilisé pour réaliser en totalité ou en partie différentes fonctions de traitement de signal de nature distincte. Donc, factoriser les traitements en se basant cet opérateur peut réduire la complexité de l'implémentation d'un équipement radio [24].

Bibliographie

[1] Wireless world research forum (wwrf) web site. http ://www.wireless-world-research.org/.

[2] M. Mueck et al. Etsi reconfigurable radio systems : status and future directions on software defined radio and cognitive radio standards. *Communications Magazine, IEEE*, 48(9) :78–86, 2010.

[3] J. Mitola. The software radio architecture. *Communications Magazine, IEEE*, 33(5) :26–38, 1995.

[4] S. Srikanteswara, J.H. Reed, P. Athanas, and R. Boyle. A soft radio architecture for reconfigurable platforms. *Communications Magazine, IEEE*, 38(2) :140–147, 2000.

[5] H.W. Tuttlebee. *Software Defined Radio : Enabling Technologies*. Wiley, 2002.

[6] A. Wiesler and F.K. Jondral. A software radio for second- and third-generation mobile systems. *Vehicular Technology, IEEE Transactions on*, 51(4) :738–748, 2002.

[7] L. Alaus, J. Palicot, C. Roland, Y. Louet, and D. Noguet. Promising technique of parameterization for reconfigurable radio, the common operators technique : Fundamentals and examples. *Journal of Signal Processing Systems*, 62(2) :173–185, 2011.

[8] A. Al Ghouwayel. *Contribution to the Study of the Common FFT Operator in Software Radio Context : Application to Channel Coding*. PhD thesis, SUPELEC Rennes, 2008.

[9] H. Wang. *Architectures reconfigurables a base d'operateur CORDIC pour le traitement du signal : Applications aux recepteurs MIMO*. PhD thesis, SUPELEC Rennes, 2009.

[10] L. Alaus. *Architecture Reconfigurable pour un Equipement Radio Multistandard*. PhD thesis, SUPELEC Rennes, 2010.

[11] S.T. Gul. *Optimization of Multi-standards Software Defined Radio Equipments : A Common Operators Approach*. PhD thesis, SUPELEC Rennes, 2009.

[12] P. Kaiser. *Optimization of a Software Defined Radio multi-standard system using Graph Theory*. PhD thesis, SUPELEC Rennes, 2012.

[13] Johann Marquez-Barja, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. An overview of vertical handover techniques : Algorithms, protocols and tools. *Computer Communications*, 34(8) :985 – 997, 2011.

[14] D. Noguet, G. Masera, V. Ramakrishnan, M. Belleville, D. Morche, and G Ascheid. Considering microelectronic trends in advanced wireless system design advances. *Advances in Electronics and Telecommunications*, 1, apr 2010.

[15] J. Mitola. *Cognitive Radio : An integrated Agent Architecture For Software Defined Radio*. PhD thesis, Royal Institute of Technology of Stockholm, 2000.

[16] C. H. (Kees) van Berkel. Multi-core for mobile phones. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 1260–1265, 3001 Leuven, Belgium, Belgium, 2009. European Design and Automation Association.

[17] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn. Magali : A network-on-chip based multi-core system-on-chip for mimo 4g sdr. In *IC Design and Technology (ICICDT), 2010 IEEE International Conference on*, pages 74–77, 2010.

[18] F Clermidy, N Cassiau, N Coste, D Dutoit, M Fantini, D Ktenas, R Lemaire, and L Stefanizzi. Reconfiguration of a 3gpp-lte telecommunication application on a 23-core noc-based system-on-chip. In *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, pages 261–262. ACM, 2011.

[19] A. Wiesler, H. Schober, R. Machauer, and F. Jondral. Software radio structure for umts and second generation mobile communication systems. In *Vehicular Technology Conference, 1999. VTC 1999 - Fall. IEEE VTS 50th*, volume 2, pages 939–942 vol.2, 1999.

[20] A.R. Rhiemeier. Benefits and limits of parameterised channel coding for software radio. In *In Proc. 2nd Workshop on Software Radios*, pages 107–112, Karlsruhe, Germany, March 2002.

[21] K. Strohmenger, M. Laugeois, Noguet D., B. Oelkrug, and K. Seo. Architectures for digital physical layer implementation in multi-mode 3G terminals. In *IST Mobile Communication Summit*, 2004.

[22] J.R. Cavallaro and M. Vaya. Viturbo : a reconfigurable architecture for viterbi and turbo decoding. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 2, pages II–497–500 vol.2, 2003.

[23] J.P. Delahaye. *Plate-forme heterogene reconfigurable : application a la radio logicielle*. PhD thesis, SUPELEC Rennes, 2007.

[24] J. Palicot and C. Roland. Fft : a basic function for a reconfigurable receiver. In *Telecommunications, 2003. ICT 2003. 10th International Conference on*, volume 1, pages 898–902 vol.1, 2003.

[25] L. Alaus, D. Noguet, and Jacques Palicot. A common operator bank to resolve scheduling issue on a complexity optimized sdr terminal. In *Telecommunications (AICT), 2010 Sixth Advanced International Conference on*, pages 142–146, 2010.

[26] Virgilio Rodriguez, Christophe Moy, and Jacques Palicot. Install or invoke ? : The optimal trade-off between performance and cost in the design of multi-standard reconfigurable radios. *Wireless Communications and Mobile Computing*, 7(9) :1143–1156, 2007.

[27] L. Zadeh. Optimality and non-scalar-valued performance criteria. *Automatic Control, IEEE Transactions on*, 8(1) :59–60, 1963.

[28] Suppapitnarm et al. Heuristicly refined multiobjective random search. *Songklanakarin J. Sci. Technol.*, 27(2) :301–312, 2005.

[29] Jack E. Volder. The cordic trigonometric computing technique. *Electronic Computers, IRE Transactions on*, EC-8(3) :330–334, 1959.

[30] J. S. Walther. A unified algorithm for elementary functions. In *Proceedings of the May 18-20, 1971, spring joint computer conference*, AFIPS '71 (Spring), pages 379–385, New York, NY, USA, 1971. ACM.

[31] A.M. Despain. Very fast fourier transform algorithms hardware for implementation. *Computers, IEEE Transactions on*, C-28(5) :333–341, 1979.

[32] A.M. Despain. Fourier transform computers using cordic iterations. *Computers, IEEE Transactions on*, C-23(10) :993–1001, 1974.

[33] S.K. Rao and T. Kailath. Orthogonal digital filters for vlsi implementation. *Circuits and Systems, IEEE Transactions on*, 31(11) :933–945, 1984.

[34] E.F. Deprettere. Synthesis and fixed-point implementation of pipelined true orthogonal filters. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '83.*, volume 8, pages 217–220, 1983.

[35] S.Y. Kung and H.J. Whitehouse. *VLSI and modern signal processing*. Prentice-Hall information and system sciences series. Prentice-Hall, 1985.

[36] P Dewilde, E Deprettere, and R Nouta. Parallel and pipelined vlsi implementation of signal processing algorithms. *VLSI and Modern Signal Processing*, pages 257–276, 1985.

[37] H. Ahmed, M. Morf, D.T. Lee, and P. Ang. A vlsi speech analysis chip set based on square root normalized ladder forms. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '81.*, volume 6, pages 648–653, 1981.

[38] Tze-Yun Sung and Yu-Hen Hu. Vlsi implementation of real-time kalman filter. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 2223–2226, 1986.

[39] Joseph R. Cavallaro and Franklin T. Luk. Cordic arithmetic for an svd processor. *Journal of Parallel and Distributed Computing*, 5(3) :271 – 290, 1988.

[40] H.M. Ahmed, J.-M. Delosme, and M. Morf. Highly concurrent computing structures for matrix arithmetic and signal processing. *Computer*, 15(1) :65–82, 1982.

[41] M. Morf, C. Muravchik, P. Ang, and J.-M. Delosme. Fast cholesky algorithms and adaptive feedback filters. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, volume 7, pages 1727–1731, 1982.

[42] I-Chang Jou, Yu-Hen Hu, and W. S. Feng. A novel implementation of pipelined toeplitz system solver. *Proceedings of the IEEE*, 74(10) :1463–1464, 1986.

[43] M. Renaudin. Architecture d'un operateur cordic. *Rapport de Stage, CNET-Grenoble, CEPHAG*, 1987.

[44] Martin Tomlinson and M.N.A. Abu-Rgheff. The tar decoder-a bandpass viterbi/fft decoder for convolutional encoded spread-spectrum signals. *Communications, IEEE Transactions on*, 44(11) :1392–1398, 1996.

[45] A Michelson. A fast transform in some galois field and an application to decoding reed-solomon codes. In *IEEE International Symposium on Information Theory, Ronneby, Sweden*, page 49, 1976.

[46] Christophe Moy, Jacques Palicot, Virgilio Rodriguez, Denis Giri, et al. Optimal determination of common operators for multi-standards software-defined radio. *Proceedings of WSR'06*, 2006.

[47] Sufi Tabassum Gul, Christophe Moy, and Jacques Palicot. Graphical approach for multi-standards radio design formalization and global optimization. In *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*, pages 116–121. IEEE, 2010.

[48] Patricia Kaiser, Amine El Sahili, and Yves Louët. An optimization algorithm for sdr multi-standard systems using directed hypergraphs. *Frequenz, issn 2191-6349*, 66(9-10) :251–260, 2012.

[49] K.S. Chatha, K. Srinivasan, and G. Konjevod. Automated techniques for synthesis of application-specific network-on-chip architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8) :1425–1438, 2008.

[50] Glenn Leary and Karam S Chatha. Automated technique for design of noc with minimal communication latency. In *Proceedings of the 7th*

IEEE/ACM international conference on Hardware/software codesign and system synthesis, pages 471–480. ACM, 2009.

[51] Anita Tino and Gul N Khan. Multi-objective tabu search based topology generation technique for application-specific network-on-chip architectures. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[52] P. Eles, A. Doboli, P. Pop, and Zebo Peng. Scheduling with bus access optimization for distributed embedded systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(5) :472–491, 2000.

[53] Dong Wu, Bashir M Al-Hashimi, and Petru Eles. Scheduling and mapping of conditional task graph for the synthesis of low power embedded systems. In *Computers and Digital Techniques, IEE Proceedings-*, volume 150-5, pages 262–73. IET, 2003.

[54] Jun Zhu, Ingo Sander, and Axel Jantsch. Constrained global scheduling of streaming applications on mpsocs. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pages 223–228. IEEE Press, 2010.

[55] Alistair Munro. Mobile middleware for the reconfigurable software radio. *Communications Magazine, IEEE*, 38(8) :152–161, 2000.

[56] I. Gomez, V. Marojevic, and A. Gelonch. Aloe : An open-source sdr execution environment with cognitive computing resource management capabilities. *Communications Magazine, IEEE*, 49(9) :76–83, 2011.

[57] Jaesoo Lee, Saehwa Kim, Jiyong Park, and Seongsoo Hong. Q-sca : Incorporating qos support into software communications architecture for sdr waveform processing. *Real-Time Systems*, 34(1) :19–35, 2006.

[58] Software communications architecture (sca) web site. http ://sca.jpeojtrs.mil/.

[59] Ismael Gomez, Vuk Marojevic, Jose Salazar, and Antoni Gelonch. A lightweight operating environment for next generation cognitive radios. In *Digital System Design Architectures, Methods and Tools, 2008. DSD'08. 11th EUROMICRO Conference on*, pages 47–52. IEEE, 2008.

[60] Ismael Gómez Miguelez. A software framework for software radio. *Final Project. UPC, Barcelona, Spain*, 2008.

[61] Youngchul Cho, Ganghee Lee, Sungjoo Yoo, Kiyoung Choi, and N-E Zergainoh. Scheduling and timing analysis of hw/sw on-chip communication in mp soc design. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 132–137. IEEE, 2003.

[62] Vincent Nollet. *Run-time management for future MPSoC platforms*. PhD thesis, PhD thesis, Eindhoven University of Technology, 2008.

[63] Sander Stuijk, T Basten, MCW Geilen, and Henk Corporaal. Multiprocessor resource allocation for throughput-constrained synchronous

dataflow graphs. In *Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE*, pages 777–782. IEEE, 2007.

[64] Zhe Ma and Francky Catthoor. Run-time task overlapping on multiprocessor platforms. *Journal of Signal Processing Systems*, 60(2) :169–182, 2010.

[65] Thomas L. Casavant and Jon G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *Software Engineering, IEEE Transactions on*, 14(2) :141–154, 1988.

[66] S.S. Bhattacharyya. *Handbook of Signal Processing Systems*. Springer US, 2010.

[67] B.M. Al-Hashimi and Institution of Electrical Engineers. *System-on-Chip : Next Generation Electronics*. IEE circuits, devices and systems series. Institution of Engineering and Technology, 2006.

[68] K. Williston. *Digital Signal Processing : World Class Designs : World Class Designs*. World Class Designs. Elsevier Science, 2009.

[69] Yuan Lin, Hyunseok Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. Soda : A low-power architecture for software radio. In *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pages 89–101, 2006.

[70] F. Pollara. Viterbi Algorithm on a Hypercube : Concurrent Formulation. *Telecommunications and Data Acquisition Progress Report*, 84 :249–255, November 1985.

[71] Lihong Jia, Yonghong Gao, J. Isoaho, and H. Tenhunen. Design of a super-pipelined viterbi decoder. In *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, volume 1, pages 133–136 vol.1, 1999.

[72] M. Biver, H. Kaeslin, and C. Tommasini. In-place updating of path metrics in viterbi decoders. *Solid-State Circuits, IEEE Journal of*, 24(4) :1158–1160, 1989.

[73] L. Biard and D. Noguet. An adaptable architecture for the viterbi algorithm. In *The 7th international symposium on wireless personal multimedia communications, Abano Terme, Padova, Italy*, 2004.

[74] Jr. Forney, G.D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3) :268–278, 1973.

[75] Hui-Ling Lou. Implementing the viterbi algorithm. *Signal Processing Magazine, IEEE*, 12(5) :42–52, 1995.

[76] L. Biard. Étude et implantation d'architectures pour les etude et implantation d'architectures pour les communications hiperman. DRT au CEA-LETI, 2004.

[77] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90) :297–301, 1965.

[78] R.B. Perlow and T.C. Denk. Finite wordlength design for vlsi fft processors. In *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, volume 2, pages 1227–1231 vol.2, 2001.

[79] Laurent ALAUS and Dominique NOGUET. Addition/subtraction hardware operator, processor and telecommunication terminal including an operator of this type. Patent Application, 10 2012. US 2012/0263212 A1.

[80] Laurent ALAUS and Dominique NOGUET. Processor for processing digital data with butterfly operator for the execution of an fft/ifft and telecommunication device. Patent Application, 11 2012. US 2012/0281739 A1.

[81] M. Hasan and T. Arslan. Implementation of low-power fft processor cores using a novel order-based processing scheme. *Circuits, Devices and Systems, IEE Proceedings* -, 150(3) :149–154, 2003.

[82] Jarmo Takala and Konsta Punkka. Scalable fft processors and pipelined butterfly units. *Journal of VLSI signal processing systems for signal, image and video technology*, 43(2-3) :113–123, 2006.

[83] J. Takala and K Punkka. Butterfly unit supporting radix-4 and radix-2 fft. In *Proceedings of The 2005 International TICSP Workshop on Spectral Methods and Multirate Signal Processing, SMMSP 2005, Riga, Latvia*, volume 30, pages 47–54, 20-22 June 2005.

[84] J.-Y. Jou and J.A. Abraham. Fault-tolerant fft networks. *Computers, IEEE Transactions on*, 37(5) :548–561, 1988.

[85] Jin-Fu Li, Shyue-Kung Lu, Shih-Arn Hwang, and Cheng-Wen Wu. Easily testable and fault-tolerant fft butterfly networks. *Circuits and Systems II : Analog and Digital Signal Processing, IEEE Transactions on*, 47(9) :919–929, 2000.

[86] Shyue-Kung Lu, Jen-Sheng Shih, and Shih-Chang Huang. Design-for-testability and fault-tolerant techniques for fft processors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(6) :732–741, 2005.

[87] H. Lange, O. Franzen, H. Schröder, M. Bücker, and B. Oelkrug. Reconfigurable multiply-accumulate-based processing element. In *IEEE Workshop on heterogeneous Systems on a Chip, Hamburg, Germany*, 2002.

[88] C.Y. Jung, M.H. Sunwoo, and S.K. Oh. Design of reconfigurable coprocessor for communication systems. In *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 142–147, 2004.

[89] A. Al Ghouwayel, Y. Louet, and Jacques Palicot. A reconfigurable architecture for the fft operator in a software radio context. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–, 2006.

[90] W.J. Dally and B. Towles. *Principles and practices of interconnection networks*. The Morgan Kaufmann Series in Computer Architecture and Design Series. Morgan Kaufmann Publishers, 2004.

[91] Lihong Jia, Yonghong Gao, Jouni Isoaho, and Hannu Tenhunen. An area-efficient acs architecture for viterbi decoders. In *IEEE 4th International Conference on Signal Processing Proceedings ICSP APOS*, pages 525–528, 1998.

[92] G. Miel. Constant geometry fast fourier transforms on array processors. *IEEE Trans. Comput.*, 42(3) :371–375, March 1993.

[93] J. Kwong and M. Goel. A high performance split-radix fft with constant geometry architecture. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1537–1542, 2012.

[94] J. Salinas, C. Feng, and F. Lombardi. Testing constant-geometry fft arrays for wafer scale integration. In *Wafer Scale Integration, 1993. Proceedings., Fifth Annual IEEE International Conference on*, pages 203–212, 1993.

[95] E.C. Chu and A. George. *Inside the FFT black box : serial and parallel fast Fourier transform algorithms*. Computational mathematics series. CRC Press, 2000.

[96] S.F. Gorman and J.M. Wills. Partial column fft pipelines. *Circuits and Systems II : Analog and Digital Signal Processing, IEEE Transactions on*, 42(6) :414–423, 1995.

[97] Yu Tai Ma. A vlsi-oriented parallel fft algorithm. *Signal Processing, IEEE Transactions on*, 44(2) :445–448, 1996.

[98] JA Johnston. Parallel pipeline fast fourier transformer. *Communications, Radar and Signal Processing, IEE Proceedings F*, 130(6) :564–572, 1983.

[99] David T Harper III. Block, multistride vector, and fft accesses in parallel memory systems. *Parallel and Distributed Systems, IEEE Transactions on*, 2(1) :43–51, 1991.

[100] Mario TRAEBER. Method for storing path metrics in a viterbi decoder. Patent, 06 2006. US 7062701.

[101] CB Shung, H-D Lin, PH Siegel, and HK Thapar. Area-efficient architectures for the viterbi algorithm. In *Global Telecommunications Conference, 1990, and Exhibition.'Communications : Connecting the Future', GLOBECOM'90., IEEE*, pages 1787–1793. IEEE, 1990.

[102] Abraham Waksman. A permutation network. *Journal of the ACM (JACM)*, 15(1) :159–163, 1968.

[103] Jarmo Takala, David Akopian, Jaakko Astola, and Jukka Saarinen. Scalable interconnection networks for partial column array processor architectures. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 513–516. IEEE, 2000.

[104] Pei-Yun Tsai and Chung-Yi Lin. A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing fft processors with rescheduling. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(12) :2290–2302, 2011.

[105] Ashoke Deb. Multiskewing-a novel technique for optimal parallel memory access. *Parallel and Distributed Systems, IEEE Transactions on*, 7(6) :595–604, 1996.

[106] Harry AG Wijshoff and Jan Van Leeuwen. The structure of periodic storage schemes for parallel memories. *Computers, IEEE Transactions on*, 100(6) :501–505, 1985.

[107] Gurindar S. Sohi. High-bandwidth interleaved memories for vector processors-a simulation study. *Computers, IEEE Transactions on*, 42(1) :34–44, 1993.

[108] T. Jarvinen, P. Salmela, and J. Takala. Interconnection optimized path metric access scheme for k/n-rate viterbi decoders. In *Signal Processing Systems Design and Implementation, 2005. IEEE Workshop on*, pages 503–508, 2005.

[109] Jean Marc Frailong. Xor-schemes : A flexible data organization in parallel memories. In *Proc. Int. Conf. Prallel Processing*, pages 276–283, 1985.

[110] Tuomas Järvinen. *Systematic methods for designing stride permutation interconnections*. PhD thesis, Tampere University of Technology, 2004.

[111] Lars Wanhammar. *DSP integrated circuits*. Academic press, 1999.

[112] Cheng-Yeh Wang, Chin-Bin Kuo, and Jing-Yang Jou. Hybrid word-length optimization methods of pipelined fft processors. *Computers, IEEE Transactions on*, 56(8) :1105–1118, 2007.

[113] Shingo Yoshizawa and Yoshikazu Miyanaga. Use of a variable word-length technique in an ofdm receiver to reduce energy dissipation. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 55(9) :2848–2859, 2008.

[114] Jaeseong Kim, Shingo Yoshizawa, and Yoshikazu Miyanaga. Variable wordlength soft-decision viterbi decoder for power-efficient wireless lan. *Integration, the VLSI Journal*, 45(2) :132–140, 2012.

[115] Ieee standard definitions and concepts for dynamic spectrum access : Terminology relating to emerging wireless networks, system functionality, and spectrum management. *IEEE Std 1900.1-2008*, pages c1–48, 2008.

[116] Wireless innovation forum. http ://www.wirelessinnovation.org/.

[117] Ahmad RS Bahai, Burton R Saltzberg, and Mustafa Ergen. *Multi-carrier digital communications : theory and applications of OFDM*. Springer Verlag, 2004.

[118] Richard E Blahut. *Algebraic codes for data transmission*. Cambridge university press, 2003.

[119] WC Gore. Transmitting binary symbols with reed-solomon codes. In *Proceedings of Princeton Conference on Information Sciences and Systems, Princeton, NJ*, pages 495–497, 1973.

[120] R Chien and D Choy. Algebraic generalization of bch-goppa-helgert codes. *Information Theory, IEEE Transactions on*, 21(1) :70–79, 1975.

[121] Abraham Lempel and Shmuel Winograd. A new approach to error-correcting codes. *Information Theory, IEEE Transactions on*, 23(4) :503–508, 1977.

[122] RE Blahut. Transform decoding without transform. In *Tenth IEEE Communication Theory Workshop, Cypress Gardens, FL*, 1980.

[123] Martin Tomlinson and MN Ali Abu-Rgheff. The tar decoder-a band-pass viterbi/fft decoder for convolutional encoded spread-spectrum signals. *Communications, IEEE Transactions on*, 44(11) :1392–1398, 1996.

[124] Christian Roland and Jacques Palicot. A self-adaptive universal receiver. *Annales Des Télécommunications*, 57(5-6) :421–456, 2002.

[125] ER Ferrara, C Cowan, and P Grant. Frequency-domain adaptive filtering, 1985.

[126] David Mansour and A Gray Jr. Unconstrained frequency-domain adaptive filter. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 30(5) :726–734, 1982.

[127] Kostas Berberidis and Jacques Palicot. A block quasi-newton algorithm implemented in the frequency domain. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 3, pages 1731–1734. IEEE, 1996.

[128] Shin-Yuan Wang and Chia-Chi Huang. On the architecture and performance of an fft-based spread-spectrum downlink rake receiver. *Vehicular Technology, IEEE Transactions on*, 50(1) :234–243, 2001.

[129] Tim Hentschel. Channelization for software defined base-stations. In *Annales des Telecommunications*, volume 57, pages 386–420. Springer, 2002.