UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
CHOKRI NEILI

CONCEPTION ET IMPLÉMENTATION SUR FPGA DE STRUCTURES MULTIPLEXES DE LA FFT EN PIPELINE

SEPTEMBRE 2017

Résumé

La transformée discrète de Fourier (DFT - *Discrete Fourier Transform*) représente une méthode de base en traitement numérique des signaux pour l'analyse des signaux numériques. Son exécution en technologie d'intégration à très grande échelle représente un problème non trivial quand il s'agit de respecter les contraintes de l'application en termes de consommation de puissance, coût d'implémentation et vitesse de calcul.

Ce travail présente l'évaluation d'une nouvelle formulation de la FFT, en vue d'une implémentation plus efficace que les propositions conventionnelles. Cette nouvelle formulation comprend de nouvelles conceptions pour des processeurs élémentaires (BPE - *Butterfly Processing Element*) selon les radix-r utilisés.

L'originalité du projet provient du multiplexage des multiplieurs complexes des coefficients de Fourier (TWF – *Twiddle Factor*) dans la conception des BPE radix-*r*. Les résultats présentés dans ce mémoire concernent le radix-4 et radix-8 de l'implémentation de la nouvelle conception de BPE radix-*r* sur FPGA. L'évaluation des performances des implémentations se base sur plusieurs critères, à savoir, débit de la BPE, latence, fréquence d'horloge et un critère performances globale.

Ces résultats se comparent favorablement aux plus récents articles de références sur le sujet utilisant la structure pipeline et parallèle de la FFT. Les résultats présentent un gain de 130% en fréquence par rapport à la référence pour N=256, ce qui permet d'améliorer la performance MS/s/Slice de 116%. Le gain en fréquence ou en performance devient plus

grand quand on utilise la technologie de Virtex-7, en effet, pour N=256, on enregistre un gain de 173% en termes de fréquence et de 209% en termes de performance.

Finalement, ce projet permet de valider l'efficacité de la nouvelle formulation de la FFT, en termes de performances de puissances de calcul et de rapidité pour une l'utilisation dans des applications différentes.

Remerciements

Je tiens à adresser mes sincères remerciements à Monsieur Daniel Massicotte, professeur régulier au département génie électrique et directeur du Laboratoire des Signaux et Systèmes Intégrés(LSSI), qui, en tant que Directeur de recherche, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire. Ses conseils pertinents avec écoute, amabilité et patience ont permis à mon travail d'aboutir et de voir le jour.

À ma très chère mère affable, honorable, aimable : Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

À mon cher père : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien-être.

À mes deux petits frères : Les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous.

À ma sœur : Ton amour m'a aidé à grandir et devenir quelqu'un de bien, car tu es mon exemple. Ton amour m'a aidé à surmonter bien d'épreuves difficiles, car tu es mon exemple. Ton amour m'a aidé à avancer en toutes circonstances, car tu es mon exemple.

À mon épouse : Avec toi, j'ai appris à sourire, à partager, j'ai appris à donner et à recevoir, j'ai appris que la vie vaut la peine d'être vécue, si l'amour est présent. Tes sacrifices, ton soutien moral et matériel, ta gentillesse sans égal, ton profond attachement m'ont permis de réussir mes études. Que Dieu réunisse nos chemins pour un long commun serein.

À tous mes collègues et amis. À tous ceux qui avec de chaleur et d'amour se sont dévoués pour ma réussite et mon bonheur, je dédie ce travail.

Table des matières

RÉSUMÉ.....

REMERCIEMENT.....

TABLE DES MATIÈRES.....

LISTE DES FIGURES.....

LISTE DES TABLEAUX.....

LISTE DES ABRÉVIATIONS .....

1. CHAPITRE 1 1

1.1 Problématique 3

1.2 Objectif 6

1.3 Méthodologie 6

1.4 Organisation du mémoire 9

2. CHAPITRE II 11

2.1 Transformée de Fourier Discrète 12

2.2 La Transformée Rapide de Fourier 14

2.2.1 Algorithmes conventionnels 15

2.2.2 Approche diviser-pour-régner 16

2.2.3 Radix-2 19

2.2.4 Radix-4 22

2.3 Architecture parallèle pour le calcul de la FFT 23

2.4 Architectures pipelines pour le calcul de la FFT 25

2.4.1 Single-path Delay Feedback SDF 26

2.4.2 Single-path Delay Commutator SDC 28

2.4.3 Multi-path Delay Commutator MDC 29

2.5 Effet de Quantification 33

2.5.1 Erreur de quantification 34

2.6 Différentes approches d'implémentation de la FFT 34

2.6.1 Processeur FFT pipeliné localement 34

2.6.2 Processeur FFT pour les applications WPAN à base de OFDM 37

2.6.3 Radix-2^4 FFT à haute performance pour les applications MIMO-OFDM 40

2.6.4 Processeur FFT pour les systèmes UWB 41

2.6.5 Radix-2^k feedforward FFT pipelinées 43

3. CHAPITRE III 47

3.1 Éléments nécessaires pour implémenter la FFT 48

3.1.1 Étapes de conception 48

3.1.2 Structure pipeline de la FFT 50

3.1.3 Conception et architecture pipeline radix-2^2 et radix-2^3 FFT 52

3.2 Synthèse des résultats d'implémentation 69

3.2.1 Critères d'évaluation 69

3.2.2 Résultats d'implémentation 72

3.2.3 Synthèse et analyse des Résultats 78

4. CHAPITRE IV 84

5. Références 87

Liste des figures

Figure 1 Structure DIT Butterfly *19*

Figure 2 Structure DIF Butterfly *20*

Figure 3 Diagramme de la FFT à base de BPE radix-2 de type DIT *20*

Figure 4 Diagramme de la FFT à base de BPE radix-2 de type DIF *21*

Figure 5 BPE du radix-4 DIT *22*

Figure 6 Architecture parallèle du radix-2 (N=8) *23*

Figure 7 Architecture parallèle du radix-4 (N=16) *24*

Figure 8 Architecture parallèle du SRFFT-4/2 (N=32) *24*

Figure 9 Architecture pipeline de base [YJ03] *25*

Figure 10 Architecture R2SDF (N = 16 points) *27*

Figure 11 Architecture R4SDF (N= 16 points) *27*

Figure 12 Architecture R4SDC (N = 256 points) *29*

Figure 13 Architecture R4MDC (N = 64 points) *30*

Figure 14 Architecture MC-MRMDC (k = 4, N = 64 points) *31*

Figure 15 Quelques méthodes de représentations en virgule fixe *33*

Figure 16 Architecture radix-16 LPPL [GA89] *36*

Figure 17 Architecture R2SD2F pipeliné pour DFT à 16 points [GA89] *36*

Figure 18 Diagramme Block d'un 2048-FFT processeur [SN10] *37*

Figure 19 Structure du module 1[SN10] *38*

Figure 20 Diagramme Block du module 4[SN10] *39*

Figure 21 4-parallel data-path d'un processeur radix-2^4 MDF FFT/IFFT à 128/64-points [HL08] 41

Figure 22 Processeur SDF FFT radix-2^4 parallèle proposé [SI08] 42

Figure 23 Structures des unités BPE (Types 1, 2, et 3) [SI08] 43

Figure 24 Architecture radix-2^2 feedforward à 4 entrées parallèles pour le calcul de DIF FFT de 16-points [MG13] 44

Figure 25 Architecture radix-2^2 feedforward proposée pour le calcul de la DIF FFT de 64-points [MG13] 45

Figure 26 Diagramme de la méthode de co-simulation Matlab-Modelsim 51

Figure 27 Architecture pipeline radix-2^2 MDC de type DIF à deux étages N=16 [MJ14] 53

Figure 28 (MDC-R22) BPE Conventionnelle de radix-2^2 54

Figure 29 (a) BPE_MUX proposée (b) diagramme de blocs de circuit du réseau d'additionneur complexe radix-2 [MJ14] 55

Figure 30 Timing block diagramme de la figure 31[MJ14] 55

Figure 31 Timing block diagramme de la figure 32[MJ14] 56

Figure 32 BPE Conventionnelle de radix-23 [MJ14] 56

Figure 33 BPE Multiplexée (BPE_MUX) proposée de radix-23 [MJ14] 57

Figure 34 Architecture d'une BPE de radix-4 [HS12] 57

Figure 35 Chemin Critique d'une BPE_MUX de radix-2^3 implémentée 58

Figure 36 Représentation des signaux de contrôle dans la structure Partial MuxMDC-$R2^2$ 60

Figure 37 Comparaison entre les Chemins critiques des BPE implémentés a) BPE_MUX (figure 29a) et b) BPE conventionnelle (figure 28) 60

Figure 38 Data flow diagramme de la structure Partial MuxMDC-R2^2 implémentée 61

Figure 39 Représentation de la latence du Partial MuxMDC-R2^2 implémentée 62

Figure 40 Structure interne de multiplieur de nombres complexes non pipeliné optimisé . 63

Figure 41 Structure interne de multiplieur de nombres complexes pipeliné optimisé 64

Figure 42 Structure implémentée du Multiplieur de nombres complexes utilisant 4 multiplieurs réels 65

Figure 43 Structure interne de commutateur 66

Figure 44 Représentation des délais liés au module du Commutateur pour une FFT de 16 points 66

Figure 45 Les transformations de commutateur 67

Figure 46 Architecture pipeline radix-4 MDC de type DIF sur 3 étages N=64 points 68

Figure 47 Erreur entre virgule fixe et flottante d'une FFT de taille 16 utilisant BPE_MUX 70

Figure 48 Erreur entre virgule fixe et flottante d'une FFT de taille 16 utilisant BPE conventionnelle 70

Figure 49 Schéma technologique complet du R4MDC à 16 points 81

Figure 50 Schéma technologique complet du R4MDC à 64 points 82

Figure 51 Schéma technologique complet du R4MDC à 256 points 82

Figure 51 Schéma technologique complet du R4MDC à 256 points 82

Liste des tableaux

Tableau 1 Comparaison du nombre d'opérations dans un calcul direct de la DFT et dans la FFT [AV03] *16*

Tableau 2 Nombre d'opérations d'une FFT de 4096 points pour différents radix *22*

Tableau 3 Comparaison en termes d'exigences matérielles entre différentes architectures FFT pipelines [SL07] *32*

Tableau 4 Outils de conception *49*

Tableau 5 Comparaison des nombres de ressources utilisées par BPE MUX et convontionnelle *59*

Tableau 6 Comparaison de l'utilisation des ressources nécessaires pour le calcul de la FFT *59*

Tableau 7 Distribution des registres dans les Commutateurs pour la FFT de radix-2^2 .... *67*

Tableau 8 Comparaison d'implémentation sur Virtex-5 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Timing Performance) *74*

Tableau 9 Comparaison d'implémentation sur Virtex-5 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Area Reduction) *74*

Tableau 10 Comparaison d'implémentation sur Virtex-5 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Balanced) *75*

Tableau 11 Comparaison d'implémentation sur Virtex-7 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Timing Performance) *75*

Tableau 12 Comparaison d'implémentation sur Virtex-7 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Area Reduction) 76

Tableau 13 Comparaison d'implémentation sur Virtex-7 des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Balanced) 76

Tableau 14 Comparaison des performances de l'implémentation de la FFT utilisant des Multiplieurs de nombres complexes conventionnels pipelinés et non pipelinés 77

Tableau 15 Comparaison des performances de l'implémentation de la FFT utilisant des Multiplieurs de nombres complexes optimisés pipelinés et non pipelinés 77

Tableau 16 Comparaison de résultats d'implémentation sur Virtex-5 entre BPE MUX et conventionnelle (Avec Multiplieurs) de radix-2^3 77

Tableau 17 Résultats d'implémentation des FFT radix2^2 selon la référemce [MG13] 78

Tableau 18 Synthèse de l'évaluation des performances du FFT radix-2^2 78

Liste des abréviations

OFDM	*Orthogonal Frequency Division Multiplexing*
FFT	*Fast Fourier Transform*
FPGA	*Field Programmable Gate Array*
DFT	*Discrete Fourier Transform*
DIT	*Decimation In Time*
DIF	*Decimation In Frequency*
LSSI	*Laboratoire des Signaux et Systèmes Intégrés*
VHDL	*Very High Description Language*
SFG	*Signal Flow Graph*
BPE	*Butterfly Processing Element*
R2MDC	*Radix2 Multipath Delay Commutator*
SQNR	*Signal Quantization Noise Ratio*
DSP	*Digital Signal Processin*
SRFFT	*Split Radix Fast Fourier Transform*
VLSI	*Processing Element*
SDF	*Single-path Delay Feedback*
SDC	*Single-path Delay Commutator*
MDC	*Multi-path Delay Commutator*
MC-MRMDC	*Multi Channel - Mixed Radix Multi-path Delay Commutator*

LPPL	*Locally Pipelined Processor*
HRSMA	*High-Radix and Shared Memory Architecture*
R2SD2F	*Radix-2 Single Deep Delay Feedback*
WPAN	*Wireless Personal Area Network*
OFDM	*Orthogonal Frequency Division Multiplexing*

CHAPITRE I

INTRODUCTION

La transformée de Fourier, qui est une méthode reconnue pour l'analyse fréquentielle ou spectrale, est un outil essentiel pour l'implémentation et la conception de nombreuses techniques numériques de traitement des signaux et des données [JR07]. Cette technologie se trouve dans plusieurs applications directes comme l'analyse harmonique des vibrations et des signaux musicaux, dans le codage à débit réduit de la musique et de la parole, la reconnaissance vocale, l'amélioration de la qualité des images, leur compression, les transmissions numériques, les nouveaux systèmes de radiodiffusion et de télédiffusion, dans

les applications biomédicales (scanner, imagerie par résonance magnétique nucléaire), en astronomie (synthèse d'image par interférométrie), en modélisation de propagation d'ondes, en analyse spectrale pour l'étude de structures moléculaires ainsi qu'en cristallographie[JR07]. La transformé de Fourier peut être aussi un élément important dans les méthodes envisagées en informatique quantique pour la factorisation de nombres.

La transformation de Fourier discrète (TFD) est l'équivalent discret de la transformation de Fourier continue qui est utilisée pour le traitement du signal analogique.

C'est un outil qui permet d'analyser un signal temporel quelconque de manière discrète (on va donc échantillonner le signal) dans l'espace fréquentiel pour étudier la puissance de chaque fréquence qui compose le signal [JR07].

La transformation de Fourier rapide (en anglais : FFT ou *Fast Fourier Transform*) est un algorithme plus avancé pour le calcul de la transformation de Fourier discrète (TFD). Elle se base sur des éléments de traitement en papillon (BPE – *Butterfly Processing Element*) à radix-r ou en français radical-r ou base-r, où r usuel sont 2, 4 et 8. Sa complexité s'exprime en $O(N\log_r N)$ nombre d'opérations avec N et r représentent respectivement, le nombre de points traité par la transformée et le radix-r de la transformation rapide, alors que la complexité de l'algorithme TFD s'exprime de l'ordre de $O(N^2)$. Ce qui résulte, pour

N=1024, le temps de calcul de l'algorithme rapide peut être ~100 fois plus court que le calcul utilisant la formule de définition de la TFD.

Cette méthode a été publiée pour la première fois en 1965 par James Cooley et John Tukey, ce qui a lancé définitivement l'utilisation massive de la FFT en traitement du signal en

générale et dans les technologies plus spécifique comme la télécommunication. Cet algorithme reste toujours utilisé dans les applications de traitement numérique du signal en particulier dans les analyseurs de spectre. Il est également à la base des algorithmes de multiplication rapide (Schönhage et Strassen, 1971), et des techniques de compression numérique ayant mené au format d'image JPEG (1991).

L'objectif essentiel de la présente étude porte sur l'implémentation au niveau matériel de l'exécution de la BPE de la FFT, l'élément principal pour augmenter les performances en termes de réduction des ressources et d'augmentation de la vitesse des calculs pour répondre aux exigences des nouvelles technologies. La cible matérielle visée dans le cadre de ce mémoire sont les FPGA (*Field Programmable Gate Array*). Les FPGA sont de plus en plus utilisés dans différentes technologies pour sa reconfigurabilité et mise en marché rapide des nouvelles technologies. Leur flexibilité, rapidité et capacité à recevoir des architectures parallèles les rendent intéressantes pour l'implémentation de la FFT.

1.1 Problématique

Pour être capable de calculer une TFD on doit utiliser N^2 nombre d'opérations (multiplications et additions), avec N représente la taille de la transformée. La taille de N ainsi que le nombre d'opérations continuent toujours à augmenter avec l'évolution des diverses applications des systèmes de communications [MJ09]. Plusieurs algorithmes ont été élaborés dans les années récentes dont le but est pour diminuer le nombre d'opérations de la TFD. Ces algorithmes sont connus sous le nom d'algorithmes FFT.

En effet, l'algorithme de la FFT permet la possibilité d'exécuter rapidement la DFT sur processeur électronique. Le premier algorithme FFT développé par Cooley-Tukey est connu sous le nom de radix-2 DIT (*Décimation In Time*), il requière $N\log_2 N$ nombre d'opérations au lieu de N^2 pour le calcul direct de la DFT [JWC65], où 'r' représente le radix. On remarque bien la diminution concrète du nombre d'opérations à effectuer. Une nouvelle approche de calcul appelé 'papillon' connu sous le nom 'Butterfly Computation' a été introduite par cet algorithme.

Avec l'augmentation constante de la vitesse d'exécution des différents nouveaux systèmes ainsi que le débit (*Throughput*) des données exigées à l'entrée et le cout croissant de l'accès à la mémoire, concevoir des BPE pour la FFT est devenu l'objectif principal de plusieurs recherches dans le domaine du traitement de signal. À travers ces nouvelles architectures, on essaye toujours d'utiliser la moindre des ressources matérielles tout en réduisant la consommation de puissance. En effet, on peut trouver une grande variance d'algorithmes qui ont été développés et étudiés pour améliorer davantage la manière d'implémentation des FFT, parmi ces algorithmes on trouve le radix-2 introduit par (*Cooley-Tukey*), le Split radix Algorithm [TYS10], Winograd Fourier Transform Algorithm (WFTA) [SW75] et bien d'autres.

Grâce à l'architecture simplifiée de leur BPE, les radix-2 et radix-4 conventionnels sont considérés comme les algorithmes les plus utilisés en industrie, ce qui leur donne un avantage par rapport aux algorithmes FFT de radix plus élevé. Plus on utilise un radix élevé, plus le nombre des opérations arithmétiques pour calculer une FFT de N points diminue, $O(N\log_r N)$. En outre, avec un radix élevé, l'architecture de la BPE pour exécuter la FFT devient plus complexe et exigeante en termes de calcul arithmétique, de connexions, et de

nombres du délai du chemin critique (*Critical Path Delay*). En contrepartie, plus le radix est élevé, moins d'itérations sont nécessaire dans l'utilisation de la BPE (moins d'étage d'exécution de BPE), ce qui, dans l'ensemble du calcul de la FFT, réduit le nombre d'opérations arithmétiques, mais introduit une grande complexité en termes d'accès mémoires, de temps de calcul par BPE et de complexité du flot des données. Ceci ne veut pas dire que l'utilisation d'un radix élevé est toujours un désavantage puisque son utilisation nous permet de réduire le nombre de multiplications et de nombre d'étages pour exécuter la FFT, il y a dons un compromis à chercher dans la dimension du radix, des ressources et du temps de calcul pour l'exécution d'une FFT de taille N. En sachant, qu'une multiplication prend plus d'espace en silicium ou de ressources matérielles qu'une simple addition et que la diminution du nombre d'étages diminue la charge d'interconnexion et d'accès mémoires, l'utilisation de BPE de radix plus grand est bénéfique au niveau de l'implémentation matérielle [MJ09]. À condition que la complexité de la BPE permette de satisfaire les compromis dans la complexité d'implémentation et la vitesse.

Une nouvelle formulation de la FFT [MJ14], développée au sein du laboratoire LSSI (Laboratoire des Signaux et Systèmes Intégrés), permet l'utilisation d'une architecture innovante de BPE avec radix plus grand tout en gagnant un avantage en termes de vitesse de calcule et des ressources utilisées (ex : nombre de Slices et DSP48 des FPGA de Xilinx).

1.2 Objectif

L'objectif principal de ce projet est l'évaluation de la nouvelle formulation de la FFT en vue d'une implémentation plus efficace que les propositions conventionnelles et des plus récentes de la littérature. Cette évaluation est faite en termes de temps de calcul et de ressources matérielles d'implémentation en technologie FPGA. Les sous-objectifs permettant de rencontrer l'objectif principal sont :

- Étude des algorithmes FFT et leur implémentation.
- Étude de la nouvelle formulation de la FFT proposée dans les travaux du LSSI et publié dans [MJ14]. Plus spécifiquement les structures BPE radix-2, radix-2^2 et radix-2^3.
- Étude des architectures FFT pipelines et parallèles, les avantages et les techniques d'implémentation.
- Modélisation, simulation et implémentation des nouvelles structures BPE (*Butterfly Processing Element*) dans des FFT de différentes grandeurs sur Matlab® et en VHDL pour évaluer leurs performances sur FPGA (Virtex-5, Virtex-7 ...) et comparer ensuite avec les derniers résultats dans la littérature.

1.3 Méthodologie

Afin de réaliser nos objectifs, notre méthode de travail consiste en premier lieu d'étudier les algorithmes conventionnels pour calculer la FFT comme : radix-2, radix-2^2 et radix-2^3, ils

serviront tout au long du projet pour comprendre les algorithmes de références puisqu'ils sont les plus utilisés en industrie.

Cette étude permettra de comprendre les différentes techniques d'implémentation des architectures MDC (*Multi path Delay Commutator*) et des BPE (*Butterfly Processing Element*) à partir des équations mathématiques qui régissent les algorithmes FFT conventionnels.

Pour bien comprendre leurs bons fonctionnements et leurs équations mathématiques qui les gèrent, on utilise Matlab® pour programmer les FFT conventionnels. L'outil Matlab® sera notre plateforme pour effectuer les vérifications et les évaluations sur tous les algorithmes qui seront programmés et implémentés durant le projet. Après étude de la nouvelle architecture de la BPE proposée par les travaux de 'Marwan Jaber' et 'Daniel Massicotte', nommé BPE_MUX, on la programme cette fois sur notre plateforme Matlab®. On testera sur cette dernière, son bon fonctionnement en l'introduisant dans une architecture FFT complète.

Une fois cette étape complétée, on programmera la nouvelle BPE en VHDL. L'outil choisi pour faire la simulation VHDL est Modelsim (PE Edition 10.2a) de Mentor Graphics®. Cette nouvelle architecture programmée en VHDL sera testée sur Modelsim® avec des programmes 'testbench' conçus pour valider la structure VHDL du BPE afin de l'utiliser pour constituer le Processeur FFT. On effectue une co-simulation Matlab® / Modelsim®. La plateforme Matlab® est utilisée pour générer les données normées entre 1 et -1 (vecteurs d'entrée et Twiddle *Factors*) en virgules fixes (16 bits), ces données sont après utilisées pour les injecter dans la BPE programmée sur Matlab® et sur Modelsim®. L'erreur entre les sorties des BPE en virgules fixes sera comparée pour valider le bon fonctionnement du BPE.

On s'attend à avoir des résultats de ces simulations dans le chapitre 3, où l'erreur doit être égale ou inférieure au bit le moins significatif. On se servira alors de cette nouvelle BPE pour construire les coprocesseurs FFT pour différentes grandeurs (16 points, 64 points et 256 points).

Les BPE des FFT programmées dans le projet ont une architecture pipeline. C'est grâce à leur bon compromis vitesse/surface qu'elles étaient choisies. L'architecture programmée est essentiellement R4MDC (*Radix4 Multipath Delay Commutator*) qui sera détaillée dans les chapitres 2 et 3.

Enfin, la synthèse FPGA de la nouvelle BPE (BPE_MUX) ainsi que son implémentation FPGA sont effectuées sur Xilinx ISE 14.6. Les résultats obtenus démontreront l'efficacité réelle de la nouvelle formulation de la BPE en termes de vitesse de calcul et des ressources matérielles d'implémentation en technologie FPGA.

L'évaluation des algorithmes FFT pour différentes tailles est donc faite suivant quatre étapes : (i) étude théorique des algorithmes FFT, (ii) étude de la nouvelle formulation de la BPE proposée dans les travaux du LSSI [MJ14], (iii) étude des architectures FFT pipelines et parallèles et leur modélisation et enfin (iv) l'implémentation des BPE constituants les BPE de la FFT sur Matlab® et en VHDL pour évaluer leurs performances sur FPGA.

Cette étude des différentes architectures FFT pipelines et parallèles, était l'occasion parfaite pour approfondir nos connaissances dans ce domaine de recherche. Pour l'évaluation des différentes architectures FFT on les a directement modélisées en VHDL sans passer par un logiciel qui génère du code VHDL. La nouvelle architecture du BPE développée a été programmée avec la façon la plus optimale pour réduire sa complexité et obtenir des résultats acceptables en comparant avec la référence.

1.4 Organisation du mémoire

Ce mémoire est organisé comme suit :

Dans le Chapitre 2, on expliquera le principe de la TFD qui sera la base pour introduire la Transformée de Fourier Rapide (*Fast Fourier Transform : FFT*). La mathématique qui gère l'algorithme conventionnel et l'approche deviser pour régner de cette architecture sera donc expliquée.

On parlera après des FFT de radix-2 et de radix-4 puisqu'ils seront la base de travail de notre projet. Une brève définition des architectures parallèles sera après faite suivit par une étude des architectures pipelinées et l'effet de la quantification (virgule fixe) sur la conception de la FFT. Dans la dernière section de ce chapitre, on introduira une présentation des dernières recherches qui étaient faites dans ce domaine avec leurs différentes approches pour améliorer le fonctionnement de la FFT en tenant compte de différents paramètres.

Le chapitre 3 discutera l'implémentation sur FPGA de la BPE_MUX proposée par Jaber&Massicotte (Simulation VHDL, Synthèse FPGA) avec toutes les étapes nécessaires pour atteindre le meilleur résultat de performance. On commencera le chapitre par introduire et expliquer la théorie derrière les différents éléments nécessaires pour l'implémentation de la FFT. On discutera après les résultats de synthèse selon les critères mis en place. Enfin, le chapitre 4 présentera la conclusion générale du projet de recherche.

CHAPITRE II

IMPLÉMENTATION DE LA FFT – ÉTAT DE L'ART

Avec l'évolution exponentielle des systèmes techniques avancés, le besoin d'une vitesse d'exécution et une complexité matérielle croissante ne cessent d'augmenter. D'où vient la nécessité de concevoir des architectures VLSI de coprocesseurs FFT qui répondent aux exigences de vitesse (haut débit binaire) et qui sont moins complexes en termes de ressources matérielles pour des raisons de coûts et de réduction de la consommation énergétique [YJ03]. Dans ce chapitre on étudiera les deux architectures les plus communes en termes d'efficacité d'implémentation de la FFT ; l'architecture parallèle et l'architecture pipeline ainsi que les

différentes approches pour les implémenter. Leurs caractéristiques, avantages et inconvénients seront aussi étudiés.

2.1 Transformée de Fourier Discrète

La Transformée de Fourier Discrète (TFD), (*DFT : Discrete Fourier Transform*) est un élément essentiel dans l'analyse, la conception et la mise en œuvre des algorithmes et systèmes du traitement du signal à temps discret. En physique numérique, on dispose presque toujours de signaux issus d'une acquisition électronique ou de résultats de mesures discrètes. Ces signaux et résultats ne sont pas infinis, et généralement sont non périodiques. La TFD a été conçue pour traiter ce genre de données. Elle réalise une décomposition d'un signal tronqué (fini, de durée T) et échantillonné (discret) en une série de Fourier, en le "périodisant", avec une période égale à T. La TFD nous permet donc de calculer de façon approchée les coefficients de Fourier des différentes harmoniques d'un signal quelconque et ainsi d'obtenir son spectre en fréquence.

Son application est donc utilisée dans plusieurs domaines technologiques tels que les télécommunications, le biomédical, le traitement séismique, etc. [YW07], [MJ08]

La DFT d'un signal discret x(n) peut-être directement calculée par l'équation (2.1),

$$X(n)^n = \sum_{n=0}^{N-1} x(n) W_N^{nk},\ \mathrm{k} \in [0, \ldots\ldots, \mathrm{N}-1] \qquad (2.1)$$

Avec $x(n)$ et $X(n)$, respectivement, représentent la séquence d'entrée et la séquence de sortie en nombre complexe et N est la longueur de la transformée. À partir de l'équation (2.1), on

constate que la complexité de calcul de la TFD s'exprime en $O(N^2)$, il augmente avec le carré de la longueur de la transformée et donc devient cher pour des N larges. Plusieurs additions et soustractions, en plus de quelques autres opérations, sont nécessaires donc pour réaliser une seule multiplication complexe. C'est pour ces raisons que la TFD n'est pas utilisée pour le traitement du signal temps réel. Puisque le processeur numérique est généralement limité par la taille de sa mémoire. La taille de la TFD N limite les calculs exécutés le processeur [WY07].

L'équation de définition de la DFT fournit une relation entre deux ensembles de N nombres complexes, en posant :

$$W_N^{nk} = e^{-j(2\pi/N)nk} \text{ avec } j^2 = -1 \tag{2.2}$$

W_N^p est appelé Facteur de Fourier (*TWF - Twiddle Factors*), plusieurs propriétés caractéristiques de la représentation de ce coefficient sont utilisées [AV03], par exemple :

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

$$W_N^{2kn} = W_{N/2}^{kn} \tag{2.3}$$

$$W_N^{k(N-n)} = (W_N^{kn})^*$$

Où x* représente le complexe conjugué de la donnée complexe x.

Une forme matricielle montrée par l'équation (2.4) de la DFT peut être donc déduite.

$$x(k) = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ \vdots & & \ddots & \vdots \\ W_N^0 & W_N^{(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ \ldots \\ x(N-1) \end{bmatrix} \tag{2.4}$$

Elle peut être représentée sous la forme matricielle suivante :

$$X(k) = [B_N][x_n] \tag{2.5}$$

Avec $[\mathbf{B}_N]$ la matrice des coefficients et $[\mathbf{x}_n]$ la matrice d'entrée de la TFD. Donc la TFD est une décomposition d'un signal échantillonné, composé de sinusoïdes.

En exploitant les propriétés de symétrie et de périodicité de la TFD, plusieurs méthodes efficaces ont été développées pour calculer la TFD et ainsi diminuer significativement la charge de calcul.

2.2 La Transformée de Fourier Rapide

En introduisant le premier algorithme de la FFT en 1965, Cooley&Tukey ont été capable de réduire d'une façon remarquable le temps de calcul de la TFD d'une suite de nombre d'échantillons N qui est une puissance de 2, connu sous le nom de radix-2 [JC65]. Depuis, et encore dans les dernières années, les algorithmes de FFT ont été dans le centre d'intérêt pour de nombreuses recherches qui ont révolutionné le traitement numérique de signal (*DSP : Digital Signal Processing*). La réduction du nombre d'opérations nécessaires, en particulier le nombre des multiplications constituent leurs avantages essentiels et est devenu l'approche de base dans toutes les nouvelles applications.

La grande majorité des algorithmes sont basés sur un même principe qui consiste à décomposer le calcul de la TFD en plusieurs sous-ensembles de TFD de longueur plus petite. En profitant des propriétés de symétrie et de périodicité suivant les équations (2.6) et (2.7) des facteurs de phases :

Symétrie :

$$W_N^{k+N/2} = -W_N^k$$
$$\left(W_N^k\right)^* = W_N^{-k} \tag{2.6}$$

Périodicité :

$$W_N^{k+N} = W_N^k \tag{2.7}$$

Où (*) désigne le complexe conjugué.

Malgré que le nombre des opérations serait réduit d'une façon remarquable, la complexité globale reste $O(N^2)$ [OA75] et son flot des données demeure complexe dans l'implémentation.

Les algorithmes de FFT permettent de faire une DFT d'une manière efficace ainsi que de réduire la charge de calcul en termes de multiplications et additions à valeurs complexes à l'ordre $O(N\log_2 N)$. Il existe de nombreux algorithmes FFT qui découlent tous de l'approche diviser pour régner (*divide and conquer approch*).

2.2.1 Algorithmes conventionnels

Les algorithmes les plus connus et les plus utilisés sont les algorithmes FFT où N est une puissance entière de deux, $N = 2^M$, où M un entier. Grâce à ces algorithmes, il est possible de réduire le nombre d'opérations nécessaires à un ordre de grandeur de $Nlog_2(N) = N\,x\,M$ [AV03].

Le tableau suivant donne un résumé des valeurs N^2, $Nlog_2(N)$, $N/log_2(N)$ pour diverses valeurs de N La dernière colonne montre de combien la vitesse de calcul peut être augmentée par l'utilisation de la FFT au lieu du calcul direct de TFD [AV03].

Tableau 1 Comparaison du nombre d'opérations dans un calcul direct de la DFT et dans la FFT [AV03]

Taille de la TFD N	**TFD** N^2	**FFT** $N.log_2(N)$	**Gain** $N/log_2(N)$
2	4	2	2.00
4	16	8	2.00
8	64	24	2.67
16	256	64	4.00
32	1024	160	6.40
64	4096	384	10.67
128	16384	896	18.29
256	64536	2048	32.00
512	262144	4608	56.89
1024	1048576	10240	102.40

2.2.2 Approche diviser-pour-régner

L'idée de cette approche consiste à représenter les vecteurs X[k] et x(n) sur deux dimensions (cas du radix- 2, radix-4, radix-8, etc…) ou bien plusieurs dimensions (cas du radix-2^i, radix-4^i). L'algorithme de la FFT et ses variantes sont détaillés dans de nombreux ouvrages, notamment Proakis et Manolakis [JG96]. Le principe de base de l'algorithme consiste à

faire un changement de variable de l'indice n sur deux dimensions entières. Par exemple, avec M pour les colonnes et L pour les lignes, on a :

$$N=ML \tag{2.8}$$

$$N= Ml+m \ ; \ 0\leq l\leq L-1, \ et \ 0\leq m\leq M-1 \tag{2.9}$$

$$k=Mp+q; \ 0\leq p\leq L-1, \ et \ 0\leq q\leq M-1 \tag{2.10}$$

Avec ce changement de variable, les points de séquence à transformer seront représentés sous forme matricielle. Ainsi, l'équation de la DFT devient :

$$\mathrm{X}[p,q] = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} X[1,m] W_N^{(Mp+q)(mL+1)} \tag{2.11}$$

Le terme $W_N^{(Mp+q)(mL+1)}$ peut être simplifié de la manière suivante :

$$W_N^{(Mp+q)(mL+l)} = W_N^{MLmp} W_N^{mLq} W_N^{MPl} W_N^{lq} \tag{2.12}$$

Toutefois selon (2.6) et (2.7)

$$W_N^{MLmp} = 1, W_N^{mLq} = W_M^{mq} = W_{N/L}^{mq} et \ W_N^{Mpl} = W_L^{lp} = W_{N/M}^{lp}$$

De là on obtient l'équation :

$$X[p,q] = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} X(1,m) W_M^{mq} \right] \right\} W_L^{lp} \tag{2.13}$$

L'équation (2.13) illustre le fonctionnement de la FFT. Soit la décomposition de la FFT à N points, en L FFT à M points. Pour résoudre l'équation (2.13), nous procédons d'abord à des TFD de M points correspondant à la sommation à l'intérieur entre crochets. Ensuite, la matrice résultante est multipliée point à point par les facteurs de phase W_L^{lp} et enfin des TFD sur L points sont effectuées en suivant l'autre dimension. En effectuant ces étapes, la complexité passe de l'ordre N^2 à l'ordre $N(M+L)$. Nous pouvons aussi récursivement faire d'autres changements de variables, en prenant un nombre premier pour L. Par la suite, nous factorisons M jusqu'à obtenir seulement des TFD ayant des tailles de nombre premier [JG96]. L'algorithme diviser pour régner peut se résumer dans les étapes suivantes [JG96] :

i. Calcul de M-point DFT, $F(l,q)$ de chaque ligne selon l'équation (2.14).

$$F(l,q) = \sum_{m=0}^{M-1} X(1,m) W_M^{mq} \tag{2.14}$$

ii. Multiplier le tableau résultant par le facteur de phase W_N^{lq}, en obtient $G(l, q)$ définit par :

$$G(l,q) = W_N^{lq} F(l,q) \tag{2.15}$$

iii. Finalement, calcul de L-points DFT de chaque colonne selon l'équation (2.16) :

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp} \tag{2.16}$$

2.2.3 Radix-2

Radix-2 ou en français Radical-2 ou base-2, signifie que la taille N de la FFT à calculer soit à base de 2. Sa structure est très simple, appelée structure Butterfly (papillon) à cause de son schéma en forme de papillon, elle peut être aussi appelée BPE (*Butterfly Processing Element*). On peut différencier entre deux algorithmes : DIT (*Decimation In Time*, Décimation dans le temps) et DIF (*Decimation In Frequency*, Décimation en fréquence). D'une part, quand la division en Butterfly, commence du côté de l'entrée (signal temporel), on l'appelle DIT et d'autre part, quand division en Butterfly, se fait du côté de la fréquence, cet algorithme est appelé DIF.

Dans ce travail nous nous intéressons à la version DIT pour l'implémentation sur FPGA sachant que la version DIF est similaire.

Le radix-2 utilise la technique 'diviser pour régner' [JW65], donc à l'aide de cette technique, il divise la FFT en sous-système de $N/2$ points, puis calcule chaque sous-système tout seul pour obtenir à la fin les composantes du spectre fréquentiel du signal.

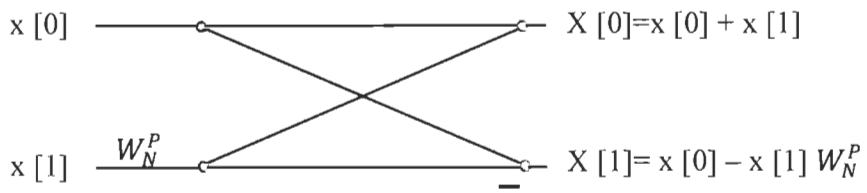


Figure 1 Structure DIT Butterfly

La figure 1 représente le schéma d'une structure Butterfly DIT de la FFT. On remarque que cette structure, avec $p = 0$, représente exactement la relation qui existe entre les

échantillons à l'entrée x[0] et x[1] et les échantillons en sortie X[0] et X[1], dans une TFD à 2 points. La figure 2 quant à elle, représente la structure d'une Butterfly DIF.

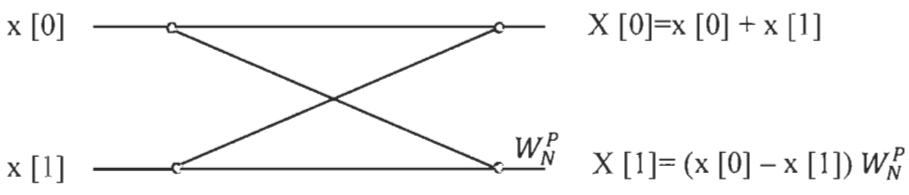


Figure 2 Structure DIF Butterfly

Les figures 1 et 2 représentent les deux structures Butterfly utilisées respectivement pour le DIT et DIF FFT. Pour le DIT, les facteurs de Fourier (TWF) sont multipliés par l'entrée x[1] puis le produit est, ou ajouté ou retranché, à x[0]. D'une autre part, pour le DIF les entrées x[0] et x[1] sont additionnées pour la première sortie puis pour la deuxième sortie, x[0] est soustraite de x[1] et on multiplie par le TWF.

Les figures 3 et 4 représentent les structures des deux algorithmes radix-2 DIT et DIF :

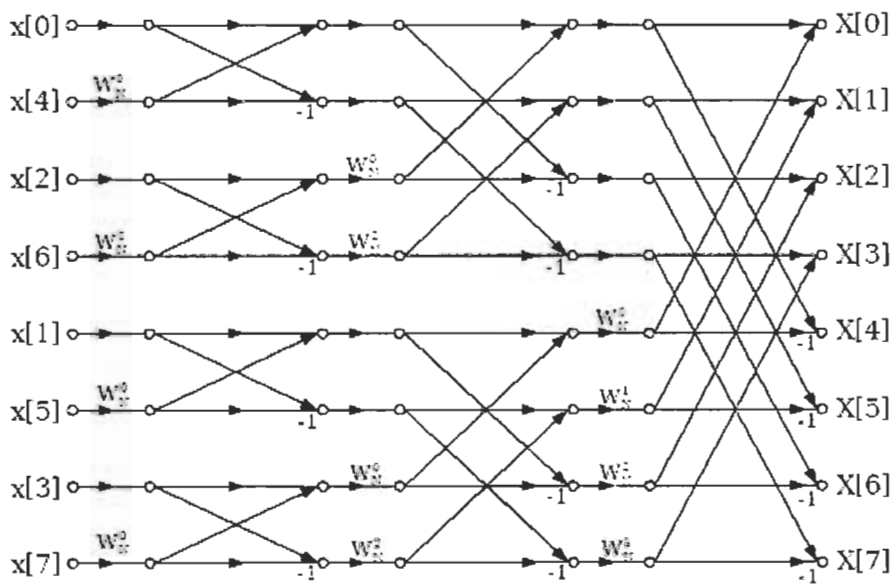


Figure 3 Diagramme de la FFT à base de BPE radix-2 de type DIT

Dans les figures 3 et 4, on représente la structure du radix-2 DIT et DIF pour 8 points.

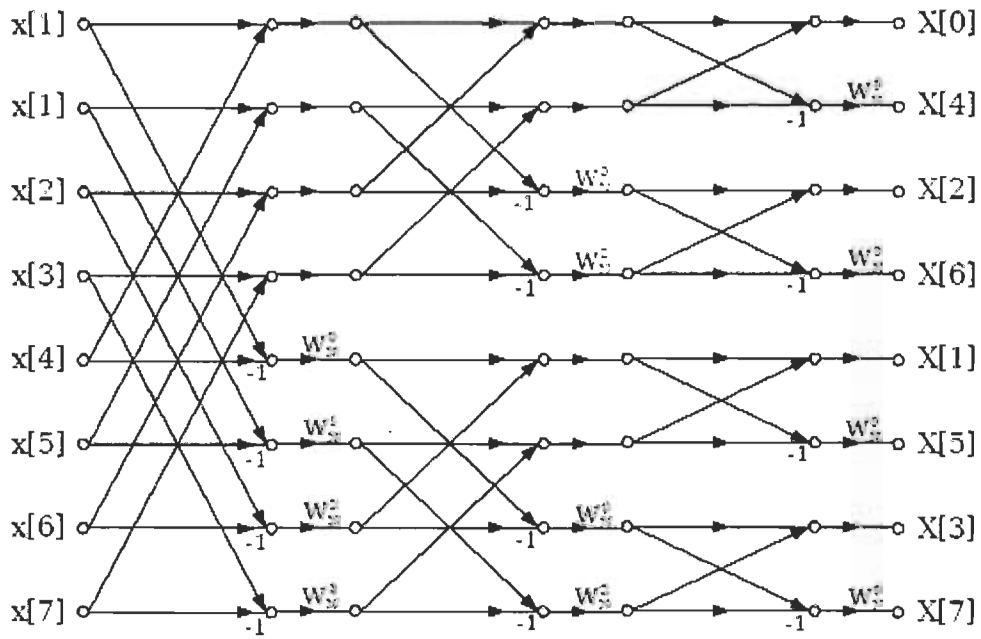


Figure 4 Diagramme de la FFT à base de BPE radix-2 de type DIF

Pour une taille N=8, pour un radix-2, on a $\log_2$ N étages pour calculer la FFT, donc trois étages pour notre cas N=8. On a aussi $N/2$ BPE à calculer par étage, soit 4 pour notre cas ici. Le radix-2, DIT ou DIF, réduit aussi l'ordre de calcul de N^2 à $N/2\ log_2N$ multiplications complexes et de N^2-N à $N\ log_2N$ additions complexes. Grâce à ces avantages, le radix-2 est l'algorithme FFT le plus utilisé actuellement. Il apparaît dans plusieurs articles de l'IEEE [YW07], [YT02], [YT04], grâce à sa simple architecture.

2.2.4 Radix-4

Le radix-4 est un algorithme FFT qui peut être utilisé dans le cas où on veut réduire le nombre de multiplications complexes d'environ 25% par rapport au radix-2 [TW97], [MB98]. Ceci est vrai dans le cas où la taille de la FFT est une puissance de 4.

La structure simplifiée du BPE radix-4, représenté dans la figure 5.

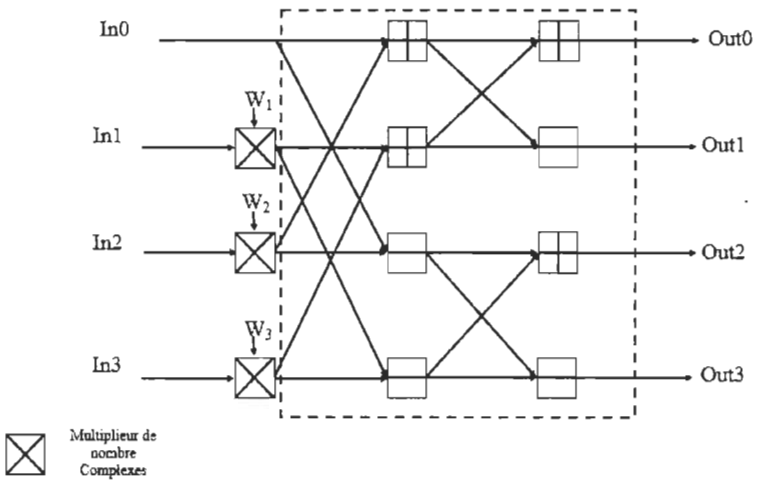


Figure 5 BPE du radix-4 DIT

Comme le montre le tableau 2, le nombre de multiplications complexes est réduit par rapport au radix-2 pour une taille de FFT plus grande.

Tableau 2 Nombre d'opérations d'une FFT de 4096 points pour différents radix

Opérations	Radix-2	Radix-4
Multiplications complexes	22528	15360
Additions complexes	49152	15360
Accès mémoire	49152	24576

2.3 Architecture parallèle pour le calcul de la FFT

L'architecture parallèle est utilisée grâce à sa rapidité et son efficacité. Cette architecture est une reproduction directe du mappage du SFG (*Signal Flow Graph*) de la FFT. En effet, *(N/r).log,N* BPE sont nécessaires pour calculer cette architecture, ce qui prouve que son besoin en ressources matérielles augmente rapidement avec la taille *N* de la FFT.

La figure 6 représente l'implémentation de l'architecture parallèle du SFG radix-2 de type DIT (avec *N*=8).

En plus de son énorme besoin de ressources matérielles, un des problèmes de cette architecture est la faible utilisation des BPE puisque les données d'entrée sont séquentielles tandis que les données de sorties sont générées en parallèle.

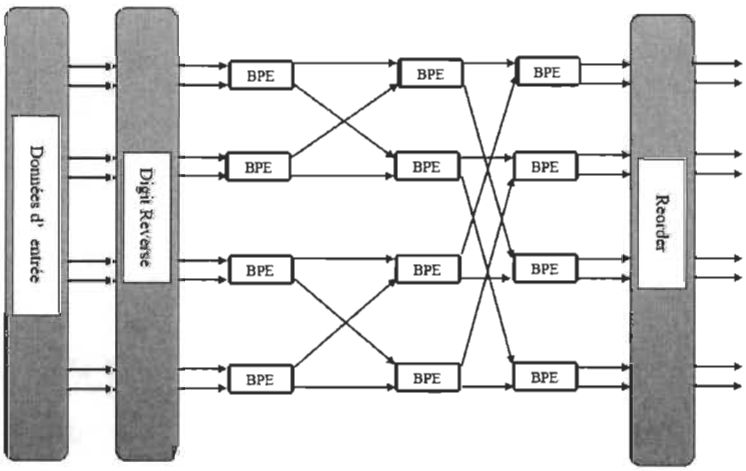


Figure 6 Architecture parallèle du radix-2 (N=8)

Ce problème peut être résolu si on alimente les entrées des BPE par N trames de données parallèles à chaque cycle d'horloge. On verra donc l'utilisation des BPE augmentée à 100% (figure 6). On constate aussi l'ajout de deux modules 'digit reverse' et 'reorder' qui assure que les données entrent et sortent dans le bon ordre pour l'obtention du bon calcul de la FFT,

[TW97]. La figure 7 montre un autre exemple de l'architecture parallèle, celui du radix-4 avec N=16 points.

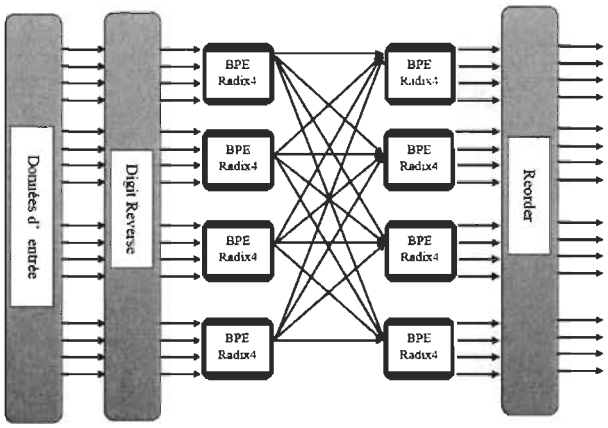


Figure 7 Architecture parallèle du radix-4 (N=16)

À partir de la figure 8, qui représente l'architecture parallèle du SRFFT (4/2) pour N=32 points, on remarque que plus la taille N augmente, plus l'architecture prend d'espace dans le processeur. Ce problème limite à des petites tailles de FFT. Cependant, elle reste une des meilleures solutions d'implémentation si la taille de la FFT ne dépasse pas 64 points.

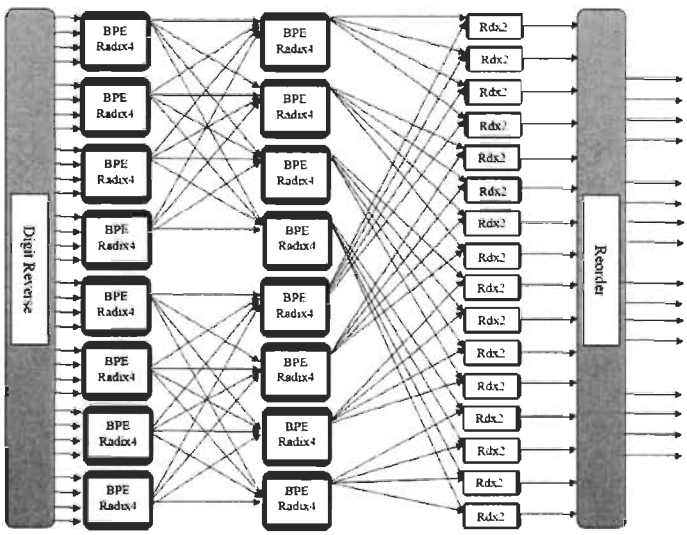


Figure 8 Architecture parallèle du SRFFT-4/2 (N=32)

D'autres avantages pour cette architecture peuvent être cités dont :

- L'implémentation efficace sur FPGA (grâce à la structure parallèle Manhattan).
- Optimisation dans l'utilisation du Butterfly.
- Réduction du nombre des TWF.
- Fréquence d'horloge qui peut atteindre 400 MHz sur les nouvelles FPGA Virtex5 de Xilinx.

2.4 Architectures pipelines pour le calcul de la FFT

L'architecture en pipeline radix-*r* FFT (figure 9) est caractérisée par un traitement continu des entrées séquentielles de la FFT. Une telle architecture est composée de $log_r(N)$ modules de calcul (MC), un par étage, qui correspond aux opérateurs papillon. La valeur de *r* correspond au radix de l'algorithme utilisé. Chaque BPE traite *N/r* opérations successives. Par conséquent, la taille maximale réalisable pour ce type d'architecture est dictée par le nombre de MC. En effet, ce type d'architecture offre un bon compromis complexité matérielle/taux de traitement de données pour les systèmes de communication à haut débit [YJ03].

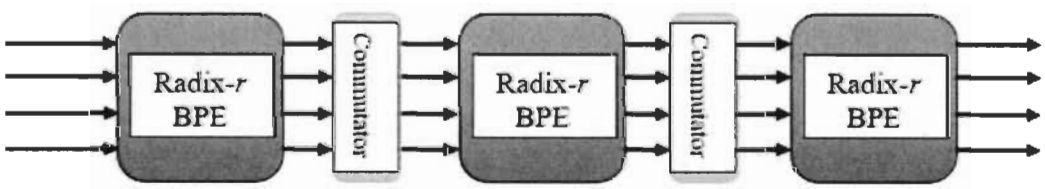


Figure 9 Architecture pipeline de base [YJ03]

Comme le montre la figure 9, l'architecture comporte un nombre de modules 'Commutator'. Ces derniers réorganisent les données entre les BPE et contiennent les modules de délais. À chaque étage du pipeline, un BPE, calcule *r* donnés de valeurs complexes, puis génère *r* donnés intermédiaires à la sortie. On a donc un débit de données *r* fois la fréquence d'horloge. Pour un BPE de radix-2 qui fonctionne à R MHz, les données sont traitées à une fréquence de 2R MHz, puisque deux données sont traitées au cours de chaque période d'horloge. [YJ03].

Il existe plusieurs architectures en pipeline basé principalement selon deux points :

1. Le premier point de différence est le nombre de chemins de données utilisés pour traiter les échantillons. Il y a deux types d'architectures : i) à chemin unique (*Single-path*; S) et ii) à chemin multiple *(Multi-path*; M).
2. Le deuxième point de différence consiste en la stratégie de mémorisation pour créer les différents délais nécessaires à l'ordonnancement des échantillons.

Selon ces deux critères, nous pouvons diviser les architectures pipelines en trois types [SH98] :

- Radix-*r* SDF: *Single-path Delay Feedback.*
- Radix-*r* SDC: *Single-path Delay Commutator.*
- Radix-*r* MDC: *Multi-path Delay Commutator.*

2.4.1 Single-path Delay Feedback SDF

L'architecture SDF (*Single-path Delay Feedback*) est largement utilisée pour les systèmes de télécommunications pour réduire la complexité matérielle. On peut trouver plusieurs

architectures SDF, tel que la R2SDF (*Radix-2 Single-path Delay Feedback*), la R4SDF, la R8SDF et la R2^{3}SDF. Par exemple, l'architecture SDF est considérée comme la meilleure approche afin de calculer la FFT pour un système OFDM à un seul canal parce qu'elle requiert le moins de ressources matérielles et qu'elle possède une seule entrée et sortie ce qui la rend très appropriée pour un système SISO-OFDM (*Single Input Single Output Orthogonal Frequency Division Multiplex*), [HS09], [SL07]. La figure 10 et 11 ci-dessous, représente deux architectures SDF, la R2SDF et la R4SDF.

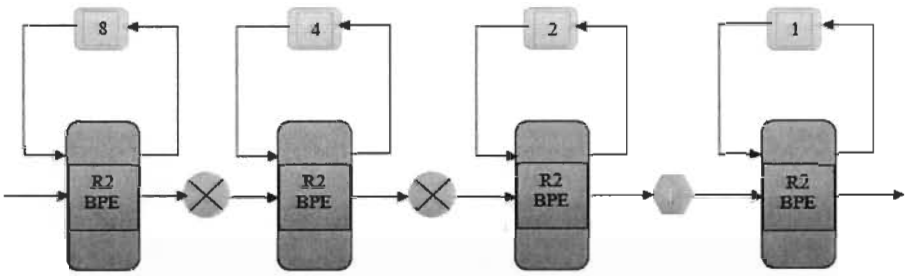


Figure 10 Architecture R2SDF (N = 16 points)

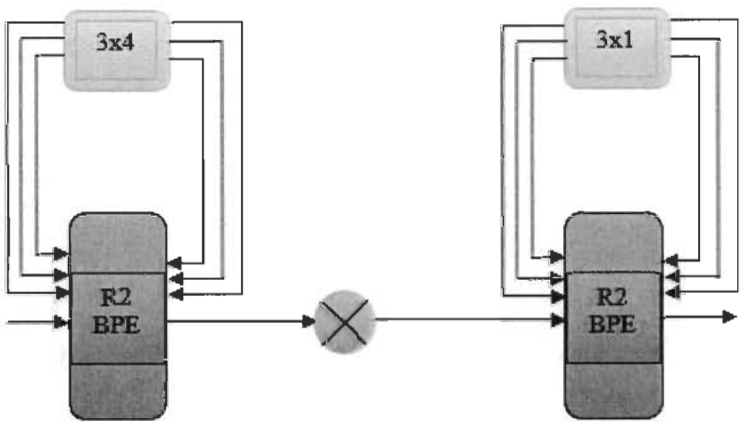


Figure 11 Architecture R4SDF (N= 16 points)

Les figures 10 et 11 représentent les architectures R2SDF et R4SDF respectivement pour N =16 points. Il est clair que l'architecture SDF utilise les registres de manières plus efficaces en stockant les sorties des BPE dans les registres à décalage de la chaîne à rétroaction. Dans l'architecture SDF, la séquence de données d'entrée passe par le chemin d'accès unique puis est réorganisée par l'unité de calcul BPE. À cause du chemin unique d'accès, un seul multiplieur complexe est utilisé après chaque BPE. La complexité matérielle est donc faible. Toutefois, l'architecture fonctionne à une faible vitesse et le contrôle de synchronisation des données demeure complexe, [YJ03]. Puisque l'architecture SDF possède un seul chemin d'accès d'entrée et de sortie et a un faible débit binaire, plusieurs coprocesseurs FFT R4SDF doivent être utilisés pour un système multi canal. Par exemple, pour un système MIMO-OFDM à 4 canaux, quatre architectures R4SDF doivent être utilisées (une pour chaque canal). On constate que la complexité matérielle augmente avec le nombre de canaux du système [HS09].

2.4.2 Single-path Delay Commutator SDC

Malgré que cette architecture utilise le même nombre de multiplieurs complexes que l'architecture pipeline SDF, elle utilise par contre un commutateur de délais qui possède une seule entrée et plusieurs sorties. Le nombre de sorties du commutateur varie selon le nombre d'entrées du BPE. La figure 12 représente une architecture pipeline R4SDC. À chaque coup d'horloge, les données sont injectées au BPE par le commutateur de délais. Ce dernier effectue ses calculs pour envoyer les résultats dans le chemin unique pour l'étage suivant.

Comme l'architecture SDF, l'architecture SDC opère à un faible débit binaire puisqu'elle possède un seul chemin d'accès d'entrée et de sortie.

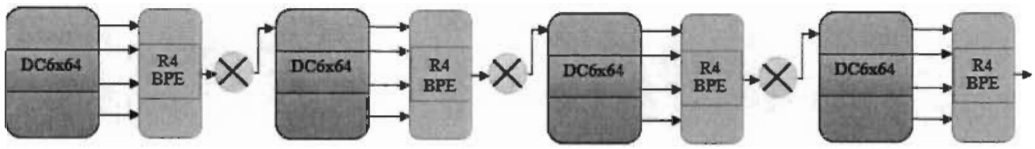


Figure 12 Architecture R4SDC (N = 256 points)

Donc, pour un système multi canal, elle a besoin de plusieurs coprocesseurs pour chaque canal du système ce qui augmente le besoin de cette architecture en ressources matérielles au cours de l'implémentation [YJ03], [HS09].

2.4.3 Multi-path Delay Commutator MDC

L'architecture radix-*r* MDC (*Multi-path Delay Commutator*) est considérée comme l'une des conceptions les plus utilisées dans la mise en œuvre des systèmes de communication à haut débit. Un des éléments qui fait partie de cette architecture est le commutateur qui a pour rôle de permuter les données entre les étages de l'architecture, on trouve aussi des délais pour synchroniser ces mêmes données. Dépendamment de l'algorithme radix-*r* utilisé, nous aurons *r*-1 délais avant et après chaque module BPE. De plus, cette architecture nécessite *r*-1 multiplieurs par module BPE.

On peut trouver plusieurs variétés de cette architecture, nous trouvons par exemple le R2MDC et le R4MDC qui sont les architectures les plus populaires, on trouve aussi les radix supérieures comme le radix-2^2 et radix-2^3. De plus, l'approche MDC est l'architecture la

plus adaptée dans les nouvelles applications puisqu'elle possède un multiple chemin d'entrées et de sorties [FB09]. Cette propriété permet aux modules FFT utilisant cette architecture d'opérer à haute vitesse avec un haut débit.

D'une autre part, cette architecture utilise plus de ressources matérielles. Une des méthodes utilisées pour réduire la consommation est d'aller plus haut dans le degré de la FFT implémentée.

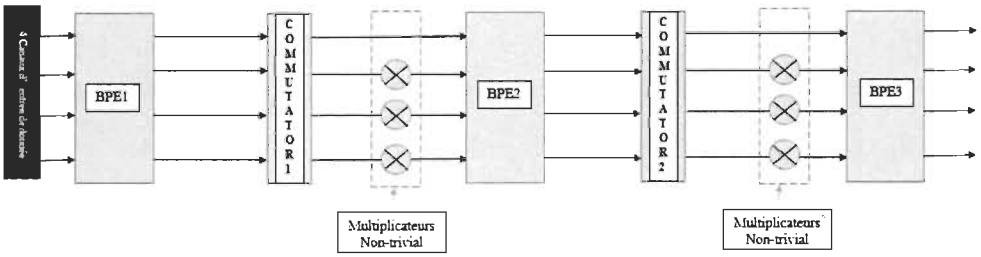


Figure 13 Architecture R4MDC (N = 64 points)

La figure 13 représente l'architecture pipeline R4MDC à 4 canaux pour N=64 points. Cette architecture utilise quatre chemins d'entrées en parallèle, et la même chose est trouvée en sortie. Le flux de données d'entrée et les données inter-étage sont contrôlés avec des commutateurs. Ces derniers sont implémentés par de simples éléments de délais et un multiplexeur, ce qui réduit leurs complexités matérielles au cours de leurs implémentations. Le rôle de ces commutateurs est de réorganiser les quatre flux d'entrées/sortie de données avant de les injecter dans l'étage suivant [YJ03].

La complexité matérielle du coprocesseur FFT de radix-r MDC, avec r la valeur du radix, peut être réduite en utilisant d'autres algorithmes comme le Mixed-radix plus et le Split-radix [ES84] [SL07].

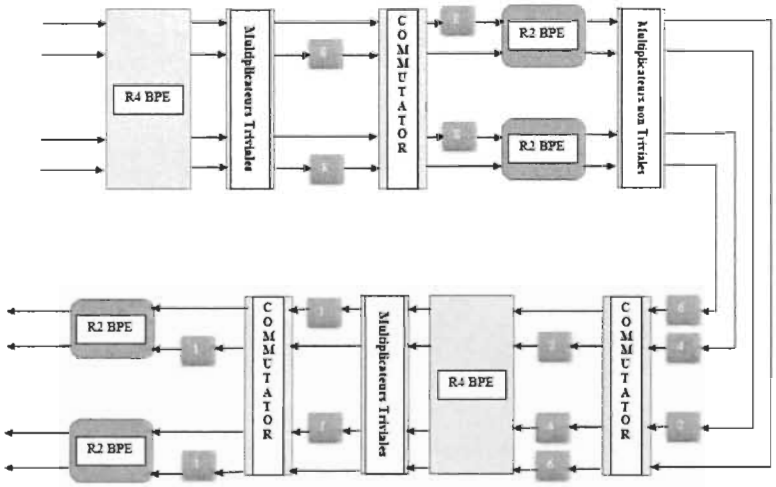


Figure 14 Architecture MC-MRMDC (k = 4, N = 64 points)

La figure 14 représente l'architecture MC-MRMDC (*Multi Channel - Mixed Radix Multipath Delay Commutator)* présentée dans [SL07] pour $N = 64$ points et $k = 4$, où k est le nombre de canaux. Cette architecture, utilise l'algorithme Mixed radix est permet de diminuer le nombre des ressources utilisées en diminuant le nombre de multiplications non-triviales.

En comparant les deux coprocesseurs radix-4 MDC et MC-MRMDC, on peut remarquer que le premier illustré dans la figure 13 utilise 6 multiplications non-triviales tandis que le deuxième illustré dans la figure 14 n'utilise que 4 multiplications non-triviales, [SL07].

Le tableau 3 contient une comparaison des exigences matérielles de plusieurs architectures FFT pipelines y compris le radix-2 2SDF, le radix-2^3 SDF et le MRMDC. La constante T désigne le nombre des additionneurs requis pour implémenter une multiplication triviale.

Tableau 3 Comparaison en termes d'exigences matérielles entre différentes architectures FFT pipelines [SL07].

	Nombre de Processeurs (P)	**Multiplieurs complexes/P**	**Additionneurs complexes/P**	**Taille Mémoire/P**	**Contrôle**
Radix2-SDF	k	Log_2 N-1	2Log_2 N	N-1	Simple
Radix4-SDF	k	Log_4 N-1	8Log_4 N	N-1	Moyen
Radix4-SDC	k	Log_4 N-1	3Log_4 N	2N-2	Complexe
Radix2^2-SDF	k	Log_2 N-1	4Log_4 N	N-1	Simple
Radix2^3-SDF	k	2(Log_8 N-1)	(62T)Log_8 N	N-1	Moyen
Radix2-MDC	k/2	Log_2 N-2	2Log_2 N	3N/2-2	Simple
Radix4- MDC	k/4	3(Log_4 N-1)	8Log_4 N	5N/2-4	Simple
MRMDC	k/4	4(Log_8 N-1)	(12+3T)Log_8 N	5N/2-4	Simple

D'après le tableau 3, pour k = 4 et N = 64, l'architecture MC-MRMDC sauve 2 multiplications non-triviales comparant à l'architecture radix-4 MDC et 76 additionneurs complexes comparant à l'architecture radix-2^3 SDF. On constate donc, qu'en utilisant l'architecture pipeline MDC, on utilise moins de processeurs. À titre d'exemple ; pour implémenter un système à 4 canaux ; on aura besoin de 4 processeurs utilisant l'architecture pipeline SDF au lieu d'un seul processeur utilisant l'architecture MDC. On réalise ainsi un grand gain en termes de ressources matérielles. Ce qui constitue la raison que la plupart des implémentations VLSI des coprocesseurs FFT tendent à utiliser l'architecture pipeline MDC. [SL07].

2.5 Effet de Quantification

Les unités de calcul en virgule flottante sont des unités trop complexes est présente un coût matériel trop important. Pour cette raison, le calcul est réalisé généralement avec des nombres représentés en virgule fixe afin de satisfaire les coûts en VLSI pour les implémentations FPGA. La figure 15 résume quelques méthodes de représentations en virgule fixe.

Généralement, l'arithmétique en virgule fixe est préférée pour les applications de DSP grâce à leurs avantages en termes de consommation d'énergie, de surface et de latence.

De plus, toutes ces performances dépendent de la représentation en virgule fixe utilisée, ainsi, un choix judicieux de la largeur binaire dans la conception FPGA peut entraîner des économies substantielles.

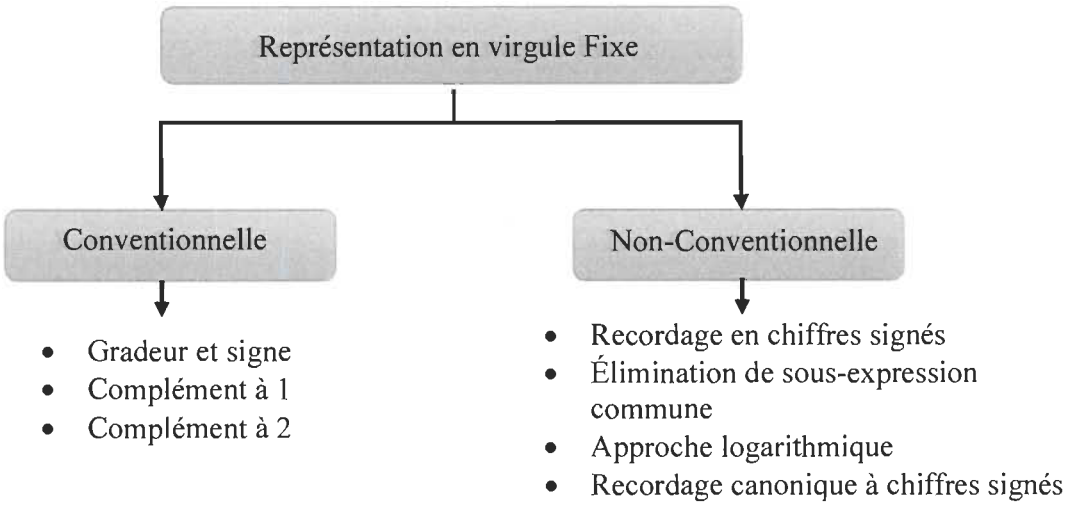


Figure 15 Quelques méthodes de représentations en virgule fixe

2.5.1 Erreur de quantification

Lorsqu'une FFT est effectuée en arithmétique à virgule fixe, des erreurs de quantification interviennent, liées à la longueur certainement finie des valeurs numériques codées à l'intérieur de la machine sous la forme de mots binaires. Elles impliquent une incertitude sur le résultat qui est ainsi affecté des sources de bruits.

Nous pouvons distinguer trois sources d'erreurs de quantification initiales selon [KM91] :

- L'erreur de quantification due au signal d'entrée liée à la résolution du convertisseur analogique-numérique.
- L'erreur de quantification due aux facteurs de Fourier W de la FFT.
- L'erreur de quantification dans les unités de calculs ; après multiplications et/ou les additions.

Ces trois types d'erreurs se propagent de l'entrée vers la sortie au sein du système et modifient la précision des calculs en sortie de l'application. Dans de nombreux cas, il est raisonnable de traiter les effets de troncatures et d'arrondis par un modèle statistique remplaçant chaque source d'erreur par un bruit blanc.

2.6 Différentes approches d'implémentation de la FFT

2.6.1 Processeur FFT pipeliné localement

L'architecture pipeline est largement utilisée dans de nombreuses applications [SS96], [TM92] à haut débit, mais pour le processeur FFT dans les applications embarquées en temps réel, un long pipeline consomme trop de silicium, d'où l'architecture [LH99] composée d'un

pipeline court et d'une mémoire partagée est plus adaptée. Cette structure nommée le processeur pipeliné localement (*Locally Pipelined Processor : LPPL*) ressemble à une architecture à radix élevé et à mémoire partagée (*High-Radix and Shared Memory Architecture : HRSMA*), mais différemment a l'HRSMA générale, son BPE à haute radix est implémentée avec un BPE à faible radix pipeliné ; par conséquent, il peut être réalisé avec un nombre de ressources matérielles acceptable et un débit élevé.

Comme le montre la figure 16, le BPE pipeliné et le générateur d'adresses à la mémoire partagée sont deux des éléments de base du processeur LPPL. Dans la conception d'architecture de pipeline [SS96], [EH84] - [SW01], l'architecture radix-4 tend à réduire les multiplicateurs à 50% par rapport à celle de radix-2, mais le nombre d'additionneurs qui augmente à 400% dans les BPE radix4 classiques (BPE4) fait disparaître cet avantage. Dans le radix-2 Single Deep Delay Feedback (R2SD2F), le BPE4 est formé par deux BPE radix-2 en cascade (BPE2), ce qui diminue l'utilisation des additionneurs par 50%, mais double l'utilisation de BPE de 25% à 50% avec maintenir le même nombre de multiplicateurs et registres comme le radix-4 Single-Path Delay Feedback (R4SDF [AM74]). Dans une autre approche, le radix-4 Single-Path Delay Commutator (R4SDC) proposé par [GA89] a réduit le nombre d'additionneurs contenus dans BPE4 de 8 à 3, mais a doublé les ressources utilisées pour les registres.

Pour la génération d'adresse, la méthode la plus populaire et la plus efficace sur le plan de la génération d'adresses de coefficients a été proposée par Cohen [DC76]. Selon cette méthode et d'autres approches similaires [YM99], [YM20], la génération d'adresse se produit par l'application de différents shifts pour traiter les lignes d'adresses, ce qui

correspond très bien à la caractéristique « *Frequent Coefficient Accessing* » et réduit également le temps d'accès pour obtenir les coefficients.

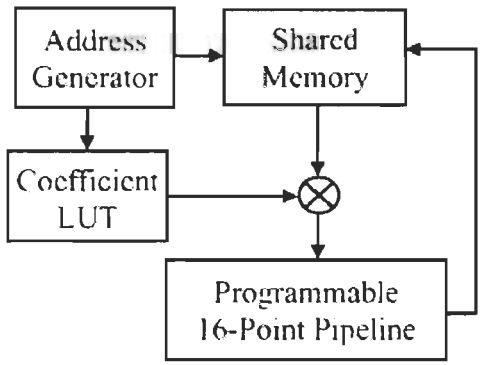


Figure 16 Architecture radix-16 LPPL [GA89]

Mais ce type d'algorithme a besoin de *M-1* ($M=log_2 N$) cycles pour calculer l'adresse de l'opérande, la latence est alors trop longue. Ma [YM94] et Luo [WZ94] ont fourni différents algorithmes de génération d'adresse rapide. Cependant, Ma [YM94] et Luo [WZ94] n'ont pas considéré la redondance de l'accès aux coefficients ce qui augmente notablement la consommation d'énergie de l'accès mémoire.

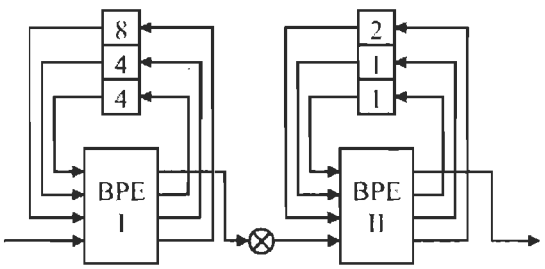


Figure 17 Architecture R2SD2F pipeliné pour DFT à 16 points [GA89]

Une autre approche appliquée au R2SD2F pipeliné pour LPPL est aussi proposée par mettre en cascade deux « *"deep" feedback butterflies* ». Dans une telle architecture, un pipeline de longueur 16 comprend quatre additionneurs et un multiplicateur. Les exigences des

opérateurs sont minimales par rapport aux autres modèles de pipelines, et l'efficacité de la performance du module de traitement peut atteindre jusqu'à 100%.

2.6.2 Processeur FFT pour les applications WPAN à base de OFDM

Une parmi les différentes approches pour implémenter la FFT consiste à concevoir un processeur de transformée de Fourier rapide (FFT) qui fournit un débit de débit élevé (*High Throughput Rate (TR)*) en appliquant l'approche pipeline de huit voies pour les applications de réseau sans fil (*Eight-data-Path Pipelined Approach for Wireless Aersonal Area Network*). Au cours de ces dernières années, des efforts ont été réalisés dans les applications WPAN (*Wireless Personal Area Network*) pour supporter une transmission élevée des données dans un environnement intérieur à courte distance (*Short Range Indoor Envirenment*).

Dans les développements précédents, la technologie « *Ultra Wide Band* » (UWB) a permis de fournir des débits de données allant jusqu'à 480 Mb/s. Cela signifie que le processeur FFT pour les systèmes WPAN avancés devrait fournir un TR élevé d'un minimum de 2.304 GS/s.

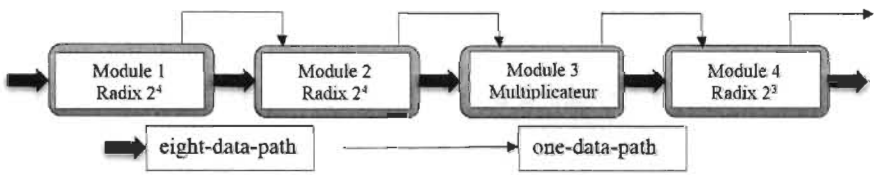


Figure 18 Diagramme Block d'un 2048-FFT processeur [SN10]

Pour cette raison, un processeur FFT de 2048 points basés sur l'approche « *Eight-Data-Path Pipelined Approach* » avec une méthode de réduction de ressources qui peut fournir un T.R élevé a été conçu pour atteindre 2,4 GS/s et un SQNR de 32,8 dB pour les applications

16-QAM (*16 Quadratic-Amplitude Modulation*).

Comme Montré à la figure 18. Il existe quatre modules basés sur l'algorithme radix-2^4 qui réduit le nombre de multiplications non triviales [MS08], [JY05]. Les données sont traitées en fonction de huit chemins de données avec un exposant partagé attribué à un groupe de huit valeurs de mantisse. Les détails de chaque module sont décrits comme suit.

A. Module 1

Comme représenté sur la figure 19 le module 1 a une structure radix-2^4 avec quatre opérations de radix-2 divisées sur 4 étapes. Les données de la mantisse sont exploitées par représente Butterfly radix-2, il existe quatre unités BF2 à huit données et quatre FIFO à huit banques avec des tailles de mots de 1024, 512, 256 et 128, respectivement. L'unité BF2 effectue une addition et une soustraction de l'entrée pour l'opération radix-2 ou bypass huit chemins de données, et l'exposant est exploité par une seule donnée chemin. Le BF2 l'entrée pour les opérations d'entrée / sortie.

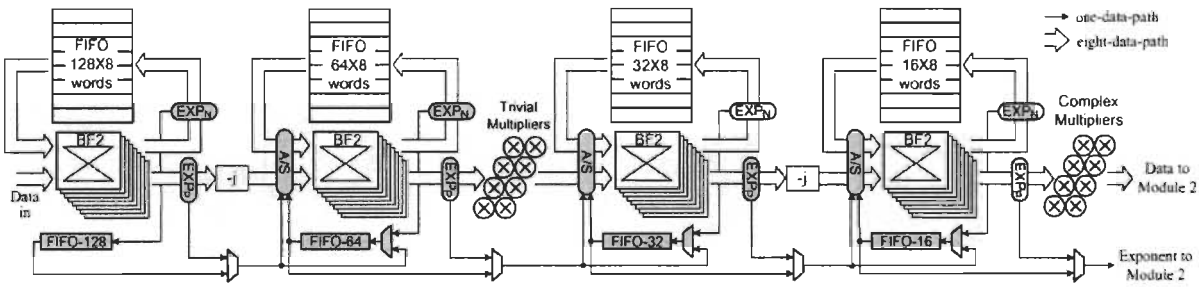


Figure 19 Structure du module 1[SN10]

B. Module 2

La structure du module 2 est similaire à celle du module 1 sauf que les tailles de données FIFO sont 64, 32, 16 et 8 mots, respectivement, pour les quatre étapes. En outre, la

multiplication au dernier stade est complétée par une unité de multiplication constante (Module 3) au lieu de huit multiplicateurs complexes généraux.

C. Module 3

Les huit valeurs de données de sortie du module 2 doivent être simultanément multipliées par les TWF non triviales dans le module 3. Certains schémas de simplification ont été présentés en [YW05] et [KM04] en utilisant des nombres de multiplicateurs constants au lieu de multiplicateurs complexes.

D. Module 4

La phase finale pour l'opération radix-2^3 est réalisée dans le module 4, comme l'illustre la figure 20, c'est une mise en œuvre directe basé sur l'organigramme du signal radix-2^3 avec 24 additionneurs complexes et deux multiplicateurs constants pour les TWF triviaux.

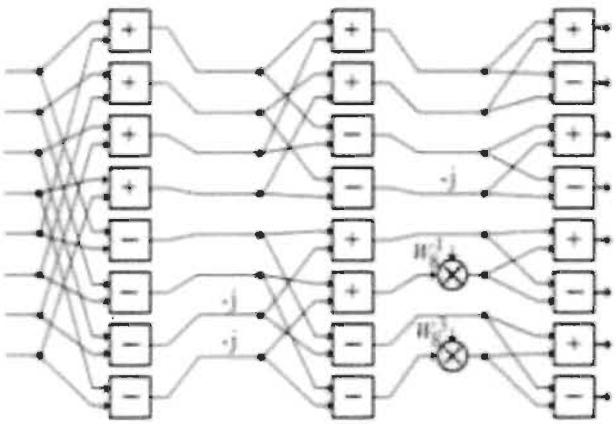


Figure 20 Diagramme Block du module 4[SN10]

2.6.3 Radix-2^4 FFT à haute performance pour les applications MIMO-OFDM

Orthogonal Frequency Division Multiplexing (OFDM) est une méthode de modulation connue pour sa capacité à atteindre une efficacité élevée et à faire face à la « *frequency selective fading and narrowband interference* ». [KM04]

Ainsi, un schéma de processeur FFT / IFFT à faible vitesse peut réduire la complexité des systèmes MIMO-OFDM [GL04] [DB04]. Et, puisque le taux de transmission de données des systèmes MIMO-OFDM ne cesse à augmenter, générer des symboles OFDM avec un débit élevé nécessite un algorithme et une architecture FFT à haute efficacité. [HL08]

L'algorithme radix-2^4 est décrit en détail dans [CC92]. L'algorithme peut prendre un multiplicateur complexe constant au lieu du multiplicateur complexe programmable.

La figure 21 montre l'architecture du processeur R2^4MDF FFT / IFFT proposé avec quatre chemins de données parallèles à 128/64 points. La conception proposée peut soutenir l'opération de FFT / IFFT dans 128 points ou 64 points. Il se compose d'unités de mémoire, d'unités de Butterfly appelées BF_64, BF_128, BF1, BF2, de multiplicateurs de nombres complexes, multiplicateurs constants, multiplexeurs et additionneurs constants. [HL08]

L'architecture '4-parallel data-path' présentée dans la figure 21 demande aussi quatre entrées et sorties qui exploitent séparément les données réelles et imaginaires. L'unité de Butterfly est le noyau de ce processeur FFT. Dans chaque Butterfly, deux types d'unités ont été conçus pour réaliser la multiplication par 'j'. Le BF1 stocke toutes les $(N / 2)^{ème}$ données d'entrée dans RAM. Pendant ce temps, la RAM maintient l'espace de mémoire tampon pour les données d'entrée suivantes. Le BF2 est identique à l'architecture BF1 sauf l'opération de $(3N / 4)^{ème}$ données d'entrée multipliées par -j. [HL08]

Cette architecture proposée nous permet de traiter des données à grande vitesse avec une faible complexité matérielle grâce à l'application d'un multiplicateur à largeur fixe, qui maintient l'entrée et la sortie à une largeur de 10 bits à 30dB SQNR. En outre, le nombre de multiplicateurs complexes et des mémoires sont effectivement réduits en utilisant l'algorithme radix-2^4 MDF FFT.

Les résultats de performance montrent que la fréquence de traitement des données est aussi élevée que 560 Msample/s avec une petite augmentation en complexité matérielle. [HL08]

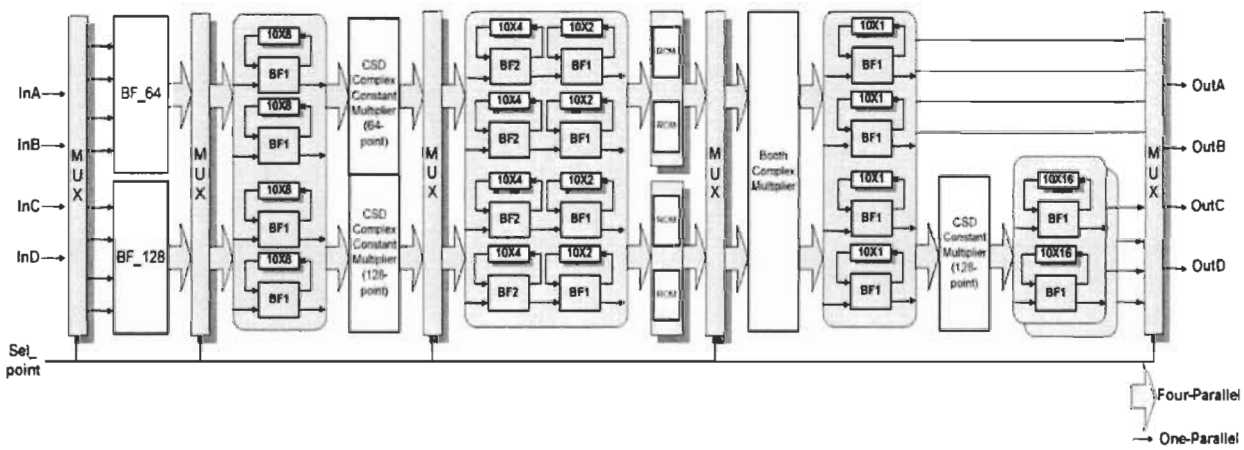


Figure 21 4-parallel data-path d'un processeur radix-2^4 MDF FFT/IFFT à 128/64-points [HL08]

2.6.4 Processeur FFT pour les systèmes UWB

Dans les systèmes Multi-Band Orthogonal Frequency Division Multiplexing (MB-OFDM), l'horloge d'échantillonnage dans les convertisseurs analogiques-numériques (A/D) est de 528 MHz, ce qui est trop élevé pour concevoir le récepteur en utilisant les technologies de processus CMOS actuelles. Dans ce cas, une structure réceptrice parallèle y compris un bloc

FFT peut être considérée pour limiter l'horloge système du modem à un maximum de 132 MHz pour l'implémentation VLSI [CH07]. Un processeur FFT a été conçu avec une architecture parallèle pour répondre à la contrainte du système. Les algorithmes DIF de radix-2^4 et Single-Path Delay Feedback (SDF) ont était également utilisé afin de réduire le nombre de multiplicateurs. [SI08]

Le schéma fonctionnel du processeur FFT parallèle à 128 points est illustré à la figure 22. L'architecture FFT proposée se compose d'unités Butterfly (Types 1, 2 et 3), CCM (*Constant Coefficient Multipliers*), CVM (*Complex Variable coefficient Multipliers*) et registres.

Les unités BPE effectuent une addition complexe et une soustraction complexe avec les deux données d'entrée complexes. Comme le montre la figure 23, trois types d'unités tampons sont utilisés dans ce processeur FFT proposé.

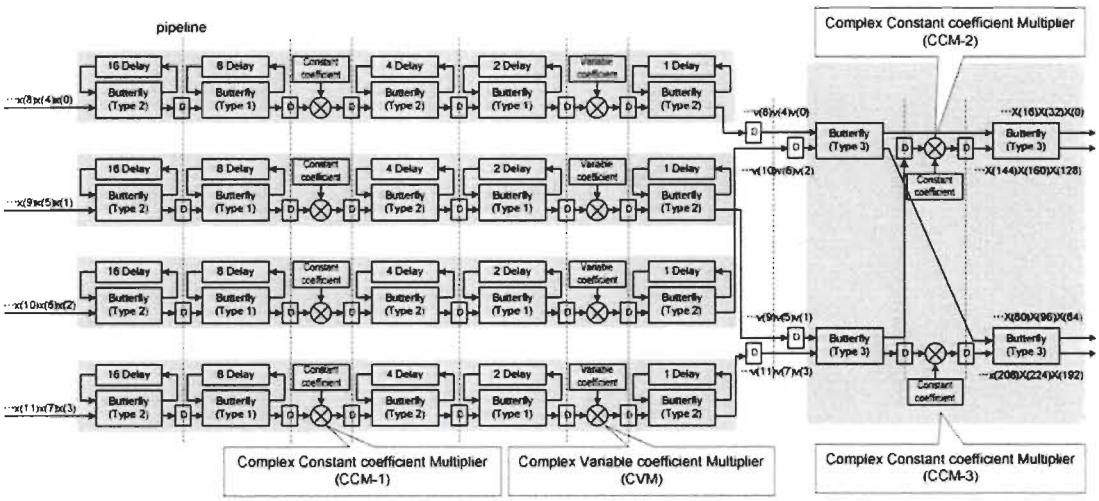


Figure 22 Processeur SDF FFT radix-2^4 parallèle proposé [SI08]

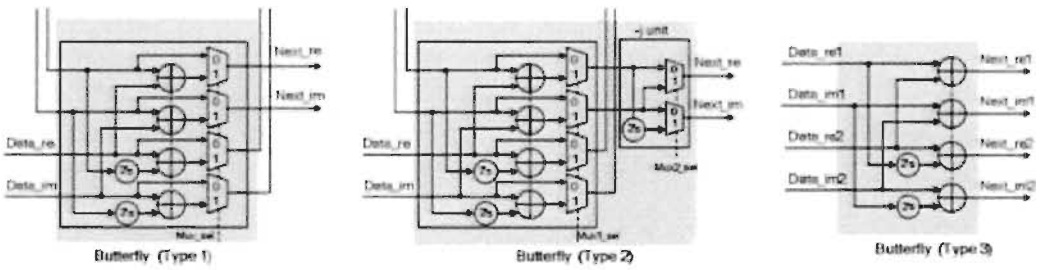


Figure 23 Structures des unités BPE (Types 1, 2, et 3) [SI08]

L'implémentation de cette architecture permet de réduire environ 40% de la complexité matérielle ainsi que la consommation d'énergie. Les résultats de la simulation ont montré que le processeur parallèle radix-2^4 FFT utilisant le DIF et les algorithmes SDF peut atteindre des fréquences de traitement allant jusqu'à 1 GSample/s avec relativement faible complexité matérielle. [SI08]

2.6.5 Radix-2^k feedforward FFT pipelinées

À l'origine, les architectures SDF et MDC ont été proposées pour radix-2 [HG70], [BG73] et radix-4 [AD74]. Quelques années plus tard, radix-2^2 a été présenté pour la SDF FFT [SH98] comme une amélioration pour radix-2 et radix-4. Ensuite, radix-2^3 et radix-2^4, qui permettent de simplifier certains multiplicateurs complexes, ont également été présentés pour la SDF FFT. Une explication des architectures radix-2^k SDF peut être trouvée dans [AC09]. Cependant, radix-2^k n'avait pas été considéré pour les architectures feedforward jusqu'à ce que les premières architectures FFT feed-forward radix-2^2 aient été introduites il y a quelques années [MG09].

La figure 24 montre l'architecture proposée pour une FFT radix-2^2 feedforward avec N=16. L'architecture est composée de BPE radix-2 (R2), de multiplieurs non triviaux (⊗), de multiplieurs triviaux en forme de diamant et de structures de commutation *(shuffling structures)*, qui consistent des buffers et des multiplexeurs. La longueur des buffers est indiquée par un nombre. [MG13]

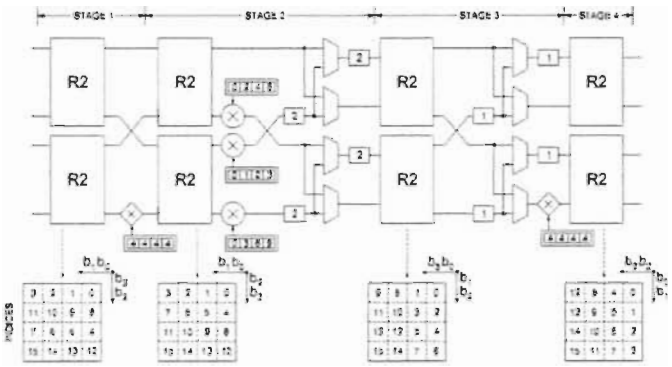
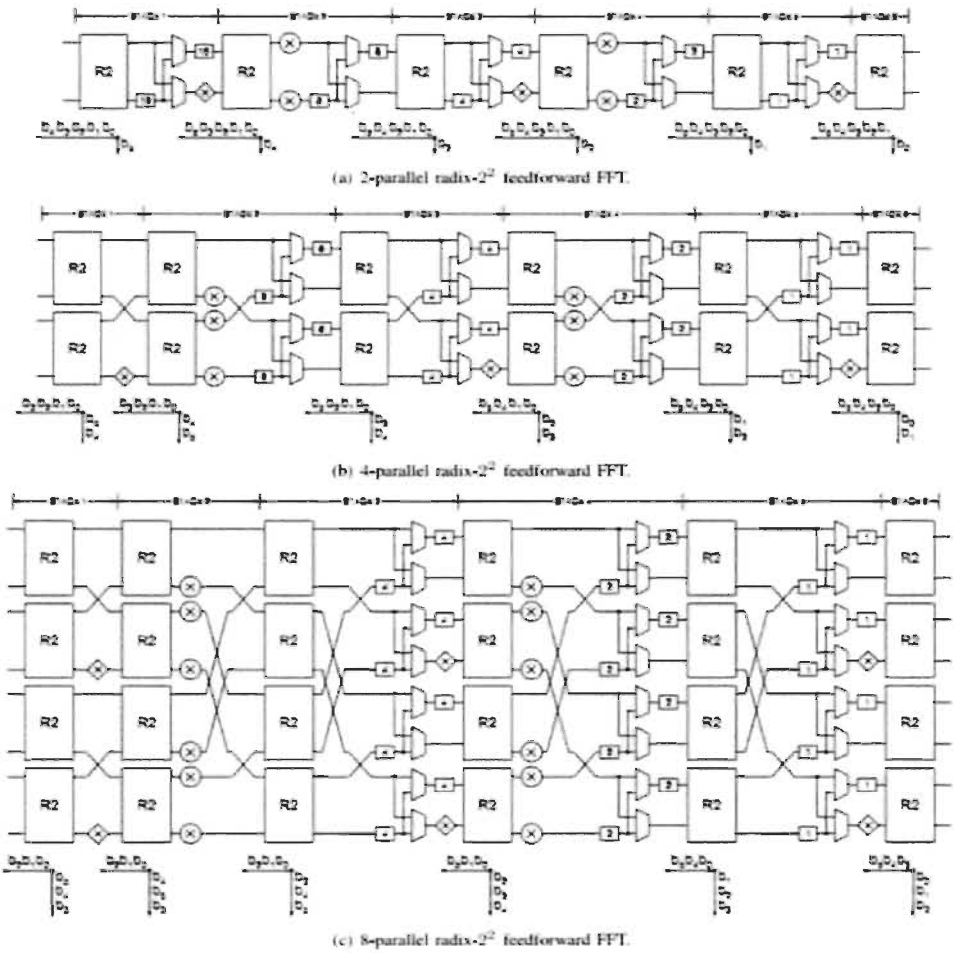


Figure 24 Architecture radix-2^2 feedforward à 4 entrées parallèles pour le calcul de DIF FFT de 16-points [MG13]

L'architecture traite quatre échantillons en parallèle dans un flux continu. L'ordre des données aux différentes étapes est représenté au bas de la figure par ses indices, avec les bits 'bi' qui correspondent à ces indices. [MG13]

La figure 25 montre les architectures feed-forward radix-2^2 proposées pour le calcul de la FFT DIF à 64 points. Les figures 25 (a), 25 (b) et 25 (c) montrent respectivement les cas de 2 entrées parallèles, 4 entrées parallèles et 8 entrées parallèles. [MG13]

Ces circuits peuvent être analysés comme cela a été fait pour l'architecture de la figure 25. Généralement, les architectures feedforward, radix-2^k peuvent être utilisés pour n'importe quel nombre d'échantillons parallèles qui est une puissance de deux. En effet, le nombre d'échantillons parallèles peut être choisi arbitrairement en fonction du débit requis. [MG13]



(a) 2-parallel radix-2^2 feedforward FFT.

(b) 4-parallel radix-2^2 feedforward FFT.

(c) 8-parallel radix-2^2 feedforward FFT.

Figure 25 Architecture radix-2^2 feedforward proposée pour le calcul de la DIF FFT de 64-points [MG13]

De plus, les décompositions DIF et DIT peuvent être utilisées. Enfin, les résultats expérimentaux montrent que les conceptions sont efficaces en performance ainsi qu'en espace utilisé, il est aussi possible d'obtenir des débits de l'ordre des GSamples/s ainsi que des latences très faibles. [MG13]

CHAPITRE III

IMPLÉMENTATION DE LA FFT RADIX-2^a

Dans ce chapitre, nous étudierons l'implémentation de la radix-2^a MDC FFT pour différentes longueurs de données (16 points, 64 points et 256 points). Nous exposerons en détail les étapes qui nous ont permis de concevoir leurs architectures. Nous testerons le bon fonctionnement de ces dernières à l'aide d'une co-simulation Matlab-Modelsim.

La partie d'implémentation (synthèse) viendra pour confirmer l'efficacité réelle de la nouvelle formulation de la BPE (*Butterfly Processing Element*) en termes de temps de calcul

et des ressources matérielles d'implémentation en technologie FPGA (*Field Programmable Gate Array*). Ce chapitre représente aussi les résultats d'implémentation obtenus des différentes grandeurs de FFT.

3.1 Éléments nécessaires pour implémenter la FFT

3.1.1 Étapes de conception

Dans un premier temps, la programmation de la FFT est effectuée sur Matlab®, cette étape permettra de valider le bon fonctionnement des différents blocks de l'architecture proposée. Le succès de cette validation nous permet de bien définir les opérateurs et les éléments sensibles qui rentrent dans la conception des architectures des BPE et dans la structure pipeline de la FFT. Pour la conception de ces derniers, on utilisera un langage de description de circuits électroniques, le 'VHDL'. Le VHDL appelé aussi VHSIC (*Very High Speed Integrated Circuit*), est un langage utilisé pour décrire le fonctionnement des structures électroniques, de concevoir des circuits logiques programmables (synthèse) et de les simuler pour valider leur fonctionnement [DL02]. En VHDL, tout composant (dans le sens logiciel) est décrit sous deux aspects : L'interface avec le monde extérieur, décrite dans une section dénommée *entity* (**entité**) et l'implémentation elle-même, décrite dans une section dénommée **architecture**. C'est donc l'architecture qui contient la fonction matérielle qu'on souhaite implémenter, [DL02].

Le logiciel utilisé pour le projet est Modelsim®. L'étape de vérification et validation du bon fonctionnement des différents blocks de l'architecture crée sur Modelsim® sera faite en utilisant la plateforme Matlab®. Une fois que l'architecture est validée et fonctionnelle pour

différentes grandeurs de données, on passe à l'étape de synthèse qui est effectuée avec l'outil Xilinx XST®.

Cette dernière étape nous permettra d'évaluer les performances de la nouvelle formulation de la FFT, en termes de temps de calcul et de ressources matérielles. Le tableau 4 résume les outils utilisés pour les différentes étapes de conception de l'architecture FFT.

Tableau 4 Outils de conception

Étape de conception	***Outils***
Simulation VHDL	Modelsim PE
Synthèse VHDL	Xilinx XST ISE
Cible FPGA	Famille Virtex-5(Pour des fins de comparaison avec la littérature) Familles plus récentes
Vérification	Matlab R2014a

Dans les sections qui suivent, on construira les différents éléments qui composent les coprocesseurs élémentaires. Il faut noter aussi que toutes les données qui seront traitées par nos coprocesseurs FFT seront comprises entre -1 et 1 (normalisation). Nous devrons donc utiliser la représentation suivante (pour un format de mots binaires de 16 bits) :

Bit 15 = signe (0 pour positif et 1 pour négatif).

Bit 14=2^{-1}, Bit 13=2^{-2}, Bit 12=2^{-3}, Bit 11=2^{-4}, Bit 10=2^{-5}, Bit 9=2^{-6}, Bit 8=2^{-7}, Bit 7=2^{-8},

Bit 6=2^{-9}, Bit 5=2^{-10}, Bit 4=2^{-11}, Bit 3=2^{-12}, Bit 2=2^{-13}, Bit 1=2^{-14}, Bit 0=2^{-15}.

Les nombres négatifs sont représentés en notation en complément à 2.

3.1.2 Structure pipeline de la FFT

Le choix de l'architecture pour implémenter la FFT est un élément très sensible et très important. Dans le chapitre 2, on a montré les différents avantages pour l'architecture pipeline. C'est à cause de ces avantages que l'architecture pipeline est une des architectures les plus populaires et les plus adaptées pour les applications du traitement de signal où le haut débit de données est une condition dominante [ES84].

Dans notre projet, on étudie et utilise particulièrement l'architecture pipeline MDC puisqu'elle propose un faible temps de latence, une forte utilisation et exploitation des BPE et surtout requière une faible utilisation de mémoire comparée à plusieurs autres architectures pipelines. Pour être conforme à la référence on n'a pas étudié l'introduction du bit reverse dans l'architecture de la FFT programmée, l'ordre d'entrée des données est contrôlé d'une façon manuelle. La figure 13 représentée dans le chapitre 2 montre la structure de l'architecture pipeline utilisée lors de notre projet.

Cette architecture répond très bien aux exigences des normes et standards de l'ITU en termes de latence, vitesse de calcul et utilisation de mémoire. L'utilisation de l'architecture pipeline MDC dans la plupart des implémentations VLSI des coprocesseurs FFT est justifié dans l'article [SL07] qui prouve que pour implémenter un système multicanal ; on aura besoin de 4 processeurs utilisant l'architecture pipeline SDF au lieu d'un seul processeur utilisant l'architecture MDC. On réalise ainsi un grand gain en termes de ressource.

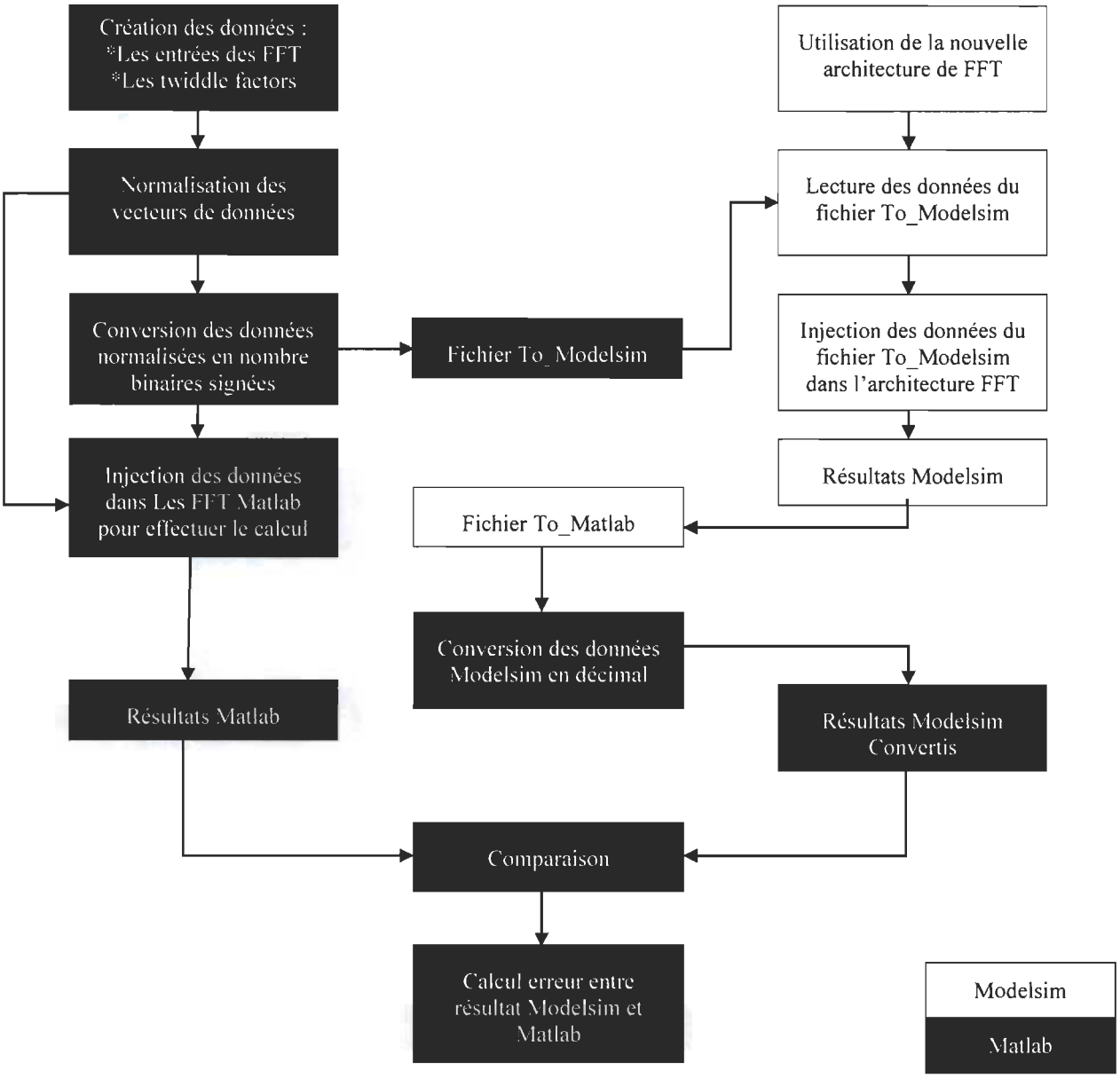


Figure 26 Diagramme de la méthode de co-simulation Matlab-Modelsim

La figure 26 montre la démarche qu'on a utilisé pour réaliser l'implémentation de la FFT, cette démarche et toujours répétée avec N points d'entrée de la FFT (N=16, 64 et 256).

3.1.3 Conception et architecture pipeline radix-2^2 et radix-2^3 FFT

Dans cette section, on introduira toutes les informations nécessaires sur les étapes de conception de la FFT radix-2^2 à 16 points, 64 points et 256 points et les BPE pour le radix-2^2 et radix-2^3. L'architecture pipeline R4MDC et les processeurs élémentaires constituants, ainsi que le rôle de chaque bloc, seront discutés individuellement et en détaille. L'évolution de l'ensemble des calculs, incluant les différentes étapes, peut donc être mieux visualisée par l'utilisation de graphe de fluence qui fait apparaitre tous les BPE intervenant dans la FFT radix-2^2.

La figure 27 donne à titre d'exemple le diagramme de fluence de la FFT radix-2^2 MDC de type DIT pour N=16.

Nous pourrons bien remarquer que l'architecture pipeline de radix-2^2 MDC à 16 points se compose de deux étages, chaque étage inclut des processeurs élémentaires (BPE). Le premier étage traite quatre trames de quatre données d'entrées liées au premier processeur élémentaire (BPE1), qui est le BPE radix-2^2 suivi par un Commutateur pour rediriger les données qui viennent du premier étage et les mettre dans le bon ordre afin de faire les opérations en papillon au deuxième BPE. Les sorties du Commutateur sont reliées aux entrées des multiplieurs complexes avec les TWF. Le deuxième étage se constitue d'un deuxième processeur élémentaire (BPE2), dont les sorties sont les sorties finales de la FFT pipeline.

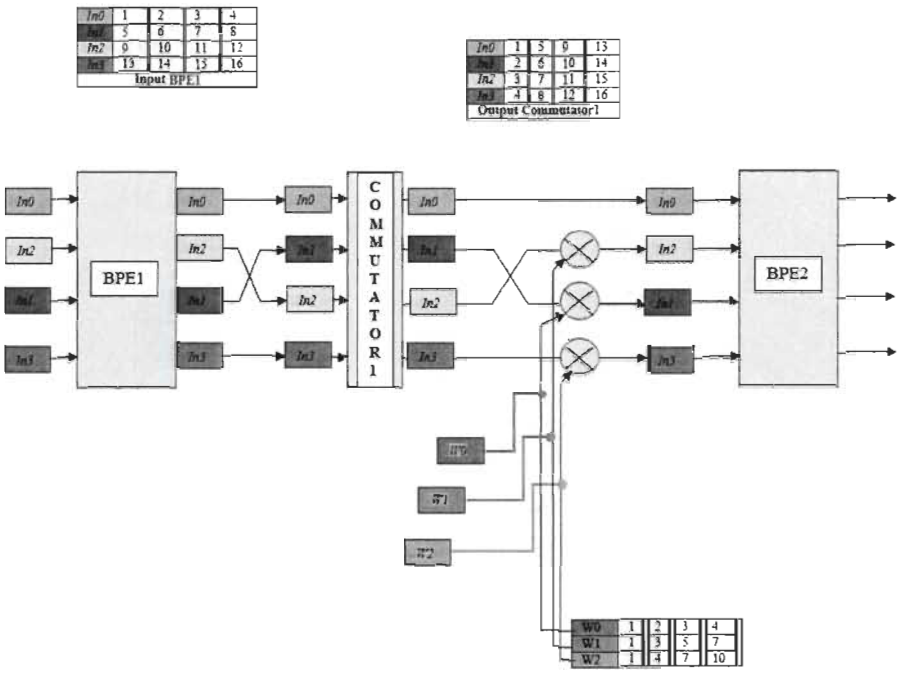


Figure 27 Architecture pipeline radix-2^2 MDC de type DIF à deux étages N=16 [MJ14]

A. BPE radix-2^2 FFT

La plupart des transformations de calcul des FFT se font dans les boucles du BPE. Tout algorithme qui réduit le nombre d'additionneurs et de multiplications dans ces boucles réduira la vitesse de calcul globale. La réduction du calcul est réalisée en ciblant les multiplications triviales qui ont une accélération limitée ou en parallélisant les FFT qui ont une accélération significative sur le temps d'exécution de la FFT. Dans cette section, nous serons limités dans l'élaboration des radix-2^{α} / 4^{β} (les familles radix-2^2 /2^3) proposées pour le processus DIT FFT.

L'utilisation des ressources pourrait également être réduite par un réseau de rétroaction et un réseau de multiplexage où le réseau de rétroaction sert à alimenter la $i^{ème}$ sortie du $j^{ème}$

réseau d'additionneur radix-2 à la $j^{ème}$ entrée du $i^{ème}$ BPE et les multiplexeurs transmettent sélectivement les données d'entrée ou la rétroaction, alternativement, au réseau additionneur radix-2 correspondant [MJ14].

Une illustration de la BPE conventionnelle radix-2^2 ainsi que celle multiplexée est présentée dans les figures 28 et 29 (a). Le diagramme blocs de circuit du réseau d'additionneur radix2 est illustré à la figure 29 (b) qui se compose de deux additionneurs complexes uniquement [MJ14].

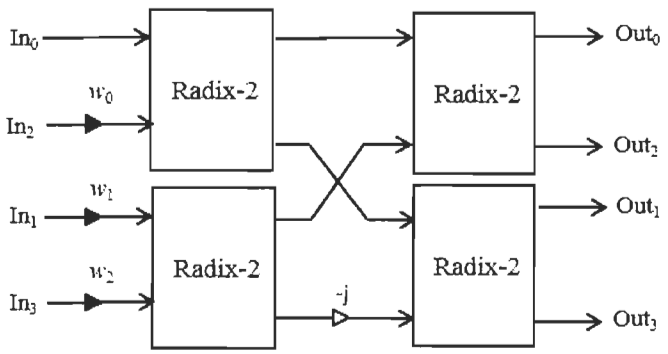


Figure 28 (MDC-R2^2) BPE Conventionnelle de radix-2^2

Avec le front montant du cycle d'horloge, les données d'entrée sont alimentées à l'entrée du BPE présentée sur la figure 29. Pour compléter les opérations du BPE dans un seul cycle d'horloge, les conditions suivantes doivent être satisfaites :

$$T_{CLK} > T_{CM} + 2T_{CA} \qquad (3.1)$$

$$\text{Débit}(\textbf{\textit{Throughput}}) = 4/T_{CLK}$$

où T_{CM} / T_{CA} est le temps nécessaire pour effectuer une multiplication / addition complexe et le schéma de bloc de synchronisation des figures 28 et 29 respectivement sont illustrés sur les figures 30 et 31. [MJ14]

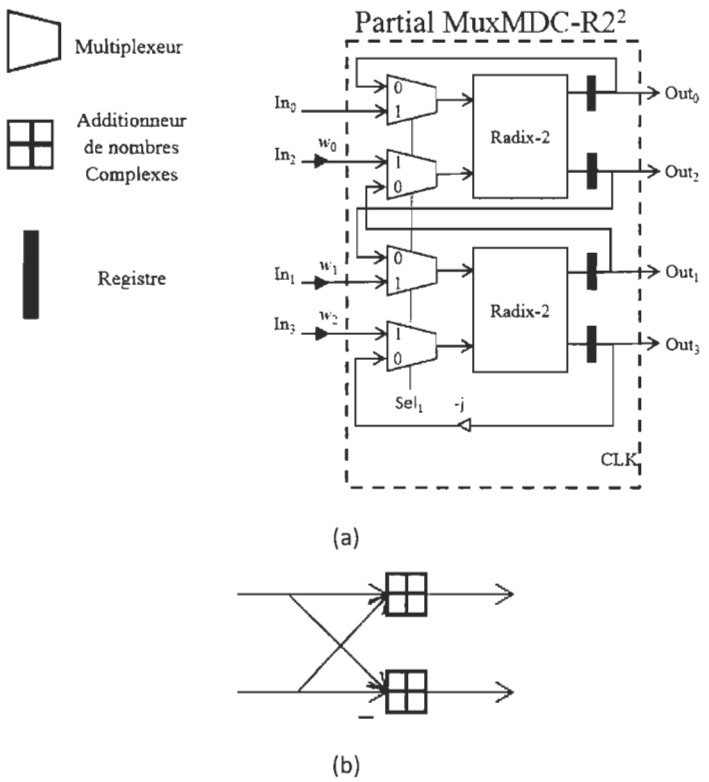


Figure 29 (a) BPE_MUX proposée (b) diagramme de blocs de circuit du réseau d'additionneur complexe radix-2 [MJ14]

Avec le front montant du cycle d'horloge, les données d'entrée sont transmises à l'entrée du système présenté à la figure 29 (a) et avec le front descendant du cycle d'horloge, les données de retour sont transmises à l'entrée du BPE.

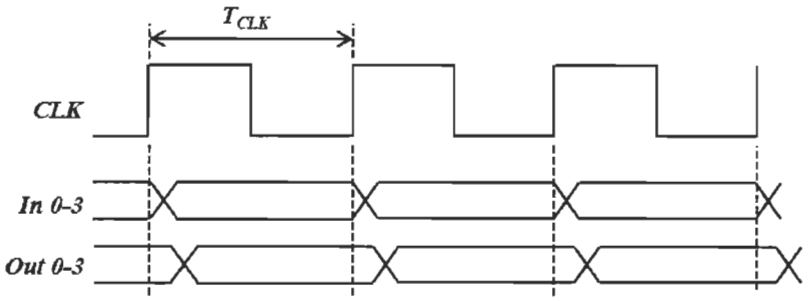


Figure 30 Timing block diagramme de la figure 31[MJ14]

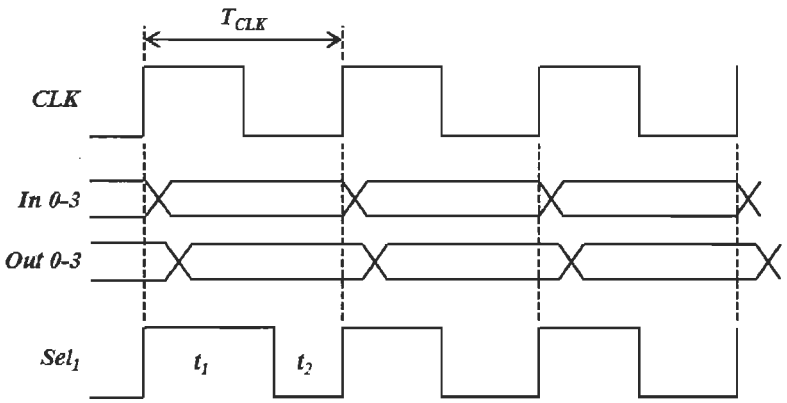


Figure 31 Timing block diagramme de la figure 32[MJ14]

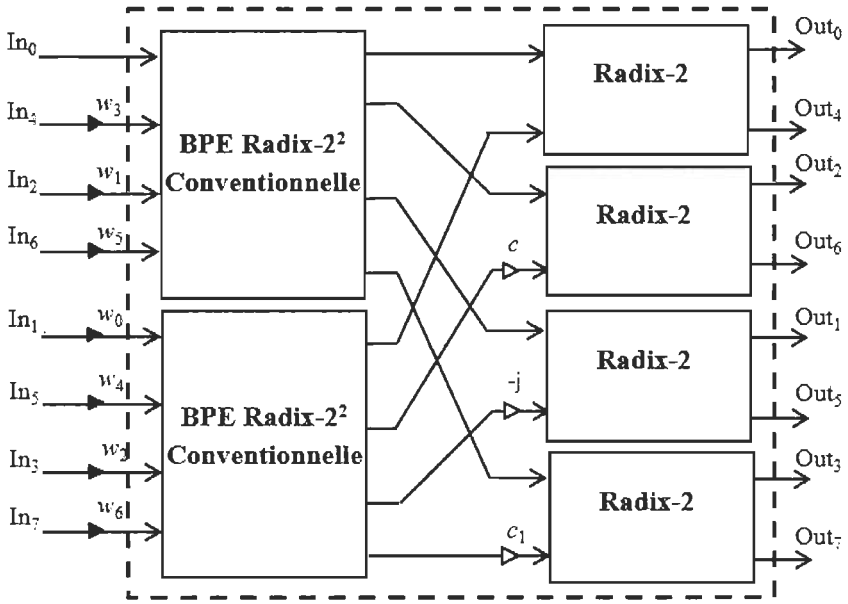


Figure 32 BPE Conventionnelle de radix-2^3 [MJ14]

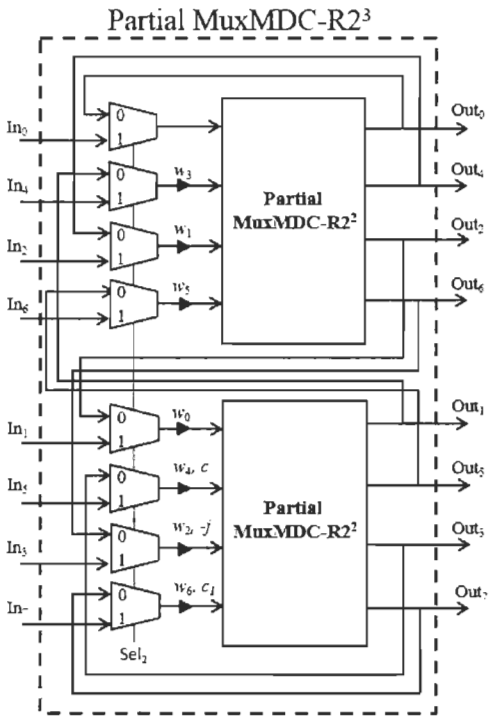


Figure 33 BPE Multiplexée (BPE_MUX) proposée de radix-2^3 [MJ14]

Les figures 32 et 33 représentent une comparaison entre les architectures conventionnelles et multiplexées d'une BPE de radix-2^3, ces architectures seront implémentées et on réalisera une comparaison en termes de fréquence et nombre de Slices occupées.

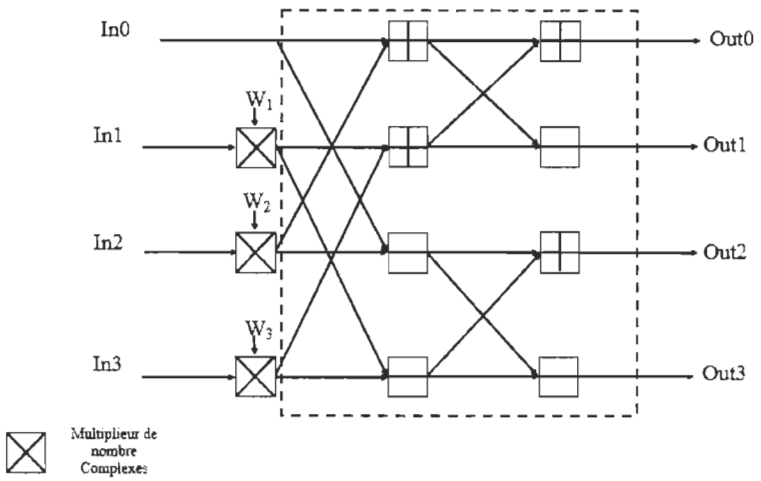


Figure 34 Architecture d'une BPE de radix-4 [HS12]

Selon les figures 32 et 34, on peut déterminer le chemin critique pour une BPE conventionnelle de radix-2^3, en effet, puisque le chemin critique de la BPE conventionnelle de radix-2 présenté dans la figure 29(b) se comporte de juste une unité d'Additionneur, on constate que celui de la BPE conventionnelle de radix-2^3 se compose d'un Multiplieur et trois unités Additionneurs.

Par contre, dans la BPE_MUX de radix-2^3 le chemin critique se compose juste de deux unités Multiplexeurs, une unité Multiplieur et une unité Additionneur, ce chemin est représenté dans la figure 35.

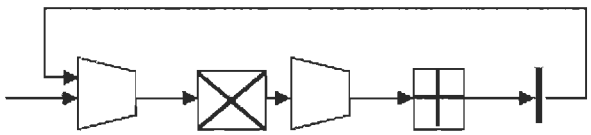


Figure 35 Chemin Critique d'une BPE_MUX de radix-2^3 implémentée

Le tableau 5 présente une comparaison entre le nombre des ressources utilisé par la BPE MUX et conventionnelle, en fait, grâce à sa structure multiplexée, la BPE radix-2^2 proposée enregistre un gain de 56% dans l'utilisation des additionneurs réels, mais elle utilise un nombre de 8 multiplexeurs et 8 registres plus que la structure conventionnelle. La différence apparaisse plus claire lors de la synthèse de la BPE de radix-2^3 qui enregistre un gain de 22% dans l'utilisation des multiplieurs réels et de 54% dans l'utilisation des additionneurs réels, mais perd en contrepartie dans l'utilisation des multiplexeurs et des registres.

Tableau 5 Comparaison des nombres de ressources utilisées par BPE MUX et convontionnelle

	Mult_réel		*Add_réel*		*Mux_réel*	*Registre*
	Nombre	*Gain %*	*Nombre*	*Gain %*		
Conventionnelle R2²	12	0	32	0	0	0
Conventionnelle R2³	36	0	66	0	0	0
MUX R2²	12	0	14	56	8	8
MUX R2³	28	22	30	54	16	16

Le tableau 6 montre qu'en comparant à la référence, l'intégration de la BPE proposée dans le calcul de la FFT a un effet visible sur l'utilisation des ressources comme les additionneurs réels et permet en même temps de gagner en termes de latence et de débit pour les FFT radix-2^3.

Tableau 6 Comparaison de l'utilisation des ressources nécessaires pour le calcul de la FFT

FFT	**Réf. [MG13]** (P=4, radix-2^2 et P=8, radix-2^3)				**Architecture proposée [MJ14]**			
	Multiplieurs de nombres complexes	Additionneurs de nombres complexes	Latence (Cycle)	Débit (É/cycle)	Multiplieurs de nombres complexes	Additionneurs de nombres complexes	Latence (Cycle)	Débit (É/cycle)
Radix-2^2	$3(\log_4 N-1)$	$16(\log_4 N)$	$N/4$	4	$3(\log_4 N-1)$	$4(\log_4 N)$	$N/4$	4
Radix-2^3	$6(\log_4 N-7)$	$8(\log_4 N)$	$N/8$	8	$7(\log_8 N-1)$	$8(\log_4 N)$	$N/8$	8

3.1.3.A.1 Structure implémentée

La structure implémentée du Partial MuxMDC-$R2^2$ est présentée à la figure 36. L'ajout des registres dans ce bloc a permis d'une part de synchroniser le fonctionnement des différentes

unités qui le constituent et d'autre part d'atteindre une fréquence élevée et meilleure que la référence.

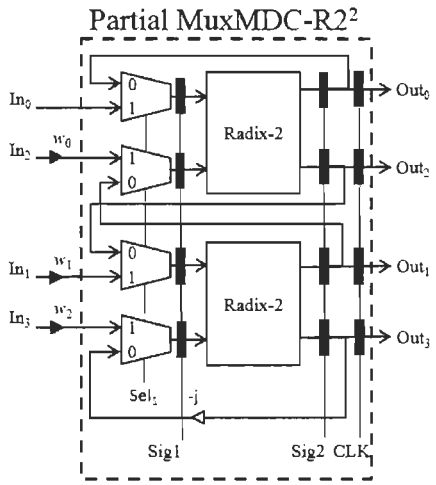


Figure 36 Représentation des signaux de contrôle dans la structure Partial MuxMDC-R2²

Différemment à la figure 31, et pour faciliter le travail et concentrer dans un premier lieu sur la synchronisation et le fonctionnement des différents blocs dans la structure BPE, on a opté dans notre implémentation à utiliser un rapport cyclique égale à 50% pour le signal Sel_1, donc puisque $t_1=t_2=T_{CLK}/2$, on peut utiliser l'horloge CLK comme étant le signal qui contrôle les multiplexeurs.

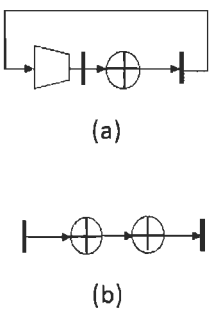


Figure 37 Comparaison entre les Chemins critiques des BPE implémentés a) BPE_MUX (figure 29a) et b) BPE conventionnelle (figure 28).

La figure 37 montre une comparaison entre les chemins critiques des deux architectures BPE_MUX et BPE conventionnelles implémentés, on remarque la présence d'une unité arithmétique de plus pour la BPE conventionnelle ce qui explique après dans les résultats sa faible fréquence par rapport à la BPE_MUX.

Dans la conception du Partial MuxMDC-R2^2 on a utilisé l'horloge (CLK) et deux signaux de contrôle (Sig_1 et Sig_2) ; l'horloge CLK (qui est aussi l'horloge principale) de période T_{CLK} est utilisée pour contrôler les Multiplexeurs de la BPE, en effet, elle représente aussi l'horloge principale qui contrôle le reste des blocs qui forment l'architecture FFT (Commutateur et Multiplieur de nombres complexes).

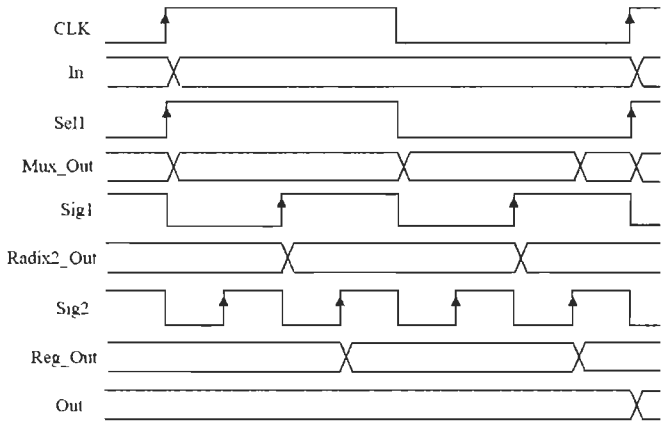


Figure 38 Data flow diagramme de la structure Partial MuxMDC-R2^2 implémentée

Les deux signaux de contrôle Sig_1 et Sig_2 sont seulement utilisés localement pour synchroniser les blocs qui constituent le Partial MuxMDC-R2^2, le premier signal de contrôle Sig_1 de période $T_{CLK}/2$ est utilisée pour assurer la synchronisation avec les unités arithmétiques dans la BPE, et enfin le deuxième signal de contrôle Sig_2 de période T_{CLK} /4 est utilisée pour contrôler les registres de sortie du Partial MuxMDC-R2^2. Le schéma de bloc de synchronisation des différents signaux dans le BPE est illustré sur la figure38. Les signaux

Mux_Out, Radix2_Out, et Reg_Out représentent respectivement, les signaux se sortie de multiplexeurs, des BPE radix-2 et des registres contrôlés par Sig2.

Comme le montre la figure 39, on peut voir les différents signaux utiliser pour synchroniser le fonctionnement entre les blocs du Partial MuxMDC-R2^2. La synchronisation entre Sel1, Sig_1 et Sig_2 permet d'obtenir des groupes de résultats après un seul cycle de latence.

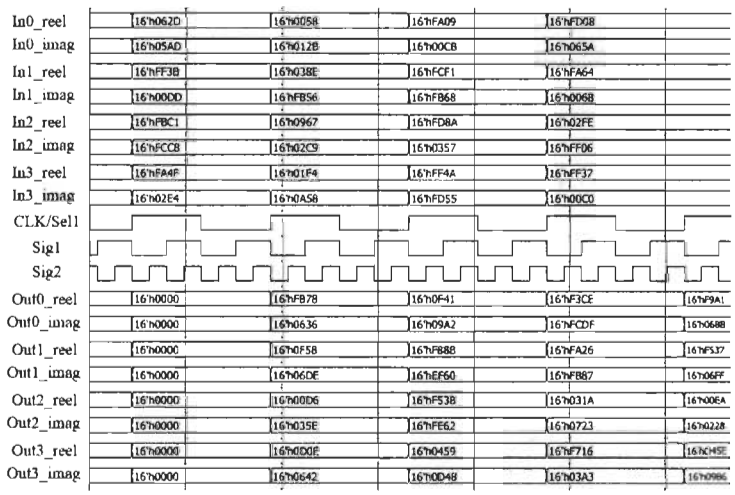


Figure 39 Représentation de la latence du Partial MuxMDC-R22 implémentée

B. Multiplieur de nombres complexes

Le multiplieur de nombres complexes et un module qui peut être trouvé dans la majorité des algorithmes de traitement de signal tel que les filtres adaptatifs (LMS, Kalman ...), la FFT, les algorithmes d'égalisation des canaux etc., [MJ04], [MS05], [CP06].

Le multiplieur de nombres complexes comme un des éléments les plus demandant en ressources arithmétiques dans l'architecture des FFT. Il doit donc être conçu d'une façon optimisé pour atteindre le maximum de performance avec les fréquences les plus élevées

tout en minimisant les ressources utilisées. La multiplication de deux nombres complexes peut être représentée de la façon suivante :

$$(A+jB)(C+jD)=(AC-BD)+j(AD+BC)=R_e+jI_{mag} \quad (3.3)$$

Où : Re=AC-BD est la partie réelle de la multiplication et

$$R_e= AC-BD=(A-B)D+A(C-D)$$

$$I_{mag}= AD+BC=(A-B)D+B(C+D) \quad (3.4)$$

I_{mag}=AD+BC est la partie imaginaire de la multiplication de nombres complexes.

Le multiplieur de nombres complexes fait intervenir quatre multiplications réelles (AC, BD, AD, BC), une addition et une soustraction. Les nombres A, B, C et D sont considérés des nombres binaires en complément à deux, signés (premier bit représente le bit de signe).

Une autre manière de décrire une multiplication de nombres complexes est la suivante :

Avec cette méthode de factorisation, le nombre de multiplications réelles diminue de quatre à trois, on passe d'une addition et une soustraction à trois additions et deux soustractions.

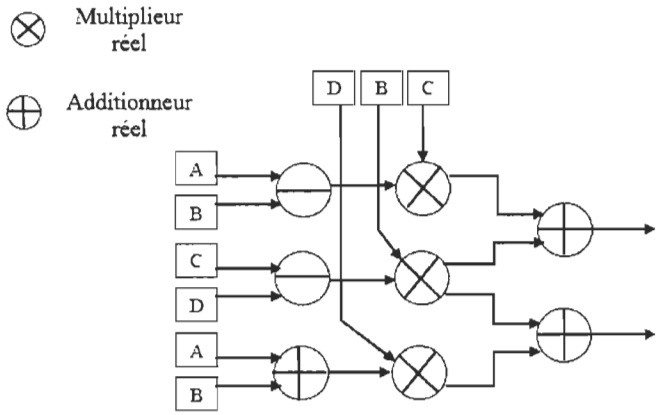


Figure 40 Structure interne de multiplieur de nombres complexes non pipeliné optimisé

Puisque le multiplieur occupe une surface silicium trois fois plus grand que celle d'un additionneur [MJ09], cette méthode de factorisation, est donc très bénéfique puisqu'elle rend le multiplieur complexe plus rapide en prenant moins d'espaces sur le circuit électrique.

Les deux figures 40 et 41 décrivent respectivement les schémas du multiplieur de nombres complexes optimisé non pipeliné (figure 40) et pipeliné (figure 41), le pipeline dans ce cas est utilisé pour améliorer la performance de l'architecture FFT au niveau de la fréquence.

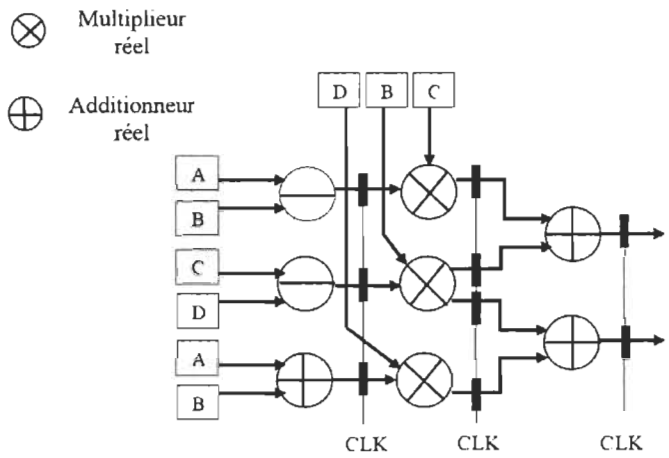


Figure 41 Structure interne de multiplieur de nombres complexes pipeliné optimisé

3.1.3.B.1 Structure implémentée

Pour augmenter la fréquence de fonctionnement de ce module, nous avons introduit des registres de pipeline, l'introduction du pipeline a réussi d'atteindre la fréquence désirée, mais d'une autre cote, on perd dans la latence du Multiplieur de nombres complexes.

La figure 42 représente la structure du multiplieur de nombres complexes qu'on a implémenté et synthétisé utilisant 4 multiplieurs réels et seulement deux additionneurs réels,

on a utilisé cette représentation dans notre projet pour répondre aux mêmes conditions que la référence.

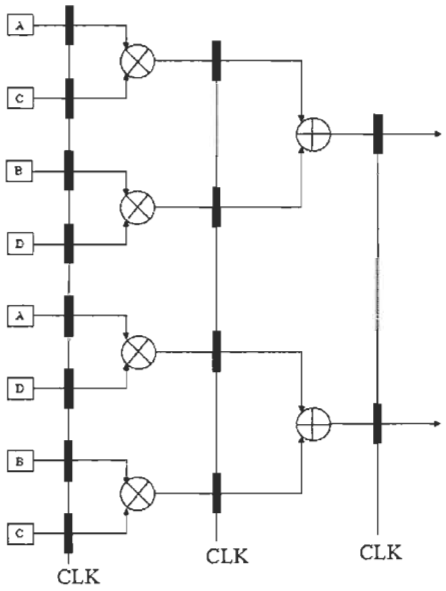


Figure 42 Structure implémentée du Multiplieur de nombres complexes utilisant 4 multiplieurs réels

C. Commutateur

Le commutateur est un bloc indispensable et non trivial dans le calcul de la FFT radix-4 MDC. Il est utilisé entre les étages de radix-4 papillon dans l'architecture pipeline. Il stocke temporairement une partie des données durant le cycle de calcul FFT afin de les réorganiser de nouveau. Le bloc commutateur est composé des multiplexeurs, de simples éléments de délais, et un compteur à deux bits pour l'activation des multiplexeurs. Le nombre des éléments de délais dépend exactement de la longueur de la FFT. La structure interne de commutateur et donnée par la figure 43 [JY03].

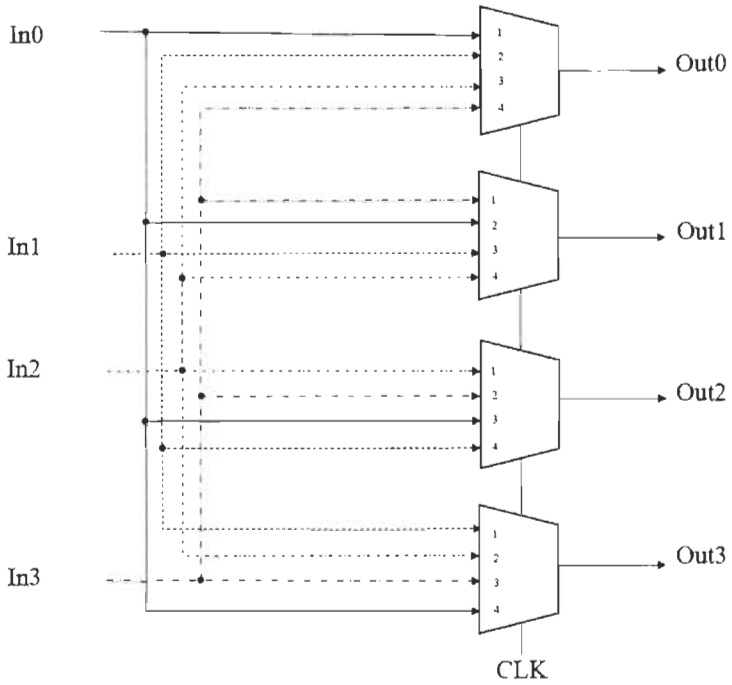


Figure 43 Structure interne de commutateur

La figure 44 présente une distribution des registres de délais qui sont liés au module du commutateur, on a choisi comme exemple le commutateur d'une FFT de 16 points, le nombre de registres de délais dépend de la grandeur de la FFT ainsi que l'étage auquel le commutateur appartient, cette représentation peut exister dans des architectures de FFT de toutes les grandeurs, mais dans les derniers étages.

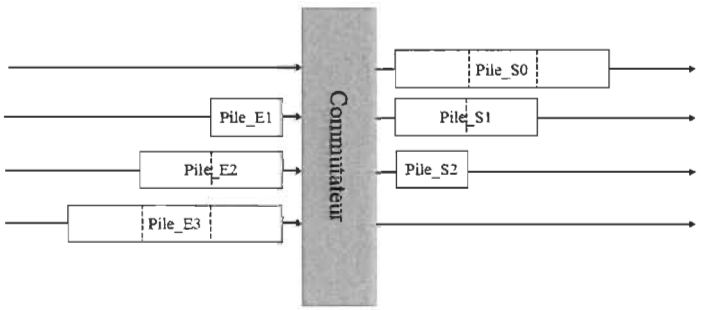


Figure 44 Représentation des délais liés au module du Commutateur pour une FFT de 16 points

Le tableau 7 décrit la distribution des registres pour les piles d'entrées et sorties dans les commutateurs des différents étages. En effet, c'est seul le nombre des registres qui change dans les commutateurs d'un étage à un autre, les transformations des données à l'intérieur sont toujours les mêmes. La structure de commutateur se constitue de quatre multiplexeurs en parallèle où chaque multiplexeur a quatre entrées et une sortie.

Tableau 7 Distribution des registres dans les Commutateurs pour la FFT de radix-2^2

Taille FFT	*Étage 1*								*Étage 2*								*Étage 3*							
	Pile Entrée				*Pile Sortie*				*Pile Entrée*				*Pile Sortie*				*Pile Entrée*				*Pile Sortie*			
	0	*1*	*2*	*3*	*0*	*1*	*2*	*3*	*0*	*1*	*2*	*3*	*0*	*1*	*2*	*3*	*0*	*1*	*2*	*3*	*0*	*1*	*2*	*3*
16	0	1	2	3	3	2	1	0																
64	0	4	8	12	12	8	4	0	0	1	2	3	3	2	1	0								
256	0	16	32	48	48	32	16	0	0	4	8	12	12	8	4	0	0	1	2	3	3	2	1	0

Les multiplexeurs sont aussi connectés par un compteur à deux bits pour l'activation de chaque multiplexeur. Un autre schéma représentatif des différentes transformations de commutateur est donné dans la figure 45 [SE84] :

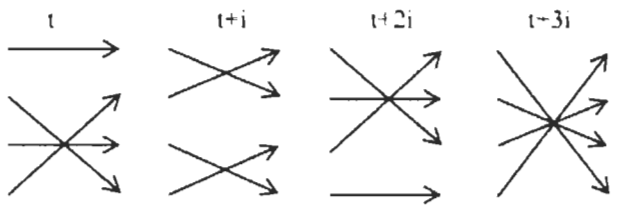


Figure 45 Les transformations de commutateur

La figure 46 représente l'architecture d'une FFT à 64 points avec l'ordre des données utilisé.

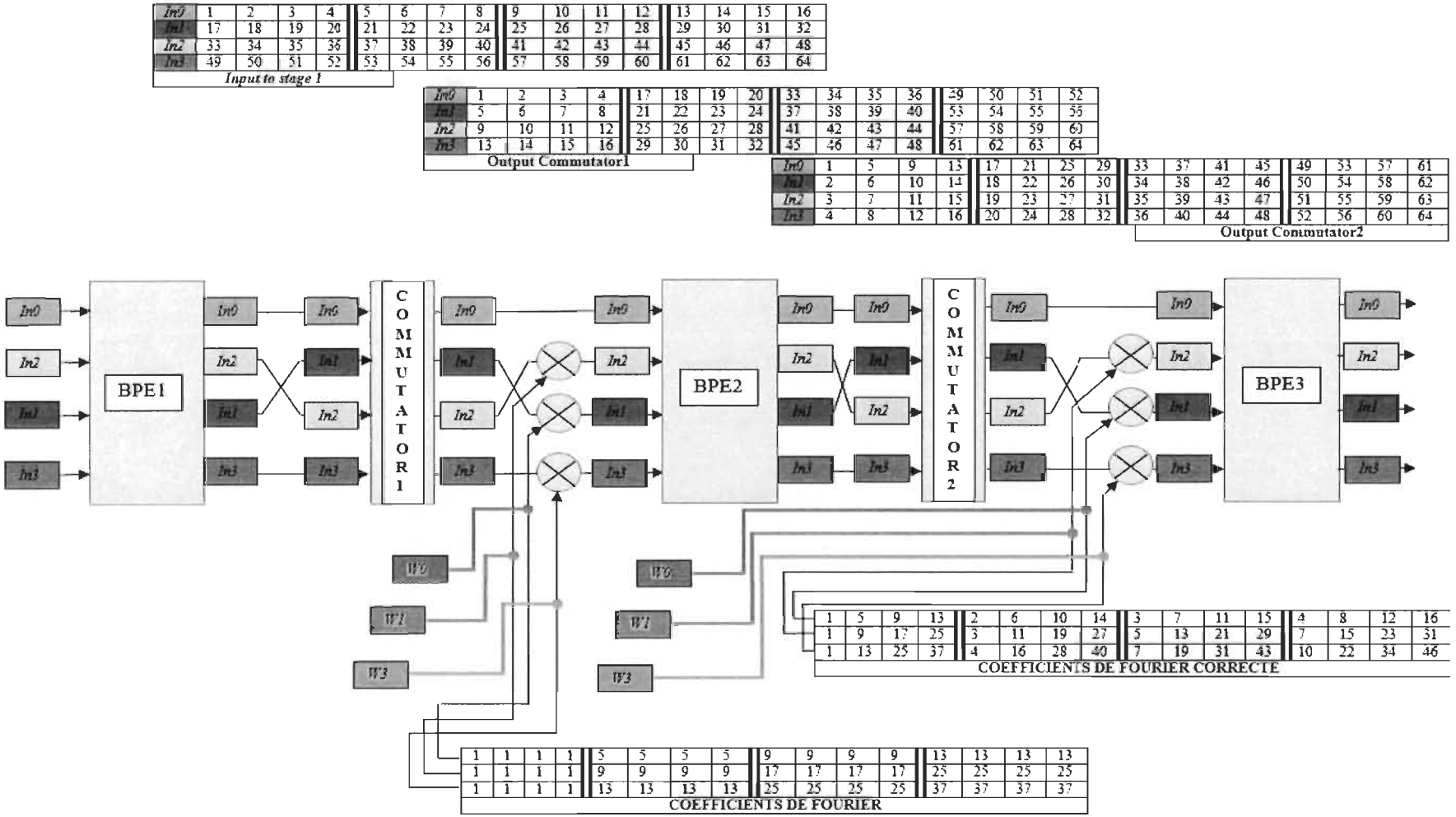


Figure 46 Architecture pipeline radix-4 MDC de type DIF sur 3 étages N=64 points

3.2 Synthèse des résultats d'implémentation

Dans cette partie on présente les résultats d'implémentation obtenus après l'implémentation de la nouvelle architecture de la FFT de radix-2^2. Les résultats sont exprimés dans un tableau de récapitulation. Nos résultats sont comparés avec les résultats obtenus par la dernière étude présentée par [MG13]. On a donc utilisé la même famille de FPGA et la même plateforme de développement.

L'évaluation de l'exécution de la FFT se base sur le compromis surface silicium/vitesse. Cependant plusieurs paramètres rentrent en jeu afin de trouver le meilleur compromis surface silicium/vitesse. En parlant de surface silicium, on parle des ressources matérielles nécessaires pour implémenter la FFT. Pour la vitesse, on parle de fréquence d'exécution de la FFT, de débit et de latence.

3.2.1 Critères d'évaluation

Il existe plusieurs critères d'évaluation de l'exécution de la FFT. Le premier critère d'évaluation est l'erreur qu'on obtient en comparant les résultats de la FFT en virgule fixe avec ceux de Matlab, la figure 47 montre la distribution de l'erreur réalisée pour plusieurs entrées aléatoires pour des FFT de 16 points utilisant la nouvelle architecture de BPE_MUX. Pour obtenir cette figure, on a réalisé 10 essais d'entrées aléatoires et on affiche l'erreur maximale des 10 résultats d'essais pour chaque point de la sortie de la FFT. Noter que la figure présente 32 points, soit le couplet (réel, imaginaire) pour chaque point. Pour mieux prouver le bon fonctionnement de la FFT on a utilisé les mêmes entrées aléatoires pour des FFT de 16 points utilisant des BPE conventionnelles, en comparant avec la figure 48, on montre qu'on a atteint la même précision des résultats.

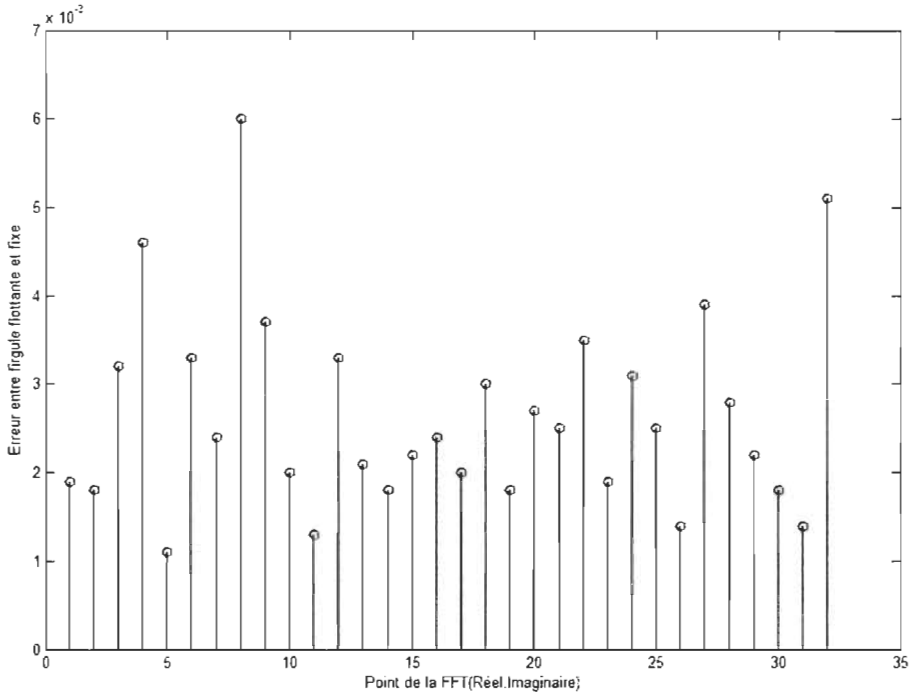


Figure 47 Erreur entre virgule fixe et flottante d'une FFT de taille 16 utilisant BPE_MUX

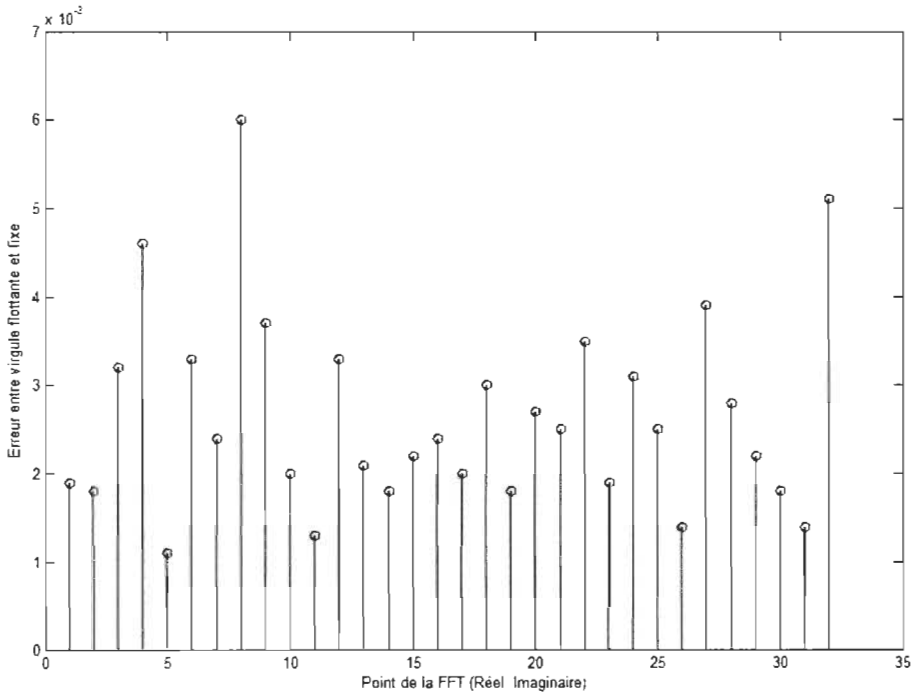


Figure 48 Erreur entre virgule fixe et flottante d'une FFT de taille 16 utilisant BPE conventionnelle

Un autre critère important est la latence qui correspond au délai entre la réception des données et la sortie des résultats. Ce qui oblige les nouveaux systèmes de réduire au minimum les délais de latence. Un autre critère aussi important est le coût d'implémentation. La meilleure méthode d'implémentation pour une exécution d'une FFT complète, doit avoir une fréquence d'exécution rapide, tout en utilisant moins de ressources matérielles sur le FPGA.

Dans notre projet, on compare nos résultats avec la méthode d'implémentation utilisée par [MG13] et cela sur plusieurs paramètres :

- Nombre de multiplieurs câblés ou DSP48 : chaque FPGA possède des multiplieurs câblés ou des blocs appelés DSP48. Ces derniers sont des blocs développés par Xilinx pour les nouvelles versions de FPGA et sont utilisés pour maximiser la vitesse d'exécution de la multiplication. On remarque ainsi une augmentation considérable dans la fréquence d'exécution de la FFT.
- Fréquence de fonctionnement : paramètres importants pour la comparaison des performances de la méthode d'implémentation. La fréquence est calculée quant à elle en (MHz). Plus la fréquence de fonctionnement est élevée plus l'architecture est rapide.
- Délai : Le délai est calculé en nanoseconde (*ns*) et représente le temps de propagation des signaux à travers l'architecture, il constitue la limite de fonctionnement du circuit.
- Nombre de cycles : s'appelle aussi la latence représente le nombre de cycle d'horloge nécessaire pour exécuter une FFT complète.

- Débit : Nombre d'échantillons de sortie de la structure en pipeline par le temps d'un cycle d'horloge.
- Coût de la FFT : représente le nombre de slices utilisés par la FFT multiplié par le temps d'exécution de la FFT en (µs). Le temps d'exécution de la FFT représente à son tour le nombre de cycles divisé par la fréquence de fonctionnement. Plus ce coût est petit mieux c'est.
- Fonction de performance (MS/s/slice) : dans les FPGA de Xilinx, les slices sont des cellules logiques élémentaires, composées de 2 LUT et 2 bascules. Les LUT (*Look Up Table*) sont des générateurs de fonctions. Le nombre et l'architecture des LUT et des bascules dans une slice varient d'une famille FPGA à une autre. La fonction dévaluation de la performance exprimée en MS/s/slice représente le débit en Méga Échantillon par seconde par slice, les ressources. Plus la fonction performance est (MS/s/Slice) grande mieux c'est.

3.2.2 Résultats d'implémentation

Dans cette section, on présente les résultats de simulation de l'implémentation de l'architecture pipeline de la FFT radix-2^2 MDC à 16 points 64 points et 256 points décrite précédemment sur une puce FPGA choisie, ainsi que les performances de l'algorithme FFT en termes de surface et la fréquence maximale de fonctionnement.

On a aussi réalisé une étude comparative de la performance des BPE MUX et conventionnelle de radix-2^3 en termes de fréquence et de nombre de Slices occupées par chaque structure, cette comparaison est présentée par le tableau 14.

L'algorithme FFT est modélisé en utilisant le langage VHDL et mis en œuvre sur FPGA du type Virtext-5 (xc5vsx240t-2ff1738) et Virtex-7 (xc7vx330t-ffg1157) de Xilinx.

L'environnement de travail nommé ISE pour « *Integrated synthesis environment* » ou environnement de synthèse intégré est utilisé pour synthétiser et simuler le code VHDL conçu.

Les tableaux suivants récapitulent l'utilisation des ressources matérielles lors de l'implémentation des processeurs FFT radix-2^2 pour N=16, N=64, N=256, pour des largeurs binaires des données d'entrées et de coefficient égal à 16 bits. Les résultats de performance de ces architectures sont obtenus en utilisant trois stratégies essentielles de synthèse :

i. *Timing Performance*, cherche à maximiser la cadence de fonctionnement.

ii. *Area Reduction*, cherche à réduire les ressources.

iii. *Balanced*, permet le meilleur compromis ressource et cadence de fonctionnement.

Les tableaux de 8 à 10 représentent une comparaison des résultats d'implémentation des BPEs MUX et conventionnelles pour des FFTs de radix-2^2 à N=16, 64 et 256, cette synthèse est faite en utilisant les stratégies '*Timing Performance*', ' *Area Reduction*' et '*Balanced*'.

La même comparaison des résultats est présentée dans les tableaux de 11 à 13, mais en utilisant la famille Virtex-7 de FPGA. Les tableaux 14 et 15 montrent l'effet sur la fréquence et le nombre de Slices de l'introduction des multiplieurs de nombres complexes conventionnels et optimisés avec l'étude des cas pipelines et non pipelines. Dans le tableau 17 on a présenté les résultats d'implémentation des structures FFT qu'on a utilisé comme référence, et on a fini par le tableau 18 dans lequel on fait une synthèse de l'évaluation des performances de nos architectures implémentées sur Virtex-5 et Virtex-7 en les comparants avec la référence.

*Tableau 8 Comparaison d'implémentation sur **Virtex-5** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Timing Performance)*

Taille FFT	Taille données et TWF	Nbre de Slices	DSP48E	Délais (ns)	Fréquence (Mhz)	Cycles	Latence (ns)	Temps de transformée		Débit (MS/s)	Cout FFT	Performance (MS/s/Slice)
								Cycles	Temps (ns)			
16	16	498	12	2.95	338	11	30	4	10	1352	1469	2.7
16	16	388	12	1.97	506	9	17	4	8	2028	765	5.22
64	16	746	24	3.8	339	30	90	16	40	1356	2835	1.8
64	16	782	24	1.97	506	26	50	16	30	2028	1540	2.6
256	16	1358	36	3.8	339	85	250	64	190	2567	5160	0.99
256	16	1421	36	1.97	506	79	150	64	120	2024	2800	1.42

BPE Conventionnelle

BPE MUX

*Tableau 9 Comparaison d'implémentation sur **Virtex-5** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Area Reduction)*

Taille FFT	Taille données et TWF	Nbre de Slices	DSP48E	Délais (ns)	Fréquence (Mhz)	Cycles	Latence (ns)	Temps de transformée		Débit (MS/s)	Cout FFT	Performance (MS/s/Slice)
								Cycles	Temps (ns)			
16	16	249	12	3.8	263	11	42	4	15	1052	946	4.2
16	16	327	12	2.2	440	9	20	4	9	1760	720	5.38
64	16	485	24	3.8	263	30	114	16	60	1052	1843	2.17
64	16	599	24	2.2	440	26	60	16	36	1760	1318	2.94
256	16	743	36	3.8	263	85	320	64	240	1052	2823	1.4
256	16	913	36	2.2	440	79	180	64	140	1760	2009	1.9

BPE Conventionnelle

BPE MUX

*Tableau 10 Comparaison d'implémentation sur **Virtex-5** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Balanced)*

Taille FFT	*Taille données et TWF*	*Nbre de Slices*	*DSP48E*	*Délais (ns)*	*Fréquence (Mhz)*	*Cycles*	*Latence (ns)*	*Temps de transformée*		*Débit (MS/s)*	*Cout FFT*	*Performance (MS/s/Slice)*
								Cycles	*Temps (ns)*			
16	16	339	12	3.8	263	11	42	4	15	1052	1288	3.1
16	16	383	12	1.97	506	9	17	4	8	2028	754	5.29
64	16	663	24	3.8	263	30	114	16	60	1052	2520	1.6
64	16	749	24	1.97	507	26	50	16	30	2028	1475	2.7
256	16	1020	36	3.8	263	85	320	64	240	1052	3876	1.03
256	16	1150	36	1.97	507	79	150	64	120	2028	2266	1.76

BPE Conventionnelle
BPE MUX

*Tableau 11 Comparaison d'implémentation sur **Virtex-7** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Timing Performance)*

Taille FFT	*Taille données et TWF*	*Nbre de Slices*	*DSP48E*	*Délais (ns)*	*Fréquence (Mhz)*	*Cycles*	*Latence (µs)*	*Temps de transformée*		*Débit (MS/s)*	*Cout FFT*	*Performance (MS/s/Slice)*
								Cycles	*Temps (ns)*			
16	16	291	12	2.16	422	11	20	4	8	1868	629	6.42
16	16	319	12	1.48	670	9	10	4	6	2692	473	8.4
64	16	479	24	1.48	422	30	40	16	23	2692	709	5.6
64	16	561	24	1.48	671	26	38	16	23	2692	831	4.8
256	16	738	36	1.48	420	85	120	64	95	2692	1093	3.6
256	16	869	36	1.48	670	79	110	64	90	2692	1287	3

BPE Conventionnelle
BPE MUX

*Tableau 12 Comparaison d'implémentation sur **Virtex-7** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Area Reduction)*

Taille FFT	Taille données et TWF	Nbre de Slices	DSP48E	Délais (ns)	Fréquence (Mhz)	Cycles	Latence (ns)	Temps de transformée		Débit (MS/s)	Cout FFT	Performance (MS/s/Slice)
								Cycles	Temps (ns)			
16	16	221	12	2.76	362	11	30	4	11	1448	610	6.55
16	16	303	12	1.49	671	9	13	4	6	2684	452	8.85
64	16	371	24	2.77	360	30	80	16	40	1440	1027	3.88
64	16	503	24	1.49	671	26	38	16	20	2684	780	5.33
256	16	578	36	2.77	360	85	230	64	170	1440	1601	2.5
256	16	793	36	1.49	671	79	110	64	90	2684	1182	3.38

BPE Conventionnelle
BPE MUX

*Tableau 13 Comparaison d'implémentation sur **Virtex-7** des FFT radix-2^2 utilisant la BPE MUX et la BPE Conventionnelle (Stratégie : Balanced)*

Taille FFT	Taille données et TWF	Nbre de Slices	DSP48E	Délais (ns)	Fréquence (Mhz)	Cycles	Latence (ns)	Temps de transformée		Débit (MS/s)	Cout FFT	Performance (MS/s/Slice)
								Cycles	Temps (ns)			
16	16	250	12	2.76	362	11	3	4	11	1448	690	5.8
16	16	325	12	1.48	673	9	1	4	6	2692	481	8.28
64	16	461	24	2.77	360	30	80	16	40	1440	1277	3.12
64	16	596	24	1.48	673	26	38	16	20	2692	883	4.51
256	16	719	36	2.77	360	85	230	64	170	1440	1992	2
256	16	852	36	1.48	673	79	110	64	90	2692	1261	3.16

BPE Conventionnelle
BPE MUX

Tableau 14 Comparaison des performances de l'implémentation de la FFT utilisant des Multiplieurs de nombres complexes ***conventionnels*** *pipelinés et non pipelinés*

Taille FFT	***DSP48E***	***Multiplieurs pipelinés***		***Multiplieurs non pipelinés***	
		Slices	*Fréquence(Mhz)*	*Slices*	*Fréquence (Mhz)*
16	*12*	*383*	*506*	*307*	*221*
64	*24*	*749*	*507*	*341*	*274*
256	*36*	*1150*	*507*	*1091*	*221*

Tableau 15 Comparaison des performances de l'implémentation de la FFT utilisant des Multiplieurs de nombres complexes ***optimisés*** *pipelinés et non pipelinés*

Taille FFT	***DSP48E***	***Multiplieurs pipelinés***		***Multiplieurs non pipelinés***	
		Slices	*Fréquence(Mhz)*	*Slices*	*Fréquence (Mhz)*
16	*9*	*450*	*345*	*392*	*291*
64	*18*	*345*	*297*	*341*	*274*
256	*27*	*1294*	*291*	*1159*	*225*

Tableau 16 Comparaison de résultats d'implémentation sur Virtex-5 entre BPE MUX et conventionnelle (Avec Multiplieurs) de radix-2^{3}

BPE	***Slices***	***Fréquence (Mhz)***	***DSP48E***	***Réduction (en Slices)***	***Gain (en DSP48E)***	***Gain (en Fréquence)***
Conventionnelle	430	263	36	0%	0%	0%
MUX	597	506	28	+28%	23%	193%

Tableau 17 Résultats d'implémentation des FFT radix2^2 de la référemce [MG13]

Taille FFT	*Slice*	*DSP48E*	*Latence (µs)*	*Fréquence(Mhz)*	*Débit (MS/s)*	*MS/s/Slice*
16	389	12	0.026	458	1831	4.7
64	695	24	0.081	384	1536	2.21
256	1024	36	0.221	389	1554	1.51

Tableau 18 Synthèse de l'évaluation des performances du FFT radix-2^2

	Architecture de la référence [MG13]				*Architecture proposée [MJ14]*											
Taille FFT	*R2²MDC (Virtex5)*				*Partial MuxMDC-R2² (Virtex5)*						*Partial MuxMDC-R2² (Virtex7)*					
	Slice	*Fréq (Mhz)*	*Débit (MS/s)*	*(MS/s/ Slice)*	*Slice*	*Fréq (Mhz)*	*Gain en Fréq (%)*	*Débit (MS/s)*	*MS/s/ Slice*	*Gain (%)*	*Slice*	*Fréq (Mhz)*	*Gain en Fréq (%)*	*Débit (MS/s)*	*MS/s/ Slice*	*Gain (%)*
16	389	458	1831	4.7	383	506	111	2028	5.29	112	325	673	147	2692	8.28	176
64	695	384	1536	2.21	749	507	132	2028	2.7	122	596	673	175	2692	4.51	204
256	1024	389	1554	1.51	1150	506	130	2028	1.76	116	852	673	173	2692	3.16	209

3.2.3 Synthèse et analyse des Résultats

En regardant les tableaux 8, 9, et 10 on peut constater que l'utilisation de la nouvelle architecture BPE permet d'avoir meilleure performance (MS/s/Slice) par rapport à la BPE conventionnelle, on gagne aussi en termes de nombre de cycles nécessaires pour avoir les résultats finals de la FFT ainsi que le nombre de slices occupées. On peut aussi constater qu'en utilisant la BPE_MUX on peut atteindre une fréquence d'horloge deux fois plus rapide qu'en utilisant la BPE conventionnelle. La rapidité de la nouvelle architecture du BPE est grâce à sa structure multiplexée qui permet d'utiliser moins de block additionneurs, l'utilisation de cette structure permet d'avoir le résultat dans un seul cycle d'horloge ce qui permet de gagner en termes de latence par rapport à la structure conventionnelle de BPE.

L'utilisation de la technologie Virtex-7 permet de plus en plus améliorer la performance (MS/s/Slice), en d'autre terme, on utilise mois de Slices avec une fréquence de traitement des données plu grande. Ces résultats sont montrés dans les tableaux 11, 12, et 13.

En regardant les résultats de différentes stratégies de synthèse on peut remarquer que la différence majeure est dans le nombre de Slices qui diminue vers un minimum dans la stratégie '*Area Reduction*' et augmente vers un maximum pour la stratégie '*Timing Performance*', en fait le meilleur compromis entre le nombre des Slices et la fréquence peut être atteinte en utilisant la stratégie '*Balanced*' qui donne toujours une fréquence maximale avec une petite augmentation dans l'utilisation des Slices.

Dans les tableaux 14 et 15 on a essayé de justifier le choix de l'utilisation d'un multiplieur de nombres complexes conventionnel et pipeliné, selon les résultats trouvés on remarque que l'introduction du pipeline nous permet d'atteindre des fréquences élevés mais avec une petite perte en termes de Slices utilisées. La même observation peut être appliquée si on compare entre l'utilisation d'un multiplieur de nombres complexes conventionnel ou optimisé.

Dans le tableau 16 on a présenté les résultats de comparaison entre l'implémentation d'un BPE MUX de radix-2^3 et un BPE conventionnel de radix-2^3, il est clair que l'utilisation de la nouvelle architecture de BPE permet d'atteindre des fréquences élevées tout en gagnant en termes de nombre de Slices.

En comparant nos résultats obtenus pour les différentes stratégies de synthèse par rapport à ceux de la référence présentés dans le tableau 17, on peut remarquer qu'on est supérieure en termes de fréquence et de nombre de Slices ce qui permet d'avoir une meilleure performance (MS/s/Slice). Cette observation peut être aussi déduite à partir du tableau 16 où on a choisi

la stratégie 'Balanced' pour réaliser notre comparaison. On remarque qu'on atteint un gain de 130% en fréquence par rapport à la référence pour N=256, ce qui nous permet d'améliorer notre performance MS/s/Slice de 116%. Le gain en fréquence ou en performance devient plus grand quand on utilise la technologie de Virtex-7, en effet, pour N=256, on enregistre un gain de 173% en termes de fréquence et de 209% en termes de performance. On peut aussi remarquer que malgré l'introduction du pipeline dans notre architecture implémentée et donc utiliser plus de registres, on utilise un nombre de Slices faible et parfois moins que la référence qui utilise une architecture non pipelinée dans un effort pour diminuer l'utilisation des Slices. On maintien aussi une fréquence constante et meilleure, en effet, on remarque que la variation de la longueur de la FFT a un effet remarquable sur la fréquence atteinte par l'article de référence et ainsi sur le débit de la structure générale de la FFT, ce qui n'est pas le cas dans notre architecture proposée qui prouve que la fréquence et débit atteints restent invariables quel que soit N=16, 64 ou 256.

Les bloc FFT R4MDC avec une longueur limitée à 16 points ont été simulés et synthétisés en utilisant le logiciel Xilinx Design ISE. Les blocs technologiques RTL ainsi du circuit complet généré par Xilinx sont représentés sans les figures suivantes. Nous distinguerons toutes les entrées et sorties globales du système complet.

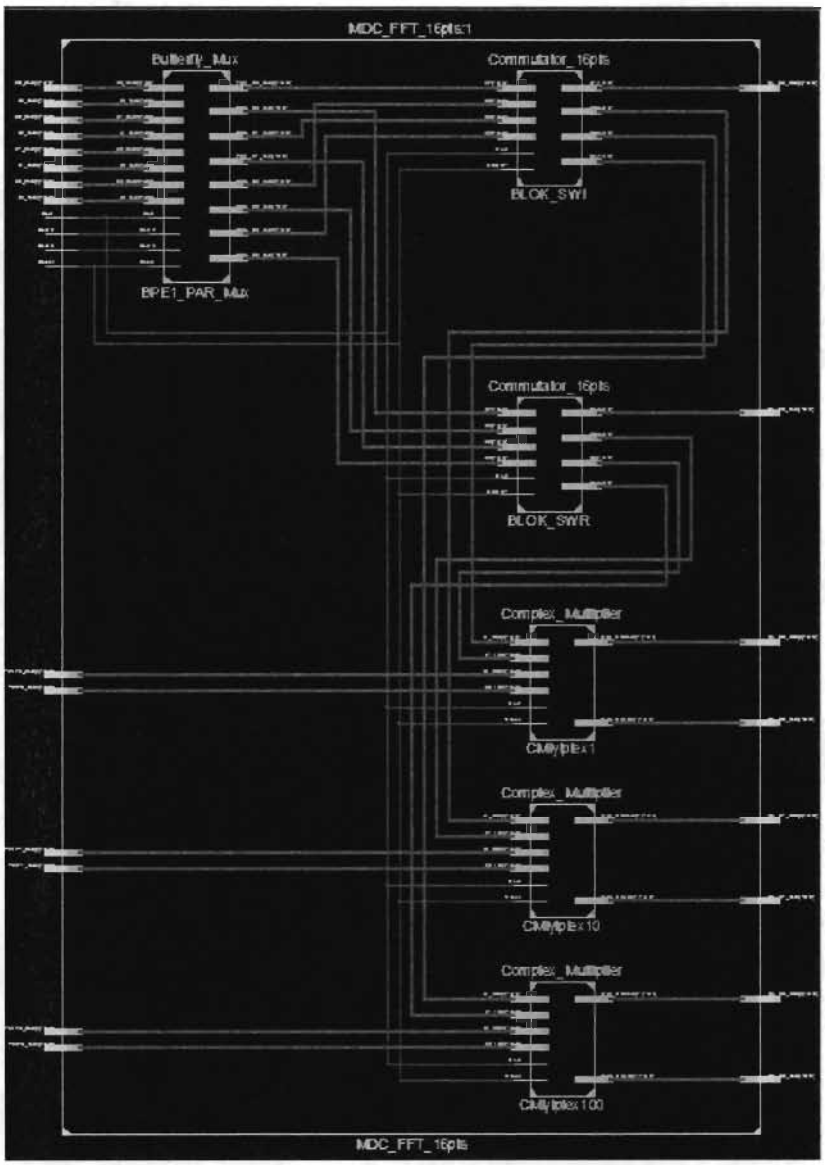


Figure 49 Schéma technologique complet du R4MDC à 16 points

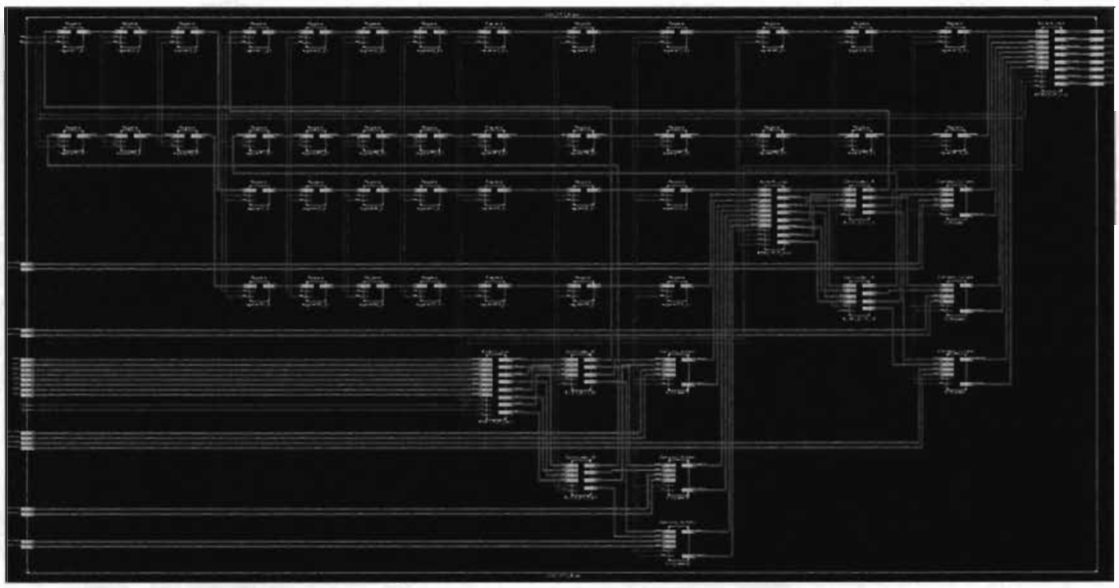


Figure 50 Schéma technologique complet du R4MDC à 64 points

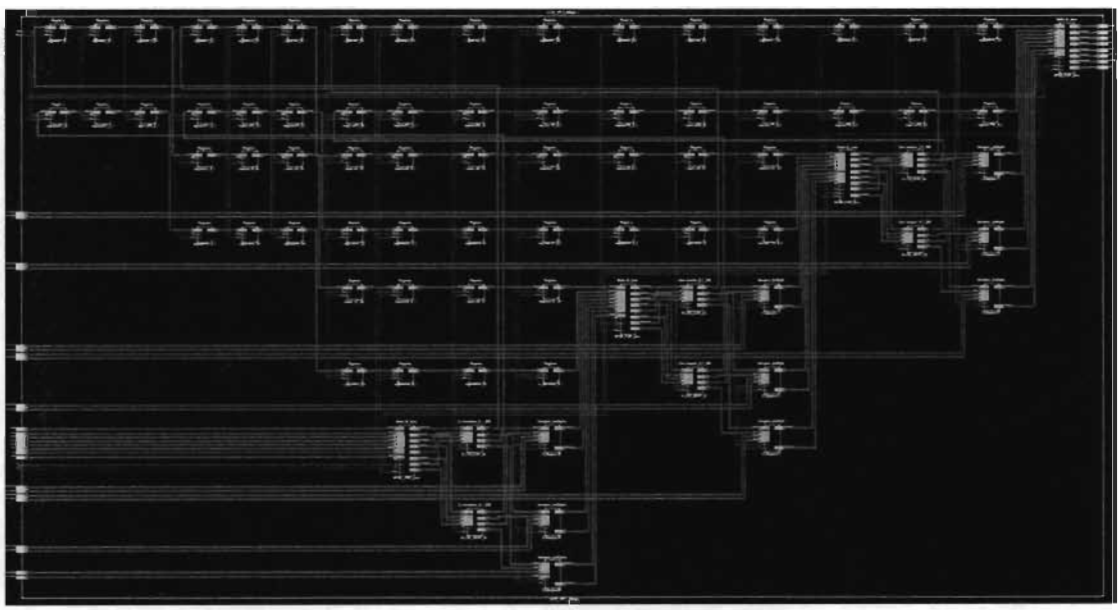


Figure 51 Schéma technologique complet du R4MDC à 256 points

CHAPITRE IV

CONCLUSION

Le centre d'intérêt actuel des chercheurs et des ingénieurs en microélectronique, c'est de concevoir un coprocesseur FFT qui réalise le compromis entre les ressources matérielles utilisées dans le processeur et la vitesse de calcul. L'architecture du coprocesseur représente donc l'élément principal de l'étude. D'une façon générale, les algorithmes FFT radix-2 et radix-4 conventionnels sont largement utilisés en industrie. L'architecture de leurs BPE facile à implémenter les privilégie par rapport à des algorithmes FFT de radix plus élevé. Les opérations arithmétiques pour calculer une FFT de N points diminue de plus en plus on utilise un grand radix, d'une autre part leur architecture du BPE devient plus complexe. La motivation de l'utilisation d'un grand radix réside donc dans la diminution globale du nombre de multiplication et addition complexes grâce à la diminution du nombre d'étages pour exécuter la FFT.

Pour répondre à ce besoin, une nouvelle formulation de la BPE a été développée au sein de notre laboratoire LSSI, [MJ14]. Ceci permet de les avec radix plus grand tout en visant une augmentation des performances du coprocesseur FFT, à savoir, augmentation du débit et réduction de la latence en rapport aux ressources d'implémentation nécessaire et ceci pour différent radix.

L'étude présentée dans ce mémoire est une évaluation comparative de l'implémentation de coprocesseurs FFT pour radix-4 sur FPGA. Cette évaluation se base sur plusieurs critères : débit du coprocesseur, latence, fonction de performance globale et autre. Les coprocesseurs FFT conventionnels sont utilisés comme référence de plancher puisqu'ils sont les plus communs. Par contre, les coprocesseurs FFT sur FPGA de l'article [MG13] sont considérés comme référence récente à surpasser grâce à leurs architectures améliorées. Les nouvelles architectures des coprocesseurs FFT ont été étudiées sur FPGA afin de connaître leur complexité et leurs performances. Ils représentent donc l'élément principal de ce travail. Les performances des BPE étudiées étaient satisfaisantes puisqu'elles permettent d'atteindre des fréquences plus grandes par rapport au BPE conventionnelle. L'introduction de ces nouvelles structures BPE aux architectures MDC FFT permet aussi d'améliorer les performances de la FFT et de surpasser ceux de la référence [MG13] dans différents aspects et surtout en termes de fréquence. Un article conférence est au cours d'être rédigé pour publier les résultats obtenus.

Références

[HS12] Hani H.M. Saleh and Earl E. Swartzlander, "FFT Implementation with Fused Floating-Point Operations", Issue No. 02 - February (2012 vol. 61).

[MJ10] Marwan A. Jaber and Daniel Massicotte, "Fast Method to Detect Specific Frequencies in Monitored Signal" Proceedings of the 4th International Symposium on Communications, Control and Signal Processing, ISCCSP 2010, Limassol, Cyprus, 3-5 March 2010

[MG13] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined Radix-2k Feedforward FFT Architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 23–32, 2013.

[MJ14] Marwan A. Jaber and Daniel Massicotte "Radix-2α/4β Building Blocks for Efficient VLSI's Higher Radices Butterflies Implementation" Hindawi Publishing Corporation VLSI Design volume 2014, 13 May 2014

[JR07] Joël LE ROUX " La Transformée de Fourier et ses Applications" 10 avr. 2007

[JWC65] J.W. Cooley and J.W. Tukey, "An Algorithm For The Machine Calculation Of Complex Fourier Series " Math. Comput., vol. 19, pp. 297-301, 1965.

[TYS10]T.Y. Sung, H.S. Hsin, Y.P. Cheng, "Low-power and High-speed CORDIC based Split-Radix FFT Processor for OFDM Systems", Digital Signal Prosessing, vol. 20, Issue 2, March 2010, Pages 511-527, Elsevier

[SW75] S. Winograd, private communications, July 1975.

[MJ09] M. Jaber, D. Massicotte. "A New FFT Concept for Efficient VLSI Implementation: Butterfly Processing Element". International Conference on Digital Signal Processing, July 5-7, Santorini, Greece 2009.

[MJ03] Y. Jung, H. Yoon, and J. Kim, "New Efficient FFT Algorithm and Pipeline Implementation Results for OFDMIDMT Applications", International Conference on Consumer Electronics, Feb 2003, Hong Kong.

[YW07] Y. Wang, Y.Tang, Y. Jiang, lG. Chung, S.S. Song and M.S. Lim, " Novel Memory Reference Reduction Methods for FFT Implementations on DSP Processors", IEEE Transaction on Signal Processing, vol. 55, no 5, pp. 2338 - 2349,2007.

[OA75] Oppenheim, A. V, Schafer. R. W, "Digital Signal Processing", Prentice-Hall, 1975.

[AV03] A.W.M. Van Den Enden, N.A.M.Verhoeckx, "Traitement Numérique Du Signal" 3e édition, Dunod 2003

[JG96] John G. Proakis, Dimitris G.Manolakis, "Digital Signal Processing, Principles, Aigorithms, and applications", Third Edition, Prentice hall.

[YT02] Y. Tang, Y. Jiang, and Y. Wang, "Reduce FFT Memory Reference For Low Power Applications", International conference on Acoustics Speech and Signal Processing, Orlando, Florida, USA, May 13-17,2002.

[YT04] Y. Tang; Wang, Y. Lim, J.G. Chung, S. Song, "High-Speed Assembly FFT Implementation with Memory Reference Reduction On DSP Processors", International Conference on Electronics, Circuits and Systems, December 13- 15, Tel-Aviv, Israel, 2004.

[YW07] Y. Wang, Y.Tang, Y. Jiang, lG. Chung, S.S. Song and M.S. Lim, " Novel Memory Reference Reduction Methods for FFT Implementations on DSP Processors", IEEE Transaction on Signal Processing , vol. 55, no 5, pp. 2338 - 2349,2007.

[TW97] T. Widhe, "Efficient Implementation of FFT Processing Elements", Linkoping studies in Science and Technology, Thesis No. 619, Linkoping University, Sweden, June 1997.

[MB98] M. Bellanger, "Traitement numérique du signal, Théorie et pratique" , Dunod, Paris 1998.

[YJ03] Y. Jung, H. Yoon, et J. Kim, "New Efficient FFT Aigorithrn and Pipeline Implementation Results for OFDMIDMT Applications", International Conference on Consumer Electronics, Feb 2003, Hong Kong.

[SH98] Shousheng. H, Torkelson.M, "Designing Pipeline FFT Processor for OFDM (de)modulation" Proceedings IEEE URSI. International Symposium on Signals, Systems and Electronics, pp. 257–262, october 1998.

[SL07] S. Lee, YJung, lKim, "Low Complexity Pipeline FFT Processor For MIMOOFDM Systems", Institute of Electronics, Information and Communication Engineers Electron Express, Vol. 4, No. 23, pp.750-754, 2007.

[HS09] Hsiang S. Hu, Hsiao-Y. Chen and Shyh-J. Jou, "Novel FFT Processor with Parallel-In-Parallel-Out in Normal Order", VLSI Design Automation and Test Symposium, Page(s): 150 - 153, 2009.

[FB09] Fu.B, Ampadu.P, "An Area Efficient FFT/IFFT Processor for MIMO-OFDM WLAN 802.11n", Journal of Signal Processing System, Springer, vol. 56, pp 59-58, september 2009.

[ES84] Swartzlander. E. E, Young. W. K.W, and Joseph, S. J, «A radix-4 delay commutator for fast Fourier transform processor implementation», IEEE J. Solid-state Circuits, vol. 19, No. 5, pp. 702-709, Oct. 1984.

[KM91] Kunt. M, Bellanger. M, De-Coulon. F, Guegen. C, «Techniques modernes de traitement numérique des signaux», volume 1 of Traitement de l'information. Presses polytechniques et universitaires romandes et CNET-ENST, 1ére édition, 1991.

[SS96] S. S. He and M. Torkelson, "A new approach to pipeline FFT processor," in Proc.10th Int. Parallel Process. Symp., Apr. 1996, pp.766–770.

[TM92] T. M. Hopkinson and G. M. Butler, "A pipelined, high-precision FFT architecture," in Proc. 35th Midwest Symp. Circuits Syst., Aug. 1992, vol. 2, pp. 835–838.

[LH99] L. H. Jia, Y. H. Gao, and H. Tenhunen, "A pipelined shared-memory architecture for FFT processors," in Proc. 42nd Midwest Symp. Circuits Syst., 1999, vol. 2, pp. 804–807.

[EH84] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," IEEE Trans. Comput., vol. C-33, no. 5, pp. 414–426, May 1984.

[SW01] S.W. Yu and E. E. Swartzlander, "A pipelined architecture for the multidimensional DFT," IEEE Transactions on Signal Processing, vol. 49, no. 9, pp. 2096–2102, Sep. 2001.

[AM74] A. M. Despain, "Fourier transform computer using CORDIC iterations," IEEE Trans. Comput., vol. C-23, no. 10, pp. 993–1001, Oct. 1974.

[GA89] G. A. Bi and E. V. Jones, "A pipelined FFT Processor For Word-Sequential Data," IEEE Transactions on Acoustics Speech and Signal Processing, vol. 37, no. ASSP-12, pp. 1982–1985, Dec. 1989.

[DC76] D. Cohen, "Simplified control of FFT Hardware," IEEE Transactions on Acoustics Speech and Signal Processing, vol. 24, no. 12, pp. 577–579, Dec. 1976.

[YM99] Y. Ma, "An Effective Memory Addressing Scheme for FFT Processors," IEEE Transactions on Signal Processing, vol. 47, no. 3, pp. 907–911, Mar. 1999.

[YM20] Y. Ma et L. Wanhammar, "A Hardware Efficient Control of Memory Addressing for High-Performance FFT Processors," IEEE Transactions on Signal Processing, vol. 48, no. 3, pp. 917–921, Mar. 2000.

[YM94] Y. Ma, "A Fast Address Generation Scheme for FFT Processors," Chinese J. Comput., vol. 17, no. 7, pp. 505–512, Jul. 1994.

[WZ94] W. Z. Luo and J. S. Xu, "Design of a New Type of FFT Address Generate IC," ACTA Electron. SINICA, vol. 22, no. 5, pp. 32–38, May 1994.

[SN10] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications," IEEE Transactions on Circuits Systems I, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[MS08] M. Shin and H. Lee, "A High-Speed, Four-Parallel radix- 2^4 FFT Processor for UWB Applications," in Proceedings, IEEE ISCAS, 2008, pp. 960–963.

[JY05] J.-Y. Oh and M. S. Lim, "Area and Power Efficient Pipeline FFT Algorithm," in Proceedings, IEEE Workshop SiPS, 2005, pp. 520–525.

[YW05] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1-GS/s FFT/IFFT Processor for UWB Applications," IEEE J. Solid-State Circuits, vol. 40, no. 8, pp. 1726– 1735, Aug. 2005.

[KM04] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-Point Fourier Transform Chip for High-Speed Wireless LAN Application Using OFDM," IEEE J. Solid-State Circuits, vol. 39, no. 3, pp. 484–493, Mars 2004.

[CC92] C. Chen and L. Wang, "A New Efficient Systolic Architecture for the 2-D Discrete Fourier Transform," in Proceedings. IEEE International Symposium on Circuits and Systems, vol. 6, ch. 732, 1992, pp. 689–692.

[GL04] G. L. Stuber, J. R. Barry, S. W. McLaughlin, Y. Li, M.A. Ingram, and T.H. Pratt, "Broadband MIMO-OFDM Wireless Communications", Proceedings IEEE, 92, (2), pp. 271–294, 2004

[DB04] D. Borkowski, and L. Bruhl, "Optimized Hardware Architecture for Real Time Equalization in Single and Multi-Carrier MIMO Systems", 3rd Workshop on Software Radio, Karlsruhe, Germany, 2004

[JL08] J. Lee and H. Lee, "A High-Speed 2-Parallel Radix-24 FFT/IFFT Processor for MB-OFDM UWB Systems" IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, vol. E91-A, no. 4, pp. 1206-1211, April 2008

[HL08] H. Liu and H. Lee, "A high performance four-parallel 128/64-point radix-24 FFT/IFFT processor for MIMO-OFDM systems," in Proceedings IEEE Asia Pacific Conference on Circuits Systems, 2008, pp. 834–837.

[SI08] S.-I. Cho, K.-M. Kang, and S.-S. Choi, "Implemention of 128-Point Fast Fourier Transform Processor for UWB Systems," in Proceedings International Wireless Communication and Mobile Computing Conference, 2008, pp. 210–213.

[CH07] C. H. Shin, S. S. Choi, H. H. Lee, and J. K. Pack, "A Design and Performance of 4-Parallel MB-OFDM UWB Receiver," IEICE Transactions on Communication, vol. E90-B, no. 3, pp. 672–675, Mar. 2007.

[YC08] Y.-N. Chang, "An Efficient VLSI Architecture for Normal I/O Order Pipeline FFT Design," IEEE Transactions on Circuits Systems II, vol. 55, no. 12, pp. 1234–1238, Dec. 2008.

[HG70] H. L. Groginsky and G. A. Works, "A Pipeline Fast Fourier Transform," IEEE Transactions Computers, vol. C-19, no. 11, pp. 1015–1019, Oct. 1970.

[BG73] B. Gold et T. Bially, "Parallelism in fast Fourier Transform Hardware," IEEE Transactions on Audio and Electroacoustics, vol. 21, no. 1, pp. 5–16, Feb. 1973.

[AD74] A. M. Despain, "Fourier Transform Computers Using CORDIC Iterations," IEEE Transactions Computers, vol. C-23, pp. 993–1001, Oct. 1974.

[SH98] S. He and M. Torkelson, "Design and Implementation of a 1024-Point Pipeline FFT Processor," in Proceedings IEEE Custom Integrated Circuits Conference, May 1998, pp. 131–134.

[AC09] A. Cortes, I. Velez et J. F. Sevillano, "Radix r^k FFTs: Matricial Representation and SDC/SDF Pipeline Implementation," IEEE Transactions on Signal Processing, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

[MG09] M. Garrido, "Efficient hardware architectures for the computation of theV FFT and other related signal processing algorithms in real time," Ph.D. dissertation, Universidad Politecnica de Madrid, 2009.

[DL02] D.L. Perry, "VHDL: Programming by Example", 4ième édition.

[MJ04] M. Jiang, B. Yang, Y. Fu, A. Jiang, Xin-a. Wang, X. Gan, B. Zhao, T. Zhang, "Design of FFT Processor with Low Power Complex Mutliplier for OFDMbased High-speed Wireless Applications", International Symposium on Communications and Information Technology, vol. 2, page 639, 2004.

[ML05] M.-S. Lim, J.-y. Oh, "Area and Power Efficient Pipeline FFT Algorithm", The IEEE Workshop on Signal Processing Systems, page 520, 2-4 Nov. 2005.

[MJ09] M. Jaber, D. Massicotte. "A New FFT Concept for Efficient VLSI Implementation: Butterfly Processing Element". International Conference on Digital Signal Processing, July 5-7, Santorini, Greece 2009.

[JY03] Jung. Y, Yoon. H, Kim. J, "New Efficient FFT Algorithm Pipeline Implementation Results OFDM/DMT Applications", IEEE Transaction on Consumer Electronics, Vol. 49, No. 1, pp.14-20, February 2003.

[SE84] Swartzlander. E. E, Young. W. K.W, and Joseph, S. J, "A radix-4 Delay Commutator for Fast Fourier Transform Processor Implementation", IEEE J. Solid-state Circuits, vol. 19, No. 5, pp. 702-709, Oct. 1984