UNIVERSITE BLIDA-1

Faculté de Technologie

THESE DE DOCTORAT

en Electronique

IMPLEMENTATION SUR CARTE FPGA D'UN DECODEUR STBC ET FFT HYBRIDE

Par

Mohammed DALI

devant le jury composé de :

H. MILIANI	Professeur, U. de Blida	Président
M. BENSEBTI	Professeur, U. de Blida	Examinateur
M. CHALLAL	Maître de conférences A, U. de Boumerdes	Examinateur
K. GHANEM	Directrice de rechechce, C.D.T.A, Alger	Examinatrice
A. GUESSOUM	Professeur, U. de Blida	Directeur de thèse
A. AMIRA	Professeur, U. West of Scotland, UK.	Co-Directeur de thèse

Blida, 28 Juin 2018

ملخص:

يندرج عملنا البحثي هذا في إطار المساهمة في تصميم و تنفيذ نماذج دارات لنظام اتصالات عالي التدفق يحوي الثنائية *STC-OFDM* أو الثلاثية *STC-OFDM-CDMA* باستعمال الدارة *FPGA*. في هذا السياق كان هدفنا هو اقتراح نماذج فعالة و تسمح بضمان تدفق عال للمعطيات لمكونين أساسيين من مكونات النظام. المكون الأول هو المكلف بفك ترميز Alamouti أما المكون الثاني فهو الذي يقوم بتنفيذ خوارزمية *FFT*

بخصوص المكون الأول قمنا باقتراح نموذجين يسمحان بتخفيض عدد حواسب الجمع و الجداء المركبين. كلا النموذجان يمكن استعمالهما في نظام Alamouti 2×1 أو 2×2.

أما فيما يخص المكون الثاني فقد اقترحنا نماذج ترتكز على مقاربة MDC و الأساس المختلط 2^2-2^3 من أجل تنفيذ خوارزمية FFT لسلسلتين أو أربعة سلاسل مستقلة للمعطيات.

نتائج تنفيذ كلا المكونين على دارة FPGA أظهرت أداءا عاليا من حيث سرعة تدفق المعطيات و فعالية من حيث استعمال الموارد و الطاقة.

كلمات مفتاحية:

MIMO, OFDM, CDMA, Alamouti, FPGA, MDC, FFT, STC.

RESUME

Notre travail s'inscrit dans le cadre de la contribution à l'implémentation efficace sur *FPGA* « Field Programmable Gate Array » d'un système de communication à haut débit qui utilise la combinaison *STC-OFDM* « Space Time Coding- Orthogonal Frequency Division Multiplex » ou *STC-OFDM-CDMA* « Space Time Coding- Orthogonal Frequency Division Multiplex- Code division Multiple Access ». Dans ce contexte, nous avons proposé et implémenté sur circuit *FPGA* récent, des architectures efficaces et à haut débit pour deux blocs élémentaires du système qui sont le décodeur espace temps d'Alamouti et le bloc *FFT*. Pour le décodeur d'Alamouti, nous avons proposé et implémenté deux architectures à haut débit qui réduisent le nombre d'additionneurs et de multiplieurs utilisés. Les architectures sont configurables pour le système Alamouti 2×1 et 2×2. Pour le bloc *FFT,* qui est le cœur du modulateur *OFDM,* nous avons proposé des architectures de type *MDC* « Multi *path delay commutator* » basées sur l'utilisation de radix mixé 2^2-2^3 et qui permettent de traiter deux ou quatre séquences indépendantes pour toutes les tailles qui sont une puissance de deux. Les

résultats d'implémentation des architectures, pour le bloc *FFT* et pour le bloc de décodage d'Alamouti, montrent que ces implémentations assurent un débit élevé avec une efficacité importante en ressources et en puissance consommée.
Mots clé : STC, MIMO, OFDM, CDMA, décodeur d'Alamouti, FPGA, MDC, FFT.

ABSTRACT

This work is a contribution to an efficient implementation of a high throughput system using Space Time Coding- Orthogonal Frequency Division Multiplex *(STC-OFDM)* or Space Time Coding- Orthogonal Frequency Division Multiplex- Code division Multiple Access *(STC-OFDM-CDMA)* combination on a Field Programmable Gate Array (FPGA) circuit. In this context, we have proposed and implemented on recent FPGA circuit, an efficient and high throughput architectures for two elementary blocks of the system, which are the Alamouti decoder, and the FFT block. For Alamouti decoder we proposed and implemented two high throughput and efficient architectures that reduce the number of adders and multipliers. Furthermore, the architectures are scalable for both Alamouti systems 2×1 and 2×2. For the *FFT* block; which is the basic block of the *OFDM* modulator; we proposed architectures based on Multi path Delay Commutator (MDC) approach using mixed Radix 2^2-2^3 which allow processing two or four independent sequences for all power of two sizes. The implementation results of the architectures for both *FFT* and Alamouti decoder show high throughput performance with resources and power consumption efficiency.
Keywords: *STC, MIMO, OFDM, CDMA, Alamouti decoder, FPGA, MDC, FFT.*

REMERCIEMENTS

Je commence avant tout, par remercier ALLAH le tout puissant, de m'avoir donné la force pour réaliser ce travail.

Puis, je tiens à remercier mon directeur de thèse, le professeur Abderrezak GUESSOUM, pour ses conseils, sa gentillesse et sa disponibilité durant toutes les années de préparation de cette thèse.

Mes sincères remerciements vont à mon co-directeur de thèse, le professeur Abbes AMIRA, pour son aide continue, sa disponibilité, et son support. Je le remercie de m'avoir accueilli au laboratoire « Visual Communication, University of the West of Schotland (UWS), UK. » où l'essentiel du travail de cette thèse a été accompli.

J'adresse mes remerciements à tous les membres du jury. Je remercie le professeur H. MILIANI de m'avoir fait l'honneur de présider le jury, le professeur M. BENSEBTI, le docteur M. CHALLAL et la docteure K. GHANEM d'avoir accepté d'être les examinateurs de cette thèse.

Je tiens à remercier aussi le professeur Naeem Ramzan qui m'a beaucoup aidé durant mon séjour à l'université *UWS*. Je le remercie pour ses orientations, ses conseils et ses discussions qui étaient très bénéfiques.

Mes remerciements vont aussi au docteur Rayan Gibson qui a partagé avec moi beaucoup de son savoir dans le domaine d'implémentation sur circuit FPGA.

Un grand merci pour le professeur Vahid Meghdadi de l'Ecole Nationale Supérieure d'Ingénieur de Limoge (ENSIL), qui m'avait accueilli dans son laboratoire et m'avait donné tous les moyens de travail durant un mois de séjour.

Un grand merci à mes parents, à ma femme et à mes enfants pour leur soutien, leur patience et leur compréhension, ainsi qu'à tous les membres de ma famille.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION GENERALE

Le développement des systèmes de télécommunications connaît un progrès très rapide surtout durant les deux dernières décennies. A cause de la diversification des applications et de types des signaux transmis, les exigences du marché des télécommunications en termes de débit de transmission et en qualité de service (*QoS*) ne cessent d'augmenter continuellement afin de répondre aux besoins des utilisateurs. Les prévisions de l'indice de réseau visuel « Visual Networking Index » *VNI* de Cisco indiquent que le trafic des réseaux mondiaux de mobiles passera de 7 Exabytes (*EB*) par mois ; i.e. 7×10^{18} bytes/mois ; en 2016 à 49 *EB* en 2021, comme le montre la figure 1 [1]. Cette augmentation explosive du trafic est causée par la prévalence de Smartphones, Laptops et Tablettes ainsi que par l'émergence des communications de type machine à machine (*M2M*)…

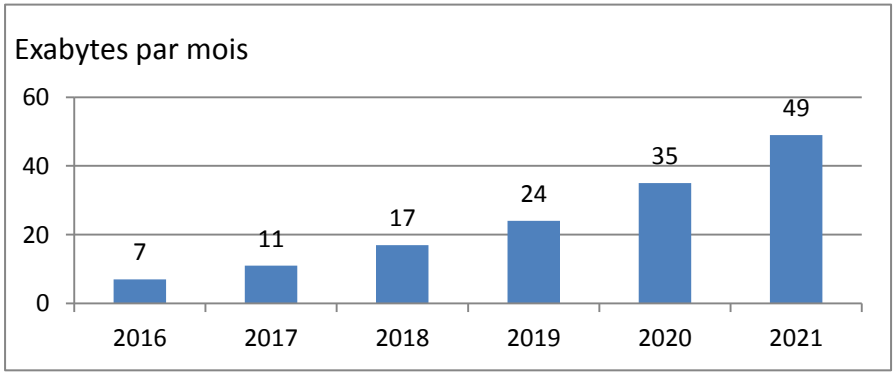

Figure 1 : Prévision de trafic de données des réseaux mondiaux de mobiles selon Cisco *VNI*.

L'augmentation du trafic ainsi que les exigences en *QoS* ont fait du spectre radio fréquence l'une des ressources rares et précieuses, qu'on doit exploiter le plus efficacement possible. Par conséquent, la couche physique des systèmes de

communication doit être capable de fournir de plus en plus de capacité en termes d'efficacité spectrale élevée, du haut débit et de grand nombre d'utilisateurs qui partagent simultanément la même bande de fréquence.

L'utilisation de techniques avancées de communication numérique est donc indispensable, afin de concevoir des systèmes répondants aux contraintes citées précédemment. Les techniques qui ont connu un grand succès dans ce contexte sont la technique multi-antennaires « Multiple Input Multiple Output » *MIMO* [2]-[5], la technique de multiplexage par répartition en fréquences orthogonales « Orthogonal Frequency-Division Multiplexing » *OFDM* [6]-[8] et la technique d'accès multiple par répartition de code « Code Division Multiple Access » *CDMA* [8]-[10].

La technique *MIMO* permet d'exploiter la dimension spatiale afin d'améliorer le débit du système par le multiplexage spatial, qui consiste à envoyer plusieurs symboles simultanément sur les différentes antennes d'émission. La technologie *MIMO* permet aussi d'améliorer la robustesse du système par l'utilisation du codage spatio-temporel « Space Time Coding » *STC* [11], [12], qui utilise le principe de la diversité spatio-temporelle pour réduire la probabilité des erreurs de transmission. L'*OFDM* exploite la dimension fréquentielle pour envoyer l'information haut débit sur plusieurs sous-porteuses orthogonales au lieu d'une seule porteuse. Cet aspect permet donc de convertir le canal sélectif en fréquence en plusieurs sous-canaux plats (non sélectifs en fréquence). L'utilisation de fréquences orthogonales réduit l'occupation spectrale des sous-porteuses ce qui augmente l'efficacité spectrale du système. La technique *CDMA* consiste à envoyer simultanément les données de plusieurs utilisateurs sur la même bande de fréquence en exploitant la diversité de code. Cette technique est largement utilisée dans les réseaux de la troisième génération.

Dans le but de tirer profit des avantages des techniques précédentes, les chercheurs ont proposé des schémas combinant deux ou trois de ces techniques. Ainsi, on trouve des combinaisons de *MIMO* et de l'*OFDM* appelées *MIMO-OFDM* [13], des combinaisons de *CDMA* et de l'*OFDM* comme le *MC-CDMA* [8],[14] et des combinaisons de *MIMO*, de *CDMA* et de l'*OFDM* [15]-[17].

L'utilisation des techniques déjà citées ci-dessus permet d'améliorer considérablement les performances des systèmes. Cependant, cette amélioration n'est pas sans prix, car leur intégration implique d'effectuer de complexes opérations en temps réel. Par conséquent, il est nécessaire d'utiliser des circuits performants avec une certaine flexibilité et avec un coût de réalisation acceptable. Le défi pour les concepteurs est d'implémenter efficacement ; sur des circuits appropriés ; des systèmes utilisant les techniques *MIMO, CDMA* et *OFDM,* dans un contexte de haut débit et en assurant une qualité de service, une flexibilité et un coût acceptables.

La révolution qu'a connue l'électronique numérique ces dernières années a permis de développer et de proposer divers types de processeurs de traitement et de calcul pour les concepteurs des systèmes de télécommunications. Chaque processeur est caractérisé par ces performances, son degré de flexibilité, son niveau de consommation de puissance, son coût de réalisation et par l'ordre de difficulté de sa mise en œuvre. Selon le type de l'application, un bon compromis entre plusieurs caractéristiques est souhaitable. Sur l'une des extrémités de l'ensemble des solutions proposées on trouve les processeurs à usage général « General Purpose Processor » *GPP* et les processeurs de signal numérique « Digital Signal Processor » *DSP,* qui assurent une grande flexibilité du fait qu'ils sont programmables, mais ils sont moins efficace en consommation de puissance et en performances. Sur l'autre extrémité du spectre, on trouve les circuits intégrés propre à une application « Application Specific Integrated Circuits » *ASIC*, appelés aussi circuits intégrés développés pour un client, qui sont des circuits spécialisés dédiés à une application bien spécifiée et dont la configuration est effectuée une seule fois par le fabriquant. Les *ASIC*s sont efficaces en performances et en puissance, mais ils sont moins flexibles. Une solution entre les deux extrémités qui permet un bon compromis entre performances et puissance d'un côté et flexibilité de l'autre côté, et qui a connu une grande émergence ces dernière année consiste à utiliser des circuits intégrés à architecture reconfigurable. Les systèmes reconfigurables sont implémentés par la logique programmable, ce qui permet la réalisation de circuits à travers la configuration par programme des interconnexions entres les ressources.

La réalisation des systèmes reconfigurables est rendue possible par l'introduction des circuits *FPGA* « Field Programmable Gate Array » [18]. Depuis leur première apparition, les circuits de type *FPGA* ont connu un progrès énorme que ce soit sur le plan de leurs performances, leur coût et leur consommation de puissance ou sur le plan des logiciels et des outils de conception développés pour faciliter et accélérer la mise en œuvre des applications à base de circuits *FPGA*. Ce progrès a encouragé les concepteurs et les chercheurs dans le domaine de télécommunication à proposer des implémentations sur *FPGA* de divers techniques de la communication numérique comme l'*OFDM*, le *CDMA*, le *MIMO*…

Notre travail entre dans le cadre de la contribution à l'implémentation de deux blocs clés des systèmes de communication combinant le codage spatio-temporel avec l'*OFDM,* et les systèmes combinant le codage spatio-temporel, l'*OFDM* et le *CDMA*. Le premier bloc est considéré parmi les blocs les plus importants du point de vue complexité arithmétique, il s'agit du bloc *FFT/IFFT* « Fast Fourier Transform/ Inverse Fast Fourier Transform » [19] qui est le cœur du modulateur et du démodulateur *OFDM*. Le deuxième bloc est le décodeur spatio-temporel, qui est indispensable au niveau de la réception pour le décodage des signaux reçus. L'organisation de cette thèse se repose sur quatre chapitres :

Le premier chapitre évoque les notions de base de la communication numérique, en commençant par la présentation des différents blocs qui constituent une chaine de communication simple. Les caractéristiques des canaux radio, leurs modèles ainsi que les principaux phénomènes de propagation en relation comme l'effet Doppler, la dispersion fréquentielle et la dispersion temporelle sont ensuite présentées. Finalement, ce chapitre se termine par l'explication des aspects de base relatifs aux trois techniques : *MIMO, OFDM* et *CDMA*.

La première partie du deuxième chapitre est consacrée à la présentation du codage espace-temps. Dans cette partie, nous abordons le principe du codage spatio-temporel, et nous donnons aussi une explication sur deux familles de codes qui sont les codes espace-temps en bloc *STBC* et les codes espace-temps en treillis. Dans cette même partie, nous présentons aussi un cas particulier des codes *OSTBC*

qui est le code d'Alamouti vu sa large utilisation. Dans la deuxième partie du chapitre 2, nous allons donner un aperçu de la combinaison du codage espace-temps avec la technique *OFDM* et la technique *CDMA*. Dans la dernière partie nous présentons les deux architectures que nous avons proposées pour l'implémentation du décodeur d'Alamouti sur un circuit *FPGA.* Nous commençons par donner les schémas des architectures avec explications de leur fonctionnement, en suite nous présentons les résultats de simulations et nous terminons par les résultats de l'implémentation des architectures sur un circuit *FPGA* de type Virtex 7.

Le troisième chapitre porte sur l'étude des algorithmes *FFT* du point de vue algorithmique et architecture d'implémentation. Ainsi, le principe et la complexité arithmétique des variantes Radix-2 et Radix-4 de l'algorithme *FFT* sont présentés dans ce chapitre. Après présentation de l'aspect algorithmique de la *FFT* nous donnons ensuite, les architectures en pipeline et les architectures à base de mémoire, qui sont proposées dans la littérature pour l'implémentation de l'algorithme *FFT* sur cible *FPGA* ou *ASIC*. Le troisième chapitre se termine par l'état de l'art des architectures *FFT* proposées dans le contexte du *MIMO-OFDM*.

Le quatrième chapitre présente les architectures *MDC* « Multi path Delay Commutator» à haut débit, basées sur l'utilisation du Radix mixé 2^2-2^3, que nous avons proposées pour l'implémentation d'un processeur *FFT* capable de traiter jusqu'à quatre séquences indépendantes, en utilisant quatre chemins de traitement. Nous commençons par un rappel sur les deux algorithmes Radix-2^2 et Radix-2^3. Par la suite, nous expliquons avec détail les différents modules de conception des architectures proposées. Nous donnons aussi les ressources utilisées par ces architectures, en comparaison avec les architectures existantes. Nous terminons le quatrième chapitre par la présentation des résultats d'implémentation des architectures proposées pour les différentes valeurs de la taille du *FFT,* sur les deux circuits *FPGA* Virtex 5 et Virtex 7. Ces résultats sont aussi comparés avec des travaux récents.

Ce document se termine par une conclusion qui résume les principaux résultats obtenus, ainsi que les perspectives du travail présenté.

Chapitre1

ASPECTS DE BASE DE LA COMMUNICTION RADIO MOBILE

1.1 Introduction

Ce premier chapitre a comme objectif la présentation des éléments de base relatifs à la communication radio mobile, afin de comprendre le contexte de notre travail. Nous commençons par la définition des blocs élémentaires d'une chaîne de communication numérique. Ensuite, nous introduisons les phénomènes liés à la propagation radio, ainsi que les différents types et les différents modèles des canaux radio. Dans la dernière partie du présent chapitre, nous présentons trois techniques : *MIMO*, *OFDM* et *CDMA* qui servent à améliorer les performances des systèmes de télécommunications.

1.2. Constitution d'une chaine de communication numérique

Un système de communication numérique a pour objectif la transmission de l'information de la source vers le destinataire, avec une grande fiabilité et un coût raisonnable. Quelle que soit la complexité du système de communication, on peut le décomposer en trois parties qui sont : l'émetteur, le récepteur et le canal de transmission, comme illustré dans la figure 1.1.

L'émetteur permet d'associer au message issu de la source ; qui est une grandeur abstraite ; une grandeur physique adaptée au canal de transmission. Il est constitué essentiellement des blocs suivants :

- Codeur de source *:* il permet d'éliminer la redondance de la source, afin de minimiser le nombre d'éléments binaires nécessaires à la représentation de

l'information délivrée par la source.

- Codeur de canal : il a pour rôle la détection et la correction des erreurs de transmission, pour améliorer la robustesse du système de transmission vis-à-vis des imperfections du canal de transmission. Le codage de canal consiste à ajouter aux éléments binaires qui représentent l'information d'autres éléments binaires de redondance, selon une loi bien déterminée afin de détecter et corriger les erreurs. Les codeurs de canal sont classés généralement en deux grandes familles, à savoir les codeurs en bloc et les codeurs convolutifs.

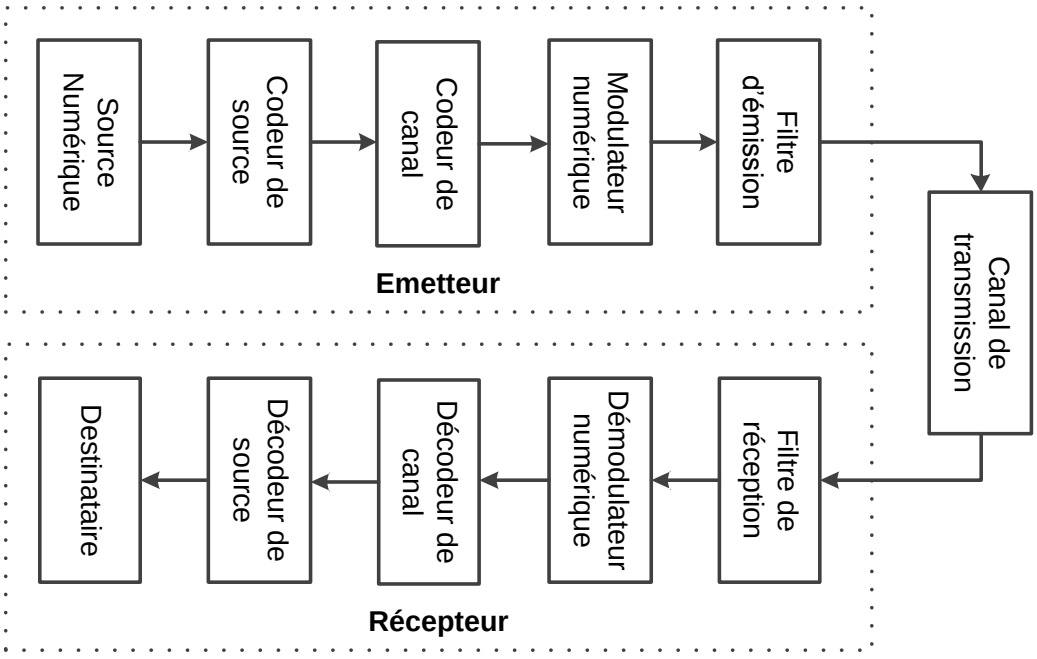

Figure 1.1 : Schéma de base d'une chaine de communication numérique.

- Modulation Numérique : la modulation numérique transforme la suite des éléments binaires en un signal numérique, elle est réalisée à travers deux étapes qui sont le codage bits à symbole et le codage symbole à signal.
  - Codage bits à symbole « mapping » : cette opération consiste à associer à chaque groupe de *n* éléments binaires ; appelé n-uplet ; un symbole qui est un élément d'un alphabet de $M=2^n$ symboles. L'affectation des différents

groupes de *n* éléments binaires aux symboles correspondants est généralement représentée par un graphe appelé *constellation*. Le choix de la constellation dépend de plusieurs paramètres tels que le type de canal de transmission, le taux d'erreur exigé …

- Codage symbole à signal : c'est l'association à chaque symbole de l'alphabet M-aire un signal modulant qui sera le support physique porteur de l'information.

  Selon le type de la modulation, l'information est inscrite sur l'un des paramètres du signal porteur qui sont la fréquence, la phase ou l'amplitude.

- Le filtrage de l'émission : cette fonction permet la mise en forme du signal porteur de l'information, en limitant son occupation spectrale pour éviter les interférences entre symboles *ISI* « Inter Symbol Interference ».

Le canal représente le support physique qui permet la transmission du signal porteur de l'information de l'émetteur vers le récepteur. Dans le cas des communications radio mobiles, le milieu de transmission est constitué de l'espace libre qui assure la propagation de l'onde électromagnétique porteuse de l'information. En communication numérique, toutes les sources de perturbation qui affectent le signal porteur de l'information au cours de sa transmission sont considérées comme parties du canal. Ainsi, les perturbations des antennes à l'émission et à la réception, les perturbations du canal de propagation, et même les imperfections de la partie *RF* « Radio Frequency » sont considérées comme parties du canal de communication. Il est important de connaitre toutes ces sources de perturbations et de leurs attribuer un modèle fiable et fidèle afin de faciliter la conception du système de communication.

Le récepteur est la partie du système de communication qui permet la restitution fidèle de l'information issue de la source. Généralement, le récepteur réalise les fonctions inverses des fonctions que réalise l'émetteur. Donc, on trouve les fonctions de démodulation, de décodage de canal et de décodage de source.

Il est important de noter que nous n'avons présenté dans ce paragraphe que les éléments les plus communs d'une chaîne de communication numérique simple. Selon l'ordre de complexité du système, selon le type de l'information à transmettre et selon d'autres contraintes imposées aux concepteurs, on peut trouver d'autres fonctions de traitement, comme l'étalement de spectre, le multiplexage temporel, spatial et fréquentiel…

1.3 Canal radio et phénomènes liés à la propagation

La puissance de l'onde porteuse de l'information subit des fluctuations et des pertes en fonction du temps et en fonction de la distance parcourue lors de propagation de cette onde de l'émetteur vers le récepteur. Ces pertes et fluctuations sont le résultat de plusieurs phénomènes qui sont classés en deux types à savoir les phénomènes à grande échelle et les phénomènes à petite échelle.

1.3.1 Phénomènes à grande échelle

Ils correspondent à l'effet de la variation de la puissance moyenne mesurée sur une distance qui est grande par rapport à la longueur de l'onde λ. Ce type de variation est lié à deux phénomènes, qui sont le masquage « shadowing » et les pertes liées à la distance parcourue « path loss ». Les pertes liées à la distance se manifestent lors de la propagation dans l'espace libre, et elles sont proportionnelles au carré de la distance parcourue. L'effet de masquage est dû à la présence d'obstacles entre l'émetteur et le récepteur. Cet effet se manifeste par des fluctuations lentes de la puissance appelées évanouissement à grande échelle « large scale fading », en plus de sa décroissance qui est due aux pertes liées à la distance.

1.3.2 Phénomènes à petite échelle

Les variations de puissance à petite échelle sont observées sur des distances suffisamment petites. Ces fluctuations sont causées par l'existence d'obstacles lors de la propagation de l'onde radio, ce qui entraîne l'atténuation, la réflexion et la réfraction de cette onde. Donc, le récepteur reçoit plusieurs ondes arrivant de

directions différentes avec des amplitudes, des retards et des phases différentes. On parle donc de la propagation multi-trajets « multipath propagation » ; où les variations en amplitude de l'onde résultante sont appelées évanouissement à petite échelle « small scale fading ».

1.3.3 Bruit radio électrique

Le bruit désigne tous les signaux qui perturbent le ou les signaux utiles, et qui ne portent aucune information utile pour le récepteur. En communication radio mobile, le bruit englobe l'ensemble des perturbations souvent de nature aléatoire qui sont liées au canal de propagation (bruit externe) et aux dispositifs électroniques du système de communication (bruit interne). Le bruit externe inclut les différentes ondes qui interfèrent avec l'onde utile et qui sont captées par l'antenne de réception. Le bruit interne est de nature thermique, il est modélisé selon le théorème de la limite centrale par un processus aléatoire Gaussien [20]. Dans la plupart des cas, le bruit interne et le bruit externe sont tous les deux représentés par un seul signal appelé bruit blanc additif Gaussien « Additif White Gaussien Noise » *AWGN.* Le mot blanc vient du fait que la densité spectrale de puissance de ce type de bruit est uniforme sur une très large bande de fréquence qui s'étend à 10^{13} *Hz,* et le mot additif signifie que le bruit s'ajoute au signal utile reçu. Lors de l'évaluation d'un système de télécommunication radio mobile l'effet du bruit *AWGN* est étudié en calculant ou en estimant le rapport entre la puissance du signal utile et celle du *AWGN,* qui est appelé rapport signal sur bruit « Signal to Noise Ratio » *SNR*.

1.3.4 Dispersion temporelle et bande de cohérence du canal

L'une des caractéristiques du canal radio est le retard que subit un signal avant son arrivée au récepteur. Ce retard est de nature aléatoire, il est caractérisé par ces moments d'ordre un et d'ordre deux qui sont la moyenne « mean delay » et la dispersion « delay spread ». La dispersion des retards dans le domaine temporel se traduit par une cohérence dans le domaine fréquentiel. Ainsi, la bande de cohérence « coherence bandwidth » du canal est définie comme étant l'écart entre deux fréquences différentes pour un degré de corrélation bien déterminé. Du point de vue

pratique, la bande de cohérence désigne l'espacement fréquentiel minimum à imposer entre deux signaux afin qu'ils subissent deux évanouissements indépendants. Pour un degré de corrélation donné, la bande de cohérence du canal est inversement proportionnelle à la dispersion des retards. Il est important de noter que l'augmentation de la dispersion des retards conduit au phénomène d'interférence entre symboles « Inter Symbol Interference » *ISI,* qui dégrade les performances du système de communication.

1.3.5 Dispersion fréquentielle et temps de cohérence du canal

En communication radio mobile, les mouvements de l'émetteur, du récepteur, et même des obstacles entrainent des décalages de fréquence par l'effet Doppler [21]. Le décalage de fréquence appelé fréquence Doppler est proportionnel à la vitesse relative de déplacement. Ce décalage conduit au changement de la fréquence de la porteuse, ce qui rend la restitution de la porteuse, au niveau de la réception, difficile. La dispersion par l'effet Doppler dans le domaine fréquentiel se traduit par une cohérence dans le domaine temporel. Ainsi, le temps de cohérence « coherence time » du canal pour une fréquence donnée et pour une position fixe des antennes émettrices et réceptrices est défini comme étant la durée pour laquelle l'enveloppe des oscillations du canal reste invariante. Le temps de cohérence du canal est inversement proportionnel à la fréquence de Doppler maximale, son expression est généralement donnée par une relation empirique. Cela implique que l'augmentation de la mobilité du récepteur et des obstacles conduit à la réduction du temps de cohérence du canal.

Le temps de cohérence du canal et sa bande de cohérence permettent de déterminer le type du canal. En fonction de ces deux paramètres, on définit la notion de sélectivité du canal en temps et en fréquence comme suivant :

- *Sélectivité en fréquence du canal* **:** le canal est sélectif en fréquence « frequency selectif » si la bande du signal à transmettre est plus large que la bande de cohérence du canal. Dans le cas contraire le canal est dit non-sélectif en fréquence ou à évanouissement plat « Flat fading ».

- *Sélectivité en temps du canal* **:** le canal est dit sélectif en temps ou à évanouissements rapides « fast fading » si la durée d'un symbole est supérieure au temps de cohérence du canal. Dans le cas contraire le canal est dit non-sélectif en temps ou à évanouissements lents « slow fading ».

En fonction de la sélectivité en temps et la sélectivité en fréquence les canaux à évanouissement peuvent être classés en quatre catégories selon la figure 1.2.

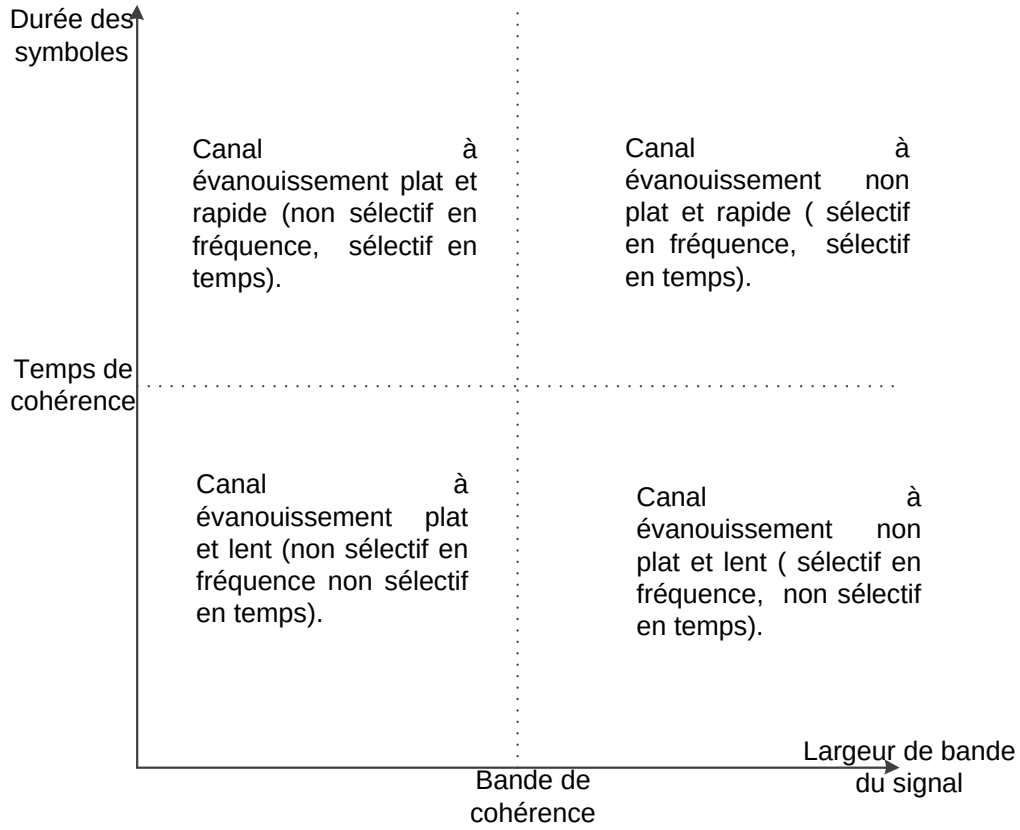

Figure 1.2 : Illustration des quatre types de canaux à évanouissement suivant la bande de cohérence et le temps de cohérence du canal.

1.3.6 Modélisation du canal de transmission

Les paramètres influant sur le canal de propagation sont nombreux et de nature aléatoire, ce qui rend l'établissement d'un modèle statistique fiable du canal très

difficile en pratique. Afin de faciliter la modélisation, des hypothèses simplificatrices sur les phénomènes liés au canal de propagation radio sont à envisager [22], [23]. Ces hypothèses sont :

- *Stationnarité au sens large* : cette hypothèse suppose que les évanouissements du canal radio restent constants pour des courtes durées et des faibles distances, ce qui permet d'assimiler le canal à un processus stochastique stationnaire au sens large « Wide Sens Stationary » *WSS.*
- *Non-corrélation des diffuseurs* **:** les diffuseurs sont considérés comme étant non corrélés « Uncorrelated Scattering » *US,* si deux réponses impulsionnelles du canal pour deux retards différents sont décorrélées en temps ou si deux signaux ayant deux angles d'incidence différents sont décorrélés en espace.
- *Ergodicité* **:** cette hypothèse suppose que les moyennes d'ensemble sont identiques aux moyennes temporelles ou aux moyennes fréquentielles.

Si les trois hypothèses citées précédemment sont valables, le canal radio est nommé *ST-WSS-US* « Spatio Temporel Wide Sens Stationary Uncorrelated Scattering ». Pour ce type de canaux les statistiques d'ordre deux dans les trois domaines naturels ; qui sont le temps, la fréquence et l'espace ; sont identiques.

Le canal radio à évanouissements suit le modèle statistique de Rayleigh s'il n'existe pas de trajets directs, « Line Of Site » *LOS,* entre l'émetteur et le récepteur, ce qui est le cas dans la plupart des situations pratiques. Pour un canal à évanouissement plat et ayant *I* trajets, le signal reçu peut être exprimé sous la forme suivante [11] :

$$r(t) = \sum_{i=1}^{I} a_i . \cos(2.\pi . f_c t + \varphi_i) + n(t) \qquad (1.1)$$

Avec a_i et φ_i sont respectivement, l'amplitude et la phase du signal correspondant au $i^{ème}$ trajet, f_c est la fréquence porteuse et $n(t)$ représente le bruit *AWGN*.

L'expansion du Cos(.) permet de réécrire l'équation (1.1) sous la forme suivante :

$$r(t) = \cos(2.\pi . f_c . t).\underbrace{\sum_{i=1}^{I} a_i . \cos(\varphi_i)}_{A} - \sin(2.\pi . f_c . t).\underbrace{\sum_{i=1}^{I} a_i . \sin(\varphi_i)}_{B} + n(t) \qquad (1.2)$$

A et *B* dans l'équation (1.2) représentent la sommation de *I* variables aléatoires. Pour une valeur de *I* suffisamment grande, et d'après le théorème de la limite centrale, *A* et *B* sont des variables aléatoires Gaussiennes indépendantes et identiquement distribuées « Independent Identically Distributed » *IID*. Cela implique que l'enveloppe du signal reçu $R = \sqrt{A^2 + B^2}$ suit la distribution de Rayleigh dont la fonction de densité de probabilité « Probability Density Function » *PDF* est donnée par :

$$f_R(r) = \frac{1}{\sigma^2}.\exp\left(\frac{-r^2}{2.\sigma^2}\right), \qquad \mathrm{r} \geq 0 \tag{1.3}$$

Où σ^2 est la variance des deux variables aléatoires *A* et *B*.

En cas d'existence de trajet direct entre l'émetteur et le récepteur, les variables aléatoires Gaussiennes *A* et *B* ont des moyennes non nulles. Dans ce cas, l'enveloppe est une variable aléatoire suivant la distribution de Rice, dont l'expression de la *PDF* est :

$$f_R(r) = \frac{r}{\sigma^2}.\exp\left(\frac{-(r^2 + D^2)}{2.\sigma^2}\right).I_0\left(\frac{D.r}{\sigma^2}\right), \qquad \mathrm{r} \geq 0, \mathrm{D} \geq 0 \tag{1.4}$$

Avec *D* dénote l'amplitude maximale du signal dominant (trajet direct) et I_0 est la fonction de Bessel modifiée d'ordre zéro.

Les équations (1.1) et (1.2) donnent le signal *r(t)* analogique au niveau du premier étage du récepteur. Après l'opération de la démodulation numérique on obtient une version du signal discret en bande de base qu'on peut écrire ; pour un canal non sélectif en fréquence ; sous la forme suivante [11] :

$$r = h.s + n \tag{1.5}$$

Où *s* et *n* sont les versions discrètes de *s(t)* et *n(t),* et *h* est une variable aléatoire complexe appelée gain du trajet « path gain », dont le module suit la distribution de Rayleigh ou de Rice suivant l'existence ou pas d'un trajet direct *LOS.*

Pour les canaux sélectifs en fréquence, le signal reçu est donné par l'expression suivante :

$$r_t = \sum_{j=0}^{J-1} h_j . s_{t-j} + n_t \qquad (1.6)$$

Le canal à évanouissement souffre du phénomène de diminution de la puissance du signal reçu qui peut atteindre pour certains cas d'environnements urbains 20 à 30 dB. Puisque la puissance de l'*AWGN* ne change pas significativement au niveau du récepteur, le rapport *SNR* subit donc une diminution dramatique. Généralement, il y a une valeur minimale du *SNR* au-dessous de laquelle le récepteur ne peut plus détecter le signal utile. Ce phénomène est appelé coupure de service « outage ». L'une des solutions efficaces pour minimiser la probabilité de coupure est l'utilisation d'un système multi-antennaires *MIMO* qui sera l'objet de la prochaine section.

1.4 Système multi antennaires *MIMO*

Les systèmes de communication *MIMO* sont apparus entre la fin des années 80 et le début des années 90 grâce aux travaux de J. Winters, [24] , G. J. Foschini et M. J. Gans [25]. Ils sont basés sur l'utilisation de N_t antennes au niveau de l'émetteur et de N_r antennes au niveau du récepteur, contre une seule antenne émettrice et une seule antenne réceptrice pour un système *SISO* « Single Input Single output ». Le canal radio existant entre l'émetteur et le récepteur est, lui aussi, appelé canal *MIMO* (figure 1.3).

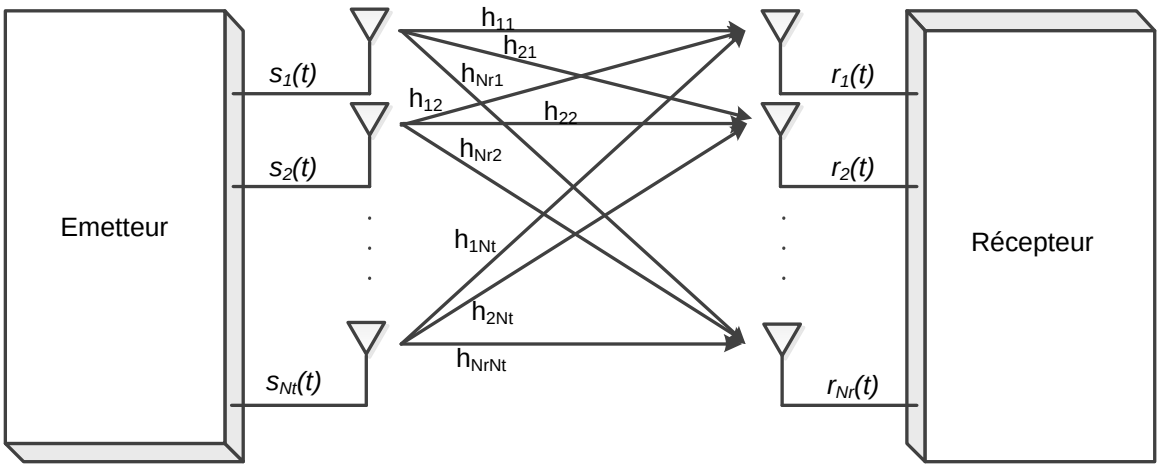

Figure 1.3 Schéma d'illustration d'un système *MIMO*.

La relation entrée-sortie pour un canal *MIMO* s'écrie sous la forme matricielle suivante :

$$\boldsymbol{r} = \boldsymbol{H}.\boldsymbol{s} + \boldsymbol{n} \tag{1.7}$$

Où : $\boldsymbol{r} = \left[r_1, r_2 \ldots r_{N_r}\right]^T$ est le vecteur des signaux reçus par les N_r antennes de réception, $\boldsymbol{s} = \left[s_1, s_2 \ldots s_{N_t}\right]^T$ est le vecteur des signaux émis par les N_t antennes d'émission, $\boldsymbol{n} = \left[n_1, n_2 \ldots n_{Nr}\right]^T$ représente le vecteur des bruits *AWGN* présents sur les N_r antennes de réception, et $\boldsymbol{H}$ est la matrice qui représente le canal *MIMO* à évanouissements, dont sa structure est donnée par :

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & . & . & . & h_{1N_t} \\ h_{21} & h_{22} & . & . & . & h_{2N_t} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ h_{Nr1} & h_{N_r 2} & . & . & . & h_{N_r N_t} \end{bmatrix} \tag{1.8}$$

Chaque paramètre h_{ij} représente le gain du trajet entre la i^{eme} antenne de réception et la j^{eme} antenne d'émission. Ces paramètres sont des variables aléatoires complexes, leurs modules ont une *PDF* suivant la distribution de Rayleigh ou de Rice selon l'existence ou pas de trajets directs *LOS*. La i^{eme} ligne de $\boldsymbol{H}$ représente les gains des trajets arrivant à la même antenne de réception *i,* tandis que la j^{eme} colonne représente les gains des trajets partant de la même antenne d'émission.

L'utilisation de la technique *MIMO* permet d'une part, un *multiplexage* spatial qui augmente le débit de transmission. De l'autre part, la technique *MIMO* offre la possibilité de bénéficier de la *diversité* spatio-temporelle qui permet de réduire le taux de l'erreur binaire « Bit Error Rate » *BER* et la probabilité de coupure « outage probability ». Donc, la diversité et le multiplexage sont les deux avantages majeurs de l'utilisation des techniques *MIMO*. L'apport de ces deux opérations est évalué par le calcul du gain de diversité et du gain de multiplexage [2]-[5], [11], [24]-[28].

1.4.1 Diversité et gain de diversité

L'idée principale de la diversité est de fournir au système de détection, au niveau du récepteur, plusieurs répliques du même signal. Si ces répliques subissent des évanouissements indépendants, la probabilité que toutes ces répliques seront fortement atténuées sera alors très faible. Dans cette situation, le récepteur peut améliorer la détection par la sélection de la réplique ayant le plus grand *SNR* ou par la combinaison de toutes les répliques reçues. Cela se manifeste par la réduction de la probabilité de coupure et du *BER*. Pour évaluer quantitativement la diversité, on calcule le gain de diversité qui est donnée par :

$$G_d = \lim_{\gamma \to +\infty} \frac{\log(p_e)}{\log(\gamma)} \tag{1.9}$$

Où G_d est le gain de diversité, p_e est la valeur de la probabilité de l'erreur à un *SNR* égal à γ. L'équation (1.9) montre que le gain de diversité n'est autre que la pente de la courbe de la probabilité de l'erreur en fonction du *SNR* dans une échelle *log-log.*

Il y a deux défis relatifs à la notion de la diversité. Le premier concerne les méthodes de diversité « diversity methods », c'est-à-dire comment fournir au récepteur les différentes répliques sans augmenter excessivement la largeur de bande, la puissance du signal utile et la complexité logicielle ou matérielle du décodeur. Le second défi concerne les méthodes de combinaison « combining methods », c'est-à-dire comment utiliser et combiner ces répliques afin de minimiser au maximum la probabilité de l'erreur.

La diversité peut être temporelle, fréquentielle, angulaire de polarisation, ou spatiale :

- *La diversité temporelle* consiste à envoyer au moins deux répliques du même signal, séparé d'un intervalle de temps supérieur au temps de cohérence du canal [29]. Si les évanouissements du canal sont lents, le temps de cohérence sera élevé, donc le temps séparant deux répliques identiques sera important ce qui augmente le temps de décodage.

- Pour *la diversité fréquentielle,* on envoie au moins deux répliques du même signal sur des fréquences séparées d'au moins la bande de cohérence du canal [30]. Si le canal possède une bande de cohérence large, on doit utiliser une bande très large pour la transmission des répliques, ce qui dégrade énormément l'efficacité spectrale du système de communication.
- Pour *la diversité angulaire,* les différentes copies du même signal sont collectées de directions différentes en utilisant des antennes directives
- *La diversité de polarisation* utilise la polarisation horizontale pour envoyer une réplique du signal, et la polarisation verticale pour l'envoi de l'autre réplique. Ce type de diversité peut fournir uniquement une diversité d'ordre 2 et pas plus.
- *La diversité spatiale* consiste à utiliser plusieurs antennes pour l'émission et la réception des différentes répliques du même signal. Si les antennes sont éloignées l'une des autres d'une distance supérieure à la moitié de la longueur d'onde, les différentes répliques émises vont subir des évanouissements indépendants, ce qui permet d'obtenir un gain de diversité important [31]. Contrairement à la diversité temporelle et à la diversité fréquentielle, la diversité spatiale ne souffre pas de dégradation de l'efficacité spectrale.

Les techniques de combinaison des répliques sont généralement divisées en trois types, à savoir la méthode de combinaison à rapport maximal « Maximum Ratio Combining » *MRC,* la méthode de combinaison par sélection et la méthode de combinaison à gain égal « Equal Gain combining » *EGC*. La méthode *MRC* consiste à combiner toutes les répliques supposées du même signal afin de trouver le signal le plus semblable au signal émis. La méthode la plus connue de la technique *MRC* est la méthode du maximum de vraisemblance « Maximum Likelihood » *ML,* qui combine les répliques en calculant leur moyenne pondérée. En choisissant des coefficients de pondération optimaux, cette technique permet d'augmenter le *SNR* de la combinaison qui est la somme des *SNRs* de toutes les répliques individuelles [11]. La méthode de combinaison par sélection est utilisée lorsque les contraintes pratiques ne permettent pas l'utilisation de plusieurs chaines *RF* indispensables pour la méthode *MRC.* Cette méthode est basée sur la sélection de la réplique ayant le *SNR* le plus élevé, afin de

l'utiliser pour le décodage. La grande difficulté de cette technique est comment faire une sélection rapide du signal qui possède le plus haut *SNR*. En outre, cette méthode ne permet pas une amélioration importante du rapport *SNR* en comparaison avec la méthode *MRC*.

1.4.2 Multiplexage et gain du multiplexage

Dans le paragraphe précédent, nous avons vu qu'un système multi antennaires *MIMO* peut être utilisé pour améliorer la qualité de la communication à travers l'augmentation du gain de diversité. Un tel système peut être également utilisé pour augmenter la capacité du canal, et par conséquent augmenter le débit de transmission par l'utilisation du multiplexage spatial. En effet un système utilisant N_t antennes d'émission et N_r antennes de réception peut transmettre jusqu'à $min(N_t, N_r)$ symboles par période. Le gain du multiplexage est donné par l'équation :

$$G_m = \lim_{\gamma \to +\infty} \frac{r}{\log(\gamma)} \tag{1.10}$$

Où *r* dénote le débit du code utilisé pour le multiplexage.

On note que l'augmentation du gain de multiplexage du système *MIMO* entraîne la diminution du gain de diversité et vice-versa. Donc, il faut toujours chercher le meilleur compromis entre le gain de multiplexage, qui est en relation avec le débit, et le gain de diversité, qui est en relation avec la probabilité de l'erreur. La théorie montre que pour un gain de multiplexage $G_m=i$ où $i=1,2\ldots\ min(N_t, N_r)$, le maximum du gain de diversité est donnée par $G_d=(N_t-i).(N_r-i)$ [32].

1.4.3 Les détecteurs pour les systèmes *MIMO*

Comme souligné par Claude Shannon « le problème fondamental de la communication est celui de la reproduction exacte ou approximative ; à un point donné ; du message sélectionné à un autre point » [33]. Pour la communication *MIMO,* on a de multiples symboles ou messages qui sont envoyés simultanément, puis contaminés par des interférences et des bruits *AWGN,* et qui doivent être

détectés et décodés au niveau du récepteur. Les symboles multiples peuvent être détectés séparément ou conjointement. La détection conjointe permet d'assurer les meilleures performances, mais en contrepartie elle augmente la complexité arithmétique du processus de détection.

Malheureusement, la complexité des détecteurs *MIMO* optimums augmente exponentiellement avec l'augmentation du nombre de variables de décision. Au sens générique le problème de la détection *MIMO* peut être défini pour un système à N_t entrées et N_r sorties dont la fonction de transfert est décrite par une matrice ayant des colonnes non-orthogonales, et dont les N_r sorties sont affectées par des bruits additifs de nature aléatoire et pas forcément Gaussienne. La relation entrée-sortie d'un tel système est donnée par l'équation (*1.7).* Le vecteur ***s*** représentant N_t entrées est aléatoirement sélectionné de l'ensemble A^{N_t} contenant des vecteurs à N_t composantes. Chaque composante est un élément de la constellation *A* qui contient *M* éléments appelés alphabet de la constellation. ***H*** est une matrice à N_r lignes et N_t colonnes, dont chaque élément est un nombre aléatoire réel ou complexe. ***r*** et ***n*** sont deux vecteurs à N_r composantes aléatoires réelles ou complexes.

On se basant sur le modèle de l'équation *(1.7),* la tâche principale de la détection *MIMO* est l'estimation du vecteur d'entrée (émis) ***s****,* en s'appuyant sur la connaissance du vecteur de sortie (reçu) ***r*** et de la matrice du canal ***H****,* comme illustrée dans la figure 1.4.

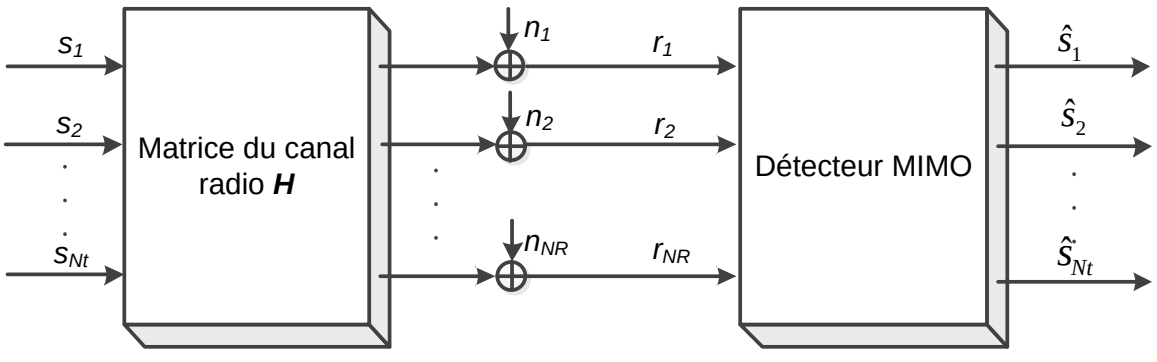

Figure 1.4 : Illustration du concept de la détection *MIMO*.

On note que la connaissance exacte de la sortie ***r*** est nécessaire, alors que la connaissance des paramètres statistiques de ***H*** est suffisante pour certains cas de détecteur. Si la valeur instantanée de ***H*** est connue par une estimation du canal, la détection est appelée *cohérente*. Par contre si l'estimation de ***H*** est évitée la détection fait partie de la famille des détections *non cohérentes*.

Les premiers travaux de recherche sur les détecteurs *MIMO* datent de la fin des années soixante. Depuis cette date de multiples détecteurs *MIMO* ont été proposé pour satisfaire les recommandations imposées par la multiplicité des applications des systèmes *MIMO*. Les détecteurs *MIMO* peuvent être classés selon plusieurs critères : optimal/sous-optimal, linéaire/non-linéaire, adaptative/non-adaptative, décision-hard/décision-soft… Ils sont très nombreux et loin d'être expliqués ou recensés dans cette section. On se limite donc à donner un aperçu des détecteurs *MIMO* les plus fréquents dans la littérature et qui sont présentés dans la figure 1.5 [34].

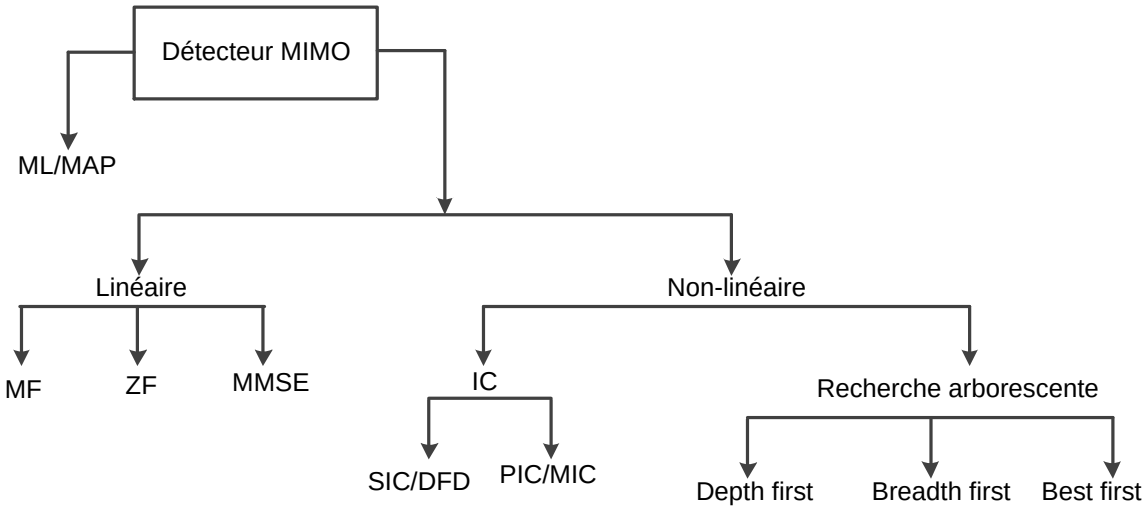

Figure 1.5 : Panorama des détecteurs *MIMO* les plus fréquents.

1.4.3.a. Détecteurs *MIMO* optimaux

Afin de faire la conception d'un détecteur optimal, il faut d'abord définir le critère de décision optimal. En communication, la minimisation de la probabilité de l'erreur est un critère d'intérêt primaire. Le critère de décision optimal qui minimise la probabilité de l'erreur en se basant uniquement sur les signaux observés et un ensemble d'hypothèses est appelé le critère du maximum a posteriori *MAP*. Pour la détection *MIMO*, ce critère est donné par :

$$D_{MAP} : \hat{\boldsymbol{s}} = \arg \max_{s \in A^{N_t}} \mathrm{p}(\boldsymbol{s/r}) \qquad (1.11)$$

Où p(x/y) dénote la probabilité conditionnelle de x sachant y.

Si les vecteurs ***s*** sont identiquement distribués dans l'ensemble A^{N_t}, le critère *MAP* de l'équation (1.11) est équivalant au critère du maximum de vraisemblance *ML* qu'on peut écrire sous la forme suivante :

$$D_{ML}: \hat{\boldsymbol{s}} = \arg \max_{s \in A^{N_t}} p(\boldsymbol{r/s}) \qquad (1.12).$$

Si le vecteur du bruit ***n*** est de type *AWGN,* le détecteur *ML* peut s'exprimer sous la forme :

$$D_{ML} : \hat{\boldsymbol{s}} = \arg \min_{s \in A^{N_t}} \left\| \boldsymbol{r} - \boldsymbol{H.s} \right\|^2 \qquad (1.13)$$

Le critère de l'équation (1.13) est appelé aussi critère de la distance Euclidienne minimale « Minimum Euclidean Distance » *MED.* L'équation (1.13) peut être résolue par une recherche dans la totalité de l'ensemble A^{N_t}. Pour une constellation *M-aire* et un système *MIMO* à N_t antennes d'émission la complexité arithmétique du détecteur *ML* est de l'ordre de M^{Nt}, ce qui rend ce détecteur très difficile ou même impossible à réaliser en pratique pour de grandes valeurs de N_t et de *M.*

1.4.3.b. Détecteurs *MIMO* linéaires

Les détecteurs *MIMO* linéaires sont basés sur la transformation linéaire du vecteur reçu ***r*** selon l'équation suivante :

$$\boldsymbol{d} = \boldsymbol{T}.\boldsymbol{r} \tag{1.14}$$

Où ***T*** est la matrice de la transformation qui est appelée aussi matrice de filtrage, et ***d*** est le vecteur résultant de la transformation linéaire du vecteur reçu ***r.***
Les détecteurs *MIMO* linéaires sont connus généralement par leur complexité arithmétique réduite en comparaison avec les détecteurs *ML,* mais ils souffrent de dégradation de leurs performances.

Détecteur *MF*

Le détecteur à filtre adapté « Matched Filter » *MF* possède la complexité arithmétique la plus basse parmi tous les détecteurs *MIMO*, sa matrice de transformation est donnée par :

$$\boldsymbol{T}_{MF} = \boldsymbol{H}^{H} \tag{1.15}$$

Le vecteur ***d*** est donné par :

$$\boldsymbol{d}_{MF} = \boldsymbol{H}^{H}.\boldsymbol{H}.\boldsymbol{s} + \boldsymbol{H}^{H}.\boldsymbol{n} \tag{1.16}$$

Le détecteur *MF* est un filtre linéaire optimal qui permet de maximiser le *SNR* en présence d'un bruit aléatoire additif.

Détecteur *ZF*

Pour le détecteur de forçage à zéro « Zero Forcing » *ZF,* la matrice de transformation est donnée par :

$$\boldsymbol{T}_{ZF} = \boldsymbol{H}^{-1*} = (\boldsymbol{H}^{H}.\boldsymbol{H})^{-1}\boldsymbol{H}^{H} \tag{1.17}$$

Ou l'opérateur (*-1**) représente le pseudo inverse de la matrice, et qui devient un vrai inverse si ***H*** est une matrice carrée à rang plein. Le vecteur ***d*** est donnée par :

$$\boldsymbol{d}_{ZF} = \boldsymbol{s} + \left[(\boldsymbol{H}^{H}.\boldsymbol{H})^{-1}\boldsymbol{H}^{H}\right]\boldsymbol{n} \tag{1.18}$$

Le détecteur *ZF* permet d'éliminer les interférences, mais la puissance du bruit additif augmente.

Détecteur *MMSE*

Pour le détecteur à erreur quadratique moyenne minimale « Minimum Mean Square Error » *MMSE*, la matrice ***T*** est donnée par :

$$\boldsymbol{T}_{MMSE} = (\boldsymbol{H}^H.\boldsymbol{H} + 2.\sigma^2.\boldsymbol{I})^{-1}.\boldsymbol{H}^H \qquad (1.19)$$

Où σ^2 est la puissance du bruit, et ***I*** est la matrice identité.

Le détecteur *MMSE* permet d'atteindre un meilleur équilibre entre l'élimination des interférences et la réduction du bruit additif en comparaison avec le détecteur *ZF,* car il permet de réduire la puissance totale des interférences et du bruit additif. Pour cette raison, le détecteur *MMSE* possède de meilleures performances que le détecteur *ZF* pour les valeurs faibles du *SNR*.

1.4.3.c Détecteurs *MIMO* à annulation d'interférences

Les détecteurs à annulation d'interférences « Interférences Concellation » *IC* forment une classe importante des détecteurs sous optimaux et non-linéaire qui permettent d'achever de meilleures performances que les détecteurs linéaires et une complexité réduite par rapport aux détecteurs optimaux. L'inconvénient de ce type de détecteur est qu'ils souffrent de la propagation de l'erreur. Cette classe de détecteurs possède plusieurs variantes comprenant :

Les détecteurs *SIC*

Pour le détecteur à annulation successive d'interférence « Successive Interference Cancellation » *SIC*, un seul symbole s_i est détecté à la fois, puis l'interférence qu'ajoute ce symbole détecté aux autres symboles à détecter est éliminée. Dans ce contexte, il est bénéfique d'annuler l'effet des signaux les plus puissants avant de détecter les signaux de faible puissance. Cela est réalisé par l'ordonnancement des signaux reçus avant de passer à la détection pour chaque itération, le détecteur ainsi

obtenu est appelé *OSIC* « Ordred Successive Interference Cancellation ». Le détecteur *SIC* possède de bonnes performances lorsque les signaux reçus simultanément présentent des différences de puissance considérables.

Les détecteurs *PIC*

Pour le détecteur à annulation parallèle d'interférences « Parallel Interferences Cancellation » *PIC,* tous les symboles sont détectés simultanément. Ce détecteur possède un temps de traitement très réduit en comparaison avec le détecteur *SIC*. En plus, et contrairement au détecteur *SIC*, il est plus adapté aux cas des signaux à puissance similaire.

Les détecteurs *MIC*

Pour le détecteur à suppression d'interférence multi étages « Multistage Interference Cancellation » *MIC,* le premier étage peut être n'importe quel détecteur sous-optimal. Les différents étages fonctionnent en cascade, ainsi les décisions de l'étage *i-1* pour tous les symboles sont utilisées comme données par l'étage *i* pour faire la suppression des interférences.

Les détecteurs *DFD*

Le concept de base du détecteur à retour de décision « Decision-Feedback Detector » *DFD* est le même que celui des égalisateurs à retour de décision. La décision faite durant une itération *i* est retournée vers l'entrée du décodeur afin de l'utiliser pour la suppression des interférences durant l'itération *i+1*.

1.4.3.d. Détecteurs *MIMO* basés sur la recherche arborescente :

Les travaux de Viterbo *et al* [35], [36] ont largement stimulé la recherche dans le domaine de la détection *MIMO* basée sur la recherche arborescente. Viterbo *et al* ont appliqué l'algorithme « Depth-first sphere decoding » à la détection *ML* pour une constellation multidimensionnelle. Le principe du « Sphere Decoding » *SD* est de réduire l'espace de recherche à une hyper sphère de rayon *R* construit autour du

vecteur reçu ***r,*** au lieu de faire la recherche sur la totalité des M^{Nt} vecteurs possibles comme illustré dans la figure 1.6.

La méthode *SD* consiste à présenter le problème de détection sous forme d'un arbre, où les branches des arbres représentent les parties réelles et les parties imaginaires des symboles transmis. Au niveau de chaque nœud de l'arbre, on doit vérifier si le symbole à tester fait partie des solutions envisageables ou non. Si oui, on descend dans les branches de ce nœud, sinon on élimine le nœud et on passe à un autre.

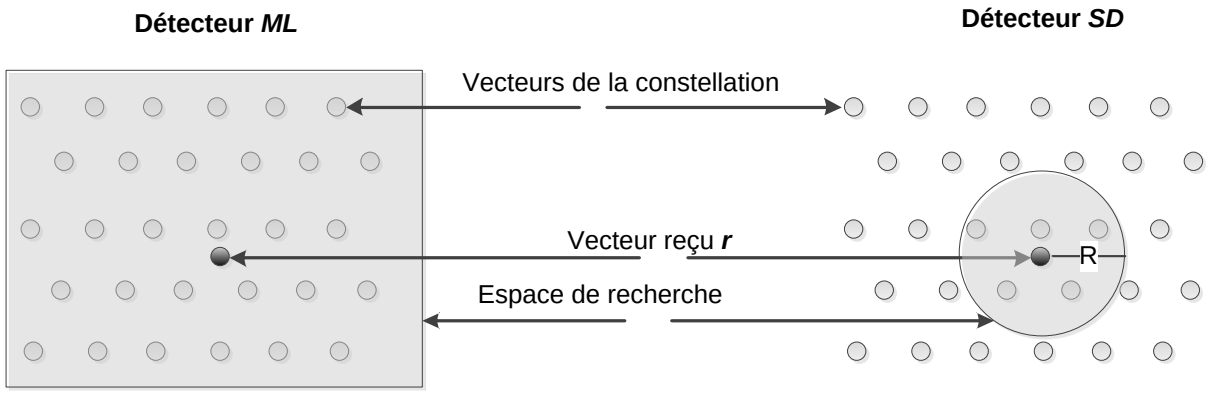

Figure 1.6 : Principe de la détection à base de de l'algorithme *SD.*

Le choix du rayon de la sphère est un paramètre crucial pour le détecteur *SD*. En effet, un grand rayon permet d'obtenir de meilleures performances, mais cela augmente le nombre de candidats à tester, ce qui augmente la complexité arithmétique. En revanche, la réduction du rayon permet de réduire la complexité arithmétique, mais au détriment de la dégradation des performances. Il existe plusieurs variantes de la méthode de recherche arborescente, qui peuvent être divisées en trois catégories qui sont « depth-first », « breath-first » et « best-first ». L'avantage majeur de ce type de détecteurs est qu'ils peuvent achever des performances très proches, ou même égales, à celles du détecteur optimal *ML,* mais avec une complexité de calcul réduite surtout pour le cas de fort *SNR* et de petit nombre d'antennes d'émission N_t [34].

1.5 Modulation à porteuses multiple de type *OFDM*

Les premiers travaux sur la modulation à porteuses multiples datent des années 1950 [24], [25] où un modem *HF,* qui permet l'émission simultanée sur plusieurs porteuses de fréquences différentes modulées à bas débit, a été proposé. Cette technique est nommée multiplexage fréquentiel « Frequency Devision Multiplex » *FDM*. Cette technique avait deux inconvénients majeurs. Le premier inconvénient est la complexité de réalisation de l'émetteur et surtout du récepteur. Une série d'égalisateurs et de filtres adaptés est nécessaire pour la séparation des différents signaux reçus simultanément. Le deuxième inconvénient est l'occupation spectrale importante, puisque les différentes composantes doivent occuper des bandes de fréquence qui ne se chevauchent pas afin d'assurer la séparation au niveau de la réception, ce qui diminue considérablement l'efficacité spectrale.

Pour résoudre le problème de l'occupation spectrale, Chang, R. W. [39] a proposé en 1966 l'utilisation de porteuses orthogonales offrant la possibilité d'exploiter le recouvrement spectral afin de réduire la bande de fréquence occupée, et par conséquent améliorer l'occupation spectrale. Cette importante contribution a donné naissance à la technique qui sera nommée par la suite *OFDM* « Orthogonal Frequency Division Multiplexing ». Chang, R. W a concrétisé sa proposition par la proposition d'un modem *OFDM* [6]. En 1971, Weinstein S. et Ebert P. [40] ont montré que le signal modulé en utilisant la technique *OFDM* peut être exprimé par la transformée de Fourier discrète "Discret Fourier Transform" *DFT*. Cela permet d'une part, de remplacer toute une série de filtres à l'émission et à la réception par un opérateur *IDFT* « Inverse *DFT* » à l'émission et un opérateur *DFT* à la réception. De l'autre part, l'utilisation de l'algorithme de la transformée de Fourier rapide « Fast Fourier Transform » *FFT/IFFT* pour le calcul de la *DFT* et de la *IDFT,* a rendu possible l'implémentation de l'*OFDM* sur un circuit numérique. Peled A. et Ruiz A. [41] ont proposé une nouvelle version de l'*OFDM* plus robuste aux interférences, par l'insertion cyclique d'un intervalle de garde dans chaque symbole. La technique *OFDM* a été retenue comme modulation standard pour la diffusion de l'audio numérique « Digital Audio Broadcasting » *DAB* en 1994 [42], puis elle est généralisée

pour plusieurs standards et applications comme le *DVB-T* « Digital Video Broadcasting- Terrestrial », *HIPERLAN/2, IEEE 802.11a*…

1.5.1 Principe de l'*OFDM*

La technique *OFDM* se repose sur trois concepts de base qui sont : le multiplexage fréquentiel *FDM* ou fréquences multiples, les fréquences orthogonales et l'insertion de l'intervalle de garde.

Le multiplexage fréquentiel consiste à diviser la séquence d'information haut débit à transmettre en N_p sous-séquences à bas débit. Chaque sous-séquence module une sous-porteuse comme illustré dans la figure 1.7.a. La division de la séquence à haut débit en sous-séquences à bas débit permet d'augmenter la durée des symboles qui va devenir supérieur à l'étalement des retards. Donc, la largeur de bande des sous-séquences devient inférieure à la bande de cohérence du canal. Ainsi, le canal sélectif en fréquence est transformé en plusieurs sous-canaux non sélectifs en fréquence (canal à évanouissement plat) ce qui permet de minimiser l'effet des *ISI*. La relation qui existe entre la durée T_d des symboles de la séquence originale et la durée T_s des symboles des sous-séquences est donnée par :

$$T_s = N_p.T_d \tag{1.20}$$

Où N_p est le nombre des sous-séquences à bas débit, qui est aussi le nombre des sous-porteuses.

L'orthogonalité est une technique qui permet la transmission de plusieurs signaux sur un canal commun, et leur détection sans interférences. La technique *OFDM* exploite l'orthogonalité en choisissant des sous-porteuses aussi proche que possible toute en maintenant l'orthogonalité entre elles (figure 1.7.b).

L'orthogonalité avec un recouvrement spectral optimal est assurée en choisissant les fréquences des sous-séquences selon l'expression suivante :

$$f_i = f_0 + \frac{i}{T_s} = f_0 + \frac{i}{N_p T_d}. \tag{1.21}$$

Où f_i dénote la fréquence de la sous-porteuse *i* avec *i=0,1,….N_p-1.*

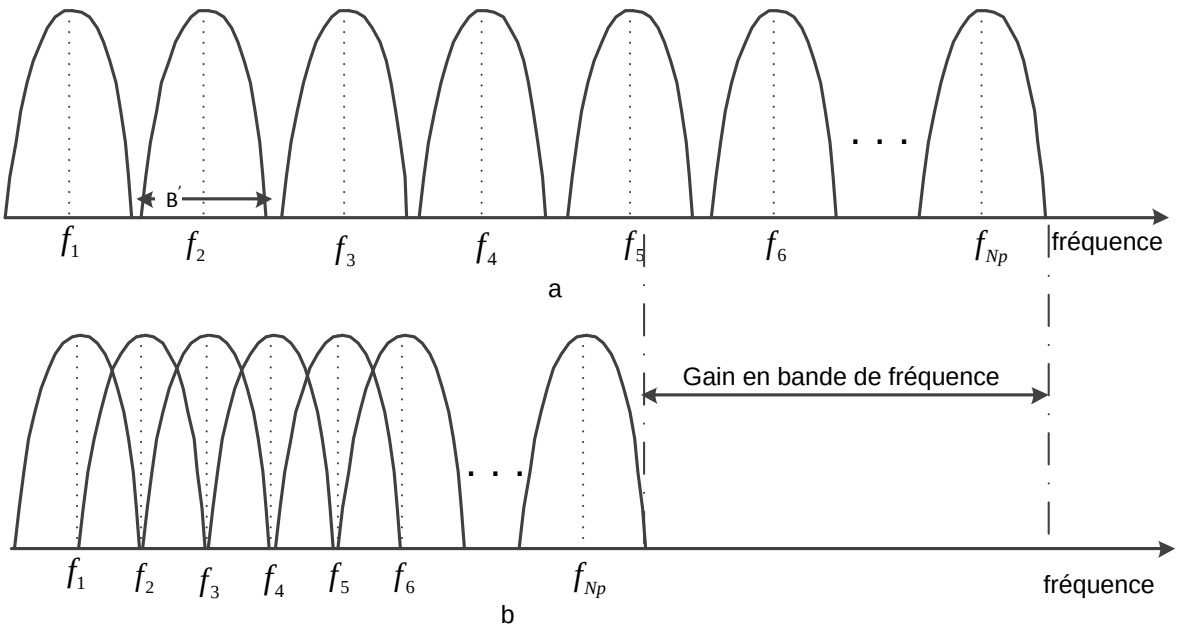

Figure 1.7 : Les bandes de fréquences occupées en multiplexage : (a) multiplexage fréquentiel conventionnel et (b) *OFDM*.

Le rapport entre la bande occupée par les sous-porteuses de l'*OFDM* et la bande occupée par les sous-porteuses du multiplexage fréquentiel conventionnel tend vers *0.5* lorsque le nombre N_p des sous-porteuses est important :

$$\lim_{N_p >> 1} \left(\frac{(N_p + 1)B'}{(2.N_p)B'} \right) = 0.5 \qquad (1.22)$$

Ou B' est la largeur de bande de chacune des sous séquences.

L'insertion de *l'intervalle de garde* est une technique qui permet de minimiser les interférences inter symboles *ISI* au niveau de la réception, et qui sont dus au fait que quelques sous-porteuses ne sont pas orthogonales au niveau de la réception à cause des multi trajets. La technique de l'insertion de l'intervalle de garde consiste à insérer

au début de chaque symbole *OFDM,* une partie de la fin du même symbole de durée T_g, comme le montre la figure 1.8.

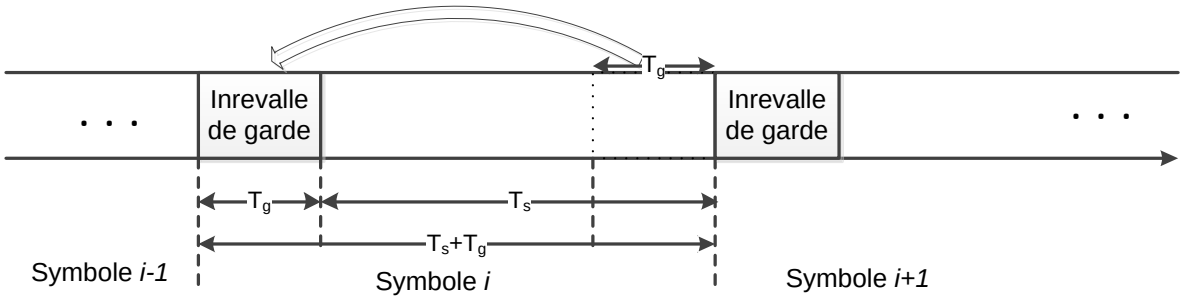

Figure 1.8 : Forme des symboles *OFDM* avec l'extension cyclique.

Le symbole ainsi obtenu, après l'insertion de l'intervalle de garde, est de durée T_s+ T_g. La durée de l'intervalle de garde doit être supérieure à l'étalement de la réponse impulsionnelle du canal. De cette façon, uniquement le début du symbole *OFDM* qui est la copie de sa fin est affecté par les *ISI,* et la partie utile du symbole *ODFM* sera conservée. A la réception, l'intervalle de garde est supprimé avant l'étape de la *FFT*, l'énergie contenue dans cette partie n'est plus donc exploitée au niveau de la réception, ce qui signifie que l'insertion de l'intervalle de garde conduit à une perte en efficacité spectrale et en puissance. La perte en puissance est de l'ordre de $10\log_{10}\left(T_s/(T_s+T_g)\right)$, alors que la perte en efficacité spectrale est de l'ordre de $T_g/(T_s+T_g)$.

Le schéma bloc d'un système de communication *SISO* à base de l'*OFDM* est présenté dans la figure 1.9. A l'émission, on trouve le bloc de calcul de la *DFT* par l'algorithme *FFT* suivi de la conversion parallèle série puis l'insertion de l'intervalle de garde. A la réception, on trouve les opérations inverses : suppression de l'intervalle de garde, conversion série parallèle et calcul de la *IDFT*.

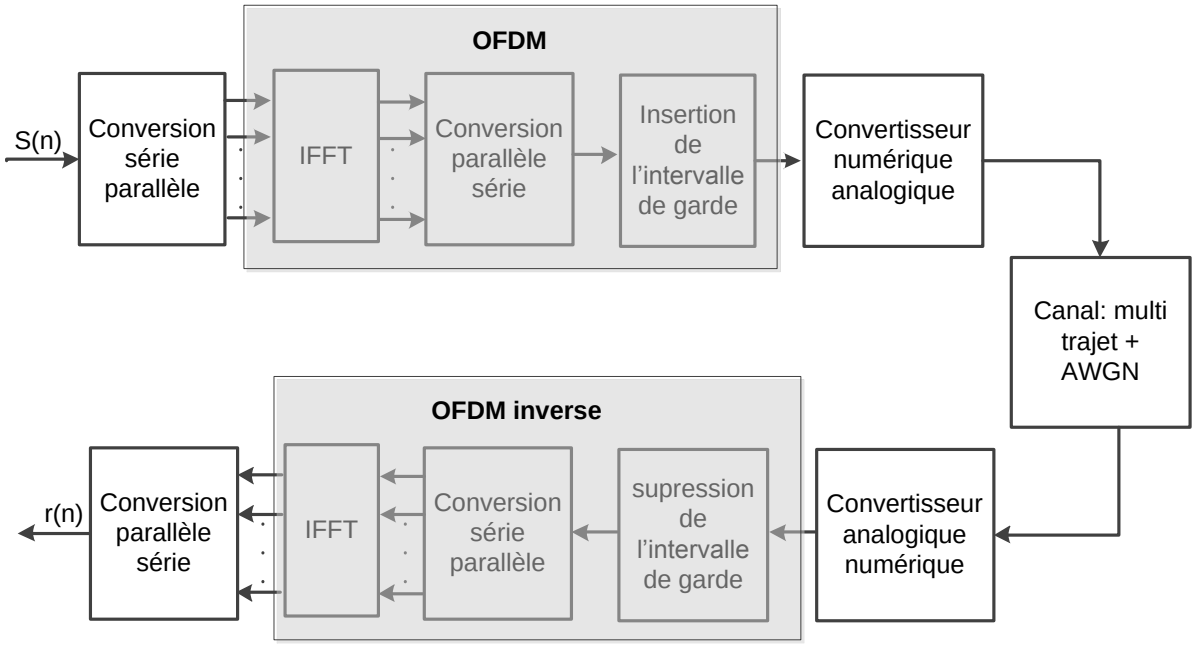

Figure 1.9 : Schéma bloc d'un système de transmission numérique *SISO* utilisant la technique *OFDM*.

1.5.2 Expression du signal *OFDM*

L'expression d'un signal résultant d'un multiplexage fréquentiel est donnée par :

$$s(t) = \sum_{i=-\infty}^{+\infty} \sum_{k=0}^{N_p-1} A_{i,\mathrm{k}} \Psi_{i,\mathrm{k}}(t) \tag{1.23}$$

Où $A_{i,\mathrm{k}}$ représente le symbole correspondant à la *i*^{ième} période qui est transmis sur la *k*^{ième} sous-porteuse, et $\Psi_{i,\mathrm{k}}(t)$ est une fonction qui fait partie d'une base orthogonale donnée par :

$$\Psi_{i,\mathrm{k}}(t) = g(t - iT_s).\exp(j.2\pi.f_k.t) \tag{1.24}$$

Où g(t) est une fonction définie sur l'intervalle *[0 T_s]*, qui est choisie pour satisfaire la condition de l'orthogonalité temporelle, et f_k *est la fréquence de la k*^{ième} sous porteuse dont l'expression est donnée par l'équation (1.21).

Pour assurer l'orthogonalité temporelle des fonctions $\Psi_{i,k}(t)$, Chang R. W. [39] a montré que le module et l'argument de la fonction *g(t)* doivent satisfaire certaines conditions. Une étude présentée par Le Floch, B. *et al* [7] a permis de donner une liste de fonctions *g(t)* satisfaisant la condition de l'orthogonalité temporelle. Parmi ces fonctions, on peut citer :

- Fenêtre rectangulaire : qui est donnée par :

$$g(t) = \Pi(t) = \begin{cases} 1 & \text{pour } t \in [0 \quad T_s] \\ 0 & \text{ailleurs} \end{cases} \tag{1.25}$$

  La fenêtre rectangulaire est la plus simple à mettre en œuvre, mais elle possède un spectre large.

- La fonction de Hanning : qui est donnée par l'expression suivante :

$$g(t) = \begin{cases} \frac{1}{2}\left(1 - \cos\left(\frac{2.\pi.t}{T_s - 1}\right)\right) & \text{pour } t \in [0 \quad T_s] \\ 0 & \text{ailleurs} \end{cases} \tag{1.26}$$

- La fonction *IOTA* « Isotropic Orthogonal Transform Algorithm » : cette fonction, qui est basée sur la théorie des ondelettes, permet la transmission par une onde qui assure une amélioration importante de l'efficacité spectrale toute en gardant de bonnes performances. En contrepartie, l'implémentation de la fonction *IOTA* sur un circuit numérique de type *ASIC* ou *FPGA* présente une complexité importante, puisqu'elle nécessite un nombre important de ressources arithmétiques et de ressources mémoires.

L'expression du signal *OFDM* durant l'intervalle *[0 T_s[*, pour une fenêtre de mise en forme rectangulaire est donnée par:

$$s(t) = \frac{1}{\sqrt{N_p}} \sum_{k=0}^{N_p - 1} \mathrm{Re}\left(A_k.\Pi(t).\exp\left(j.2.\pi.f_k.t\right)\right) \tag{1.27}$$

Si on choisit f_c comme la fréquence centrale de toutes les sous porteuse :

$$f_c = f_0 + \frac{N_p}{2.T_s} \tag{1.28}$$

L'expression de *s(t)* devient :

$$s(t) = \mathrm{Re}\left(\Pi(t).\exp\left(j.2.\pi.f_c.t\right).\underbrace{\sum_{k=0}^{N_p-1} \frac{A_k}{\sqrt{N_p}}.\exp\left(j.2.\pi.t.\left(k - N_p/2\right)/T_s\right)}_{X(t)} \right) \qquad (1.29)$$

D'où :

$$s(t) = \mathrm{Re}\left(\Pi(t).X(t).\exp\left(j.2.\pi.f_c.t\right)\right) \qquad (1.30)$$

L'échantillonnage du signal X(t) à la fréquence N_p/T_s permet d'écrire :

$$\begin{aligned} X(n.T_s/N_p) &= \sum_{k=0}^{N_p-1} \frac{A_k}{\sqrt{N_p}}.\exp\left(j.2.\pi.n.\left(k - N_p/2\right)/N_p\right) \\ &= (-1)^n \underbrace{\sum_{k=0}^{N_p-1} \frac{A_k}{\sqrt{N_p}}.\exp\left(j.2.\pi.n.k/N_p\right)}_{IDFT} \end{aligned} \qquad (1.31)$$

L'équation (1.31) montre que l'enveloppe complexe du signal *OFDM* peut être exprimée par l'*IDFT,* qui est efficacement implémenté sur des circuits numériques à l'aide de l'algorithme *IFFT*.

1.6. Etalement de spectre et *CDMA*

La technique de l'étalement de spectre « Spread spectrum technique » a été introduite au milieu des années cinquante [9]. Cette technique repose sur la relation de la capacité de canal développée en théorie d'information par Shannon C.E., et qui montre que : pour la transmission à travers un canal, avec une probabilité d'erreur négligeable, et avec une puissance du signal faible, il faut augmenter la bande de fréquence du signal. L'étalement de spectre a été destiné initialement aux applications militaires comme les tactiques d'anti-brouillage « anti-jamming », system de guidage…. Littéralement, un systeme à étalement de spectre est un système pour lequel le signal à transmettre est étalé sur une bande de fréquence beaucoup plus large que la bande de fréquence minimale requise pour sa transmission. La largeur

de bande du signal après l'étalement de spectre ne doit pas dépasser la largeur de la bande de cohérence du canal. L'étalement de spectre est assuré par un code qui est indépendant de l'information à transmettre. Avec l'évolution rapide des systèmes de communication mobile la technique de l'étalement de spectre a été adoptée par plusieurs standards comme : *IS95, UMTS, IEEE 802.11*…

La technique de l'étalement de spectre présente plusieurs avantages dont les principaux sont :

- Les systèmes utilisant l'étalement de spectre sont connus par leur *faible probabilité d'interception,* car ils sont caractérisés par leur faible densité spectrale en comparaison avec les systèmes sans étalement comme l'illustre la figure 1.10. Seuls les récepteurs possédant une copie synchrone du code d'étalement peuvent intercepter le signal étalé en fréquence.
- Les systèmes à étalement de spectre sont plus *robustes aux interférences à bande étroite,* puisque l'opération de désétalement au niveau de la réception, qui est identique à l'opération d'étalement, permet l'étalement du spectre des interférences, ce qui conduit donc à réduire leur puissance et à faciliter leur suppression.
- Le fait que la puissance du signal étalé est faible permet aux autres systèmes de télécommunications de *partager la même bande de fréquence,* vu que le spectre du signal étalé se confond avec celui du bruit, surtout pour les séquences pseudo aléatoires à débit élevé. C'est l'avantage le plus important de l'étalement de spectre dans le domaine des télécommunications cellulaires, car on peut mettre en œuvre une technique *d'accès multiple par répartition de code* « Code Division Multiple Access » *CDMA* [8], [10], afin de permettre à plusieurs utilisateurs d'émettre leurs signaux simultanément et sur la même bande de fréquence.

L'étalement de spectre présente deux inconvénients majeurs, à savoir la diminution de l'efficacité spectrale et l'augmentation de la complexité d'implémentation de l'émetteur et du récepteur, à cause de l'opération de l'étalement et du désétalement.

Il y a deux types de base de la technique *CDMA* [8], [33] qui sont la *CDMA* à séquence directe « Direct Sequence-CDMA » *DS-CDMA,* et la *CDMA* à saut de fréquence « Frequency Hopping-CDMA » *FH-CDMA*. La technique *DS-CDMA,* qui est la technique la plus utilisée, consiste à multiplier le signal utile, de largeur de bande $1/T_x$, par un autre signal appelé code d'étalement, de largeur de bande $1/T_c$, beaucoup plus large que $1/T_x$ (figure 1.10), où T_x et T_c sont, respectivement, la durée symbole du signal utile et du code.

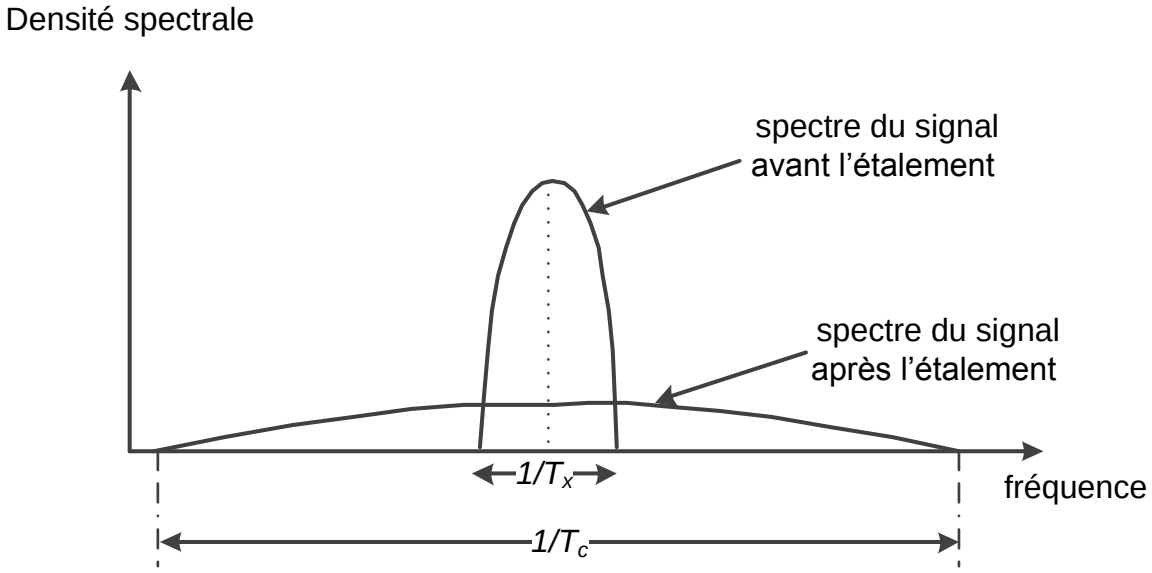

Figure 1.10 Densité spectrale du signal avant et après l'étalement *DS-CDMA*

La technique *FH-CDMA* est similaire à la technique *DS-CDMA,* où un code est utilisé pour l'étalement du signal utile. Cependant, au lieu d'étaler le signal sur une bande de fréquence continue, la bande de fréquence du signal étalé présente des sauts de fréquence.

Pour permettre la restitution des informations de chaque utilisateur d'une façon correcte au niveau de la réception, les signaux des différents utilisateurs doivent être décorrélés le plus que possible. Pour les canaux non-sélectifs en fréquence, les codes orthogonaux comme les codes *OVSF* « Orthogonal Variable Spreading

Factor » et les codes de Walsh-Hadamard permettent d'assurer les performances optimales. Par contre, pour les canaux sélectifs en fréquence, il faut utiliser d'autres types de code comme les codes de Gold,de Kasami et de Zado-Chu pour assurer de bonnes performances.

1.7. Combinaison de étalement de spectre et de la modulation multiporteuses

Le grand succès qu'a connu la technique d'étalement de spectre pour la deuxième génération de réseaux radio mobile, et qu'a connu la technique *OFDM* pour la diffusion numérique de la vidéo *DVB*, la diffusion numérique de l'audio *DAB,* et les réseaux *LAN* a motivé les chercheurs pour examiner la combinaison des deux techniques. La combinaison de la technique *CDMA* et de la technique *OFDM* a été proposée en 1993 [14], [43]-[45]. Il y a deux réalisations principales de la combinaison étalement de spectre-porteuses multiples « Spread spectrum-Multiple Carrier » SS-MC. La première réalisation est appelée *MC-CDMA* ou encore *OFDM-CDMA,* et la deuxième et appelée *MC-DS-CDMA*. Pour les deux schémas de réalisation, les différents utilisateurs partagent simultanément la même bande de fréquence, la séparation entre les données des différents utilisateurs est effectuée par le code d'étalement spécifique à chaque utilisateur. De plus, les deux réalisations utilisent la modulation à porteuses multiples pour réduire le débit des signaux résultants après l'étalement de spectre, afin de minimiser l'effet des *ISI*. Donc, en quelque sorte la modulation à porteuses multiples compense l'effet de l'étalement de spectre en termes de la largeur de la bande occupée.

La combinaison *MC-SS* trouve un large domaine d'application tel que : les communications radio mobiles cellulaires, où la technique *MC-CDMA* est adaptée à la liaison descendante, alors que la technique *MC-DS-CDMA* est la plus adaptée à la liaison montante, la diffusion *DVB-T* à lien de retour « *DVB-T* Return Link », les systèmes de distribution multipoints micro-onde/locale « Microwave/Local Multi-point Distribution Systems » MMDS/LMDS, communications aéronautiques…

1.7.1. Technique *MC-CDMA*

Pour la technique *MC-CDMA*, les symboles sont arrangés de façon à permettre l'émission des symboles d'un utilisateur simultanément à travers plusieurs sous-canaux, comme l'illustre la figure 1.11. La technique *MC-CDMA* est la technique la plus utilisée en comparaison avec la technique *MC-DS-CDMA*. Des études de comparaisons entre les deux techniques ont montré la supériorité de la technique *MC-CDMA* par rapport à la technique *MC-DS-CDMA* [46]-[48].

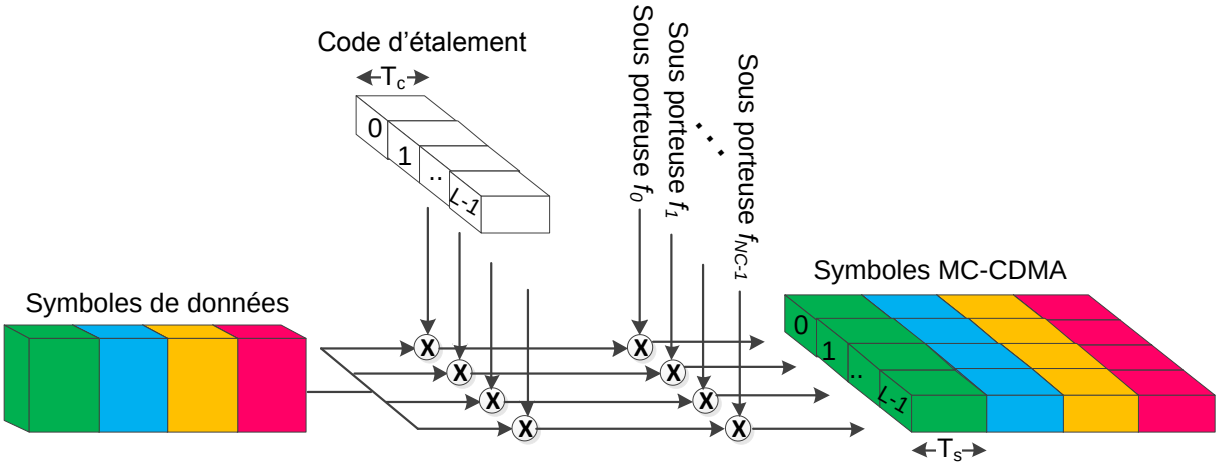

Figure 1.11 : Principe de génération de symboles *MC-CDMA* pour un seul utilisateur.

Comme le montre la figure 1.11, les données de chaque utilisateur sont étalées dans le domaine fréquentiel. Autrement dit, chaque symbole de données d_i, spécifique à l'utilisateur *i,* est d'abord multiplié par les chips $c_{i,k}$ du code propre à cet utilisateur avant d'être modulé en *OFDM*. Ainsi, chaque sous-porteuse transmet le produit du symbole de données par le chip propre à cette sous porteuse. Lorsque la longueur du code d'étalement L_c est égale au nombre des sous porteuses N_p, la durée T_s du symbole *MC-CDMA* sur chaque sous porteuse est :

$$T_s = T_c = T_d \tag{1.32}$$

L'espacement entre les sous-porteuses est donnée par :

$$\Delta f = \frac{1}{T_s} = \frac{1}{T_c} = \frac{1}{T_d} \tag{1.33}$$

L'enveloppe complexe X_i du signal *MC-CDMA* $s_i(t)$ *échantillonné à la fréquence* N_p/T_s, et qui correspond à l'utilisateur *i* est donné par :

$$X_i(n.T_s / N_p) = (-1)^n d_i . \underbrace{\sum_{k=0}^{N_p-1} \frac{c_{i,k}}{\sqrt{N_p}} . \exp\left(j.2.\pi.n.k / N_p\right)}_{IDFT} \tag{1.34}$$

L'équation (1.34) montre que l'enveloppe complexe du signal *MC-CDMA* peut être générée en effectuant l'*IDFT* sur les chips du code d'étalement. Cela mit en évidence l'influence du code d'étalement sur les variations de l'enveloppe complexe, d'où la nécessité d'un choix judicieux de ce code afin de limiter les variations de l'enveloppe complexe. La figure 1.12 illustre le principe de génération des symboles *MC-CDMA.*

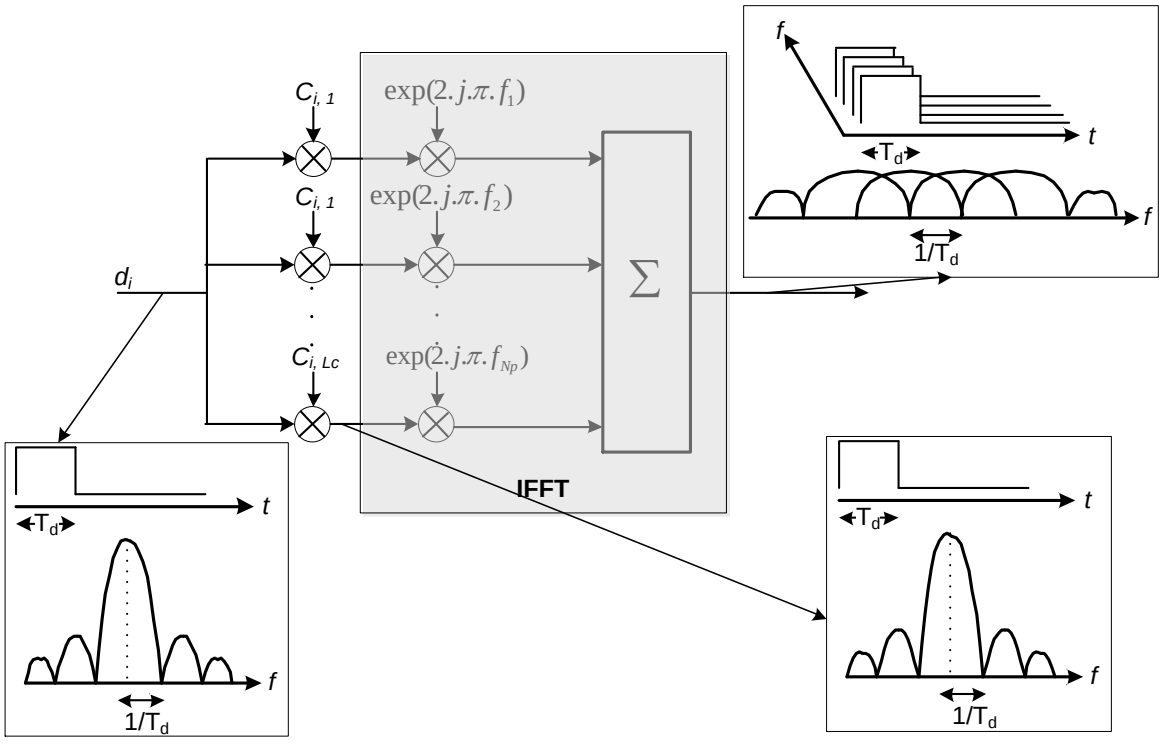

Figure 1.12. Structure du modulateur *MC-CDMA* du $i^{ième}$ utilisateur, avec nombre de sous porteuses égale à la longueur du code d'étalement.

1.7.2. Technique *MC-DS-CDMA* :

Cette technique a été proposée pour la première fois par Kondo S. et Milstein L. [45]. Le principe de la modulation *MC-DS-CDMA* est illustré dans la figure 1.13, pour lequel on considère que le nombre de sous porteuses N_p est égal à la longueur du code d'étalement L_c. La séquence de données d_i de l'utilisateur *i* ayant une durée symbole T_d est convertie en N_p sous-séquences parallèles, chacune ayant une durée symbole $N_p.T_d$. Ensuite, chacune des sous-séquences est étalée dans le domaine temporel par le code d'étalement $\boldsymbol{sc}_i$, approprié à l'utilisateur *i*. Finalement, chaque sous-séquence étalée est transmise à l'aide d'une sous-porteuse différente par rapport aux autres. Cette opération correspond exactement à un étalement de type séquence directe appliqué à chaque sous-canal.

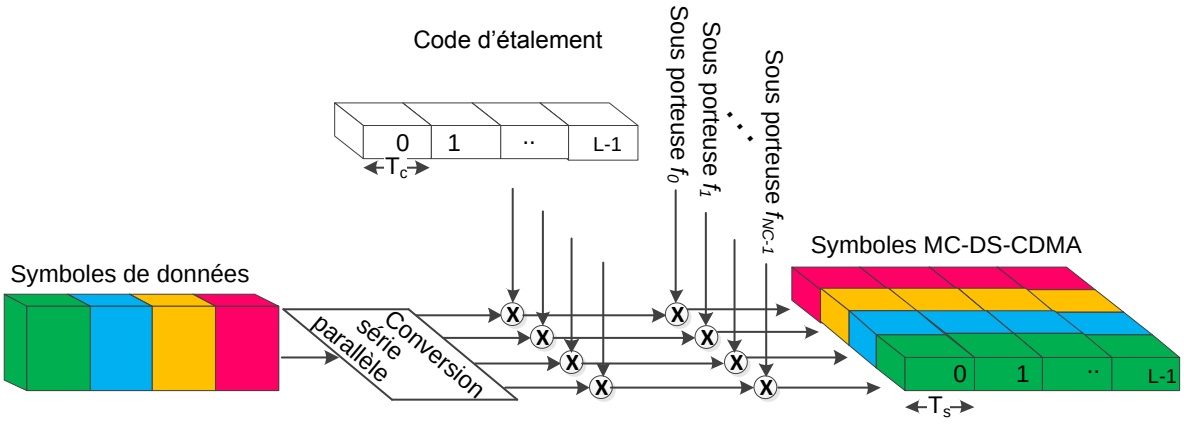

Figure 1.13 : Principe de génération de symboles *MC-DS-CDMA* pour un seul utilisateur.

La durée d'un symbole *MC-DS-CDMA* est donnée par :

$$T_s = N_P T_d = L_c T_c \tag{1.35}$$

L'espacement entre deux sous porteuses successives est donnée par :

$$\Delta f = \frac{1}{T_c} = \frac{L_c}{N_P T_d} \tag{1.36}$$

L'enveloppe complexe X_i du signal *MC-DS-CDMA* $s_i(t)$, échantillonné à la fréquence N_p/T_c, et qui correspond à l'utilisateur *i* est donné par :

$$X_i(n.T_c/N_p) = (-1)^n sc_i(n.T_c/N_p).\underbrace{\sum_{k=0}^{N_p-1} \frac{d_{i,k}}{\sqrt{N_p}}.\exp\left(j.2.\pi.n.k/N_p\right)}_{IDFT} \tag{1.37}$$

L'équation (1.37) montre que les symboles *MC-DS-CDMA* peuvent être générés par l'*IDFT* des symboles $d_{i,k}$ issus de la conversion série parallèle.

L'augmentation de la durée de symboles de T_d à $N_p.T_d$ permet de tirer profit de la diversité temporelle. Cependant, la technique *MC-DS-CDMA* ne présente pas de diversité fréquentielle puisque chaque sous-porteuse transmit un symbole différent par rapport aux autres sous-porteuses. La structure d'un modulateur *MC-DS-CDMA* est détaillée dans la figure 1.14.

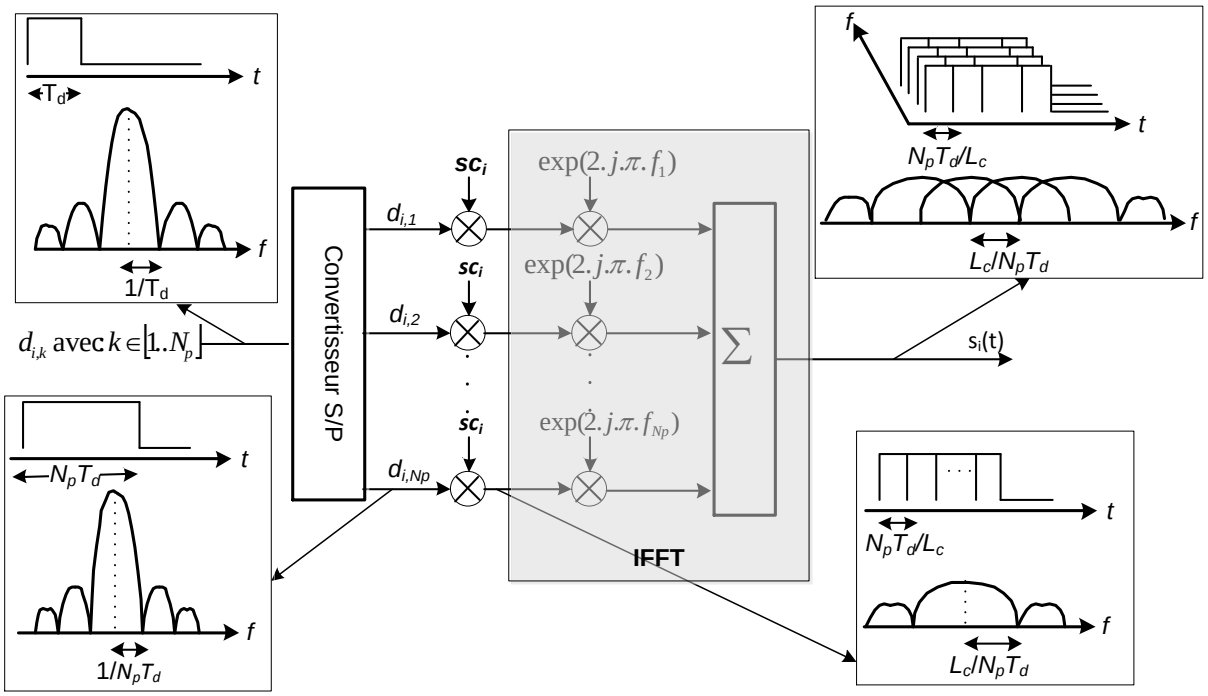

Figure 1.14. Structure du modulateur *MC-DS-CDMA* correspondant au $i^{ième}$ utilisateur.

Le tableau 1.1 montre les avantages et les inconvénients des deux techniques *MC-CDMA* et *MC-DS-CDMA* [8].

Tableau 1.1 : Comparaison entre *MC -CDMA* et *MC-DS-CDMA*

MC-CDMA		***MC-DS-CDMA***	
Avantages	Inconvénients	Avantages	Inconvénients
▪ Implémentation simple. ▪ Récepteur à complexité réduite. ▪ Efficacité spectrale élevée. ▪ Gain de diversité fréquentielle élevé puisque l'étalement est appliqué sur l'axe de fréquence.	▪ Rapport *PAPR* « peak to Average Power Ratio » élevé surtout pour les liaisons montantes « Uplink » ▪ Nécessité de la synchronisation.	▪ Rapport *PAPR* faible pour les liaisons montantes « Uplink ». ▪ Gain de diversité temporelle élevé puisque l'étalement est appliqué sur l'axe de temps.	▪ Possibilité d'apparition d'interférence *ISI/ICI* ce qui augmente la complexité du récepteur. ▪ Moins d'efficacité spectrale si autre technique que l'*OFDM* est utilisée.

1.8. Conclusion

Les canaux de communication radio mobile sont caractérisés par les multi-trajets, les bruits *AWGN,* et les mouvements relatifs entre l'émetteur et le récepteur, ce qui engendre des dispersions en temps et en fréquence. Ces dispersions déterminent la nature des canaux en termes de la sélectivité fréquentielle et temporelle. Les systèmes multi-antennaires *MIMO* servent à augmenter le débit de transmission par multiplexage spatial, et à augmenter la robustesse vis-à-vis des évanouissements du canal par l'utilisation du codage spatio-temporel. La modulation à porteuses multiples de type *OFDM* est une manière efficace pour transformer le canal sélectif en fréquence en plusieurs sous-canaux non-sélectifs en fréquences. Dans ce cadre, nous avons vu que l'orthogonalité des sous-porteuses minimise l'occupation

spectrale, et que l'insertion de l'intervalle de garde minimise les *ISI.* La mise en œuvre de l'*OFDM* est concrétisée par l'utilisation des algorithmes *FFT*. La technique *CDMA* utilise la diversité de code pour rendre possible le partage de la même bande de fréquence par plusieurs utilisateurs. La combinaison de la technique de l'étalement de spectre avec la modulation à porteuses multiples permet de tirer profit des avantages des deux techniques. Les combinaisons les plus connues sont le *MC-CDMA* et le *MC-DS-CDMA.*

Chapitre 2

CODAGE ESPACE-TEMPS : ETUDE THEORIQUE ET IMPLEMENTATION SUR *FPGA*

2.1 Introduction

Ce chapitre est consacré à l'étude théorique du codage espace-temps, et à la présentation des résultats d'implémentation d'un décodeur espace-temps d'Alamouti sur un circuit reconfigurable de type *FPGA*. Dans la première partie, nous présentons les deux types de codes spatio-temporels, qui sont les codes en blocs et les codes en treillis. Nous évoquons aussi les codes en blocs orthogonaux qui constituent une classe importante des codes en bloc. Dans la deuxième partie de ce chapitre, nous abordons la combinaison du codage espace-temps avec la modulation *OFDM* et la technique *CDMA*. Finalement, nous présentons dans la dernière partie deux architectures pour l'implémentation du décodeur espace-temps d'Alamouti sur un circuit *FPGA*.

2.2. Techniques de codage espace-temps

Le codage spatio-temporel, ou codage espace-temps « Space Time Coding » *STC* [11], [12] est une application très importante des systèmes multi antennaires de type *MISO* ou de type *MIMO.* Le *STC* consiste *à* exploiter la diversité spatiale par l'introduction de la redondance afin de lutter contre les évanouissements des canaux radiomobiles. Depuis l'apparition du principe de codage spatio-temporel, une multitude de techniques *STC* ont vu le jour. Ces techniques peuvent être classées en deux catégories principales, qui sont le codage espace-temps en bloc « Space Time

Bloc Coding » *STBC* et le codage espace-temps en treillis « Space Time Treillis Coding » *STTC*.

2.2.1 Codage espace-temps en bloc

Le codage espace-temps en bloc (*STBC)* est une technique qui consiste à transformer chaque bloc de données, de longueur déterminée, en matrice qui représente le bloc de données codées en espace et en temps. Le principe du codage et du décodage *STBC* est illustré dans la figure 2.1.

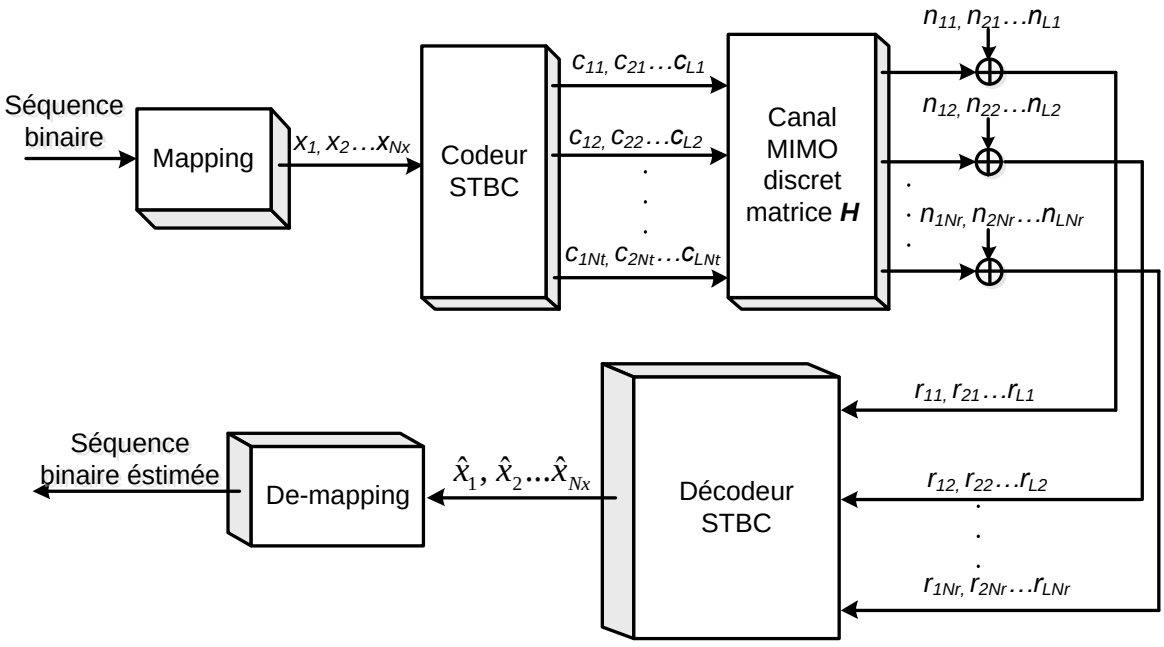

Figure 2.1 : Principe de codage et de décodage *STBC*.

La procédure de codage et de décodage *STBC* peut être décrite par les étapes suivantes :

- Une séquence binaire est d'abord mappée en séquence de symboles complexes ou réels selon la modulation numérique choisie (*QAM, PSK, ASK….).*

- Le codeur *STBC* prend un bloc de N_x symboles pour former une matrice de $N_t \times L$ codes répartis sur N_t antennes et L intervalles de temps selon la structure suivante :

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12} & . & . & . & c_{1N_t} \\ c_{21} & c_{22} & . & . & . & c_{2N_t} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ c_{L1} & c_{L2} & . & . & . & c_{LN_t} \end{bmatrix} \tag{2.1}$$

Les éléments c_{ij} de la matrice de code sont généralement des combinaisons linéaires de plusieurs symboles. Ces éléments sont choisis d'une façon à permettre au code de satisfaire certaines conditions et certains critères. La matrice qui exprime les éléments c_{ij} en fonction des symboles x_i est appelée matrice génératrice du code ***G***. Un élément c_{ij} représente le code à transmettre au $i^{ième}$ intervalle de temps sur la $j^{ième}$ antenne d'émission. Une ligne de la matrice ***C*** représente donc l'ensemble des symboles à transmettre sur toutes les antennes d'émission pendant une seule période, alors qu'une colonne représente les symboles à envoyer sur une seule antenne pendant L périodes. Ainsi, le système à base de codage *STBC* utilise N_t antennes d'émission pendant L périodes de temps pour transmettre N_x symbole. Le gain du codage est donc :

$$G_c = \frac{N_x}{L} \leq 1 \tag{2.2}$$

- Les différents codes sont passés à travers un canal *MIMO* discret qui est représenté par une matrice de canal ayant la forme :

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & . & . & . & h_{1Nr} \\ h_{21} & h_{22} & . & . & . & h_{2N_r} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ h_{Nt1} & h_{Nt2} & . & . & . & h_{NtN_r} \end{bmatrix} \tag{2.3}$$

L'élément h_{ij} représente le gain complexe entre la $i^{ième}$ antenne d'émission et la $j^{ième}$ antenne de réception.

- Après l'ajout du bruit *AWGN*, le décodeur *STBC* reçoit pendant *L* périodes un ensemble de signaux représentés par une matrice de $N_r \times L$ éléments dont la structure est :

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & . & . & . & r_{1N_r} \\ r_{21} & r_{22} & . & . & . & r_{2N_r} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ r_{L1} & r_{L2} & . & . & . & r_{LN_r} \end{bmatrix} \tag{2.4}$$

Un élément r_{ij} représente le signal reçu sur la $j^{ième}$ antenne de réception durant le $i^{ième}$ intervalle de temps. La matrice ***R*** est donnée par l'équation matricielle suivante :

$$\boldsymbol{R} = \boldsymbol{C.H} + \boldsymbol{N} \tag{2.5}$$

Où ***N*** est la matrice du bruit *AWGN* présent sur les N_r récepteurs pendant *L* intervalles de temps, dont la structure est donnée par :

$$\boldsymbol{N} = \begin{bmatrix} n_{11} & n_{12} & . & . & . & n_{1N_r} \\ n_{21} & n_{22} & . & . & . & n_{2N_r} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ n_{L1} & n_{L2} & . & . & . & n_{LN_r} \end{bmatrix} \tag{2.6}$$

- Le décodeur utilise la matrice des signaux reçus ***R*** et les informations sur le canal, afin de calculer les N_x symboles estimés $\hat{x}_1, \hat{x}_2 \ldots \hat{x}_{Nx}$, selon une méthode d'estimation déterminée.
- Finalement, l'opération de dé-mapping permet de convertir les symboles estimés en séquence de bits.

Afin de permettre l'évaluation d'un code *STBC,* on compare ses caractéristiques par rapport à un ensemble de caractéristiques qu'on appelle caractéristiques souhaitées ou caractéristiques de bons codes, dont les principales sont :

1. Une faible complexité de décodage avec un minimum d'antennes de réception. Pour le décodage *ML,* la simplicité est assurée en choisissant des matrices

génératrices ayant des colonnes orthogonales indépendamment de la constellation utilisée.

2. Diversité d'émission maximale pour N_t antennes d'émission. L'orthogonalité des colonnes de la matrice génératrice du code permet d'assurer la condition de la diversité d'émission maximale.
3. Un rendement égal à 1, ce qui signifie que le nombre de symboles à l'entrée du codeur N_x est égal au nombre de périodes d'envoi du code *L,* appelé aussi longueur de la trame du code.
4. Afin de minimiser le retard de décodage, la longueur de la trame doit être la plus petite possible. Puisque la longueur *L* doit être supérieure ou égale à N_t afin d'avoir un code linéairement décodable, la solution idéale est donc d'avoir une matrice génératrice carrée.
5. La puissance transmise doit être la même en temps et en espace. Autrement dit, il faut éviter les zéros dans la matrice génératrice.

2.2.2. Les codes *STBC* orthogonaux

Une famille très intéressante des codes *STBC* est celle des codes *STBC* orthogonaux « Orthogonal Space Time Bloc Code » *OSTBC*. Leur importance vient de l'orthogonalité des colonnes de leurs matrices génératrices. Cette orthogonalité permet de réduire la complexité du décodeur *ML* de l'ordre exponentiel à un ordre linéaire. Cela conduit à réduire significativement la complexité arithmétique et matérielle de décodeur *OSTBC,* surtout lorsque le nombre d'antennes de réception est élevé, et lorsque l'alphabet de la constellation est large. Les codes *OSTBC* peuvent être classés en construction orthogonale réelle, pour les constellations réelles, et en construction orthogonale complexe, pour les constellations complexes.

2.2.2.a Construction orthogonale réelle

Une étude détaillée sur la construction orthogonale réelle a été fournie par Radon et Hurwitz [49], [50]. Pour $N_t=2$, un exemple de matrice génératrice 2×2 est donnée par :

$$\boldsymbol{G_2} = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix} \tag{2.7}$$

Pour $N_t=4$, un exemple de matrice génératrice 4×4 obtenue est :

$$\boldsymbol{G_4} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \end{bmatrix} \tag{2.8}$$

On note que les codes obtenus par les deux matrices génératrices $\boldsymbol{G_2}$ et $\boldsymbol{G_4}$ possèdent une diversité pleine, c'est-à-dire une diversité d'ordre $N_t \times N_r$. En outre, le rendement de code est égal à 1. Notant aussi que la construction orthogonale n'est pas unique pour N_t donné.

2.2.2.b Construction orthogonale complexe

Une construction orthogonale complexe de taille N_t est une matrice carrée de $N_t \times N_t$ éléments complexes, dont les entrées sont : les symboles $x_1, x_2, \ldots x_{Nt}$, leurs négatifs $-x_1, -x_2, \ldots -x_{Nt}$, leurs conjugués $x_1^*, x_2^*, \ldots x_{Nt}^*$, les négatifs de leurs conjugués $-x_1^*, -x_2^*, \ldots -x_{Nt}^*$ ainsi que toutes les variables précédentes multipliées par l'imaginaire pur j. La matrice Génératrice de code $\boldsymbol{G}_{N_t}$ doit satisfaire la condition de l'équation :

$$\boldsymbol{G}_{N_t}^H . \boldsymbol{G}_{N_t} = \sum_{i=1}^{N_t} |x_i|^2 . \boldsymbol{I}_{N_t} \tag{2.9}$$

Avec $\boldsymbol{I}_{Nt}$ est la matrice identité de $N_t \times N_t$ éléments.

La construction orthogonale complexe la plus simple à implémenter est celle obtenue pour $N_t=2$, et qui est connue sous le nom du code d'Alamouti [51].

2.2.2.c Codage/décodage *OSTBC* d'Alamouti

Le codeur espace-temps en bloc d'Alamouti est le premier codeur proposé qui possède une diversité pleine pour un système à deux antennes d'émission $N_t=2$. Sa matrice génératrice de code est donnée par :

$$\boldsymbol{G}_2 = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} \tag{2.10}$$

Durant le premier intervalle de temps, le système à base du codeur d'Alamouti envoie le symbole x_1 sur la première antenne d'émission et le symbole x_2 sur la deuxième antenne d'émission. Pour le deuxième intervalle de temps, $-x_2^*$ est envoyé sur la première antenne alors que x_1^* est envoyé sur la deuxième antenne. Le codeur d'Alamouti permet l'envoi de deux symboles durant deux périodes, donc c'est un codeur à rendement unitaire. Il est clair aussi que le codeur d'Alamouti fait partie de la famille *OSTBC* puisque les deux colonnes de sa matrice génératrice sont orthogonales, et sa matrice génératrice satisfait la condition de la construction orthogonale complexe de l'équation (2.9). Au niveau de la réception, le système à base du codeur d'Alamouti possède deux configurations possibles. La première configuration consiste à utiliser une seule antenne de réception (N_r*=1),* on parle alors d'un système d'Alamouti *2×1*. La deuxième configuration utilise deux antennes de réception, et on parle donc d'un système d'Alamouti *2×2.*

Système d'Alamouti *2×1*

Le schéma de principe du codeur/décodeur d'Alamouti *2×1* est donné dans la figure 2.2.

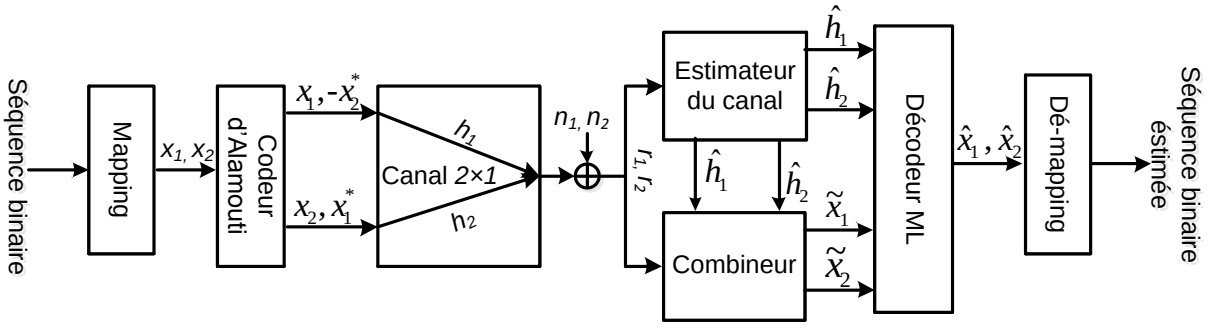

Figure 2.2 : Schéma de principe du codeur/décodeur d'Alamouti 2×1.

Pour un tel système, la matrice du canal ***H*** possède uniquement deux éléments : h_1 qui est le gain de trajet entre la première antenne d'émission et l'unique antenne de réception, et h_2 qui représente le gain de trajet entre la deuxième antenne d'émission et l'antenne de réception.

Pour un canal quasi statique on considère que les deux gains restent constants pendant les deux périodes d'envoi, c'est-à-dire :

$$\begin{aligned} &h_1(t) = h_1(t+T) \\ &\quad\text{et} \\ &h_2(t) = h_2(t+T) \end{aligned} \tag{2.11}$$

Où T représente la durée d'un seul symbole.

Si on considère la représentation discrète du canal, le vecteur des signaux reçu à l'entrée du décodeur d'Alamouti est obtenu en substituant (2.10) et (2.11) dans l'équation (2.5) :

$$r_1 = h_1.x_1 + h_2.x_2 + n_1 \tag{2.12}$$

$$r_2 = -h_1.x_2^* + h_2.x_1^* + n_2 \tag{2.13}$$

Avec r_1 est le signal reçu durant la première période, r_2 est le signal reçu durant la deuxième période, n_1 et n_2 sont les bruits *AWGN* de la première et la deuxième période respectivement.

Le récepteur estime les coefficients du canal pour les utiliser comme information de l'état du canal « Channel State Information » *CSI*. En supposant que les symboles de la constellation sont équiprobables, le détecteur *ML* est utilisé pour choisir, parmi toutes les valeurs possibles de x_1 et x_2, la paire des signaux $(\hat{x}_1, \hat{x}_2)$ qui permet de minimiser la distance :

$$\begin{aligned} &d^2(r_1, h_1.\hat{x}_1 + h_2.\hat{x}_2) + d^2(r_2, -h_1.\hat{x}_2^* + h_2.\hat{x}_1^*) \\ &= \left|r_1 - h_1.\hat{x}_1 - h_2.\hat{x}_2\right|^2 + \left|r_2 + h_1.\hat{x}_2^* - h_2.\hat{x}_1^*\right|^2 \end{aligned} \tag{2.14}$$

En substituant (2.12) et (2.13) dans l'équation (2.14), le décodeur *ML* peut être exprimé sous la forme :

$$(\hat{x}_1, \hat{x}_2) = \arg \min_{(\hat{x}_1, \hat{x}_2) \in C} \left(\left(|h_1|^2 + |h_2|^2 - 1 \right) \left(|\hat{x}_1|^2 + |\hat{x}_2|^2 \right) + d^2(\hat{x}_1, \tilde{x}_1) + d^2(\hat{x}_2, \tilde{x}_2) \right) \quad (2.15)$$

Avec, *C* est l'ensemble de toutes les paires $(\hat{x}_1, \hat{x}_2)$ possibles, $\tilde{x}_1$ et $\tilde{x}_2$ sont deux paramètres de décisions données par :

$$\tilde{x}_1 = h_1^* . r_1 + h_2 . r_2^* \quad (2.16)$$

$$\tilde{x}_2 = h_2^* . r_1 - h_1 . r_2^* \quad (2.17)$$

En remplaçant r_1 et r_2 par leurs expressions données par les deux équations (2.12) et (2.13) respectivement, on obtient :

$$\tilde{x}_1 = \left(|h_1|^2 + |h_2|^2 \right) . x_1 + h_1^* . n_1 + h_2 . n_2^* \quad (2.18)$$

$$\tilde{x}_2 = \left(|h_1|^2 + |h_2|^2 \right) . x_2 - h_1 n_2^* + h_2^* . n_1 \quad (2.19)$$

Les deux équations (2.18) et (2.19) montrent que le paramètre de décision $\tilde{x}_1$ est indépendant du symbole x_2 et $\tilde{x}_2$ est indépendant de x_1. Donc, l'expression du décodeur *ML* de l'équation (2.15) peut être réécrite d'une façon séparée comme suit :

$$\hat{x}_1 = \arg \min_{\hat{x}_1 \in S} \left(\left(|h_1|^2 + |h_2|^2 - 1 \right) |\hat{x}_1|^2 + d^2(\hat{x}_1, \tilde{x}_1) \right) \quad (2.20)$$

$$\hat{x}_2 = \arg \min_{\hat{x}_2 \in S} \left(\left(|h_1|^2 + |h_2|^2 - 1 \right) |\hat{x}_2|^2 + d^2(\hat{x}_2, \tilde{x}_2) \right) \quad (2.21)$$

Où *S* représente l'ensemble des symboles de la constellation.

En comparant les équations (2.20) et (2.21) avec l'équation (2.15), on remarque qu'il y a une réduction importante du nombre d'opérations de recherche du minimum puisque pour (2.20) et (2.21) la recherche est effectuée sur l'ensemble des symboles au lieu de l'ensemble de paires de symboles pour l'équation (2.15).

Pour les constellations de type *M-PSK,* la valeur de $\left(|h_1|^2 + |h_2|^2 - 1 \right) |\hat{x}_i|^2$ est constante indépendamment de $\hat{x}_i$, ce qui nous permet d'écrire les deux règles de décision du décodeur *ML* comme suivant :

$$\hat{x}_1 = \arg \min_{\hat{x}_1 \in S} d^2(\hat{x}_1, \tilde{x}_1) \tag{2.22}$$

$$\hat{x}_2 = \arg \min_{\hat{x}_2 \in S} d^2(\hat{x}_2, \tilde{x}_2) \tag{2.23}$$

La méthode de détection se résume donc en deux étapes :

1. Calculer d'abord les deux paramètres de décision $\tilde{x}_1$ et $\tilde{x}_2$ selon les deux équations (2.16) et (2.17) respectivement. Cette opération est effectuée à l'aide du combineur, comme le montre la figure 2.2.
2. Chercher dans l'ensemble des symboles de la constellation le symbole le plus proche ; au sens de la distance euclidienne minimale ; du paramètre $\tilde{x}_1$ et le symbole le plus proche du paramètre $\tilde{x}_2$. Les deux symboles ainsi trouvés sont les estimations de x_1 et x_2 qui sont notés $\hat{x}_1$ et $\hat{x}_2$ respectivement.

Système d'Alamouti *2×2*

Un système d'Alamouti 2×2 utilise deux antennes de réception au lieu d'une seule pour augmenter le gain de diversité, et par conséquent réduire le *BER* en fonction du rapport *SNR*. Dans ce cas, la matrice du canal est constituée de quatre éléments h_{ij} avec *i=1,2* et *j=1,2.* La matrice des signaux reçus est donnée par :

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} . \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} + \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} \tag{2.24}$$

Avec r_{ij} (pour *i,j=1,2)* représente le signal reçu sur l'antenne de réception *j* durant la période *i* .

Le combineur calcule les deux paramètres de décision $\tilde{x}_1$ et $\tilde{x}_2$ selon les deux équations suivantes :

$$\tilde{x}_1 = h_{11}^*.r_{11} + h_{12}^*.r_{12} + h_{21}.r_{21}^* + h_{22}.r_{22}^* \tag{2.25}$$

$$\tilde{x}_2 = h_{21}^*.r_{11} + h_{22}^*.r_{12} - h_{11}.r_{21}^* - h_{12}.r_{22}^* \tag{2.26}$$

Les deux règles utilisées pour le décodage séparé des deux symboles x_1 et x_2 sont données par :

$$\hat{x}_1 = \arg\min_{\hat{x}_1 \in S} \left[\sum_{j=1}^{2} \left(|h_{1j}|^2 + |h_{2j}|^2 - 1 \right) |\hat{x}_1|^2 + d^2(\hat{x}_1, \tilde{x}_1) \right] \quad (2.27)$$

$$\hat{x}_2 = \arg\min_{\hat{x}_2 \in S} \left[\sum_{j=1}^{2} \left(|h_{1j}|^2 + |h_{2j}|^2 - 1 \right) |\hat{x}_2|^2 + d^2(\hat{x}_2, \tilde{x}_2) \right] \quad (2.28)$$

Pour les constellations *M-PSK,* les deux règles de décision se réduisent à la minimisation des deux distances $d^2(\hat{x}_1, \tilde{x}_1)$ et $d^2(\hat{x}_2, \tilde{x}_2)$ pour le décodage séparé de x_1 et x_2 respectivement.

2.2.3 Codage espace-temps en treillis :

Le codage espace-temps en treillis *STTC* est une technique de codage convolutif, qui permet d'assurer à la fois un gain de codage et un gain de diversité en utilisant N_t antennes à l'émission et N_r antennes à la réception. En 1998, Tarokh *et al* [52] ont proposé la construction du codage espace-temps en treillis comme extension de la modulation en treillis « Trellis Coded Modulation » *TCM* au cas des systèmes *MIMO*. Pour le codage *STTC,* chaque antenne envoie un symbole qui dépend des symboles envoyés à l'instant précèdent sur la même antenne et sur les autres antennes. Cette technique permet donc de créer une corrélation en espace et en temps entre les différents symboles envoyés.

2.2.3.a Structure du codeur *STTC*

La figure 2.3 représente la structure générale d'un codeur *STTC*. Pour une modulation *M-PSK,* le codeur reçoit à son entrée $m = \log_2(M)$séquences binaires données par :

$$\boldsymbol{C} = \left(\boldsymbol{c}^1, \boldsymbol{c}^2, \ldots \boldsymbol{c}^m \right) \quad (2.29)$$

Où la $k^{ième}$ séquence $\boldsymbol{c}^k = \left(c_1^k, c_2^k, \ldots c_t^k \ldots \right)$ est passée à travers le $k^{ième}$ registre à décalage et est multipliée par une série de coefficients de multiplication. Les résultats

de décalages et de multiplications sont ensuite, additionnées en modulo *M* pour former la sortie du codeur qui est donnée par :

$$\boldsymbol{x} = \left(\boldsymbol{x}^1, \boldsymbol{x}^2, ..\boldsymbol{x}^{N_t}\right) \tag{2.30}$$

Où $\boldsymbol{x}^i$ représente l'ensemble des symboles codés en espace et en temps, et qui sont envoyés sur la $i^{ième}$ antenne d'émission au fil du temps.

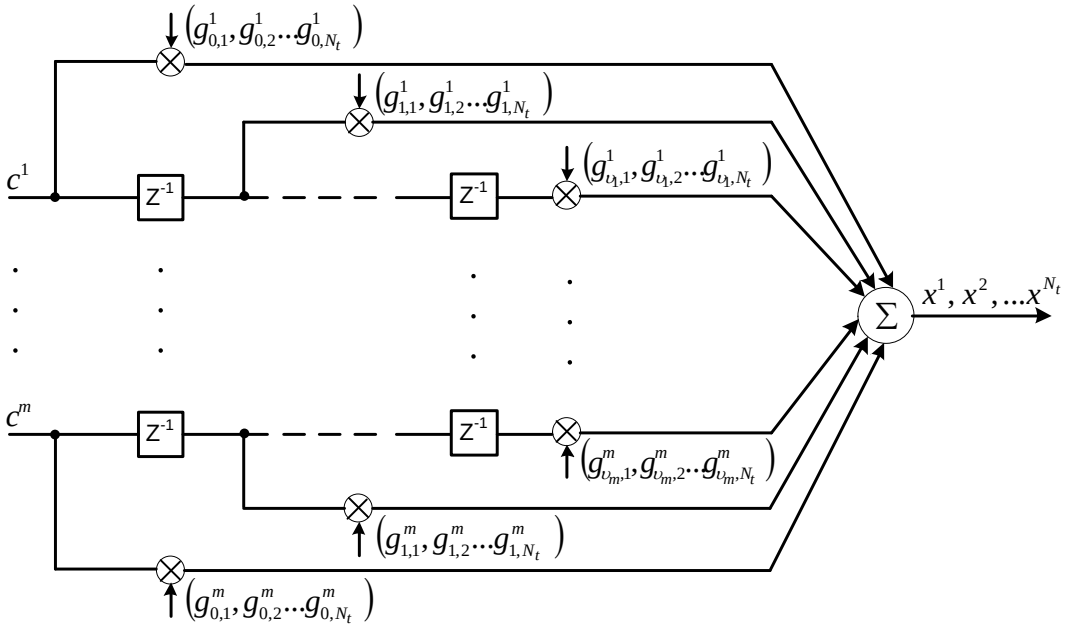

Figure 2.3 : structure générale d'un codeur *STTC* [12].

Les coefficients de multiplication, connus sous le nom de *séquences génératrices,* sont donnés par :

$$\boldsymbol{g}^1 = \left[\left(g_{0,1}^1, g_{0,2}^1 ... g_{0,N_t}^1\right), \left(g_{1,1}^1, g_{1,2}^1 ... g_{1,N_t}^1\right), .. \left(g_{\upsilon_1,1}^1, g_{\upsilon_1,2}^1 ... g_{\upsilon_1,N_t}^1\right)\right] \tag{2.31}$$

$$\boldsymbol{g}^2 = \left[\left(g_{0,1}^2, g_{0,2}^2 ... g_{0,N_t}^2\right), \left(g_{1,1}^2, g_{1,2}^2 ... g_{1,N_t}^2\right), .. \left(g_{\upsilon_2,1}^2, g_{\upsilon_2,2}^2 ... g_{\upsilon_2,N_t}^2\right)\right] \tag{2.32}$$

Jusqu'à :

$$\boldsymbol{g}^m = \left[\left(g_{0,1}^m, g_{0,2}^m ... g_{0,N_t}^2\right), \left(g_{1,1}^m, g_{1,2}^m ... g_{1,N_t}^m\right), .. \left(g_{\upsilon_m,1}^m, g_{\upsilon_m,2}^m ... g_{\upsilon_m,N_t}^m\right)\right] \tag{2.33}$$

Où $g_{j,i}^{k}$ *k=0,1,…m, i=1,2,…N_t, j=0,1,…υ_k*, sont des éléments de la constellation *M-PSK* et υ_k représente l'ordre du *k^ième^* registre à décalage.

La sortie du codeur qui correspond à l'antenne d'émission *i* durant le temps *t* est donnée par :

$$x_t^i = \sum_{k=0}^{m} \sum_{j=0}^{\upsilon_k} g_{j,i}^{k} c_{t-j}^{k} Mod\ M \quad i=1, 2, \ldots N_t \tag{2.34}$$

Pour une modulation de type *M-PSK,* la technique *STTC* permet d'atteindre une efficacité de largeur de bande égale à *m bits/s/Hz*.

L'ordre total de mémoire du codeur est donné par :

$$\upsilon = \sum_{k=1}^{m} v_k \tag{2.35}$$

En connaissant l'ordre totale du codeur, l'ordre du *k^ième^* registre à décalage est donnée par :

$$v_k = \left\lfloor \frac{v + k - 1}{\log_2(M)} \right\rfloor \tag{2.36}$$

Où l'opérateur $\lfloor x \rfloor$ représente la valeur entière de *x*.

Si on prend l'exemple de la constellation *QPSK* (*m=2*) avec deux antennes d'émission (N_t=*2*), les deux séquences génératrices auront la forme :

$$\boldsymbol{g}^1 = \left[\left(g_{0,1}^1, g_{0,2}^1\right), \left(g_{1,1}^1, g_{1,2}^1\right), \ldots \left(g_{\upsilon_1,1}^1, g_{\upsilon_1,2}^1\right) \right] \tag{2.37}$$

$$\boldsymbol{g}^2 = \left[\left(g_{0,1}^2, g_{0,2}^2\right), \left(g_{1,1}^2, g_{1,2}^2\right), \ldots \left(g_{\upsilon_2,1}^2, g_{\upsilon_2,2}^2\right) \right] \tag{2.38}$$

Où $g_{j,i}^{k} \in \{0, 1, 2, 3\}$, *k=1,…m, i=1,2* et *j=0,1,…υ_k.*

La construction des codes *STTC* optimaux consiste à trouver les séquences génératrices de codes selon des critères d'optimisation (critère du rang, du déterminant, de la trace….) [11], [12]. La construction est basée généralement sur des algorithmes de recherche, et elle dépend du type d'évanouissement

(évanouissement lent ou rapide), de la constellation, du nombre d'antennes d'émission et de l'ordre de la mémoire.

La technique *STTC* fait partie des codes convolutifs dont la représentation la plus populaire est celle utilisant le diagramme en treillis [11], [12]. Ce diagramme met en évidence les transitions des états du codeur, ainsi que les symboles à envoyer sur chaque antenne d'émission en fonction de l'état actuel du codeur et de l'entrée du codeur. Généralement, les états du codeur sont représentés sur les nœuds alors que les branches représentent les transitions de l'état présent aux états suivants. Le nombre de branches qui arrivent ou partent du même nœud (état) dépend du nombre de bits par symbole à transmettre. Pour transmettre *m* bits, il y a 2^m branches qui partent et qui arrivent au même nœud.

Revenant à l'exemple du codeur *STTC* pour un système à deux antennes d'émission ($N_t=2$) qui utilise une constellation *QPSK (m=2).* Si on considère les deux séquences génératrices $\boldsymbol{g}^1 = [(0,2), (2,0)]$ et $\boldsymbol{g}^1 = [(0,1), (1,0)]$, la représentation en treillis du codeur aura la structure donnée dans la figure 2.4. Le treillis contient 4 états ($\upsilon=2$) notés état 0 à 3. A chaque instant, le codeur prend deux bits ($m=2$) comme entrée, il y a donc 4 branches partant de chaque état et qui correspondent aux quatre combinaisons possibles. Pour chaque branche, on présente les deux bits d'entrée b_1b_2 et les deux symboles codés en espace et en temps x^1, x^2, et qui vont être transmis sur la première et la deuxième antenne respectivement (b_1b_2/x^1x^2). Les représentations de gauche à droite correspondent aux branches du haut en bas.

Considérant, par exemple, que le codeur se trouve à l'état 1(deuxième ligne du treillis) :

- Si l'entrée est 00, le codeur génère le symbole 1 sur l'antenne 1 et le symbole 0 sur l'antenne 2, et transite vers l'état 0.
- Si l'entrée est 01, le codeur génère le symbole 1 sur les deux antennes, et reste dans l'état 1.
- Si l'entrée est 10, le codeur génère le symbole 1 sur l'antenne 1 et le symbole 2 sur l'antenne 2, et passe à l'état 2.

- Si l'entrée est 11, le codeur génère le symbole 1 sur l'antenne 1 et le symbole 3 sur l'antenne 2, et passe à l'état 3.

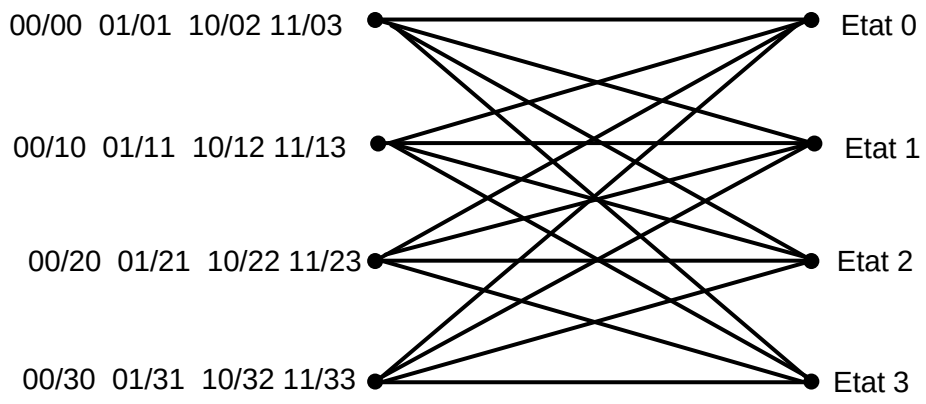

Figure 2.4 : Représentation en treillis d'un codeur *STTC* à 4 états pour deux antennes d'émission

2.2.3.b Décodage des codes *STTC*

Pour le codage *STTC,* l'émission des mots de code est réalisée par des trames qui débutent et se terminent par le même état, qui est par convention l'état "0". Cela permet au décodeur de connaitre à priori l'état initial et l'état final pour chaque trame émise. La transition d'un état à un autre dans le diagramme en treillis crée une dépendance entre les états du chemin de la trame. Le décodage des codes *STTC* est basé sur la connaissance des deux états de début et de fin de la trame et de la dépendance existant entre les différents états du chemin de la trame. La technique qui répond à ce type de détection est l'algorithme de Viterbi. L'algorithme de Viterbi est un algorithme qui permet l'estimation séquentielle au sens du maximum de vraisemblance «Maximum Likelihood Sequential Estimation » *MLSE.* Il utilise la trame reçue pour chercher le chemin présentant la distance minimale en se basant sur la connaissance de la structure du codeur, de son état initial et de son état final. Ce chemin correspond à la séquence la plus probable.

2.3 Combinaison du codage espace-temps et de l'*OFDM*

Les techniques de codage espace-temps, vues dans la section précédente, permettent d'exploiter la diversité spatiale afin de lutter contre les évanouissements relatifs aux canaux de transmission radio mobile. La plupart de ces codes sont conçus pour combattre les évanouissements des canaux non-sélectifs en fréquences. Pour les systèmes de communication à haut débit, le canal devient rapidement sélectif en fréquence, ce qui dégrade les performances des techniques de codage espace-temps et perdent leur efficacité. L'utilisation du codage espace-temps pour les canaux sélectifs en fréquence nécessite alors sa combinaison avec d'autres techniques appropriées à ce type de canaux. Une façon efficace pour bénéficier de la diversité fréquentielle et de la diversité spatiale consiste à utiliser conjointement le codage spatio-temporel et la modulation *OFDM*. Selon le type du codage utilisé, plusieurs réalisations de cette combinaison ; appelée STC-OFDM ou MIMO-OFDM ; sont possibles [13], [15], [53], [54].

La figure 2.5 montre le schéma de principe de la combinaison du codage espace-temps et de l'*OFDM*. Les symboles mappés selon une constellation déterminée sont d'abord codés en espace et en temps, puis modulés en symboles *OFDM*. Le codeur *STC* dispose de N_t sorties, donc le nombre de modulateurs *OFDM* nécessaires pour un système *MIMO* est multiplié par N_t en comparaison avec un système *SISO.* De même, le nombre de démodulateurs *OFDM* nécessaire à la réception est multiplié par N_r. Cela montre que la technique *OFDM* est plus compliquée dans le contexte *MIMO,* et elle nécessite des ressources matérielles importantes pour son implémentation sur un circuit numérique.

Pour les codeurs espace-temps de type *STBC,* le codage peut s'effectuer sur les symboles *OFDM,* dans ce cas on parle de *STBC-OFDM*. Le codage peut aussi s'effectuer sur les échantillons d'un symbole *OFDM,* on parle donc de *SFBC-OFDM*. L'inconvénient des systèmes *MIMO-OFDM* est qu'ils sont sensibles aux offsets de fréquence entre l'oscillateur de l'émetteur et celui du récepteur, et au décalage de fréquence causé par l'effet Doppler.

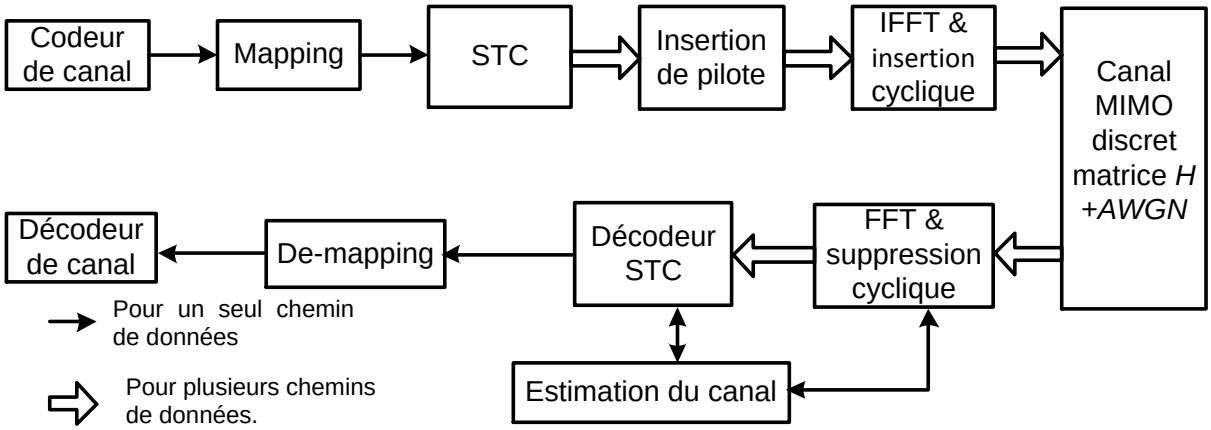

Figure 2.5 : Schéma de principe d'un système combinant le codage espace-temps et l'*OFDM*.

2.4. Combinaison du codage espace-temps et du *MC-CDMA*

La technique *MC-CDMA* offre la capacité de l'accès multiple par répartition de code, une efficacité spectrale importante par l'exploitation de la diversité temporelle, fréquentielle et de code. Le codage espace-temps permet d'améliorer les performances d'un système *MIMO* en exploitant la diversité spatiale. La combinaison du *MC-CDMA* et du codage *STC* permet donc d'exploiter les quatre diversités (spatiale, temporelle, fréquentielle et du code) [16], [17], [55], [56]. Le système de communication combinant le codage O*STBC* et le *MC-CDMA* est présenté dans la figure 2.6.

Selon le principe de la combinaison du codage espace-temps et du *MC-CDMA,* l'opération de codage espace-temps devrait normalement précéder l'opération de l'étalement de spectre. Cependant, les opérations dans le codage espace-temps et dans le *MC-CDMA* sont linéaires pour le cas de l'*OSTBC*. Ces opérations peuvent donc être interverties sans changer le signal modulé. On peut donc effectuer l'étalement de spectre avant le codage spatio-temporel, comme le montre la figure 2.6. Cette permutation permet d'utiliser un seul circuit d'étalement au lieu de N_t circuits.

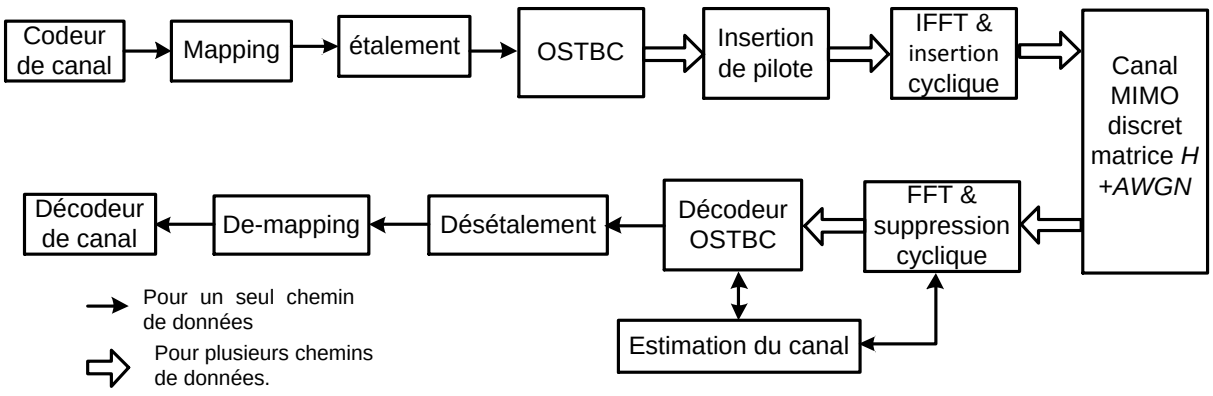

Figure 2.6 : Schéma synoptique de base d'un système *OSTBC-MC-CDMA*.

2.5. Présentation des architectures proposées pour l'implémentation sur *FPGA* du décodeur espace-temps d'Alamouti

Dans cette partie, nous présentons deux architectures pour l'implémentation du décodeur d'Alamouti, configurables pour le cas 2×2 ($N_r=2$) et 2×1 ($N_r=1$). Nous commençons d'abord par les schémas des architectures afin d'expliquer leurs fonctionnements. Nous passons ensuite à la présentation des résultats de simulation, en comparant le *BER* obtenu par les architectures proposées avec celui obtenu en utilisant le décodeur d'Alamouti fourni par la bibliothèque de Simulink. Enfin, nous donnons les résultats d'implémentation sur *FPGA* des architectures proposées, en comparaison avec l'architecture conventionnelle du décodeur d'Alamouti.

2.5.1. Présentation de la première architecture

Afin de simplifier l'explication, nous commençons par la présentation de la première architecture pour le cas 2×1, dont le schéma est celui de la figure 2.7. Le décodeur d'Alamouti utilise les données reçues sur deux périodes de temps pour estimer les deux symboles $\hat{x}_1$ et $\hat{x}_2$. L'idée principale de l'architecture proposée est de réutiliser le même multiplieur pour calculer deux produits complèxes sur deux périodes de temps. De cette façon, le décodeur d'Alamouti pour $N_r=1$ ne nécessite que deux multiplieurs au lieu de quatre pour une implémentation conventionnelle. Ainsi le nombre de multiplieurs est réduit à la moitié.

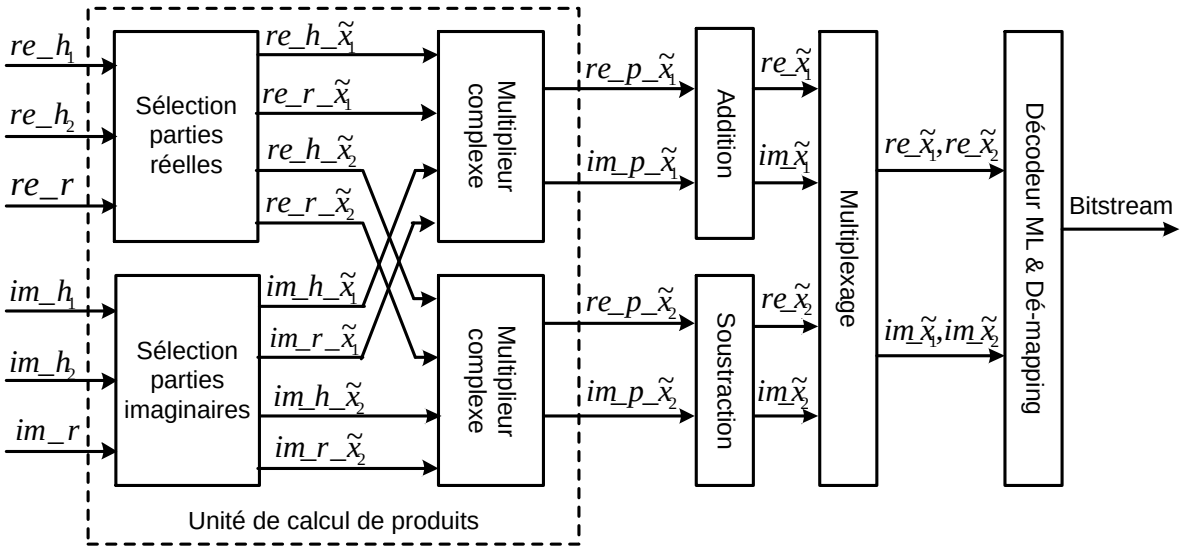

Figure 2.7 : Schéma de la première architecture proposée pour le décodeur d'Alamouti 2×1.

Revenant aux expressions (2.16) et (2.17) des deux métriques d'estimation $\tilde{x}_1$ et $\tilde{x}_2$, qu'on peut réécrire sous la forme :

$$\tilde{x}_1 = \overbrace{h_1^*.r(1)}^{P\_\tilde{x}_1(1)} + \overbrace{h_2.r^*(2)}^{P\_\tilde{x}_1(2)} \tag{2.39}$$

$$\tilde{x}_2 = \underbrace{h_2^*.r(1)}_{P\_\tilde{x}_2(1)} - \underbrace{h_1.r^*(2)}_{P\_\tilde{x}_2(2)} \tag{2.40}$$

Le premier multiplieur de la figure 2.7 permet le calcul de $p\_\tilde{x}_1(1)$ durant la première période et $p\_\tilde{x}_1(2)$ pendant la deuxième période. De même, le deuxième multiplieur calcule $p\_\tilde{x}_2(1)$ et $p\_\tilde{x}_2(2)$ durant la première et la deuxième période, respectivement. Durant la première période, les deux unités de sélection permettent de sélectionner $(h_1^*, r(1))$ pour le premier multiplieur et $(h_2^*, r(1))$ pour le deuxième multiplieur. Durant la deuxième période $(h_2, r^*(2))$ et $(h_1, r^*(2))$ sont sélectionnés pour le premier et le deuxième multiplieur, respectivement. L'unité

d'addition permet de calculer $\tilde{x}_1$ par la sommation de $p\_\tilde{x}_1(1)$ et $p\_\tilde{x}_1(2)$ alors que l'unité de soustraction calcule $\tilde{x}_2$ en effectuant la différence entre $p\_\tilde{x}_2(1)$ et $p\_\tilde{x}_2(2)$. Les deux flux $\tilde{x}_1$ et $\tilde{x}_2$ sont regroupés en un seul flux afin de procéder à l'opération de décodage *ML* et de dé-mapping. Pour les opérations de décodage *ML* et de dé-mapping, nous avons utilisé une technique très simple, et qui ne nécessite pas de ressources additionnelles. Pour la modulation *BPSK,* on utilise uniquement le bit de signe de la partie réelle. Pour la modulation *QPSK,* on utilise le bit de signe de la partie réelle et celui de la partie imaginaire pour effectuer le décodage *ML* et de dé-mapping

Après présentation de la première architecture pour le cas 2×1, passons maintenant à la présentation de l'architecture configurable pour le cas 2×1 et 2×2, dont le schéma est donné dans la figure 2.8. Pour expliquer le fonctionnement de l'architecture, on réécrit les deux équations (2.25) et (2.26) sous la forme suivante :

$$\tilde{x}_1 = \overbrace{\underbrace{h_{11}^*.r_1(1)}_{p\_ant_1\_\tilde{x}_1(1)} + \underbrace{h_{12}^*.r_2(1)}_{p\_ant_2\_\tilde{x}_1(1)}}^{s\_\tilde{x}_1(1)} + \overbrace{\underbrace{h_{21}.r_1^*(2)}_{p\_ant_1\_\tilde{x}_1(2)} + \underbrace{h_{22}.r_2^*(2)}_{p\_ant_2\_\tilde{x}_1(2)}}^{s\_\tilde{x}_1(2)} \quad (2.41)$$

$$\tilde{x}_2 = \underbrace{\underbrace{h_{21}^*.r_1(1)}_{p\_ant_1\_\tilde{x}_2(1)} + \underbrace{h_{22}^*.r_2(1)}_{p\_ant_2\_\tilde{x}_2(1)}}_{s\_\tilde{x}_2(1)} - \underbrace{(\underbrace{h_{11}.r_1^*(2)}_{p\_ant_1\_\tilde{x}_2(2)} + \underbrace{h_{12}.r_2^*(2)}_{p\_ant_2\_\tilde{x}_2(2)})}_{s\_\tilde{x}_2(2)} \quad (2.42)$$

Chacune des deux unités de calcul de produits est identique à celle de la figure 2.7. Chaque unité contient deux multiplieurs complexes avec leurs unités de sélection des entrées. Ainsi, l'unité de calcul de produits #1 permet de calculer $\{p\_ant_1\_\tilde{x}_1(1), p\_ant_1\_\tilde{x}_2(1)\}$ durant la première période et $\{p\_ant_1\_\tilde{x}_1(2), p\_ant_1\_\tilde{x}_2(2)\}$ durant la deuxième période. De la même manière, l'unité de calcul de produits #2 calcule $\{p\_ant_2\_\tilde{x}_1(1), p\_ant_1\_\tilde{x}_2(1)\}$ et $\{p\_ant_2\_\tilde{x}_1(2), p\_ant_2\_\tilde{x}_2(2)\}$ pendant la première et la deuxième période, respectivement.

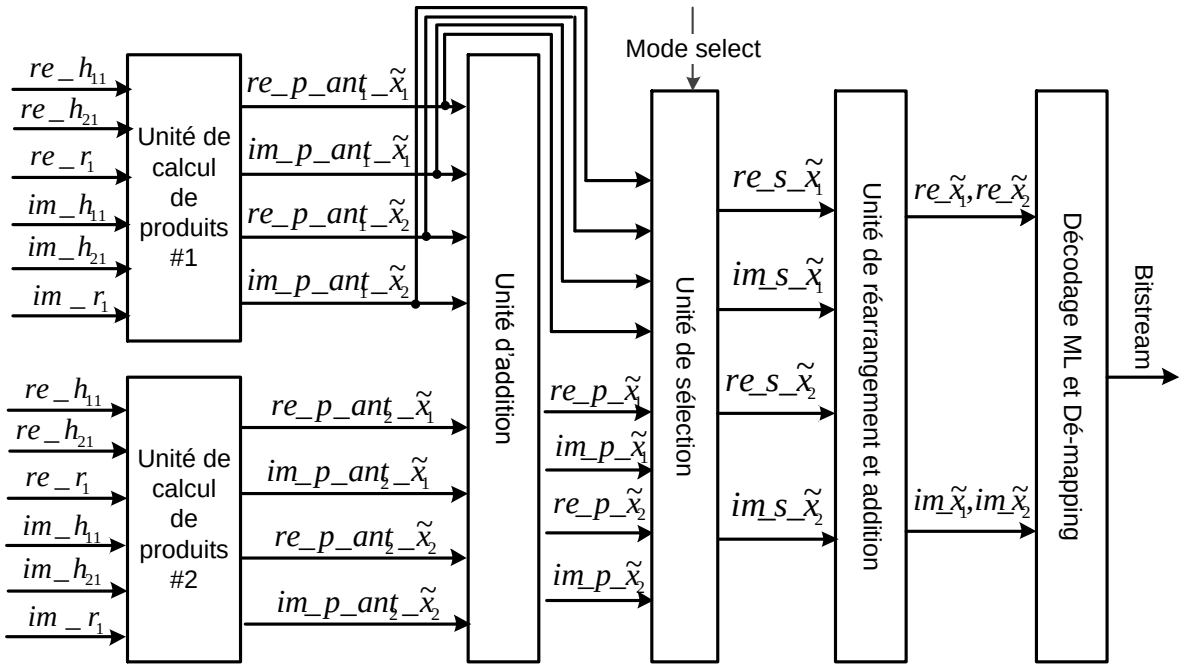

Figure 2.8 : Schéma de la première architecture proposée pour le décodeur d'Alamouti configurable pour 2×1 et 2×2.

L'unité d'addition contient deux additionneurs complexes. Durant la première période, le premier additionneur réalise la somme de $p\_ant_1\_\tilde{x}_1(1)$ et $p\_ant_2\_\tilde{x}_1(1)$ pour former $s\_\tilde{x}_1(1)$, alors que le deuxième additionneur effectue la somme de $p\_ant_1\_\tilde{x}_2(1)$ et $p\_ant_2\_\tilde{x}_1(1)$ pour former $s\_\tilde{x}_2(1)$. Durant la deuxième période, l'unité d'addition calcule $s\_\tilde{x}_1(2)$ en sommant $p\_ant_1\_\tilde{x}_1(2)$ avec $p\_ant_2\_\tilde{x}_1(2)$, et $s\_\tilde{x}_2(2)$ en sommant $p\_ant_1\_\tilde{x}_2(2)$ avec $p\_ant_2\_\tilde{x}_2(2)$. L'unité de sélection permet de choisir entre le mode *2×1 et 2×2.* Pour le cas 2×1, l'unité d'addition est ignorée et les sorties de l'unité de calcul de produits #1 sont passées directement vers l'unité de réarrangement et d'addition. Pour le cas 2×2, ce sont les sorties de l'unité d'addition qui sont acheminées vers l'unité de réarrangement et d'addition. L'unité de réarrangement permet de réarranger les données d'une façon qui permet l'utilisation d'un seul additionneur pour calculer

$\tilde{x}_1$ suivie de $\tilde{x}_2$. Finalement, l'unité de décodage *ML* et de Dé-mapping utilise le bit de signe de la partie réelle et imaginaire de $\tilde{x}_1$ et $\tilde{x}_2$ pour l'estimation du bitstream.

Cette architecture permet l'estimation de deux symboles en deux cycles horloge, ce qui permet d'assurer un débit symbole égale à la fréquence du signal horloge. On outre, elle utilise quatre multiplieurs complexes et trois additionneurs complexes au lieu de huit multiplieurs et six additionneurs pour l'architecture conventionnelle.

2.5.2 Présentation de la deuxième architecture

La deuxième architecture proposée utilise uniquement deux multiplieurs complexes au lieu de quatre pour la première architecture, mais elle permet l'estimation de deux symboles en quatre cycles horloge. Ce qui revient à un débit symbole qui est égale à la moitié de la fréquence du signal horloge. Le schéma de la deuxième architecture est donné dans la figure 2.9. Pour mettre en évidence le fonctionnement de la deuxième architecture proposée, on commence par la réécriture des deux équations (2.25) et (2.26) relatives aux deux métriques d'estimation $\tilde{x}_1$ et $\tilde{x}_2$, sous la forme suivante :

$$\tilde{x}_1 = \overbrace{\overbrace{h_{11}^{*}.r_1(1)}^{p\_ant_1\_\tilde{x}_1(1)} + \overbrace{h_{21}.r_1^{*}(2)}^{p\_ant_1\_\tilde{x}_1(2)}}^{s\_\tilde{x}_1(1)} + \overbrace{\overbrace{h_{12}^{*}.r_2(1)}^{p\_ant_2\_\tilde{x}_1(3)} + \overbrace{h_{22}.r_2^{*}(2)}^{p\_ant_2\_\tilde{x}_1(4)}}^{s\_\tilde{x}_1(3)} \quad (2.43)$$

$$\tilde{x}_2 = \underbrace{\underbrace{h_{21}^{*}.r_1(1)}_{p\_ant_1\_\tilde{x}_2(1)} - \underbrace{h_{11}.r_1^{*}(2)}_{p\_ant_1\_\tilde{x}_2(2)}}_{s\_\tilde{x}_2(2)} + \underbrace{\underbrace{h_{22}^{*}.r_2(1)}_{p\_ant_2\_\tilde{x}_2(3)} - \underbrace{h_{12}.r_2^{*}(2)}_{p\_ant_2\_\tilde{x}_2(4)}}_{s\_\tilde{x}_2(4)} \quad (2.44)$$

La première unité de sélection joue un double rôle :

- d'un côté, elle permet de choisir entre le mode *2×1 et le* mode *2×2* selon la valeur du bit de sélection de mode (mode select) ;
- de l'autre côté, elle permet l'acheminement des entrées vers l'unité de calcul de produit selon le mode sélectionné. En mode *2×1* les entrées relatives à la première antenne de réception, qui sont les six entrées du haut sont acheminées en permanence vers l'entrée de l'unité de calcul de produit. En

mode *2×2*, les six entrées du haut qui sont relatives à la première antenne de réception sont acheminées vers l'entrée de l'unité de calcul de produit durant la première et deuxième période de traitement, suivies des six entrées du bas qui sont relatives à la deuxième antenne de réception durant la troisième et la quatrième période.

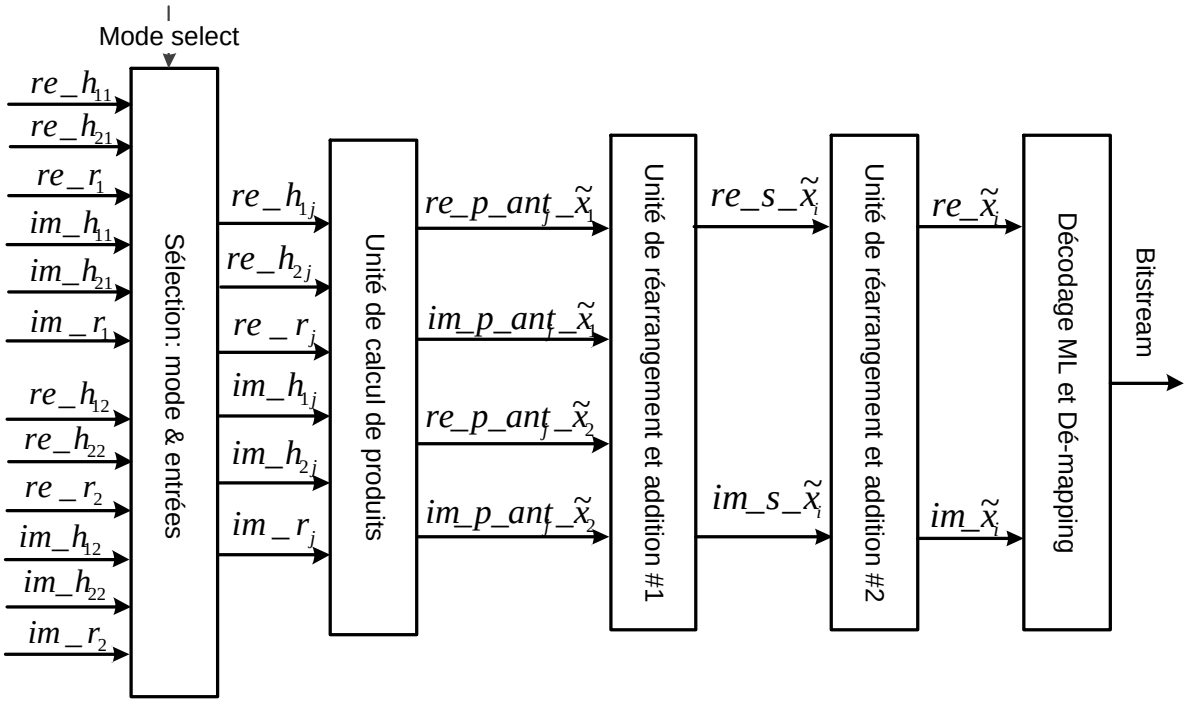

Figure 2.9 : Schéma de la deuxième architecture proposée pour le décodeur d'Alamouti configurable pour 2×1 et 2×2.

L'unité de calcul de produits ; qui est identique à celle de la figure 2.7 ; calcule huit produits durant les quatre périodes de traitement selon l'ordre suivant :

- durant la première période, $p\_ant_1\_\tilde{x}_1(1)$ et $p\_ant_1\_\tilde{x}_2(1)$;
- durant la deuxième période, $p\_ant_1\_\tilde{x}_1(2)$ et $p\_ant_1\_\tilde{x}_2(2)$;
- durant la troisième période, $p\_ant_2\_\tilde{x}_1(3)$ et $p\_ant_2\_\tilde{x}_2(3)$;
- durant la quatrième période, $p\_ant_2\_\tilde{x}_1(4)$ et $p\_ant_2\_\tilde{x}_2(4)$.

L'unité de réarrangement et d'addition #1 permet de réarranger les huit produits issus de l'unité de calcul de produits. Elle réalise aussi deux opérations d'addition et deux opérations de soustraction pour fournir $s\_\tilde{x}_1(1)$, $s\_\tilde{x}_2(2)$, $s\_\tilde{x}_1(3)$ et $s\_\tilde{x}_2(4)$ durant les périodes 1 à 4, respectivement. Le réarrangement des données permet d'utiliser un seul additionneur complexe pour réaliser les quatre opérations. L'unité de réarrangement et d'addition #2, réarrange les sorties de son prédécesseur avant de calculer $\tilde{x}_1$ suivie de $\tilde{x}_2$. Finalement, l'unité de décodage *ML* et de dé-mapping est la même que celle des figures (2.7) et (2.8).

2.5.3. Résultats de simulations

Pour évaluer les performances des deux architectures proposées, on a examiné le taux d'erreur binaire *BER* obtenu en utilisant les deux architectures en fonction du rapport signal sur bruit *SNR*. Afin de valider les résultats, on les a comparés avec ceux obtenus en utilisant le bloc de décodage *STBC* de la bibliothèque de Simulink. On rappelle que le bloc *STBC* de la bibliothèque de Simulink utilise la virgule flottante pour la représentation des données.

Pour la conception des architectures proposées, nous avons utilisé le logiciel « Xilinx System Generator » XSG [57], [58]. Ce logiciel est compatible et interactif avec l'environnement *Simulink* et l'environnement *Matlab*. Il permet d'importer des données de l'environnement *Matlab* et de l'environnement *Simulink.* De même, il peut exporter les données vers ces deux environnements. Cette propriété permet de faciliter la simulation, la détection et la correction des erreurs de conception, car il est beaucoup plus simple de visualiser et d'analyser les résultats sous l'environnement Matlab qu'avec d'autres outils utilisant le langage *VHDL* ou le langage Verilog comme l'outil ModelSim ou l'outil ISim. On outre, le logiciel *XSG* fournit une bibliothèque très riche en composants optimisés et configurables selon le besoin du concepteur. Cette bibliothèque contient une panoplie de composants, allant des composants les plus simples, comme les portes logiques jusqu'aux éléments compliqués dédiés au traitement numérique du signal (*DSP*) et à la communication numérique, comme les codeurs convolutifs, les décodeurs de Viterbi, les filtre *FIR*...etc. En plus, le logiciel

XSG permet de générer automatiquement le modèle *HDL* (Hardware Discription Language) en langage *VHDL* ou Verilog afin de passer rapidement vers l'étape d'implémentation.

Le logiciel *XSG* utilise la virgule fixe pour la représentation des données. Le choix de la longueur du mot binaire choisie pour la représentation des données (partie entière et partie fractionnaire) influe sur les résultats de calculs et donc sur le *BER* obtenu. La représentation de la partie entière par un nombre insuffisant de bits risque de générer des débordements de capacité (over flow). De même, l'utilisation d'un nombre insuffisant de bits pour la partie fractionnaire engendre une perte de la précision. De l'autre part, la sur-utilisation de bits pour la partie entière ou la partie fractionnaire entraîne l'augmentation inutile des ressources *FPGA* utilisées pour l'implémentation. Il est donc très important de chercher le nombre de bits de représentation qui assure un bon compromis entre les ressources utilisées et le *BER* obtenu.

Les deux figures 2.10 et 2.11 montrent l'influence de la longueur totale du mot (partie réelle et partie fractionnaire) *wl,* utilisée pour la représentation en virgule fixe des coefficients du canal h_{ij} et des signaux reçus r_1 et r_2. Nous avons constaté à travers les simulations effectuées, que 4 bits suffisent pour la représentation de la partie entière des entrées. Le nombre de bits restant pour la partie fractionnaire est donc égal à *wl-4.* Pour les sorties des multiplieurs, nous avons tronqué 8 bits de la partie fractionnaire *(tr=8).* Les deux figures montrent que le *BER* obtenu en virgule flottante pour un *SNR=12* est proche de 0.01 pour la modulation *QPSK* et est proche de 0.0025 pour la modulation *BPSK.* Ce qui confirme la supériorité de la modulation *BPSK* par rapport à la modulation *QPSK* en termes du *BER*. En outre, les deux figures indiquent que les architectures proposées permettent d'assurer des performances très proches de celles de la virgule flottante pour *wl=10.* Les deux figures 2.10 et 2.11 montrent aussi que les performances se dégradent avec la diminution de la valeur de *wl,* jusqu'à ce que le codage espace-temps perde son efficacité pour *wl=8,* où la partie fractionnaire des sorties de multiplicateurs est totalement tronquée.

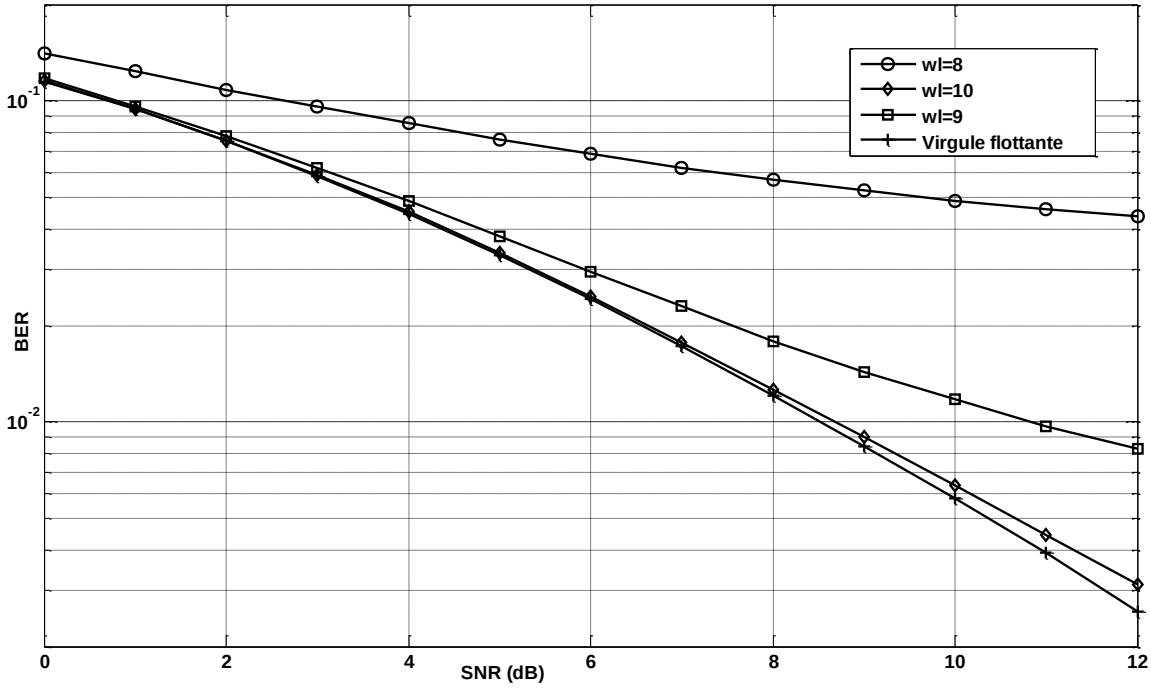

Figure 2.10 : Influence de la longueur de mots *wl* sur le *BER* en fonction du *SNR* pour le cas 2×1 et la modulation *BPSK*.

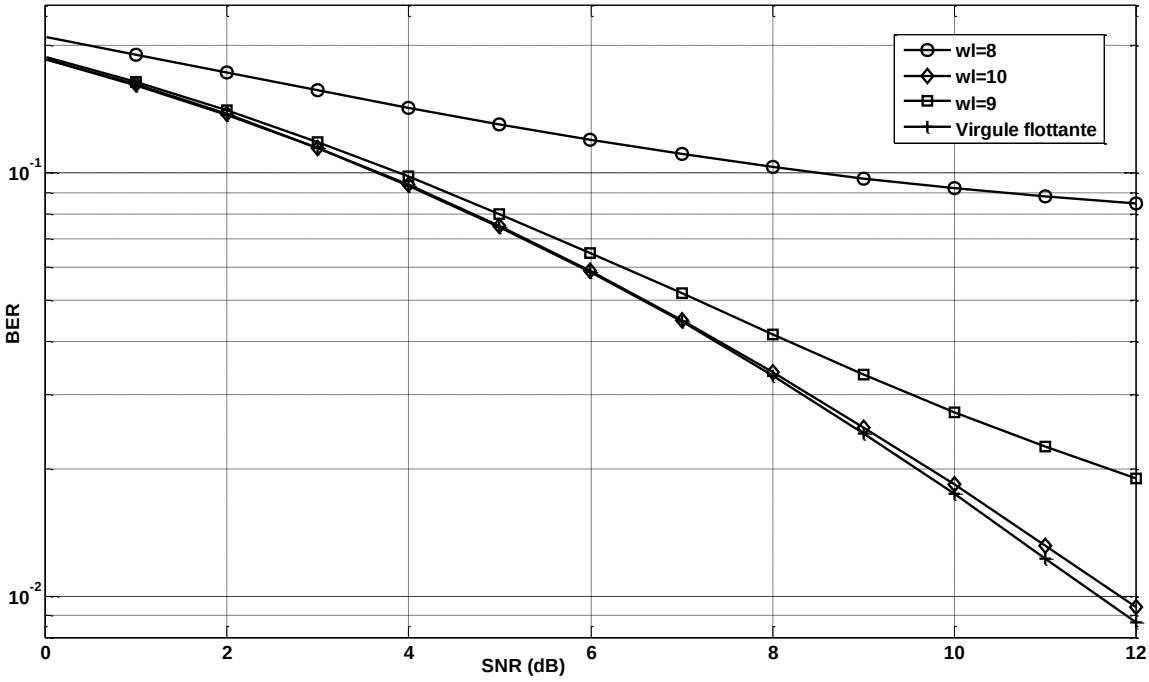

Figure 2.11 : Influence de la longueur de mots *wl* sur le *BER* en fonction du *SNR* pour le cas 2×1 et la modulation *QPSK*.

Les deux figures 2.12 et 2.13 présentent l'influence de *wl* sur le *BER* en fonction du *SNR* pour le cas *2×2,* et en utilisant les modulations *BPSK* et *QPSK,* respectivement. A partir de ces deux figures, on peut constater clairement l'amélioration apportée par l'augmentation du nombre d'antennes de réception à deux antennes au lieu d'une seule. En effet, les résultats de simulations montrent qu'en virgule flottante et pour un *SNR=12*, le *BER* passe de $25{,}74\times10^{-4}$ pour un système *2×1* à 0.28×10^{-4} pour un système *2×2* utilisant la modulation *BPSK*. En modulation *QPSK* et pour la même valeur du *SNR,* le *BER* passe de $87{,}22\times10^{-4}$ à $2{,}40\times10^{-4}$. En outre, les deux figures 2.12 et 2.13 montrent que l'utilisation de 10 bits comme longueur de mots pour les entrées permet d'assurer un *BER* très proche de celui obtenu par l'utilisation de la virgule flottante. Cependant, le cas de *wl=8* rend pratiquement inutile le codage spatio-temporel d'Alamouti 2×2 et 2×1.

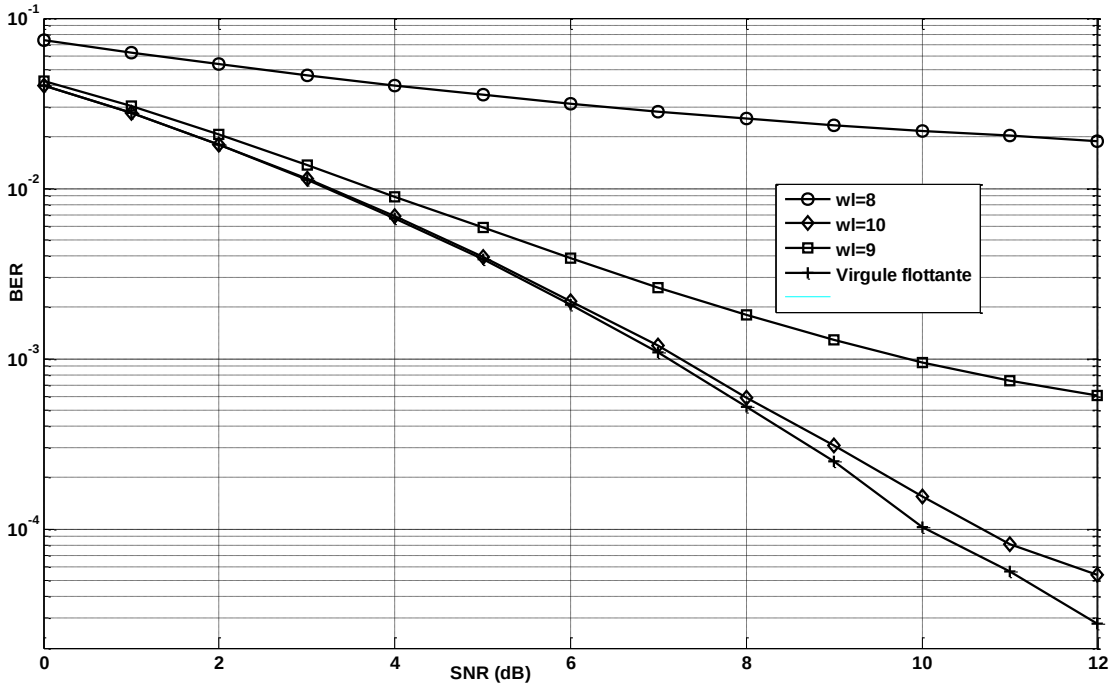

Figure 2.12 : Influence de la longueur de mots *wl* sur le *BER* en fonction du *SNR* pour le cas 2×2 et la modulation *BPSK*.

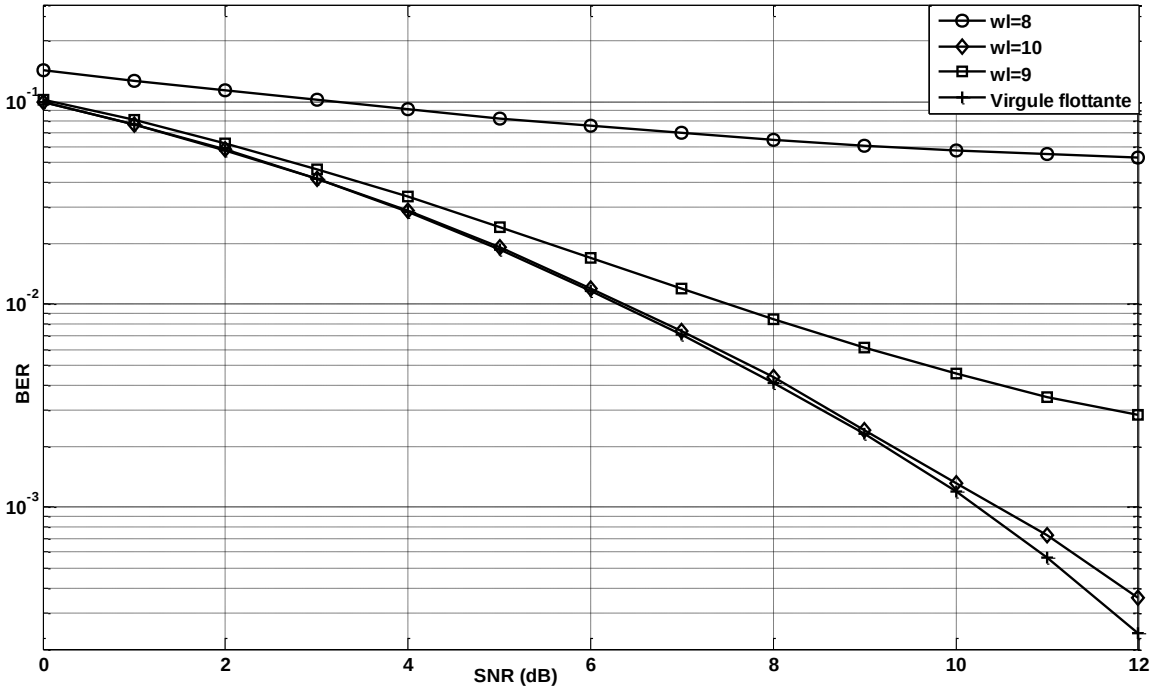

Figure 2.13 : Influence de la longueur de mots *wl* sur le *BER* en fonction du *SNR* pour le cas 2×2 et la modulation *QPSK*.

Pour connaitre l'effet de la troncature des sorties des multiplieurs, nous avons effectué plusieurs simulations en changeant le nombre de bits tronqués *tr* pour les deux modes 2×2 et 2×1 et pour les modulations *BPSK* et *QPSK*. Le format des entrées pour ces simulations est fixé à 10.6, c'est-à-dire dix bits pour le mot total dont 6 pour la partie fractionnaire.

En ce qui concerne l'influence du nombre d'antennes de réception et du type de la modulation, nous avons constaté les mêmes remarques que celles montrées dans les figures 2.10 à 2.13. On se limite donc à présenter ; dans la figure 2.14 ; les résultats de simulation pour la configuration d'Alamouti *2×2* et la modulation *BPSK.* Cette figure montre que la troncature de six bits offre un *BER* presque identique à celui obtenu en utilisant la virgule flottante. De même, la troncature de 8 bits assure des résultats très proches de celles de la virgule flottante. Alors que la troncature de la totalité de la partie fractionnaire (*tr=12*) rend inutile l'utilisation du codage d'Alamouti même en augmentant le *SNR.*

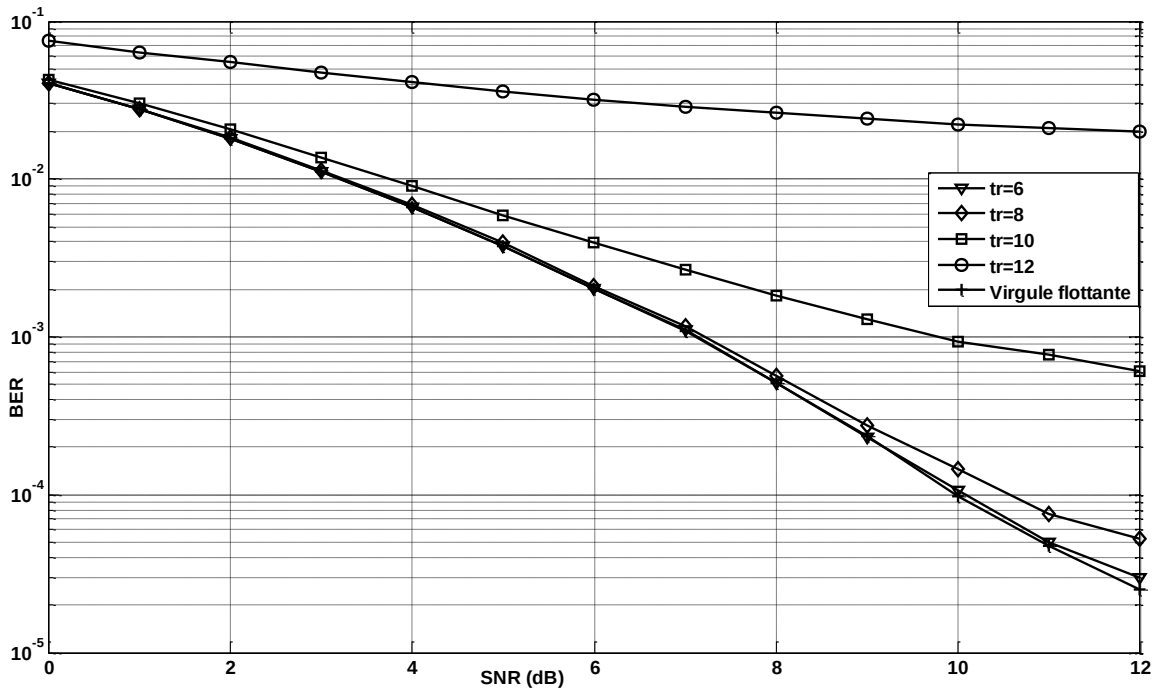

Figure 2.14 : Influence du nombre de bits tronqués *tr* sur le *BER* en fonction du *SNR* pour le cas 2×2 et la modulation *BPSK*.

2.5.4 Résultats de l'implémentation

Après vérification par simulations des deux architectures proposées, on présente dans cette partie les résultats d'implémentation de ces deux architectures sur un circuit *FPGA* de type *Virtex-7 FPGA XC7VSX485T-2FFG1761* [59], [60]. L'implémentation a été effectuée en utilisant l'outil *ISE* de *Xilinx* [61]. Les différents composants et modules des deux architectures sont configurés pour fonctionner à des fréquences élevées à travers l'insertion d'un nombre approprié de registres de pipeline à leurs sorties, afin d'améliorer le débit des données [18]. Les éléments de retard utilisés dans les architectures sont implémentés à l'aide de ressource *SRL* «Shift Register Look-up table ». Les multiplieurs complexes sont implémentés par les ressources *FPGA* de type *DSP48E,* et pour des raisons de comparaison ils sont implémentés aussi à l'aide des *LUT*s (Look Up Tables). Tous les autres composants des deux architectures sont implémentés par des slices standards du circuit *FPGA*. Le tableau 2.1 donne les résultats de l'implémentation des deux architectures

proposées en comparaison avec l'architecture conventionnelle. La comparaison est faite pour les deux types d'implémentation basse et haute fréquence. Les implémentations à basse fréquence n'utilisent pas de registres de pipeline aux sorties des composants. Ce type d'implémentations permet de réduire le nombre de registres Flip-Flop (*FF*) utilisés, mais au détriment d'une grande diminution de la fréquence de fonctionnement.

Tableau 2.1 : Ressources utilisées et fréquences maximales des architectures proposées implémentées sur le circuit Virtex-7 FPGA XC7VSX485T-2FFG1761738.

Architecture	HF/BF	Mults par :	Registres FF	LUT	Slices Occupés	DSP48E	Fréquence (MHz)
Conventionnelle	BF	DSP	145	241	110	32	334
		LUT	145	3653	1049	0	279
	HF	DSP	512	436	221	32	547
		LUT	4472	4318	1462	0	534
Proposée 1	BF	DSP	67	272	114	16	124
		LUT	89	1984	599	0	112
	HF	DSP	774	677	339	16	584
		LUT	2754	2650	857	0	512
Proposée 2	BF	DSP	201	299	132	8	109
		LUT	201	1149	341	0	106
	HF	DSP	690	367	164	8	602
		LUT	1680	1343	450	0	545

Les résultats du tableau 2.1 nous permettent de citer les remarques suivantes :

- L'implémentation haute fréquence assure une amélioration très importante de la fréquence en comparaison avec l'implémentation à basse fréquence, surtout pour les deux architectures proposées.
- L'amélioration importante de la fréquence n'est pas accompagnée par une augmentation excessive de ressources utilisées, sauf les registres *FF*. Pour la première architecture proposée, et pour le cas d'utilisation de *DSP48E* pour les

multiplieurs, la fréquence est améliorée par un facteur de 5,36 alors que le nombre de *LUT*s utilisés est augmenté d'un rapport de 2,49, et le nombre des slices d'un rapport de 2.97. Pour la deuxième architecture proposée, le facteur d'amélioration de la fréquence est égal à 5.52 avec un facteur d'augmentation de *LUTs* égale à 1,23 et un coefficient d'augmentation de Slices occupés égal à 1,24. Ce qui signifie que l'implémentation à haute fréquence est efficace en termes du rapport fréquence à ressources utilisées.

- La première architecture proposée permet de réduire le nombre des slices de type *DSP48E* par un facteur de 2 en comparaison avec l'architecture conventionnelle, et la deuxième architecture permet de réduire le nombre par un facteur de 4. Cette réduction est très significative, car les circuits *FPGA* disposent d'un nombre très limité de ce type de slices en comparaison avec d'autres ressources. A titre d'exemple, le circuit *Virtex-7 FPGA XC7VSX485T-2FFG1761* ne contient que *2800 DSP48E* contre *607200* registres, *303600 LUT*s et *75900* Slices.
- Si on utilise les *LUT*s pour implémenter les multiplieurs, la deuxième architecture proposée est la moins exigeante en ressources, suivie de la première architecture alors que l'architecture conventionnelle nécessite un nombre excessif de ressources. Cela est dû au fait que les multiplieurs complexes utilisent un nombre important de ressources lorsqu'ils sont implémentés par des *LUT*s. Les résultats d'implémentation montrent qu'un seul multiplieur configuré pour fonctionner à la fréquence maximale nécessite *538* registres *FF*, 639 *LUT*s et *225* slices.

Afin de connaître la consommation de puissance des architectures proposées, nous avons effectué l'estimation de la puissance consommée en utilisant le logiciel *Xpower Analyser* [62]. Les figures 2.15 et 2.16 donnent les graphes de la puissance dynamique consommée par les deux architectures proposées, et de l'architecture conventionnelle, en fonction de la fréquence, en cas d'utilisation des *DSP48E* et en cas d'utilisation de *LUT*s pour l'implémentation des multiplieurs.

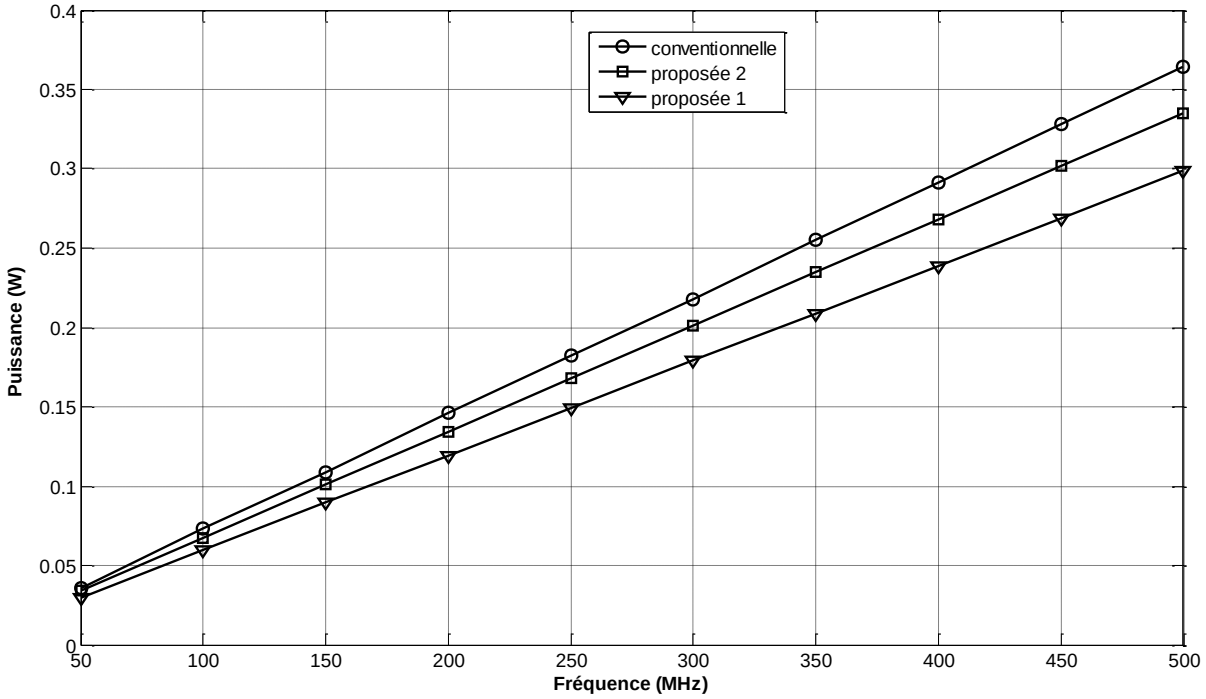

Figure 2.15 : Puissance dynamique consommée par les architectures proposées en fonction de la fréquence. (Multiplieurs implémentés par *DSP48E*).

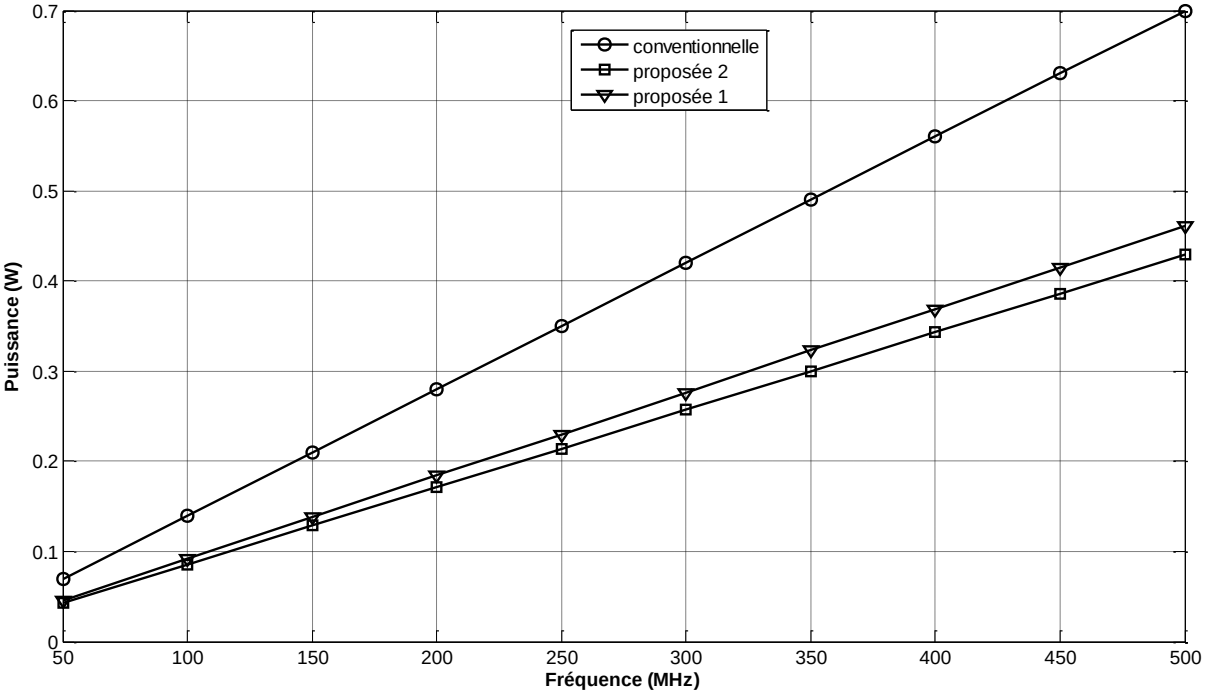

Figure 2.16 Puissance dynamique consommée par les architectures proposées en fonction de la fréquence. (Multiplieurs implémentés par *LUT*s).

Les deux figures montrent que la puissance consommée varie linéairement avec la fréquence, ce qui est en cohérence avec la théorie. La figure 2.15 indique que si les multiplieurs sont réalisés à l'aide de *DSP48E*, la première architecture proposée consomme moins de puissance, suivie de la deuxième architecture proposée puis de l'architecture conventionnelle. L'écart de puissance pour les trois architectures n'est pas important parce que les slices *DSP48E* sont optimisés pour une faible consommation de puissance. Pour une fréquence de *500 MHz,* l'écart de puissance entre la première et la deuxième architecture est égal à 36 *mW,* et celui entre la deuxième architecture et l'architecture conventionnelle est égal à 29 mW. A partir de la figure 2.16, on peut constater qu'en cas d'utilisation de *LUT*s pour l'implémentation des multiplieurs, les deux architectures proposées permettent de réduire considérablement la puissance dynamique consommée, en comparaison avec l'architecture conventionnelle. L'écart entre la consommation de la deuxième architecture proposée et l'architecture conventionnelle atteint 271 mW à la fréquence 500 *MHz,* ce qui correspond à une réduction de presque 40%.

2.6. Conclusion

Dans ce deuxième chapitre, nous avons mené une étude théorique sur le codage spatio-temporel, suivie de la présentation de deux architectures proposées pour l'implémentation du décodeur d'Alamouti sur un circuit de type *FPGA*. Le codage spatio-temporel est une stratégie qui exploite la diversité spatiale, qu'offrent les systèmes *MIMO,* pour améliorer les performances du système de communication radio mobile caractérisé par un canal à évanouissement. Le codage espace-temps en bloc orthogonal *OSTBC* est une classe importante des codes *STBC,* car elle assure un décodage simple du fait de l'orthogonalité entre les différentes colonnes de sa matrice génératrice de code. Pour la famille des codes OSTBC, le code d'Alamouti s'inscrit au sommet des codes les plus utilisés grâce à sa diversité maximale et à son rendement unitaire. Pour les canaux sélectifs en fréquence, et afin de bénéficier du codage espace-temps il est nécessaire de le combiner avec la modulation *OFDM* pour rendre les canaux non-sélectifs en fréquence. On trouve ainsi plusieurs combinaisons comme *STC-OFDM*, *STC-CDMA*, *STC-MC-CDMA*….

Dans la dernière partie du présent chapitre, nous avons présenté les deux architectures que nous avons proposées pour l'implémentation efficace et à haut débit du décodeur d'Alamouti. Les résultats de simulations montrent qu'avec un bon choix de la longueur de mot utilisé pour la représentation des données, il est possible d'assurer des performances en *BER* très proches de celles obtenues par la représentation en virgule flottante, tout en minimisant le nombre de ressources utilisées. Les résultats d'implémentation des deux architectures sur un circuit *FPGA* de type Virtex-7 montrent que les implémentations à haute fréquence peuvent atteindre un débit très élevé avec une efficacité importante en puissance et en ressources. La comparaison des deux architectures proposées avec l'architecture conventionnelle montre la supériorité des architectures proposées en termes de ressources utilisées et de la puissance consommée.

Chapitre 3

TRANSFORMEE DE FOURIER RAPIDE (FFT) : ALGORITHMES ET ARCHITECTURES D'IMPLEMENTATION

3.1. Introduction

La transformée de Fourier discrètes « Discret Fourier Transfom » *DFT* est considérée parmi les transformées discrètes les plus utilisées dans le domaine du traitement numérique du signal [63]. La transformée de Fourier rapide « Fast Fourier Transform » *FFT* est une procédure qui permet l'implémentation efficace de la *DFT*. Le développement de la *FFT* a été à l'origine des travaux de Cooley–Tukey [19], puis il a été suivi par plusieurs modifications et améliorations par d'autres chercheurs. Aujourd'hui, de nombreuses compagnies offrent des programmes et des modules personnalisés pour l'implémentation de l'algorithme *FFT* sur différents types de plateformes (*DSP*, microprocesseurs, microcontrôleur, *FPGA*, *ASIC*…).

Dans ce chapitre, nous menons une étude sur la *FFT* du point de vue algorithmique et surtout du point de vue architectures de réalisation. Nous commençons par un rappel sur le principe et la complexité arithmétique des algorithmes *FFT* Radix-2 et Radix-4. Ensuite, nous abordons les architectures d'implémentation des algorithmes *FFT,* en se concentrant sur les architectures en pipeline qui sont les plus adaptées aux applications en temps réel et à haut débit. Nous finissons le présent chapitre par l'état de l'art des architectures proposées spécialement pour les systèmes de communication *MIMO-OFDM*.

3.2. Rappel sur les algorithmes *FFT*

L'expression de la *DFT* d'une séquence de *N* points est donnée par :

$$X(k) = \sum_{n=0}^{N-1} x(n).W_N^{nk} \tag{3.1}$$

Où *x(n), n=0,1,…..N-1,* représente la séquence dans le domaine temporelle uniformément échantillonnée avec une période d'échantillonnage *T, X(k), k=0,1,…..N-1,* représente la séquence dans le domaine fréquentielle et W_N^{nk} représentent les coefficients de rotation « twiddle factors » qui sont donnés par :

$$W_N^{nk} = \exp(\frac{-j2\pi nk}{N}) \tag{3.2}$$

Le passage du domaine fréquentiel au domaine temporel est effectué par la transformée de Fourier inverse « Inverse Discret Fourier Tansform » *IDFT,* dont l'expression est donnée par :

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).W_N^{-nk} \tag{3.3}$$

Le calcul de la *DFT* par l'utilisation directe de l'expression de l'équation (3.1) nécessite un nombre d'opérations arithmétiques complexes proportionnel à N^2 (complexité d'ordre $O(N^2)$). Cependant, cette complexité arithmétique peut être réduite significativement par l'utilisation des algorithmes efficaces de calcul. Ces algorithmes sont communément nommés *FFT* « Fast Fourier Transform ». En plus de la réduction de la complexité arithmétique, les algorithmes *FFT* permettent aussi la réduction de l'espace mémoire nécessaire, et la réduction de l'erreur de quantification qui est dûe à l'utilisation d'un nombre limité de bits pour la représentation des données. Ces propriétés ont rendu possible l'implémentation de la *DFT* sur des circuits numériques (*DSP, FPGA, ASIC…).*

3.2.1. L'algorithme *FFT* Radix-2 *DIT* :

L'algorithme *FFT* Radix-2 est un algorithme qui possède la structure papillon « Butterfly » la plus simple pour le calcul de la *DFT*. Cet algorithme existe en deux versions : une version dite *DIT* « Decimation In Time », utilisant la décimation en temps, et une autre version appelée *DIF* « Decimation In Frequency », utilisant la décimation en fréquence. L'algorithme Radix-2 *DIT* est basé sur la décomposition de la séquence de *N* points ; avec $N = 2^{n_s}$ *;* en deux sous-séquences de *N/2* points chacune, une sous-séquence pour les points pairs *x(2.r),* et une autre sous-séquence pour les points impairs *x(2.r+1)*. La *DFT* de la séquence de *N* points peut être obtenue à partir des *DFT*s des deux sous-séquences *x(2.r)* et *x(2.r+1)*. L'opération de la décomposition est répétée d'une façon récursive jusqu'à obtenir des sous-séquences à deux points uniquement, comme illustré dans la figure 3.1.

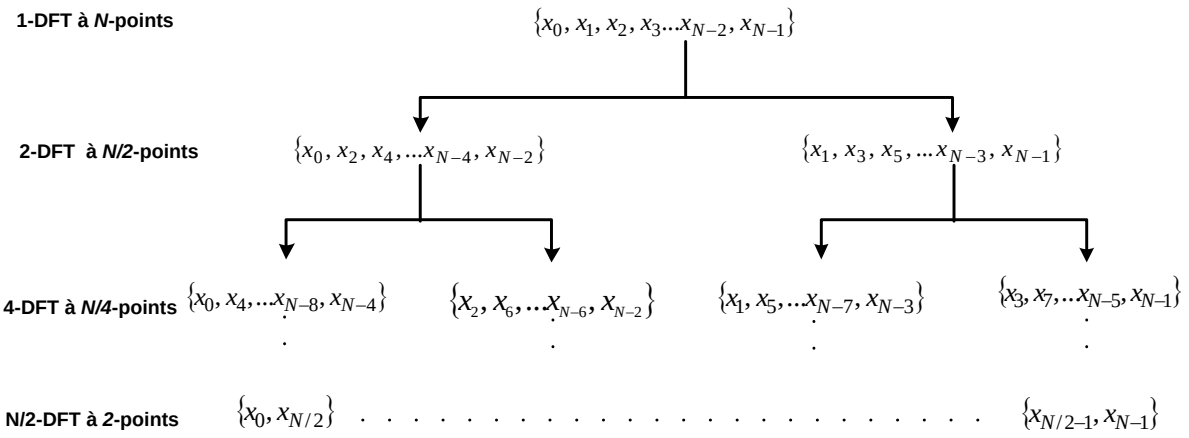

Figure 3.1 : Principe de l'algorithme *FFT* Radix-2 *DIT.*

En décomposant la séquence *x(n)* en deux sous-séquences, pair *x(2.r)* et impair *x(2.r+1),* on peut réécrire l'équation (3.1) comme suivant :

$$X(k) = \sum_{r=0}^{N/2-1} x(2r).W_N^{2.r.k} + \sum_{r=0}^{N/2-1} x(2r+1).W_N^{(2.r+1).k} \qquad (3.4)$$

En utilisant les propriétés de la *DFT* [63], l'expression de *X(k)* peut être donnée comme suivant :

$$X(k) = G(k) + H(k).W_N^k \quad \text{pour } k = 0,1\ldots(N/2)-1 \tag{3.5}$$

$$X(k+N/2) = G(k) - H(k).W_N^k \quad \text{pour } k = 0,1\ldots(N/2)-1 \tag{3.6}$$

Avec :

$$G(k) = \sum_{r=0}^{N/2-1} x(2r).W_{N/2}^{r.k} \tag{3.7}$$

$$H(k) = \sum_{r=0}^{N/2-1} x(2r+1).W_{N/2}^{r.k} \tag{3.8}$$

G(k) est la *DFT* à *N/2* points de la séquence pair *x(2.r),* alors que *H(k)* est la *DFT* à *N/2* points de la séquence impair *x(2.r+1).*

Chacune des *DFTs G(k)* et *H(k)* nécessite $(N/2)^2$ additions complexes et $(N/2)^2$ multiplications complexes, tandis que le calcul de *X(k)* par l'expression (3.1) nécessite N^2 additions complexes et N^2 multiplications complexes. Ce qui signifie que l'utilisation des équations (3.5) et (3.6) au lieu de l'équation (3.1) pour le calcul de *X(k)* réduit considérablement la complexité arithmétique. A noter que *G(k)* et *H(k)* peuvent être obtenue par une nouvelle décomposition de chacune des deux sous-séquences *x(2.r)* et *x(2.r+1)* en deux sous-séquences. La décomposition est répétée d'une manière récursive jusqu'à obtenir des sous-séquences à deux points uniquement. Cette opération est illustrée en utilisant un graphe *SFG* (Signal Flow Graph). A titre d'exemple, le *SFG* de l'algorithme *FFT* Radix-2 *DIT* à 8 points est donné dans la figure 3.2. Chaque étape de décomposition en sous-séquences est représentée par un étage dans le *SFG*. Le nombre d'étapes de décomposition nécessaire pour arriver à des sous-séquences de deux points uniquement est donnée par :

$$n_s = \log_2(N) \tag{3.9}$$

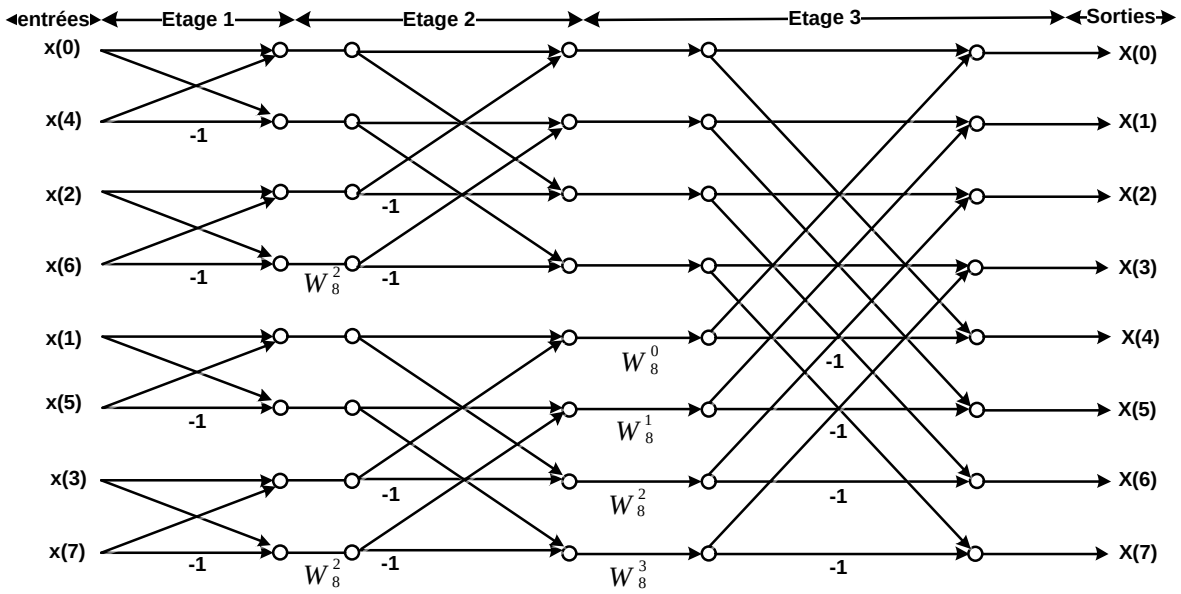

Figure 3.2 : Le *SFG* de l'algorithme *FFT* Radix-2 *DIT* pour *N=8.*

La figure 3.2 montre que les sorties *X(k)* représentant la séquence fréquentielle sont dans l'ordre naturel alors que les entrées représentant la séquence temporelle sont dans l'ordre renversé « bit reversed order », d'où vienne l'appellation *DIT* « Decimation In Time ».

3.2.2. L'algorithme *FFT* Radix-2 *DIF*

L'algorithme *FFT* Radix-2 *DIF* est basé sur la décomposition de la séquence initiale *x(n)* en deux sous-séquences. La première sous-séquence concerne la première moitié de la séquence initiale (*n=0,1…(N/2)-1),* tandis que la seconde sous-séquence concerne la deuxième moitié de la séquence initiale (*n=N/2, (N/2)+1…N-1).* Suivant cette décomposition, l'expression de la *DFT* peut s'écrire :

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n).W_N^{n.k} + \sum_{n=N/2}^{N-1} x(n).W_N^{n.k} \qquad (3.10)$$

Après plusieurs étapes de calcul, l'expression de la *DFT* peut être donnée par [62]:

$$X(2.r) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x(n+N/2)\right].W_{N/2}^{r.k} \quad pour \quad \mathrm{r} = 0,1,\ldots(\mathrm{N/2})\text{-}1 \qquad (3.11)$$

$$X(2.r+1)=\sum_{n=0}^{(N/2)-1}\left[x(n)-x(n+N/2)\right].W_N^n.W_{N/2}^{r.k} \quad pour \ \ \mathrm{r}=0,1,...(\mathrm{N/2})\text{-}1 \quad (3.12)$$

X(2.r) est la *DFT* des *N/2* points *[x(n)+x(n+N/2)]* et *X(2.r+1)* est la *DFT* des *N/2* points $\left[x(n)-x(n+N/2)\right].W_N^n$

Comme pour l'algorithme *DIT*, l'opération de la décomposition est répétée jusqu'à l'obtention des *DFT*s à deux points uniquement. Le *SFG* de l'algorithme *FFT* Radix-2 *DIF* pour *N=8* est donné dans la figure 3.3.

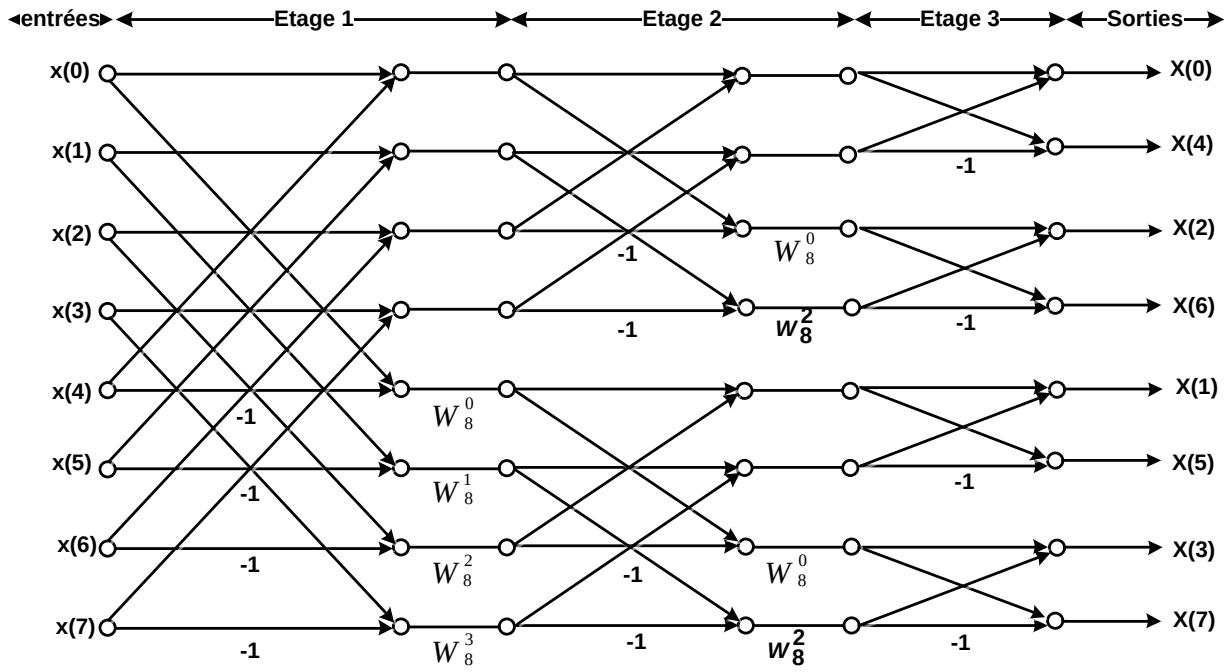

Figure 3.3 : Le *SFG* de l'algorithme *FFT* Radix-2 *DIF* pour *N=8.*

Pour l'algorithme Radix-2 *DIF,* on remarque que les entrées sont dans l'ordre naturel alors que les sorties sont dans l'ordre renversé. Il important de mentionner que la complexité arithmétique de l'algorithme *DIF* est la même que celle de l'algorithme *DIT*.

3.2.3. L'algorithme *FFT* Radix-4 *DIT*

Lorsque la longueur de la séquence d'entrée est une puissance de 4 ($N = 4^{n_s}$), la *DFT* peut être calculée en utilisant un algorithme *FFT* Radix-4. Pour l'algorithme Radix-4 *DIT*, la séquence initiale est divisée en quatre sous-séquences *x(4.n), x(4.n+1), x(4.n+2)* et *x(4.n+3).* Cette décomposition permet de remplacer l'équation (3.1) par les quatre expressions suivantes [63] :

$$X(k) = A(k) + B(k).W_N^k + C(k).W_N^{2.k} + D(k).W_N^{3.k} \quad (3.13)$$

$$X(k + N/4) = A(k) - j.B(k).W_N^k - C(k).W_N^{2.k} + j.D(k).W_N^{3.k} \quad (3.14)$$

$$X(k + N/2) = A(k) - B(k).W_N^k + C(k).W_N^{2.k} - D(k).W_N^{3.k} \quad (3.15)$$

$$X(k + 3.N/4) = A(k) + j.B(k).W_N^k - C(k).W_N^{2.k} - j.D(k).W_N^{3.k} \quad (3.16)$$

Avec *A(k), B(k), C(k)* et *D(k)* pour *k=0,1…(N/4-1),* sont les *DFT*s à *N/4* points des séquences *x(4.n), x(4.n+1), x(4.n+2)* et *x(4.n+3)* respectivement. Les équations (3.13) à (3.16) montrent que la *DFT* d'une séquence de *N* points peut être obtenue en utilisant quatre *DFT*s de *N/4* points chacune. Le processus de la décomposition est répété d'une manière récursive jusqu'à l'obtention de *DFT*s à quatre points uniquement.

3.2.4 L'algorithme *FFT* Radix-4 *DIF*

Pour l'algorithme Radix-4 *DIF,* la séquence initiale *x(n)* est divisée en quatre sous-séquences dont la première sous-séquence représente le premier quart de la séquence initiale (*n=0 à (N/4)-1)*, la deuxième sous-séquence est le deuxième quart de la séquence initiale (*n=N/4 à (N/2)-1)* et ainsi de suite. Suivant cette décomposition l'expression (3.1) devient :

$$X(k) = \sum_{n=0}^{(N/4)-1} x(n).W_N^{n.k} + \sum_{n=N/4}^{(N/2)-1} x(n).W_N^{n.k} + \sum_{n=N/2}^{(3N/4)-1} x(n).W_N^{n.k} + \sum_{n=3N/4}^{N-1} x(n).W_N^{n.k} \quad (3.17)$$

Après plusieurs étapes de calcul, les expressions de la *DFT* sont données par [63]:

$$X(4.r) = \sum_{n=0}^{(N/4)-1} \left[x(n) + x(n+N/4) + x(n+N/2) + x(n+3.N/4)\right].W_{N/4}^{n.r} \quad (3.18)$$

$$X(4.r+1) = \sum_{n=0}^{(N/4)-1} \left(\left[x(n) - j.x(n+N/4) - x(n+N/2) + j.x(n+3.N/4)\right].W_N^n\right).W_{N/4}^{n.r} \quad (3.19)$$

$$X(4.r+2) = \sum_{n=0}^{(N/4)-1} \left(\left[x(n) - x(n+N/4) + x(n+N/2) - x(n+3.N/4)\right].W_N^{2n}\right).W_{N/4}^{n.r} \quad (3.20)$$

$$X(4.r+3) = \sum_{n=0}^{(N/4)-1} \left(\left[x(n) + j.x(n+N/4) - x(n+N/2) - j.x(n+3.N/4)\right].W_N^{3n}\right).W_{N/4}^{n.r} \quad (3.21)$$

Avec *r=0,1…(N/4)-1.*

Les équations (3.18) à (3.21) montrent que la *DFT* à *N* points peut être calculée à partir de la *DFT* de *N/4* combinaisons des points *x(n), x(n+N/4), x(n+N/2)* et *x(n+3.N/4)* pour *n=0,1…(N/4)-1*. D'une façon similaire à l'algorithme *DIT,* la décomposition est répétée d'une manière récursive jusqu'à arriver à des *DFT*s à quatre entrées dont chaque entrée est une combinaison de quatre points.

3.2.5. Autres algorithmes *FFT*

Il existe dans la littérature d'autres algorithmes *FFT*, comme les algorithmes de Radix supérieure (Radix-8, Radix-16…). Il est possible aussi d'utiliser un mélange de plusieurs Radix pour le calcul d'une *DFT*. Dans ce cas on parle de Radix-Mixé « Mixed Radix » [64-65]. Un autre groupe d'algorithmes communément nommé Radix-2^i a été développés par les chercheurs. Ces algorithmes sont basés sur la décomposition de la structure de base des Radix-4, 8,16 en plusieurs modules papillon « butterfly » similaires au Radix-2. Les algorithmes les plus utilisés sont Radix-2^2, Radix-2^3 et Radix-2^4. Les deux Algorithmes Radix-2^2 et Radix-2^3 seront étudiés en détail dans le chapitre 4.

3.2.6. Utilisation de la décomposition de l'indice en facteurs premiers

La décomposition des deux indice *n* et *k* en facteurs premiers est une autre façon pour exprimer mathématiquement les algorithmes *FFT*. L'idée de la décomposition des indices est basée sur la division d'un problème de calcul de grande complexité en plusieurs sous-problèmes de complexité réduite. Pour simplifier, nous allons présenter uniquement le cas de la décomposition des indices en deux facteurs [66] :

L'indice de temps *n* est mappé suivant l'expression :

$$n =< A.n_1 + B.n_2 >_N \text{ avec } \begin{cases} 0 \le n_1 \le N_1 \\ 0 \le n_2 \le N_2 \end{cases} \quad (3.22)$$

Où *A* et *B* sont deux constantes de type entier qu'on doit déterminer , $N=N_1.N_2$ et $<.>_N$ désigne l'opérateur *modulo N.*

En utilisant cette représentation, la séquence d'entrée *x(n)* est réorganisée selon la forme suivante :

$$\left[x(0), x(1), \ldots x(N-1)\right] = \begin{bmatrix} x(0,0) & x(0,1) & \cdot & \cdot & \cdot & x(0,N_2-1) \\ x(1,0) & x(1,1) & \cdot & \cdot & \cdot & x(1,N_2-1) \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & & \cdot & \\ x(N_1-1,0) & x(N_1-1,1) & & & & x(N_1-1,N_2-1) \end{bmatrix} \quad (3.23)$$

L'indice de la fréquence *k* est mappé de la façon suivante :

$$k =< C.k_1 + D.k_2 >_N \text{ avec } \begin{cases} 0 \le k_1 \le N_1 \\ 0 \le k_2 \le N_2 \end{cases} \quad (3.24)$$

Selon le choix des entiers N_1 et N_2, il existe deux types d'algorithmes *FFT*. Si gcd(N_1, N_2)>1, les algorithmes sont appelés « Common Factor Algorithms » *CFA,* dans le cas contraire les algorithmes sont appelés « Prime Factor Algorithms » *PFA.*

L'algorithme *FFT* de Cooley et Tukey [19], qui est le plus universel des Algorithmes *FFT* utilise le « mapping » suivant :

$$n =< N_2.n_1 + n_2 >_N \quad \text{avec} \qquad \begin{cases} 0 \le \mathrm{n}_1 \le N_1 \\ 0 \le \mathrm{n}_2 \le N_2 \end{cases} \tag{3.25}$$

Et :

$$k =< k_1 + N_1.n_2 >_N \quad \text{avec} \qquad \begin{cases} 0 \le \mathrm{k}_1 \le N_1 \\ 0 \le \mathrm{k}_2 \le N_2 \end{cases} \tag{3.26}$$

En remplaçant les indices *n* et *k* par les expressions (3.25) et (3.26), l'expression de la *DFT* de l'équation 3.1 peut être exprimée comme suivant [66]:

$$X(k_1, k_2) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2.k_2} \left(\underbrace{W_N^{n_2.k_1} \underbrace{\sum_{n_1=0}^{N_1-1} x(n_1, n_1).W_{N_1}^{n_1.k_1}}_{DFT \text{ à } \mathrm{N}_1 \text{ points}}}_{\bar{\mathrm{x}}(\mathrm{n}_2, \mathrm{k}_1)} \right) \tag{3.27}$$

D'où :

$$X(k_1, k_2) = \sum_{n_2=0}^{N_2-1} \underbrace{W_{N_2}^{n_2.k_2}.\bar{\mathrm{x}}(\mathrm{n}_2, \mathrm{k}_1)}_{DFT \text{ à } \mathrm{N}_2 \text{ points}} \tag{3.28}$$

Selon les deux équations (3.27) et (3.28), pour calculer la *DFT* à *N* points il faut d'abord calculer les N_2 *DFT*s à N_1 points chacune. Ces *DFTs* intermédiaires sont notées $\bar{x}(n_2, k_1)$ dans l'équation (3.28). Les *DFTs* intermédiaires sont ensuite multipliées par les coefficients $W_N^{n_2.k_1}$. Finalement, la *DFT* finale est calculée à partir des N_2 DFTs intermédiaires.

Notons qu'il existe dans la littérature d'autres algorithmes utilisant d'autres méthodes de « mapping », comme l'algorithme de Good [67], Thomas [68] et Winograd [69].

3.2.7. Complexité arithmétique des algorithmes *FFT*

L'implémentation d'un algorithme *FFT* sur un processeur numérique, que ce soit logiciel (*DSP*, Microprocesseur, Microcontrôleur) ou matériel (*FPGA*, *ASIC*…) dépend

en grande partie de la complexité arithmétique de l'algorithme. Par complexité arithmétique, on désigne le nombre d'opérations d'additions et de multiplications nécessaires pour achever le calcul de la *DFT*. A noter que les multiplications triviales par *+j, -j, +1*, et *-1* ne sont pas prises en compte, puisqu'elles sont réalisées par de simples opérations de négation et de permutation entre la partie réelle et la partie imaginaire des données. Notant aussi que la complexité arithmétique est la même pour les deux versions duales *DIF* et *DIT*.

La réalisation d'une multiplication complexe peut être effectuée soit d'une façon classique en utilisant quatre multiplications réelles et deux additions réelles (*4M2A*), soit d'une façon à optimiser les ressources en utilisant trois multiplications réelles et trois additions réelles (*3M3A*) [66]. Les expressions donnant le nombre d'additions réelles et le nombre de multiplications réelles pour les deux cas *4M2A* et *3M3A* sont données dans les tableaux 3.1 et 3.2. [65], [70].

Tableau 3.1 : Complexité arithmétiques de quelques algorithmes *FFT* pour les multiplications *4M2A.*

Algorithmes	Nombre de multiplications réelles non triviales	Nombre d'additions réelles
Radix-2	$2.N.\log_2(N) - 7.N + 12$	$3.N.\log_2(N) - 3.N + 4$
Radix-4, Radix-2^2	$(3.N/2).\log_2(N) - 5.N + 8$	$(11.N/4).\log_2(N) - 26.N/12 + 8/3$
Radix-8, Radix-2^3	$(4.N/3).\log_2(N) - 59.N/14 + 40/7$	$(33.N/12).\log_2(N) - 57.N/28 + 16/7$

Les deux tableaux 3.1 et 3.2 montrent que les deux algorithmes Radix-8 et Radix-2^3 sont les meilleurs en matière de la complexité arithmétique, suivis des deux algorithmes Radix-4 et Radix-2^2. A noter que plus le Radix de l'algorithme est élevé plus son implémentation sur un processeur nécessite un processus de contrôle

compliqué. Donc, bien que la complexité arithmétique est un paramètre important, elle n'est pas le seul paramètre pour l'évaluation de l'efficacité d'un l'algorithme *FFT*.

Tableau 3.2 : Complexité arithmétiques de quelques algorithmes *FFT* pour les multiplications *3M3A*.

Algorithmes	Nombre de multiplications réelles non triviales	Nombre d'additions réelles
Radix-2	$(3.N/2).\log_2(N)-5.N+8$	$(7.N/2).\log_2(N)-5.N+8$
Radix-4, Radix-2^2	$(9.N/8).\log_2(N)-43.N/12+16/3$	$(25.N/8).\log_2(N)-43.N/12+16/3$
Radix-8, Radix-2^3	$(25.N/24).\log_2(N)-350.N/112+4$	$(146.N/48).\log_2(N)-25.N/8+4$

3.3. Les architectures classiques d'implémentation de la *FFT*.

L'architecture d'implémentation d'un l'algorithme *FFT* offre une idée plus détaillée sur la façon de la réalisation des différentes opérations, en utilisant les différentes ressources matérielles du circuit électronique qui héberge l'algorithme. En d'autres termes, l'architecture est une présentation détaillée du circuit qui permet la réalisation matérielle de l'algorithme *FFT*. De ce fait, l'architecture d'implémentation est un aspect très important pour la détermination des performances d'un algorithme *FFT* et des ressources matérielles qu'il utilise. En effet, le même algorithme peut être implémenté par plusieurs architectures différentes, chaque architecture est caractérisée par les ressources matérielles utilisées, la puissance consommée, et le débit de traitement des données achevé.

Vu l'importance des architectures d'implémentation des algorithmes *FFT*, ce domaine de recherche est l'un des domaines les plus actifs et les plus productifs.

Dans ce contexte, diverses architectures ont été proposées pour les différents domaines d'application de la *DFT*. Ces architectures peuvent généralement être classées en deux grandes catégories, qui sont les architectures à base de mémoire et les architectures en pipeline.

3.3.1. Architectures en pipeline :

Comme son nom l'indique, l'architecture en pipeline est constituée de plusieurs étages de calcul en pipeline, dont le nombre d'étages de calcul dépend de la taille de la *DFT* et du Radix de l'algorithme *FFT*. La figure 3.4 présente le schéma bloc des architectures en pipeline. Chaque étage est constitué généralement d'un module de calcul *BPE* « Butterfly Processor Element » qui réalise les opérations du papillon, d'un module de contrôle qui contrôle l'acheminement des échantillons entre les différents éléments de l'étage, et des éléments de mémorisation.

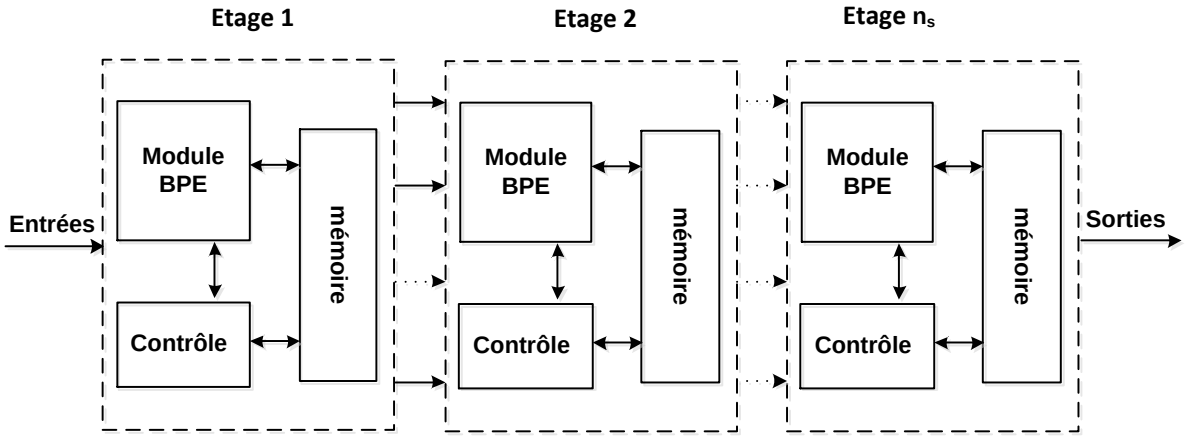

Figure 3.4 : Schéma bloc des architectures *FFT* en pipeline.

Les éléments de mémorisation servent à stocker les valeurs des coefficients de rotation, et à réaliser la synchronisation entre les données à travers l'introduction des retards « delay » appropriés à chaque étage de traitement. Pour les architectures en pipeline, les données sont traitées séquentiellement et d'une façon continue. Ce type d'architectures est le plus adapté aux opérations de traitement numérique du signal

nécessitant un débit élevé et un traitement en temps réel, comme les systèmes de télécommunications à haut débit.

Les différentes architectures en pipeline sont classifiées selon deux caractéristiques qui sont :

- Le nombre de chemins utilisés pour le traitement des données : on trouve des architectures utilisant un chemin unique « *single path* », et des architectures utilisant plusieurs chemins « *mutliple path* ».
- La méthode ou la stratégie suivie pour créer les différents retards nécessaires pour la synchronisation et à l'ordonnancement des échantillons : on trouve ainsi les architectures utilisant les commutateurs et les retards « Delay Commutator », et les architectures utilisant les retards et le retour « Delay Feedback ».

Selon ces deux caractéristiques, les architectures en pipeline sont en général classées en quatre catégories : *SDF* « *Single-path Delay Feedback* », *MDF* « *Multi-path Delay Feedback* », *SDC* « *Single-path Delay commutator »* et *MDC* « *Multi-path Delay commutator ».*

3.3.1.a. L'architecture *SDF*

L'architecture *SDF* utilise des registres à décalage pour réaliser le retour d'une partie des entrées de l'étage et pour le stockage d'une partie des sorties du module de calcul *BPE*. A cause de l'utilisation d'un chemin unique de traitement, l'approche *SDF* ne nécessite qu'un seul multiplieur à la sortie de chaque étage. En outre, les registres à décalage sont utilisés pour réaliser le retour d'une partie des entrées de l'étage durant la première moitié du temps de traitement, puis les mêmes registres sont utilisés pour le stockage d'une partie des sorties de *BPE* durant la deuxième moitié du temps de calcul. Par conséquent, l'architecture *SDF* utilise deux fois moins de registres en comparaison avec l'architecture *MDC.* Donc, les registres sont utilisés d'une façon optimale avec un taux d'utilisation de 100%. Le taux d'utilisation des additionneurs et des multiplieurs est de 50% pour les architectures *SDF* classique.

Suivant le Radix utilisé, plusieurs versions de l'architecture *SDF* ont été proposées. On trouve ainsi des architectures à base du Radix-2 appelées *R2SDF* [72], des architectures *R4SDF* à base de Radix-4 [73], des architectures à base de de Radix-2^2 appelées R2^2SDF [74], [75]. La figure 3.5 donne un exemple de l'architecture *R2SDF* pour un algorithme *FFT* Radix-2 de taille *N=8.*

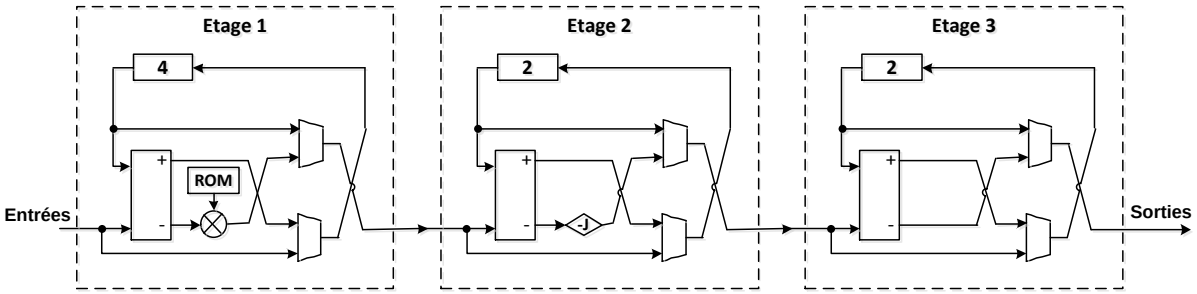

Figure 3.5 : Schéma de l'architecture *SDF* pour un algorithme *FFT* Radix-2 de taille 8 (*R2SDF, N=8*).

L'architecture est constituée de trois étages, chaque étage contient : un seul registre à décalage dont sa taille est en fonction de la taille de la *DFT N,* et du numéro de l'étage, une unité d'addition et de soustraction complexe et deux multiplexeurs à deux entrées. En outre, le premier étage utilise une mémoire à lecture seule *ROM* « Read Only Memory » pour stocker les valeurs des coefficients de rotation, et un multiplieur complexe pour effectuer la multiplication par les coefficients de rotation.

L'extension de l'architecture de la figure 3.5 au cas des algorithmes *FFT* Radix-2 de taille *N* quelconque est effectuée en ajoutant en amont des étages identiques à celui de l'étage 1, avec le choix approprié de la taille des registres à décalage et de la taille et du contenu des *ROMs*. L'architecture *SDF* est considérée comme la plus adaptée aux systèmes *OFDM* utilisant un seul canal appelés *SISO-OFDM* du fait qu'elle nécessite les moindres ressources, et elle possède une seule entrée et une seule sortie. Pour les systèmes *MIMO-OFDM,* la méthode classique consiste à utiliser un nombre de processeurs *FFT* égale au nombre de canaux du système

MIMO-OFDM. Par exemple pour un système *MIMO-OFDM* à deux antennes en émission et deux antennes en réception (2x2), il faut utiliser deux processeurs *IFFT* en émission et deux autres processeurs *FFT* en réception. Cela signifie que la complexité augmente rapidement avec l'augmentation du nombre de canaux pour un système *MIMO-OFDM* ou *MIMO-MC-CDMA* [75].

3.3.1.b. L'architecture *MDC*

L'architecture *MDC* utilise plusieurs chemins pour le traitement parallèle des échantillons, et elle utilise des commutateurs pour la permutation et l'acheminement des échantillons d'un étage à un autre. La synchronisation entre les différents échantillons est assurée par des éléments de retard souvent implémentés par des registres à décalage. Suivant l'algorithme *FFT* Radix-r utilisé, on trouve dans la littérature plusieurs variantes de l'architecture *MDC*. On trouve par exemple l'architecture *R2MDC* qui utilise l'algorithme *FFT* Radix-2 [76]-[78], l'architecture *R4MDC* qui utilise l'algorithme Radix-4 [76], [77], [79].

Suivant l'algorithme Radix-r utilisé, l'architecture *MDC* nécessite *2.(r-1)* éléments de retard pour chaque étage, *r-1* éléments avant la commutation et *r-1* autres éléments après la commutation. Pour effectuer la multiplication par les coefficients de rotation, l'architecture MDC requiert *r-1* multiplieurs complexes pour chaque module *BPE*. L'architecture *MDC* possède l'inconvénient d'avoir un faible taux d'utilisation de ressources en comparaison avec l'approche *SDF*. Les multiplieurs et les modules papillon sont utilisés avec un taux de 50% pour l'architecture *R2MDC,* alors qu'ils sont utilisés à un taux de 25% uniquement pour le cas du *R4MDC* [73], [74].

La figure 3.6 présente l'architecture *R2MDC* pour un algorithme *FFT* de taille 8. Cette architecture permet de traiter deux échantillons par cycle horloge car elle possède deux chemins de traitement. Par conséquent, l'architecture *R2MDC* est deux fois plus rapide que l'architecture *R2SDF*. Donc, l'avantage majeur des architectures M*DC* est l'utilisation de chemins multiples pour le traitement parallèle des échantillons. De ce fait, les architectures *MDC* sont les plus adaptées aux systèmes utilisant la technique *MIMO-OFDM* ou *MIMO-MC-CDMA* [80].

Pour réduire les ressources utilisées par l'approche *MDC,* des versions plus élaborées ont été proposées par plusieurs chercheurs. A titre d'exemple, on peut citer les architectures *R2^{i}MDC* [81]-[83], qui utilisent l'algorithme *FFT* Radix-2^i, et les architectures *MRMDC,* qui utilisent un algorithme FFT avec un Radix mixé « Mixed Radix » [80].

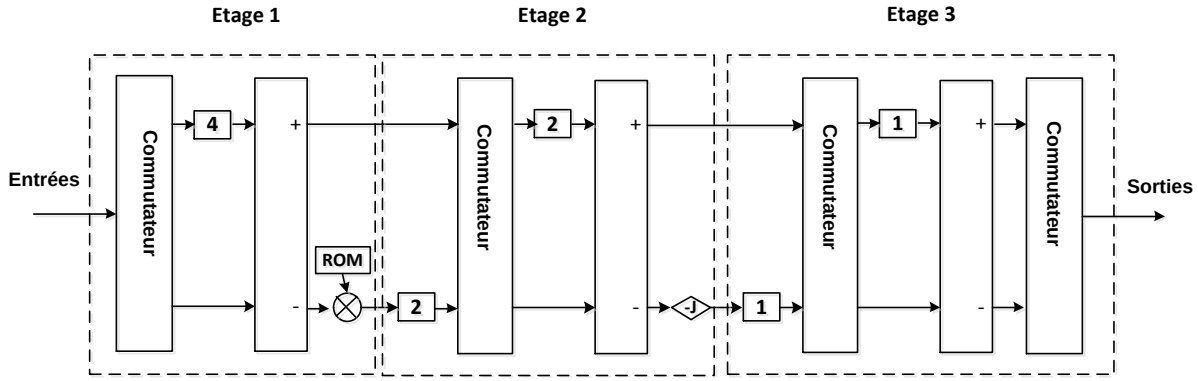

Figure 3.6 : Schéma de l'architecture *MDC* pour un algorithme *FFT* Radix-2 de taille 8 (*R2MDC, N=8*)

3.3.1.c. L'architecture *SDC*

L'architecture *SDC* utilise les commutateurs et les retards comme stratégie d'ordonnancement et de synchronisation, et un seul chemin pour le traitement des échantillons [84]-[87]. L'idée de base de l'architecture *SDC* consiste à modifier le module de calcul papillon *BPE,* afin de réduire le nombre d'opérations arithmétiques nécessaires par rapport au module *BPE* conventionnel. Cependant, la réduction du nombre d'opérations n'est pas sans prix, car elle conduit à l'augmentation de la complexité du processus de contrôle. La figure 3.7 illustre le schéma de l'architecture *SDC* pour un algorithme *FFT* Radix-4 de taille 16 (*R4SDC, N=16*). Chaque module *BPE* modifié nécessite 3 additionneurs/soustracteurs au lieu de 8 additionneurs/soustracteurs pour le module *BPE* Radix-4 conventionnel. Pour ne pas compliquer le schéma, nous n'avons pas détaillé le module de calcul modifié, le schéma détaillé avec l'explication de son principe de fonctionnement peuvent être

trouvé dans la référence [85]. La figure 3.7 montre que l'architecture *R4SDC* nécessite un seul multiplieur par étage, alors que l'architecture *R4MDC* utilise trois multiplieurs pour chaque étage. Ce qui permet d'augmenter le taux d'utilisation des multiplieurs à 75% au lieu de 25% pour l'architecture *R4MDC*.

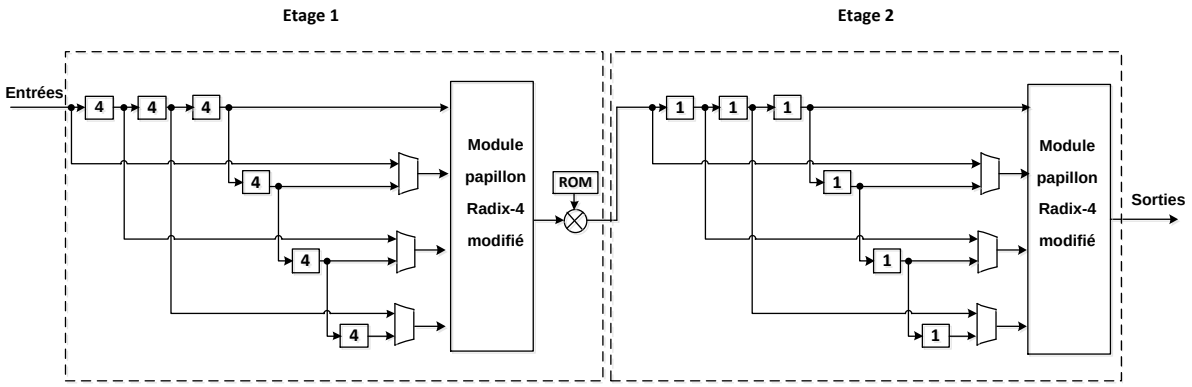

Figure 3.7 : Schéma de l'architecture *SDC* pour un algorithme *FFT* Radix-4 de taille *16* (*R4SDC, N=16*).

A cause de son chemin unique de traitement des échantillons, l'architecture *SDC* opère à un faible débit puisque elle permet le traitement d'un seul échantillon par cycle horloge. Par conséquent, cette architecture n'est pas adaptée aux systèmes de télécommunications de type *MIMO-OFDM* puisqu'elle nécessite l'utilisation de plusieurs processeurs *FFT* comme l'architecture *SDF*. Cela augmente considérablement la complexité matérielle de son intégration dans un système *MIMO-OFDM* ou *MIMO-MC-CDMA*.

3.3.1.d. Architecture *MDF*

L'architecture *MDF* [88]-[91] possède plusieurs chemins de traitement des échantillons, et elle utilise les retards et les retours comme technique de synchronisation et d'ordonnancement. Le schéma de principe de l'architecture *MDF* est donné dans la figure 3.8. Chaque étage de l'architecture contient des modules papillon, une unité de multiplication, et des éléments de retard. Pour les modules papillon, on trouve souvent plusieurs modules papillon Radix-2 en parallèle [88]-[90],

dont chaque module traite les échantillons d'un seul chemin. L'unité de multiplication contient un ou plusieurs multiplieurs complexes pour réaliser les multiplications par les coefficients de rotation pour les différents chemins. Les multiplieurs peuvent être standard ou à structure modifiée afin de réduire la complexité matérielle du circuit d'implémentation [88], [89].

Puisque l'architecture *MDF* possède plusieurs chemins de traitement, elle est utilisée dans le contexte des systèmes *MIMO-OFDM*. Cependant, on ne trouve pas dans la littérature une approche systématique pour l'obtention de l'architecture *MDF* pour les différentes valeurs de la taille *N* et les différents radix-*r*. La proposition d'une architecture avec une complexité matérielle raisonnable reste toujours un défi.

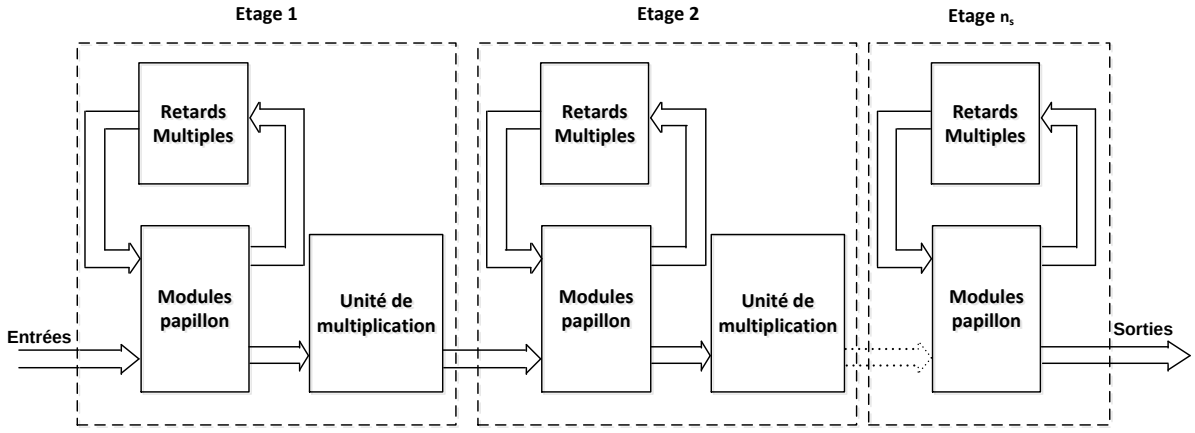

Figure 3.8 : Schéma de principe de l'architecture *MDF*

3.3.1.e. Comparaison entre les différentes architectures pipelines

Le tableau 3.3 donne une comparaison entre les différentes architectures pipelines en termes de : la complexité arithmétique (nombre d'additionneurs et nombre de multiplieurs complexes), de la taille de la mémoire utilisée comme éléments de retard, de la complexité de contrôle, et du débit. Le débit désigne ici le nombre d'échantillons traités par unité de temps. Ce débit est égal à la fréquence de fonctionnement du circuit cible d'implémentation (f_{op}) multipliée par le nombre de chemin de traitement de l'architecture en question.

Tableau 3.3 : Comparaison entre les architectures pipelines conventionnelles.

Architecture	# de multiplieurs complexes	# d'additionneurs complexes	Taille de la mémoire	Complexité de contrôle	débit
R2SDF	$2.\log_4(N)-1$	$4.\log_4(N)$	$N-1$	simple	f_{op}
R4SDF	$\log_4(N)-1$	$8.\log_4(N)$	$N-1$	moyenne	f_{op}
R2^{2}SDF	$\log_4(N)-1$	$4.\log_4(N)$	$N-1$	simple	f_{op}
R2^{3}SDF	$\approx \log_4(N)-1$	$4.\log_4(N)$	$N-1$	simple	f_{op}
R2^{4}SDF	$\leq 0.7.\log_4(N)-1$	$4.\log_4(N)$	$N-1$	simple	f_{op}
R4^{2}SDF	$\leq 0.7.\log_4(N)-1$	$4.\log_4(N)$	$N-1$	simple	f_{op}
R2MDC	$2.\log_4(N)-1$	$4.\log_4(N)$	$\frac{3.N}{2}-2$	simple	$2.f_{op}$
R4MDC	$3.(\log_4(N)-1)$	$8.\log_4(N)$	$\frac{5.N}{2}-4$	simple	$4.f_{op}$
R4SDC	$\log_4(N)-1$	$3.\log_4(N)$	$2.N-2$	complexe	f_{op}

En termes de nombre de multiplieurs, on constate que les deux architectures *R2^{4}SDF* et *R4^{2}SDF* sont les meilleures, puis viennent en deuxième lieu les architectures *R2^{3}SDF*, *R2^{2}SDF, R4SDF* et *R4SDC* avant *R2SDF* et *R2MDC,* et finalement l'architecture *R4MDC* en dernière position. En termes de nombre d'additionneurs complexes, l'architecture *R4SDC* est la moins exigeante, alors que *R4MDC* et *R4SDF* sont les plus exigeantes. Les autres architectures viennent au milieu. On précise que le nombre de multiplieurs complexes est le plus significatif par rapport au nombre d'additionneurs complexes pour l'évaluation de la complexité matérielle d'une architecture. Cela est dû au fait qu'un multiplieur complexe nécessite beaucoup plus de ressources en comparaison avec un additionneur complexe. En terme de l'espace mémoire, l'approche *SDF* nécessite moins d'espace mémoire suivie de l'architecture *R2MDC*, puis l'architecture *R4SDC,* et finalement l'architecture *R4MDC*. En matière de la complexité du processus de contrôle, le tableau 3.3 montre que le contrôle de l'architecture *R4SDC* est le plus compliqué, le contrôle de

l'architecture *R4SDF* est considéré de complexité moyenne, alors que les autres architectures possèdent un contrôle simple. Concernant le débit de traitement, on remarque que les architectures *MDC* sont les plus rapides. Cette rapidité est proportionnelle au Radix de l'algorithme *FFT* utilisé. Les autres architectures ont un débit fixe égale à la fréquence de fonctionnement du circuit cible d'implémentation.

On conclusion, chaque architecture possède des points forts et des points de faiblesse par rapport aux autres architectures. Le choix de l'architecture est une tâche très délicate qui dépend de plusieurs paramètres comme : les ressources disponibles dans le circuit cible, le temps nécessaire pour la conception, le nombre de séquences indépendantes que doit traiter le circuit, le débit que doit assurer le processeur…

3.3.2 Les architectures à base de mémoire

Le schéma de principe des architectures *FFT* à base de mémoire [92]-[95] est illustré dans la figure 3.9.

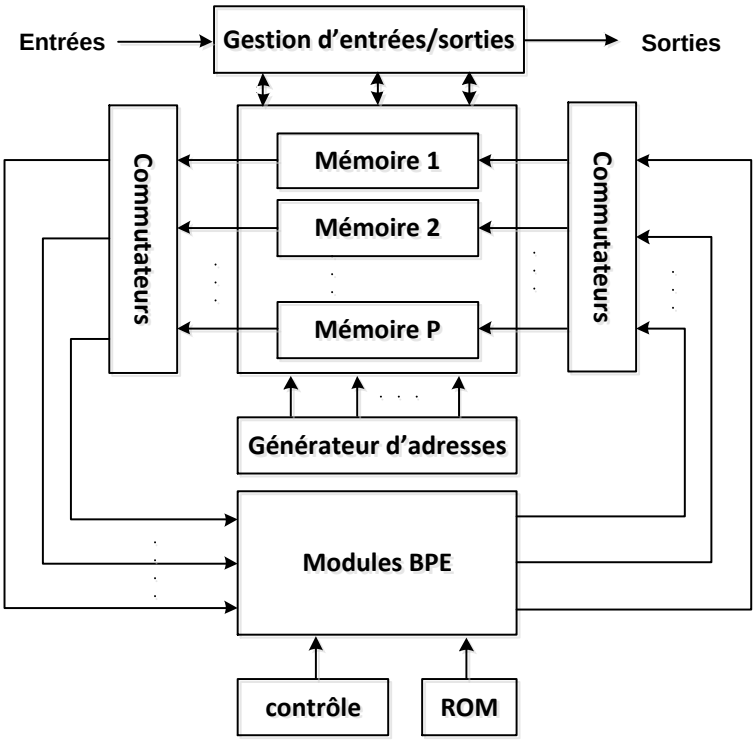

Figure 3.9 : Schéma de principe des architectures *FFT* à base de mémoire.

Ces architectures utilisent un nombre limité de modules *BPE* pour effectuer les différentes opérations d'addition, de soustraction et de multiplication que nécessite le calcul de la *TFD*. Elles utilisent aussi des mémoires pour la mémorisation des échantillons d'entrées, et des résultats de calcul intermédiaires afin de les réinjecter aux entrées des modules de calcul. Le nombre et la structure des modules papillon sont choisis selon le degré de parallélisme et la profondeur de pipeline qu'on veut obtenir. La figure 3.10 illustre quatre configurations différentes des modules papillon.

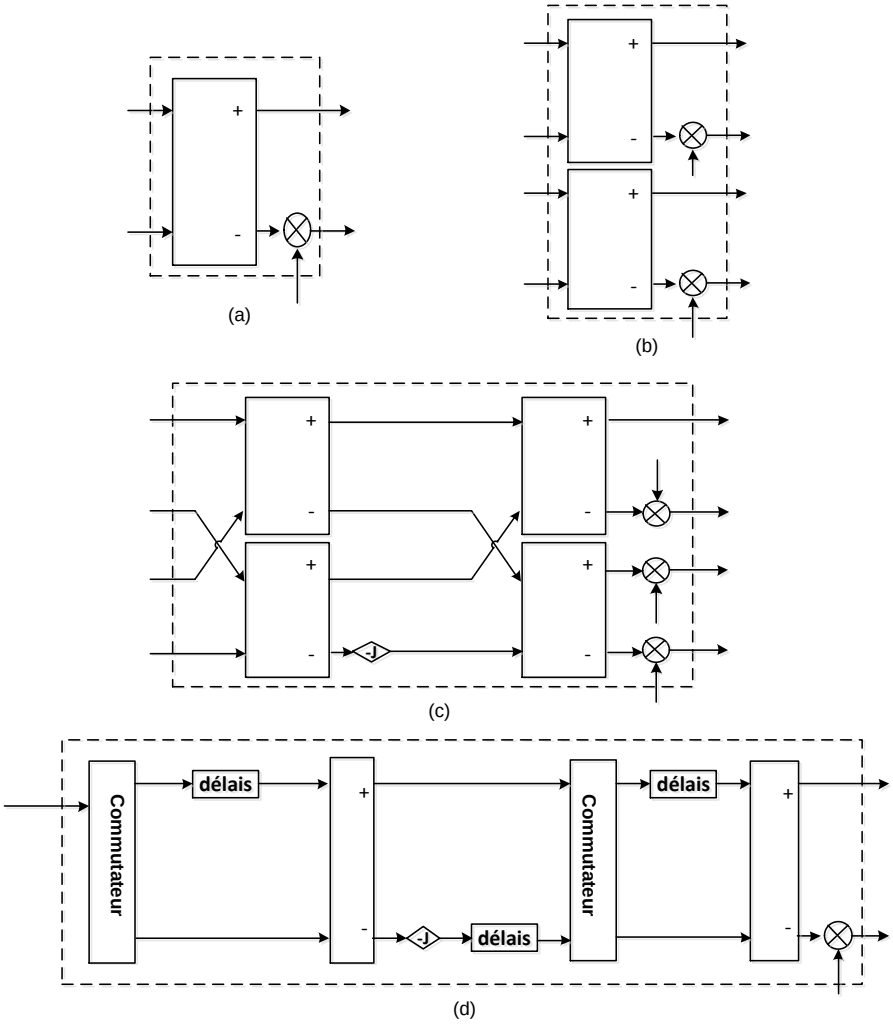

Figure 3.10 : Exemple de configuration des modules *BPE* pour les architectures à base de mémoire. (a) Un module Radix-2. (b) Deux modules Radix-2 en parallèle. (c) un module Radix-2^2 à quatre chemins parallèles. (d) : un module Radix-2^2 à deux chemins parallèles.

La première configuration (figure 3.10(a)) qui consiste en un seul module papillon Radix-2 est la plus simple mais elle nécessite le plus grand nombre de cycles horloge par échantillon, elle est caractérisée donc par un très faible débit. Pour réduire le nombre de cycles par échantillons et améliorer le débit, il faut augmenter le parallélisme des modules *BPE*. Cela est possible en utilisant plusieurs modules simples en parallèle, ou bien en utilisant des modules à base de Radix élevé. La figure 3.10(b) présente un exemple de deux modules simple Radix-2 en parallèle qui assurent un niveau de parallélisme égal à 4. L'approche de la figure 3.10(c) possède aussi un parallélisme de 4 niveaux, mais avec deux niveaux de pipeline ce qui signifie que cette configuration permet de réaliser à la fois les opérations de deux étages consécutifs. Cependant, elle nécessite plus de ressources en comparaison avec la première configuration (figure 3.10(a)). La configuration de la figure 3.10(d) est un bon compromis entre la première configuration (figure 3.10(a)) et la troisième configuration (figure 3.10(c)), car elle assure un débit meilleur que la première configuration, tout en utilisant moins de ressources que la troisième configuration. On précise que l'augmentation du niveau de parallélisme et de la profondeur de pipeline permet de réduire le nombre d'opérations de lecture et d'écritures dans les mémoires. Cette réduction permet d'une part de réduire le temps de calcul, et par conséquent d'augmenter le débit. De l'autre part, elle permet de réduire considérablement la puissance consommée par le circuit cible.

L'avantage des architectures à base de mémoire est qu'elles nécessitent un minimum de ressources en comparaison avec les architectures en pipeline. Cependant, le calcul de la *DFT* par les architectures à base de mémoire nécessite plusieurs cycles horloge pour chaque échantillon, ce qui rend le processeur de calcul très lent par rapport au type pipeline. Cela signifie que ce type d'architecture n'est pas convenable pour les applications nécessitant le traitement à haut débit.

3.4. Architectures *FFT* pour les systèmes *MIMO-OFDM*

Le module *FFT* est considéré parmi les modules les plus compliqués dans un système *MIMO-OFDM.* Ce module nécessite des ressources matérielles importantes et une architecture d'implémentation bien appropriée, afin de répondre aux

recommandations en débit, en temps de réponse et en l'efficacité d'utilisation des ressources, et de consommation de la puissance. Les systèmes *MIMO-OFDM* nécessitent la transmission simultanée de plusieurs séquences de données, ce qui signifie que le module *FFT/IFFT* doit être capable de calculer plusieurs *DFT*s à la fois. L'approche classique consiste à utiliser plusieurs processeurs *FFT* à l'émission et à la réception. Pour un système *MIMO-OFDM* à N_t antennes d'émission et N_r antennes de réception, l'approche classique nécessite N_t processeurs *IFFT* pour l'émetteur et N_r processeurs *FFT* au niveau du récepteur. Donc, la complexité matérielle de cette solution augmente considérablement avec l'augmentation du nombre d'antennes. Ce qui signifie que la solution classique n'est pas pratique pour les systèmes *MIMO-OFDM*.

Les chercheurs dans ce domaine ont proposé une multitude d'architectures *FFT* qui permettent de traiter plusieurs séquences indépendantes à la fois, mais en réduisant les ressources matérielles consommées [87]-[90], [96]-[107]. Le modèle général du processus *FFT* pour les systèmes *MIMO-OFDM* est donné dans la figure 3.11. Le processeur contient une unité d'ordonnancement et de réorganisation des entrées, plusieurs modules de traitement en pipeline, et une unité d'ordonnancement et de réorganisation des sorties.

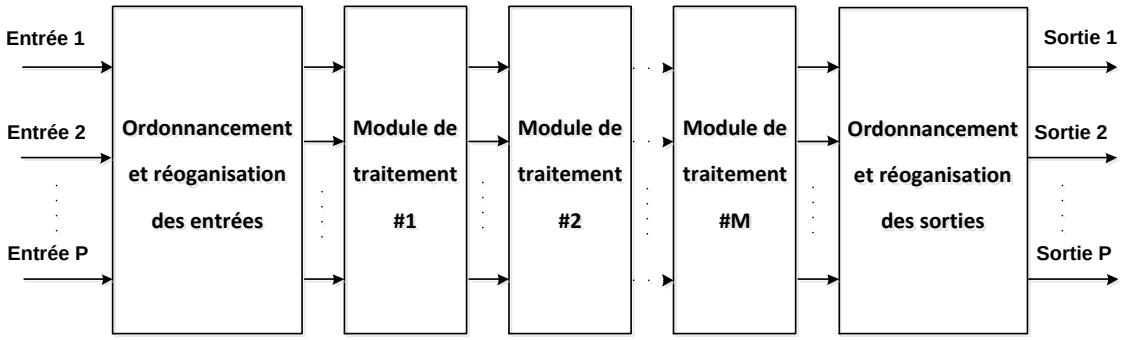

Figure 3.11 Modèle général d'un processeur FFT pour un système MIMO-OFDM

L'ordonnancement et la réorganisation des entrées sont des opérations importantes pour l'utilisation efficace des modules de traitement. Un bon

ordonnancement permet de réduire la complexité matérielle et d'augmenter le taux d'utilisation des ressources. Le but de ce module est de réorganiser les échantillons des différentes séquences d'entrée afin de permettre aux modules de traitement de fonctionner correctement, et d'une façon efficace avec un taux d'utilisation le plus élevé possible. Généralement, c'est la structure du premier module de traitement qui détermine la façon de réorganisation des différentes entrées.

Les différents modules de traitement permettent de calculer les *DFT*s des différentes séquences selon l'ordre défini par l'unité de réorganisation des entrées. Ces modules sont constitués de structures en papillon, de multiplieurs complexes, d'éléments de retard et de commutateurs. Généralement, la structure en papillon consiste en un seul papillon de Radix élevé ou une configuration parallèle/pipeline de plusieurs papillons Radix-2. L'unité de réorganisation des sorties permet de séparer les échantillons des différentes séquences d'une part, et de réordonner les échantillons de chaque séquence dans l'ordre naturel de l'autre part.

Locharla, G.R. *et al* [105] ont proposé une architecture pour le calcul des *DFT*s de 8 séquences indépendantes avec une taille *N* égale à 64, 128, 256 ou 512. L'architecture est composée d'un module d'ordonnancement et de réorganisation des entrées, qui utilise des registres de taille *7.N*, d'un autre module d'ordonnancement et réorganisation des sorties utilisant des registres de taille *2.N,* et de quatre modules de traitement. Le premier module de traitement utilise un module papillon Radix-8, 7 multiplieurs complexes, des commutateurs, et des éléments de retard. Le deuxième module est similaire au premier, sauf qu'il utilise 6 multiplieurs au lieu de 7. Le troisième module contient des modules papillon Radix-2 en parallèle, des commutateurs et 4 multiplieurs. Le dernier module de traitement est constitué uniquement de modules papillon Radix-2 en parallèle.

Le circuit proposé dans la référence [106] permet aussi le calcul des *DFTs* de 8 séquences indépendantes avec une taille *N=128*. La structure générale est similaire à celle proposée en [105], avec un module d'ordonnancement des entrées, un autre module pour l'ordonnancement des sorties, et trois modules de traitement. Le module d'ordonnancement des entrées est réalisé à l'aide d'un réseau de 8x8 bancs de

mémoires *RAM.* Le premier module de traitement contient un papillon Radix-8,7 multiplieurs complexes et une unité de commutation et de retards. Le deuxième module contient un papillon Radix-8 et 7 multiplieurs complexes, alors que le troisième module est composé de 8 papillons Radix-2 en parallèle en plus de leurs éléments de retards avec retour selon l'approche *SDF*.

L'architecture proposée en [107] présente beaucoup de similarités avec celle qui est présentée en [106]. Le schéma de base est pratiquement le même, le nombre de modules de traitement est le même, les modules papillons sont aussi les même. La différence, c'est que les auteurs de [106] utilisent un réseau de 8x8 bancs de RAM pour l'ordonnancement et la réorganisation des entrées tandis que les auteurs de [107] utilisent 8 bancs de *RAM,* et des commutateurs en amont (pre-commutateurs) et en aval (post-commutateurs) des *RAM*s.

L'architecture proposée en [102] permet de calculer les *DFT*s de 1 à 4 séquences indépendantes avec une taille *N* variable (128, 512,1024 et 2048). Elle est constituée d'une unité d'ordonnancement des entrées, de 5 modules de traitement et d'une unité pour l'ordonnancement des sorties. L'unité d'ordonnancement des entrées est réalisée à l'aide de 12 bancs de RAMs, chacun de taille N/4. Le bloc d'ordonnancement des sorties est constitué de 3 étages d'éléments de retard et de commutation. Les quatre premiers modules de traitement utilisent une structure papillon Radix-4, des commutateurs, des éléments de retard et des multiplieurs complexes. Le cinquième module de traitement contient une structure papillon reconfigurable selon la taille *N* (Radix-4 pour *N=1024* et Radix-8 pour les autres valeurs de *N*)

Le processeur *FFT* proposé dans la référence [99] est capable de calculer jusqu'à *4 DFT*s correspondant à 4 séquences indépendantes de taille 64 ou 128. Ce processeur est basé sur l'approche *MDF* et il est constitué principalement d'une unité d'ordonnancement des entrées et de trois modules de traitement. L'ordonnancement des entrées est effectué en utilisant des éléments de retard et des commutateurs. Le premier module de traitement utilise 4 papillons Radix-2 en parallèles, 4 bancs de *RAM* chacun de taille 64 pour effectuer les retards et les retours, deux multiplieurs

complexes, en plus de quelques multiplexeurs indispensables pour l'acheminement des échantillons. Le deuxième module de traitement est constitué de 3 étages de papillons en pipeline. Chaque étage contient 4 papillons Radix-2 en parallèle en plus de leurs éléments de retard et de retour. En plus, le deuxième module de traitement contient un multiplieur complexe à structure modifiée qui permet de réaliser efficacement les opérations de multiplication nécessaires. Le troisième module de traitement est constitué de trois étages papillon en pipeline. Le premier étage contient 4 papillons Radix-2 en parallèle en plus de leurs éléments de retard et retour, le deuxième étage contient deux papillons Radix-2 en parallèle, et deux multiplieurs triviaux, et le troisième étage contient seulement deux papillons Radix-2 en parallèle.

L'architecture présentée dans [103] est basée sur l'approche *MDF,* elle permet de calculer les DFTs de 8 séquences indépendantes de taille 512. Elle contient 3 modules de traitement et une unité d'ordonnancement des sorties. Le premier module de traitement est composé de 8 configurations de papillons identiques et en parallèle en plus de leurs multiplieurs complexes. Chaque configuration est une réalisation de la structure papillon du Radix-2^4 selon l'approche *SDF,* et elle contient 4 unités en cascade. Chaque unité est composée d'un papillon radix-2, des éléments de retard et retour et de trois multiplieurs par constantes. Le deuxième module de traitement réalise 8 structures papillon Radix-2^2 suivant l'approche *SDF* en parallèle, suivies de multiplieurs complexes. Le troisième module consiste en trois étages en cascade, dont chaque étage contient 4 papillons Radix-2 en parallèle sans éléments de retard. L'unité d'ordonnancement des sorties permet d'arranger les *DFT*s des 8 séquences en parallèle et dans l'ordre naturel. Le processus d'ordonnancement est assuré par des opérations de lecture et d'écriture dans une mémoire *RAM* unique de taille 512 mots avec une logique de contrôle très simple.

Fu, B., et Ampadu.P. ont proposé dans [104] un processeur *FFT* pour quatre séquences indépendantes de taille 64 ou 128, qui est constitué d'une unité d'ordonnancement des entrées et de trois modules de traitement. L'idée de base consiste à utiliser une combinaison de la technique de retard et commutateur « delay commutator » et de la technique de retard et retour « delay feedback » afin

d'optimiser les ressources utilisées. L'unité d'ordonnancement des entrées permet de transformer les entrées parallèles en séquences entrelacées pour assurer le fonctionnement correct et efficace des différents modules de traitement. Cette unité est réalisée à l'aide d'éléments de retard de tailles différentes, en plus des éléments de commutation. Le premier module de traitement, qui est utilisé uniquement pour les séquences de taille 128, est composé de 4 papillons Radix-2 en parallèle en plus de leurs éléments de retard et de retour et de 3 multiplieurs par des constantes « constant multipliers ». Le deuxième module est formé d'une structure papillon Radix-(4+2) et de 4 multiplieurs complexes. La structure papillon Radix-(4+2) est constituée d'un papillon Radix-4, suivie de trois multiplieurs triviaux, d'éléments de retard, de commutateurs, et de deux papillons Radix-2 en parallèle. Le troisième module est principalement constitué d'une structure papillon Radix-(4+2).

3.5. Conclusion

Dans ce chapitre, nous avons mené une étude sur les algorithmes *FFT* et les architectures de leurs implémentations sur des circuits numériques de type *FPGA* ou *ASIC*. Les chercheurs ont proposé plusieurs variantes pour l'algorithme *FFT.* Chaque variante est caractérisée principalement par sa complexité arithmétique, qui est exprimée en nombre d'opérations d'addition et de multiplications nécessaire, par le nombre d'échantillons traités par cycle horloge, et par le degré de complexité du processus de contrôle. Ces caractéristiques sont utilisées comme paramètres d'évaluation de ces architectures.

Les architectures d'implémentation sont classées en deux grandes classes qui sont les architectures à base de mémoire et les architectures en pipeline. Les architectures à base de mémoire utilisent moins de ressources, mais elles nécessitent plusieurs cycles horloge par échantillon. Les architectures en pipeline sont les plus adaptées aux systèmes à haut débit, car elles permettent de calculer un ou plusieurs échantillons par cycle horloge. Les architectures en pipelines sont classées en quatre types qui sont : le *MDC*, le *SDF*, le *SDC* et le *MDF,* dont chacune possède des points forts et des points de faiblesse vis-à-vis des paramètres d'évaluation cités précédemment.

Les processeurs *FFT* conçus pour fonctionner dans un système de télécommunications à base de la technique *MIMO-OFDM ou MIMO-MC-CDMA* doivent être capables de calculer les *DFT*s de plusieurs séquences indépendantes en parallèle. De ce fait, les architectures proposées pour les systèmes *MIMO-OFDM* ou *MIMO-MC-CDMA* doivent assurer un traitement parallèle des échantillons tout en assurant une grande efficacité d'utilisation des ressources par exploitation de la technique de partage de ressources.

Chapitre 4

ARCHITECTURES *FFT* PROPOSEES POUR LES SYSTEMES *MIMO-OFDM* ET *MIMO-MC-CDMA*

4.1. Introduction

Nous avons vu dans le deuxième chapitre que la combinaison du codage spatio-temporel avec la technique *OFDM* ou la technique *MC-CDMA* permet d'améliorer significativement les performances du système de télécommunications radio mobile en termes de débit, de robustesse et de l'efficacité spectrale. La mise en œuvre de la technique *OFDM* ou la technique *MC-CDMA* nécessite le calcul de la *DFT* par l'utilisation d'un algorithme *FFT* approprié. Pour un système *MIMO,* le processeur *FFT* doit être capable de traiter plusieurs séquences indépendantes avec une utilisation efficace des ressources.

Dans ce présent chapitre, nous présentons les architectures que nous avons proposées dans le cadre de cette thèse pour l'implémentation d'un processeur *FFT* à haut débit qui permet le calcul des DFTs de deux ou de quatre séquences indépendantes. Ces architectures, appelées *MR2^2-2^3 MDC,* sont basées sur l'utilisation d'un mélange de Radix-2^2 et de Radix-2^3 et de l'approche *MDC* à deux ou à quatre chemins de traitement. Dans la section deux nous rappelons les deux algorithmes Radix-2^2 et de Radix-2^3, qui sont utilisés pour les architectures proposées. La section trois est consacrée à la présentation des architectures proposées. Nous commençons par l'explication du schéma de principe des architectures. Ensuite, nous passons à l'explication de chaque module élémentaire de la construction des architectures. Enfin, nous discutons les ressources utilisées

par les architectures proposées en comparaison avec d'autres architectures. Dans la quatrième, section nous présentons les résultats de l'implémentation des différentes architectures sur deux circuits FPGA de type Virtex-5 et Virtex-7. Ces résultats sont comparés avec les résultats d'autres travaux récents.

4.2 Rappels des deux algorithmes Radix-2^2 et Radix-2^3

Puisque les architectures que nous avons proposées sont basées sur l'utilisation d'un mélange de Radix-2^2 et de Radix-2^3, nous allons présenter dans cette section le développement mathématique et le *SFG* de ces deux algorithmes.

4.2.1. L'algorithme *FFT* Radix-2^2

L'expression de la *DFT* d'une séquence de taille *N* est donnée par l'expression de l'équation (3.1) du chapitre précédent. En appliquant aux indices *n* et *k* un « mapping » selon les deux équations suivantes [74]:

$$n = < \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 >_N \tag{4.1}$$

$$k = < k_1 + 2k_2 + 4k_3 >_N \tag{4.2}$$

avec : n_1=0,1, n_2=0,1, n_3=0,1,…N/4, k_1=0,1, k_2=0,1 et k_3=0,1,…N/4,

l'expression de la *DFT* s'écrie sous la forme suivante :

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)} \tag{4.3}$$

En effectuant la sommation suivant l'indice n_1 on obtient :

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \left[B_{\frac{N}{2}}(\tfrac{N}{4} n_2 + n_3, k_1) W_N^{(\frac{N}{4} n_2 + n_3) k_1} \right] W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)} \tag{4.4}$$

Où $B_{\frac{N}{2}}(\frac{N}{4} n_2 + n_3, k_1)$ représente la structure papillon Radix-2, dont l'expression mathématique est donnée par :

$$B_{\frac{N}{2}}(\tfrac{N}{4}n_2+n_3,k_1)=x(\tfrac{N}{4}n_2+n_3)+(-1)^{k_1}x(\tfrac{N}{4}n_2+n_3+\tfrac{N}{2}) \quad (4.5)$$

$B_{\frac{N}{2}}(\tfrac{N}{4}n_2+n_3,k_1)$ représente la somme (pour k_1 pair) ou la différence (pour k_1 impair) des différents couples d'échantillons qui sont espacés de *N/2.* Si la multiplication $B_{\frac{N}{2}}(\tfrac{N}{4}n_2+n_3,k_1).W_N^{(\frac{N}{4}n_2+n_3)k_1}$ est effectuée d'abord avant l'opération de décomposition suivante, l'algorithme obtenu est un Radix-2 simple. L'idée cruciale de l'algorithme Radix-2^2 est de ne pas effectuer cette multiplication, mais de cascader le facteur de rotation $W_N^{(\frac{N}{4}n_2+n_3)k_1}$ dans l'étape de décomposition suivante.

La décomposition du facteur de rotation permet d'écrire :

$$W_N^{(\frac{N}{4}n_2+n_3)k_1}W_N^{(\frac{N}{4}n_2+n_3)(2k_2+4k_3)}=(-j)^{n_2(k_1+2k_2)}W_N^{n_3(k_1+2k_2)}W_N^{4n_3k_3} \quad (4.6)$$

En substituant la décomposition de l'équation (4.6) dans l'expression de l'équation (4.4) on obtient :

$$X(k_1+2k_2+4k_3)=\sum_{n_3=0}^{\frac{N}{4}-1}\left[H_{\frac{N}{4}}(n_3,k_1,k_2)W_N^{n_3(k_1+2k_2)}\right]W_{\frac{N}{4}}^{n_3k_3} \quad (4.7)$$

Où $H_{\frac{N}{4}}(n_3,k_1,k_2)$ est donnée par l'expression suivante :

$$H_{\frac{N}{4}}(n_3,k_1,k_2)=\overbrace{\underbrace{x(n_3)+(-1)^{k_1}x(n_3+\tfrac{N}{2})}_{BFI}+(-j)^{(k_1+2k_2)}\left[\underbrace{x(n_3+\tfrac{N}{4})+(-1)^{k_1}x(n_3+\tfrac{3N}{4})}_{BFI}\right]}^{BFII} \quad (4.8)$$

L'équation (4.8) donne l'expression d'une nouvelle structure en papillons qui est la structure en papillons de l'algorithme Radix-2^2. Elle représente les deux premiers étages de papillons dont le *SFG* ne contient que des multiplications triviales par les coefficients $(-j)^{(k_1+2k_2)}$. La multiplication complexe standard par les facteurs de rotation $W_N^{n_3(k_1+2k_2)}$ est nécessaire après ces deux étages de papillons.

L'algorithme *FFT* Radix-2^2 *DIF* complet est obtenu en appliquant l'opération de décomposition d'une façon récursive sur les *N/4* points restants. La structure de base

de cet algorithme est présentée dans la figure 4.1. Cette structure est composée de 4 entrées, 4 sorties et de deux étages de calcul. Le premier étage contient deux papillons Radix-2 chevauchés, et une multiplication triviale par (-j). Le deuxième étage comprend deux papillons Radix-2 en parallèle, et trois multiplications générales (non triviales).

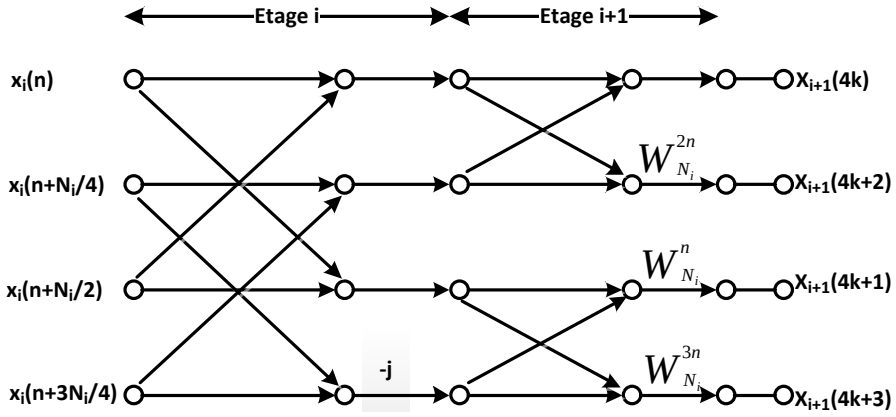

Figure 4.1 : Structure de base de l'algorithme *FFT* Radix-2^2.

Lorsque *n* et *k* varient de 0 à $N_i/4-1$ dans la structure de la figure 4.1, un groupe complet dans le *SFG* de l'algorithme Radix-2^2 *DIF* est formé. Avec N_i donné par :

$$N_i = \frac{N}{2^{(i-1)}} = 2^{n_s - i + 1} \tag{4.9}$$

$n_s = \log_2(N)$ représente le nombre d'étages dans le *SFG* de l'algorithme. Radix-2^2.

Un groupe est répété verticalement N/N_i fois pour obtenir deux étages consécutifs dans le *SFG* complet de l'algorithme Radix-2^2. Finalement, la formation de deux étages consécutifs est répétée $n_s/2$ fois horizontalement pour obtenir le *SFG* complet de l'algorithme *FFT* Radix-2^2. Prenant l'exemple de $N=16$: pour former un groupe pour les deux premiers étages ($i=1$ donc $N_i=N=16$) les valeurs de *n* et de *k* varient de 0 à 3 et les deux premiers étages sont formés d'un seul groupe ($N/Ni=1$). Il ne reste que deux autres étages pour former le *SFG* complet de l'algorithme car $n_s/2=2$. Pour ces deux derniers étages, on a $i=3$ donc $N_i=N/4=4$, ce qui signifie que $n=0$ et $k=0$ pour former un groupe, ce groupe est répété verticalement $16/4=4$ fois pour former

les deux derniers étages. La figure 4.2 illustre le *SFG* complet de l'algorithme *FFT* Radix-2^2 *DIF* pour une taille *N=16*.

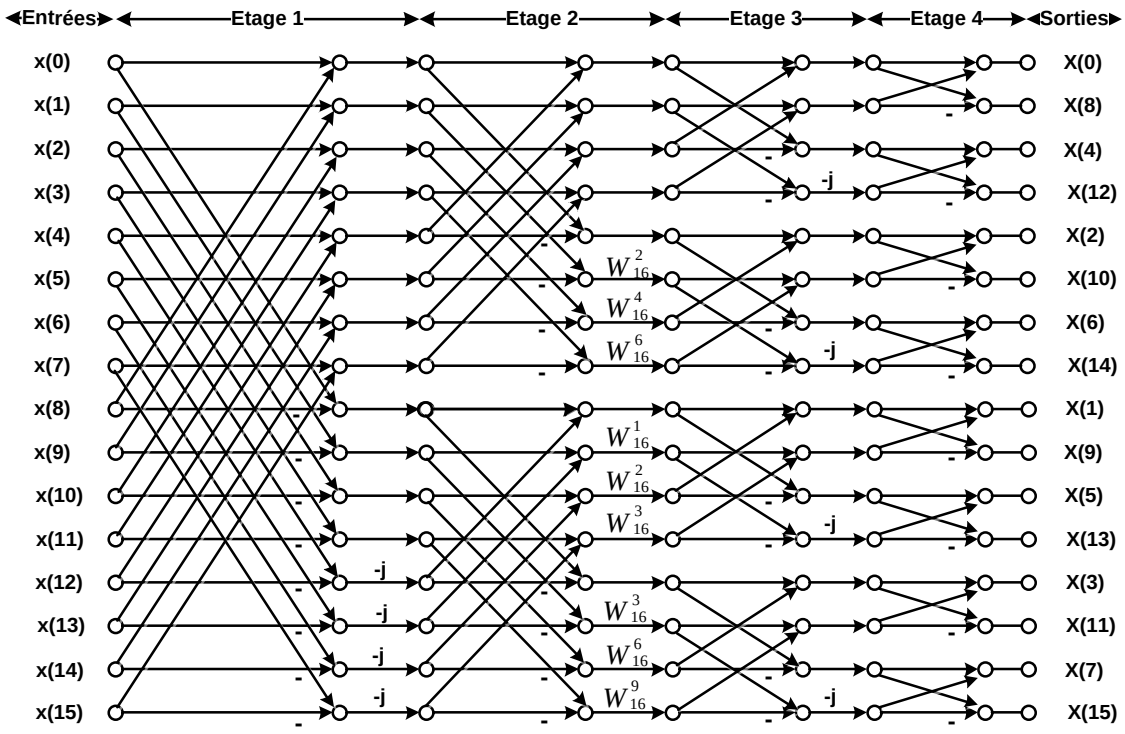

Figure 4.2 : Le *SFG* de l'algorithme *FFT* Radix-2^2 *DIF* pour *N=16.*

Les équations (4.7) et (4.8) montrent que l'algorithme Radix-2^2 requiert des multiplications complexes standards chaque deux étages comme illustré dans la figure 4.2, alors que l'algorithme Radix-2 nécessite des multiplications complexes pour chaque étage. Cela signifie que l'utilisation de l'algorithme Radix-2^2 au lieu de l'algorithme Radix-2 permet de réduire le nombre de multiplieurs de 50%, tout en gardant une structure de papillon simple. De ce fait, l'algorithme Radix-2^2 possède la même complexité arithmétique que le Radix-4, mais avec une structure papillon qui est très simple par rapport à ce dernier.

4.2.2. L'algorithme *FFT* Radix-2^3

L'algorithme Radix-2^3 est obtenu par l'application d'un « mapping » des deux indices *n*, *k* de l'équation (3.1) selon les deux formes suivantes :

$$n = < \frac{N}{2} n_1 + \frac{N}{4} n_2 + \frac{N}{8} n_3 + n_4 >_N \tag{4.10}$$

$$k = < k_1 + 2k_2 + 4k_3 + 8k_4 >_N \tag{4.11}$$

Où : n_1=0,1, n_2=0,1, n_3=0,1, n_4=0,1 …N/8, k_1=0,1, k_2=0,1, k_3=0,1, et k_4=0,1,…N/8.

La substitution des deux équations (4.10) et (4.11) dans l'équation (3.1) permet de réécrire l'expression de la *DFT* selon la forme suivante :

$$X(k_1 + 2k_2 + 4k_3 + 8k_4) = \sum_{n_4=0}^{\frac{N}{8}-1} \sum_{n_3=0}^{1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + \tfrac{N}{8} n_3 + n_4) W_N^{(\frac{N}{2} n_1 + \frac{N}{4} n_2 + \frac{N}{8} n_3 + n_4).(k_1 + 2k_2 + 4k_3 + 8k_4)} \tag{4.12}$$

La décomposition cascade du facteur de rotation de l'équation (4.12) selon les indices n_1, n_2, n_3 et n_4 est donnée par :

$$\begin{aligned} W_N^{n.k} &= W_N^{\frac{N}{2} n_1 k_1} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{\frac{N}{8} n_3 (k_1 + 2k_2 + 4k_3)} W_N^{n_4 (k_1 + 2k_2 + 4k_3 + 8k_4)} \\ &= (-1)^{n_1 k_1} (-j)^{n_2 (k_1 + 2k_2)} W_N^{\frac{N}{8} n_3 (k_1 + 2k_2 + 4k_3)} W_N^{n_4 (k_1 + 2k_2 + 4k_3)} W_{\frac{N}{8}}^{n_4 k_4} \end{aligned} \tag{4.13}$$

En remplaçant l'équation (4.13) dans l'équation (4.12), on obtient la formule suivante :

$$X(k_1 + 2k_2 + 4k_3 + 8k_4) = \sum_{n_4=0}^{\frac{N}{8}-1} \left[T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) W_N^{n_4 (k_1 + 2k_2 + 4k_3)} \right] W_{\frac{N}{8}}^{n_4 k_4} \tag{4.14}$$

Où $T_{\frac{N}{8}}(n_4, k_1, k_2, k_3)$ est l'expression d'une troisième structure de papillon qui est donnée par :

$$\begin{aligned} T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) &= H_{\frac{N}{4}}(n_4, k_1, k_2) + W_N^{\frac{N}{8}(k_1 + 2k_2 + 4k_3)} H_{\frac{N}{4}}(n_4 + \tfrac{N}{8}, k_1, k_2) \\ &= \underbrace{\overbrace{H_{\frac{N}{4}}(n_4, k_1, k_2)}^{BFII} + (\tfrac{\sqrt{2}}{2}(1-j))^{k_1} (-j)^{(k_2 + 2k_3)} \overbrace{H_{\frac{N}{4}}(n_4 + \tfrac{N}{8}, k_1, k_2)}^{BFII}}_{BFIII} \end{aligned} \tag{4.15}$$

L'équation (4.14) est l'expression de la *DFT* de *N/8* points de type $T_{\frac{N}{8}}(n_4, k_1, k_2, k_3)$ dont chaque point est une combinaison de 8 échantillons élémentaire selon l'équation (4.15). Ce mélange est obtenu à l'aide d'une structure élémentaire appelée structure en papillons de l'algorithme Radix-2^3 (notée *BFIII* dans

l'équation (4.15)). L'algorithme *FFT* Radix-2^3 complet est obtenu en appliquant d'autres décompositions d'une façon récursives sur les *DFT*s de *N/8* points. La structure de base de l'algorithme *FFT* Radix-2^3 *DIF* est donnée dans la figure 4.3.

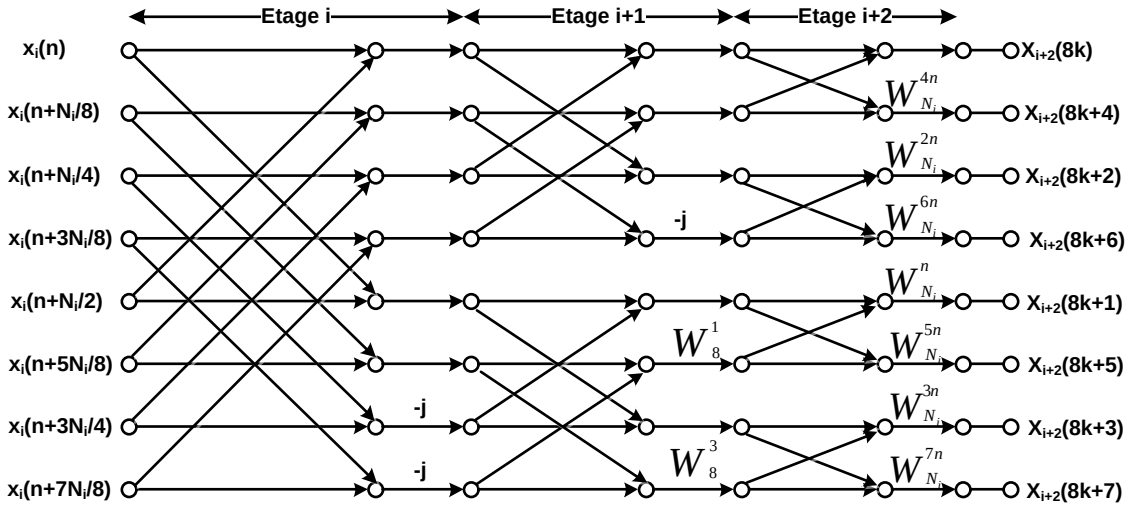

Figure 4.3 : Structure de base de l'algorithme Radix-2^3.

La structure de base de l'algorithme Radix-2^3 est composée de 8 entrées, 8 sorties et 3 étages de calcul. Le premier étage est composé de 4 papillons Radix-2 chevauchés et de deux multiplications par *–j.* Le deuxième étage comprend deux groupes papillons dont chaque groupe contient deux papillons Radix-2 chevauchés. Le deuxième étage contient aussi trois multiplications par *–j*, W_8^1 et W_8^3. Le troisième étage est constitué de 4 papillons Radix-2 en parallèle suivis de 7 multiplications générales non-triviales.

La structure de la figure 4.3 montre que l'algorithme Radix-2^3 nécessite des multiplications non-triviales chaque trois étages, au lieu de chaque deux étages pour l'algorithme Radix-2^2, et de chaque étage pour l'algorithme Radix-2. Cela permet une réduction importante de la complexité arithmétique tout en gardant une structure papillon très simple qui est celle du Radix-2.

Comme pour l'algorithme Radix-2^2, le *SFG* complet de l'algorithme Radix-2^3 est

obtenu par une suite de répétitions verticales et horizontales de la structure de base de la figure 4.3. Un groupe de N_i entrées et N_i sorties est formé lorsque *n* et *k* varient de *0 à Ni/8-1.* Trois étages consécutifs du *SFG* sont formés par N/N_i groupes parallèles (répétition verticale). Finalement, l'opération de construction des trois étages consécutifs est répétée $n_s/3$ fois pour former le *SFG* complet. Prenant l'exemple de *N=64 :* pour former un groupe dans les trois premiers étages (*i=1* donc $N_i=N=64$) les valeurs de *n* et de *k* varient de 0 à 7 et les trois premiers étages sont formés d'un seul groupe (*N/Ni=1*). On doit ajouter trois autres étages pour former le *SFG* complet de l'algorithme puisque $n_s/3=2$. Pour ces trois étages on a *i=4,* donc $N_i=N/8=8$, ce qui signifie que *n=0* et *k=0* pour former un groupe, ce groupe est répété verticalement *64/8=8* fois pour former les trois derniers étages du *SFG*.

4.3. Présentation des architectures proposées :

Pour les processeurs *FFT* implémentés sur des cibles de type *FPGA*, la fréquence de fonctionnement est un paramètre très important dans l'évaluation de leurs performances. En effet, l'augmentation de la fréquence de fonctionnement permet d'augmenter le débit de traitement du processeur sans augmentation importante de ressources utilisées, ce qui permet d'obtenir un bon rapport débit à ressources utilisées. A notre connaissance, les travaux de recherche dans le domaine de l'implémentation des processeurs *FFT* sur *FPGA* n'ont pas encore présenté des études détaillées sur la manière d'augmentation de la fréquence de fonctionnement des processeurs *FFT* basés sur l'architecture *MDC*. La fréquence de fonctionnement du processeur *FFT* reste limitée, si on n'adopte pas une méthode appropriée pour son amélioration. Cette limitation de la fréquence de fonctionnement est un handicap pour les systèmes de communications nécessitant un débit de transmission, et donc un débit de traitement, très élevé comme les systèmes de la quatrième génération (*4G)* et de la cinquième génération (*5G)*. La plupart des architectures proposées dans la littérature fonctionnent à des fréquences très limitées, si elles sont implémentées sur des cibles *FPGA* sans introduire des modifications qui permettent l'augmentation de leurs fréquences. La limitation de la fréquence devient plus significative pour les processeurs *FFT* traitant des séquences de taille *N* élevée. Les résultats

d'implémentation montrent que plus la taille de la *FFT* est élevée plus la fréquence diminue.

La méthode d'amélioration de la fréquence de fonctionnement d'une architecture implémentée sur un circuit *FPGA* consiste à insérer un nombre approprié de registres Flip-Flop appelés registres de pipeline, aux sorties des composants qui constituent l'architecture [18]. L'insertion des registres de pipeline permet de réduire la longueur des chemins critiques, ce qui amène à l'augmentation de la fréquence de fonctionnement du circuit. L'insertion des registres de pipeline au niveau des sorties des composants introduit des retards sur les réponses de ces composants. Ces retards ; qui sont exprimés en nombre de cycles horloge que fait le composant pour répondre ; affectent le bon fonctionnement du processeur *FFT* parce qu' ils entraînent la perte de la synchronisation entre les différents échantillons. Pour résoudre ce problème, il est indispensable de corriger et de compenser l'effet des registres de pipelines par l'insertion aux endroits appropriés du circuit et avec le nombre approprié, des éléments de retard afin de rétablir la synchronisation et de garder la distance correcte entre les différents échantillons.

Pour déterminer le nombre de registres pipeline qui sont nécessaires pour chaque type de composants utilisé dans les architectures proposées (afin d'atteindre la fréquence de fonctionnement maximale), nous avons procédé à une série de simulations et d'implémentations en utilisant l'outil « Xilinx System Generator » et l'outil *ISE*. Les résultats montrent que le nombre de retards pour atteindre la fréquence maximale est égal à : un pour les additionneur/soustracteur et les multiplexeurs, deux pour les blocs *RAM* et *ROM* et quatre pour les multiplieurs

La structure générale des architectures que nous avons proposées est donnée dans la figure 4.4. Cette structure est composée d'une unité d'ordonnancement des entrées, de *m* modules de calcul selon la structure Radix-2^2 ou Radix-2^3, et d'une unité de contrôle.

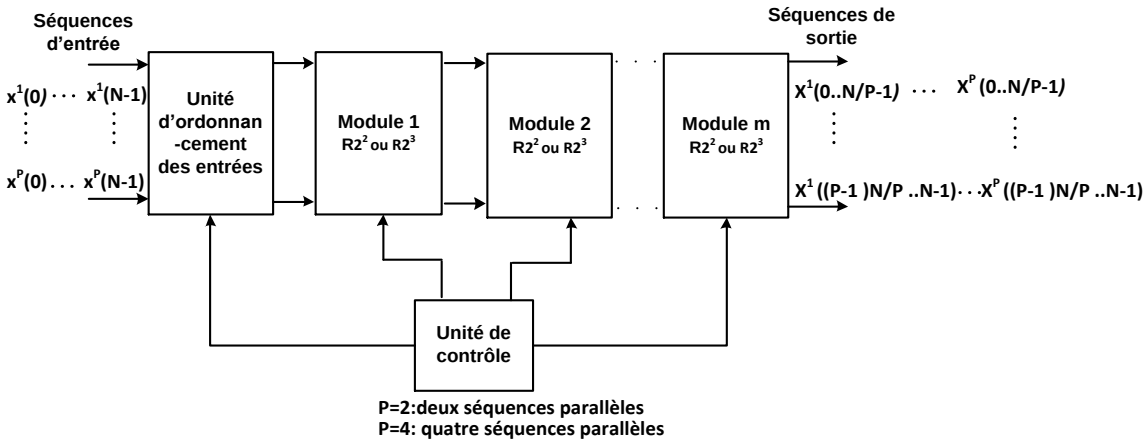

Figure 4.4 : Structure générale de l'architecture *MR2²-R2³ MDC* proposée.

L'architecture contient un mélange de m_2 modules Radix-2^2 et de m_3 modules Radix-2^3. Chaque module Radix-2^2 permet de réaliser les opérations de deux étages consécutifs du *SFG,* et chaque module Radix-2^3 implémente trois étages consécutifs du *SFG*. Par conséquent, les paramètres m_2, m_3 *et* n_s sont reliés par l'équation :

$$n_s = 2m_2 + 3m_3 \tag{4.16}$$

Pour chaque entier supérieur à deux n_s, on peut trouver au moins deux autres entiers m_2 et m_3 qui permettent d'exprimer n_s selon l'équation (4.16). Cela signifie que l'utilisation d'un mélange de modules Radix-2^3 et Radix-2^2 (MR2^2-2^3) permet de concevoir une architecture pour toutes les valeurs de la taille *N* qui sont une puissance de 2 ($N = 2^{n_s}$). Cette possibilité n'est pas offerte si on utilise des modules Radix-2^2 seuls ou des modules Radix-2^3 seuls. Donc, l'architecture *MR-2²-2³ MDC* est évolutive « scalable » pour toutes les tailles qui sont une puissance de deux.

4.3.1. L'architecture *MR-2²-2³ MDC* pour deux séquences d'entrée

Le processeur *FFT* selon l'architecture proposée permet de calculer les *DFT*s de deux séquences indépendantes $A = x^1(0..N-1)$ et $B = x^2(0..N-1)$. L'unité d'ordonnancement des entrées permet de réarranger les deux séquences d'entrée

selon l'ordre donné dans la figure 4.5. La séquence d'entrée $A = x^1(0..N-1)$ est subdivisée en deux sous-séquences parallèles : la sous-séquence supérieure $A_1 = x^1(0..\frac{N}{2}-1)$ qui contient la première moitié de la séquence *A* et la sous-séquence inférieure $A_2 = x^1(\frac{N}{2}..N-1)$ qui contient la deuxième moitié de la séquence *A*. La séquence *B* est ensuite subdivisée en deux sous-séquences B_1 et B_2 de la même manière.

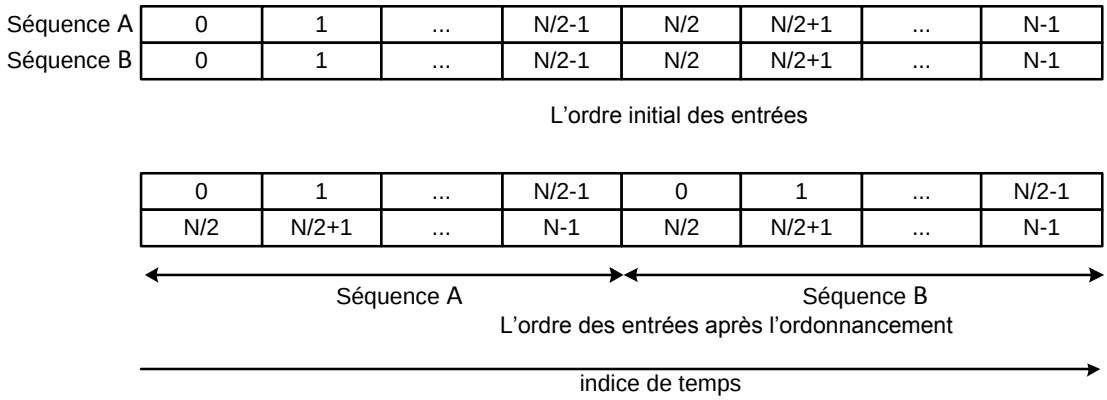

Figure 4.5 : Réarrangement des entrées par l'unité d'ordonnancement pour l'architecture *MR2²-2³MDC* à deux séquences.

L'opération de réarrangement est réalisée à l'aide de deux mémoires *RAM* chacune de taille *N/2.* La chronologie des différentes opérations de lectures et d'écritures dans les deux *RAM*s est donnée dans le tableau 4.1. L'opération de réarrangement s'effectue en quatre phases, chaqu'une de durée *N/2* cycles horloge, selon l'ordre suivant :

- Phase 1 : c'est une phase transitoire durant laquelle la sous-séquence A_1 est écrite dans la *RAM1* et la sous-séquence B_1 est écrite dans la *RAM2*. Les sorties de l'unité d'ordonnancement sont nulles.
- Phase 2 : la sous-séquence A_1 est lue de la *RAM1,* au même temps que la sous-séquence B_2 est écrite dans *RAM1*. La sous-séquence A_1 est acheminée vers la sortie supérieure de l'unité d'ordonnancement et la sous-séquence A_2

est acheminée de l'entrée supérieure de l'unité vers sa sortie inférieure. La *RAM2* reste inactive durant cette deuxième phase.

- Phase 3 : une nouvelle sous-séquence A_1' est écrite dans *RAM1* au même temps que la sous-séquence B_2 est lue de la *RAM1* et est acheminée vers la sortie inférieure de l'unité d'ordonnancement. La nouvelle sous-séquence B_1' est écrite dans *RAM2,* au même temps que la sous-séquence précédente B_1 est lue de *RAM2* et est acheminée vers la sortie supérieure de l'unité d'ordonnancement.
- La phase 4 est identique à la phase 2, mais avec de nouvelles sous-séquences et le cycle recommence. Le fonctionnement se réduit à la succession des deux phases 2 et 3 d'une façon répétitive.

Tableau 4.1 Chronologie des différentes opérations de lectures et d'écritures pour les deux *RAM*s de l'unité d'ordonnancement des entrées.

Phase / Opération	Phase1 0..N/2-1	Phase2 N/2..N-1	Phase 3 N..3N/2-1	Phase 4 3N/2..2N-1
Ecriture dans *RAM* 1	A_1	B_2	A_1'	B_2'
Lecture de RAM1	0	A_1	B_2	A_1'
Ecriture dans RAM2	B_1	rien	B_1'	rien
Lecture de RAM2	0	rien	B_1	rien
Sortie supérieure	0	A_1	B_1	A_1'
Sortie inférieure	0	A_2	B_2	A_2'

Le circuit qui permet de réaliser l'ordonnancement des entrées selon le fonctionnement décrit ci-dessus est donné dans la figure 4.6. Ce circuit contient ; en plus des deux *RAMs* notées *RAM1* et *RAM2* ; un multiplexeur à l'entrée afin de sélectionner la sous-séquence à écrire dans *RAM1*, deux multiplexeurs à la sortie qui servent à acheminer les données soit vers la sortie supérieure ou vers la sortie

inférieure selon le principe décrit précédemment et de deux éléments de retard. Les éléments de retard sont nécessaires pour compenser l'effet des retards introduits par les différents composants du circuit. Ces retards sont représentés à l'intérieur de chaque composant par la notation Z^{-i} dont *i* représente le retard en nombre de cycles horloge. Il est important de noter que les deux mémoires *RAM* sont configurées en mode lecture avant écriture « Read Before Write Mode » pour ne pas écraser les anciennes données avant l'écriture des nouvelles données.

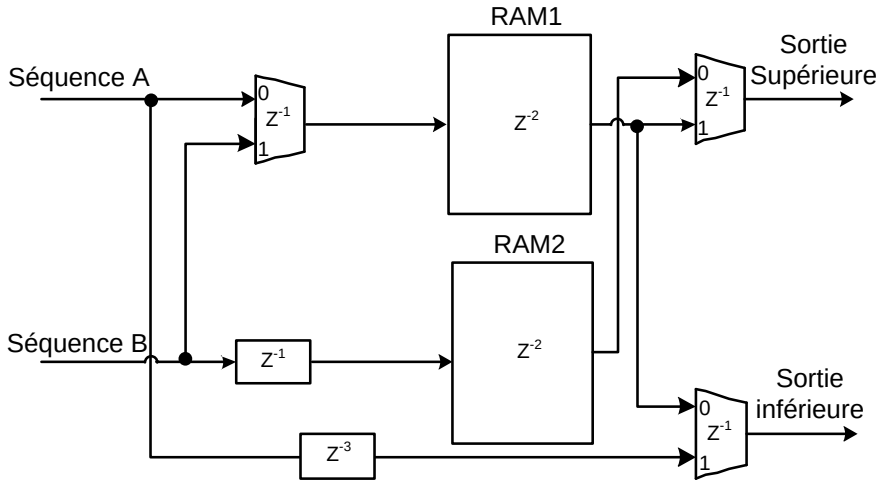

Figure 4.6 Circuit de l'unité de réordonnancement des entrées pour l'architecture à deux séquences.

Après réordonnancement des deux séquences *A* et *B* selon l'ordre donnée dans la figure 4.5, les deux sous-séquences A_1 et A_2 sont passées vers les différents modules de calcul (module 1 à module *m*) pour subir les traitements nécessaires au calcul de la *DFT* de la séquence *A*. Le traitement des deux sous-séquences A_1 et A_2 est ensuite suivi par le traitement des deux sous-séquences B_1 et B_2 pour calculer la *DFT* de la séquence *B.* les modules de calcul sont un mélange de modules Radix-2^2 et de modules Radix-2^3. Les circuits réalisant les modules Radix-2^2 sont donnés dans les deux figures 4.7 et 4.8.

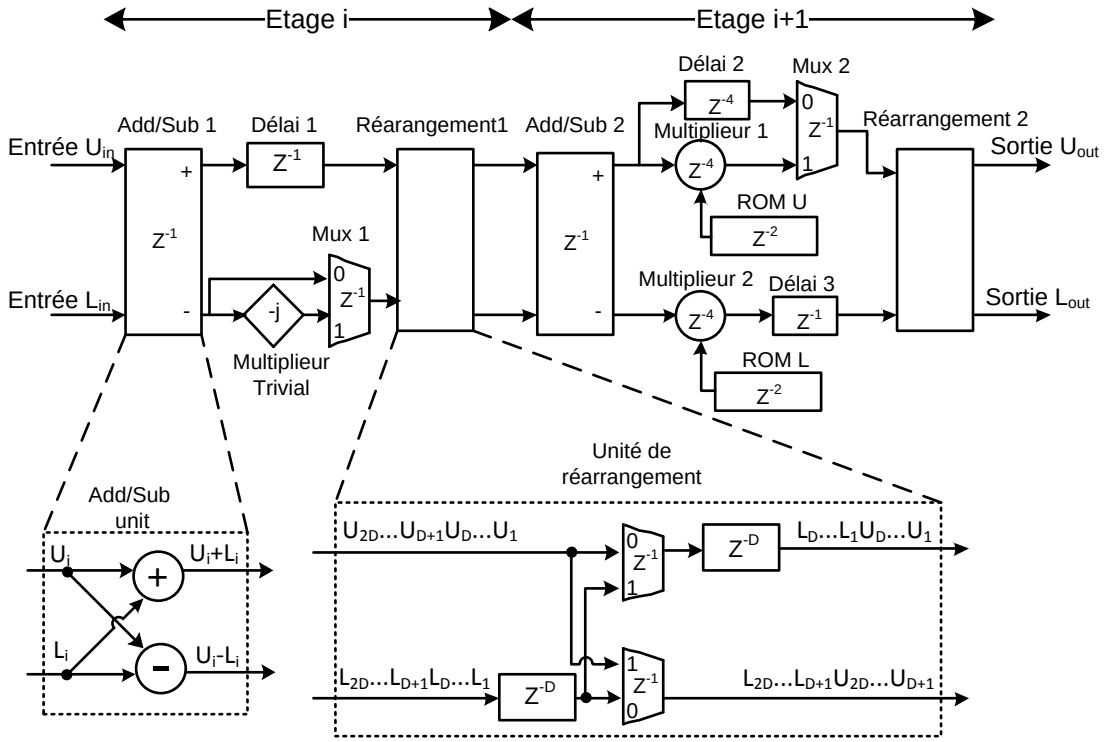

Figure 4.7 : Schéma du module Radix-2^2 pour l'architecture *MR2^2-2^{3}MDC* à deux séquences utilisé pour les étages de 1 à n_s*-2*.

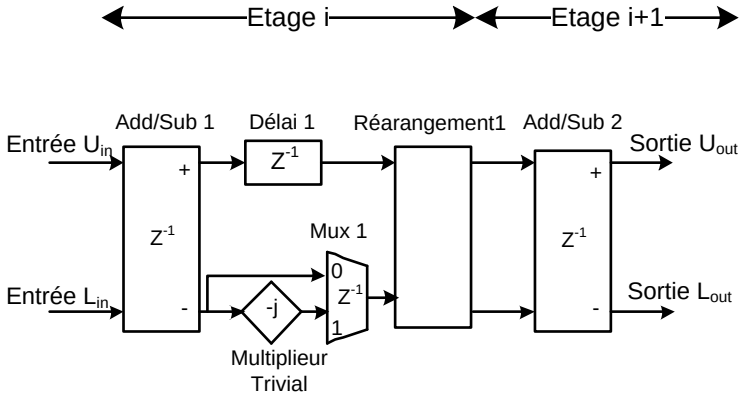

Figure 4.8 Schéma du module Radix-2^2 pour l'architecture *MR2^2-2^3 MDC* à deux séquences utilisé pour les deux étages n_s*-1* et n_s.

Le schéma de la figure 4.7 permet de réaliser les opérations de deux étages consécutifs (étage *i* et étage *i+1*) pour *i* allant de *1* à n_s*-2* dans le *SFG* de l'algorithme *FFT*. Les différentes opérations de traitement sont appliquées parallèlement selon deux voies de traitement (voie supérieure et voie inférieure) sur les deux entrées U_{in} et L_{in} afin de produire les deux sorties U_{out} et L_{out}. Les deux sorties U_{out} et L_{out} sont à leur tour injectées aux entrées du module suivant qui peut être un module Radix-2^2 ou un module Radix-2^3, pour être traitées puis injectées vers un autre module, et ainsi de suite jusqu'au dernier module.

Le module Radix-2^2 contient deux additionneurs/soustracteurs, deux multiplieurs complexes avec les deux ROMs de coefficients de rotation, deux unités de réarrangement des échantillons, deux multiplexeurs, trois éléments de retard pour établir la synchronisation, et un multiplieur trivial par *–j.*

Les unités de réarrangement des échantillons sont insérées entre la sortie de chaque étage (étage *i*) et l'entrée de l'étage suivant (étage *i+1*). Chaque unité de réarrangement permet de réorganiser les échantillons de la voie supérieure et ceux de la voix inférieure sortant de l'étage courant (étage *i*) afin de les introduire dans l'ordre et avec la distance correcte à l'additionneur/soustracteur de l'étage suivant (étage *i+1*). Le schéma détaillé de l'unité de réarrangement est donné en bas de la figure 4.7. Elle est constituée de *2.D* retards élémentaires ; dont *D* retards en amont de la voie inférieure et *D* autres retards en aval de la voie supérieure ; en plus de deux multiplexeurs à deux entrées chacun. Pour une unité de réarrangement qui est insérée entre les deux étages *i* et *i+1*, la valeur de *D* est donnée par :

$$D = {}^{N}\!/_{2^{(i+1)}} \tag{4.17}$$

Le signal de contrôle des deux multiplexeurs ; noté C_{n_s-i-1} ; est directement obtenu d'un compteur binaire simple. Ce compteur est suffisant pour générer les signaux de contrôle de tous les multiplexeurs de l'architecture, et pour générer les adresses de toutes les mémoires (RAM et ROM) de l'architecture. Le signal C_{n_s-i-1} prend la valeur

zéro pendant *D* cycles horloge, suivie de la valeur 1 pour *D* autres cycles horloge *i.e*. $C_{n_s-i-1} = \overbrace{00...0}^{D}\overbrace{11...1}^{D}$.

Les unités de réarrangement des échantillons opèrent de la manière suivante :

- durant *D* cycles horloge, les sorties de la voie supérieure de l'étage *i* ; notées U_1 à U_D ; sont acheminées vers l'entrée supérieure de l'étage *i+1* au même temps que les données U_{D+1} à U_{2D} sont acheminées vers l'entrée inférieure de l'étage *i+1.*
- durant les *D* cycles horloge qui suivent, les sorties de la voie inférieure de l'étage *i* ; notées L_1 à L_D ; sont acheminées vers l'entrée supérieure de l'étage *i+1* au même temps que les données L_{D+1} à L_{2D} sont acheminées vers l'entrée inférieure de l'étage *i+1.*

Les facteurs de rotation $W_{N_i}^{n}$, avec *n=0..Ni/4-1,* sont stockés dans la *ROM* du chemin supérieur *(ROMU)* de taille $N_i/4$ tandis que les facteurs de rotation $W_{N_i}^{2n}$ et $W_{N_i}^{3n}$, avec *n=0..Ni/4-1,* sont stockés dans la *ROM* de la voie inférieure (*ROML)* de taille $N_i/2$. La *ROMU* est adressée à l'aide des signaux C_0 à C_{n_s-i-2} du compteur binaire, alors que *ROML* est adressée à l'aide des signaux C_0 à C_{n_s-i-1} du même compteur binaire. On note que le module Radix-2^2 pour les deux derniers étages du *SFG* (étages n_s*-1* et n_s) ne contient pas de multiplieurs complexes ni de la deuxième unité de réarrangement comme illustré dans la figure 4.8.

Le module Radix-2^3 permet d'effectuer toutes les opérations de trois étages consécutifs du *SFG*. La figure 4.9 donne le schéma du module Radix-2^3 utilisé pour les étages allants de *1* à n_s*-3,* alors que le schéma du même module pour les trois derniers étages *(*n_s*-2,* n_s*-1 et* n_s*)* est donné dans la figure 4.10. Le module Radix-2^3 utilise trois additionneurs/soustracteurs, un multiplieur trivial, deux multiplieurs standards avec les deux *ROMs* de facteurs de rotation, une unité de multiplication par constante complexe *CCMU1* « Constant Complex Multiplier Unit », un multiplexeur, et deux éléments de retard.

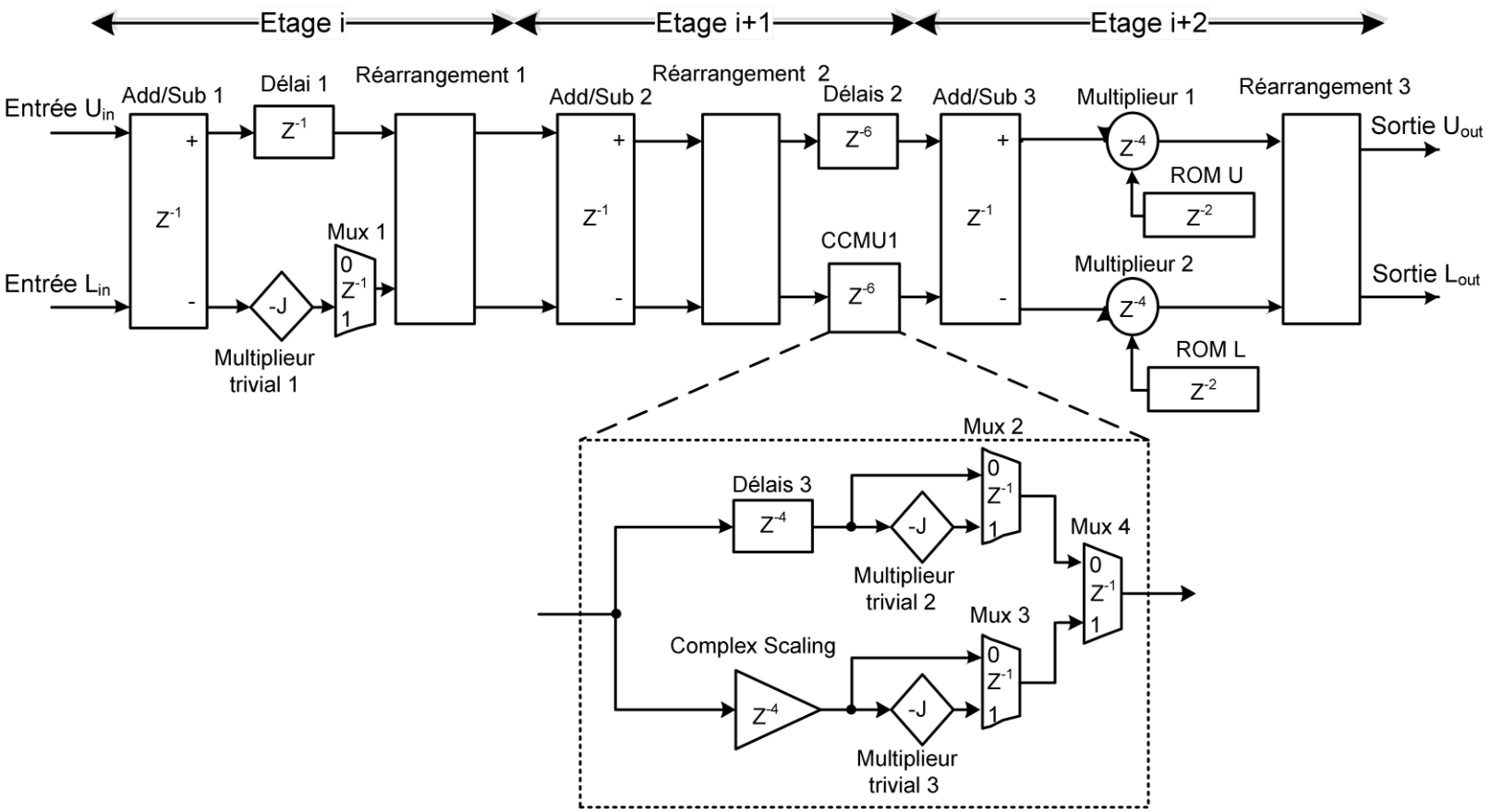

Figure 4.9 : Schéma du module Radix-2^3 pour l'architecture MR2^2-2^3 à deux séquences utilisé pour les étages allants de 1 à n_s-3.

L'unité de multiplication par constante complexe *CCMU1* permet d'effectuer la multiplication de quelques échantillons de la voie inférieure par les coefficients de rotation *–j*, W_8^1 et W_8^3 comme indiqué par la structure en papillons de la figure 4.3. Cette unité comprend un multiplieur par une constante complexe « complexe scaling », qui permet la multiplication par W_8^1, deux multiplieurs triviaux, trois multiplexeurs et un élément de 4 retards. Pour réduire le nombre de composants de l'unité *CCMU1*, le même multiplieur par la constante W_8^1 est cascadé avec un multiplieur trivial par *–J* pour réaliser la multiplication par W_8^3.

Les facteurs de rotation $W_{N_i}^0$, $W_{N_i}^n$, $W_{N_i}^{2n}$ et $W_{N_i}^{3n}$ pour, *n=0..N$_i$/8-1,* sont stockés dans la *ROM* de la voie supérieure *ROMU* tandis que les facteurs de rotation $W_{N_i}^{4n}$, $W_{N_i}^{5n}$, $W_{N_i}^{6n}$ et $W_{N_i}^{7n}$ sont stockés dans la *ROM* de la voie inférieure *ROML*. Les deux *ROM*s ont la même taille *N/2* et elles sont adressées par les mêmes signaux C_0 à C_{n_s-i-1} issus d'un compteur binaire

Le circuit du module Radix-2^3 ne contient pas de multiplieurs standard ni de la troisième unité de réarrangement, s'il est utilisé pour réaliser les trois derniers étages du *SFG* (étages *n_s-2, n_s-1 et n_s*) comme illustré dans la figure 4.10.

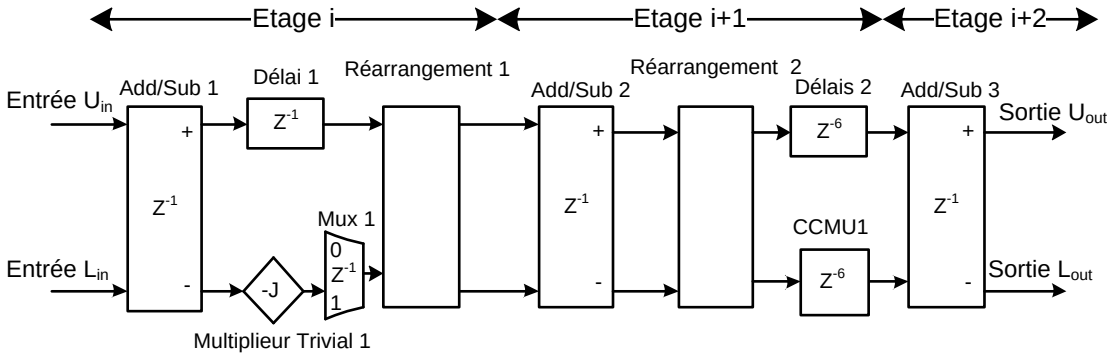

Figure 4.10 : Schéma du module Radix-2^3 pour l'architecture *MR-2^2-2^3-MDC* à deux séquences utilisé pour les étages de *n_s-2, n_s-1* et *n_s*.

4.3.2. L'architecture *MR-2²-2³-MDC* pour quatre séquences d'entrée :

L'unité d'ordonnancement des entrées pour l'architecture à quatre séquences d'entrée permet de réorganiser les quatre séquences d'entrées *A, B, C* et *D* selon l'ordre donnée dans la figure 4.11. La séquence $A = x^1(0..N-1)$ est subdivisée en quatre sous-séquences parallèles $A_1 = x^1(0..\frac{N}{4}-1)$, $A_2 = x^1(\frac{N}{4}..\frac{N}{2}-1)$, $A_3 = x^1(\frac{N}{2}..\frac{3N}{4}-1)$ et $A_4 = x^1(\frac{3N}{4}..N-1)$. De la même façon, chacune des trois autres séquences *B, C* et *D* est subdivisée en quatre sous séquences.

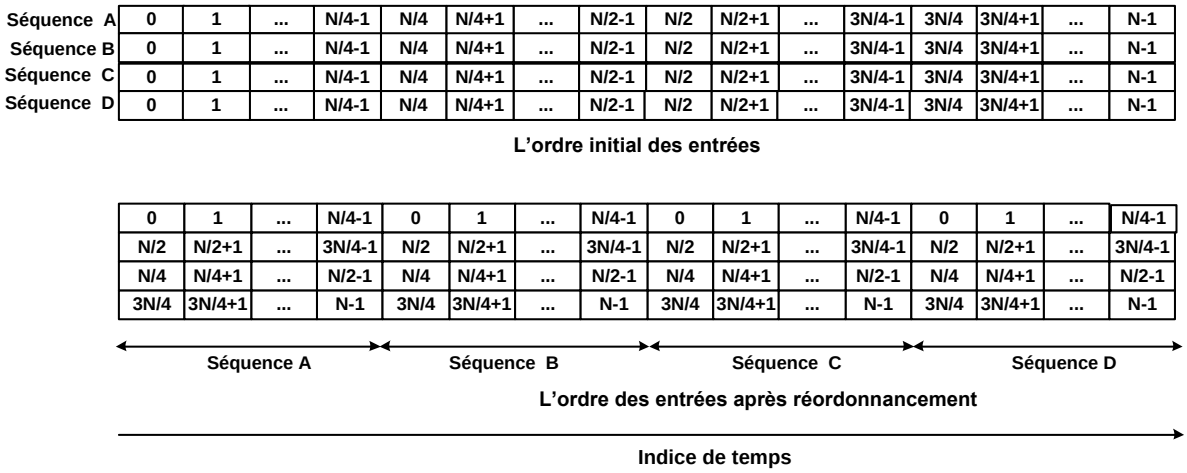

Figure 4.11 Réarrangement des entrées par l'unité d'ordonnancement pour l'architecture *MR2²-2³-MDC* à quatre séquences.

L'idée de réalisation de l'unité d'ordonnancement des entrées est inspérée de la méthode proposée par Yang, Tsai et Chuang [102], avec l'introduction des modifications appropriées sur le nouvel ordre des séquences d'entrée. L'unité nécessite quatre groupes de *RAM* notés *RAM_a*$_i$, *RAM_b*$_i$, *RAM_c*$_i$ et *RAM_d*$_i$, chaque groupe englobe trois *RAMs*. Ainsi, le groupe *RAM_a*$_i$ contient les trois *RAMs*, *RAM_a1, RAM_a2* et *RAM-a3,* la même chose pour les trois autres groupes de *RAM*s. L'unité d'ordonnancement contient donc 12 *RAM*s organisées selon la structure de la figure 4.12.

Figure 4.12 : Illustration de la procédure de lecture/écriture des *RAM*s pour l'unité d'ordonnancement des entrées pour l'architecture *MR2^2-2^3-MDC* à quatre séquences.

Le fonctionnement de l'unité d'ordonnancement peut être décrit en 11 phases chacune de durée *N/4*. Les trois premières phases sont transitoires et ne concernent que le début de fonctionnement, tandis que les 8 autres phases sont permanentes. La procédure de fonctionnement est la suivante :

- Phase 1 : les sous-séquences A_1, B_1, C_1 et D_1 sont écrites dans RAM_a_1, RAM_b_1, RAM_c_1 et RAM_d_1 respectivement.
- Phase 2 : les sous-séquences A_2, B_2, C_2 et D_2 sont écrites dans RAM_a_2, RAM_b_2, RAM_c_2 et RAM_d_2 respectivement.
- Phase 3 : les sous-séquences A_3, B_3, C_3 et D_3 sont écrites dans RAM_a_3, RAM_b_3, RAM_c_3 et RAM_d_3 respectivement.

Après la fin de la phase 3, on arrive aux 8 phases permanentes de fonctionnement qui sont illustrées en détail dans la figure 4.12. Les 12 blocs de *RAM*s sont arrangés sous forme de matrice, où chaque ligne représente les *RAMs* du même groupe et chaque colonne représente les *RAMs* ayant le même indice. Pour chaque phase, les *RAMs* qui sont actives (en lecture et écriture) sont représenté en gras, leurs entrées (lecture) et leurs sorties (écriture) sélectionnées sont représentées.

- Phase 4 : les sous-séquences A_1, A_2 et A_3 sont lues de *RAM_*a_1, *RAM_*a_2, *RAM_*a_3, et elles sont acheminées vers les sorties 1, 3 et 2 respectivement. La sous-séquence A_4 est directement acheminée de l'entrée vers la sortie 4. En même temps, les sous-séquences B_4, C_4 et D_4 sont écrites dans RAM_a_1, RAM_a_2 et RAM_a_3 respectivement.
- Phase 5 : B_1, B_2, B_3 et B_4 sont lues de *RAM_b1, RAM_b2, RAM_b3* et *RAM_a1* et acheminées vers les sorties 1, 3, 2 et 4 respectivement. Au même temps B_1', C_1', D_1' *et* A_1' sont écrites dans *RAM_b1, RAM_b2, RAM_b3* et *RAM_a1* respectivement.
- Phase 6 : C_1, C_2, C_3 et C_4 sont lues à partir de *RAM_c1, RAM_c2, RAM_c3* et *RAM_a2,* et sont acheminées vers les sorties 1, 3, 2 et 4 respectivement. Au même temps, A_2', B_2', C_2' et D_2' sont écrites dans *RAM_a2, RAM_c1, RAM_c2* et *RAM_c3* respectivement.
- Phase 7 : D_1, D_2, D_3 et D_4 sont lues à partir de *RAM_d1, RAM_d2, RAM_d3* et *RAM_a3,* et acheminées vers les sorties 1, 3, 2 et 4 respectivement. Au même

temps $A_3^{'}$, $B_3^{'}$, $C_3^{'}$ et $D_3^{'}$ sont écrites dans *RAM_a3, RAM_c1, RAM_c2* et *RAM_c3* respectivement.

- Phase 8 : $A_1^{'}$, $A_2^{'}$, *et* $A_3^{'}$ sont lues à partir de *RAM_a1, RAM_a2, RAM_a3,* et acheminées vers les sorties 1,3 et 2 respectivement. Au même temps, $B_4^{'}$, $C_4^{'}$ et $D_4^{'}$ sont écrites dans *RAM_a1, RAM_a2* et *RAM_a3*. $A_4^{'}$ est directement dirigée vers la sortie 4.
- Phase 9 : $B_1^{'}$, $B_2^{'}$, $B_3^{'}$ et $B_4^{'}$ sont lues de *RAM_b1, RAM_c1, RAM_d1* et *RAM_a1,* et routées vers les sorties 1, 3, 2 et 4 respectivement. Au même temps, A_1, B_1, C_1, D_1 sont écrites dans *RAM_a1, RAM_b1, RAM_c1* et *RAM_d1* respectivement.
- Phase 10 : $C_1^{'}$, $C_2^{'}$, $C_3^{'}$ et $C_4^{'}$ sont lues de *RAM_b2, RAM_c2, RAM_d2* et *RAM_a2,* et routées vers les sorties 1, 3, 2 et 4 respectivement. Au même temps, A_2, B_2, C_2 et D_2 sont écrites dans *RAM_a2, RAM_b2, RAM_c2* et *RAM_d2* respectivement.
- Phase 11 : $D_1^{'}$, $D_2^{'}$, $D_3^{'}$ et $D_4^{'}$ sont lues de *RAM_b3*, *RAM_c3, RAM_d3* et *RAM_a3,* et routées vers les sorties 1, 3, 2 et 4 respectivement. Au même temps, A_3, B_3, C_3 et D_3 sont écrites dans *RAM_a3*, *RAM_b3*, *RAM_c3* et *RAM_d3* respectivement.

A la fin de la phase 11 on arrive à une situation identique à celle de la fin de la phase 3, et on revient donc à la phase 4 et la procédure recommence à nouveau.

A partir de la description de la procédure de lecture et d'écriture, nous avons proposé le circuit de la figure 4.13 qui permet la concrétisation matérielle de cette procédure. Le circuit est constitué principalement de quatre groupes de *RAM*s dont chaque groupe contient trois *RAM*s, de commutateurs d'entrée, de commutateurs de sortie et, d'un bloc de génération des signaux de lecture des *RAM*s. Les commutateurs d'entrée sont réalisés à l'aide de multiplexeurs et ils permettent l'acheminement des sous-séquences qui sont présentes aux entrées vers les *RAM*s correspondantes selon la description donnée dans le paragraphe précédent.

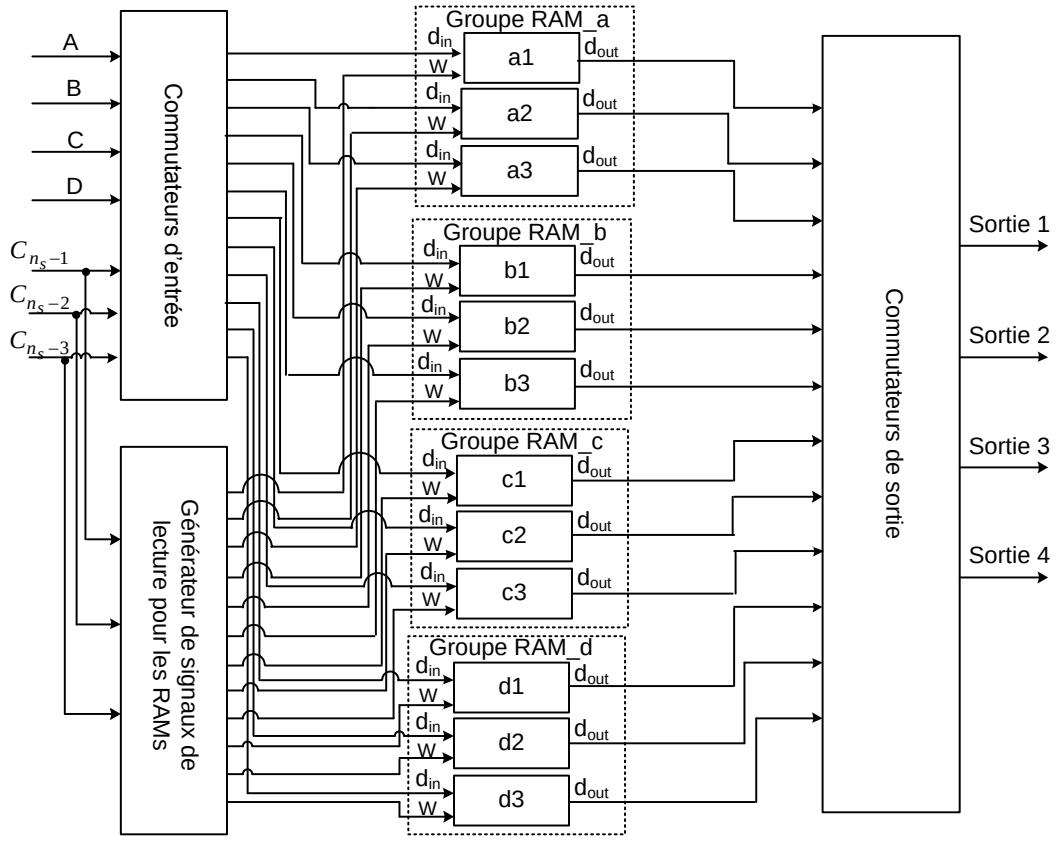

Figure 4.13 : Schéma du circuit proposé pour l'unité de l'ordonnancement des entrées pour l'architecture *MR2^2-2^3-MDC* à quatre séquences.

Les commutateurs de sortie permettent à leur tour l'acheminement des sous-séquences à partir des mémoires *RAM* vers les quatre sorties de l'unité de réordonnancement, toujours selon la description du paragraphe précédent. Les commutateurs de sortie sont aussi réalisés à l'aide de multiplexeurs. Finalement, le générateur de signaux de lecture permet de générer les 12 signaux de commande de lecture/écriture des 12 mémoires *RAM*s. Les chronogrammes de ces signaux de commande sont donnés dans la figure 4.14. On rappelle que chaque opération de lecture est suivie d'une opération d'écriture, ce qui permet de réaliser une lecture et une écriture par cycle horloge pour chaque *RAM* active. Le bloc de génération de signaux de lecture est réalisé à l'aide d'éléments logiques simple et il ne consomme qu'un nombre très limité de ressources *FPGA*.

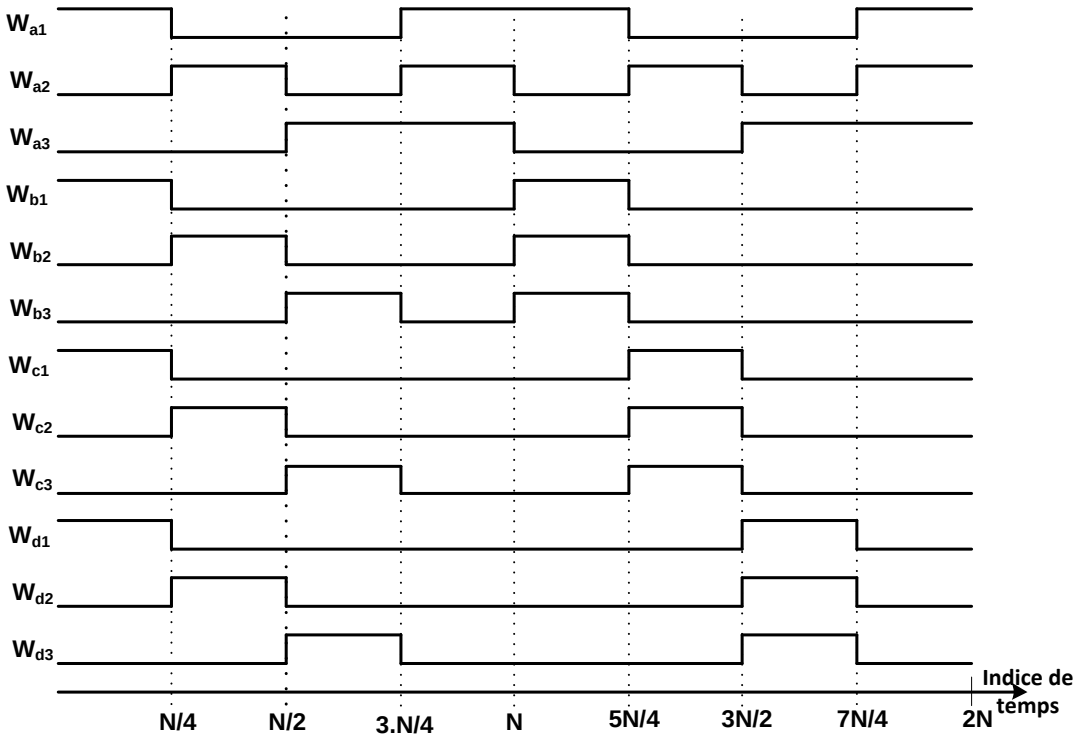

Figure 4.14 : Chronogrammes des signaux de lecture des *RAM*s de l'unité d'ordonnancement des entrées.

Après leurs réordonnancement, les séquences *A, B, C* et *D* sont injectées consécutivement dans les modules 1 à *m* afin de calculer leurs *DFT*s. Les modules Radix-2^3 et Radix-2^2 possèdent quatre chemins de traitement ce qui permet le traitement de quatre échantillons en parallèle. La structure interne du module Radix-2^2, lorsqu'il est utilisé pour les deux premiers étages du SFG (*i=1*), est donnée dans la figure 4.15. Ce module contient quatre additionneurs/soustracteurs, deux unités de réarrangement des échantillons, trois multiplieurs non-triviaux, un multiplieur trivial par *–j,* et quatre éléments de retard. Les unités de réarrangement des échantillons possèdent la même structure que celle donnée dans la figure 4.7, mais avec *D* éléments de délais pour chaque unité au lieu de *2.D* éléments pour l'architecture à deux chemins de traitement.

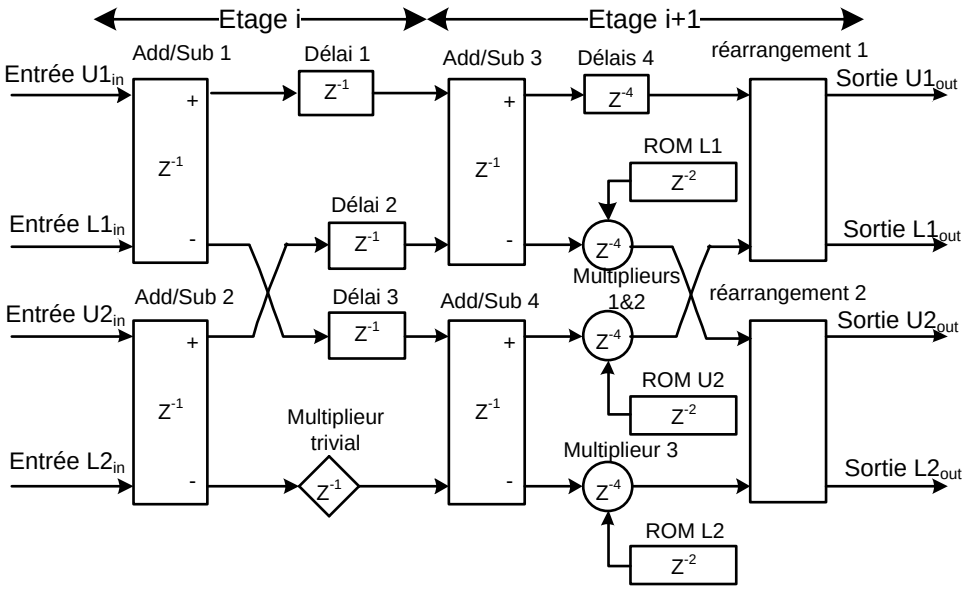

Figure 4.15. Schéma du module Radix-2^2 pour l'architecture *MR2^2-2^3-MDC* à quatre séquences utilisé pour les deux premiers étages (*i=1*).

Le schéma du module Radix-2^2 pour les étages intermédiaires (le schéma complet), et pour les deux derniers étages (uniquement le schéma à l'intérieur du carré en pointillé) est représenté dans la figure 4.16. Cette figure montre qu'il est indispensable d'insérer deux unités de réarrangement des échantillons entre les deux premiers additionneurs soustracteurs et les deux autres qui les succèdent.

Pour les deux figures 4.15 et 4.16, les facteurs de rotation $W_{N_i}^{2n}$, $W_{N_i}^{n}$ et $W_{N_i}^{3n}$, avec *n=0..N$_i$/4-1,* sont stockés dans *ROM D1*, *ROM U2* et *ROM D2* respectivement. Les trois *ROMs ;* chacune de taille *N$_i$/4 ;* sont adressées par les signaux C_0 à C_{n_s-i-2} issus d'un compteur binaire.

La figure 4.17 illustre la structure du module Radix-2^3 à quatre chemins de traitement pour les trois premiers étages du *SFG* (*i=1*). Ce module est constitué de six additionneurs/soustracteurs, six unités de réarrangement des échantillons, quatre multiplieurs non-triviaux avec quatre *ROMs* pour les facteurs de rotation, cinq éléments de retard, un additionneur trivial, et deux unités de multiplication complexes

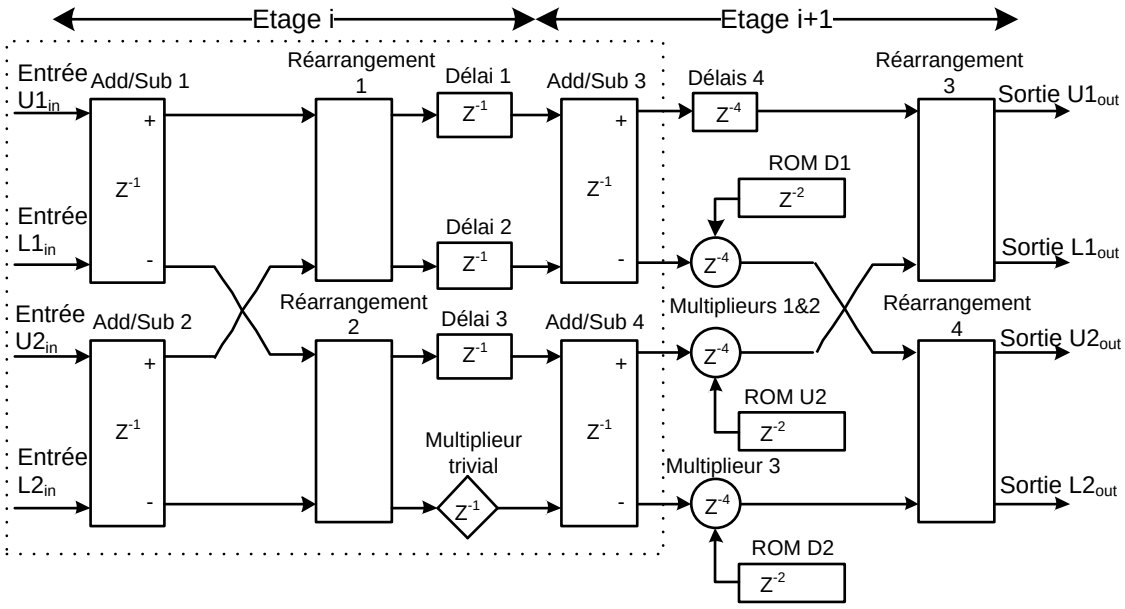

Figure 4.16. Schéma du module Radix-2^2 pour l'architecture *MR2^2-2^3-MDC* à quatre séquences utilisé pour les étages intermédiaire (schéma complet) et pour les deux derniers étages (schéma à l'intérieure du rectangle en pointillé).

par constante (*CCMU2* et *CCMU3*). L'unité *CCMU2* permet la multiplication par *1* et par W_8^1, et l'unité *CCMU3* réalise la multiplication par *–j* et par W_8^3.

Le schéma du module Radix-2^3 utilisé pour les étages intermédiaires (le schéma complet) et les derniers étages (schéma à l'intérieur du rectangle en pointillé) est donné dans la figure 4.18. Ce schéma nécessite deux unités de réarrangement après le premier étage d'addition/soustraction. En outre, le module Radix-2^3 pour les trois derniers étages ne contient pas de multiplieurs complexes non-triviaux ni des deux dernières unités de réarrangement.

Les facteurs de rotation ($W_{N_i}^0$,$W_{N_i}^n$), ($W_{N_i}^{4n}$,$W_{N_i}^{5n}$), ($W_{N_i}^{2n}$,$W_{N_i}^{3n}$) et ($W_{N_i}^{6n}$,$W_{N_i}^{7n}$), pour *n=0..N$_i$/8-1,* sont stockés dans *ROM U1*, *ROM D1, ROM U2* et *ROM D2* respectivement. Ces quatre *ROM*s ; chacune de taille *N$_i$/4 ;* sont adressées par les signaux C_0 à C_{n_s-i-2} d'un compteur binaire.

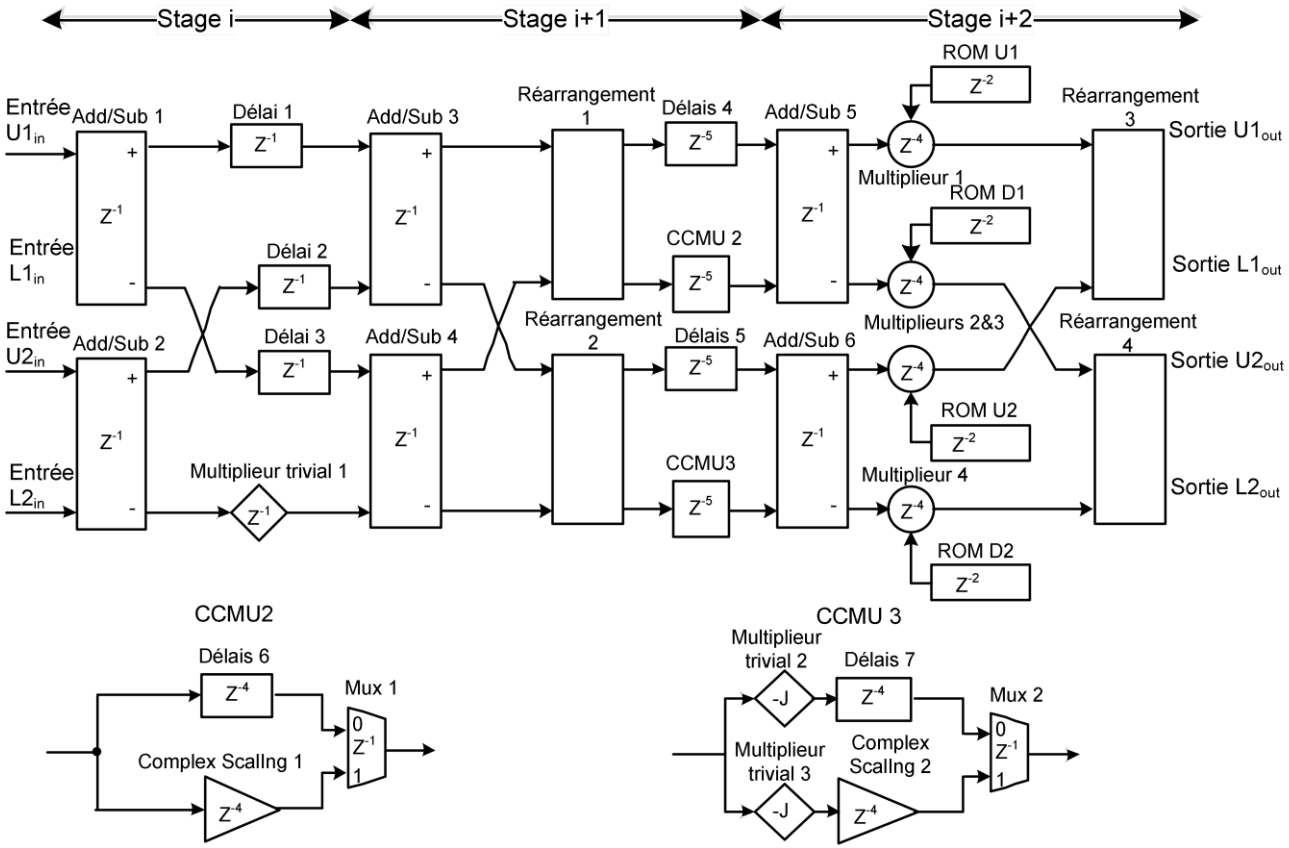

Figure 4.17 : Schéma du module Radix-2^3 pour l'architecture *MR2²-2³-MDC* à quatre séquences utilisé pour les trois premiers étages (*i=1*).

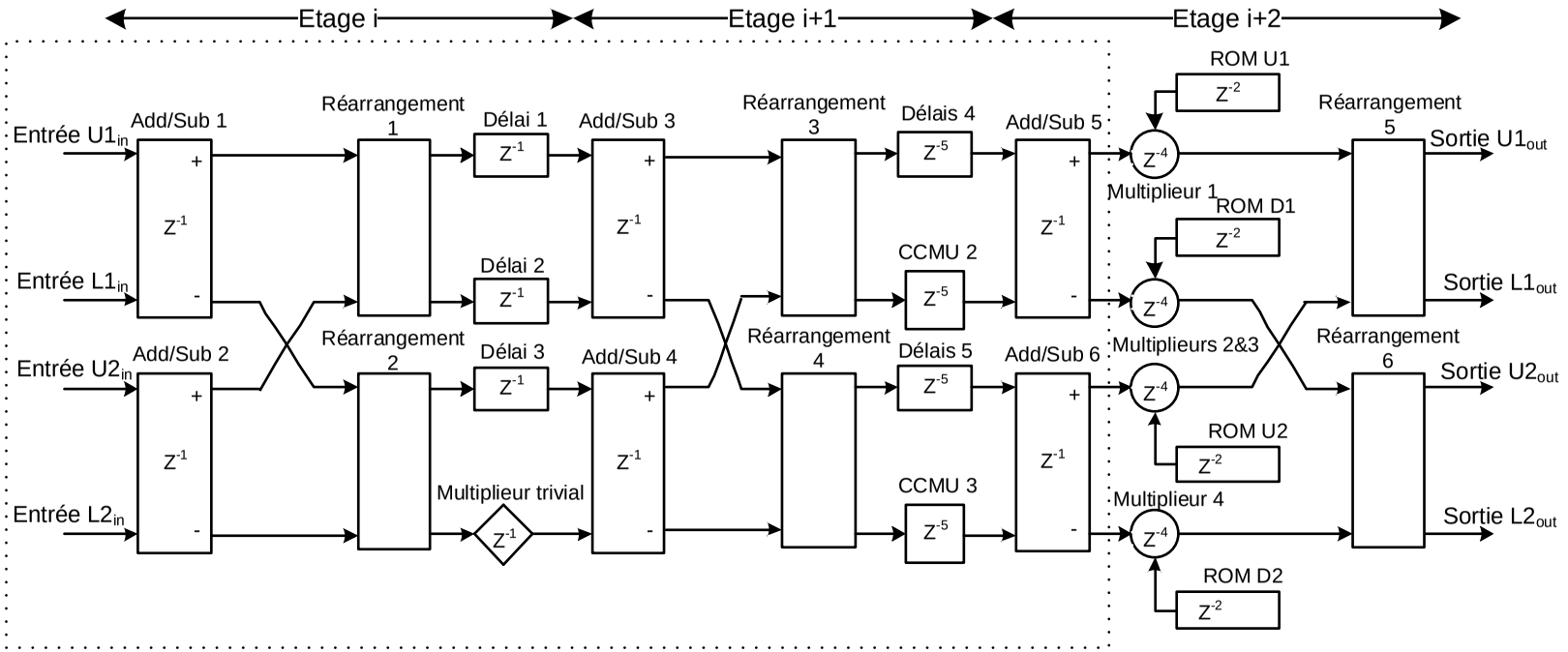

Figure 4.18 : Schéma du module Radix-2^3 pour l'architecture *MR2²-2³-MDC* à quatre séquences utilisé pour les étages intermédiaires (schéma complet) et les trois derniers étages (schéma à l'intérieur du rectangle en pointillé).

4.3.3. Complexité matérielle des architectures proposées :

La complexité matérielle d'un circuit ; qui désigne les ressources matérielles utilisées par ce circuit lors de son implémentation sur cible *FPGA* ou *ASIC* ; est un paramètre d'évaluation important. Les ressources les plus significatives sont les additionneurs/soustracteurs, les multiplieurs non-triviaux (multiplieurs standards) et les mémoires. Le tableau 4.2 présente les ressources matérielles utilisées par les architectures proposées en comparaison avec d'autres architectures existantes. Le paramètre *p* dans le tableau indique le nombre de séquences indépendantes qui sont traitées par le processeur, et le paramètre *φ* représente le nombre de chemins de traitement dans l'architecture. Le tableau montre que le nombre d'additionneurs/soustracteurs utilisés par les architectures *FFT* proposées dépend du nombre de chemins de traitement et de la taille *N* de la *DFT* indépendamment de la combinaison de modules Radix-2^2 et Radix-2^3. Ainsi, l'architecture pour quatre séquences nécessite le double d'additionneurs/soustracteurs que l'architecture pour deux séquences.

L'espace mémoire utilisé par les architectures proposées englobe celui des *RAM*s de l'unité de réordonnancement des entrées, celui des éléments de retard des unités de réarrangement des échantillons, en plus de celui des éléments de retard supplémentaires qui sont indispensable pour l'implémentation haut fréquence.

Pour l'architecture à deux séquences, l'unité de réordonnancement des entrées utilise deux *RAMs,* chacune de taille *N/2,* avec quatre retard supplémentaires pour garder la synchronisation. Les unités de réarrangement des échantillons utilisent des éléments de retard avec une taille totale égale à *N-2.* Un module Radix-2^3 utilise des éléments de retard de taille globale égale à 11, quel que soit sa position dans l'architecture, alors qu'un module Radix-2^2 nécessite 6 retard pour les positions *1* à *m-1,* et un seul retard pour les modules de position *m*. Donc, la taille totale de la mémoire utilisée par l'architecture à deux séquences est donnée par :

$$T_2 = 2.N + 2 + 6.m_2 + 11.m_3 - 5.k_1 \qquad (4.18)$$

Avec $k_1=1$ si le module *m* (dernier module) est un Radix-2^2 sinon $k_1=0$.

Tableau 4.2 Complexité matérielle des architectures *MR-2^2-2^3-MDC* à haut débit en comparaison avec d'autres architectures existantes.

Designs	**Type**	**Radix**	**N**	**P**	**φ**	**Additionneurs complexes**	**Taille de mémoire**	**Multiplieurs Complexes**	**Rotation par W_8**
Proposées	MDC	MR-2^2-2^3	2^n	2	2	$4\log_4 N$	$2N+2+(11m_3+6m_2-5k_1)^*$	$2(m-1)$	m_3
				4	4	$8\log_4 N$	$4N+2+(21m_3+7m_2-4k_1)^*$	$3m_2+4m_3+k_1-4$	$2m_3$
Lin *et al.* [99]	SDF	R-2 /R-8	64, 128	4	4	48	508	2+(4×0.62)	0
Fu *et al.* [104]	SDF/MDC	R2 /R4	64, 128	4	4	32	508	4	5
Yang *et al.* [102]	MDC	R4/ R8	128, 512, 1024, 2048	4	4	$8\log_4 N$	$3N+\sum_{s=1}^{\lfloor \log_4 N \rfloor -1} \frac{3N}{4^s}$	$3(\log_4 N-1)$	0
Garrido *et al.* [82], [83]	MDC	$R2^2$	4^n	1	2	$4\log_4 N$	N	$2(\log_4 N-1)$	0
					4	$8\log_4 N$	N	$3(\log_4 N-1)$	0
		$R2^3$	8^n	1	2	$4\log_4 N$	N	$3(\log_8 N-1)$	$\log_8 N$
					4	$8\log_4 N$	N	$4(\log_8 N-1)$	$2\log_8 N$

Pour l'architecture à quatre séquences, l'unité de réordonnancement des entrées nécessite 12 mémoires *RAM* chacune de taille *N/4 ;* en plus des 6 retards supplémentaires utilisés pour la synchronisation ; donnant une taille de *3N+6*. Les unités de réarrangement nécessitent des éléments de mémoire de taille totale égale à *N-4*. Chaque module Radix-2^3 nécessite 21 retards supplémentaires de synchronisation indépendamment de sa position dans l'architecture. Un module Radix-2^2 de position 1 à *m-1* utilise 7 retards de synchronisation, alors que celui de position *m* (dernier module) utilise uniquement 3 retards. Donc, la taille totale de la mémoire utilisée par l'architecture à quatre séquences est donnée par :

$$T_4 = 4.N + 2 + 7.m_2 + 21.m_3 - 4.k_1 \tag{4.19}$$

Chaque module de position 1 à *m-1* dans l'architecture à deux séquences nécessite un seul multiplieur complexe non-trivial indépendamment du Radix choisi. Cela implique que le nombre de multiplieurs complexes non-triviaux utilisés par l'architecture à deux séquences est égale à *2.(m-1).* Par ailleurs, chaque module de type Radix-2^3 dans l'architecture à deux séquences nécessite une seule rotation par W_8, réalisée à l'aide de l'unité *CCMU1* comme indiqué dans la figure 4.9. Par conséquent, le nombre total de rotations par la constante W_8 pour l'architecture à deux séquences est égal à m_3.

Pour l'architecture à quatre séquences, chaque module Radix-2^2 de position 1 à *m-1* nécessite 3 multiplieurs complexes non-triviaux, alors qu'un module Radix-2^3 de position 1 à *m-1* exige 4 multiplieurs complexes non-triviaux. Donc, le nombre de multiplieurs complexes non-triviaux utilisés par l'architecture à quatre séquences est donnée par :

$$N_{mult4} = 3.(m_2 - k_1) + 4.(m_3 + k_1 - 1) \tag{4.20}$$

De plus, chaque module Radix-2^3 nécessite deux rotations par W_8 effectuées par les deux unités *CCMU2* et *CCMU3* selon le schéma de la figure 4.17 ce qui rend le nombre total de rotations par W_8 égal à $2.m_3$.

En comparaison avec les autres architectures, le tableau 4.2 montre que les architectures proposées nécessitent le même nombre d'additionneurs/soustracteurs que les architectures proposées en [82], [83], et [102], et moins d'additionneurs/soustracteurs que les architectures proposées en [99] et [104]. En outre, les architectures proposées, et pour certaines combinaisons du Radix-2^2 et Radix-2^3 nécessitent moins de multiplieurs complexes non-triviaux que les architectures donnée en [82], [83], et [102]. Par exemple, pour la taille *N=256,* les processeurs présentés dans [82], [83], et [102] exigent 6 multiplieurs pour le cas de deux chemins de traitement et 9 multiplieurs pour le cas de quatre chemins, alors que les architectures proposées ne nécessitent que 4 multiplieurs pour le cas de deux chemins et 7 multiplieurs pour le cas de quatre chemins si on choisit la combinaison Radix-2^3-2^2-2^3. Cela signifie que les architectures proposées permettent de réduire le nombre de multiplieurs de 33% et de 22% pour deux et quatre chemins respectivement par rapport aux architectures présentées dans [82], [83], et [102]. L'architecture proposée par Lin *et al* dans [99] utilise deux multiplieurs standards et quatre autres multiplieurs à structure modifiée, dont chacun nécessite 0.62% de ressources d'un multiplieur standard. Ce circuit utilise donc un équivalent de 4.48 multiplieurs en comparaison avec 6 multiplieurs utilisés par la combinaison Radix-2^2-2^2-2^3 de notre architecture. Cependant le processeur proposé dans [99] est limité au traitement des séquences de taille 64 et 128 uniquement, alors que les architectures que nous avons proposées peuvent traiter des séquences de toutes les tailles exprimées en puissance de deux ($N = 2^{n_s}$).

L'espace mémoire utilisé par nos architectures est supérieur à celui utilisé par l'architecture proposée par Garrido *et al* dans [82], [83], cela est dû au fait que cette dernière permet de traiter uniquement une seule séquence au lieu de deux ou quatre séquences. On outre, l'espace mémoire utilisé par nos architectures est légèrement supérieur à celui utilisé par les architectures dans [99], [102] et [104], car les architectures proposées utilisent des éléments de retard supplémentaires qui sont indispensable pour l'implémentation haute fréquence.

4.4. Organigramme de conception

La figure 4.19 présente un organigramme simplifié du flot de conception suivi pour l'implémentation des architectures proposées.

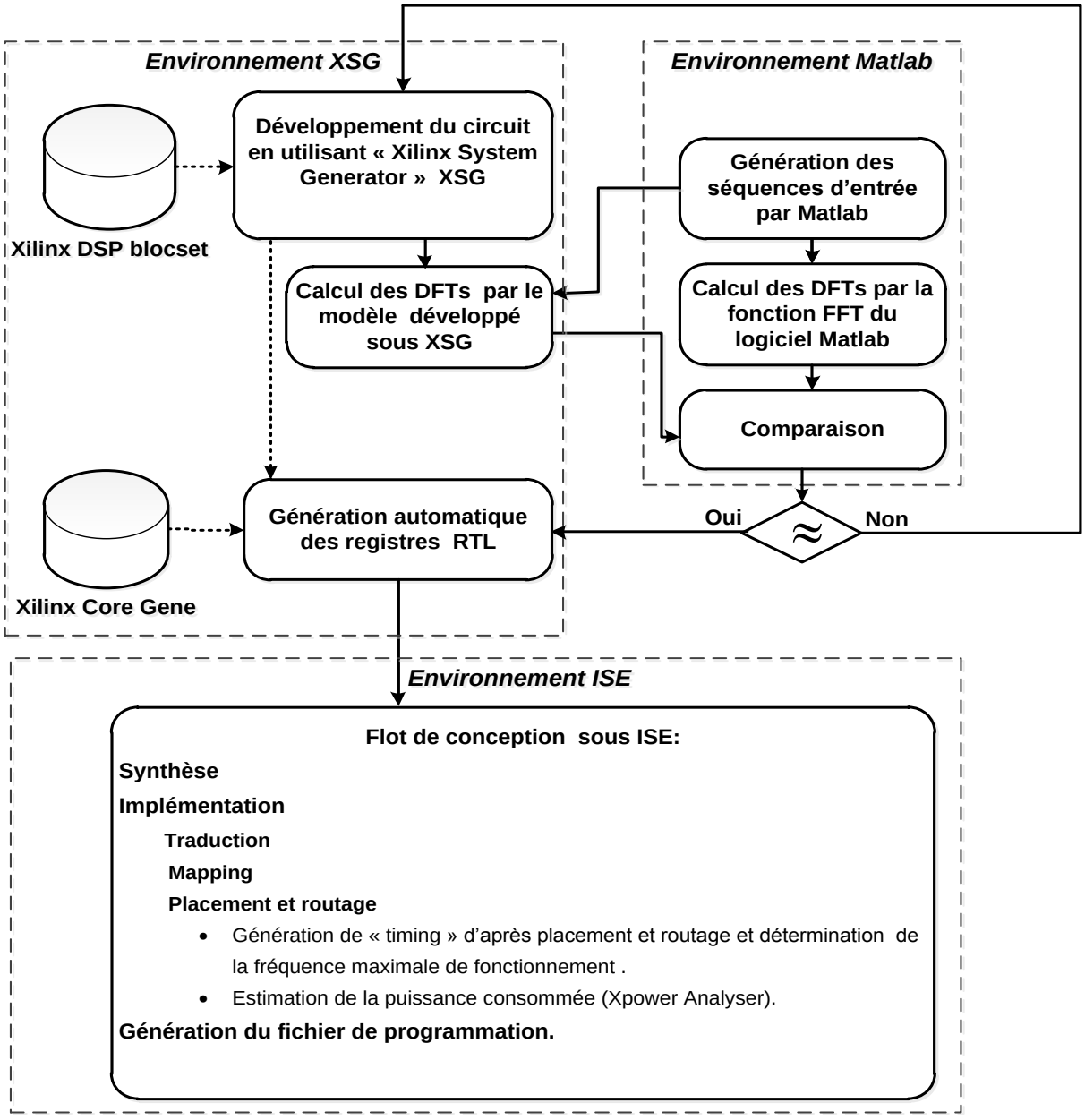

Figure 4.19 : Organigramme de conception pour les architectures proposées.

Les étapes principales de développement des architectures proposées sont les suivantes :

a)- *Génération des entrées par le logiciel Matlab* : un programme *Matlab* a été développé pour la génération de deux ou quatre séquences aléatoires complexes de taille allant de 8 à 1024.

b)- Conception des schémas : pour la conception des modèles *FFT* selon les architectures proposées, nous avons utilisé le logiciel « Xilinx System Generator » *XSG* [57], [58]. Une brève explication de cet outil a été fournie dans la section 2.5.3 du chapitre 2. Les architectures *MR2^2-2^3-MDC* pour deux séquences et pour quatre séquences ont été conçues pour les différentes valeurs de la taille *N,* et pour les différentes combinaisons de Radix-2^2 et de Radix-2^3. Le logiciel *XSG* utilise le format virgule fixe pour la représentation des données. Pour éviter le problème de saturation à la sortie des additionneurs /soustracteurs, nous avons introduit des opérations de mise en échelle en divisant par deux les sorties des additionneurs/soustracteurs des premiers étages de chaque processeur *FFT*. Notons que la division par deux est réalisée par une simple opération de décalage à droite.

c) Calcul des DFTs par le processeur FFT conçu sous XSG : les séquences complexes générées sous *Matlab* sont d'abord exportées vers *Simulink,* puis elles sont passées vers l'environnement *XSG*. Après lancement de la simulation, les modèles développés permettent de calculer les *DFTs* des séquences complexes. Les *DFTs* ainsi calculées sont exportées vers *Simulink* puis vers *Matlab*.

d)- Calcul des DFTs sous *MATLAB :* dans cette étape, les *DFTs* des séquences complexes sont calculées en utilisant la fonction *FFT* de *Matlab*. On note ici que la fonction *FFT* du logiciel *Matlab* utilise la virgule flottante pour la représentation des données, alors que les modèles développés sous *XSG* utilisent la représentation en virgule fixe. Ce qui signifie que les *DFTs* calculées en utilisant *Matlab* présentent une meilleure précision, et elles sont utilisées comme références de comparaison pour les *DFTs* calculés par les architectures proposées.

e)- Comparaison : après calcul des *DFTs* par *Matlab* et par les modèles proposés, on fait une comparaison par le calcul de la différence entre les deux *DFT*s obtenues. Cette différence représente l'erreur de quantification due à l'utilisation de la virgule fixe au lieu d'utiliser la virgule flottante.

f)-Génération du modèle *RTL* équivalent : après vérification du bon fonctionnement des modèles développés en utilisant le logiciel *XSG,* on passe à l'étape de la génération des modèles *RTL* « Register Transfert Level » pour les architectures proposées. Dans cette étape, on doit spécifier le langage *HDL* « Hardware Description Langage » à utiliser (Verilog ou *VHDL*), le type du circuit *FPGA* cible de l'implémentation et la période du signal horloge.

g) Conception sous environnement *ISE* : après génération des différents fichiers des modèles *RTL,* on passe à l'environnement *ISE* [61], afin de faire la synthèse, l'implémentation, l'estimation de la puissance consommée par le circuit, et la génération du fichier de configuration du circuit *FPGA*.

A la fin de l'étape d'implémentation, l'outil *ISE* permet d'offrir un rapport détaillé sur les différents résultats d'implémentation. Les résultats les plus significatifs sont les ressources utilisées « Device Utilisation Summary », les erreurs, les avertissements, les contraintes temporelles, et le rapport de « timing » après l'étape de placement et de routage « PAR timing report ».

L'estimation de la puissance est effectuée à l'aide du logiciel « XPower Analyzer » [62]. Cet outil permet l'estimation de la puissance statique et de la puissance dynamique consommée par les processeurs développés. En plus, il permet de fournir un rapport détaillé sur la puissance consommée par chaque type de ressources dans le circuit *FPGA* (DSP, Logic, IO…)

4.5. Présentation des résultats

La vérification des résultats de calcul des *DFTs* a été d'abord effectuée par simulation en utilisant l'outil *XSG*. Ensuite, nous avons utilisé la technique de co-simulation hardware [108] afin de vérifier les résultats de calcul sur un circuit *Virtex-7*

FPGA XC7VSX485T-2FFG1761 [59] en utilisant le kit d'évaluation *VC707* [109]. La co-simulation hardware permet de faire l'implémentation sur une plateforme *FPGA* d'une architecture qui est déjà conçu sous *XSG* Après l'implémentation sur *FPGA* de l'architecture, l'outil *XSG* offre la possibilité d'incorporer cette architecture, qui est en état de fonctionnement sur le circuit *FPGA*, directement dans une simulation sous Simulink. L'outil de compilation de *XSG* génère automatiquement le bitstream de l'architecture à implémenter, et il lui associe un bloc. La figure 4.20 montre le bloc nommé *FFT16_SYS_GEN* qui est conçu en utilisant l'outil XSG pour faire la simulation et le bloc nommé *VC_707_Virtex*_7 qui est généré à partir du bloc *FFT16_SYS_GEN* lors de la compilation pour gérer la co-simulation hardware.

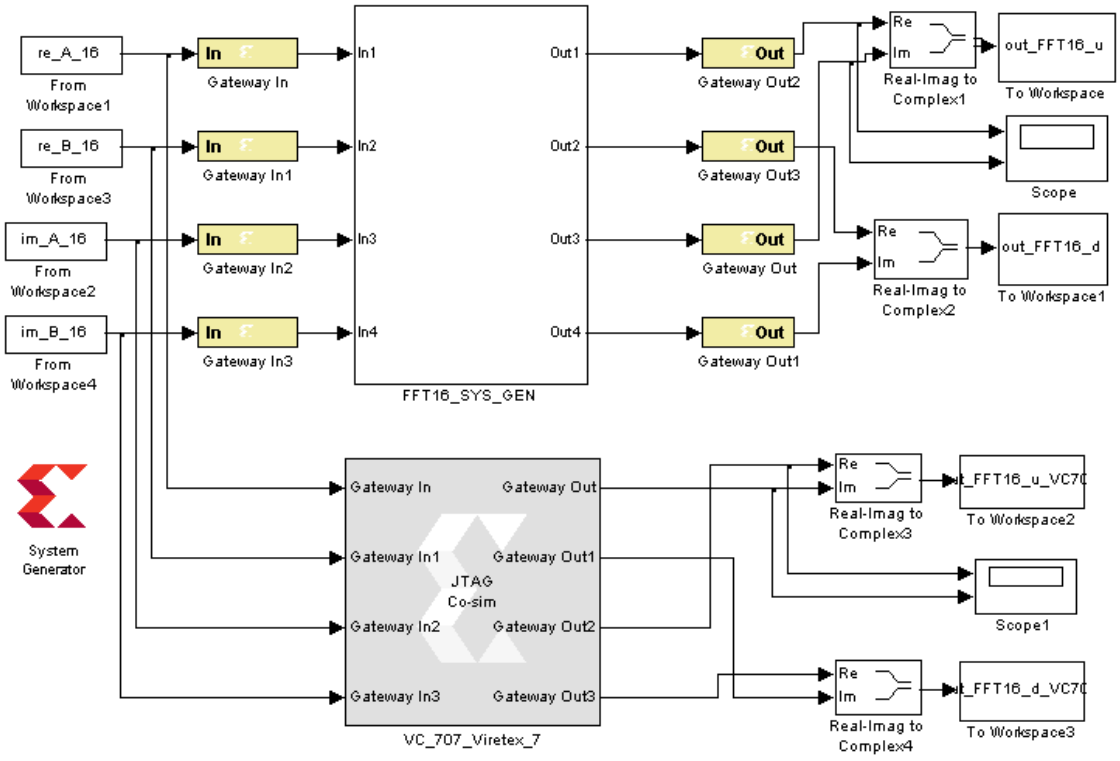

Figure 4.20 Les blocs utilisés pour la simulation (FFT16_SYS_GEN) et pour la co-simultion hardware (VC_707_Virtex_7).

Lorsque la simulation est lancée sous Simulink, l'outil *XSG* charge le bitstream dans le circuit *FPGA*. Ensuite, les calculs sont effectués dans le circuit *FPGA* et les résultats des calculs sont retournés vers l'environnement Simulink. Donc, la

technique de co-simulation hardware offre la possibilité de la vérification des architectures sur un circuit *FPGA* réel d'une façon simple, facile et rapide. La figure 4.21 est une image du kit *VC707* en état de calcul de la *FFT* pour deux séquences et en communication avec l'environnement Simulink à travers la liaison *JTAG* « **J**oint **T**est **A**ction **G**roup ».

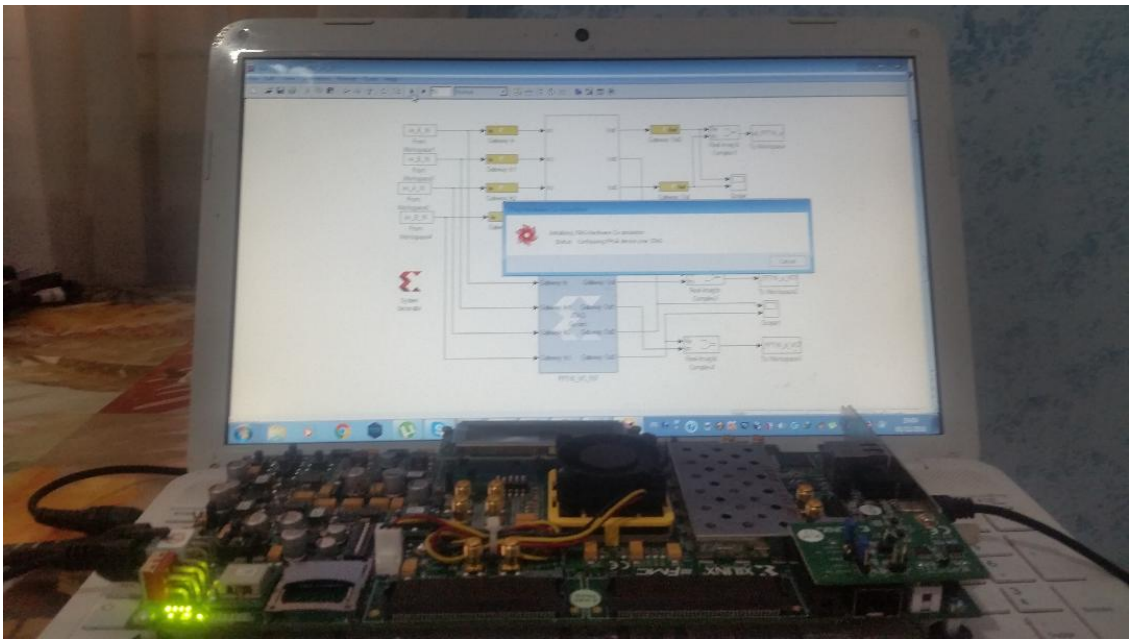

Figure 4.21 Image du kit VC707 en état d'exécution d'une co-simulation hardware pour une architecture *FFT* à deux séquences.

Nous avons effectué la co-simulation hardware pour les architectures à deux séquences et pour les architectures à quatre séquences pour les valeurs de *N* allant de 16 à 1024. Dans le tableau 4.3, nous avons reporté les résultats de calcul de la *DFT* d'une seule séquence complexe en utilisant une architecture *FFT* à deux séquences pour *N=16.* A partir du tableau on constate que les résultats obtenus par le modèle *XSG* sont identiques aux résultats obtenus par le circuit *FPGA* du kit VC_707. Dans la dernière colonne du tableau, nous avons présenté l'erreur de quantification qui est la différence entre la *DFT* calculée par la fonction *FFT* de Matlab et la *DFT* calculée en utilisant l'architecture proposée. A partir de cette colonne on remarque que l'erreur est de l'ordre de 10^{-4}, ce qui signifie qu'elle est négligeable.

Tableau 4.3 Résultats de simulation et de co-simulation hardware pour l'architecture FFT *MR2^2-R2^3-MDC* à deux séquences et pour N=16.

Entrée A	**DFT**			**Erreur**
	Par matlab	**Par XSG**	**Par VC_707**	
0.6294-0.6388i	3.5027+0.7317i	3.5022+0.7314i	3.5022+0.7314i	0.0005 +0.0003i
0.8116-0.9099i	4.2658-0.8351i	4.2656-0.8350i	4.2656-0.8350i	0.0002-0.0002i
-0.7460+0.4463i	1.6697-2.9096i	1.6689-2.9094i	1.6689-2.9094i	0.0008-0.0002i
0.8268-0.3051i	0.3433-3.9242i	0.3428-3.9241i	0.3428-3.9241i	0.0005-0.0002i
0.2647+0.3212i	-1.8587-0.2113i	-1.8596-0.2114i	-1.8596-0.2114i	0.0009+0.0002i
-0.8049-0.2323i	-1.3011-2.7060i	-1.3008-2.7058i	-1.3008-2.7058i	-0.0003-0.0002i
-0.4430+0.2547i	-4.2600-1.9626i	-4.2595-1.9622i	-4.2595-1.9622i	-0.0005-0.0005i
0.0938-0.9567i	-1.8415-1.8042i	-1.8416-1.8035i	-1.8416-1.8035i	0.0001-0.0007i
0.9150+0.8211i	-0.6024 +2.4959i	-0.6023+2.4961i	-0.6023+2.4961i	-0.0001-0.0002i
0.9297+0.6011i	3.7277-1.3118i	3.7275-1.3120i	3.7275-1.3120i	0.0002+0.0002i
-0.6848+0.4917i	0.1258 +6.2753i	0.1260+6.2747i	0.1260+6.2747i	-0.0002+0.0007i
0.9412+0.6262i	-0.6771+0.9366i	-0.6772+0.9358i	-0.6772+0.9358i	0.0001+0.0008i
0.9143- 0.2334i	3.8685-0.0577i	3.8684-0.0571i	3.8684-0.0571i	0.0001-0.0006i
-0.0293+0.2346i	0.3674-0.2664i	0.3672-0.2659i	0.3672-0.2659i	0.0002-0.0005i
0.6006+0.1510i	1.2729-0.4832i	1.2732-0.4822i	1.2732-0.4822i	-0.0003-0.0010i
-0.7162+0.0601i	1.4678-4.1881i	1.4685-4.1873i	1.4685-4.1873i	-0.0007-0.0008i

Afin de bien évaluer l'erreur de quantification nous avons procédé au calcul du rapport *SQNR* « Signal to Quantification Noise Ratio ». Le tableau 4.4 reporte l'évolution des SQNRs des quatre séquences d'entrée en fonction de la taille *N*. à partir du tableau, nous constatons que les architectures permettent d'assurer un *SQNR* très élevé ce qui signifie que l'erreur de quantification influe légèrement sur les performances du system. En outre, nous remarquons que le SQNR diminue avec l'augmentation de la taille *N*.

Tableau 4.4 Evolution du SQNR pour les architectures *MR2^2-R2^3-MDC* à quatre séquences.

N	16	32	64	128	256	512	1024
$SQNR_A$	71.1	68.8	68.4	65.9	65.0	61.7	61.4
$SQNR_B$	71.4	68.8	68.5	66.3	65.1	61.8	61.6
$SQNR_C$	71.2	68.8	68.1	66.2	65.2	61.9	61.5
$SQNR_D$	71.7	67.2	68.1	66.0	65.2	61.7	61.3

Les architectures développées ont été évaluées par implémentation en utilisant l'outil *ISE* pour les différentes valeurs de la taille *N* allant de *16* à *1024*. De plus, l'utilisation de la technique de Radix mixés permet la réalisation du processeur *FFT* par plusieurs combinaisons pour chaque valeur de la taille *N.* Par exemple, un processeur *FFT* de taille *N=256* peut être réalisé par un module Radix-2^3 suivi d'un module Radix-2^2 suivi à son tour d'un module Radix-2^3, ce qui donne la combinaison (2^3-2^2-2^3). Le même processeur peut être réalisé par la combinaison (2^3-2^3-2^2) ou par la combinaison (2^2-2^2-2^2-2^2)

Les blocs de *RAM* (BRAM) sont utilisés pour l'implémentation des *RAMs* de l'unité d'ordonnancement des entrées, et des *ROMs* de stockage des facteurs de rotation. Les éléments de retard utilisés dans les unités de réarrangement des échantillons, et ceux utilisés pour garder la synchronisation sont implémentés à l'aide de ressources *SRL* «Shift Register Look-up table ». Les multiplieurs standards non-triviaux sont implémentés par les ressources *FPGA* de type *DSP48E*. Tous les autres composants des architectures sont implémentés par des ressources logiques de type Slices standards du circuit *FPGA*. Les processeurs *FFT* développés ont été implémentés sur les deux circuits F*PGA* Virtex-7 *FPGA XC7VSX485T-2FFG1761* [59], [60] et Virtex-5 *FPGA XC5VSX240T-2FF1738* [110], et ils sont évalués en termes de ressources *FPGA* utilisées, de la fréquence maximale qui est en relation directe avec le débit et de la puissance dynamique consommée.

Les deux tableaux 4.5 et 4.6 donnent les ressources *FPGA* consommées par les architectures fonctionnant à haute fréquence, pour deux et pour quatre séquences respectivement. Les ressources sont comparées avec celles consommées par des architectures équivalentes qui fonctionnent avec des fréquences basses. Notant que les architectures à basses fréquences n'utilisent aucun registre de pipeline pour améliorer la fréquence de fonctionnement, ce qui signifie que les composants de cette architecture ne présentent aucun retard de réponse exprimé en nombre de cycles horloge.

Pour faire la comparaison entre les architectures à haute fréquence et celle à basse fréquence, un rapport de ressources utilisées « Resource Utilisation Report »

RUR a été calculé pour chaque cas d'implémentation. Pour chaque type de ressource, le *RUR* est le rapport entre le nombre de ressources utilisées par l'architecture à haute fréquence et le nombre de ressources utilisées par l'architecture à basse fréquence. Les deux tableaux 4.5 et 4.6 montrent que le rapport *RUR* ne dépasse pas dans le pire des cas la valeur *1.41* pour l'architecture à deux séquences et *1.74* pour l'architecture à quatre séquences. Ce qui signifie que la grande amélioration du débit n'est pas pénalisée par une augmentation excessive des ressources utilisées. A partir des deux tableaux, on peut constater que l'augmentation du débit de traitement ne demande aucune augmentation de nombre de *BRAM* ou de *DSP48E*. On outre, les résultats montrent que l'utilisation du Radix mixé offre une

Tableau 4.5. Ressources utilisées par les architectures *MR2^2-R2^3-MDC* à deux séquences implémentées sur le circuit *Virtex-5 FPGA XC5VSX240T-2FF1738*

N	Structure	Slice LUTs			Slices			BRAM	DSP48E
		Basse Freq.	Haute Freq.	RUR	Basse Freq.	Haute Freq.	RUR		
16	2^2-2^2	674	787	1.17	279	347	1.24	8	8
32	2^3-2^2	1071	1279	1.19	400	501	1.25	8	8
	2^2-2^3	1070	1342	1.25	391	535	1.37	8	8
64	2^2-2^2-2^2	1100	1222	1.11	413	534	1.29	12	16
	2^3-2^3	1467	1840	1.25	553	674	1.22	8	8
128	2^2-2^2-2^3	1519	1850	1.22	537	684	1.27	12	16
	2^3-2^2-2^2	1527	1794	1.17	557	679	1.22	12	16
256	2^2-2^2-2^2-2^2	1749	1927	1.10	685	760	1.11	16	24
	2^3-2^2-2^3	2102	2516	1.20	709	906	1.28	12	16
	2^3-2^3-2^2	2090	2485	1.19	703	851	1.21	12	16
512	2^3-2^3-2^3	2887	3497	1.21	1012	1280	1.26	12	16
	2^3-2^2-2^2-2^2	2547	2879	1.13	937	1141	1.22	16	24
	2^2-2^2-2^2-2^3	2543	2932	1.15	781	1102	**1.41**	16	24
1024	2^2-2^2-2^2-2^2-2^2	3465	3717	1.07	1089	1372	1.26	20	32
	2^3-2^3-2^2-2^2	3799	4287	1.13	1171	1386	1.18	16	24
	2^3-2^2-2^2-2^3	3798	4343	1.14	1247	1655	1.33	16	24

Tableau 4.6 : Ressources utilisées par les architectures *MR2^2-R2^3-MDC* à quatre séquences implémentées sur le circuit Virtex-*5 FPGA XC5VSX240T-2FF1738*

N	Structure	Slice LUTs			Slices			BRAM	DSP48E
		Basse Freq.	Haute Freq.	RUR	Basse Freq.	Haute Freq.	RUR		
16	2^2-2^2	1249	1355	1.08	508	619	1.22	30	12
32	2^3-2^2	1956	2301	1.18	693	991	1.43	32	16
	2^2-2^3	1938	2327	1.20	751	931	1.24	30	12
64	2^2-2^2-2^2	1920	2059	1.07	633	895	1.41	36	24
	2^3-2^3	2637	3294	1.25	924	1284	1.39	32	16
128	2^2-2^2-2^3	2607	3052	1.17	992	1123	1.13	36	24
	2^3-2^2-2^2	2632	3002	1.14	935	1262	1.35	38	28
256	2^2-2^2-2^2-2^2	2762	2914	1.06	629	1095	**1.74**	42	36
	2^3-2^2-2^3	3439	4152	1.21	1256	1744	1.39	38	28
	2^3-2^3-2^2	3426	4125	1.20	1222	1518	1.24	40	32
512	2^3-2^3-2^3	4530	5443	1.20	1656	2067	1.25	40	32
	2^3-2^2-2^2-2^2	3889	4256	1.09	1378	1596	1.16	44	40
	2^2-2^2-2^2-2^3	3859	4283	1.11	1607	1554	0.97	42	36
1024	2^2-2^2-2^2-2^2-2^2	4932	4915	1.00	1850	1659	0.90	48	48
	2^3-2^3-2^2-2^2	5470	6133	1.12	2063	2342	1.14	46	44
	2^3-2^2-2^2-2^3	5483	6192	1.13	1897	2433	1.28	44	40

grande flexibilité vis-à-vis des ressources *FPGA* disponible. Par exemple, un processeur *FFT* de taille 256 utilise 42 *BRAM*s et 36 *DSP48E* pour la combinaison 2^2-2^2-2^2-2^2 alors que le processeur pour la même taille utilise 38 *BRAM* et 28 *DSP48E* pour la combinaison 2^3-2^2-2^3. Il est important de noter ici que la surface *FPGA* moyenne occupée « FPGA's occupied area » par le circuit ; qui est la moyenne des pourcentages d'occupation des *BRAM*s, slices et *DSP48E ;* varie de 0.82% à 2.88% pour l'architecture à deux séquences et de 1.89% à 4.96% pour l'architecture à

quatre séquences. Par conséquent, les circuits proposés occupent une très petite portion de la surface totale du circuit *FPGA.* La surface restante peut être exploitée pour l'implémentation d'autres modules du système de télécommunications.

Les performances des architectures à haute fréquence sont comparées avec ceux des architectures équivalentes à basse fréquence dans les deux tableaux 4.7 et 4.8.

Tableau 4.7 : Performances de l'architecture *MR2^2-R2^3-MDC* à deux séquences implémentée sur le circuit Virtex-5 *FPGA XC5VSX240T-2FF1738*

N	Structure	Fréquence (MHz)		Débit (Ms/S)			Latence (ns)	
		Basse Freq.	Haute Freq.	Basse Freq.	Haute Freq.	TR	Basse Freq.	Haute Freq.
16	2^2-2^2	62.8	458	125.6	**916**	7.29	127	48
32	2^3-2^2	45.5	455	91.0	910	10.00	352	81
	2^2-2^3	62.6	447	125.2	894	7.14	256	85
64	2^2-2^2-2^2	41.7	444	83.4	888	10.65	767	126
	2^3-2^3	43.5	439	87.0	878	10.09	736	139
128	2^2-2^2-2^3	40.0	442	80.0	884	11.05	1600	217
	2^3-2^2-2^2	30.3	439	60.6	878	14.49	2112	216
256	2^2-2^2-2^2-2^2	30.3	442	60.6	884	14.59	4224	367
	2^3-2^2-2^3	30.3	436	60.6	872	14.39	4224	383
	2^3-2^3-2^2	25.0	440	50.0	880	17.60	5120	377
512	2^3-2^3-2^3	25.0	445	50.0	890	17.80	10240	679
	2^3-2^2-2^2-2^2	24.4	440	48.8	880	18.03	10492	675
	2^2-2^2-2^2-2^3	29.4	440	58.8	880	14.97	8707	677
1024	2^2-2^2-2^2-2^2-2^2	23.8	429	47.6	858	18.03	21513	1296
	2^3-2^3-2^2-2^2	20.7	431	41.4	862	**20.82**	24734	1299
	2^3-2^2-2^2-2^3	23.9	433	47.8	866	18.12	21423	1293

La fréquence, la latence et le débit de traitement sont reportés dans les deux tableaux 4.7 et 4.8. La latence est le temps entre l'instant de début de traitement et l'instant où le premier échantillon de la *DFT* est disponible à la sortie du processeur. Le débit de traitement est égal au nombre d'échantillons traités par le processeur par unité de temps. Il est exprimé en Mega échantillons par second « Mega samples per Second » Ms/S, et il est égal à deux fois la fréquence pour l'architecture à deux chemins et à quatre fois la fréquence pour l'architecture à quatre chemins.

Tableau 4.8 *:* Performances de l'architecture *MR2^2-R2^3-MDC* à quatre séquences implémentée sur le circuit Virtex-*5 FPGA XC5VSX240T-2FF1738*

N	Structure	Fréquence (MHz)			Débit (Ms/S)		Latence (ns)	
		Basse Freq.	Haute Freq.	TR	Basse Freq.	Haute Freq.	Basse Freq.)	Haute Freq.
16	2^2-2^2	62.6	439	7.01	250.4	**1756**	64	36
32	2^3-2^2	46.6	424	9.10	186.4	1696	172	64
	2^2-2^3	56.4	422	7.48	225.6	1688	142	64
64	2^2-2^2-2^2	55.7	421	7.56	222.8	1684	287	88
	2^3-2^3	43.6	417	9.56	174.4	1668	367	101
128	2^2-2^2-2^3	50.1	417	8.32	200.4	1668	639	144
	2^3-2^2-2^2	33.4	411	12.31	133.6	1644	958	146
256	2^2-2^2-2^2-2^2	48.9	404	8.26	195.6	1616	1309	233
	2^3-2^2-2^3	33.4	413	12.37	133.6	1652	1916	240
	2^3-2^3-2^2	26.3	417	15.86	105.2	1668	2433	237
512	2^3-2^3-2^3	25.0	409	16.36	100.0	1636	5120	416
	2^3-2^2-2^2-2^2	29.4	412	14.01	117.6	1648	4354	400
	2^2-2^2-2^2-2^3	47.7	417	8.74	190.8	1668	2683	396
1024	2^2-2^2-2^2-2^2-2^2	42.6	400	9.39	170.4	1600	6009	738
	2^3-2^3-2^2-2^2	21.5	411	**19.12**	86.0	1644	11907	730
	2^3-2^2-2^2-2^3	29.5	405	13.73	118	1620	8678	741

Afin de mettre en évidence la comparaison entre les architectures à haut débit proposées et leurs équivalents à basse fréquence, le rapport de débit *TR* « Through-put Ratio » a été calculé. Les deux tableaux montrent que les architectures à haute fréquence portent une amélioration très significative du débit de traitement. Ce débit peut atteindre 916 Ms/S pour l'architecture à deux séquences et 1756 Ms/S pour l'architecture à quatre séquences. En termes de comparaison avec les architectures à basse fréquence, le rapport d'amélioration atteint jusqu'à ***20.82*** pour l'architecture à deux séquences et jusqu'à ***19.12*** pour l'architecture à quatre séquences. Ce qui est très important, c'est que cette grande amélioration de débit est accompagnée par seulement une légère augmentation de ressources *FPGA* utilisées. Cela peut être directement déduit en comparant les valeurs du paramètre *RUR* des deux tableaux 4.5 et 4.6 avec les valeurs du rapport *TR* données dans les deux tableaux 4.7 et 4.8 respectivement. Par conséquent, les architectures à haute fréquence proposées sont très efficaces et assurent un très bon rapport débit de traitement à ressources *FPGA* utilisées.

La puissance dynamique consommée par les architectures *MR2^2-R2^3-MDC* à haute fréquence est comparée avec celle consommée par les architectures équivalentes à basse fréquence. Les résultats de comparaison, pour les deux architectures à deux séquences et à quatre séquences, sont présentés dans les deux tableaux 4.9 et 4.10 respectivement. Comme prévu, la puissance dynamique consommée par les architectures à haute fréquence est supérieure à celle consommée par les architectures à basse fréquence. En effet, la théorie et les résultats expérimentaux montrent que la puissance dynamique consommée par un processeur implémenté sur *FPGA* ou *ASIC* est proportionnelle à la fréquence de fonctionnement du processeur. Pour faire une comparaison plus significative de la consommation de puissance des architectures à haute fréquence et leurs équivalents à basse fréquence, le rapport puissance dynamique à débit de traitement « Dynamic Power to Throughput Ratio » *DPTR* a été calculé et reporté dans les deux tableaux 4.9 et 4.10. Les résultats montrent que le *DPTR* des architectures à haute fréquence est légèrement supérieur à celui de leurs équivalents à basse fréquence pour les tailles *N* inférieures à 64. Au-delà de la taille 64, le *DPTR* des architectures à haute

fréquence est inférieur à celui de leurs équivalents à basse fréquence. Par conséquent, il n'y a pas de puissance supplémentaire consommée lorsqu'on utilise un seul processeur à haute fréquence au lieu d'utiliser plusieurs processeurs à basse fréquence pour assurer le même débit de traitement. On peut même avoir un gain de puissance pour les processeurs *FFT* de taille supérieure à 64, ce qui est le cas de la plupart des standards pour les systèmes des télécommunications qui utilisent l'*OFDM*.

Tableau 4.9 : Puissance dynamique consommée par l'architecture *MR2^2-R2^3-MDC* à deux séquences implémentée sur le circuit Virtex-5 *FPGA XC5VSX240T-2FF1738*

N	**Structure**	**Basse fréquence**		**Haute Fréquence**	
		Puissance dynamique (W)	***DPTR***	**Puissance dynamique (W)**	***DPTR***
16	2^2-2^2	0.128	1.02	1.079	1.18
32	2^3-2^2	0.110	1.21	1.095	1.20
	2^2-2^3	0.147	1.17	1.145	1.28
64	2^2-2^2-2^2	0.107	1.28	1.162	1.31
	2^3-2^3	0.111	1.28	1.176	1.34
128	2^2-2^2-2^3	0.106	1.33	1.180	1.33
	2^3-2^2-2^2	0.087	**1.44**	1.186	**1.35**
256	2^2-2^2-2^2-2^2	0.091	**1.50**	1.101	**1.25**
	2^3-2^2-2^3	0.092	**1.52**	1.044	**1.20**
	2^3-2^3-2^2	0.077	**1.54**	0.968	**1.10**
512	2^3-2^3-2^3	0.089	**1.78**	1.120	**1.26**
	2^3-2^2-2^2-2^2	0.087	**1.78**	1.190	**1.35**
	2^2-2^2-2^2-2^3	0.095	**1.62**	1.227	**1.39**
1024	2^2-2^2-2^2-2^2-2^2	0.092	**1.93**	1.345	**1.57**
	2^3-2^3-2^2-2^2	0.084	**2.03**	1.291	**1.50**
	2^3-2^2-2^2-2^3	0.088	**1.84**	1.364	**1.58**

Tableau 4.10 : Puissance dynamique consommée par l'architecture *MR2^2-R2^3-MDC* à quatre séquences sur le circuit Virtex-5 *FPGA XC5VSX240T-2FF1738.*

N	**Structure**	**Basse fréquence**		**Haute Fréquence**	
		Puissance dynamique (W)	***DPTR***	**Puissance dynamique (W)**	***DPTR***
16	2^2-2^2	0.295	1.18	2.16	1.23
32	2^3-2^2	0.236	1.27	2.251	1.33
	2^2-2^3	0.253	1.12	2.225	1.32
64	2^2-2^2-2^2	0.278	1.25	2.239	1.33
	2^3-2^3	0.225	1.29	2.346	1.41
128	2^2-2^2-2^3	0.264	**1.32**	2.033	**1.22**
	2^3-2^2-2^2	0.183	**1.37**	1.817	**1.11**
256	2^2-2^2-2^2-2^2	0.261	**1.33**	1.897	**1.17**
	2^3-2^2-2^3	0.195	**1.46**	2.069	**1.25**
	2^3-2^3-2^2	0.159	**1.51**	1.811	**1.09**
512	2^3-2^3-2^3	0.165	**1.65**	2.107	**1.29**
	2^3-2^2-2^2-2^2	0.182	**1.55**	2.071	**1.26**
	2^2-2^2-2^2-2^3	0.284	**1.49**	2.191	**1.31**
1024	2^2-2^2-2^2-2^2-2^2	0.270	**1.58**	2.132	**1.33**
	2^3-2^3-2^2-2^2	0.160	**1.86**	1.708	**1.04**
	2^3-2^2-2^2-2^3	0.201	**1.70**	2.113	**1.30**

Dans le but de connaître les améliorations en performances, en ressources et en puissance que peut porter l'utilisation d'un circuit *FPGA* récent comme cibles d'implémentation, nous avons aussi implémenté les architectures proposées sur un circuit *FPGA* récent de type Virtex-7 FPGA *XC7VSX485T-2FFG1761*. Le tableau 4.11 donne les ressources consommées par les architectures à haut débit pour deux et pour quatre séquences implémentées sur le circuit *FPGA* Virtex-7.

Tableau 4.11 : Ressources utilisées par les architectures *MR2^2-R2^3-MDC* à haut débit implémentées sur le circuit Virtex-7 *FPGA XC7VSX485T-2FFG1761.*

N	Structure	Deux séquences		Quatre séquences	
		Slice LUTs	Slices	Slice LUTs	Slices
16	$2^2\_2^2$	806	278	1575	606
32	$2^3\_2^2$	1277	391	2534	850
	$2^2\_2^3$	1340	413	2527	882
64	$2^2\_2^2\_2^2$	1232	388	2258	814
	$2^3\_2^3$	1788	539	3466	1141
128	$2^2\_2^2\_2^3$	1824	556	3236	1059
	$2^3\_2^2\_2^2$	1770	529	3190	1054
256	$2^2\_2^2\_2^2\_2^2$	1911	587	3117	1033
	$2^3\_2^2\_2^3$	2461	722	4334	1352
	$2^3\_2^3\_2^2$	2418	706	4290	1367
512	$2^3\_2^3\_2^3$	3408	958	5559	1694
	$2^3\_2^2\_2^2\_2^2$	2823	828	4418	1404
	$2^2\_2^2\_2^2\_2^3$	2886	858	4435	1405
1024	$2^2\_2^2\_2^2\_2^2\_2^2$	3719	1052	5063	1569
	$2^3\_2^3\_2^2\_2^2$	4192	1182	6240	1920
	$2^3\_2^2\_2^2\_2^3$	4271	1187	6307	1872

Les résultats montrent que le nombre de *LUTs* utilisés par le circuit Virtex-7 est presque le même que celui utilisé par le circuit *Virtex-5*. Cependant, le nombre de Slices consommés par le circuit *FPGA Virtex-7* est inférieur au nombre de Slices consommés par le circuit *Virtex-5*. Donc, l'utilisation du *Virtex-7* permet la réduction du nombre de Slices utilisés en comparaison avec le *Virtex-5*. On note que les deux circuits *FPGA* (Virtex-5 et Virtex-7) utilisent le même nombre de *BRAM* et le même nombre de *DSP48E*.

Les performances en termes du débit et de la latence des architectures à haute fréquence implémentées sur le circuit *Virtex-7* sont reportées dans le tableau 4.12.

Tableau 4.12 : Performances des architectures *MR2^2-R2^3-MDC* implémentées sur le circuit Virtex-7 *FPGA XC7VSX485T-2FFG1761738*.

N	Structure	Deux séquences		Quatre séquences	
		Débit (M.s/S)	**Latence (ns)**	**Débit (M.s/S)**	**Latence (ns)**
16	$2^2\_2^2$	954	46	1908	34
32	$2^3\_2^2$	954	78	1884	57
	$2^2\_2^3$	954	80	1908	57
64	$2^2\_2^2\_2^2$	954	117	1908	78
	$2^3\_2^3$	954	128	1900	88
128	$2^2\_2^2\_2^3$	954	201	1908	126
	$2^3\_2^2\_2^2$	954	199	1904	126
256	$2^2\_2^2\_2^2\_2^2$	954	340	1908	197
	$2^3\_2^2\_2^3$	954	350	1892	209
	$2^3\_2^3\_2^2$	954	348	1888	210
512	$2^3\_2^3\_2^3$	954	633	1900	358
	$2^3\_2^2\_2^2\_2^2$	954	623	1896	348
	$2^2\_2^2\_2^2\_2^3$	954	625	1900	347
1024	$2^2\_2^2\_2^2\_2^2\_2^2$	954	1166	1888	625
	$2^3\_2^3\_2^2\_2^2$	954	1174	1896	633
	$2^3\_2^2\_2^2\_2^3$	954	1174	1888	636

Le tableau 4.12 montre que l'implémentation sur le circuit *FPGA Virtex-7* permet d'améliorer le débit de traitement et de réduire la latence du processeur par rapport à l'implémentation sur le circuit *FPGA* Virtex-5 pour les deux architectures à deux séquences et quatre séquences. Cette amélioration devient plus significative pour les grandes valeurs de la taille *N*.

La puissance dynamique consommée par les architectures proposées implémentées sur le circuit *FPGA* Virtex-7 est donnée dans le tableau 4.13. Pour comparer avec le cas d'implémentation sur le circuit *FPGA Virtex-5,* le rapport *DPTR* a été aussi calculé. Les résultats du tableau 4.13 montrent que l'implémentation sur le circuit *Virtex-7* permet de réduire la puissance consommée et le rapport *DPTR* à presque la moitié par rapport à l'implémentation sur le circuit *Virtex-5*.

Tableau 4.13 : Puissance dynamique consommée par les architectures *MR2^2-R2^3-MDC* implémentées sur le circuit Virtex-7 *FPGA XC7VSX485T-2FFG1761738*.

N	Structure	Deux séquences		Quatre séquences	
		Puissance dynamique (W)	**DPTR (mW/M.s.S^{-1})**	**Puissance dynamique (W)**	**DPTR (mW/M.s.S^{-1})**
16	$2^2\_2^2$	0.508	0.53	1.097	0.57
32	$2^3\_2^2$	0.511	0.54	1.173	0.62
	$2^2\_2^3$	0.528	0.55	1.162	0.61
64	$2^2\_2^2\_2^2$	0.575	0.60	1.249	0.65
	$2^3\_2^3$	0.557	0.58	1.245	0.66
128	$2^2\_2^2\_2^3$	0.605	0.63	1.201	0.63
	$2^3\_2^2\_2^2$	0.604	0.63	1.250	0.66
256	$2^2\_2^2\_2^2\_2^2$	0.654	0.69	1.328	0.70
	$2^3\_2^2\_2^3$	0.592	0.62	1.310	0.69
	$2^3\_2^3\_2^2$	0.615	0.64	1.340	0.71
512	$2^3\_2^3\_2^3$	0.668	0.70	1.492	0.79
	$2^3\_2^2\_2^2\_2^2$	0.743	0.78	1.483	0.78
	$2^2\_2^2\_2^2\_2^3$	0.721	0.76	1.426	0.75
1024	$2^2\_2^2\_2^2\_2^2\_2^2$	0.861	0.90	1.620	0.86
	$2^3\_2^3\_2^2\_2^2$	0.508	0.93	1688	0.89
	$2^3\_2^2\_2^2\_2^3$	0.511	0.87	1.572	0.83

Cette réduction de la puissance dynamique consommée est très importante pour l'évaluation des processeurs implémentés sur FPGA, surtout pour les appareils qui utilisent la batterie comme source d'alimentation, afin d'augmenter la durée de leur autonomie. En effet, l'augmentation de la durée de l'autonomie de la batterie est un défi pour les concepteurs des appareils de télécommunication portables qui fonctionnent à très haut débit.

La figure 4.22 met en évidence la comparaison ; en termes de l'évolution de la fréquence de fonctionnement en fonction de la taille *N* ; entre les deux cibles d'implémentation pour les architectures à deux et à quatre séquences.

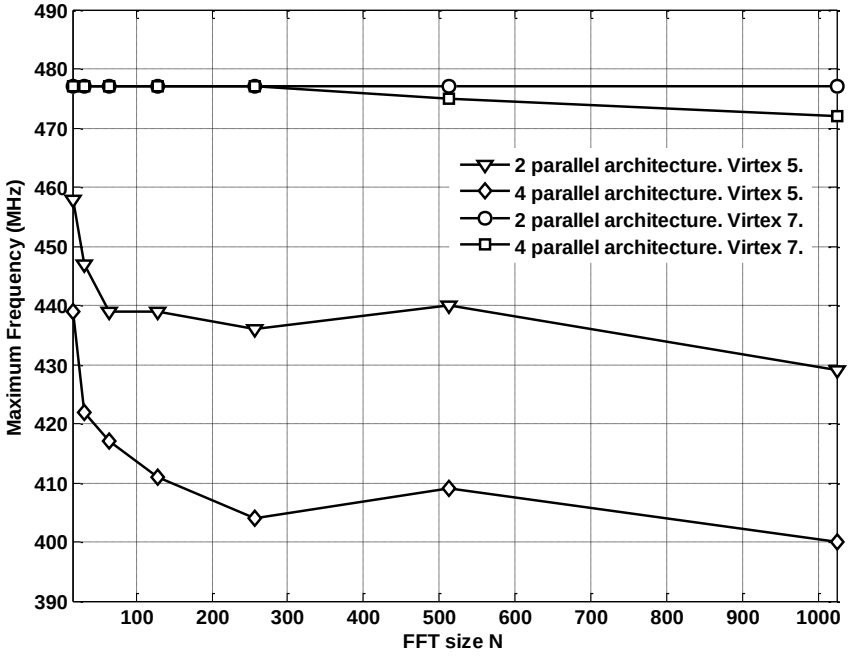

Figure 4.22 : Fréquence de fonctionnement en fonction de la taille *N* : comparaison entre l'implémentation sur le *Virtex-5* et le *Virtex-7* des architectures proposées.

A partir des courbes de la figure 4.22 on peut conclure que :

- l'implémentation sur le *FPGA Virtex-7* permet d'achever une fréquence nettement supérieure à celle du circuit *FPGA Virtex-5*,

- l'implémentation sur le *FPGA Virtex*-7 ne présente pas de grande différence de fréquence entre l'architecture à deux séquences et celle à quatre séquences, contrairement à l'implémentation sur le *FPGA Virtex-5*,
- la taille *N* influe légèrement sur la fréquence du processeur implémenté sur le *Virtex-7,* à l'opposé du *FPGA Virtex-5* où la fréquence diminue avec l'augmentation de la taille *N*.

Les comparaisons de la puissance consommée par les processeurs *FFT* à deux et à quatre séquences réalisés sur les circuits *Virtex-5* et Virtex-7 en fonction de la fréquence sont illustrées dans les deux figures 4.23 et 4.24 respectivement. Pour mettre en évidence l'influence de la taille *N* sur la puissance nous avons tracé les graphes pour les deux valeurs extrêmes de la taille (*N*=16 et *N*=1024).

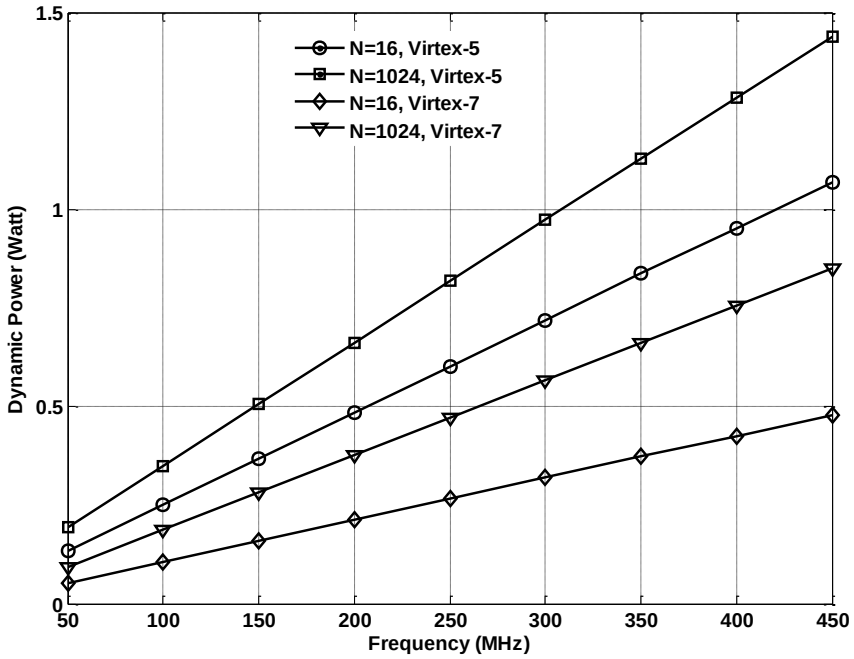

Figure 4.23 : Puissance dynamique consommée par le processeur *FFT* à deux séquences en fonction de la fréquence : comparaison entre le *Virtex-5* et le *Virtex-7*.

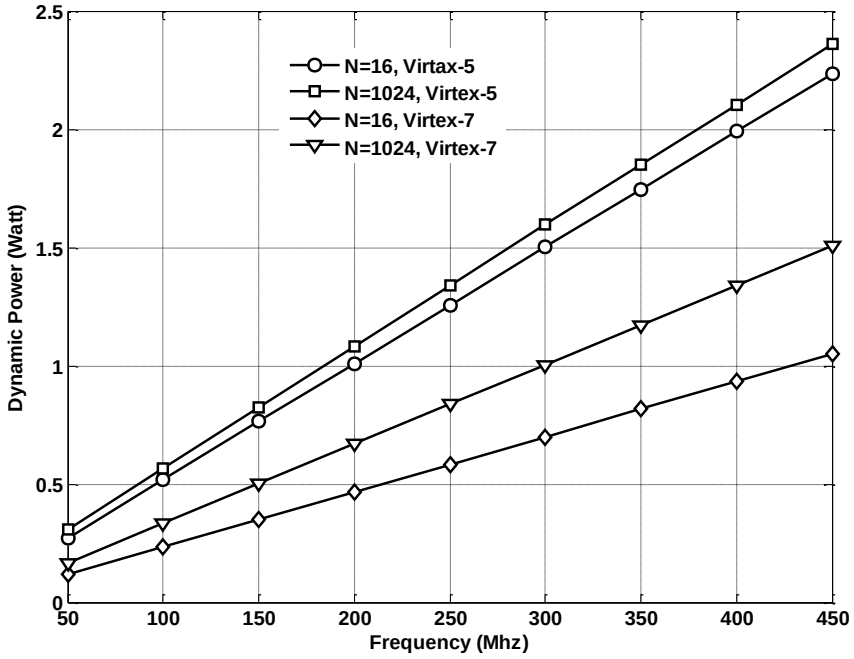

Figure 4.24 Puissance dynamique consommée par le processeur FFT à quatre séquences en fonction de la fréquence : comparaison entre le *Virtex-5* et le *Virtex-7*.

Les courbes des deux figures 4.23 et 4.24 montrent clairement que :

- la puissance consommée augmente linéairement avec la fréquence de fonctionnement ces résultats expérimentaux sont en concordance avec la théorie.
- l'implémentation sur le *FPGA* Virtex-7 permet de réduire considérablement la puissance consommée par rapport au Virtex-5. Cette réduction devient de plus en plus importante pour les fréquences élevées.
- la taille *N* du processeur *FFT* influe sur la puissance consommée, sauf le cas des circuits à quatre séquences implémentées sur le Virtex-5.

La comparaison des résultats d'implémentation des architectures à haute fréquence pour quatre séquences avec d'autres architectures récentes est reportée dans le tableau 4.14.

Tableau 4.14 : Comparaison des résultats d'implémentation des architectures proposées avec d'autres travaux.

N	Référence	Type FPGA	Slice LUTs	Slices	DSPs	Freq. (MHz)
16	**Proposé [111]**	**XC5VSX240T-2FF1738**	**1355**	**619**	**12**	**439**
		XC7VSX485T-2FFG1761	**1575**	**606**	**12**	**477**
		XC6CSX475T-1 FF1156	**1307**	**506**	**12**	**400**
		XC5VLX20T	**1136**	**534**	**12**	**384**
	[82]	XC5VSX240T-2FF1738	-	386	12	458
	[83]	XC6CSX475T-1 FF1156	-	567	12	335
	[81]	XC5VLX20T	2348	1046	-	370
	[112]	XC5VSX240T-2FF1738	2688		16	322
32	**Proposé [111]**	**XC5VSX240T-2FF1738**	**2301 - 2327**	**931 - 991**	**12 - 16**	**422 - 424**
		XC7VS485T-2FFG1661	**2527 - 2534**	**850 - 882**	**12 - 16**	**471 - 477**
		XC5VLX20T	**2109 - 2135**	**829 - 835**	**12 - 16**	**382 - 385**
	[81]	XC5VLX20T	3088	1290	-	370
64	**Proposé [111]**	**XC5VSX240T-2FF1738**	**2059 - 3294**	**895 - 1284**	**16 - 24**	**417 - 421**
		XC7VS485T-2FFG1661	**2258 - 3466**	**814 - 1141**	**16 - 24**	**475 - 477**
		XC6CSX475T-1 FF1156	**2831 - 4028**	**699 - 1006**	**16 - 24**	**394 - 400**
		XC5VLX20T	**2007 - 3026**	**811 - 1151**	**16 - 24**	**375 - 394**
	[82]	XC5VSX240T-2FF1738	-	695	24	384
	[83]	XC6CSX475T-1 FF1156	-	782	24	335
	[81]	XC5VLX20T	3832	1560	-	370
	[112]	XC5VSX240T-2FF1738	4440	-	32	303
256	**Proposé [111]**	**XC5VSX240T-2FF1738**	**2914 - 4152**	**1095 -**	**28 - 36**	**404 - 417**
		XC7VS485T-2FFG1661	**3117 - 4334**	**1033 -**	**28 - 36**	**472 - 477**
		XC6CSX475T-1 FF1156	**2837-4046**	**928 - 1305**	**28 - 36**	**400**
	[82]	XC5VSX240T-2FF1738	-	1024	36	389
	[83]	XC6CSX475T-1 FF1156	-	924	36	240
	[112]	XC5VSX240T-2FF1738	6932	-	48	297
1024	**Proposé [111]**	**XC5VSX240T-2FF1738**	**4932 - 5483**	**1659 -**	**40 - 48**	**400 - 411**
		XC7VS485T-2FFG1661	**5063 - 6307**	**1569 -**	**40 - 48**	**472 - 474**
		XC6CSX475T-1 FF1156	**4828-6011**	**1434 -**		**400**
	[82]	XC5VSX240T-2FF1738	-	1425	48	270
	[83]	XC6CSX475T-1 FF1156	-	1351	48	227
	[112]	XC5VSX240T-2FF1738	11216	-	64	298

Il est important de noter que Ayinala *et al* ont proposé dans [81] une architecture à deux chemins de traitement, mais qui permet de traiter uniquement une seule séquence puisqu'elle n'utilise pas d'unité de réordonnancement des entrées. Wang *et al* ont proposé dans [112] une architecture à un seul chemin de traitement pour le calcul de la *DFT* d'une seule séquence d'entrée. Donc, pour fournir une comparaison valide, les ressources utilisées par ces architectures ont été multiplié par 2 pour [81] et par 4 pour [112] afin de refléter le même nombre de chemins de traitement que les architectures proposées.

Le tableau 4.13 montre clairement que les architectures proposées permettent d'achever la plus haute fréquence de fonctionnement, en comparaison avec toutes les autres architectures. L'amélioration de la fréquence de fonctionnement devient de plus en plus importante l'orque la taille de la *DFT*, *N* augmente. En outre, et contrairement aux autres méthodes, la méthode proposée permet l'implémentation du processeur *FFT* pour toutes les valeurs de la taille *N* qui sont une puissance de deux ($N = 2^{n_s}$).

En terme du nombre de Slice de type *DSP48E*, la technique proposée nécessite moins de *DSP* que la technique proposée par Wang *et al* [112] et le même nombre de *DSP ; ou* même moins pour certaines combinaisons ; que l'architecture proposée par Gariddo et al dans [82], [83]. Il est difficile de fournir une comparaison valide avec le travail de Garrido *et al* [82], [83] en nombre de Slices et de *LUT*s, car les architectures ne contiennent pas d'unité de réordonnancement des entrées. Cependant, les architectures proposées utilisent moins de *LUT*s et de Slices que l'architecture présentée dans [81]. De plus, nos architectures utilisent moins de *LUT*s que [112] alors que le nombre de Slices utilisés dans [112] n'est pas fourni dans le rapport d'utilisation des ressources.

4.6. Conclusion

Dans ce chapitre, nous avons présenté les architectures proposées pour l'implémentation efficace sur *FPGA* d'un processeur *FFT* à haut débit, et qui est capable de traiter deux ou quatre séquences indépendantes. Les architectures

proposées sont caractérisées par un processus de contrôle très simple, qui ne nécessite qu'un simple compteur binaire qui fournit les signaux de contrôle pour les différents étages des architectures. Les architectures sont basées sur l'utilisation d'un mélange de modules Radix-2^2 et Radix-2^3, ce qui permet une grande flexibilité vis-à-vis de la taille *N.* Les circuits sont implémentés sur un circuit Virtex-5 et sur un autre circuit de type Virtex-7, pour les déférentes valeurs de la taille *N,* et pour plusieurs combinaisons de Radix-2^2 et Radix-2^3 pour chaque valeur de la taille. Les architectures proposées montrent un rapport d'amélioration de débit jusqu'à 20.82 pour l'architecture à deux séquences, et jusqu'à 19.12 pour l'architecture à quatre séquences, par rapport aux architectures à basse fréquence. En outre, l'implémentation des architectures sur le circuit de type Virtex-7 permet une amélioration en termes du débit, de la puissance dynamique consommée, et du nombre de Slices utilisés en comparaison avec le circuit Virtex-5.

CONCLUSION GENERALE ET PERSPECTIVES

Conclusion générale

Le nombre d'utilisateurs des réseaux de télécommunications mobiles ne cesse d'augmenter d'une façon exponentielle ces dernières années. En outre, et avec la diversification des services et de type de données transmises qu'offrent les réseaux actuels (*3G* et *4G*) ou ceux qui sont envisagés pour les générations futures (*5G*), les exigences en débit et en qualité de service sont devenu de plus en plus rigoureux. De ce fait, il est indispensable d'utiliser des techniques avancées de communication numérique, permettant une utilisation efficace du spectre radio disponible en assurant un maximum de débit, un maximum de nombre d'utilisateurs, et une bonne robustesse. Parmi ces techniques, le trio *MIMO, OFDM* et *CDMA* est presque omniprésent dans les agendas des chercheurs et des industriels du domaine de télécommunications.

L'utilisation des techniques avancées de communication numérique comme le *MIMO, l'OFDM* et la *CDMA* nécessite des circuits et des techniques d'implémentation et de mise en œuvre très puissantes, efficaces et flexibles. Le développement dans le domaine des circuits intégrés numériques, et spécialement dans le domaine des circuits reconfigurables de type *FPGA,* offre une alternative pour la conception de systèmes qui assure un meilleur compromis entre performances et puissance consommée d'un côté et flexibilité de l'autre côté.

Le présent travail de thèse, a porté sur la contribution à l'implémentation sur *FPGA* d'un système de télécommunications qui combine le codage spatio-temporel ; qui fait partie des techniques *MIMO ;* avec la modulation *OFDM* ou avec la technique *MC-CDMA*. Dans le cadre de cet objectif, nous avons ciblé deux blocs importants dans le système qui sont le bloc *FFT* et le bloc de décodage spatio-temporel d'Alamouti. Ces

deux blocs sont connus par leurs complexités arithmétiques élevées dont leur implémentation sur un circuit numérique présente un vrai défi pour les chercheurs.

Nous avons commencé le chapitre 1 par la présentation des notions de base sur les communications numériques, en débutant par les blocs de base d'un système de communication simple. Puis, les différents modèles de canaux radio et des phénomènes relatifs à la propagation dans un canal radio ont été présentés. Nous avons terminé ce chapitre par l'introduction des trois techniques : *MIMO, OFDM* et *CDMA*. Dans ce chapitre, nous avons vu que les canaux radio sont généralement modélisés par un modèle de Rayleigh en cas d'absence d'un trajet direct dominant entre l'émetteur et le récepteur, et par le modèle de Rice en cas de présence d'un trajet direct. Nous avons vu aussi que le canal peut être considéré comme plat ou sélectif en fréquence, selon la bande du signal à transmettre et la bande de cohérence du canal. Du point de vue temporel, le canal est à évanouissement lent ou rapide selon son temps de cohérence et la durée de symbole. Dans la dernière partie du chapitre 1, nous avons vu que les techniques *MIMO* permettent d'augmenter le débit par le multiplexage spatial ou la robustesse par le codage spatio-temporel. La modulation *OFDM,* quant à elle, convertit le canal sélectif en fréquence en plusieurs sous-canaux plats, par la répartition de flux de données à haut débit sur plusieurs sous-porteuses orthogonales. La technique *CDMA* offre la possibilité à plusieurs utilisateurs de partager la même bande de fréquence par l'utilisation de la diversité de code.

Dans le deuxième chapitre, nous avons vu que le codage spatio-temporel *STC* est une technique qui exploite la dimension spatiale ajoutée par les systèmes *MIMO* afin d'augmenter la robustesse du système. Nous avons constaté que les codes spatio-temporels en blocs orthogonaux *OSTBC* forment une classe importante, car elle assure un décodage *ML* à complexité réduite du fait que les colonnes de leurs matrices génératrices des codes sont orthogonales. Un cas particulier des *OSTBC,* qui est le codage d'Alamouti, a fait l'objet d'une étude détaillé dans ce chapitre. Dans la dernière partie du chapitre 2, nous avons présenté les deux architectures que nous avons proposées pour l'implémentation efficace et à haut débit du décodeur d'Alamouti pour le cas 2×2 et 2×1. Nous avons vu que la première architecture

proposée réduit le nombre de multiplieurs complexes de huit à quatre et le nombre d'additionneurs complexes de six à trois par rapport à l'architecture conventionnelle. La deuxième architecture proposée n'utilise que deux additionneurs et deux multiplieurs complexes. La première architecture assure un débit de 584 Ms/S et la deuxième architecture assure un débit de 300 Ms/S.

Le troisième chapitre a été dédié à l'étude des algorithmes *FFT* du côté représentation mathématique, complexité de calcul, ainsi que du côté architecture d'implémentation. Les variantes de l'algorithme *FFT* sont multiples (Radix-2, Radix-4, Radix-8, Radix-2^2…). Chacune est caractérisée par sa complexité arithmétique qui est exprimée en nombre d'opérations d'additions et de multiplications complexes nécessaires. Nous avons vu dans ce chapitre que les architectures en pipeline sont les plus adaptées aux systèmes à haut débit et fonctionnant en temps réel en comparaison avec les architectures à base de mémoire. Nous avons expliqué dans ce chapitre que les processeurs *FFT* dédié aux systèmes *MIMO-OFDM* ou *MIMO-MC-CDMA* sont basés sur un concept d'implémentation qui utilise plusieurs chemins de traitement d'un côté, et qui permet de partager au maximum les ressources de l'autre côté dans le but de minimiser les ressources, sinon le nombre de ressources nécessaires augmente intensivement avec l'augmentation du nombre d'antennes.

Dans le dernier chapitre de ce document, nous avons détaillé les architectures que nous avons proposées pour l'implémentation d'un processeur *FFT* dans le contexte du *MIMO-OFDM* et *MIMO-MC-CDMA*. Nous avons montré d'abord l'intérêt des deux algorithmes Radix-2^2 et Radix-2^3, du fait qu'ils réduisent le nombre de multiplieurs par rapport à l'algorithme Radix-2 tout en gardant la structure Butterfly simple du Radix-2. Les circuits que nous avons proposés sont basés sur l'utilisation du Radix mixé 2^2-2^3 et de l'approche *MDC*. Les processeurs proposés sont capables de traiter deux ou quatre séquences indépendantes en utilisant deux ou quatre chemins de traitement, pour n'importe quelle taille du *FFT* qui est une puissance de deux. On outre, nous avons démontré que les architectures proposées présentent une grande flexibilité, du fait qu'un processeur *FFT* pour la même taille peut être réalisé par plusieurs combinaisons de module Radix-2^2 et Radix-2^3. Les architectures sont configurées pour pouvoir fonctionner à des fréquences très élevées dans le but d'assurer un débit

de traitement très élevé. Les résultats d'implémentation des architectures montrent que la fréquence maximale de fonctionnement peut atteindre jusqu'à 477MHz pour le Virtex-7 et jusqu'à 458 MHz pour le Virtex-5. Cette amélioration importante de la fréquence et du débit n'est pas pénalisée par une augmentation excessive des ressources utilisées et de la puissance consommée.

Perspectives :

Le travail mené dans cette thèse peut conduire à plusieurs perspectives, entre autres, on peut citer :

- L'extension des architectures *FFT* proposées à plus de quatre chemins de traitement. Afin de permettre de traiter un nombre important de séquences indépendantes dans un contexte de systèmes *MIMO* à grande échelle « Large Scale MIMO » *LS-MIMO* envisagés pour les réseaux de la cinquième génération *5G*. Pour ce faire, il faut chercher une stratégie efficace pour l'ordonnancement des entrées. En plus, il faut trouver une architecture de calcul à chemins multiples, mais avec une complexité matérielle réduite. Dans ce cadre, on peut même envisager une architecture à nombre d'entrées variables selon le nombre d'antennes en émission et en réception pour l'utiliser dans les systèmes adaptatifs.
- Les deux architectures que nous avons proposées dans le chapitre 2 pour le décodage d'Alamouti peuvent être modifiées, pour les rendre capables de décoder d'autres codes espace-temps et pour plusieurs configurations de nombre d'antennes d'émission et de nombre d'antennes de réception. Une autre piste de recherche consiste à modifier les architectures afin de les utiliser aussi pour la détection en multiplexage spatial. De telles architectures sont souhaitables pour les systèmes qui basculent entre le codage espace-temps et le multiplexage spatial.
- Dans ce présent travail, nous avons proposé des architectures pour deux blocs importants d'un système *MIMO-OFDM* ou *MIMO-MC-CDMA*. Nous pouvons envisager à implémenter la totalité du système en émission et en réception en proposant des architectures pour les différents blocs restants du système.

Ensuite, il est possible de faire une étude sur les performances et sur la consommation des ressources et de la puissance de la totalité du système.

- Récemment, la firme Xilinx a développé un circuit *Zynq* [113] qui regroupe, en une seule puce, un processeur *ARM* dual cortex *A9* et un *FPGA* de la série 7. Un tel circuit fournit à la fois les performances et la consommation en puissance des A*SIC*s et la flexibilité des processeurs *ARM*s et des circuits *FPGA*. Il est très bénéfique d'envisager une implémentation de parties ou de la totalité d'un système à base de *MIMO-OFDM/MIMO-MC-CDMA* sur ce type de circuit. Dans ce cadre, on peut examiner plusieurs configurations de partage de tâches entre le système de traitement « processing system » *PS* et la logique programmable « Programmable Logic » *PL,* pour assurer un meilleur compromis entre performance, ressources et puissance.

REFERENCES

1. CISCO, “Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021”, white paper, (February 2014), 1-35.
2. Oestges, C. and Clerckx, B., “MIMO wireless communications: From real-world propagation to space-time code design”, Academic Press, (2010), 480 p.
3. Tsoulos, G.,*MIMO* “system technology for wireless communications”, CRC press,Florida, USA, (2006), 400 p.
4. Bessai, H., “MIMO signals and systems”, Springer Science & Business Media, New York, USA (2006). 215 p.
5. Kuhn, V., “Wireless communications over MIMO channels: applications to CDMA and multiple antenna systems”,. John Wiley & Sons, England, (2006), 388 p.
6. Chang, R. W. “Orthogonal frequency division multiplexing”, US. patent 3-488-445, (January. 1970), 445-488.
7. Le Floch, B., Alard M. and Berrou, C., “Coded Orthogonal Frequency Division Multiplex”, Proceeding of the IEEE, V. 83, n° 6, (Juin 1995), 982-996.
8. Fazel, K. and Kaiser, S., “Multi-Carrier and Spread Spectrum Systems From OFDM and MC-CDMA to LTE and WiMAX”, John Wiley and Sons Inc; 2nd edition,, United Kingdom, (2008), 374 p.
9. Dixon, R. C., “Spread spectrum systems”, John Wiley & Sons Inc; 2nd edition, New York, USA, (1984), 440 p.
10. Glisic, S.and Vucetic, B., “Spread spectrum CDMA systems for wireless communications”, Artech House, Norwood, (1997), 383 p.
11. Jafarkhani, H., “Space-time coding theory and practice”, Cambridge University Press, Cambridge, UK., (2005), 320 p.
12. Vucetic, B., and Yuan, J. “Space-time coding”, John Wiley & Sons, U.K. (2003), 332 p.

13. Bölcskei, H., “MIMO-OFDM Wireless Systems : Basics, Perspectives, and Challenges”, IEEE Wireless Communincation. Magazine., V. 13, n° 4, (August 2006), 31–37.
14. Chouly, A., Brajal, A. and Jourdan, S., “Orthogonal multicarrier techniques applied to direct sequence spread spectrum CDMA systems”, Proceeding. IEEE Global Telecommunications Conference (GLOBECOM ’93), Houston, USA, (November./December 1993), 1723–1728.
15. Zhang, W. Xia X.G. and Ben Letaief, K., “Space-Time/Frequency Coding for MIMO-OFDM in Next Generation Broadband Wireless Systems”, IEEE Wireless Communication Magazine, V. 14, n° 3, (June 2007), 32–43.
16. Hu, X. and Chew, Y.H., “Performance of space-time block coded MC-CDMA system over frequency selective fading channel using semi-blind channel estimation technique”, WCNC 2003 - IEEE Wireless Communications and Networking Conference, V. 4, n° 1, (March 2003), 414–419.
17. Kafle, P.L. and Sesay, A.B., “Iterative semi-blind multiuser detection for turbo-coded MC-CDMA system with space-time transmit diversity”, MILCOM 2002, IEEE Military Communications Conference, V. 21, n° 1, (October 2002), 1038–1042.
18. Kilts, S., “Advanced FPGA Design Architecture, Implementation, and Optimization”, John Wiley & Sons, Inc, New Jersey, (2007), 355 p.
19. Cooly, J.W. and Tukey, J.W., “An algorithm for the machine calculation of complex Fourier series”, Mathematics of computation, V. 19, n° 90, (April 1965), 297–301.
20. Proakis J. G. “Digital Communications”, Fourth Edition, McGraw Hill, New York, (2000), 936 p.
21. Tse, D., Viswanath, P., “Fundamentals of Wireless Communication”, Cambridge University Press, Cambridge, UK, (2005), 647 p.
22. Kattenbach, R. “Considerations about the validity of WSSUS for indoor radio channels”,COST 259 TD, 070, 3rd Management Committee Meeting. (1997).
23. Guillet, J. and Ghaïs El Zein. “Caractérisation spatio-temporelle des canaux linéaires de propagation sans fil: contexte MIMO”, 4e journée d'étude

Propagation électromagnétique dans l'atmosphère, du décamétrique à l'Angström. (2002).

24. Winters, J., “On the capacity of radio communication systems with diversity in a Rayleigh fading environment”, IEEE journal on selected areas in communications, V. 5, n° 5, (June 1987), 871-878.
25. Foschini, G. J. and Gans, M. J “On limits of wireless communications in a fading environment when using multiple antennas”, Wireless Personal Communications, V. 6, n°. 3, (1998), 311–335.
26. Gershman, A., B. and Sidiropoulos, N., D., “Space-Time Processing for MIMO Communications”, John Wiley & Sons Ltd, West Sussex, England, (2005), 390 p.
27. Telatar, E., “Capacity of multi-antenna gaussian channels,” Transactions on Emerging Telecommunications Technologies V. 10, n° 6, (December 1999), 585-595.
28. Lau, V.K.N., Y-K., Ricky “Channel-Adaptive Technologies and Cross-Layer Designs for Wireless Systems with Multiple Antennas Theory and Applications”, John Wiley & Sons New Jersey USA (2006), 544 p.
29. Sundberg, C.E.W. and Seshadri, N., “Digital cellular systems for North America”,. IEEE Globecom, (December 1990), 533–537.
30. Balaban, N. and Salz, J., “Dual diversity combining and equalization in digital cellular mobile radio”, IEEE Transaction on Vehicular Technology, V. 40, n° 2, (May 1994), 342–354.
31. Winters, J., Salz, J. and Gitlin, R. D., “The impact of antenna diversity on the capacity of wireless communication systems”, IEEE Transaction. on Communications, V. 42, n° 234(February/March./April 1994), 1740–1751.
32. Zheng, L. and Tse, D. N. C., “Diversity and multiplexing: a fundamental trade-off in multiple-antenna channels”, IEEE Transaction. on Information Theory, V. 49, n° 5: (May 2003), 1073–1096.
33. Shannon, C. E., “A mathematical theory of communication”, the Bell System Technical Journal, V. 27, n° 3, (July 1948), 379-423.

34. Yang, S. and Hanzo, L., “Fifty years of MIMO detection: The road to large-scale MIMOs”, IEEE Communications Surveys & Tutorials, V.17, n°4 (September 2015), 1941-1988.
35. Viterbo, E. and Boutros, J., “A universal lattice code decoder for fading channels”, IEEE Transactions on Information theory, 1999, V. 45, n° 5, (July 1999), 1639-1642.
36. Viterbo, E. and Biglieri, E., “A universal decoding algorithm for lattice codes”, 14° Colloque sur le traitement du signal et des images, France, (1993).
37. Doelz, M. L. Heald, E. T. and Martin, D. L., “Binary data transmission techniques for linear systems,” Proceedings of the IRE, V. 45, n° 5, (May 1957), 656–661.
38. Mosier, R. R. and Clabaugh, R. G. “Kineplex, a bandwidth-efficient binary transmission system,” American Institue of Electrical Engineers, V. 76, (January 1958), 723–728.
39. Chang, R. W., “Synthesis of band-limited orthogonal signals for multi-channel data transmission”, Bell System Technical Journal, V. 45, n° 10 , (December 1966), 1775,1796.
40. Weinstein, S. and Ebert, P., “Data transmission by frequency-division multiplexing using the discrete fourier transform”, IEEE Transactions on Communications, , V. 19, n°. 5, (October 1971), 628–634.
41. Peled, A. and Ruiz, A., “Frequency domain data transmission using reduced computational complexity algorithms”, in International Conference on Acoustics, Speech, and Signal Processing, vol. 5, (May 1980), 964-967.
42. Standard, “Radio broadcast systems; digital audio broadcasting (DAB) to mobile, portable and fixed receivers”, Tech. Rep. Pre., (1994), 300-401.
43. DaSilva, V. and Sousa, E. S., “Performance of orthogonal CDMA codes for quasi-synchronous communication systems”, in Proceeding. IEEE International Conference on Universal Personal Communications (ICUPC ’93), Ottawa, Canada, V. 2, (October 1993), 995–999.
44. Fazel, K., “Performance of CDMA/OFDM for mobile communication system”, Proceeding. IEEE International Conference on Universal Personal Communications (ICUPC ’93), Ottawa, Canada,V. 2, (October 1993), 975–979.

45. S. Kondo, S. and Milstein, L., “On the use of multicarrier direct sequence spread spectrum systems”, Military Communications Conference, (MILCOM'93), V. 1, (October 1993), 52-56.
46. Hara, S., Lee, T. H. and Prasad, R., “BER comparison of DS-CDMA and MC-CDMA for frequency selective fadings channels”, Proceeding of Signal Processing in Telecommunications, Springer, London, (1996), 3-14.
47. Kaiser, S. “OFDM-CDMA versus DS-CDMA: performance evaluation for fading channels”, IEEE International Conference on Communications, (Juin 1995), 1722-1726.
48. Abeta, S., Atarashi, H.and Sawahashi, M. “Forward link capacity of coherent DS-CDMA and MC-CDMA broadband packet wireless access in a multi-cell environment”,. IEEE Vehicular Technology Conference, V. 5 (Mars 2000), 2213-2218.
49. Geramita, A. V. and Seberry, J. “Orthogonal Designs, Quadratic Forms and Hadamard Matrices”,. Lecture Notes in Pure and Applied Mathematics, n° 04; QA166. 4, G4, (1979).
50. Radon, J. “Lineare scharen orthogonaler matrizen”, Abhandlungen aus dem Mathematischen Seminar der Hamburgishen Universitat, I: (1922), 1–14.
51. Alamouti, S. M., “A simple transmit diversity technique for wireless communications”, IEEE Journal Select. Areas Communication, V. 16, n° 8, (October 1998), 1451–1458.
52. Tarokh, V., Seshadri N. and Calderbank, A. R., “Space-time codes for high data rate wireless communication: Performance criterion and code construction ”, IEEE transactions on information theory, (March 1998), V. 44, n° 2, 744-765.
53. Lee, K.F and Williams, D.B., “A space-Time Coded Transmitter Diversity Technique for Frequency Selective Fading Channels”,. In Proceedings of IEEE Sensor Array and Multichannel Signal Processing Workshop, (March 2000), 149–152.
54. Bölcskei, H. and Paulraj, A.J., “Space-Frequency Coded Broadband OFDM Systems”, IEEE Wireless Communications and Networking Conference, V. 1, (September 2000), 1–6,.

55. Zeng, Y., Yin, Q., Zhang, Y., Deng, K., Luo, M. and Ren, A. “Direct decoding of uplink space-time block coded multicarrier code division multiple access systems”, Communications, 2004 IEEE International Conference on, V. 4, (June 2004), 2372–2376.
56. Yang, Z., Lu, B. and Wang, X., “Bayesian Monte Carlo multiuser receiver for space-Time coded multicarrier CDMA systems”, IEEE Journal on Selected Areas in Communications, V. 19, n° 8, (August 2001),1625–1637.
57. Xilinx, “System Generator for DSP, Getting Started Guide”, UG639, v 14.3, (October, 2012), 79 p.
58. Xilinx, “System Generator for DSP, Reference Guide”, UG638, v 14.3, (October 2012), 626 p.
59. Xilinx, “7 Series FPGAs Data Sheet: Overview ”, DS180, v2.5, (August , 2017), 18 p.
60. Xilinx, “7 Series FPGAs Configurable Logic Block User Guide”, UG474 v1.8, (September 2016), 74 p.
61. Xilinx, “ISE In-Depth Tutorial”, UG695, v14.1, (April 2012), 150 p.
62. Xilinx, “Xilinx Power Tools Tutorial”, UG733, v14.4 (December, 2012), 26 p.
63. Rao, K.R.l., Kim, D.N.l and Hwang, J.J., “Fast Fourier Transform: Algorithms and Applications”, Springer, London, UK. (2010), 443 p.
64. Duhamel, P., “Implementing of split-radix FFT algorithms for complex, real, and real symmetric data”, IEEE Transactions on Acoustics, Speech, and Signal Processing, V. 34, n° 2, (April 1986), 285–295.
65. Duhamel, P. and Vetterli, M., “Fast Fourier transforms: A tutorial review and a state of the art”, Signal Processing, Elsevier, V. 19, n° 4, (April 1990), 259–299
66. Uwe Meyer-Baese “Digital Signal Processing with Field Programmable Gate Arrays”, Springer, Berlin, Germany, (2007). 744 p.
67. Good, I.J., “The Relationship between Two Fast Fourier Transforms”, IEEE Transactions on Computers V. **20-C**, n° 3, (March 1971), 310–317.
68. Thomas, L., “Using a Computer to Solve Problems in Physics,” Applications of Digital Computers, V. 458, (1963), 44-45.

69. Winograd, S., “On Computing the Discrete Fourier Transform”, *Mathematics of Computation, V.* **32**, n° 141, (1978), 175–199.
70. Sahnine, C., “Architecture de circuit intégré reconfigurable, très haut débit et basse consommation pour le traitement numérique de l'OFDM avancé”, these de doctorat, institut Polytechnique de Grenoble, France 2009.
71. Wold, E.H. and Despain, A.M., “Pipeline and parallelpipeline FFT processors for VLSI implementation”, IEEE Trans. Comput., C-33(5):414-426, May 1984.
72. Despain, A.M., “Fourier transform computer using CORDIC iterations”, IEEE Transactions on Computers”, V. 100, n° 10, (October 1974), 993-1001.
73. He, S., Torkelson, M., “A new approach to pipeline FFT processor”, Proceeding International Parallel Processing Symposium (IPPS '96), (April 1996), 766-770.
74. He, S., Torkelson, M., “Designing pipeline FFT processor for OFDM (de)modulation”, Proceeding. International Signals, Systems, and Electronics Symposium (ISSSE 98), (September 1998), 257- 262.
75. Hsiang S., Hu, H.Y., Chen and Jou, S.J. “Novel FFT Processor with Parallel-In-Parallel-Out in Normal Order”, VLSI Design Automation and Test Symposium, Page(s): 150-153, 2009.
76. Rabiner, L.R. and Gold, B. “Theory and Application of Digital Signal Processing”, Prentice-Hall, Inc., (1975), 762 p.
77. Cheng, C., and Parhi, K. K., “High-throughput VLSI architecture for FFT computation”, IEEE Transactions on Circuits and Systems II: Express Briefs,V. 54, n° 10, (October 2007), 863-867.
78. Meletis, C., Bougas, P., Economakos, G., Kalivas, P., and Pekmestzi, K., “High-speed pipeline implementation of radix-2 DIF algorithm”, In *Proceedings of World Academy of Science, Engineering and Technology,* V. 2, (2005), 1-4.
79. Swartzlander, E.E., Young, W.K.W. and Joseph, S.J.A., “radix 4 delay commutator for fast Fourier transform processor implementation”, *IEEE* Journal *Solid-State Circuits,* V. 19, n° 5, (October 1984), 702-709.
80. Lee, S., Jung, Y., Kim, l., “Low Complexity Pipeline FFT Processor For MIMO-OFDM Systems”, Institute of Electronics, Information and Communication Engineers Electron Express, V. 4, n° 23, (2007), 750-754.

81. Ayinala, M., Brown, M., Parhi, K.K., “Pipelined parallel FFT architectures via folding transformation”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, V. 20, n° 6, (June 2012), 1068-1081.
82. Garrido, M., Rajal, J.G., Sánchez, M.A., *and* Gustafsson, O., “Pipelined Radix-2^k Feedforward FFT Architectures”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, V. 21,n° 1, (January 2013), 23-32.
83. Garrido, M., Acevedo, M., Ehliar, A. and Gustafsson, O., “Challenging the limits of FFT performance on FPGAs,” in Proc. IEEE International Symposium on Integrated Circuits (ISIC), (December 2014), 172-175.
84. Han, W., Arslan, T., Erdogan, A.T. and Hasan, M., “Low power commutator for pipelined FFT processors”, IEEE International Symposium on Circuits and Systems (ISCAS 2005), (May 2005), 5274-5277.
85. Saponara, S., Serani, L. and Fanucci, L. “Low-power FFT/IFFT VLSI macro cell for scalable broadband VDSL modem”, The 3rd IEEE International Workshop on Systemon-Chip for Real-Time Applications,(June-July 2003), 161-166.
86. Bi, G. and Jones, E.V., “A pipelined FFT processor for word-sequential data”, IEEE Transaction, Acoustic., Speech, Signal Processing, V. 37, n° 12, (December 1989), 1982-1985.
87. Bidet, E., Castelain, D., Joanblanq, C. and Stenn, P., “A fast single-chip implementation of 8192 complex point FFT”, IEEE Journal, Solid-State Circuits, V. 30, n° 3, (March 1995), 300-305.
88. Lin, Y.W., Liu, H.Y. and Lee, C.Y., “A 1-GS/s FFT/IFFT processor for UWB applications,” IEEE Journal, Solid-State Circuits, V. 40, n° 8, (August. 2005), 1726–1735.
89. Shin, M. and Lee, H. “A high-speed four-parallel radix- FFT/IFFT processor for UWB applications,” *IEEE International. Symposium Circuits and Systems (ISCAS 2008).*, (May 2008), 960–963.
90. Tang, S.N., Tsai, J.W. and Chang, T.Y., “A 2.4-GS/s FFT processor for OFDM-Based WPAN applications,” IEEE Transaction, Circuits and Systems. II,Exp. Briefs, V. 6, n°. 57, (June), 2010 451–455.

91. Yang, S.W., and Lee, J.Y., “Constant twiddle factor multiplier sharing in multipath delay feedback parallel pipelined FFT processors”, Electronics Letters,V. 50, n° 15, (2014), 1050-1052.
92. Jo, B. and Sunwoo, M., “New continuous-fow mixed-radix (CFMR) FFT processor using novel in-place strategy”, IEEE Transactions on Circuits and Systems, V. 52, n°. 5, (May 2005),.911-919.
93. Wey, C.L., Tang, W.C. and Lin, S.Y., “Efficient VLSI implementation of memory-based FFT processors for DVB-T applications”, IEEE Computer Society Annual Symposium on VLSI, (March 2007), 98-106.
94. Wey, C.L., Lin, S.Y., Tang, W.C. and Shiue; M.T., “High-speed, low cost parallel memory-based FFT processors for OFDM applications”, 14th IEEE International Conference on Electronics, Circuits and Systems ICECS 2007, (December. 2007), 783-787.
95. Chen, K.H. and Li, Y.S. “A multi-radix FFT processor using pipeline in memory-based architecture (PIMA) for DVB-T/H systems”, 15th International Conference on Mixed Design of Integrated Circuits and Systems, MIXDES, (June 2008), 549-553.
96. Jang, S., Yang, G., Lee, S., and Jung, Y., “Area-efficient FFT processor for MIMO-OFDM based SDR systems”, IEICE Electronics Express, V. 10, n° 15, (July 2013), 1-6.
97. Glittas, A.X., Sellathurai, M. and Lakshminarayanan, G. “A Normal I/O Order Radix-2 FFT Architecture to Process Twin Data Streams for MIMO”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, V. 24, n° 6, (June 2016), 2402-2406.
98. Tsai, P.Y., Chen, C.W., Huang, M.Y., “Automatic IP generation of FFT/IFFT processors with word-length optimization for MIMO-OFDM systems”, EURASIP Journal on Advances in Signal Processing, V. 2011, (November 2011), 1-15.
99. Lin, Y.W., Lee, C.Y., “Design of an FFT/IFFT processor for MIMO OFDM systems”, IEEE Transactions on Circuits and Systems I: Regular Papers, V. 54, n° 4, (April 2007), 807-815.

100. Liu, H. and Lee, H., “A high performance four-parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems ”, IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2008, (November/December 2008), 834-837.
101. Tang, S.N., Liao, C.H. and Chang, T.Y., “An area- and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems”, IEEE Journal of Solid-State Circuits, V. 47, n° 6, (June 2012), 1419-1435.
102. Yang, K.J., Tsai, S.H. and Chuang, G.C.H., “MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, V. 21, n° 4, (April 2013), 720-731.
103. Wang, C., Yan, Y. and Fu, X. “A High-Throughput Low-Complexity Radix-2^4-2^2-2^3 FFT/IFFT Processor With Parallel and Normal Input/Output Order for IEEE 802.11 ad Systems”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, V. 23, n° 11, (November 2015), 2728-2732.
104. Fu, B., and Ampadu.P., “An area efficient FFT/IFFT processor for MIMO-OFDM WLAN 802.11 n”, Journal of Signal Processing Systems V. 56, n° 1, (September 2009), 59-68.
105. Locharla, G.R., Kallur, S.K., Mahapatra, K.K., and Ari, S. “Implementation of MIMO data reordering and scheduling methodologies for eight-parallel variable length multi-path delay commutator FFT/IFFT”, IET Computers & Digital Techniques V. 10, n° 5, (2016), 215-225.
106. Shi, L.Z., and Guo J.M., “Design of an 8-Channel FFT Processor for IEEE 802.11 ac MIMO-OFDM WLAN System”, Circuits, Systems, and Signal Processing, V. 35, n° 10 (2016), 3759-3769.
107. Yoshizawa, S., Orikasa, A. and Miyanaga, Y. “An area and power efficient pipeline FFT processor for 8× 8 MIMO-OFDM systems”, IEEE International Symposium on Circuits and Systems (ISCAS), (May 2011), 2705-2708.
108. Xilinx, “Vivado Design Suite User Guide, Model-Based DSP Design using System Generator”, UG 897,v2014.1, (April 2014), 188p.
109. Xilinx, “VC707 Evaluation Board for the Virtex-7 FPGA User Guide”, UG885, v1.7.1, (August, 2016), 116p.

110. Dali, M., Guessoum, A., Gibson, R. M., Amira, A., & Ramzan, N. (2017). “Efficient FPGA implementation of high-throughput mixed radix multipath delay commutator FFT processor for MIMO-OFDM”, Advances in Electrical and Computer Engineering, V.17, n° 1, (February 2017), 27-39.
111. Xilinx, “Virtex-5 FPGA User Guide”, UG190, v5.4, (March, 2012), 385 p.
112. Wang, Z., Liu, X., He, B. and Yu, F., “A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT, ” IEEE Transaction on Very Large Scale Integration (VLSI) Systems, V. 23, n° 5, (May 2015), 793-977.
113. Crockett, L., H., Elliot, R., A., Enderwitz, M., A. and Stewart, R. W., “The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC”, Strathclyde Academic Media, UK, (July 2014), 484 p.

LISTE DES SYMBOLES ET ABREVIATIONS

ASIC : Application Specific Integrated Circuits
AWGN : Additive White Gaussian Noise
BER : Bit Error Rate.
BPE : Butterfly Processor Element
BRAM : Block Random Access Memory.
CDMA : Code Division Multiple Access CDMA.
CFA : Common Factor Algorithms.
CSI : Channel State Information.
DAB : Digital Audio Broadcasting.
DFT : Discrete Fourier Transform.
DFD : Decision-Feedback Detector.
DIT : Decimation In Time.
DIF : Decimation In Frequency.
DPTR : Dynamic Power to Throughput Ratio.
DS-CDMA: Direct Sequence Code Division Multiple Access.
DSP : Digital Signal Processor.
DVB-T : : Digital Video Broadcasting Terrestrial.
EGC : Equal Gain Combining.
FDM : Frequency Devision Multiplex.
FFT : Fast Fourier Transform.
FH-CDMA: Frequency Hopping Code Division Multiple Access .
FPGA : Field Programmable Gate Arrays.
GPP : General Purpose Processor.
HDL : Hardware Description Language.
IC : Interference Cancellation.
ICI : Inter Chanel Interference.

IID : Independent Identically Distributed.
IOTA : Isotropic Orthogonal Transform Algorithm
ISI : Inter Symbol Interference.
LAN : Local Area Network.
JTAG : Joint Test Action Group.
LMDS : Local Multi-point Distribution Systems.
LOS : Line Of Site.
LR : Lattice Reduction.
LUT : Look Up Table.
MAI : Multiple Access Interference.
MAP : Maximum a posteriori.
MED : Minimum Euclidean Distance.
MF : Matched Filter.
MIMO : Multiple Inputs Multiple Outputs.
MIC : Multistage interference Cancellation.
ML : Maximum Likelihood.
MLSE : Maximum Likelihood Sequential Estimation.
MMDS : Microwave Multi-point Distribution Systems.
MMSE : Minimum Mean Square Error.
MRC : Maximum Ratio Combining.
Ms/S : Mega samples per Second.
OFDM : Orthogonal Frequency Division Multiplexing.
OSIC : Ordered Successive interference Cancellation.
OSTBC : Orthogonal Space Time Bloc Code.
OSVF : Orthogonal variable spreading factor.
PAPR : Peak to Average Power Ratio.
PDF : Probability Density Function.
PFA : Prime Factor Algorithms.
PIC : Parallel interference Cancellation.
R2SDF : Radix-2- Single path Delay Feedback.
RAM : Random Access Memory.

ROM: : Read Only Memory.

RTL : Register Transfer Level

RUR : Resources Utilization Ratio.

SD : Sphere Decoding .

SDF : Single path Delay Feedback.

SFG : Signal Flow Graph.

SIC : Successive interference Cancellation.

SISO : Single Input Single output.

SNR : Signal to Noise Ratio.

SQNR : Signal to Quantification Noise Ratio.

SRL : Shift Register Look-up table.

SS-MC : Spread spectrum-Multiple Carrier.

STBC :Space Time Bloc Coding.

STC : Space Time Coding.

STTC : Space Time Trellis Coding.

TR : Through-put Ratio.

US : Uncorrelated Scattering.

WSS : Wide Sense Stationary.

XSG : Xilinx System Generator.

ZF : Zero Forcing.

λ : la longueur d'onde.

$\|\boldsymbol{x} - \boldsymbol{y}\|^2$: Distance Euclidienne entre les deux vecteurs complexes **x** et **y.**

$\langle x \rangle_y$: x modulo y.

arg : fonction argument.

exp(x) : exponentiel de x.

GCD(a,b) : plus grand commun diviseur de a et b.

G_d : Gain de diversité.

G_m : Gain de multiplexage.

H : Matrice d'un canal MIMO.

I : Matrice identité.

Im(x) : partie imaginaire du nombre complexe x.

min(x, y) : le minimum de x et y.

N : La taille de la DFT.

n **:** vecteur de bruit AWGN.

N **:** Matrice de bruit AWGN.

n_s : le nombre d'étages pour un FFT de taille N.

N_t : nombre d'antenne d'émission pour un système MIMO.

N_r : nombre d'antenne de réception pour un système MIMO.

N_p : nombre de sous porteuse utilisées pour la modulation OFDM.

R: **:** matrice de signaux reçus.

r **:** vecteur de signaux reçus.

Re(x) : partie réelle du nombre complexe x.

s **:** vecteur de signaux émis.

T **:** matrice de transformation pour les décodeurs linéaires.

wl : longueur d'un mot binaire en virgule fixe.

x(n) : séquence d'entrée de l'algorithme FFT.

X(k) : Séquence de sortie de l'algorithme FFT.