Mise en œuvre d'un processeur FFT pour des applications OFDM

FFT processor implementation scheme for OFDM applications

Raouf Mokhnache[1*] & Azzouz Mokhnache[2]

[1] Département de Physique, Université Badji Mokhtar, BP 12, Annaba, 23000, Algérie.
[2] Département d'Electronique, Université Badji Mokhtar, BP 12, Annaba, 23000, Algérie.

Info. Article

Historique de l'article

Reçu le 10/12/2018
Révisé le 04/02/2019
Accepté le 05/02/2019

Mots-clés

Transformée de Fourier rapide (FFT) , architecture processeur FFT – FPGA, OFDM, mise en œuvre en temps réel

RESUME

La mise en œuvre de l'algorithme FFT dans une cible FPGA représente un problème non négligeable lorsqu'il s'agit de respecter les contraintes de l'application en termes de consommation d'énergie, de coût de mise en place, de surface occupée, et de vitesse de calcul.

Cet article étend l'utilisation de Radix-k à une structure appropriée d'un filtre basé sur un vecteur principal d'un composant d'une matrice FFT 2D réarrangée à l'aide de la décomposition DIT (Decimation In Time, décimation en temps). Les résultats obtenus ont démontré que la structure proposée est plus efficace que celles basées sur la rétroaction quand plusieurs séquences parallèles d'entrée doivent être traitées.

Keywords

Fast Fourier Transform (FFT) - FFT processor architecture - FPGA - OFDM, real time implementation

ABSTRACT

The implementation of the FFT algorithm in an FPGA target represents a significant problem when it comes to meeting the constraints of the application in terms of energy consumption, cost of implementation, occupied area, and calculation speed.

This work extends the use of Radix-k to an appropriate filter structure based on principal vector ofa component having 2D FFT matrix rearranged using DIT (time decimation) decomposition. The obtained results demonstrated that the proposed structure is more efficient than those based on feedback when multiple parallel input sequences are to be processed.

*** Auteur Correspondant**
Azzouz Mokhnache
Département d'Electronique, Université Badji Mokhtar, BP 12, Annaba, 23000, Algérie.

Email:mokhnache.azzouz.1234@gmail.com

1. INTRODUCTION

La transformée de Fourier rapide (FFT) est une méthode efficace pour calculer la transformée de Fourier discrète (DFT). Dans ce contexte la transformée de Fourier discrète d'un signal de N échantillons dans le domaine temporel x (n) est donnée par l'équation (1).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0,1,2,\cdots,N-1 \qquad (1)$$

Où$W_N^{nk} = e^{-j2\pi\left(\frac{kn}{N}\right)}$est une valeur complexe appelée facteur de pondération. Le calcul direct d'une DFT de N points nécessite$O(N^2)$opérations, alors que l'algorithme FFT nécessite approximativement un nombre total de multiplications complexes :$O\left((N/2)\log_2(N)\right)$ [1]. En fait, l'algorithme FFT utilise l'approche diviser et conquérir pour réduire les calculs [2]. L'idée est de diviser le problème en sous-problèmes avec la relation suivante :

$$\sum \quad Coût(sous-problémes) + Coût(découpageensous-problémes) <$$

$$Coût\ (problemeoriginal)$$

La méthode de division de la DFT en sous-problèmes classe les algorithmes FFT en deux familles : Cooley-Tukey et algorithmes PFA (Prime-Factor Algorithm). La méthode Cooley-Tukey est une méthode simple qui convient à tous les N puissance de 2 si l'on utilise des méthodes de base [3]. Chaque type d'algorithme est en outre classé en fonction d'autres caractéristiques : utilisation de mémoire supplémentaire, choix du type de décimation (Décimation En Temps),(Décimation En fréquence), etc. Dans l'algorithme de Cooley-TukeyRadix-k, la DFT de N points est subdivisée en deux DFTs de (N / k points) qui est ensuite divisée en deux DFTs plus petites jusqu'à obtenir une DFT de taille deux points, incluant les structures papillons qui sont juste des additions et des soustractions de nombres complexes de manière récursive. En fait, c'est le meilleur algorithme spécialement adapté pour un nombre N puissance de 2 (Radix-2). Des algorithmes de Radix supérieur à 2 peuvent être utilisés pour faciliter l'exploitation de la multiplication complexe. D'un autre côté, ils génèrent une complexité algorithmique qui impose une structure papillon plus compliquée [4]. Ainsi, un nouvel algorithme appelé Split Radix Algorithm est adopté pour tirer avantage des algorithmes radix-2 et de radix-4 afin d'obtenir un minimum de multiplications complexes avec la tenue d'une structure de papillon simple.

2. ALGORITHME PROPOSÉ

2.1Présentation de l'algorithme

Il est important que l'on comprenne exactement la structure de cet algorithme. Ce genre d'entité permettra au concepteur du circuit de tirer avantage de tout aspect de parallélisme ou de compilation particulière de l'algorithme et de les faire correspondre aux capacités des architectures les plus récentes.

Considérons une séquence entrante de N points de données complexes $S: \{x_0, x_1, x_2, \ldots, x_{N-1}\}$. Si N est positif et non premier, alors il peut être exprimé comme un produit d'au moins deux nombres L et M, c.-à-d.,

$$N = L \times M$$

Une séquence N-point de données peut être construite en utilisant la somme d'un certain nombre L de séquences contiguës de M points. Ces séquences auxiliaires sont successivement prélevées à partir de la séquence d'origine en gardant un sous-ensemble d'échantillons et de remplir le reste avec des zéros (Equ.2).

Une construction spécifique peut être définie comme suit :La séquence initiale à N points est divisée en L séquences auxiliaires somme de L $\{S^0, S^1, S^2, \ldots, S^{L-1}\}$, qui sont dénotées dans l'équation ci-dessous :

$$\begin{bmatrix} S^0 \\ S^1 \\ \cdot \\ \cdot \\ \cdot \\ S^{L-1} \end{bmatrix} =$$

$$\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{M-1} & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & x_M & x_{M+1} & \cdots & x_{2M-1} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & \cdots & \cdots & 0 & x_{(L-1)M} & x_{(L-1)M+1} & \cdots & x_{N-1} \end{bmatrix}$$

(2)

Il est à noter qu'il n'y a pas de chevauchement des segments d'entrée non nuls prélevés parmi les sous-séquences auxiliaires. Partant du fait que la DFT est une combinaison linéaire, la DFT de la séquence originale d'échantillons de N points est égale à la somme des DFT des L DFT des segments auxiliaires (3).

$$\begin{bmatrix} X_k^0 \\ X_k^1 \\ \vdots \\ X_k^{L-1} \end{bmatrix} = \begin{cases} \sum_{m=0}^{M-1} x_m W_{LM}^{mk} \\ \sum_{m=0}^{M-1} x_{m+M} W_{LM}^{mk} \left(W_L^k\right) \\ \vdots \\ \vdots \\ \sum_{m=0}^{M-1} x_{m+(L-1)} W_{LM}^{mk} \left(W_L^{k(L-1)}\right) \end{cases} \qquad (3)$$

$k = 0,1,2,\cdots,N-1$

En utilisant le principe de superposition, nous pouvons exprimer la DFT de la séquence indiquée comme suit :

$$X_k = \sum_{i=0}^{L-1} \left(X_k^i\right) \qquad (4)$$

L'équation précédente peut être réécrite sous la forme d'une double sommation avec l'introduction de l'index:

$$X_k = \sum_{l=0}^{L-1} \left(\sum_{m=0}^{M-1} x_{(m+lM)} W_{LM}^{mk}\right) W_L^{lk} \qquad (5)$$

Où $\begin{cases} l = 0,1,2,\cdots,L-1 \\ m = 0,1,2,\cdots,M-1 \end{cases}$

À partir de cette nouvelle représentation, il apparait que les propriétés de symétrie de la TFD jouent un rôle clé dans le développement de l'algorithme FFT. Il convient de noter que le terme exponentiel de la dernière double sommation dans l'équation (5) se répète de manière cyclique comme $\langle (lk)\, mod L \rangle$.

Considérons les comportements en fréquence des sous-groupes :

$\{X_0 \quad X_L \quad X_{2L} \quad \cdots \quad X_{(M-1)L}\}$;

$\{X_1 \quad X_{L+1} \quad X_{2L+1} \quad \cdots \quad X_{(M-1)L+1}\}; \cdots; \{X_{(L-1)} \quad X_{L+(L-1)} \quad X_{2L+(L-1)} \quad \cdots \quad X_{(M-1)L+(L-1)}\}$

La première séquence combinée de fréquences peut être découplée comme suit :

$$\begin{cases} X_0 = \sum_{l=0}^{L-1}\left(\sum_{m=0}^{M-1} x_{(m+lM)}\right) \\ X_L = \sum_{l=0}^{L-1}\left(\sum_{m=0}^{M-1} x_{(m+lM)} W_M^{m}\right) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ X_{(M-1)L} = \sum_{l=0}^{L-1}\left(\sum_{m=0}^{M-1} x_{(m+lM)} W_M^{m(M-1)}\right) \end{cases} \quad (6)$$

Avec l'introduction de l'index $k = \{k_0 \quad + \quad k_1 L\}$, où $\begin{bmatrix} k_0 = 0,1,2,\cdots L-1 \\ k_1 = 0,1,2,\cdots M-1 \end{bmatrix}$, un modèle commence à apparaître. Nous remarquons que la DFT peut maintenant être considérée comme L termes, contenant chacun M échantillons. Par conséquent, nous pouvons reformuler la DFT de la $N-point$ séquence initiale S comme une matrice bidimensionnelle [Tab.1] en introduisant cette nouvelle forme de l'index k :

$$X_k = X_{\underline{k_0 \quad + \quad k_1 L}} = \sum_{l=0}^{L-1}\left(\sum_{m=0}^{M-1} x_{(m+lM)} W_{LM}^{mk_0}\right)\left(W_L^{lk_0}\right)\left(W_M^{mk_1}\right) \quad (7)$$

Après la réorganisation des termes dans l'équation (7), on obtient :

$$X_{\underline{k_0 \quad + \quad k_1 L}} = \sum_{m=0}^{M-1} Y_{(mk_0)} W_M^{mk_1} \quad (8)$$

Où : $Y_{mk_0} = \left(W_{LM}^{mk_0}\right)\sum_{l=0}^{L-1}\left(x_{(m+lM)} W_L^{lk_0}\right)$

Selon la définition de base de la FFT, on voit tout de suite que la sommation sur $\boldsymbol{L}$ dans le terme $\boldsymbol{Y_{mk_0}}$ pour un $\boldsymbol{m}$ fixé n'est autre que la FFT d'un bloc décimé de $\boldsymbol{L}$ échantillons de la séquence N points d'origine $\boldsymbol{S}$ dans des $\boldsymbol{L}$ blocs contigus de $\boldsymbol{M}$ échantillons chacun, accumulés l'un au-dessus de l'autre. Pour être précis, les blocs décimés sont obtenus en prenant en compte tous les autres M échantillons de la séquence à N points introduits avec l'index $\boldsymbol{m}$, indiquant que le premier échantillon commence avec elle. Il est facile d'envisager l'élargissement de l'action de décimation en fractionnant la séquence à N points en $\boldsymbol{L}$ blocs contigus de $\boldsymbol{M}$ échantillons chacun, accumulés l'un au-dessus de l'autre.

De là, on dégage une topologie pour l'algorithme proposé, décomposée en quatre parties essentielles :

1. Un commutateur qui nécessite un circuit décimateur avec un circuit splitter pour gérer le flux des données et ainsi former les sous-groupes de données qui peuvent être empilées dans un espace mémoire préliminaire avant d'être acheminées vers le prochain étage. Cette étape sera matérialisée par une machine d'état spécifique pour répondre à notre exigence d'une telle conception. Bien sûr, cette étape nécessite environ N cycles en termes de latence et d'un grand espace mémoire comme des registres internes.

2. Une structure Radix-4 qui représente des facteurs de pondération triviaux ainsi que les facteurs de pondération principaux. Il convient de noter que cette structure est la plus gourmande en ressources internes et pour laquelle l'optimisation sera ostensiblement recommandée, pour accélérer la convergence de cet algorithme de manière significative.

3. Un bloc de multiplication complexe impliquant deux processeurs élémentaires réduisant la surface occupée ainsi que le temps de traitement. De la même manière, un espace mémoire sera prévu afin de stocker les résultats obtenus avant leur exploitation par une autre unité de traitement.

4. Il est crucial de prévoir un générateur d'adresses pour sélectionner la zone mémoire correspondante et donc d'établir la structure papillon imposée par le choix de la taille de l'unité de traitement en fonction du type de Radix à adopter pour l'architecture sélectionnée.

2.2 Exemple Introductif d'un processeur Radix-16

En application de ce qui précède, nous présentons ci-après un exemple introductif d'un processeur Radix-16. Cet algorithme est présenté comme une sorte de Radix-4 avec une architecture de type Single Path Delay feedback FFT [5]. N Doit être décomposé en un produit $N = LM = 16$. De toute évidence, la séquence de données d'entrée $x(n), 0 \leq n \leq 15$ est arrangée en une matrice 2-D [Tab.1] en prenant les index n comme une paire d'index (l, m) où $0 \leq l \leq 3$ et $0 \leq m \leq 3$. Ainsi, le réarrangement de $x(n)$ sous forme matricielle peut être ainsi exprimé:

Tableau 1 : Matrice bidimensionnelle N = l. x m

$$A = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} \quad (9)$$

Un réarrangement semblable est fait dans le domaine de fréquences par le remplacement de l'index $k$par une paire (k_0, k_1) d'index, d'où $0 \leq k_0 \leq 3$ et $0 \leq k_1 \leq 3$.

De ce fait, en sortie, les échantillons X_k sont stockés dans la mémoire intermédiaire (registres) en respectant l'ordre$k = 4k_1 + k_0$. Dans ce cas la matrice 2D est réarrangée comme suit :

$$X_{\underline{k_0 + 4k_1}} = \sum_{m=0}^{3} Y_{(mk_0)} W_4^{mk_1} \qquad (10)\text{où: } Y_{mk_0} = \left(W_{16}^{m\,k_0}\right) \sum_{l=0}^{3} \left(x_{(m+4l)}\, W_L^{lk_0}\right)$$

L'architecture préliminaire décrivant cet algorithme est détaillée sur la figure 1.

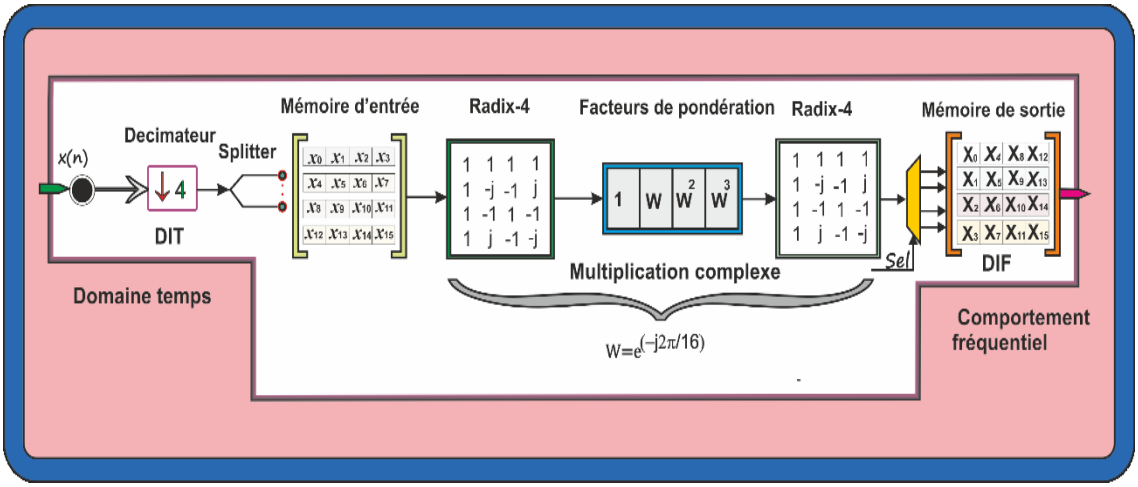

Figure.1 :Conception graphique d'un processeur élémentaire pour le calcul d'une FFT Radix-16.

3. CONCEPTION D'UN MULTIPLIEUR COMPLEXE

3.1 Présentation du problème

Avant la phase d'implémentation d'un tel composant et le choix d'une architecture appropriée,il est primordial de proposer une réduction ostensible en termes de ressources primitives afin d'améliorer de manière significative les performances souhaitées. Les multiplieurs complexes sont les plus gros consommateurs de ressources dans un processeur [6]. L'effort va donc porter sur la réduction de leur nombre au strict minimum.

En premier, considérons l'opération de multiplication fondamentale de deux quantités complexes :

$$P = (a + j \cdot b) \cdot (c + j \cdot s)$$

$$P = (a \cdot c - b \cdot s) + j(a \cdot s + b \cdot c)$$

Donc, cela donne :

La partie réelle :$Re(P) = a \cdot c - b \cdot s$ et la partie imaginaire :$Im(P) = a \cdot s + b \cdot c$

En fait, ces combinaisons doivent utiliser au total quatre composants multiplieurs et deux blocs additionneurs pour réaliser cette opération. Nous apportons une modification à la structure de cette équation pour construire un autre substitut en supprimant un composant multiplieur au profit d'un additionneur (Fig. 2). Évidemment, cette nouvelle configuration nécessite trois multiplieurs au lieu de quatre, combinés avec quatre blocs additionneurs pour effectuer l'opération de multiplication complexe.

$$\boldsymbol{Re(P) = (a + b) \cdot c - (c + s) \cdot b}$$
$$\boldsymbol{Im(P) = (a + b) \cdot c - (c - s) \cdot a}$$

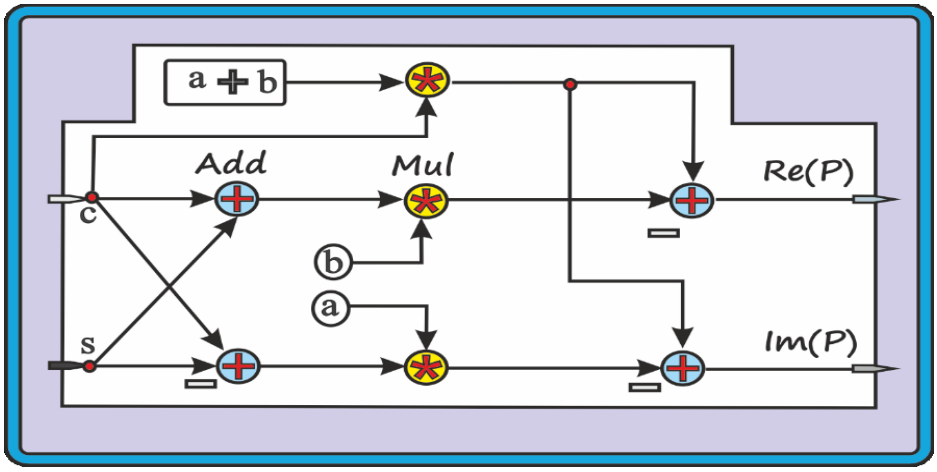

Figure.2 : Diagramme de base du circuit multiplieur basé sur la multiplication complexe

3.2 Problème de modélisation et simulation par Matlab

Pour bien comprendre le flux de notre approche, il convient d'élaborer une représentation graphique qui peut conduire au processeur de DFT portée sur un échantillon spécifique des données d'entrée. En introduisant le diagramme schématique du multiplieur complexe tel qu'illustré par la figure3, nous pouvons aisément décrire le circuit proposé concernant la première raie spectrale comme montré sur la figure 4.

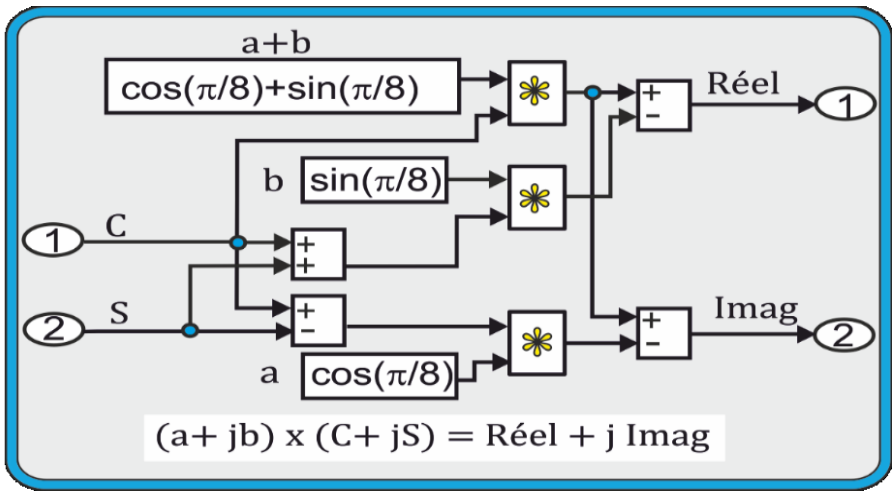

Figure.3 : Implémentation du premier facteur de pondération du multiplieur complexe

Le résultat obtenu, après avoir appliqué cette approche sur la séquence d'entrée $x\,(n) = [115, 97, 91, 31, 63, 255, 153, 41, 21, 3, 135, 1, 109, 145, 59, 128]$ en conservant le premier facteur de pondération: $e^{\left(\frac{-j\pi}{8}\right)}$ du vecteur composant principal, est clairement décortiqué via le schéma fonctionnel indiqué par la figure 2 et par la figure 3.

4. CONFRONTATION DES RESULTATS ET DISCUSSION

4.1 implémentation matérielle de l'algorithme proposé

Le grand challenge réside dans la façon de migrer de l'environnement Matlab (Fig. 4) vers une description matérielle en introduisant une cible FPGA, tout en veillant à réduire de façon significative la disproportionnalité entre le composant conçu et la simulation du modèle lui-même.

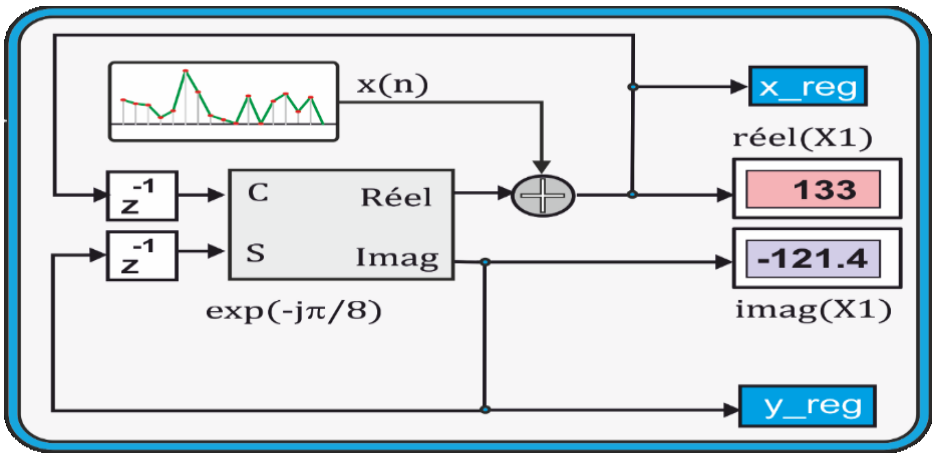

Figure.4 : Implémentation du processeur élémentaire pour le calcul de la première raie spectrale

Cette description illustre l'utilisation implicite d'un certain nombre de registres en pipeline où nous pouvons stocker les résultats intermédiaires, d'un multiplieur complexe et de deux unités de retard en amont du multiplieur pour les parties réelle et imaginaire. Donc, après activation du composant, celui-ci passe de l'état initial à l'état de fonctionnement et y reste, en mode occupé, jusqu'à ce que le compteur atteigne $(N + 1)$ cycles d'exécution, chaque cycle d'exécution comportant deux cycles d'horloge, N cycles d'exécution nécessaires au

calcul de la raie spectrale en question pendant lesquels la sortie est mémorisée dans les registres de sortie. En conséquence, un seul cycle est requis pour rendre la donnée disponible en sortie. Le processeur élémentaire revient alors à son état initial et le composant est prêt à réaliser un autre cycle Comme nous avons utilisé l'arithmétique en virgule fixe, certaines troncatures sont nécessaires pour conserver la plage dynamique des résultats et des sorties intermédiaires. Ces restrictions peuvent toucher la précision de l'algorithme FFT en introduisant le bruit de quantification. Il est donc important d'évaluer la précision du point fixe via la mesure du rapport du bruit signal-à-quantification SQNR (Signal-to-Quantization Noise Ratio). Notre objectif dans ce cas, est d'évaluer les répercussions de la troncature. En pratique nous n'avons considéré que la troncature pour les multiplieurs constants (pas pour les additionneurs). En effet, pour une multiplication avec un facteur de pondération codé en utilisant M bits, nous appliquons la troncature en utilisant M opérations de décalages à droite En conséquence la résolution de la sortie de l'architecture FFT à N points proposée est évaluée à $N + 2log_4(N)$ bits. D'ordinaire, le SQNR est défini par.

$$SQNR = \frac{\sum_{k=0}^{N-1}|x_k|^2}{\sum_{k=0}^{N-1}|x_{km}-x_k|^2}, \quad (11)$$

X_{km} Désigne e résultat obtenu en utilisant la précision 64 bits de l'environnement Matlab.

Selon les variations SQNR présentées [Fig. 5], on peut notamment montrer que pour une FFT de grande résolution, la valeur du SQNR diminue (la résolution du facteur de pondération est fixée à 10 bits). Ce phénomène est dû à la propagation du bruit de quantification. Enfin il convient d'indiquer que l'erreur quadratique moyenne maximale (MSE) obtenue en utilisant une FFT de 256 points est d'environ 2% [7].

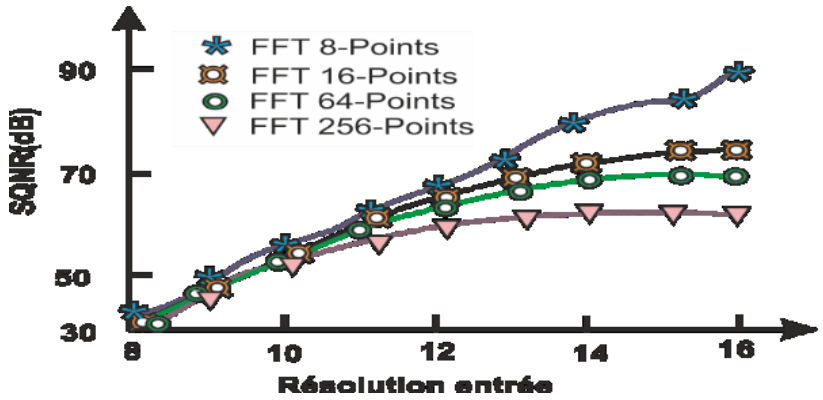

Figure.5:Evolution Du SQNR pour différentes résolutions d'entrées et FFT de différentes tailles

Les valeurs requises du facteur de pondération pour une multiplication pour l'algorithme proposé sont répertoriées dans le tableau 2. Elles montrent l'efficacité de cette approche par rapport à Radix-n FFT en termes de réduction du nombre de multiplications par les facteurs de pondération. Par exemple dans l'architecture FFT Radix-2 il faut 15 multiplieurs pour effectuer le calcul FFT [8, 9]. Mais dans notre cas ; seuls 9 multiplieurs sont nécessaires pour effectuer le calcul FFT. De cette manière, l'approche proposée réduit la complexité matérielle et la consommation d'énergie pour la conception du processeur FFT proposé.

Tableau 2 : Valeurs du facteur de pondération pour une FFT-16 points

Facteur de pondération	Valeurs
W_{16}^0	**1**
W_{16}^1	$0.9238 - j0.3826$
W_{16}^2	$0.707 - j0.707$

W_{16}^{3}	$0.3826 - j0.9238$
W_{16}^{4}	$-j$
W_{16}^{5}	$-0.3826 - j0.707$
W_{16}^{6}	$-0.707 - j0.707$
W_{16}^{7}	$-0.9238 - j0.3826$

La structure de la multiplication SBPM (Standard Bit Parallel base complex) de $Cos\left(\frac{\pi}{8}\right) = 0.9238$ est illustré par la suite. L'opération de multiplication d'un facteur de pondération de valeur $Cos\left(\frac{\pi}{8}\right)$ par une valeur quelconque xin de donnée en entrée est réalisée comme suit :

En factorisant et en écrivant $Cos\left(\frac{\pi}{8}\right) = [2^{-1} + 2^{-2} + (2^{-2} + 1)(2^{-6} + 2^{-3})]$, la multiplication $xin \times Cos\left(\frac{\pi}{8}\right)$ revient à multiplier par les différents termes entre crochets

$$xin \times Cos\left(\frac{\pi}{8}\right) = xin \times 0.9238 = xin \times [2^{-1} + 2^{-2} + (2^{-2} + 1)(2^{-6} + 2^{-3})]$$

En synthétisant le circuit tel qu'indiqué sur la figure 6, la multiplication se réduit à cinq opérations de décalage et quatre additionneurs.

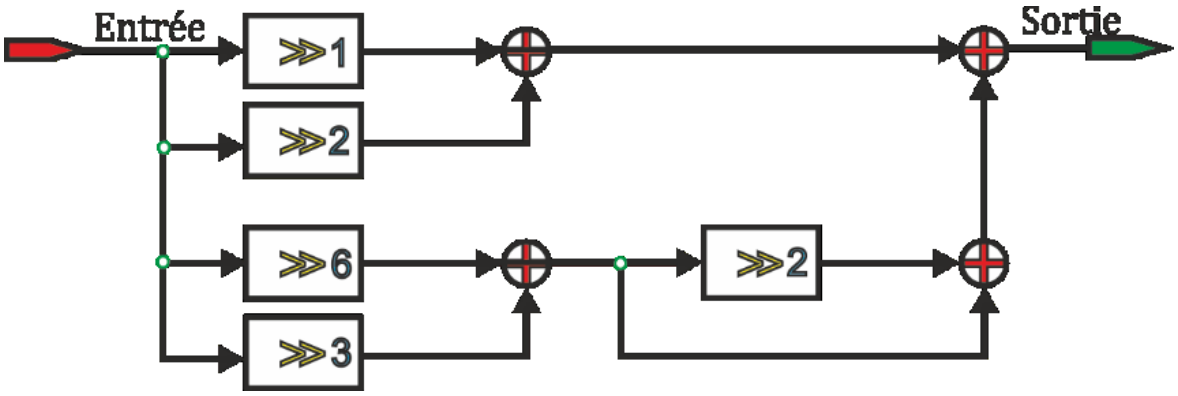

Figure 6 : Multiplieur du facteur $Cos(\pi/8)$.

De manière similaire, la multiplication du twiddle $Sin\left(\frac{\pi}{8}\right) = 0.3826$ avec les données entrées est illustrée par la suite (Fig.7). De manière significative, la multiplication de la valeur du facteur de pondération 0.3826 nécessite trois opérations de décalage et deux additionneurs pour effectuer cette multiplication. La forme pondérée de la valeur fractionnelle 0.3826 basée sur la notion de facteur commun peut réduire avantageusement la complexité algorithmique

$$xin \times Sin\left(\frac{\pi}{8}\right) = xin \times 0.3826 = xin \times 2^{-2}[1 + 2^{-1} + 2^{-5}]$$

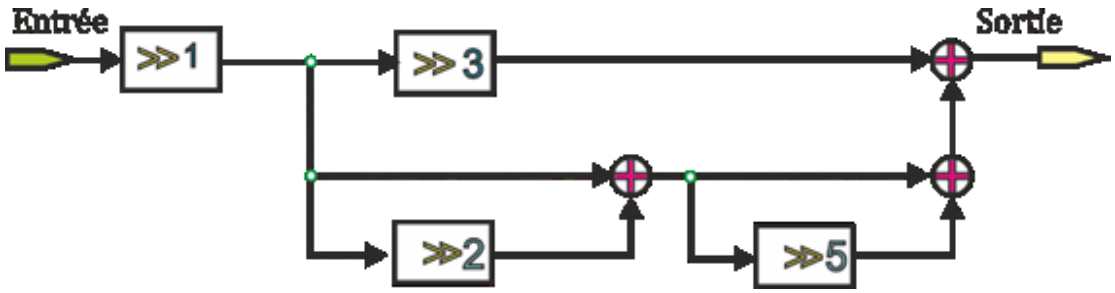

Figure 7 :Multiplieur du facteur $\boldsymbol{sin(\pi/8)}$

Le circuit conçu nécessite au total quatre registres à décalage et trois blocs additionneurs pour effectuer l'opération de multiplication (Fig.7). Finalement, le twiddle de valeur $\frac{\sqrt{2}}{2} = 0.707$, tenant compte de ce qui précède, peut être exprimé comme suit, en factorisant et en écrivant de manière similaire :

$$xin \times \frac{\sqrt{2}}{2} = in \times 0.707 = xin \times 2^{-1}[2^{-3} + (1 + 2^{-2})(1 + 2^{-5})]$$

4.2 Confrontation et discussion

Nous nous proposons alors de mesurer les performances métriques de l'algorithme adopté. L'algorithme a été vérifié sous MATLAB et les résultats comparés avec les fonctions propres à MATLAB. Les résultats obtenus sont identiques.

Le code VHDL de l'algorithme a été développé et simulé à l'aide du logiciel Modelsim version 10.1d. La séquence d'entrée :

$x\ (n)\ =\ [115, 97, 91, 31, 63, 255, 153, 41, 21, 3, 135, 1, 109, 145, 59,\ 128]$ sert de séquence test pour évaluer les performances.

L'architecture suggérée a été élaborée pour les cibles FPGA. La conception proposée a été synthétisée avec le logiciel ISE 14.7 et mise en œuvre sur une architecture Xilinx Virtex-6 xcvlx75t-2ff484. Le modèle a été conçu sous simulink (Fig.4). Les conceptions sont configurables en nombre d'échantillons à utiliser, en résolution du facteur de pondération et en structure de multiplication complexe (parallèle base complexe multiplication). Les résultats après placement et routage pour différentes configurations de N (taille de la FFT) et en utilisant différentes résolutions d'entrée (résolutions P = 8, 10, 12, 14 et 16 bits) sont présentés [Tab.3].Dans la conception proposée, ces blocs ont été utilisés pour mettre en œuvre les multiplieurs complexes qui réalisent le processeur élémentaire de l'algorithme FFT. Comme reporté dans le tableau 3, la complexité de l'algorithme augmente lorsqu'une FFT avec un grand nombre de points N est utilisée dans la conception. Un niveau de complexité plus élevé peut conduire à un schéma de configuration difficile sur cible FPGA. Plus de complexité implique aussi plus d'interconnexions, qui sont beaucoup plus lentes en termes de latence sur FPGA que la logique primitive elle-même. Lorsque la fréquence de fonctionnement maximale est augmentée, plusieurs registres de pipeline sont nécessaires pour satisfaire les exigences du timing. Des registres supplémentaires ainsi que des dispositifs DSP pourraient figurer au lieu des blocs actuels. Dans ce cas, on peut augmenter la fréquence de fonctionnement et ainsi améliorer la performance et la fréquence de fonctionnement peut augmenter.

Tableau 3 : Performances de l'algorithme proposé - Ref : [8],[9],[10] Ap : Algorithme proposé

Tailles FFT	**Slices**		**Latences**				**Fmax (MHZ)**		**Débits (Ms/s)**	
	Ref.	Ap.	Ref.		Ap.		Ref.	Ap.	Ref.	Ap.
			µs	Cycles	µs	Cycles				
16	386	381	0.026	12	0.022	11	458	470	1831	1880
64	695	663	0.081	32	0.070	28	384	392	1536	1568
256	1024	1019	0.221	86	0.190	76	389	400	1554	1600
1024	1425	1415	1.055	285	1.022	280	270	280	1081	1120
4096	2388	2355	6.120	1058	5.990	1048	173	175	693	700

Il peut être observé que la conception proposée requiert moins d'espace en termes de ressources internes (colonne registres slice du tableau 3) que les autres architectures pour n'importe quelle longueur N de la FFT [8, 9, 10]. Cette amélioration augmente avec la longueur de la FFT. Ainsi, ces chiffres montrent que le concept proposé est clairement plus amélioré que les architectures radix-4. L'algorithme proposé nécessite plus de registres en pipeline, ce qui se traduit par une augmentation significative du nombre de registres à utiliser pour le circuit du processeur FFT. Cependant, la différence devient plus petite lorsque la fréquence de fonctionnement augmente, on constate aussi que la conception de radix-4 commence aussi à exiger plus de registres pipelines. Basé sur ces résultats, l'algorithme proposé exige ostensiblement moins de ressources que les architectures radix-4 jusqu'à 350 Mhz. En conséquence, la conception radix-4 reste une perspective prometteuse pour des fréquences de fonctionnement de près de 200 Mhz. D'autre part les résultats répertoriés dans la colonne débits du tableau [Tab.3]illustrent les comparaisons de débit entre l'approche proposée et divers modèles basés sur des FFT Radix-4 en pipeline. Comme on peut le constater l'architecture proposée atteint les débits les plus élevés par rapport aux autres conceptions [8], [9], [10]. Un débit encore plus élevé peut être obtenu en ayant recours à des topologies à base 2 si le besoin s'en fait sentir.

5. CONCLUSION

Cet article étend l'utilisation de Radix-k à une structure appropriée d'un filtre basé sur un vecteur principal d'un composant d'une matrice FFT 2D réarrangée à l'aide de la décomposition DIT (décimation en temps). En effet, il est démontré que cette structure est plus efficace que celles basées sur la rétro actions quand plusieurs séquences parallèles d'entrée doivent être traitées. De toute évidence, les architectures Radix-k peuvent être utilisées pour n'importe quelle séquence de données d'entrée puissance de deux. Généralement, le nombre d'échantillons parallèles peut être choisi arbitrairement en fonction de la performance métrique requise à cet effet. En outre, le choix du vecteur principal du composant de la matrice FFT 2D réarrangée, censé être une information révélant le comportement fréquentiel de la séquence d'entrée à étudier, joue un rôle clé dans la mise en œuvre optimale de cet algorithme. Enfin, les résultats expérimentaux confirment que le modèle proposé est clairement efficace aussi bien en surface occupée, en latence et en performances de débit, rendant possible l'obtention de débits plus élevés, autour de deux Giga échantillons/s ,avec de faibles latences.

REFERENCES

[1] Arif M., "A novel approach for DFT computation", 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], 19-20 March 2015.,

[2] Arun C.A. & Prakasam P. , "Design of high speed FFT algorithm For OFDM technique", 2016 Conference on Emerging Devices and Smart Systems (ICEDSS), 4-5 March 2016,

[3] Kwong J. & Goel M., "A high performance split-radix FFT with constant geometry architecture," 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE).

[4] B V Uma et al., Area and time optimized realization of 16 point FFT and IFFT blocks by using IEEE 754 single precision complex floating point adder and multiplier", 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), DOI: 10.1109/ICSCTI.2015.7489546

[5] Wang Z.K.er al. "A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT," IEEE Transaction on Very Large Scale Integration (VLSI) Systems (Volume: 23, may 2015),

[6] Nguyen T.T.B. & Lee H. , "Shared CSD complex constant multiplier for parallel FFT processors", 2015 International SoC Design Conference (ISOCC).

[7] Gupta P , "Accurate performance analysis of a fixed point FFT", 2016 Twenty Second National Conference on Communication (NCC).

[8] Gálvez M.G. , Grajal J., Sanchez M. A. and Gustafsson O. , "Pipelined Radix-2(k) Feed forward FFT Architectures", 2013, IEEE Transactions on Very Large Scale Integration (VLSI Systems).

[9] Sanchez M.A., Garrido M., Lopez-Vallejo M., Grajal J., "Implementing FFT-based digital channelized receivers on FPGA platforms," IEEE Transactions on Aerospace and Electronic Systems (Volume: 44, Issue: 4, Oct. 2008),.

[10] Wang C., Yan Y., Fu X., "A high-throughput low-complexity radix-24 -22 -23 FFT/IFFT processor with parallel and normal input/output order for IEEE 802.11ad systems," IEEE Transactions on Very Large Scale Integration Systems (VLSI 2015).