

Université KASDI-MERBAH Ouargla

Faculté des Nouvelles Technologies de l'Informatique et de la communication

Département d'Electronique et Des Télécommunications



Mémoire

Présenté pour l'obtention du diplôme de

MASTER ACADEMIQUE

Domaine : Sciences et Technologies.

Filière : Télécommunication.

Spécialité : Système de communication.

Présenté par : **Ouargli Yamna, Benaicha djazia**

Thème :

Implémentation HDL d'un système de communication OFDM LTE pour FPGA

Devant le jury composé de :

| | | | |
|--------------------------------|-----------|-----------------------|-------------|
| Mr. BENARABI Bilal | MCB | Président | UKM-Ouargla |
| Mr. M ^{ED} SAYAH Moad | MAA | Encadreur/ Rapporteur | UKM-Ouargla |
| Mr. BOUNEGAB Abdelhamid | Doctorant | Co-Encadreur | UKM-Ouargla |
| Mr. MELHEGUEG Nacer | MAA | Examineur | UKM-Ouargla |

Année universitaire : 2019/2020

ملخص

OFDM (تعدد الإرسال المتعامد بتقسيم التردد) هو معيار الإرسال اللاسلكي الأكثر استخدامًا في العالم في العديد من الأنظمة مثل ADSL و DAB و DVB-T2 و DVB-S2 و Wifi و Wimax و LTE (4G) و NR (5G.) يتمتع OFDM بمزايا الكفاءة الطيفية العالية والمتانة المتأصلة ضد خبو القناة ، لذا فهو مستخدم في نظام LTE يقدم هذا العمل جهاز إرسال واستقبال باستخدام تخطيطي لمغير OFDM وكاشف عبر قناة AWGN. يتم تنفيذ جهاز الإرسال والاستقبال بلغة HDL على / Matlab Simulink. تتم مقارنة نتائج تطبيق HDL مع إشارة مرجعية ثم دراسة التباين في عرض النطاق الترددي و SNR وتأخير القناة بعد إنشاء رمز خط HDL ، يكون التنفيذ جاهزًا للتنفيذ في الأجهزة على (FPGA مصفوفة البوابة القابلة للبرمجة الميدانية) عبر نظام الجيل Xilinx .

الكلمات الرئيسية: دائرة FPGA ، نظام اتصالات OFDMLTE ، تنفيذ HDL ، نقل لاسلكي.

RESUME

L'OFDM (Orthogonal Frequency-Division Multiplexing) est la transmission sans fil standard la plus utilisée au monde dans plusieurs systèmes tel que L'ADSL, DAB, DVB-T2, DVB-S2, Wifi, Wimax, LTE (4G) et NR (5G.) L'OFDM présente les avantages d'une efficacité spectrale élevée et d'une robustesse inhérente contre les évanouissements de canal, de sorte qu'il est utilisé dans le système LTE. Ce travail présente un émetteur-récepteur en utilisant un schéma d'un modulateur et d'un détecteur OFDM via un canal AWGN. L'émetteur-récepteur est implémenté en langage HDL sur Matlab/Simulink. Les résultats d'implémentation HDL sont comparés avec un signal de référence ensuite l'étude de variation de la bande passante, SNR et Delay canal. Après la génération de code ligne HDL, l'implémentation réalisée est prête à une implémentation en matériels sur FPGA (Field Programmable Gate Array) via le système de génération Xilinx.

MOTS-CLES : circuit FPGA, système de communication OFDMLTE, implémentation HDL, transmission sans fil.

ABSTRACT

LTE is the most widely used standard wireless transmission in the world such as ADSL, DAB, DVB-T2, DVB-S2, Wifi, Wimax, LTE 4G and NR 5G. OFDM (Orthogonal Frequency-Division Multiplexing) has the advantages of high spectral efficiency and inherent robustness against channel fading, so it is used in LTE system. This work presents a transceiver using a schematic of a Modulator and OFDM Detector via an AWGN channel. The transceiver is implemented over HDL on matlab / Simulink. The results of HDL implementation are compared with a reference signal then the study of variation in bandwidth, SNR and channel delay. After the generation of HDL line code, the implementation is ready for implementation in hardware on FPGA (Field Programmable Gate Array) via the Xilinx generation system.

KEYWORDS: FPGA circuit, OFDMLTE communication system, HDL implementation, wireless transmission.

Sommaire

| | |
|--|----|
| INTRODUCTION GÉNÉRALE..... | a |
| CHAPITRE 1..... | 1 |
| 1.1 Introduction | 1 |
| 1.2 Principe OFDM | 2 |
| 1.3 Implémentation numérique de la technique OFDM | 4 |
| 1.3.1 Implémentation numérique du modulateur OFDM | 4 |
| 1.3.2 Implémentation numérique du démodulateur OFDM..... | 5 |
| 1.4 Canal multi-trajet..... | 5 |
| 1.4.1 Aperçu sur le canal multi trajet..... | 5 |
| 1.4.2 Caractéristique d'un canal à trajet multiple | 6 |
| 1.5 Chaîne de transmission pour OFDM LTE | 7 |
| 1.5.1 Accès radio LTE..... | 7 |
| 1.6 Techniques de transmission OFDMA..... | 9 |
| 1.7 Technique de transmission SC-FDMA | 11 |
| 1.8 Bilan de liaison | 12 |
| 1.9 Planification de fréquence | 12 |
| 1.10 Les Avantages et l'inconvénient de l'OFDM LTE..... | 13 |
| 1.10.1 Les Avantages et l'inconvénient de l'OFDMA :(liaison montante)..... | 13 |
| 1.10.2 Les Avantages et l'inconvénient de l'OFDMA :(liaison descendante) | 13 |
| 1.11 Comparaison entre technologie base sur l'OFDM | 13 |
| 1.12 Evolution pour implémentation LTE | 15 |
| 1.13 Conclusion..... | 16 |
| CHAPIRE 2..... | 17 |
| 2.1 Introduction | 18 |
| 2.2 Langages de programmation HDL | 18 |

| | | |
|-------|---|----|
| 2.3 | Le FPGA (Field Programmable GateArrays) | 19 |
| 2.3.1 | Facteurs clés pour comparer les FPGA | 21 |
| 2.3.2 | Description des familles FPGA | 22 |
| 2.4 | Procédure d'implémentations et de Vérification HDL | 22 |
| 2.5 | Architecture des circuits FPGA | 23 |
| 2.5.1 | Les Interconnexions FPGA | 23 |
| 2.6 | Principe duVHDL | 24 |
| 2.6.1 | Historique | 24 |
| 2.6.2 | Les caractéristiques du VHDL | 25 |
| 2.6.3 | Structure du langage VHDL..... | 25 |
| 2.7 | Implémentation d'un circuit numérique de communication | 26 |
| 2.7.1 | Implémentation XSG MATLAB | 26 |
| 2.8 | Implémentation In the loop..... | 27 |
| 2.9 | Implémentation Xilinx | 28 |
| 2.10 | Simulations avec Simulink..... | 29 |
| 2.11 | Conclusion..... | 31 |
| | CHAPITRE 3..... | 32 |
| 3.1 | Introduction | 33 |
| 3.2 | Logiciels de simulation | 34 |
| 3.3 | Simulation HDL implémentation OFDM LTE | 34 |
| 3.3.1 | Aperçu | 34 |
| 3.2 | Fonctions de System Toolbox LTE..... | 35 |
| 3.2.1 | Fonctions utilisées dans le script d'initialisation LTE..... | 36 |
| 3.2.2 | Fonctions utilisées dans le script d'analyse après simulation | 36 |
| 3.3 | Structure de Simulations avec Simulink | 36 |
| 3.4 | Sous-système HDL du modulateur LTE | 37 |

| | | |
|-------|--|-----|
| 3.5 | Sous-système HDL détecteur LTE..... | 38 |
| 3.6 | La structure du sous-système canal..... | 39 |
| 3.7 | Résultats de simulations..... | 39 |
| 3.8 | Analyse des résultats..... | 44 |
| 3.8.1 | Influence de variation Bande Passante, Atténuation canal et SNR canal..... | 44 |
| 3.9 | Génération du code HDL du model modulateur /Détecteur OFDM LTE | 58 |
| 3.10 | Conclusion :..... | 59 |
| | Conclusion Générale | 60 |
| | ANNEXE 1 | rrr |
| | ANNEXE 2 | yyy |

| | |
|--------------------|---|
| Figure 1.1 | Schéma de principe d'un modulateur OFDM |
| Figure 1.2 | Spectre du signal OFDM avec N porteuse |
| Figure 1.3 | Modulateur OFDM numérique |
| Figure 1.4 | Démodulateur OFDM numérique |
| Figure 1.5 | Schéma de principe des trajets multiple |
| Figure 1.6 | Émetteur / Récepteur des systèmes OFDMA et SC-FDMA. |
| Figure 1.7 | Représentation fréquentielle et temporelle d'un signal OFDM. |
| Figure 1.8 | Signal OFDM dans les domaines temporel et fréquentiel |
| Figure 1.9 | Chaîne de transmission et de réception SC-FDMA. |
| Figure 1.10 | de l'ensemble des sous-porteuses |
| Figure 2.1 | Part de marche PLD (dispositifs logiques programmables) |
| Figure 2.3 | Xilinx système génération pour similitude |
| Figure 2.4 | Carte FPGA |
| Figure 2.5 | LTE Structure du modulateur. |
| Figure 2.6 | Structure du détecteur LTE |
| Figure 3.1 | Structure du modulateur et du détecteur LTE |
| Figure 3.2 | l'implémentation HDL du modulateur et détecteur OFDM LTE |
| Figure 3.3 | Structure d'implémentation HDL d'un modulateur OFDMLTE |
| Figure 3.4 | Structure d'implémentation d'un détecteur LTE HDL |
| Figure 3.5 | La structure du sous-système canal AWGN |
| Figure 3.6 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM |
| Figure 3.7 | Graphique d'estimation de fréquence |
| Figure 3.8 | Diagramme de corrélation croisée PSS |
| Figure 3.9 | Diagramme de corrélation croisée SSS |
| Figure 3.10 | Résultats basés sur les étapes finales de la simulation |
| Figure 3.11 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=1.4MHZ) |
| Figure 3.12 | Diagramme de corrélation croisée SSS (BW=1.4MHZ) |
| Figure 3.13 | Diagramme de corrélation croisée PSS (BW=1.4MHZ) |

| | |
|--------------------|--|
| Figure 3.14 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ) |
| Figure 3.15 | Diagramme de corrélation croisée PSS (BW=10MHZ) |
| Figure 3.16 | Diagramme de corrélation croisée SSS (BW=10MHZ) |
| Figure 3.17 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ) |
| Figure 3.18 | Diagramme de corrélation croisée SSS (BW=10MHZ) |
| Figure 3.19 | Diagramme de corrélation croisée PSS (BW=10MHZ) |
| Figure 3.20 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ) |
| Figure 3.21 | Diagramme de corrélation croisée PSS (BW=10MHZ) |
| Figure 3.22 | Diagramme de corrélation croisée SSS (BW=10MHZ) |
| Figure 3.23 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ) |
| Figure 3.24 | Diagramme de corrélation croisée SSS (BW=10MHZ) |
| Figure 3.25 | Diagramme de corrélation croisée PSS (BW=10MHZ) |
| Figure 3.26 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=15MHZ) |
| Figure 3.27 | Diagramme de corrélation croisée SSS (BW=15MHZ) |
| Figure 3.28 | Diagramme de corrélation croisée PSS (BW=15MHZ) |
| Figure 3.29 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=20MHZ) |
| Figure 3.30 | Diagramme de corrélation croisée PSS (BW=20MHZ) |
| Figure 3.31 | Diagramme de corrélation croisée SSS (BW=20MHZ) |
| Figure 3.32 | Graphique d'estimation de fréquence (BW=3MHZ) |
| Figure 3.33 | Diagramme de corrélation croisée PSS (BW=3MHZ) |
| Figure 3.34 | Diagramme de corrélation croisée SSS (BW=3MHZ) |
| Figure 3.35 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ) |

Listes des figures

| | |
|--------------------|--|
| Figure 3.36 | Diagramme de corrélation croisée PSS (BW=10MHZ) |
| Figure 3.37 | Diagramme de corrélation croisée SSS (BW=10MHZ) |
| Figure 3.38 | Graphique d'estimation de fréquence (BW=10MHZ) |
| Figure 3.39 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=15MHZ) |
| Figure 3.40 | Graphique d'estimation de fréquence (BW=10MHZ) |
| Figure 3.41 | Diagramme de corrélation croisée PSS (BW=15MHZ) |
| Figure 3.42 | Diagramme de corrélation croisée SSS (BW=15MHZ) |
| Figure 3.43 | Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=20MHZ) |
| Figure 3.44 | Graphique d'estimation de fréquence (BW=20MHZ) |
| Figure 3.45 | Diagramme de corrélation croisée PSS (BW=20MHZ) |
| Figure 3.46 | Diagramme de corrélation croisée SSS (BW=20MHZ) |

Liste des tableaux

| | |
|--------------------|--|
| Tableau 1.1 | Comparaison entre LTE, IEEE 802.11, WIMAX et HSI-OFDM |
| Tableau 2.1 | Principaux domaines d'application FPGA |
| Tableau 2.2 | Familles de conception FPGA |
| Tableau 2.3 | comparaison des différentes familles de FPGA de Xilinx |
| Tableau 3.1 | Bande Passante OFDM LTE |

Liste des abréviations

| | |
|----------------|---|
| OFDM | Orthogonal Frequency Division Multiplexing |
| FFT | Fast Fourier transform |
| FDM | Frequency Division Multiplexing |
| (TEB) | Le taux d'erreur binaire |
| (EQM) | Erreur quadratique moyenne |
| (RSB) | Le rapport signal sur bruit |
| LTE | Long Term Evolution |
| SC-FDMA | Single Carrier Frequency Division Multiple Access) |
| IFFT | Inverse Fast Fourier Transform). |
| QAM | Quadrature amplitude modulation |
| OFDMA | orthogonal frequency-division multiple access |
| Wifi | Wireless Fidelity |
| WIMAX | acronyme pour Worldwide Interoperability for Microwave Access |
| VHDL | Hardware Description Language |
| FPGA | field-programmable gate array |
| IEEE | Institute of Electrical and Electronics Engineers |
| CLB | Canadian Language Benchmark |
| HDL | high-density lipoprotein |
| CP | Cyclic Prefix |
| AWGN | Additive white Gaussian noise |
| DAC | digital-to-analog converter |
| ADC | analog-to-digital converter |
| FIR | Flight Information Region |

INTRODUCTION GÉNÉRALE

Les systèmes mobiles avec le standard LTE est le portail vers les réseaux 4G qui propose une technique OFDMA (Orthogonal Frequency Division Multiple Access) à la voie montante et une autre technique d'accès pour la voix appelée le SC-FDMA (Single Carrier Frequency Division Multiple Access). Et actuellement les futurs systèmes 5G ont l'intention de réaliser des débits de données encore plus élevés avec la nouvelle interface physique NR (New radio) basé sur l'OFDMA avec numérogie.

Les systèmes à base de modulation OFDM LTE sont capables de fournir un débit élevé, une réduction de la complexité du récepteur et une amélioration de l'efficacité spectrale. Cependant les inconvénients de l'OFDM sont le niveau élevé du PAPR et des lobes qui sont hors de la bande.

L'objectif de ce mémoire est l'étude et la simulation d'implémentation HDL d'un système de communication OFDM LTE afin de tester les performances de la modulation OFDM implémenté et la generation du code HDL a l'aide des nouvelles methodes et fonctionnalité introduit par Matlab/Simulink choses que nous permet de faire une implémentation matériels sur FPGA dans un futur travail

Ce mémoire est organisé en trois chapitres, dans le premier chapitre on se verra de présenter le principe de la Technique OFDM, on commencera tout d'abord par l'étude de leur implémentation après on va parler sur les canaux multitrajet connu, ensuite on passera à la generation LTE sur l'OFDMA en détaillons le bilan de liaison, la planification de fréquence et leur evolution d'implémentation.

Le second chapitre traite contient une description générale d'implémentation HDL utilisé pour une implémentation matériels sur FPGA dans laquelle, ou nous avons expliquent le principe et les methode d'implémentation du système OFDM. Une implémentation OFDMA qui est capable de fournir un débit binaire élevé, de réduire la complexité du récepteur et d'améliorer l'efficacité spectrale avec l'utilisation d'implémentation FFT/IFFT et de filtrage accordé.

Enfin le troisième chapitre montre une simulation pour une implémentation de la modulation OFDM LTE présentées précédemment avec Leur performance Ainsi la generation d'un code HDL pour une implémentation Matériels sur FPGA.

Chapitre I :

Modulation multi porteuse OFDM

LTE

1.1 Introduction

Le multiplexage par répartition orthogonale de la fréquence (OFDM) concept que l'on appellera "Orthogonal Frequency Division Multiplexing" est une technique de transmission qui garantit une utilisation bien organisée du spectre par le concept de chevauchement des porteuses. Il s'agit d'un amalgame de modulation et de multiplexage qui est utilisé dans la transmission d'informations et de données. Avec l'augmentation du trafic, la nécessité d'une utilisation efficace du spectre joue un rôle crucial dans la future modulation sans fil. Le système de communication 4G est utilisé dans certains des pays développés et non développés. Il devrait avoir des débits de données plus élevés et une efficacité spectrale élevée. Pour atteindre ce débit de données, la sélection d'un schéma de modulation multi-porteuse est nécessaire. Certaines des techniques utilisées sont FDMA, TDMA, CDMA, WCDMA, OFDMA, etc. Parmi elles, OFDM est la dernière technique qui a de nombreuses applications telles que la diffusion audio numérique (DAB), la diffusion vidéo numérique (DVB) et pour les liaisons filaires ADSL (Digital Subscriber Line) à haut débit. Ainsi cette technique a connu un vif succès ces dernières années et est en phase de normalisation dans différents standards sans fils (IEEE802.11a, WiMAX, LTE, DVB S2).et aussi a été validé par la release 16 pour l'interface NR (New radio) dans le système 5G avec l'utilisation de numérologie.

Les Signaux indépendants qui sont un sous-ensemble d'un signal principal multiplexé dans OFDM et le signal est d'abord divisé en canaux indépendants, les données sont modulées puis re-multiplexées pour créer OFDMTransporteur. L'orthogonalité des sous-porteuses est le concept principal de l'OFDM. Il permet la transmission simultanée de nombreuses sous-porteuses dans un espace de fréquence restreint sans interférence les unes des autres. Cela constitue un avantage indéniable dans l'OFDM.

Un des problèmes majeurs dans la télécommunication sans fils est la sélection aléatoire des canaux sans dont ils peuvent être sélectifs en fréquence, et ça nous endommage le signal transmit ; donc une mauvaise transmission qui va être corrigé par l'utilisation de la technique OFDMLTE.

La technique OFDM LTE est adaptésà la technique de modulation multi-porteuse OFDM et le multi accès pour former la technique OFDMA qui nous présente plusieurs avantages y compris sa robustesse contre l'évanouissement et l'interférence.

1.2 Principe OFDM [1]

La modulation multi porteuse OFDM consiste à répartir les symboles sur un grand nombre de porteuses à bas débit. A l'opposé des systèmes conventionnels qui transmettent les symboles en série, chaque symbole occupe toute la bande passante disponible. Alors on repartit de manière aléatoire des symboles de durée T_u (temps symbole utile) sur des différentes porteuses (ou sous-porteuses) modulées en QPSK ou QAM (selon le compromis robustesse/ débit).

Dans le système OFDM le canal est divisé en cellule suivant les axes de temps et de fréquence. Le canal est désormais constitué de suite de segments de fréquence et de suite de segments temporels. Une porteuse est affectée à chaque cellule fréquence/temps et l'information à transporter est repartisur l'ensemble de ces porteuses modulées, chacune suivant le type BPSK, QPSK ou QAM. Un symbole OFDM comprend alors l'ensemble des informations contenues dans l'ensemble des porteuses à un instant t . Pour que les fréquences des porteuses soient les plus proches possibles et pour ainsi transmettre le maximum d'information sur une portion de fréquences donnée, l'OFDM utilise l'orthogonalité, cité précédemment. Les signaux des différentes porteuses se chevauchent mais grâce à l'orthogonalité n'interfèrent pas entre elles. Ainsi, pour un train de symboles initial de période T_i , les symboles seront répartis en N trains plus lents et auront alors une durée $T = NT_i$. Pour répartir les données à transmettre sur les N porteuses, on regroupe les symboles c_k par paquets de N . Les c_k sont des nombres complexes définis à partir des éléments binaires par une constellation (mapping) souvent de modulation QAM ou PSK à 2^n états.

La séquence de N symboles c_0, c_1, \dots, c_{N-1} constitue un symbole OFDM. Chaque donnée c_k module un signal à la fréquence f_k .

Le signal individuel s'écrit sous forme complexe :

$$c_k e^{j2\pi f_k t} \tag{1.1}$$

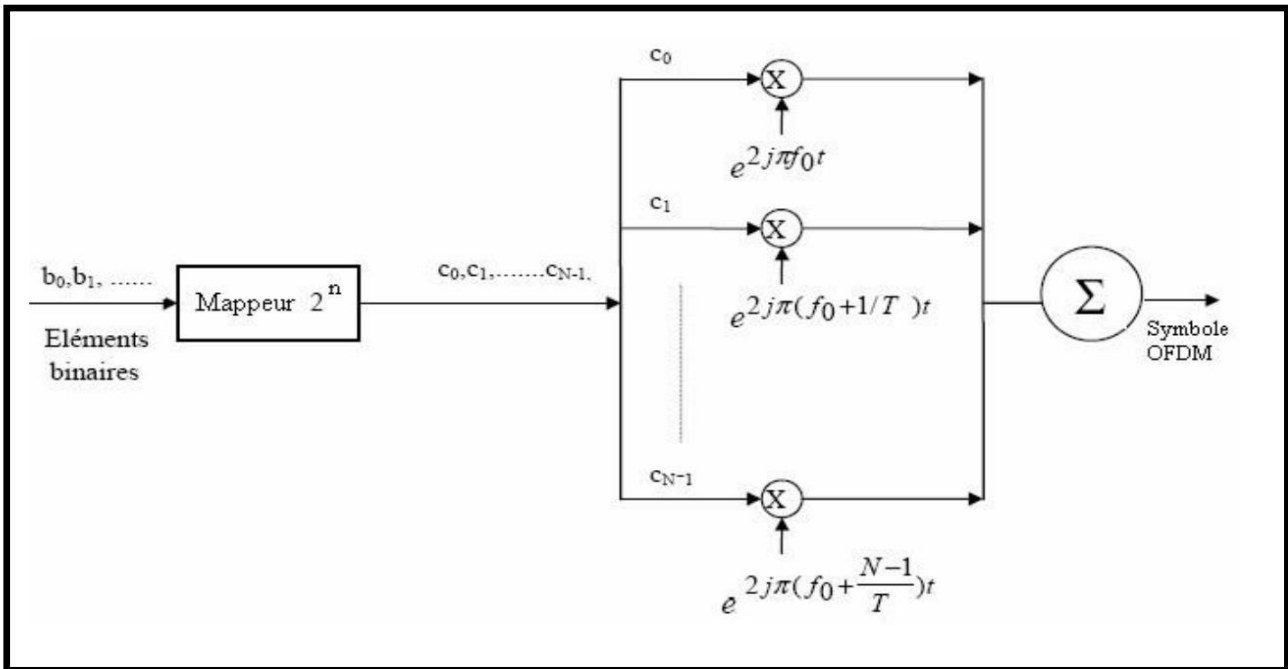


Figure 1.1 : Schéma de principe d'un modulateur OFDM

La figure 1.2 montre que l'espace entre chaque sous-porteuse $1/T$ permet, lorsque le spectre d'une sous-porteuse est maximal, d'annuler le spectre de toutes les autres : c'est la condition d'orthogonalité.

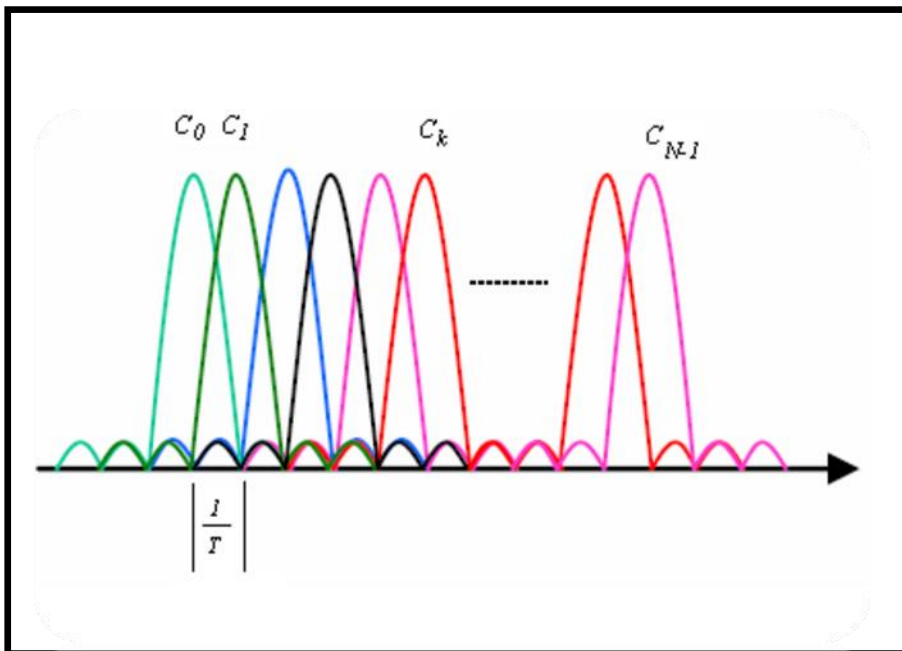


Figure 1.2 : Spectre du signal OFDM avec N porteuses

Pour que le signal modulé ait une grande efficacité spectrale, il faut que les fréquences des porteuses soient les plus proches possibles, tout en garantissant que le récepteur soit

capable de les séparer et retrouver le symbole numérique émis sur chacune d'entre elles. Ceci est vérifié si le spectre d'une porteuse est nul aux fréquences des autres porteuses.

1.3 Implémentation numérique de la technique OFDM

L'implémentation numérique de la technique OFDM est très simple, les traitements nécessaires font appel à des blocs de conversion S/P (série-parallèle) et P/S (parallèle-série), à des blocs FFT et IFFT, et aux convertisseurs CAN et CNA.

1.3.1 Implémentation numérique du modulateur OFDM

Le signal $s(t)$ à la sortie du modulateur peut se mettre sous la forme suivante :

$$s_i(t) = \sum_{n=-N/2}^{N/2-1} X_i \cdot n e^{j2\pi \frac{nt}{T}} \tag{1.2}$$

Ce signal est constitué de deux parties :

Une partie bande de base :
$$W_n = \sum_{k=0}^{N-1} c_k e^{j2\pi \frac{kn}{N}} \tag{1.3}$$

$e^{j2\pi f_0 t}$ Et une partie de transposition en fréquence

En discrétisant le signal $w(t)$ à la période T_i ($T=NT_i$), on obtient une sortie w_n sous la

forme :
$$w(t) = \sum_{K=0}^{N-1} c_K e^{j2\pi K \frac{t}{T}} \tag{1.4}$$

Les w_n sont donc obtenus par une transformée de Fourier inverse discrète des c_k , de sorte que le calcul peut se faire par simple IFFT si le nombre de porteuses N est choisi comme puissance de 2, conduisant au schéma numérique suivant :

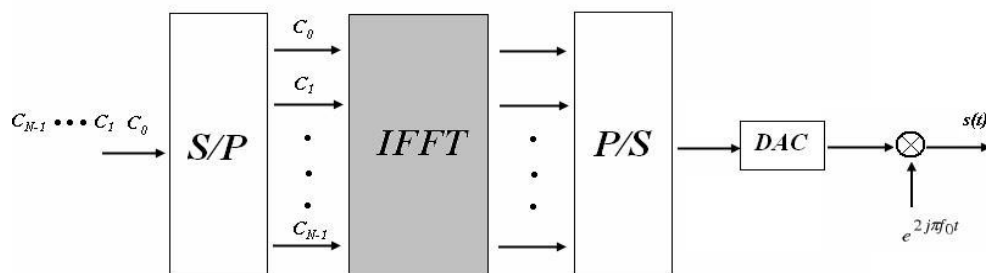


Figure.1. 3 : Modulateur OFDM numérique

1.3.2 Implémentation numérique du démodulateur OFDM

Le signal à l'entrée du récepteur peut se mettre sous la forme suivante :

$$r(t) = e^{j2\pi f_0 t} \cdot \sum_{k=0}^{N-1} c_k H_k e^{j2\pi \frac{k}{T} t} \quad (1.5)$$

Le signal bande base discrétisé est alors :

$$Z_n = \sum_{k=0}^{N-1} c_k H_k e^{j2\pi \frac{kn}{N}} \quad (1.6)$$

Z_n est la transformée de Fourier discrète inverse de $c_k H_k$, la démodulation consiste donc à effectuer une transformée de Fourier directe discrète, qui se fera par simple FFT. (Fig.1.4)

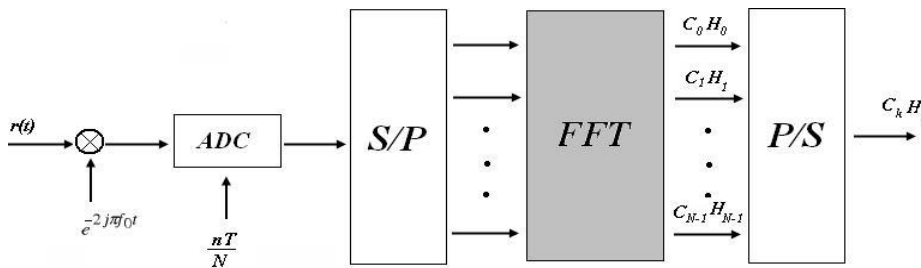


Figure 1.4 : Démodulateur OFDM numérique

1.4 Canal multi-trajet

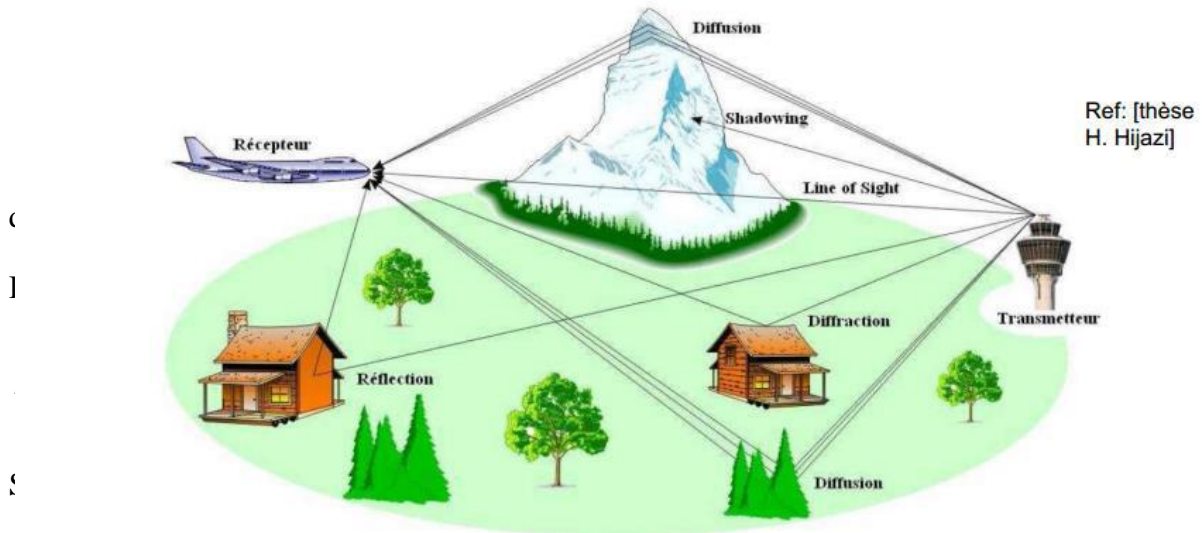
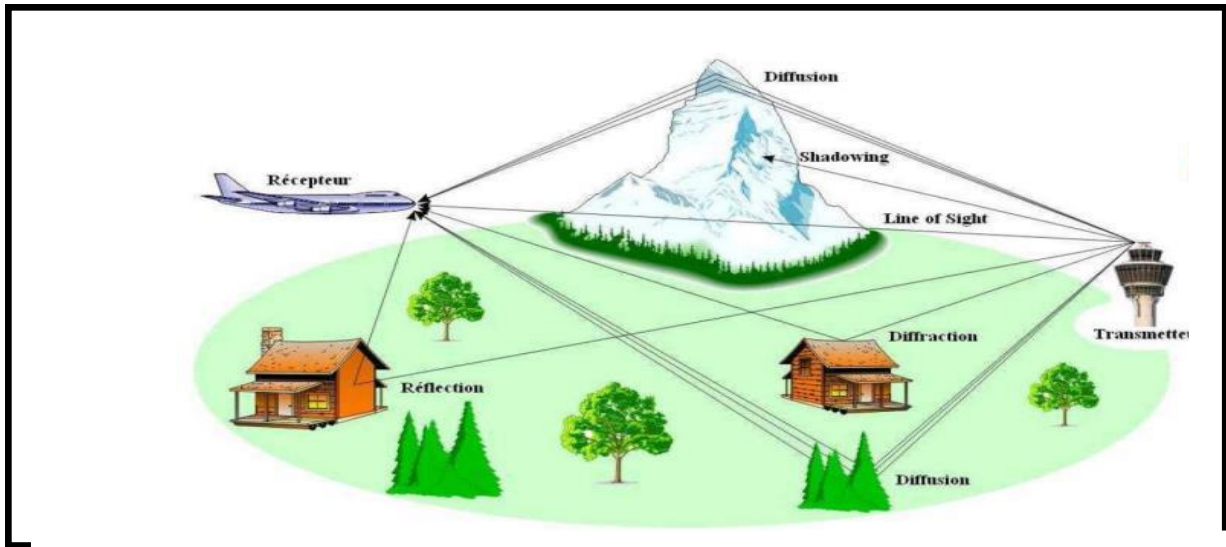
Après d'exposer le principe des OFDM, nous allons donner un Aperçu et quelques caractéristiques du canal radio mobile pour lequel ce type de modulations est intéressant.

1.4.1 Aperçu sur le canal multi trajet

Dans un système de communication sans fil, les signaux de transmission s'interfèrent les uns avec les autres à cause de différents mécanismes. Le récepteur peut recevoir un signal sous plusieurs versions à travers des trajets multiples. On appelle ce phénomène l'effet multi-trajet.

En effet, il existe cinq phénomènes principaux: la réflexion, la réfraction, la diffraction, la diffusion et le guidage d'onde qui ont un impact direct sur la propagation du signal. Les ondes émises lors de la communication radio subissent généralement une combinaison de ces divers phénomènes. Par conséquent, le signal reçu est une somme de tous les signaux arrivants au récepteur sur les différents trajets, et cette somme peut s'effectuer d'une manière

constructive ou destructive. Les obstacles peuvent être considérés comme un avantage ou un inconvénient pour la transmission [2].



$$H(f) = \sum_{i=1}^I h_i e^{-2j\pi f \tau_i} \quad (1.8)$$

1.4.2 Caractéristique d'un canal à trajet multiple

L'évaluation des systèmes de transmission est faite selon deux caractéristiques : la qualité de transmission et la complexité de calcul liée aux opérations de modulation/démodulation. Les grandeurs permettant de quantifier la qualité de la transmission sont :

- Le taux d'erreur binaire (TEB): permet de mesurer la fréquence à laquelle les erreurs se produisent, il correspond au rapport entre le nombre de bits erronés et le nombre total des bits émis.

- Erreur quadratique moyenne (EQM): détermine l'écart moyen entre les symboles émis et les symboles reçus.
- L'efficacité spectrale: mesure le débit binaire par unité de fréquence pour une transmission de 'q' bits sur une durée T_s et une largeur de bande B allouée à la transmission. Le débit binaire étant donné par le rapport q/T_s , l'efficacité spectrale est exprimée donc par le rapport $q/B T_s$.
- Le rapport signal sur bruit (RSB): est généralement adopté en transmission numérique comme paramètre d'entrée du récepteur pour lequel on va évaluer la qualité du message numérique restitué, il permet ainsi de qualifier la sensibilité du récepteur aux perturbations subies par le signal lors de propagation dans le canal. Le RSB est déterminé par le rapport E_b/N_0 avec N_0 la densité spectrale de puissance du bruit blanc à l'entrée du récepteur et E_b l'énergie moyenne par bit du signal modulé. [4]
- La bande de cohérence: est une mesure statistique de la bande de fréquence dans laquelle le canal peut être considéré comme plat (non sélectif).[5]

1.5 Chaîne de transmission pour OFDM LTE

1.5.1 Accès radio LTE[6]

Pour offrir des débits élevés le LTE emploie la technologie OFDMA (Orthogonal Frequency Division Multiple Access) dans le sens descendant, et le SC-FDMA (Single Carrier-Frequency Division Multiple Access) dans le sens montant. Le LTE respecte les délais requis par le trafic temps-réel. Le LTE respecte les délais requis par le trafic temps-réel. Cette technologie prend en charge la mobilité des utilisateurs en exécutant le Handover à une vitesse allant jusqu'à 350 km/h.

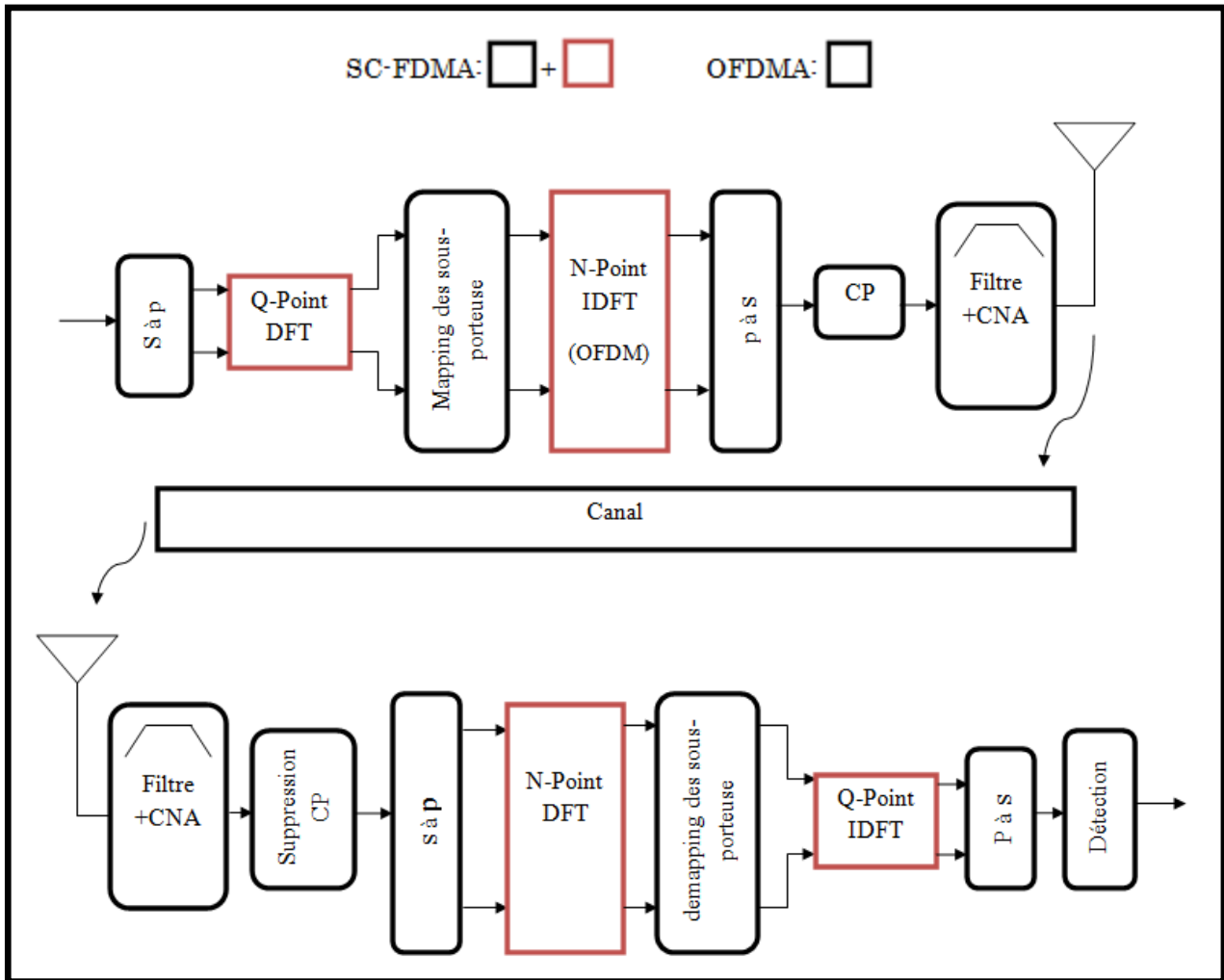


Figure 1.6 : Émetteur / Récepteur des systèmes OFDMA et SC-FDMA.

Le principe de l'OFDM (Orthogonal Frequency Division Multiplexing) c'est la base du LTE et de son interface Air. L'OFDM consiste à répartir sur un grand nombre de sous-porteuses le signal numérique que l'on veut transmettre.

Pour que les fréquences des sous-porteuses soient les plus proches possibles et ainsi transmettre le maximum d'information sur une portion de fréquence donnée, l'OFDM utilise des sous-porteuses orthogonales entre elles. Les signaux des différentes sous-porteuses se chevauchent mais grâce à l'orthogonalité n'interfèrent pas entre elles.

Cette technique éprouvée maintenant a été précédemment utilisée sur des lignes fixes en ADSL ou au niveau des modems câblées, ainsi qu'en radio pour les transmissions satellites. Le schéma ci-dessous montre une représentation temps/fréquence de l'OFDM.

- en fréquence, la représentation des sous porteuses orthogonales entre elles

- en temps, le symbole OFDM encodé sur x micros secondes et un intervalle de garde ou rien n'est émis pour éviter les interférences inter-symboles.

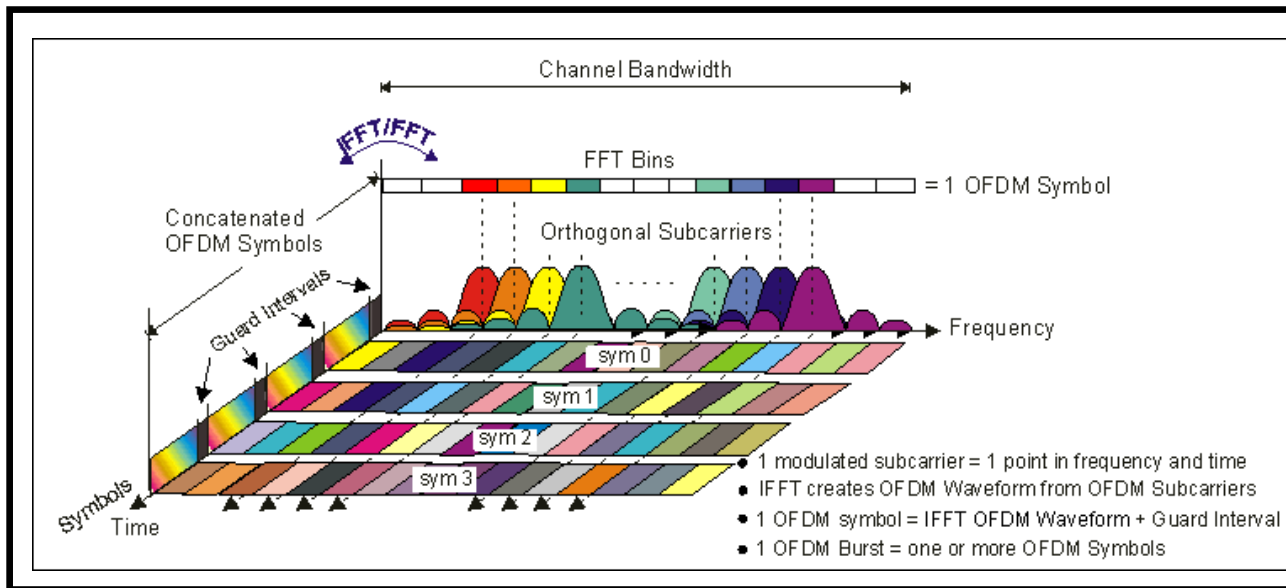


Figure 1.7 : Représentation fréquentielle et temporelle d'un signal OFDM.

1.6 Techniques de transmission OFDMA

En LTE, la technique de transmission utilisée est basée sur l'OFDM (Orthogonal Frequency Division Multiplexing) et plus précisément l'OFDMA sur la voie descendante, et le SC-FDMA sur la voie montante pour le support de l'accès multiple sur le réseau LTE 4G. Le signal OFDM est très simple à moduler et à démoduler ; il ne nécessite pas un traitement séparé par sous-porteuse comme cela peut être le cas en FDM. Le signal de la figure 1.4 peut en effet être facilement obtenu par une transformée de Fourier rapide inverse ou IFFT (Inverse Fast Fourier Transform). De la même manière, les sous-porteuses peuvent être démodulées par une transformée de Fourier rapide ou FFT (Fast Fourier Transform). IFFT et FFT sont d'implémentations très simples et rapides, notamment lorsque le Nombre de sous-porteuses est une puissance de deux [7].

La figure 1.8 montre une chaîne de transmission et de réception OFDM simplifiée : les sous-porteuses sont modulées et démodulées conjointement. Chacune d'entre elles peut transmettre un symbole issu d'une constellation QAM, ce qui rend l'allocation des ressources très flexible.

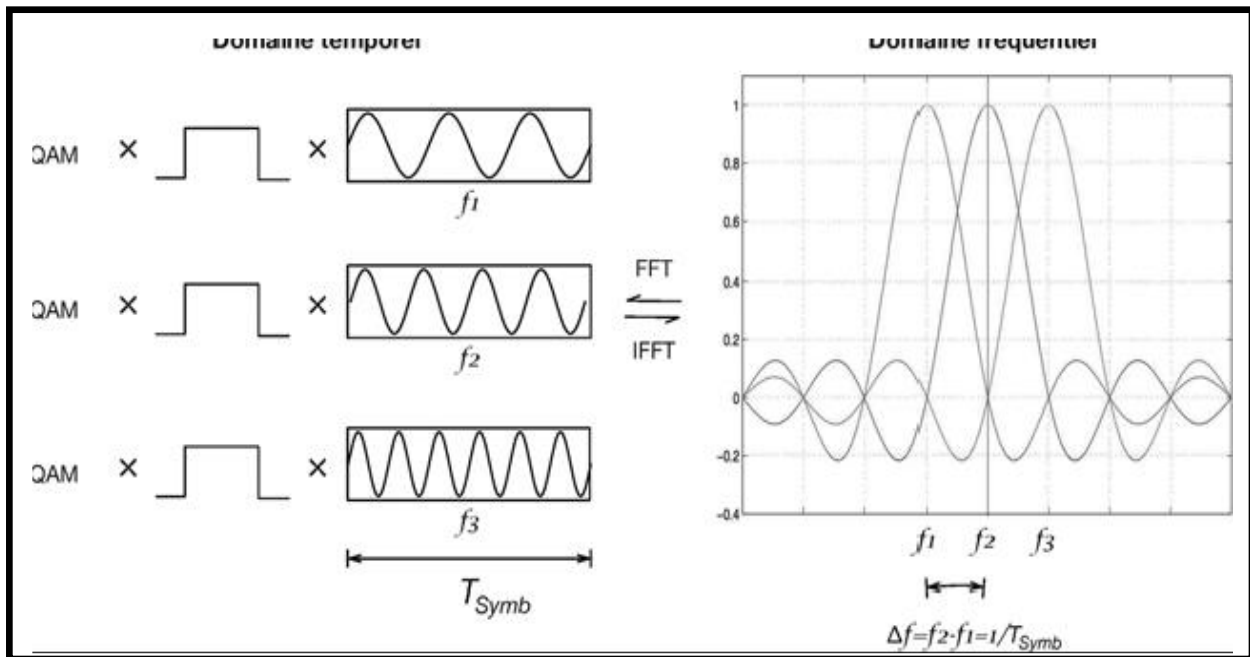


Figure 1.8 : Signal OFDM dans les domaines temporel et fréquentiel

En LTE, les symboles OFDM peuvent être encodés en utilisant 4 modulations différentes, suivant le niveau de protection que l'on veut apporter aux informations à transmettre.

- BPSK, qui permet d'encoder 1 bit. Elle est utilisée que pour encoder des informations de contrôles.
- QPSK, qui permet d'encoder 2 bits.
- 16QAM, qui permet d'encoder 4 bits.
- 64QAM, qui permet d'encoder 6 bits.

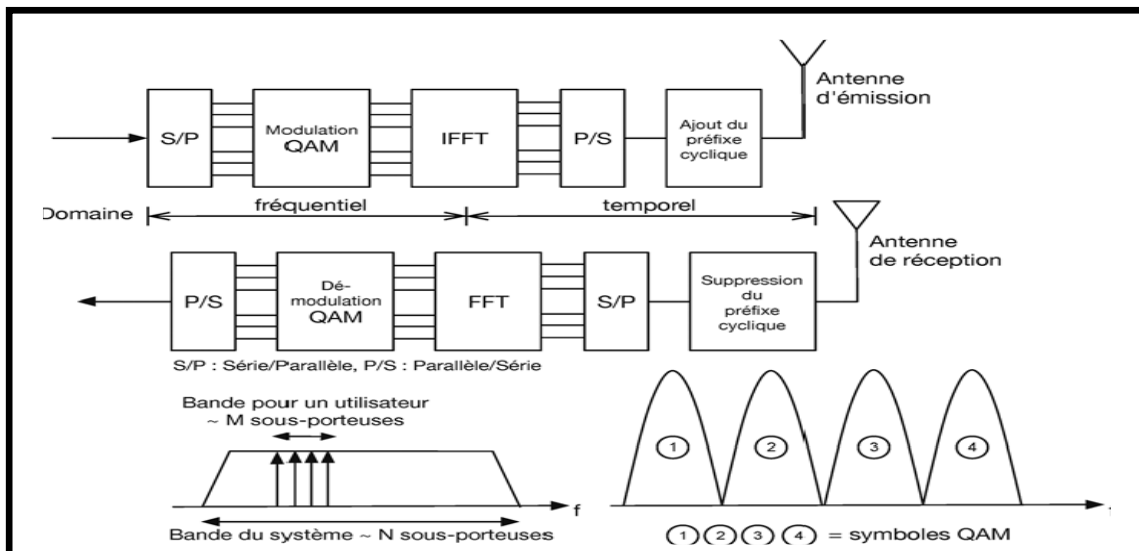


Figure 1.9 : Chaîne de transmission et de réception SC-FDMA.

1.7 Technique de transmission SC-FDMA

L'OFDM a pour grand inconvénient un fort PAPR. Sur la voie descendante, ce défaut est amoindri car l'amplificateur de puissance de l'eNode-B est de bonne qualité et la station de base dispose d'une grande puissance. C'est en revanche plus problématique sur la voie montante car l'amplificateur des terminaux doit rester à un prix raisonnable. En outre, il faut préserver l'autonomie des batteries sans trop affecter la couverture en bordure de cellule. La norme a donc prévu une technique de transmission avec PAPR réduit sur la voie montante, le SC-FDMA. Les chaînes de transmission et de réception du SC-FDMA sont assez semblables à celles de l'OFDM/OFDMA. Les différences sont indiquées en figure I6-(partie grisée).

Après la modulation QAM, les symboles générés par un utilisateur sont convertis dans le domaine fréquentiel par une transformée de Fourier rapide à N points. À la sortie de cette FFT, les N échantillons retrouvent le domaine temporel grâce à une transformée de Fourier rapide inverse à N points comme en OFDM. Les $N - N_u$ points restants ont une entrée nulle.

Seule une sous-partie de l'ensemble des sous-porteuses est donc modulée, la partie restante de la bande pouvant être utilisée par d'autres terminaux. [7]

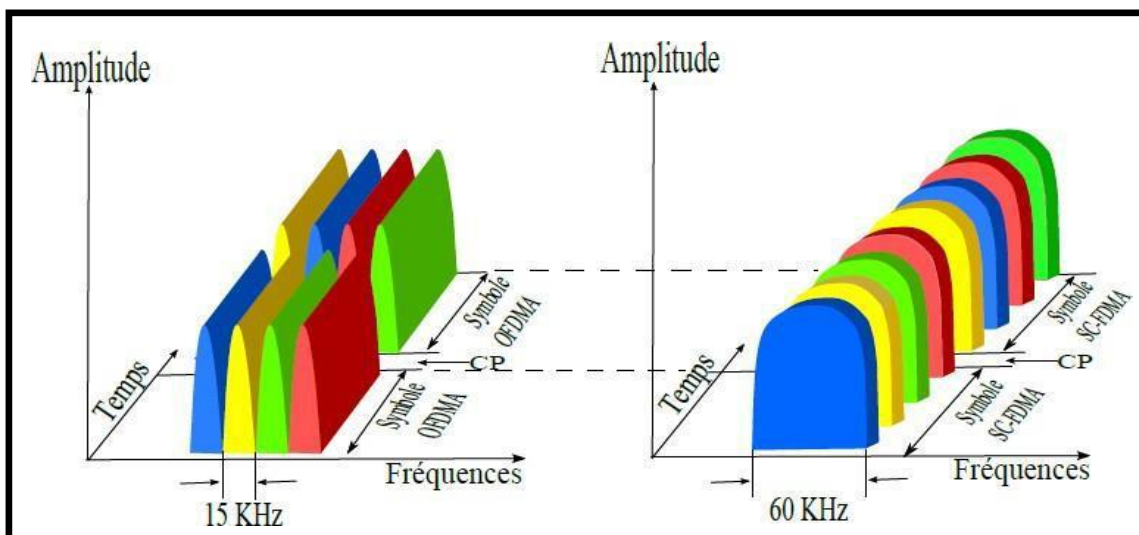


Figure.1.10 : de l'ensemble des sous-porteuses

1.8 Bilan de liaison

Le but du bilan de liaison effectué est d'avoir le rayon moyen de la cellule en UL/DL, d'avoir le seuil RSRP/RSRQ à rentrer pour la planification Asset et d'avoir le débit espérés par cellule et le débit à la bordure de la cellule. Pour ce faire, des paramètres de planification doivent être rentrés dans l'outil :

- Type de LTE : FDD ou TDD.
- Bande de fréquence.
- La largeur de bande utilisée.
- La probabilité de couverture.
- La catégorie du terminal.
- Le débit maximal ciblé.
- La puissance maximale de l'eNB.
- Le Noise Figure de l'eNodeB.
- RRH activé : ce qui correspond à une perte de feeder minime (0.4 dB).

1.9 Planification de fréquence

Puisque toutes les cellules LTE utilisent la même fréquence, de l'interférence se produit entre les UE au bord de la cellule. Dans ce cas, les performances des UE se détériorent. L'ICIC (Inter-Cell Interférence Coordination) peut être utilisée pour modifier la distribution de l'interférence, ce qui améliore le débit des UE au bord d'une cellule.

Dans les systèmes LTE, les utilisateurs occupent des PRB à la fois pour la liaison descendante et la liaison montante. L'interférence inter-cellules peut être relativement élevée, car toutes les cellules voisines peuvent fournir des services sur toute la bande du système. En conséquence, l'interférence inter-cellulaire est importante pour les utilisateurs à la bordure de la cellule en particulier.

Pour augmenter le débit des utilisateurs à la bordure de la cellule et améliorer la couverture, l'interférence inter-cellules doit être minimisée. L'ICIC est une technique qui utilise le contrôle de puissance et le scheduling NB pour lutter contre l'interférence inter-cellule. L'ICIC divise l'ensemble de la bande du système en plusieurs bandes de fréquences et les utilise différemment, en particulier au niveau des zones de chevauchement entre les cellules voisines. Par conséquent, les utilisateurs à la limite de couverture de la cellule sont préférentiellement alloués dans la partie de la bande qui peut effectivement atténuer

l'interférence inter-cellules. L'ICIC, fonctionne comme une technique de «respiration de cellule», pour laquelle l'ensemble de la bande d'une cellule est ajusté lors de la détermination de l'allocation des différentes bandes à la limite de couverture d'une cellule.

Pour utiliser cette technique efficacement, les cellules voisines doivent échanger des informations sur l'interférence et les niveaux de charge par l'intermédiaire de l'interface X2. A chaque fois que les bandes à la limite de couverture de la cellule sont ajustées, la cellule doit informer ces voisins de l'opération pour que les dites voisins puissent à leurs tour ajuster leurs propres bandes.

1.10 Les Avantages et l'inconvénient de l'OFDM LTE

1.10.1 Les Avantages et l'inconvénient de l'OFDMA :(liaison montante)

Le principal est le suivant :

- Anti interférence multiphathes
- Anti-évanouissement sélectif de fréquence
- Efficacité spectrale plus élevée
- Facile à coopérer avec MIMO pour un débit plus élevé
- Planification flexible pour multi-utilisateurs

1.10.2 Les Avantages et l'inconvénient de l'OFDMA (liaison descendante)

Le principal est le suivant :

- économisez le coût terminal et la consommation d'énergie
- faible PAPR modulation technologie DFT - S-OFDM qui est similar à OFDM
- une efficacité spectrale plus élevée rivalisent avec la technologie simple traditionnelle de transporteur.

1.11 Comparaison entre technologie base sur l'OFDM

Plusieurs technologies de réseaux sans fil existent, les caractéristiques principales des différentes technologies sont la fréquence d'émission utilisée, la modulation, la puissance et la sensibilité du signal radio, le débit et la portée du réseau. Pour l'instant, les principales technologies des réseaux sans fil sont : LTE, HSI-OFDM, IEEE 802.11/Wifi, et IEEE 802.16/WIMAX. Puisque les différents organismes de normalisation et les différents constructeurs tentent chacun d'imposer leur technique, ces différentes technologies ne sont pas compatibles entre elles.

| System paramètre | LTE | Norme IEEE 802.11 | WIMAX | OFDM pour 60GHz (mode HSI-OFDM) | |
|--|----------|----------------------------|--------------------|---------------------------------|----------|
| Bande de base fréquence | 15.36MHz | 2-5GHz | 2-11GHz | 2.592GHz | |
| Bande passante | 10MHz | 22-160MHz | 1.2-20MHz | 1.815GHz | |
| FFT size (Taille FFT) | 1024 | 64 | 1024 | 512 | |
| CP size | 256 | 16 | 64 | 64 | |
| Débits | 100Mb/s | 54Mb/s | 75Mb/s | 30Mb/s | |
| Modulation QPSK | 16QAM | BPSK, QPSK, 16-QAM, 64-QAM | QPSK. 16QAM. 64QAM | BPSK , M-QAM | |
| Nombres de porteuses | 900 | 64 | 16 | 500 | |
| Espace entre sous porteuses | 15KHz | 312.5KHz | 10.94KHz | 5.0625MHz | |
| Durée des données utiles | 66.66µs | 3.2ps | 91.43 µs | 197.53ns | |
| Durée du préfixe cyclique (maximal) | 16.66µs | 0.8ps | 11.43 µs | 22ns | 50.18ns |
| Symbol duration (symbol+CP)durée totale du symbole | 83.32µs | 4ps | 102.86 µs | 219.53 ns | 247.71ns |
| IF fréquence d'échantillonnage | 61.44MHZ | 20MHz | 11.2MHz | 2.640GHz | |

| | | | | |
|-----------------------------------|--------------------|----------|----------|---------|
| La fréquence d'oscillateur | 15MHz | 16.61MHz | 10.94KHz | 2.33GHz |
| Mobilité supportée | >350Km /h | 70m | 120Km /h | 10m |
| Technologie d'accès | OFDMA , SC-FDMA | OFDMA | OFDMA | OFDMA |

Tableau I.1 : Comparaison entre LTE, IEEE 802.11, WIMAX et HSI-OFDM

Les réseaux sans fil sont classés de plusieurs manières, selon que nous nous intéressons à un critère ou à un autre (portée, débit, architecture, services...).

Le tableau ci-dessus présente une comparaison entre LTE, WIFI (IEEE 802.11), WIMAX et HSI-OFDM, cette comparaison nous amène à mieux comprendre et savoir les complémentarités de ces systèmes

1.12 Evolution pour implémentation LTE [8]

Le LTE n'est pas formellement une norme 4G car elle n'est pas entièrement conforme aux exigences de l'IMT Advanced 4G défini par l'UIT, mais elle est une étape vers la norme de 4e génération (LTE Advanced).

La technologie LTE (3.9 G) a été finalisée par la release 9 du 3GPP. Il est essentiellement consisté à Compléter les fonctionnalités de base introduites en Release 8 3GPP, et intégrer un certain nombre de corrections de la Release 8 profitant de l'expérience acquise par les constructeurs dans le cadre des premières implémentations matérielles. Les évolutions principales sont :

- Extension des techniques de transmission multi-antennes pour le TDD afin de permettre la transmission simultanée de deux blocs de données pour 8 antennes d'émission (Beamforming double couches)
- Définition de protocoles de localisation, notamment motivés par la législation des États-Unis qui impose de localiser les appels d'urgence.
- Définition d'une architecture et de protocoles autorisant des services de diffusion et d'envois multiples, aussi appelés MBMS (Multimedia Broadcast Multicast Service). Ces services permettent d'optimiser l'efficacité spectrale lors de la transmission d'un contenu commun à un groupe d'utilisateurs, comme de la télévision.
- Définition de nouvelles fonctionnalités d'auto-optimisation.
- Approfondissement des spécifications techniques notamment des HeNB les (Home eNodeB) dans domaines de la mobilité, de la sécurité et de l'architecture

Ils sont associés à des puissances de l'ordre d'une centaine de mW (20 dBm)

1.13 Conclusion

L'OFDM présente une technique très demandée et très utile dans le domaine sans fils à cause de son efficacité spectrale, sa combinaison des données et sa résistance contre les interférences.

Dans ce chapitre introductif, nous avons présenté d'une façon générale les principe de l'OFDM, les différentes technique de transmission utilisé dans le LTE et leur implémentation facile à l'aide de l'utilisation d'FFT/IFFT.

Ainsi, la mise en service d'une chaine de transmission LTE OFDM qui implique une évolution matérielle et logiciel s'ajoutant au réseau cellulaire pour l'accès au réseau LTE initial et a la 4G (LTE Advanced) par la suite .

A decorative orange scroll graphic with a gradient from light to dark orange. It has rounded corners and a vertical strip on the left side that extends downwards. There are three grey circular elements: one at the top left corner, one at the top right corner, and one at the bottom left corner of the scroll's main body.

Chapitre II :

Implementation HDL

2.1 Introduction

L'implémentation HDL a pour objectifs de faire la conception des circuits numériques. Les deux variétés HDL les plus largement utilisées, les mieux prises en charge dans l'industrie et les deux principaux langages de description du matériel sont Verilog et VHDL.

Un FPGA (Field Programmable Gate Array) est un circuit en silicium constitué de plusieurs blocs logique reprogrammable à l'aide d'un logiciel basé sur un langage nommé VHDL.

A chaque fois un FPGA est programmé, il adopte une nouvelle « personnalité » si on recompile une nouvelle configuration de circuits sans avoir besoin d'utiliser une maquette ou un fer à souder.

FPGA Il s'agit d'un circuit intégré qui peut être programmé sur le terrain pour fonctionner selon la conception prévue. Ceux-ci peuvent être utilisés pour effectuer des opérations logiques et arithmétiques, pour le stockage de variables et pour transférer des données entre différentes parties du système. Cela signifie qu'il peut fonctionner comme un microprocesseur, ou comme une unité de cryptage, ou une carte graphique, ou même les trois à la fois. FPGA est composé de milliers de blocs logiques configurables (CLB) intégrés dans un océan d'interconnexions programmables. Les CLB sont principalement constitués de tables de consultation (LUT), de multiplexeurs et de bascules. Ils peuvent implémenter des fonctions logiques complexes. Outre les CLB et les interconnexions de routage, de nombreux FPGA contiennent également des blocs de silicium dur dédiés pour diverses fonctions telles que le bloc RAM, les blocs DSP, les contrôleurs de mémoire externe, les PLL, les émetteurs-récepteurs multi-Gigabit, etc. Une tendance récente est de fournir un processeur en silicium dur core (comme ARM Cortex A9 dans le cas de Xilinx Zynq) à l'intérieur du même FPGA die lui-même afin que le processeur puisse prendre en charge des tâches banales et non critiques, tandis que FPGA peut prendre en charge une accélération à grande vitesse qui ne peut pas être effectuée à l'aide de processeurs. Ces blocs matériels dédiés sont essentiels pour concurrencer les ASIC (Application Specific Integrated Circuits). Les ASIC sont spécifiques à l'application. Par exemple, le processeur à l'intérieur d'un téléphone mobile est un ASIC. Il est conçu pour fonctionner comme un processeur pendant toute sa durée de vie. Sa fonction logique ne peut être changée en rien d'autre car son circuit numérique est composé de portes et de bascules connectées en permanence en silicium.

Le système générateur ainsi la génération du code VHDL à partir de MATLAB / Simulink vers Xilinx devient le plus court chemin pour une implémentation FPGA.

A nos jours, de nombreuses entreprises utilisent des FPGA pendant la phase de conception initiale et les phases de pré-production, puis passent plus tard à l'ASIC pour la production en série. Pour les applications dont le futur succès commercial est inconnu, la voie FPGA présente un risque moindre.

2.2 Langages de programmation HDL [9]

Un langage de description de matériel (Hardware Description Language, HDL) est une instance d'une classe de langage informatique ayant pour but la description formelle d'un système électronique.

Le but des HDL est double (la simulation et la synthèse) :

- **La simulation** : L'un des objectifs des HDL est d'aboutir à une représentation exécutable d'un circuit, soit sous forme autonome, soit à l'aide d'un programme externe appelé simulateur. Cette forme exécutable comporte une description du circuit à simuler, un générateur de stimuli (vecteurs de test), ainsi que le dispositif implémentant la sémantique du langage et l'écoulement du temps. Il existe deux types de simulateurs, à temps discret, généralement pour le numérique, et à temps continu pour l'analogique. Des HDL existent pour ces deux types de simulations.
- **La synthèse** : En n'utilisant qu'un sous-ensemble d'un HDL, un programme spécial appelé synthétiseur peut transformer une description de circuit en une Netlist de portes logiques ayant le même comportement que le circuit de départ. Le sous-ensemble du langage utilisé à ce propos est alors dit synthétisable. La sémantique synthétisable ignore typiquement toutes les constructions ayant un rapport avec le temps.

Il existe un grand nombre de Langages de programmation HDL, les principaux étant :

- Verilog.
- VHDL.
- SystemC, sur-ensemble de C++ possédant des classes spéciales pour modéliser le matériel, et incluant un moteur de simulation. Pour l'instant, sa synthèse n'est pas facile.
- Confluence, langage déclaratif GPL pouvant générer du Verilog, du VHDL, des modèles exécutables en C et des modèles de vérification formelle.
- ABEL ("Advanced Boolean Expression Language", un langage propriétaire développé par Data I/O Corporation, maintenant possédé par Lattice).
- AHDL (Altera HDL, langage propriétaire d'Altera pour la programmation de leurs FPGA).
- CUPL (langage propriétaire de LogicalDevices, Inc.).
- VHDL-AMS : extension de VHDL destinée aux circuits analogiques ou mixtes.

2.3 Le FPGA (Field Programmable Gate Arrays) [10]

FPGA est un circuit intégré composé d'un réseau de cellules programmables. Chaque cellule est capable de réaliser une fonction désirée.

Le tableau suivant répertorie les domaines d'application courants des FPGA aujourd'hui :

| Marchés finaux | Sous-segments | Application |
|-----------------------------------|-------------------|------------------------------|
| Les communications | Sans fil | Stations de base cellulaires |
| | | LAN sans fil |
| | La mise en réseau | Réseaux métropolitains |
| | | Réseaux optiques |
| | | Modems DSL |
| | | Commutateurs |
| | | Les routeurs |
| | | Stockage de masse |
| Espace de rangement | | Réseaux de stockage |
| | | Stockage en Réseau |
| | | Serveurs haute vitesse |
| | | Périphériques d'ordinateur |
| | | Stockage de masse |
| Bureautique | | Copieurs, imprimantes |
| Consommateur, Industriel Et autre | Les consommateurs | Écrans plasma |
| | | DVR |
| | | Décodeurs |

| | | |
|--|------------|------------------------------------|
| | | Lecteur MP3 |
| | | Caméras digitales |
| | Industriel | Automatisation d'usine |
| | | L'imagerie médicale |
| | | Équipement de test |
| | Automobile | Systèmes multimédias |
| | | Systèmes de navigation GPS |
| | | Reconnaissance vocale |
| | Militaire | Surveillance par satellite |
| | | Communication sécurisée du système |
| | | radar et sonar |

Tableau2.1 : Principaux domaines d'application FPGA

La concurrence sur le marché des FPGA est amère. La raison devient plus qu'évidente quand on regarde les chiffres de marché actuels des principaux acteurs : Xilinx, Altera, Lattice et Actel.

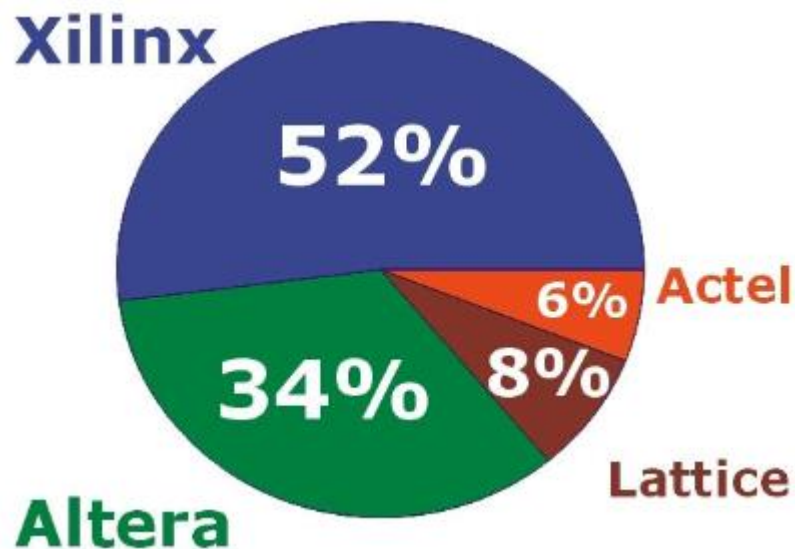


Figure 2.1 : Part de marché PLD (dispositifs logiques programmables)

Les deux sociétés proposent une vaste gamme de FPGA, avec des capacités à peu près équivalentes, ainsi que des kits de développement logiciel complets, ainsi que de nombreux blocs IP («propriété intellectuelle») pour les besoins spécifiques d'autres entreprises

| | Altera | Xilinx |
|---------------------------|---------|-----------|
| FPGA haut de gammefamille | Stratix | Virtex |
| FPGA bas de gammefamille | Cyclone | spartiate |

Tableau 2.2 : Familles de conception FPGA

2.3.1 Facteurs clés pour comparer les FPGA

Avant de pouvoir comparer les FPGA réels, il est important de visualiser les facteurs clés qui distinguent un FPGA d'un autre:

- Processus de fabrication
- Densité logique
- Gestion de l'horloge
- Mémoire sur puce
- Capacités DSP
- Compatibilité E / S
- Support logiciel et autres services de conception

2.3.2 Description des familles FPGA

| | Virtex-II 1000 | Virtex-II 3000 | Sparta n-3 1000 | Spartan -3 2000 | Virtex -5 LX30 | Virtex -5 LX50 | Virtex -5 LX85 | Virtex -5 LX110 |
|---------------------------|---------------------------|---------------------------|--------------------------------|--------------------------------|-------------------------------|-------------------------------|-------------------------------|--------------------------------|
| Nombre de portes | 1million | 3millions | 1 million | 2 millions | ----- | ----- | ----- | ----- |
| Bascules Flip-Flop | 10240 | 28672 | 15360 | 40960 | 19200 | 28800 | 51840 | 69120 |
| IUT | 10240 | 28672 | 15360 | 40960 | 19200 | 28800 | 51840 | 69120 |
| Multiplicateur | 40 | 96 | 24 | 40 | 32 | 48 | 48 | 64 |
| Blocs de RAM(kbit) | 720 | 1728 | 432 | 720 | 1152 | 1728 | 3456 | 4608 |

Tableau2.3 : comparaison des différentes familles de FPGA de Xilinx

Le tableau de comparaison ci-dessus montre les spécifications de différentes familles de FPGA de Xilinx. Le nombre de portes est un moyen habituel de comparer la taille des circuits FPGA par rapport à la technologie ASIC ; cependant ce critère ne donne pas vraiment le nombre de composants qui constituent un FPGA. C'est notamment pourquoi Xilinx n'a pas précisé le nombre de portes pour sa nouvelle famille Virtex-5. Pour obtenir davantage d'informations sur le fonctionnement des FPGA.

2.4 Procédure d'implémentations et de Vérification HDL [11]

La proposition d'un programme HDL pour la résolution et l'accélération d'un problème donné (accélérateur matériel). Dans ce cas plusieurs outils et plusieurs étapes sont nécessaires :

- ✓ Le choix du logiciel et de la carte Xilinx ;
- ✓ La saisie du code HDL dans un éditeur de texte
- ✓ L'ajout d'un testbench pour assurer des entrées/sorties synthétiques
- ✓ Simulation du fichier testbench et vérification fonctionnelle du code HDL
- ✓ Synthèse des codes pour la carte Xilinx choisie

- ✓ L'ajout des contraintes (affectation des pins = fichier user file .ucf)
- ✓ Génération des fichiers nécessaires pour la configuration du circuit FPGA (NGC, NDD, ... BIT)
- ✓ Transfert du fichier de configuration vers le circuit FPGA en utilisant un outil de configuration

Les logiciels proposés par XILINX : Xilinx propose plusieurs outils pour le développement des codes HDL :

- ✓ L'outil ISE pour le développement des accélérateurs
- ✓ L'outil ISE-simulator pour la simulation
- ✓ L'outil ISE-XST pour la synthèse
- ✓ L'outil EDK (ou bien XPS pour Xilinx Platform Studio) pour la conception des SoC autour des processeurs MicroBlaze de chez Xilinx
- ✓ Plus plusieurs autres outils pour la vérification et la post-vérification (PleanAhead, ChipScop, ...etc).

Dans ces dernières années, la société Xilinx fournit en parallèle avec les plateformes et les circuits FPGA un nouveau outil nommé Vivado qui regroupe tous les autres outils.

2.5 Architecture des circuits FPGA

Un FPGA est un réseau (matrice) de blocs combinatoires et séquentiels (CLB).

- Des blocs d'entrée/sortie (IOB) sont associés aux broches du circuit.
- Les CLB et IOB sont interconnectés entre eux par des dispositifs variés.
- Les matrices s'organisent de 8x8 à 128x120. [13]

2.5.1 Les Interconnexions FPGA

Il existe trois types d'interconnexions entre les différents blocs des circuits FPGA

- Interconnexion directe : entre les différents blocs logiques.
- Interconnexions par le biais d'une matrice.
- Interconnexion par les grandes lignes relient tous les CLB dans les extrémités des circuits FPGA.

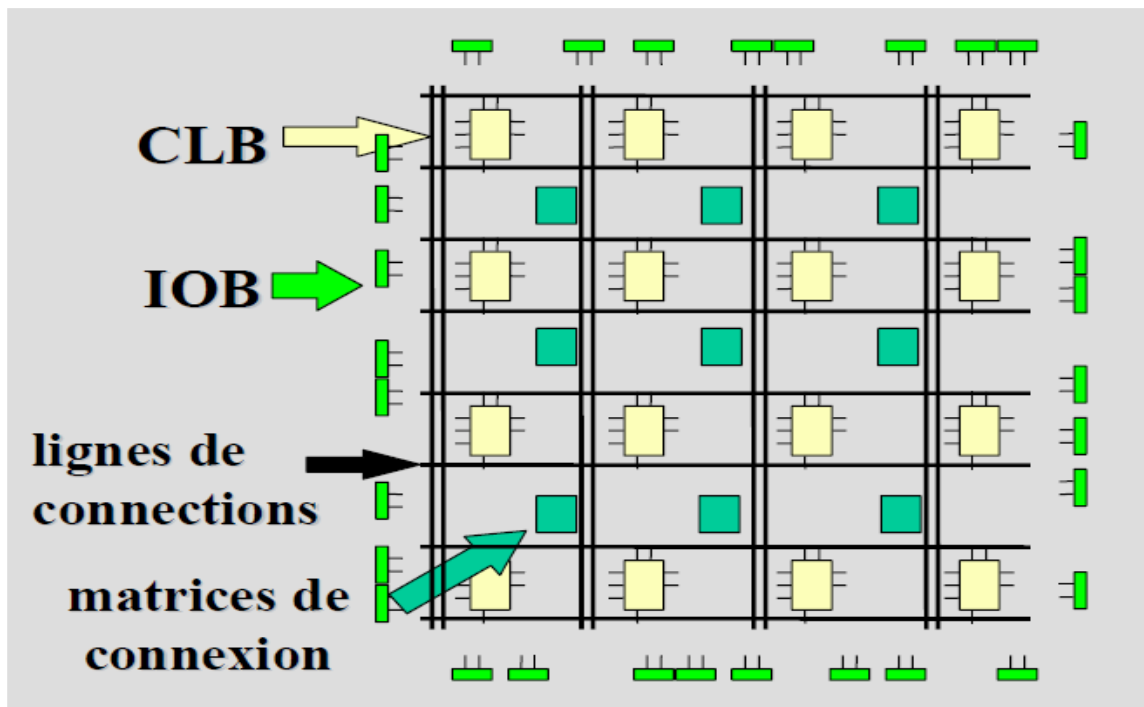


Figure 2.2 : Architecture générale d'un FPGA

2.6 Principe du VHDL

2.6.1 Historique [13]

Le langage VHDL a été commandé dans les années 1980 par le Département de la Défense des États-Unis dans le cadre de l'initiative VHSIC. Dans un effort de rationalisation, le VHDL reprend la même syntaxe que celle utilisée par le langage Ada (ce dernier étant aussi développé par le département de la défense).

La version initiale de VHDL, standard IEEE 1076-1987, incluait un large éventail de types de données, numériques (entiers, réels), logiques (bits, booléens), caractères, temps, plus les tableaux de bits et chaînes de caractères.

L'un des principaux problèmes concernait le type bit. Celui-ci ne pouvant prendre que deux valeurs (0, 1), il était impossible de représenter les signaux de valeur inconnue ou encore les signaux en haute impédance, ainsi que la « force » d'un signal (faible, forte ou nulle). La norme IEEE 1164 définit le type `std_logic` avec 9 états possibles. Ceci a été adopté dans le VHDL-93 (seconde version de la norme IEEE 1076).

Afin de répondre aux différents problèmes de l'électronique, la norme VHDL a dû évoluer. L'IEEE Design Automation Standards Committee (DASC) a créé la norme IEEE 1076.1(1999), ou VHDL-AMS (VHDL-Analog and Mixed Systems)

Cette nouvelle norme est une extension de la norme IEEE 1076-1987 déjà existante. Elle permet la description et la simulation de circuits analogiques, numériques, et mixtes (analogique et numérique). Pour cela elle utilise en complément des instructions séquentielles et concurrentes un nouveau type d'instructions, dites « simultanées », et qui ont valeur d'équations. En pratique, de plus en plus de simulateurs implémentent cette extension. Par contre, les outils de synthèse analogique associés n'en sont encore qu'à leurs balbutiements.

2.6.2 Les caractéristiques du VHDL

- Standard (indépendant du logiciel ⇒ échange facile)
- Haut niveau d'abstraction (indépendant de la technologie)
- Méthodologies de conception diverses
- Outil complet (design, simulation, synthèse)

2.6.3 Structure du langage VHDL

Déclaration des bibliothèques

Toute description VHDL utilisée pour la synthèse a besoin de bibliothèques. Et plus particulièrement la bibliothèque IEEE 1164. Elle contient les définitions du type de signaux électroniques, fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques. Les bibliothèques sont spécifiées par le mot clé Library. Pour avoir accès à une partie de la bibliothèque on utilise l'énoncé use. On a donc accès dans l'entité Incrémenter à tous les types définis dans le package

Entité

Un bloc d'entité est le début de bloque de construction d'un VHDL, chaque conception a un seul bloque qui décrit l'interface des signaux dans et hors l'unité de conception. Une même entité peut être associée à plusieurs architectures différentes. Elle décrit alors une classe d'unités de conception qui présente au monde extérieur le même aspect, avec des fonctionnements internes éventuellement différents [15].

Architecture

Une fois l'entité définit, il faut décrire son fonctionnement, c'est ce que l'on fait avec l'architecture.

2.7 Implémentation d'un circuit numérique de communication

2.7.1 Implémentation XSG MATLAB

Dernièrement, des outils de programmation haut niveau de conception des systèmes numériques sur circuits FPGA. Matlab est aussi entrée dans ce domaine avec l'outil XilinxSystem Generator (XSG) ajouté à la bibliothèque Matlab-Simulink.

En utilisant cette solution, l'utilisation ne programme pas en HDL mais utilise des blocs configurables Simulink déjà codés en HDL. Matlab-Simulink assure les outils pour la connexion des blocs ainsi que les outils pour la génération des signaux de test et pour l'affichage des résultats.

Cette solution nous permet d'éviter les détails de la programmation HDL et facilite les modifications dans les propositions chose qu'on a utilisée dans cette mémoire.

Après la vérification et la simulation des modèles Simulink, Matlab utilise l'outil XSG pour l'auto-génération d'un code HDL globale pour la proposition. Nous pouvons passer directement vers le fichier NGC ou bien BIT à partir d'un modèle Matlab-Simulink. Ensuite nous utilisons les mêmes étapes et les mêmes outils que la première solution.

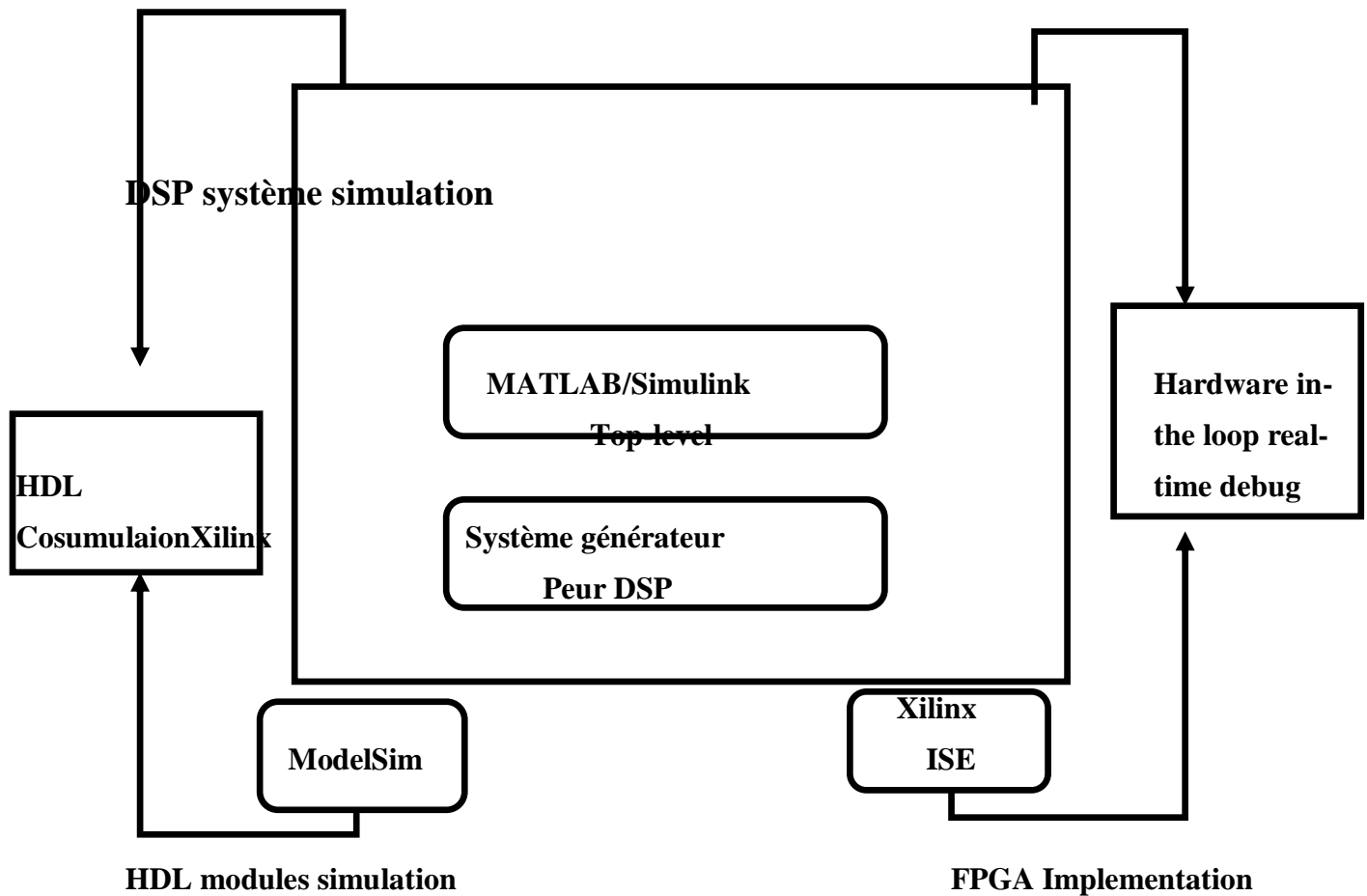


Figure 2.3 : Xilinx système generation pour symilink

2.8 Implémentation In the loop

Dernièrement, les ingénieurs de Mathworks ont ajoutés une nouvelle technique de programmation haute niveau nommée « FPGA in the loop ». Cette solution nous permet de simuler un modèle Matlab-Simulink sur FPGA sans passer par la programmation HDL. En plus elle permet l'implémentation de ce modèle directement sur le circuit FPGA. Ce modèle sera exécuté par le circuit FPGA et les autres modules Matlab-Simulink seront exécutés par le processeur du PC. A l'aide du technique « in the loop », Matlab-Simulink communique des données pour le modèle implémenté sur le circuit FPGA et récupère les données traitées.

En utilisant cette technique, nous pouvons profiter des avantages fournis par Matlab-Simulink (simulation et de Co-simulation faciles, possibilités de changement rapides des modèles, ...etc.) avec même la possibilité d'utiliser les circuits FPGA en temps réel. Nous pouvons aussi utiliser les autres outils de Matlab et Matlab-Simulink surtout concernant la connexion avec le monde extérieur à travers les ports d'entrées/sorties ainsi que les convertisseurs analogique.

2.9 Implémentation Xilinx[16]

Xilinx System Génération est un outil de programmation FPGA fourni par Xilinx. Xilinx - (prononcé "zylinks") a été fondée en 1984 et a expédié son premier produit commercial en 1985. Il est spécifiquement axé sur les FPGA Xilinx, permettant aux développeurs de travailler dans l'environnement Simulink et de générer des cœurs paramétrés particulièrement optimisés pour les FPGA Xilinx. L'outil est intégré à Xilinx ISE/ EDK Design Suite (System Edition) sont les anciens outils qui doivent être utilisés pour tous les appareils Virtex-6 et plus anciens et qui peuvent être utilisés pour certains appareils Virtex-7 de petite / moyenne taille par contre Xilinx Vivado HL (System Edition) est le nouvel outil qui ne prend en charge que Virtex-7, UltraScale et toutes les familles plus récentes.

Par défaut, le Xilinx Blockset contient plus de 90 blocs DSP, allant de simples additionneurs, multiplicateurs, etc. à des blocs complexes tels que des blocs de correction d'erreur directe, des FFT, des filtres et des mémoires, etc. Certains de ces blocs prennent également en charge le DSP à virgule flottante. La bibliothèque Floating-Point fournie par Xilinx répertorie ces blocs. De plus, le générateur de système inclut également les blocs mcode et Black Box, qui peuvent être utilisés pour intégrer respectivement les codes mcode et HDL directement dans l'environnement de conception Simulink.



Figure 2.4 : Carte FPGA

2.10 Simulations avec Simulink

Cet exemple traite des problèmes du monde réel associés à la mise en œuvre de la modulation et de la détection OFDM pour la génération de code HDL. Le modulateur comprend l'insertion de porteuse CC, l'insertion de préfixe cyclique et le fenêtrage tandis que le détecteur met en œuvre la récupération de fréquence, la détection du signal de synchronisation primaire (PSS) et du signal de synchronisation secondaire (SSS) pour déterminer l'identité de la cellule de la couche physique. LTE System Toolbox™ est utilisé pour vérifier la fonctionnalité des modèles HDL en fournissant un stimulus d'entrée et des formes d'onde de sortie de référence dorées. Les schémas du modulateur et du détecteur sont présentés ci-dessous.

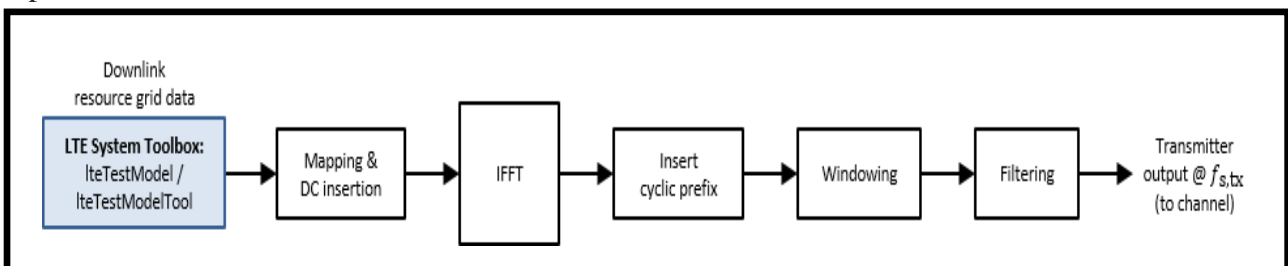


Figure 2.5 : LTE Structure du modulateur.

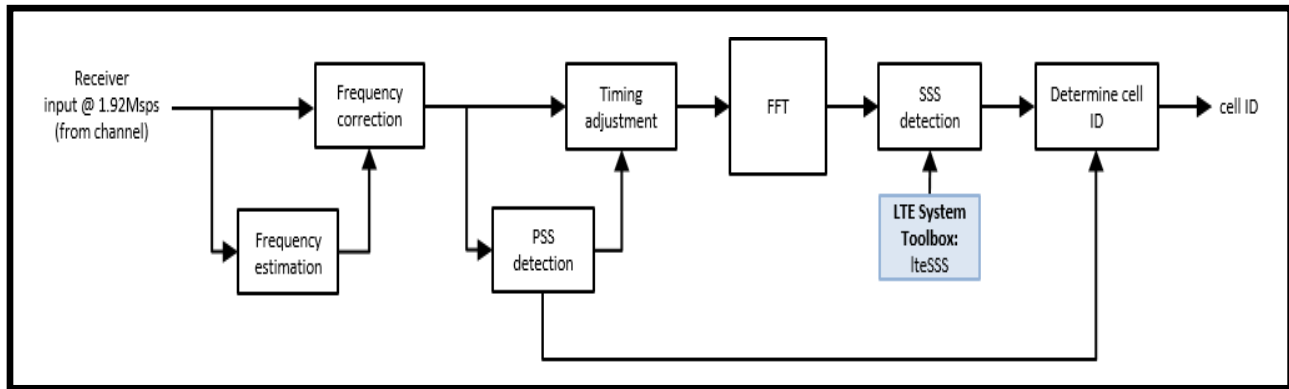


Figure 2.6 : Structure du détecteur LTE

Le sous-système HDL du modulateur LTE utilise une grille de ressources de liaison descendante (DL) à évolution à long terme (LTE) pré-générée qui est créée à l'aide de la boîte à outils du système LTE et exécute la modulation OFDM conformément à la norme LTE. Le sous-système du modulateur est paramétrable et prend en charge toutes les largeurs de bande de canal LTE standard. Pour vérifier la sortie de l'implémentation HDL, une référence dorée, une forme d'onde modulée OFDM est créée à l'aide des fonctions LTE System Toolbox `lteTestModel` et `lteTestModelTool`. La forme d'onde de référence dorée par défaut utilisée pour la vérification est générée conformément au modèle de test E-UTRA (E-TM) 1.1 et à une bande passante.

2.11 Conclusion

Après la présentation des différents langages et niveaux de description du matériel, ainsi que les types d'accélération matériel, nous avons choisi le HDL Coder avec un système générateur XSG basé sur Matlab/Simulink comme langage de programmation. Ce choix a été faite à cause de sa simplicité pour la description des processus et plus précisément le VHDL pour représenter le comportement ainsi que l'architecture d'un système électronique numérique, qui peut être vérifiée par simulation, avant que la conception détaillée ne soit terminée.

En fin nous avons choisi les outils de conception de Xilinx à cause de leurs disponibilités logiciel et matériel pour une implémentation hardware future, leurs flots de conception simples et rapides mais aussi à cause de l'utilisation des plateformes multi-composants de la même compagnie (Xilinx).



Chapitre III :

SIMULATION ET RESULTATS

3.1 Introduction

Après avoir finalisé la partie théorique, nous passons dans cette partie à traiter et implémenter la modulation et la détection OFDM LTE dans le but de montrer ses performances dans la communication sans fils et de générer le code HDL afin d'utiliser dans une carte FPGA dans un travail future.

Nous avons utilisé des blocs Matlab / Simulink pour la simulation de la technique OFDM LTE avec une structure d'implémentation HDL.

Nous nous sommes intéressés dans ce travail à préparer un code HDL du modulateur et des détecteur pour une implémentation matérielle sur FPGA.

Pour montrer l'implémentation HDL du modèle OFDM LTE, nous allons la comparer avec un signal de référence OFDM selon cinq paramètres qui sont la densité spectrale de puissance (PSD), le domaine temporel, l'estimation de fréquence, le diagramme de corrélation croisée PSS, le diagramme de corrélation croisée SSS et résultat final de simulation. Le deuxième résultat est les performances de l'implémentation en fonction de l'atténuation canal et.

Enfin, Le troisième résultat est la génération du code HDL à l'aide du codeur HDL pour la Vérification du système et leur implémentation sur la carte FPGA.

La simulation sera réalisée en faisant varier les paramètres de bande passante de 1.4Mhz jusqu'à 20Mhz avec la fixation du paramètre canal (SNR, Gain, Delay, fréquence offset) et autres.

Le canal de transmission étant un élément essentiel de tous systèmes de transmission sans fil, nous avons, alors, mené notre simulation en considérant plusieurs types de canal en variant le SNR et l'atténuation du signal pour faire conclure l'effet de canal sur notre implémentation HDL du système de communication OFDM LTE

3.2 Logiciels de simulation

MATLAB/ Simulink qui développé par MathWorks est le logiciel. Les utilisateurs peuvent analyser données, peut développer des algorithmes et créer des modèles mathématiques. Il offre une large gamme de domaines d'utilisation. MATLAB est l'outil le plus pratique pour analyser les signaux numériques.

Simulink a également été développé par MathWorks et intégré à MATLAB. Il est logiciel de programmation graphique qui offre une conception au niveau du système, une simulation, génération automatique de code et test et vérification continus des systèmes intégrés. Il prend également en charge la conception matérielle basée sur un modèle à l'aide du générateur de système (Xilinx). Dans ce projet nous utiliseront s la version MATLAB 2016b.

3.3 Simulation HDL implémentation OFDM LTE

3.3.1 Aperçu :

Le model utilisé dans notresimulation a une structure de niveau supérieur qui aborde les problèmes de la mise en œuvre de la modulation OFDM et de la détection pour la génération de code HDL. Le modulateur constitué de l'insertion de porteuse CC, l'insertion de préfixe cyclique et le fenêtrage tandis que le détecteur met en œuvre la récupération de fréquence, la détection du signal de synchronisation primaire (PSS) et du signal de synchronisation secondaire (SSS) pour déterminer l'identité des cellules de la couche physique. LTE System Toolbox™ est utilisé pour vérifier la fonctionnalité des modèles HDL en fournissant un stimulus d'entrée et des formes d'onde de sortie de référence. Les schémas du modulateur et du détecteur sont présentés ci-dessous dans la figure 3.1

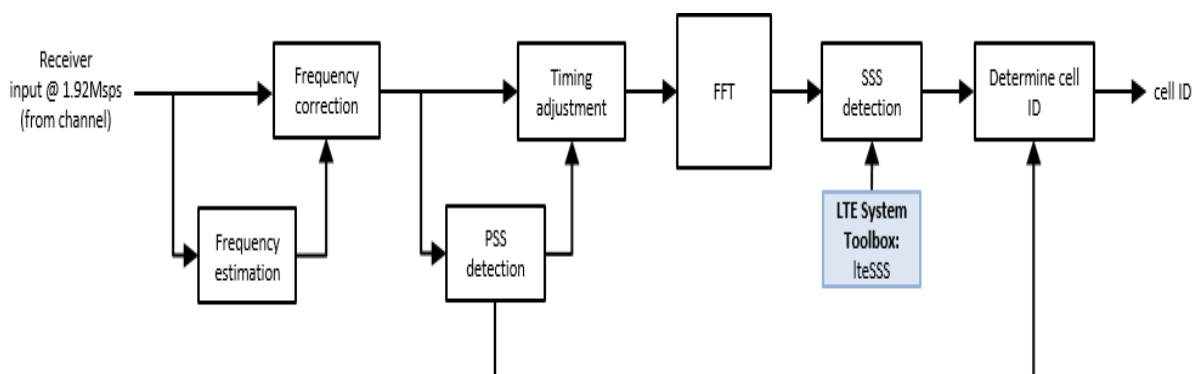


Figure 3.1 Structure du modulateur et du détecteur LTE

La figure 3.1 représente Le sous-système HDL du modulateur LTE. Il prend une grille de ressources de liaison descendante (DL) d'évolution à long terme (LTE) pré-générée qui est créée à l'aide du bloc LTE System Toolbox™ et effectue une modulation OFDM conformément à la norme LTE. Le sous-système modulateur est paramétrable et prend en charge toutes les largeurs de bande de canal LTE standard. Pour vérifier la sortie de l'implémentation HDL, une référence, la forme d'onde modulée OFDM est créée à l'aide de la LTE System Toolbox, les fonctions `lteTestModel` et `lteTestModelTool`. La forme d'onde de référence par défaut utilisée pour la vérification est générée conformément au modèle de test E-UTRA (E-TM) 1.1, et à une bande passante variable (Tableau 3.1)

Les détails des bandes passantes prises en charge, ainsi que les longueurs IFFT associées et les fréquences d'échantillonnage, sont fournis dans le tableau ci-dessous.

| BW N° | Bande passante(BW) | Longueur IFFT | Debit symbole (Msps) |
|-------|--------------------|---------------|----------------------|
| 1 | 1.4 | 128 | 1.92 |
| 2 | 3 | 256 | 3.84 |
| 3 | 5 | 512 | 7.68 |
| 4 | 10 | 1024 | 15.36 |
| 5 | 15 | 2048 | 30.72 |
| 6 | 20 | 2048 | 30.72 |

Tableau 3.1 Bande Passante OFDM LTE

Pour démontrer le fonctionnement du récepteur. Un modèle de canal à virgule flottante (floating –point channels), est utilisé pour ajouter l'atténuation, le bruit du canal et le retard. Le sous-système `LTE_Detector_HDL` implémente les étapes initiales de la fonctionnalité de récepteur OFDM pour identifier l'ID de cellule LTE.

3.2 Fonctions de System Toolbox LTE

Diverses fonctions de la Toolbox du système LTE sont utilisées dans cette structure de simulation pour générer une forme d'onde modulée de référence qui est utilisée pour vérifier

l'implémentation HDL. Ces fonctions et leur utilité dans la structure de simulation sont mis en évidence ci-dessous.

3.2.1 **Fonctions utilisées dans le script d'initialisation LTE :**

- lteTestModel: Génère une configuration de structure LTE de référence correspondant à un modèle de test (E-TM) d'accès radio terrestre universel évolué (E-UTRA)
- lteOFDMInfo: Renvoie les informations de modulation OFDM relatives à la forme d'onde de référence
- lteTestModelTool: Génère une forme d'onde E-TM de liaison descendante LTE de référence en or basée sur la structure créée par la fonction lteTestModel

3.2.2 **Fonctions utilisées dans le script d'analyse après simulation :**

- lteSSS: Génère les séquences LTE SSS pour l'identification des cellules

Les scripts d'analyse d'initialisation et du résultat après simulation peuvent être visualisés en double-cliquant sur les boutons étiquetés dans le niveau supérieur du modèle Simulink.

3.3 Structure de Simulations avec Simulink

La modulation est illustrée dans la figure suivante ci-dessous (figure 3.2) qui représente le modèle d'implémentation HDL réalisé pour l'interface physique de la chaîne de transmission LTE OFDM, constitué de deux sous-système, les sous-systèmes modulateur et le sous-système détecteur qui sont configurés pour la génération de code HDL.

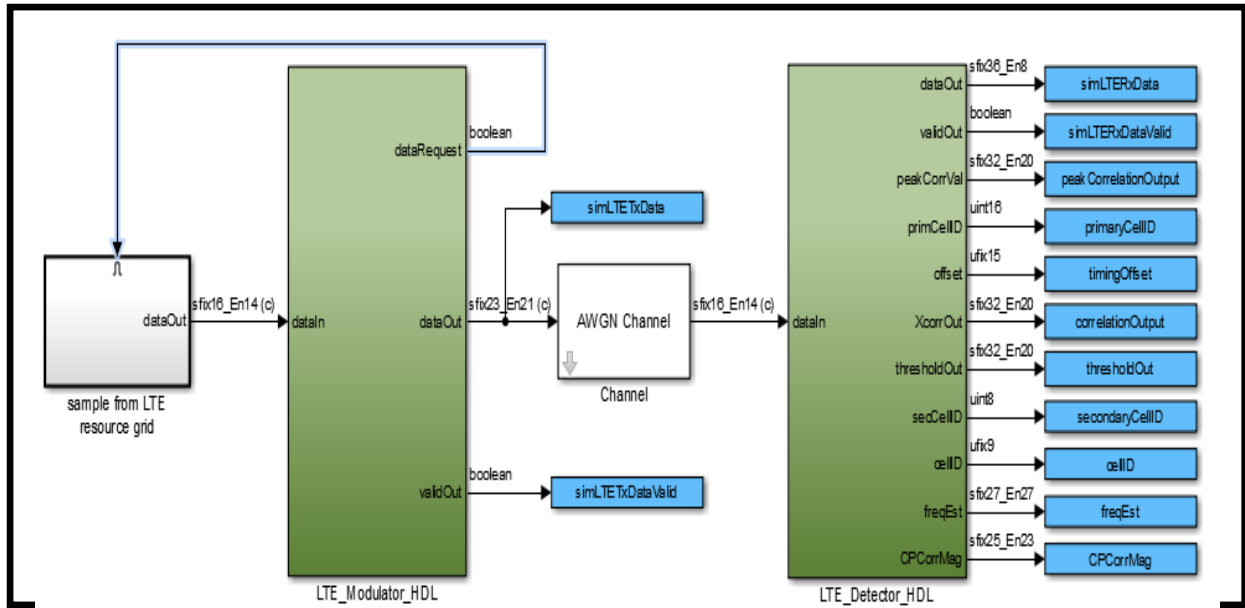


Figure 3.2 l'implémentation HDL du modulateur et détecteur OFDM LTE

3.4 Sous-système HDL du modulateur LTE

La structure détaillée du sous-système HDL du modulateur LTE est illustrée dans la figure ci-dessous.

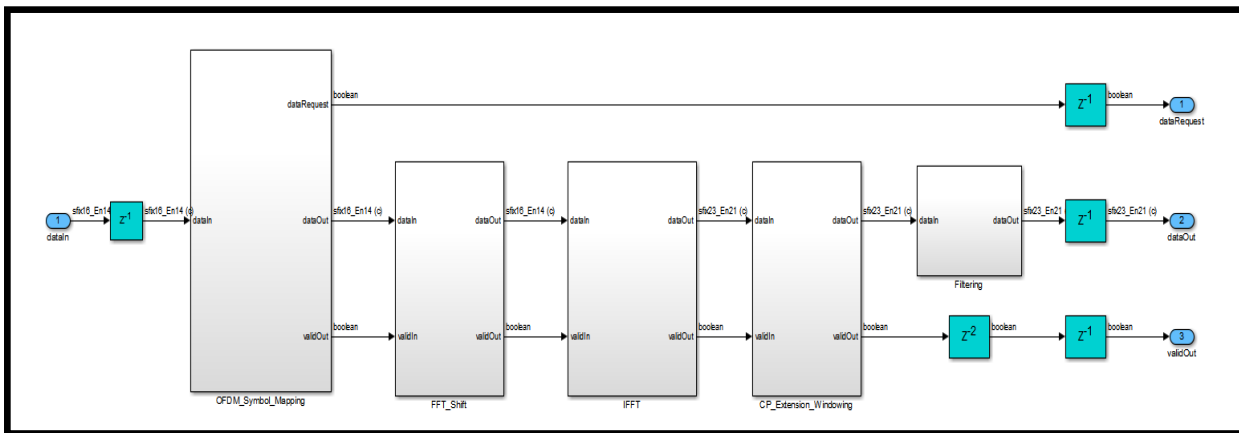


Figure 3.3 Structure d'implémentation HDL d'un modulateur OFDMLTE

Le sous-système du modulateur (LTE Modulator HDL) contient les composants suivants :

- **OFDM_Symbol_Mapping:** Pour faire mapper les échantillons de données d'entrée aux sous-porteuses centrales de l'IFFT et d'effectue l'insertion de la sous-porteuse DC ainsi de réserver le nombre correct d'échantillons pour l'extension Cyclic Prefix (CP)
- **FFT_Shift:** Pour réorganise les échantillons de données d'entrée vers l'IFFT afin que le signal LTE modulé soit correctement aligné dans le spectre de fréquences

- **IFFT:**Pour effectuer la transformation de Fourier rapide inverse (IFFT) et moduler les données de la grille de ressources LTE
- **CP_Extension_Windowing :**Pour planifie l'extension CP et effectuer le chevauchement ainsi d'ajoute une opération de fenêtrage
- **Filtrage :**Pour filtrer le signal de transmission afin s'assurer qu'il répond aux exigences de masque spectral requises

3.5 Sous-système HDL détecteur LTE

Le diagramme suivant ci-dessous figure 3.4 montre la structure détaillée de LTE_Detector_HDL

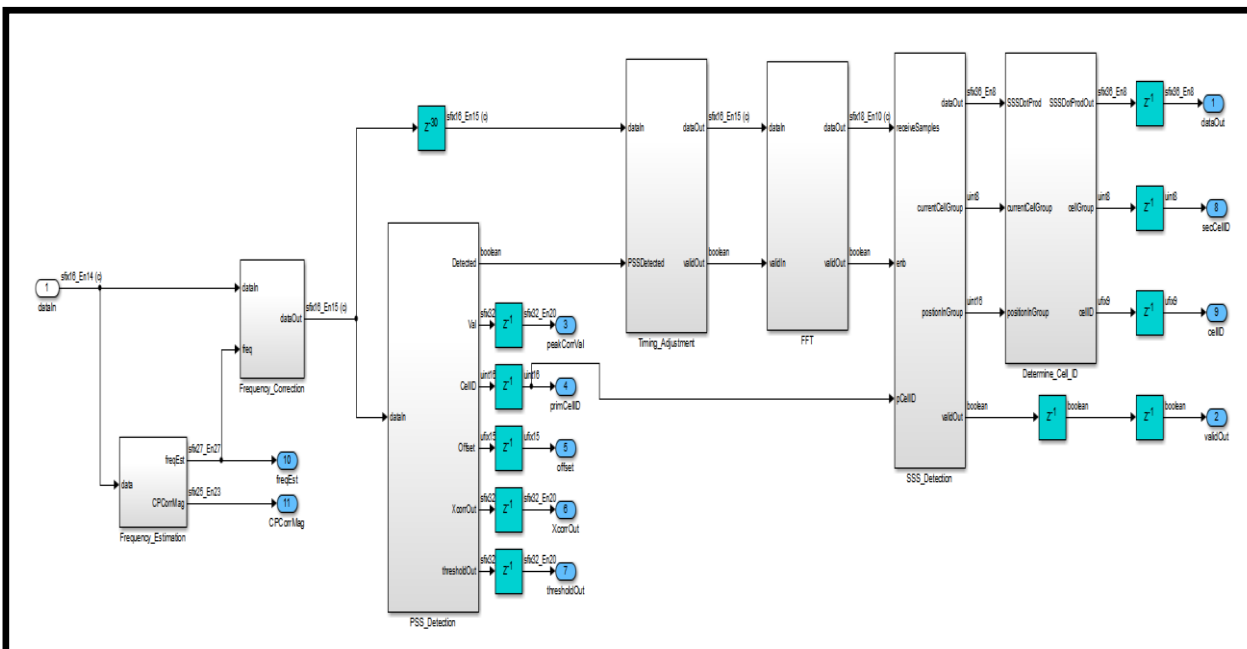


Figure 3. 4 Structure d'implémentation d'un détecteur LTE HDL

Le sous-système du détecteur (LTE_Detector_HDL) contient les composants suivants

- **Frequency_Estimation:** estime le décalage de fréquence du signal reçu à l'aide du préfixe cyclique.
- **Frequency_Correction:** corrige le décalage de fréquence déterminé par le sous-système Frequency_Estimation.
- **PSS_Detection:** effectue une corrélation croisée du signal reçu avec trois signaux de synchronisation primaire (PSS) possibles pour déterminer l'identité de cellule au sein du groupe et pour calculer le décalage de synchronisation reçu.

- **Timing_Adjustment:** utilise la valeur de décalage de synchronisation calculée à partir de PSS_Detection pour planifier l'entrée de données dans la FFT
- **FFT:** effectue l'opération de transformation de Fourier rapide (FFT) pour démoduler le flux de données reçu
- **SSS_Detection:** effectue un produit scalaire des échantillons de domaine de fréquence reçus et des 168 signaux de synchronisation secondaire (SSS) possibles pour déterminer le groupe de cellules.
- **Determine_Cell_ID:** calcule l'identité de la cellule LTE à partir du groupe de cellules détecté (à partir de la détection SSS) et de la position détectée dans le groupe de cellules (à partir de la détection PSS).

3.6 La structure du sous-système canal

La structure du sous-système canal est illustrée ci-dessous. Le modèle de canal se compose d'un bruit AWGN, d'un retard, d'une atténuation et d'un décalage de fréquence. Le signal est d'abord converti en double, imitant le fonctionnement d'un convertisseur numérique-analogique (DAC). Le bruit additif, la temporisation, l'atténuation et le décalage de fréquence sont ensuite appliqués. Enfin, le taux d'échantillonnage est réduit à 1,92 Msps à l'aide d'un filtre de décimation FIR, et le signal est converti en données à virgule fixe 16 bits, modélisant un convertisseur analogique-numérique (ADC).

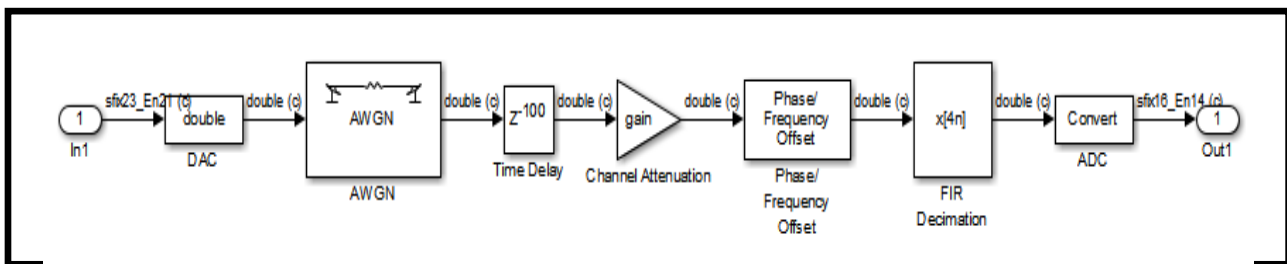


Figure 3.5 : La structure du sous-système canal AWGN

3.7 Résultats de simulations

Après avoir exécuté la simulation, le modèle utilisé présente trois figures différentes illustrant les sorties et les résultats. Ces données sont présentées ci-dessous, accompagnés d'une explication de chaque figure. La première figure illustre la sortie du modulateur LTE OFDM, la deuxième figure affiche la sortie de la détection PSS dans le détecteur LTE OFDM,

tandis que le tracé final fournit divers résultats textuels à partir des étapes finales du détecteur. Pour réaliser notre expérience nous avons utilisé un ordinateur qui possède les caractéristiques techniques et logiciel suivantes :

- Processeur intel(R) Core™ i3-4005U CPU
- Mémoire installée (RAM) 4.00GHZ
- Système d'exploitation Windows 7 (64 bit)
- MATLAB /SIMULINK

Les figures des résultats obtenus illustrent la sortie du modulateur OFDM LTE et est divisé en deux sous-graphiques (densité spectrale et domaine temporel) , un graphique d'estimation, un diagramme de corrélation croisée (PSS et SSS) et des sorties de calcul final de simulation :

1. Tracé de densité spectrale de puissance (PSD): Ce sous-graphique montre la densité spectrale de puissance (PSD) de la sortie du modulateur OFDM LTE. Le résultat est tracé au-dessus de la PSD du signal de sortie de référence d'or qui a été généré à l'aide de la LTE System Toolbox pour montrer visuellement l'équivalence des deux signaux. La bande passante de transmission LTE, BW, est également affichée dans le titre de la figure.
2. Tracé du domaine temporel: ce sous-tracé montre une partie agrandie de la sortie à valeur absolue du modulateur OFDM LTE au fil du temps. Le résultat est tracé au-dessus de la même partie du signal de sortie de référence dorée à valeur absolue qui a été généré à l'aide de la LTE System Toolbox du système LTE pour montrer visuellement l'équivalence des deux signaux. Afin de comparer d'avantage la sortie à celle du signal de référence, un troisième signal est tracé qui montre la différence de valeur absolue entre la sortie de l'implémentation HDL et le signal de référence (c'est-à-dire $\text{abs}(LSTreference - HDLImplementation)$). Cela illustre l'erreur minimale entre les deux signaux.

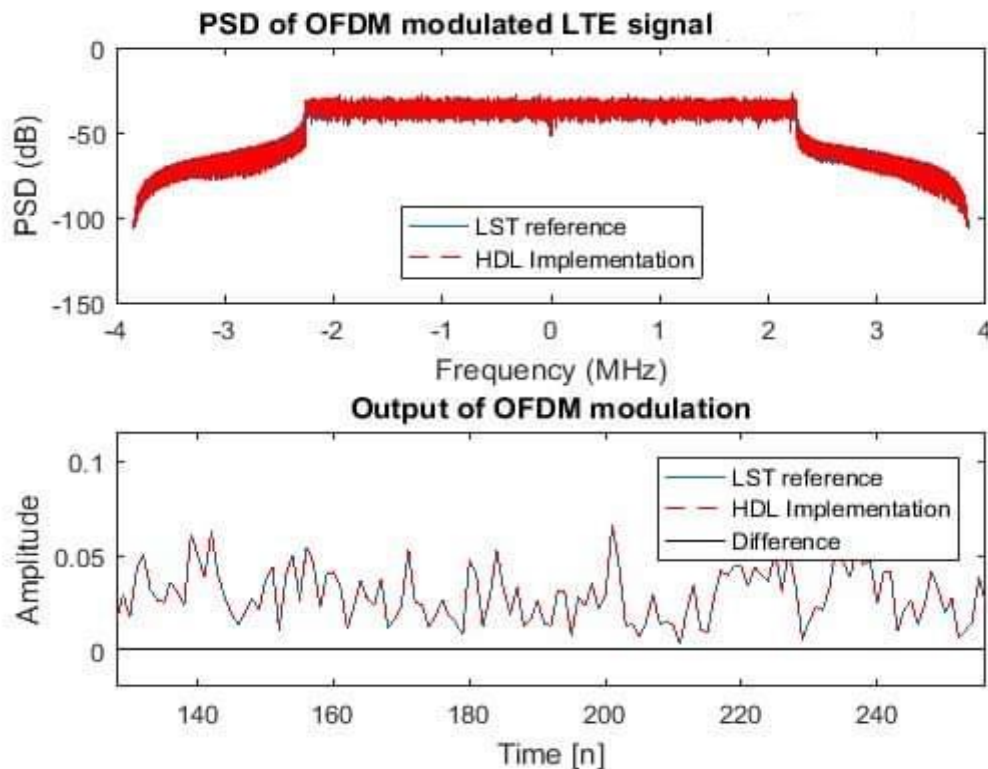


Figure 3.6 : Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM

3. Graphique d'estimation de fréquence : Le graphique suivant montre l'estimation de fréquence convergente. Une nouvelle estimation de fréquence est générée tous les 960 échantillons, correspondant à la période d'intervalle. Un filtre unipolaire est utilisé pour lisser les estimations, ce qui explique la courbe montrée dans le graphique.

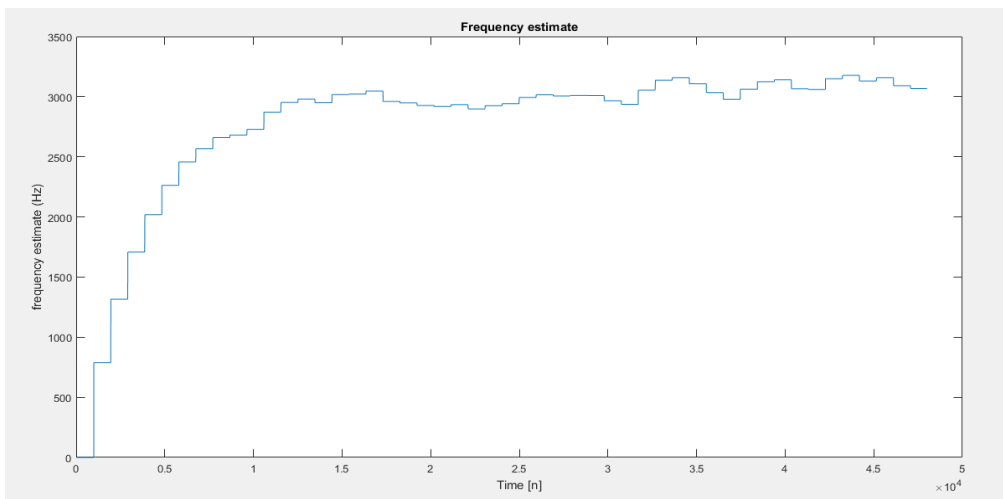


Figure 3.7:Graphique d'estimation de fréquence

4. Diagramme de corrélation croisée PSS : Le graphique suivant illustre la sortie de la détection PSS. Le graphique montre la puissance de sortie de la corrélation croisée du signal reçu avec la séquence PSS détectée au fil du temps. Le signal de seuil moyen est également tracé pour illustrer le processus d'identification. Les pics visibles qui dépassent le seuil indiquent que cette séquence PSS a été détectée. Deux pics sont visibles car la séquence PSS est transmise deux fois par trame radio LTE.

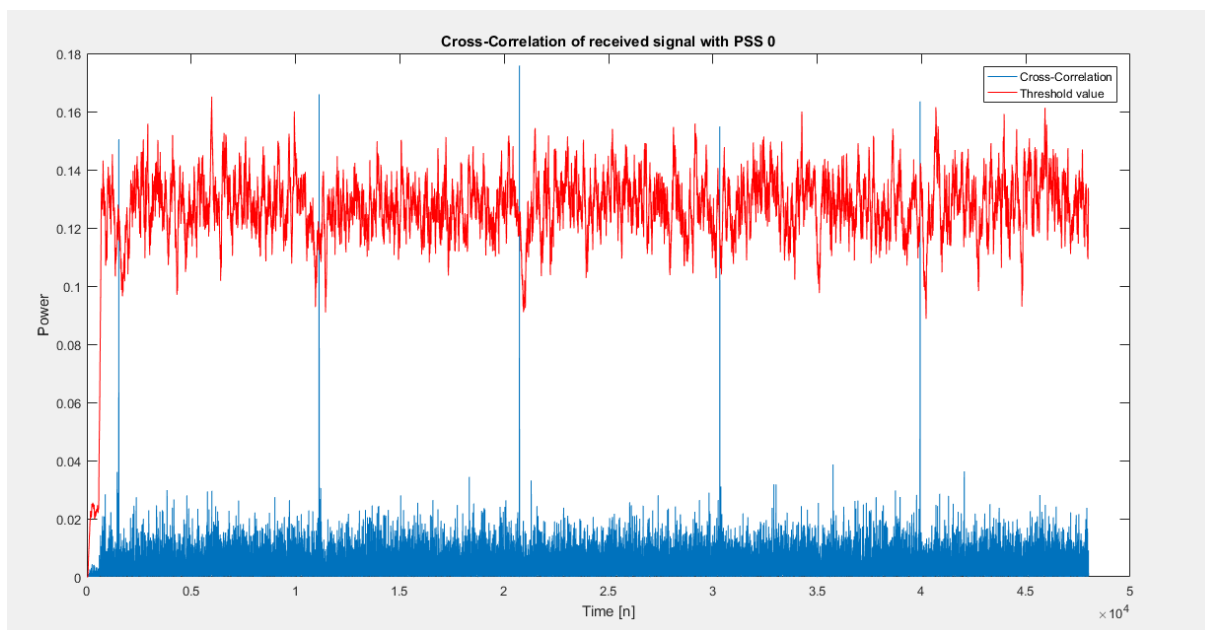


Figure 3.8 : Diagramme de corrélation croisée PSS

5. Diagramme de corrélation croisée SSS : Le graphique suivant montre la sortie de SSS_Detection. Le graphique illustre la puissance de sortie du produit scalaire entre le signal reçu et les séquences SSS possibles. Les valeurs sur l'axe X correspondent aux séquences SSS possibles. Pour chaque séquence SSS, seule la valeur finale du produit scalaire est tracée. La séquence SSS avec la plus grande puissance de sortie de produit scalaire est choisie comme séquence transmise.

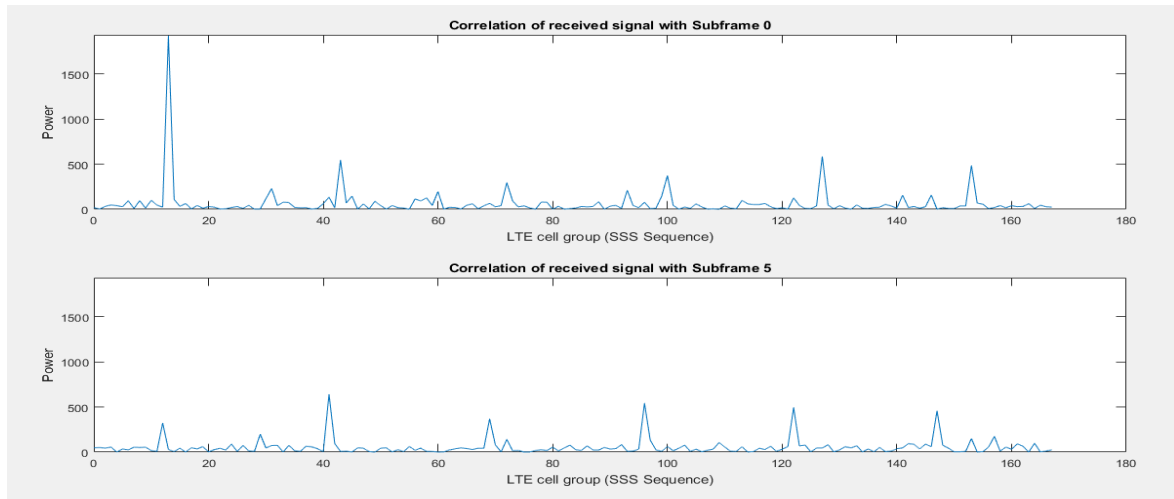


Figure 3.9 : Diagramme de corrélation croisée SSS

6. Sorties finales de simulation : Le graphique suivant fournit des résultats textuels pour les étapes finales de la simulation. Il est divisé en trois sections :

- ID de cellule détecté : affiche les résultats finaux de la recherche de cellule LTE.
- Valeurs de corrélation croisée PSS : affiche la puissance de corrélation croisée de crête et le décalage de synchronisation à partir de la détection PSS.
- HDL Implémentation TX Error Vector Magnitude (EVM): affiche les valeurs de pic et RMS Error Vector Magnitude (EVM) de l'implémentation HDL du modulateur OFDM LTE par rapport au signal de référence d'or LTE System Toolbox.

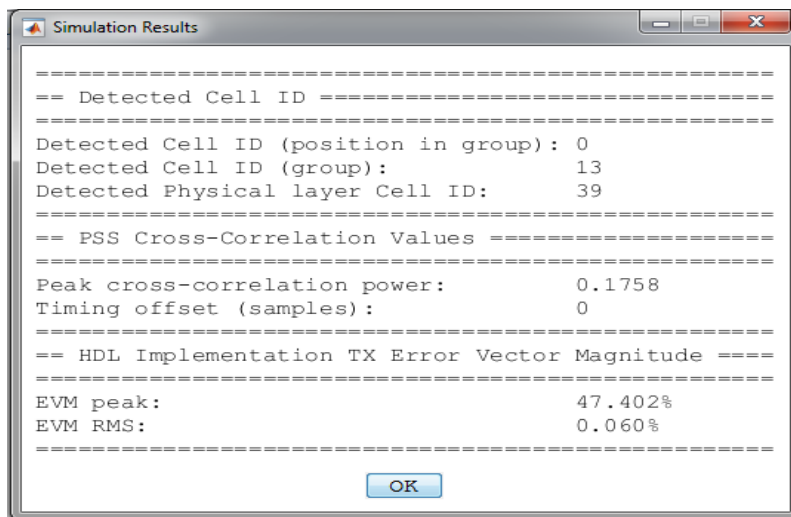


Figure 3.10 : Résultats basés sur les étapes finales de la simulation

7. Génération des codes HDL du modulateur et détecteur OFDM LTE (voir Annexe)

3.8 Analyse des résultats

Après avoir exécuté la simulation, les résultats obtenus sont présentés dans les différentes figures ci-dessous à l'exception des résultats avec bande passante 5MHz sont présentés ci-dessus.

3.8.1 Influence de variation Bande Passante, Atténuation canal et SNR canal

1. BW=1.4 Mhz

- SNR=5 DB Gain=0.9

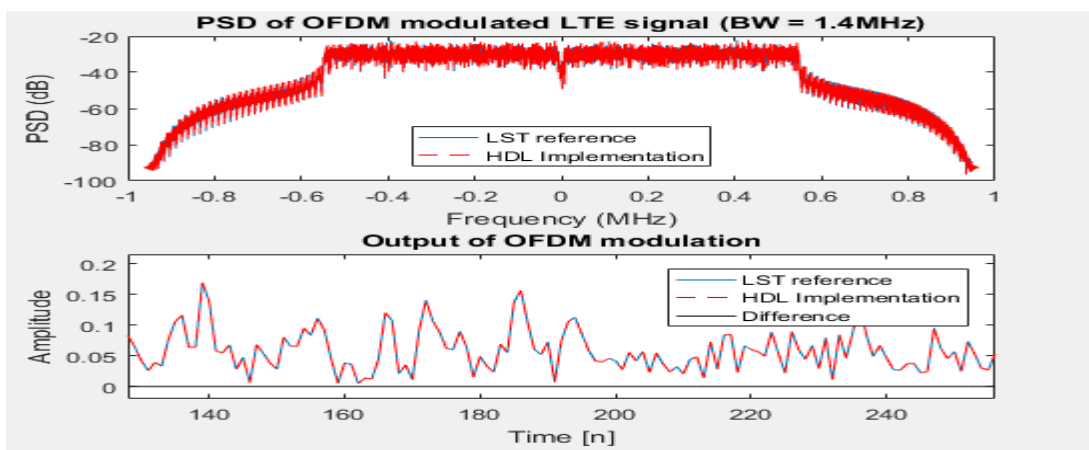
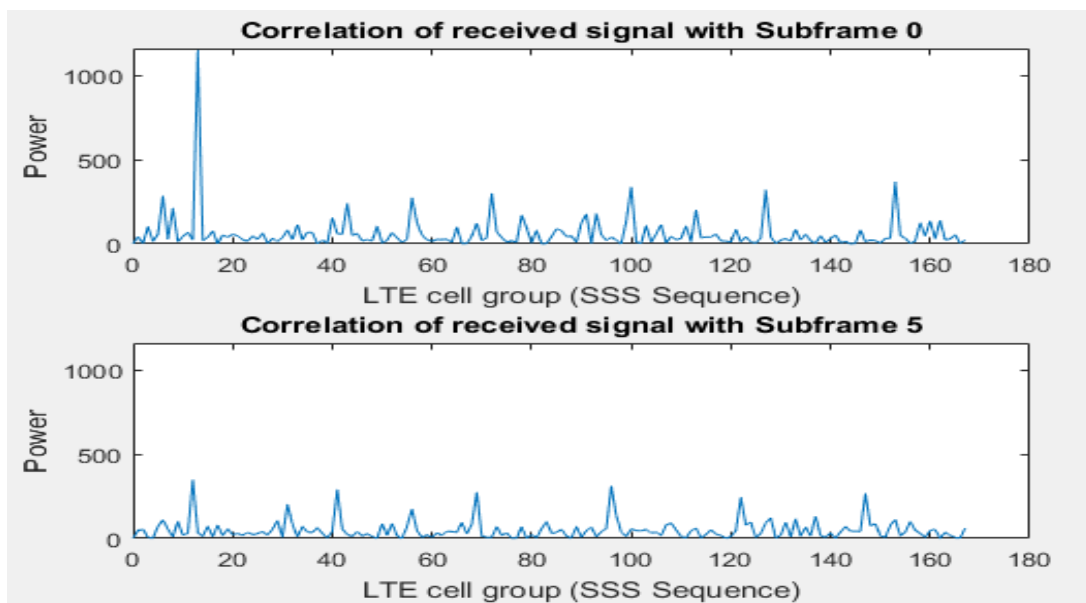


Figure 3.11 : Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM

(BW=1.4MHZ)



- Figure 3.12 : Diagramme de corrélation croisée SSS(BW=1.4MHZ)

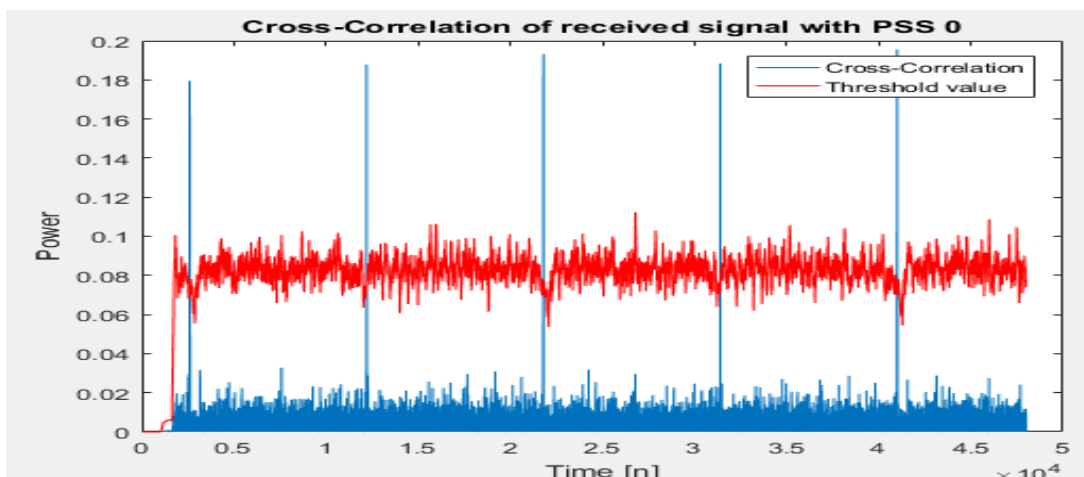


Figure 3.13 : Diagramme de corrélation croisée PSS (BW=1.4MHZ)

- SNR=10 dB
- Gain =0.2

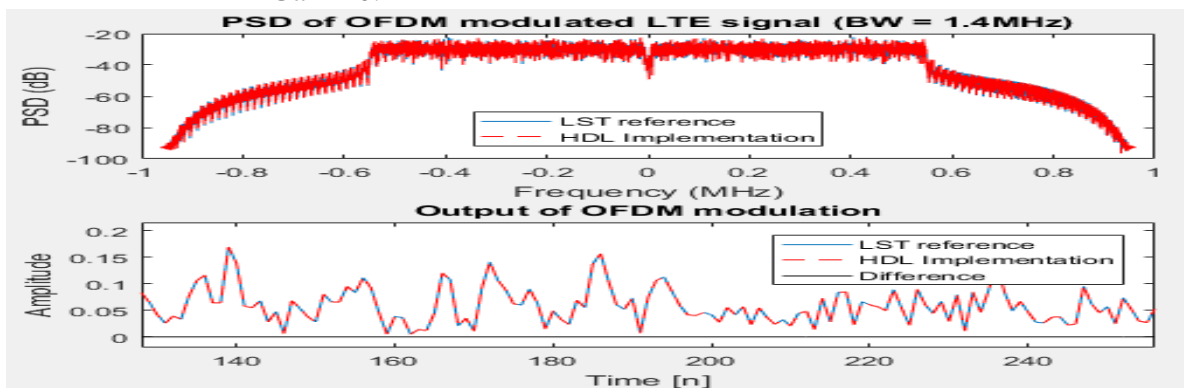


Figure 3.14 : Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM

(BW=10MHZ)

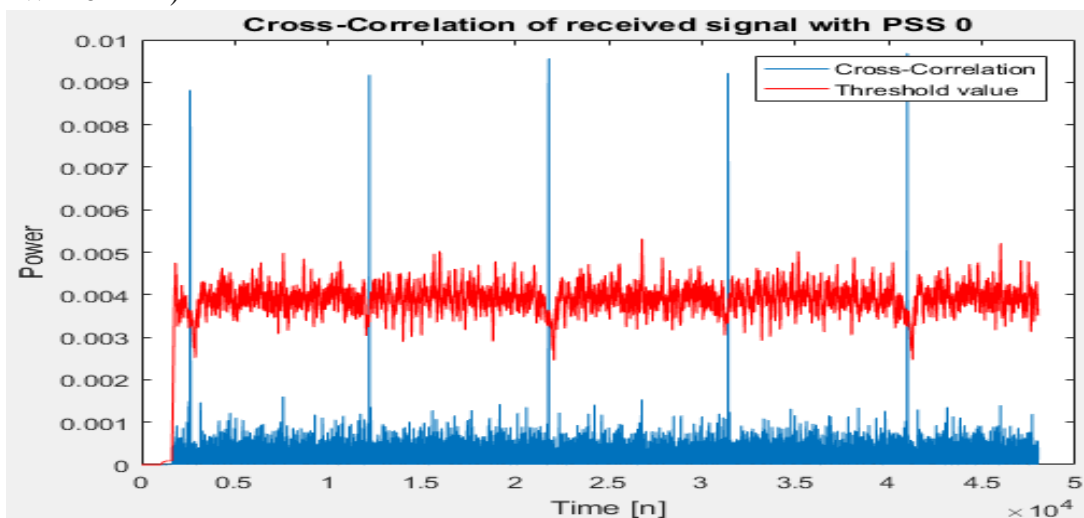


Figure 3.15 : Diagramme de corrélation croisée PSS (BW=10MHZ)

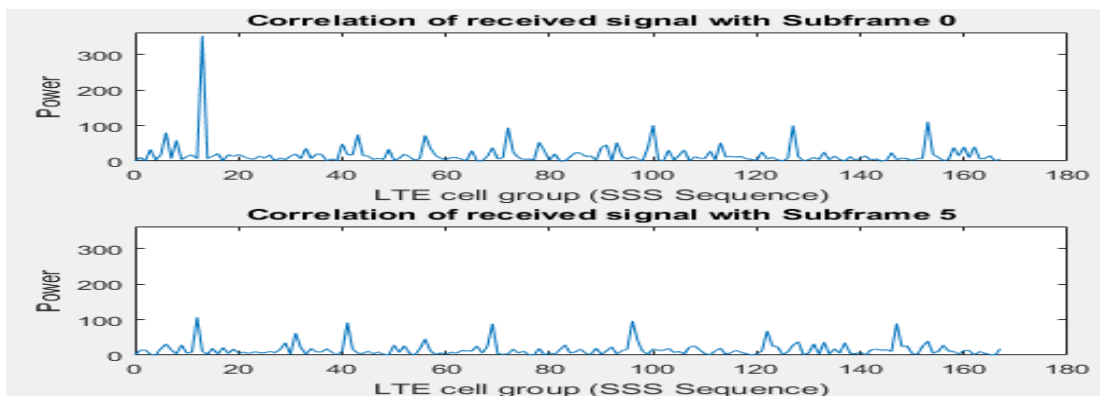


Figure 3.16 : Diagramme de corrélation croisée SSS(BW=10MHZ)

➤ Gain =0.5

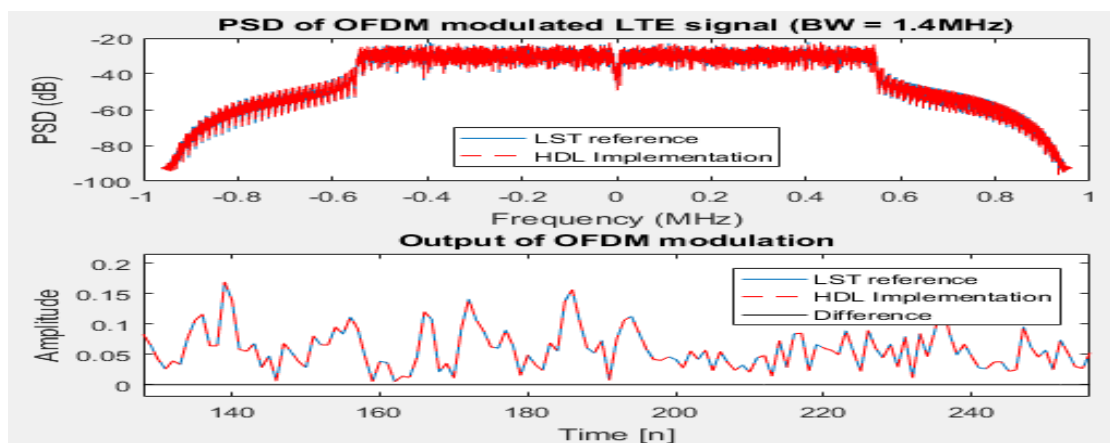


Figure 3.17 : Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM

(BW=10MHZ)

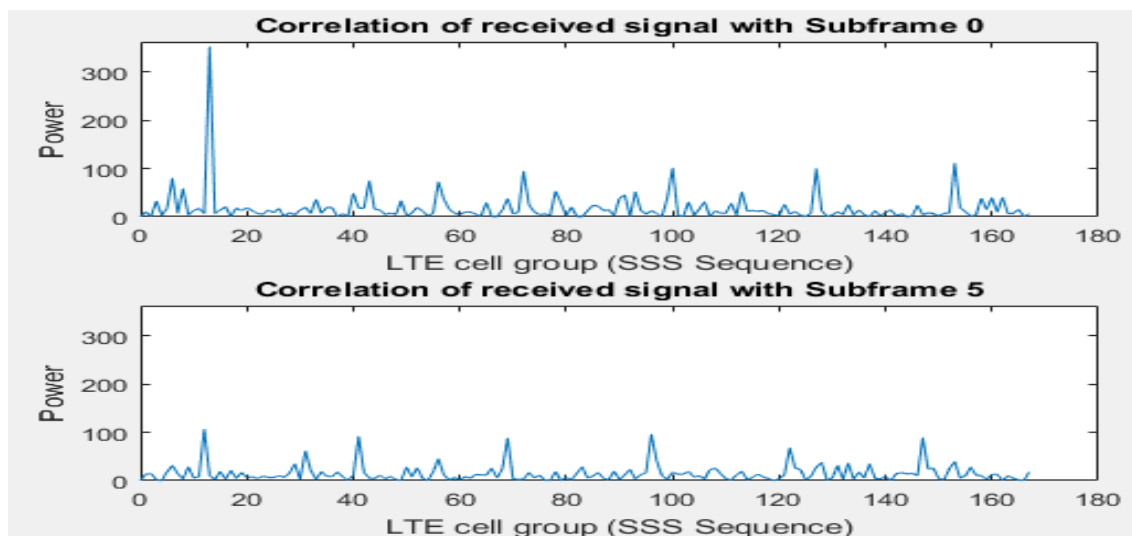


Figure 3.18:Diagramme de corrélation croisée SSS (BW=10MHZ)

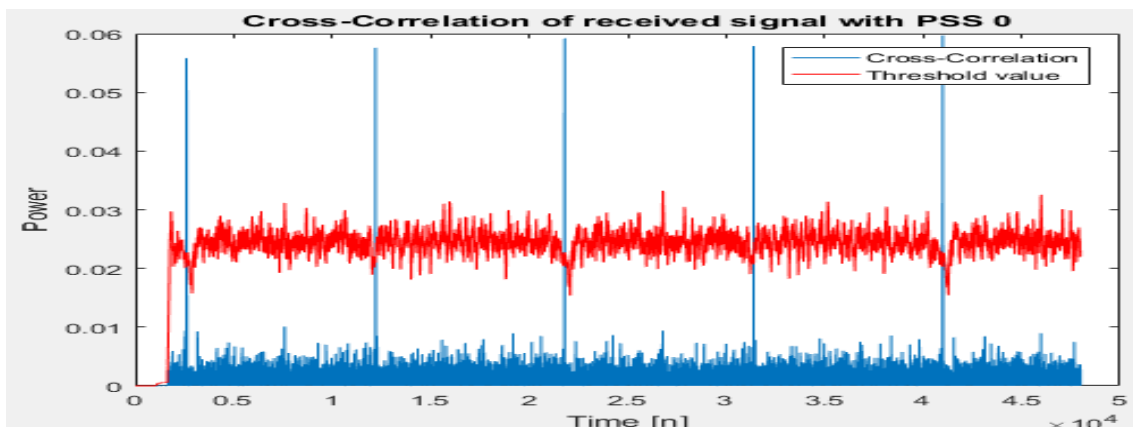


Figure 3.19: Diagramme de corrélation croisée PSS (BW=10MHZ)

➤ Gain =0.7

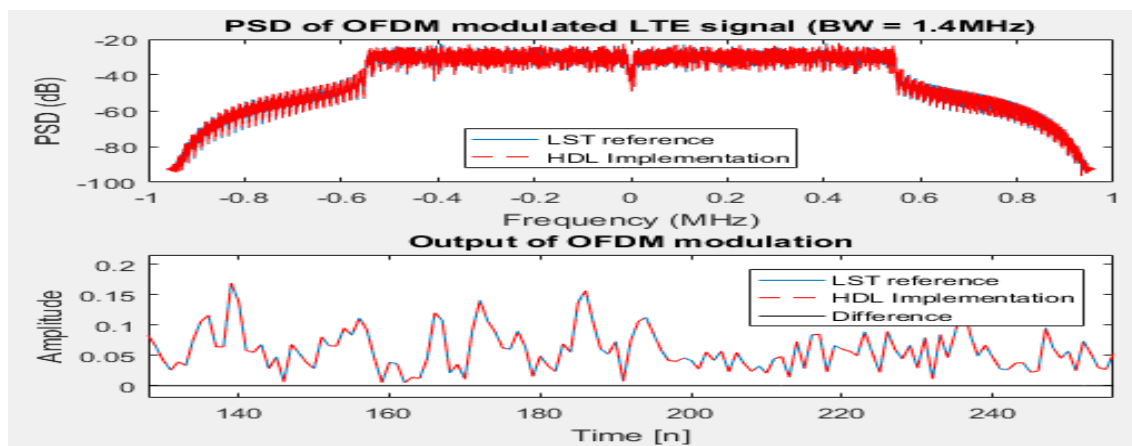


Figure 3.20: Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ)

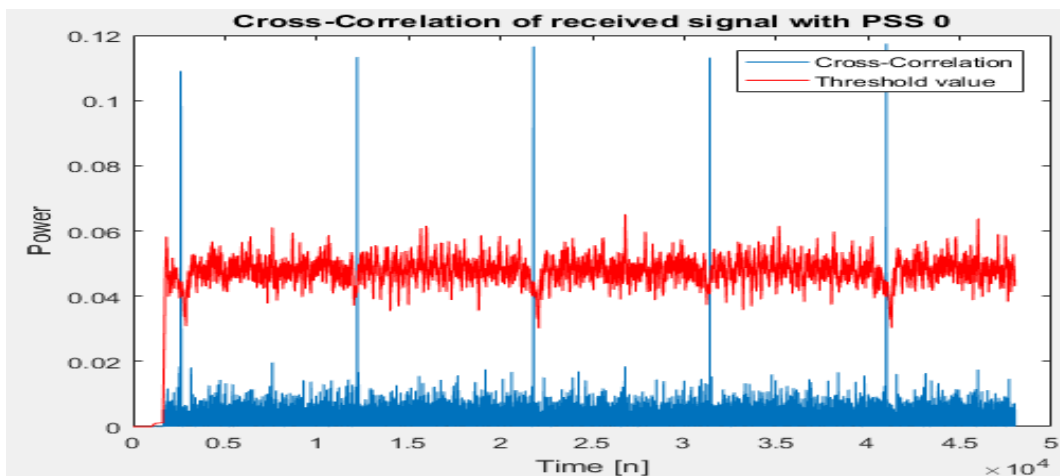


Figure 3.21 : Diagramme de corrélation croisée PSS (BW=10MHZ)

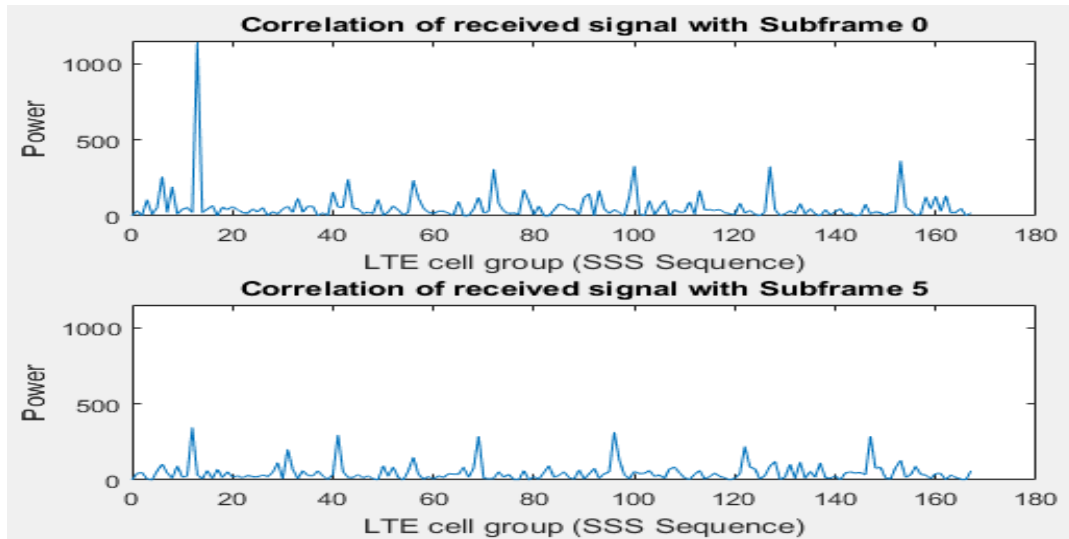


Figure 3.22: Diagramme de corrélation croisée SSS (BW=10MHz)

➤ Gain = 0.9

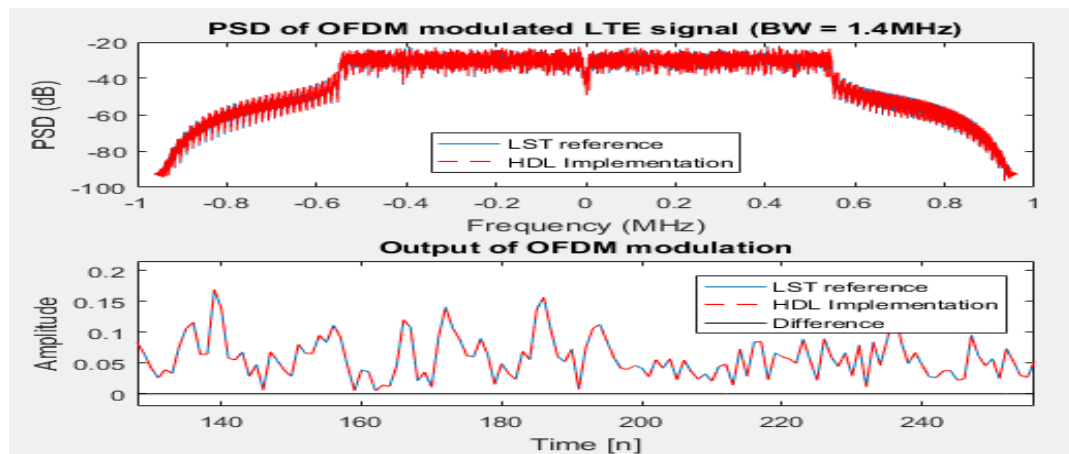


Figure 3.23: Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHz)

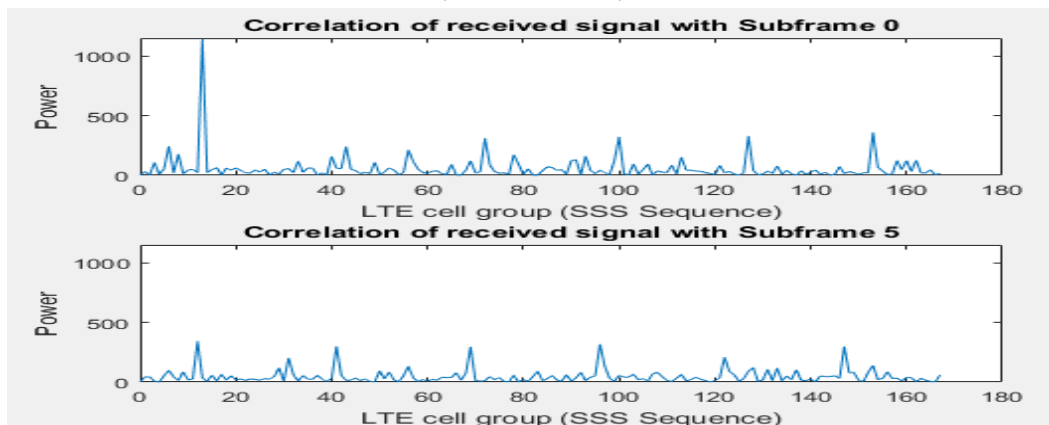


Figure 3.24: Diagramme de corrélation croisée SSS (BW=10MHz)

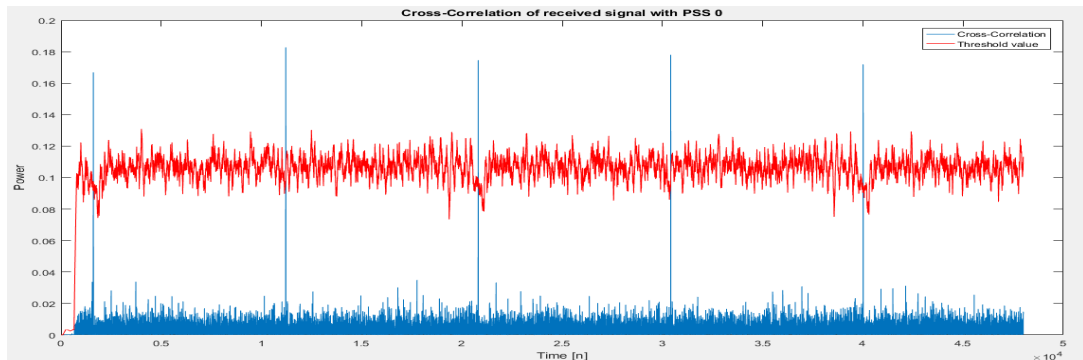


Figure 3.25 : Diagramme de corrélation croisée PSS (BW=10MHZ)

- SNR=15dB, GAIN= 0.9

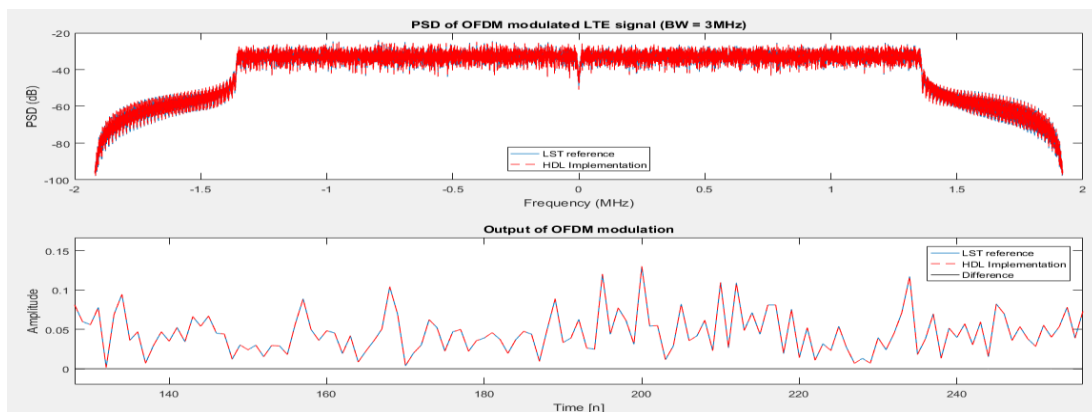


Figure 3.26: Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM (BW=15MHZ)

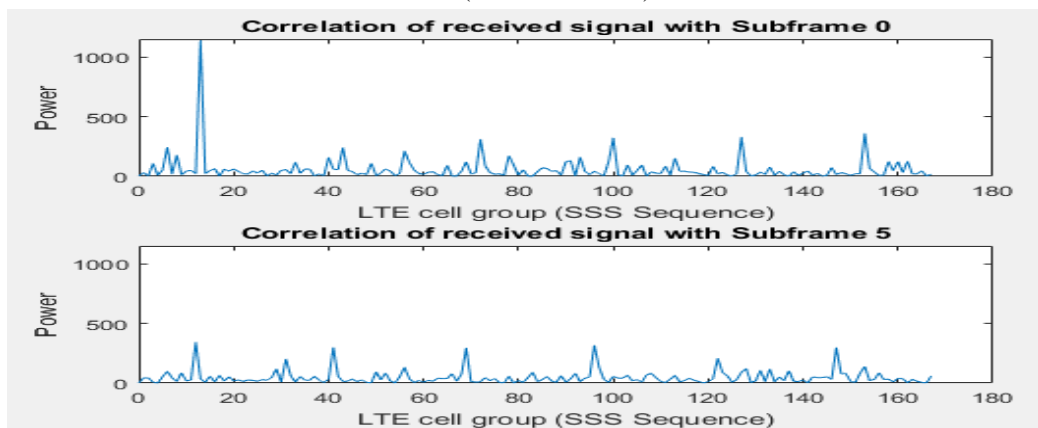


Figure 3.27: Diagramme de corrélation croisée SSS(BW=15MHZ)

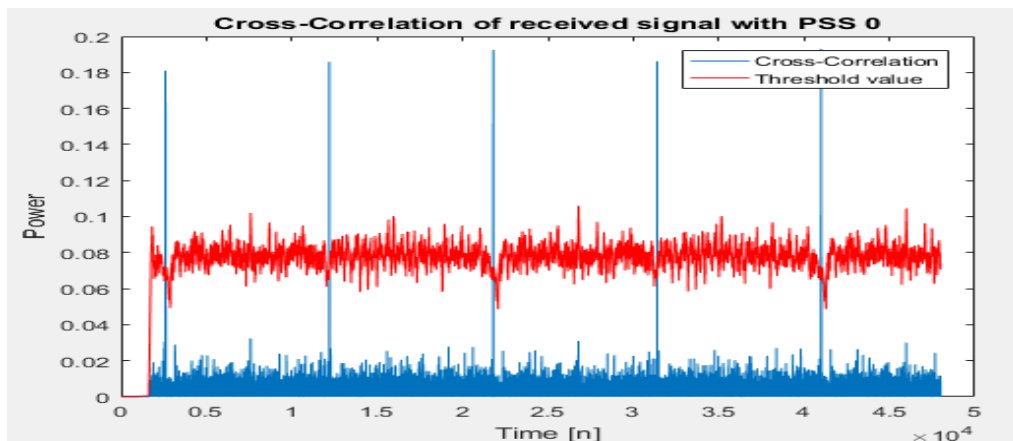


Figure 3.28 : Diagramme de corrélation croisée PSS (BW=15MHZ)

- SNR=20dB, GAIN= 0.9

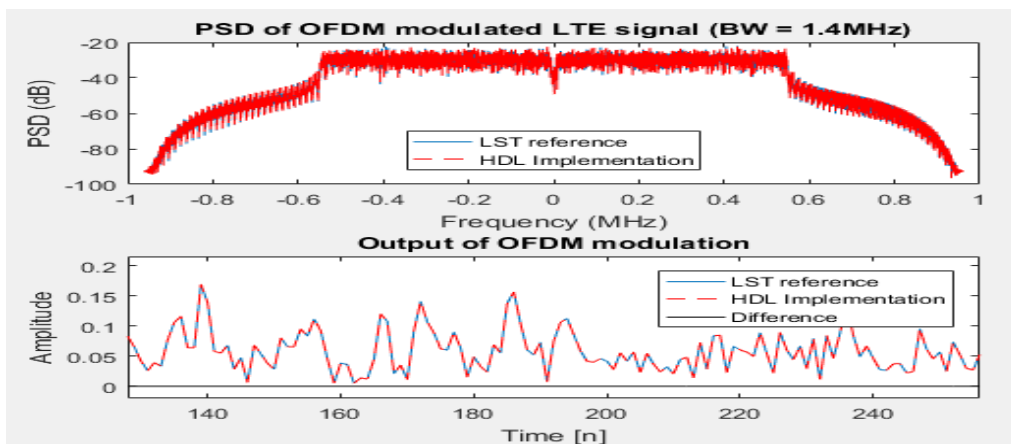


Figure 3.29: Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=20MHZ)

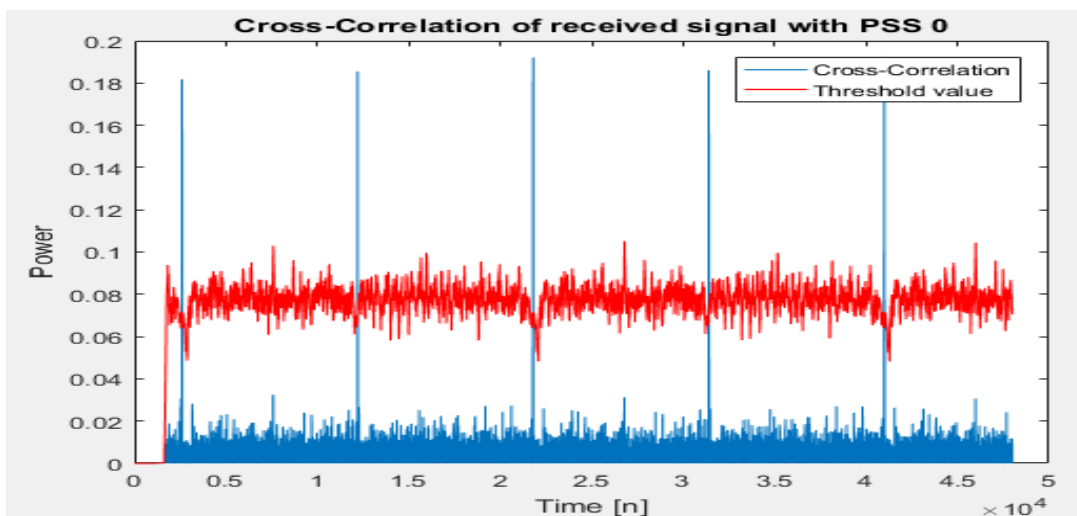


Figure 3.30 : Diagramme de corrélation croisée PSS (BW=20MHZ)

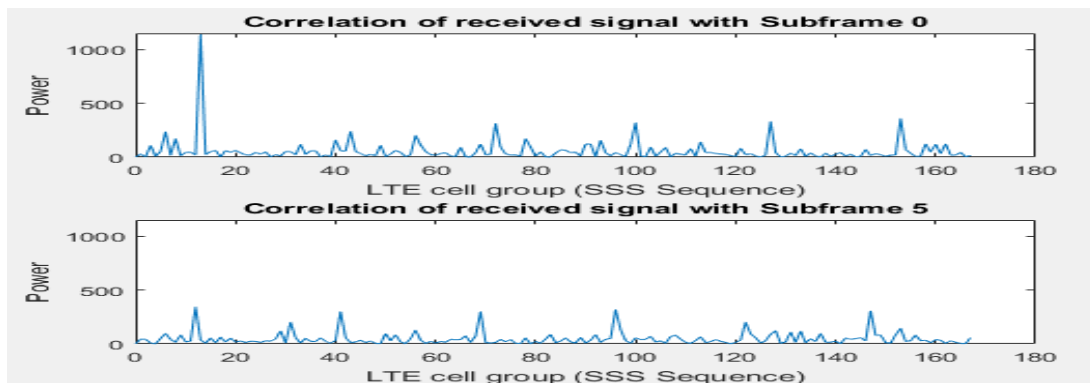


Figure 3.31: Diagramme de corrélation croisée SSS (BW=20MHZ)

Après avoir changé la valeur de paramètre de bande passante à 1.4MHZ le taux d'erreur a donner un pourcentage égale a : Taux d'erreur (RMS) =0.714 %

2. BW= 3 Mhz

Avec le paramètre bande passante égale a 3 (BW=3MHZ) :

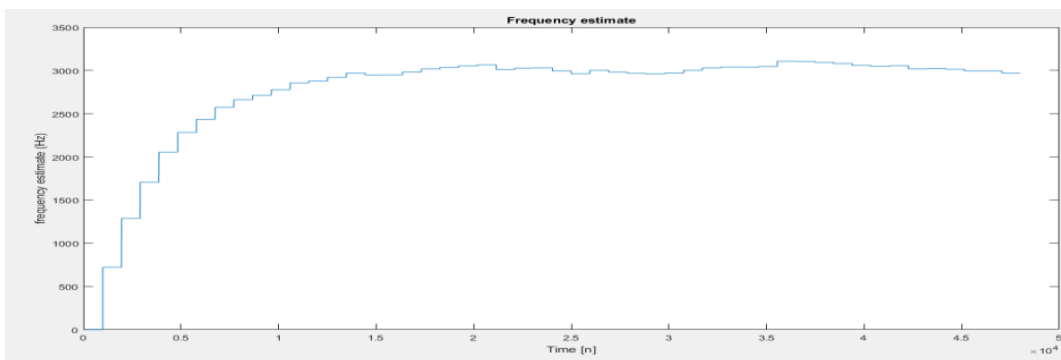


Figure 3.32 : Graphique d'estimation de fréquence (BW=3MHZ)

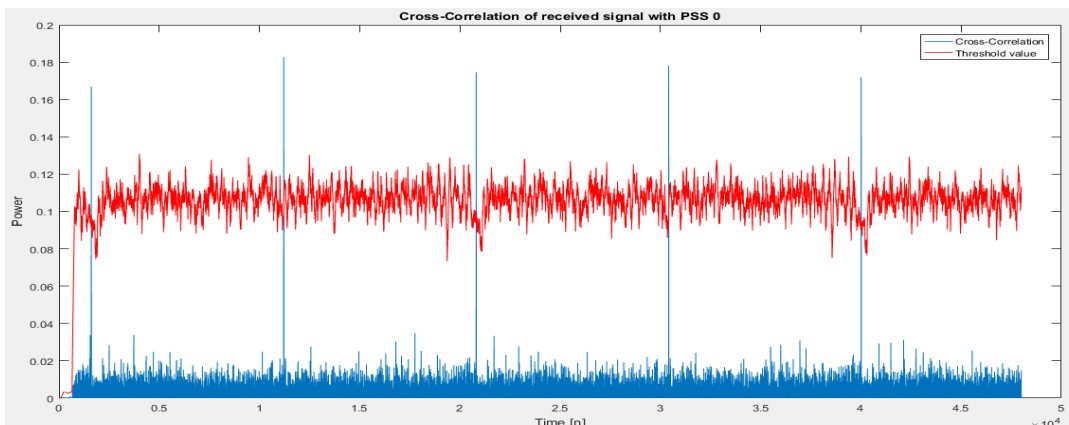


Figure 3.33 : Diagramme de corrélation croisée PSS (BW=3MHZ)

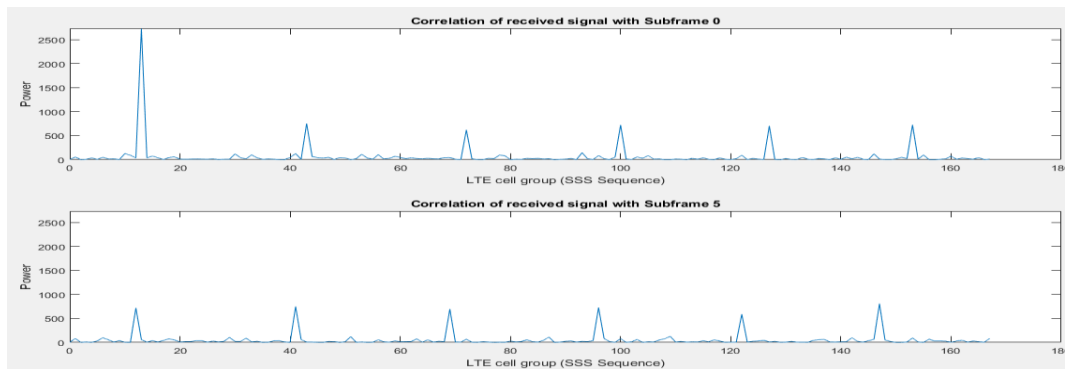


Figure 3.34 : Diagramme de corrélation croisée SSS(BW=3MHZ)

Dans le cas de paramètre bande passante égale a 3MHZ le Taux d'erreur (RMS) a diminuée a 0.327%.

3. BW= 5Mhz

La variation avec 5Mhz est mentionné dans les figures ci-dessus 3.6 ,3.7 ,3.8,3.9,3.10 Dans le cas de paramètre bande passante égale a 5MHZ le Taux d'erreur a diminuée a 0.274%

4. BW= 10 Mhz

Avec le paramètre bande passante égale a 10 (BW=10MHZ) :

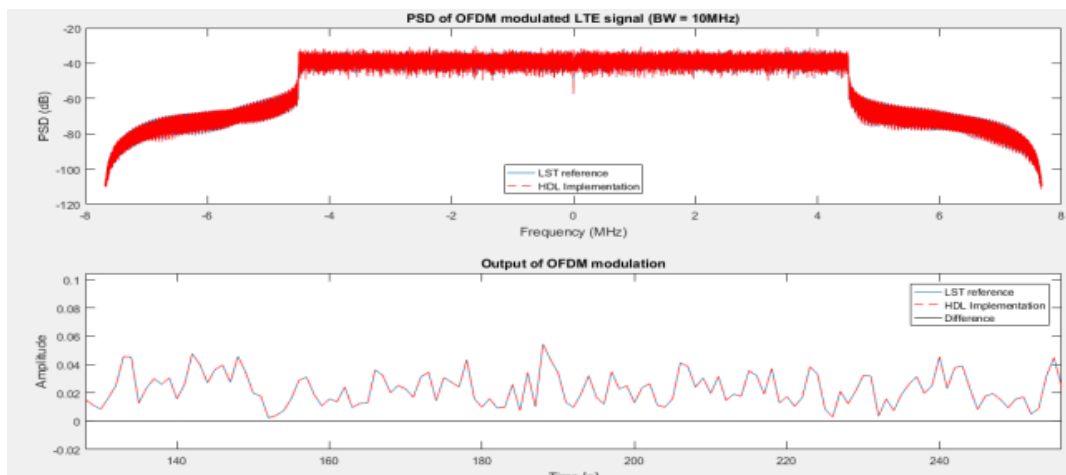


Figure 3.35: Densité spectral de puissance (PSD) et sortie de du Modulateur OFDM (BW=10MHZ)

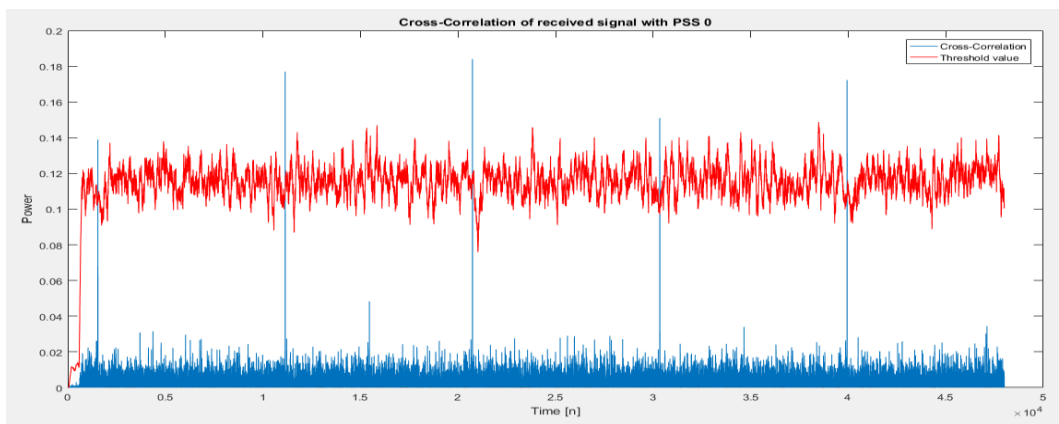


Figure 3.36 : Diagramme de corrélation croisée PSS (BW=10MHZ)

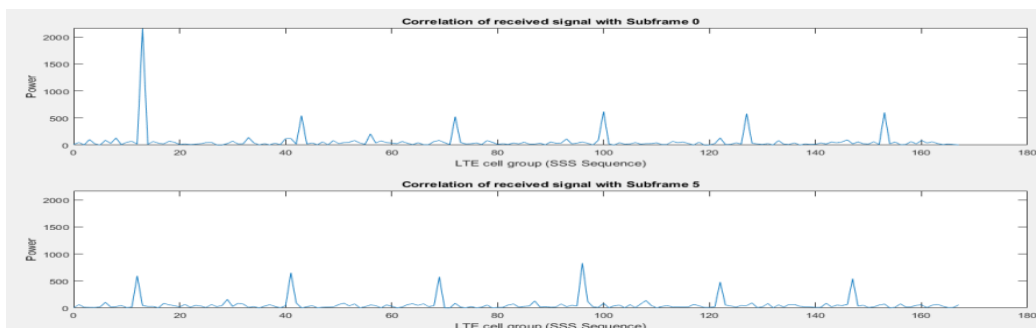


Figure 3.37: Diagramme de corrélation croisée SSS (BW=10MHZ)

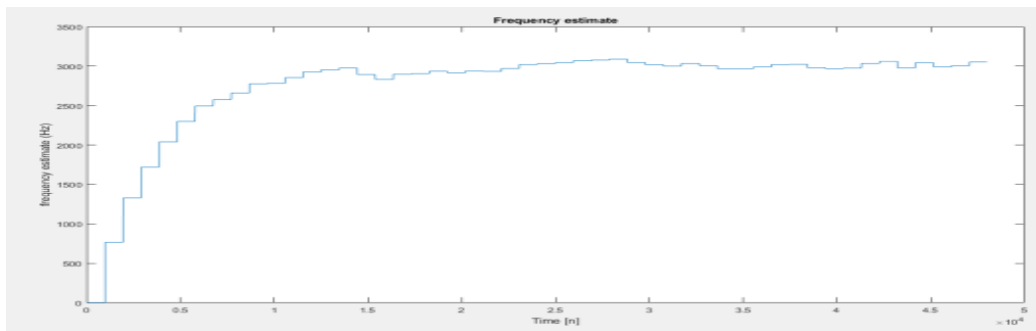


Figure 3.38 : Graphique d'estimation de fréquence (BW=10MHZ)

Dans le cas de paramètre bande passante égale a 10MHZ le Taux d'erreur a diminuée a 0.192%

5. BW= 15 Mhz

Avec le paramètre bande passante égale a 15 (BW=15MHZ) :

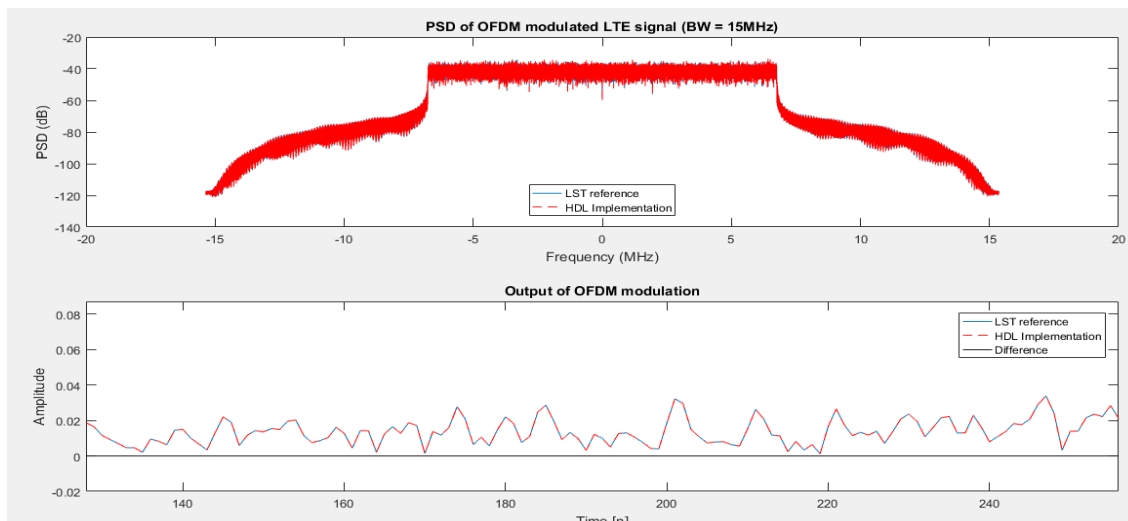


Figure 3.39 : Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM (BW=15MHZ)

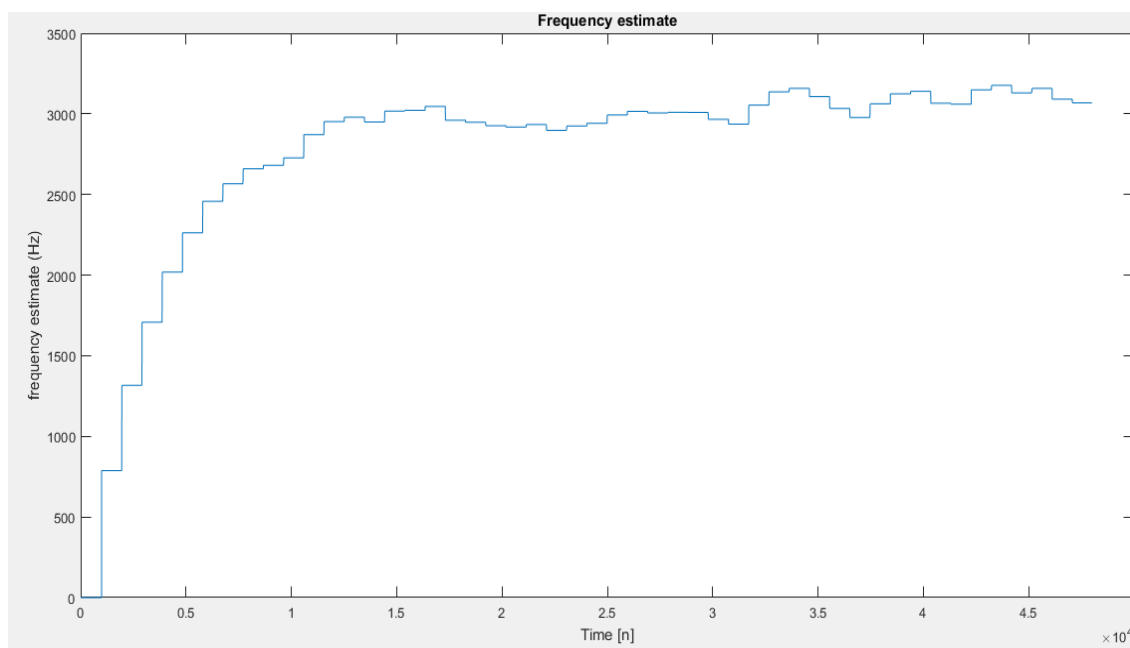


Figure 3.40 : Graphique d'estimation de fréquence (BW=10MHZ)

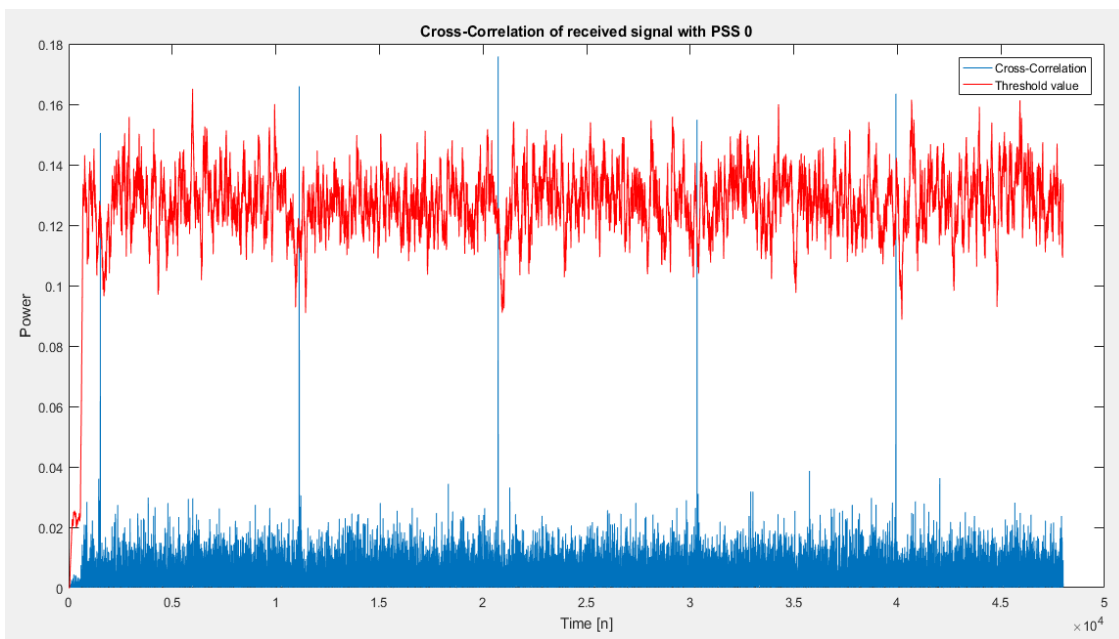


Figure 3.41 : Diagramme de corrélation croisée PSS (BW=15MHZ)

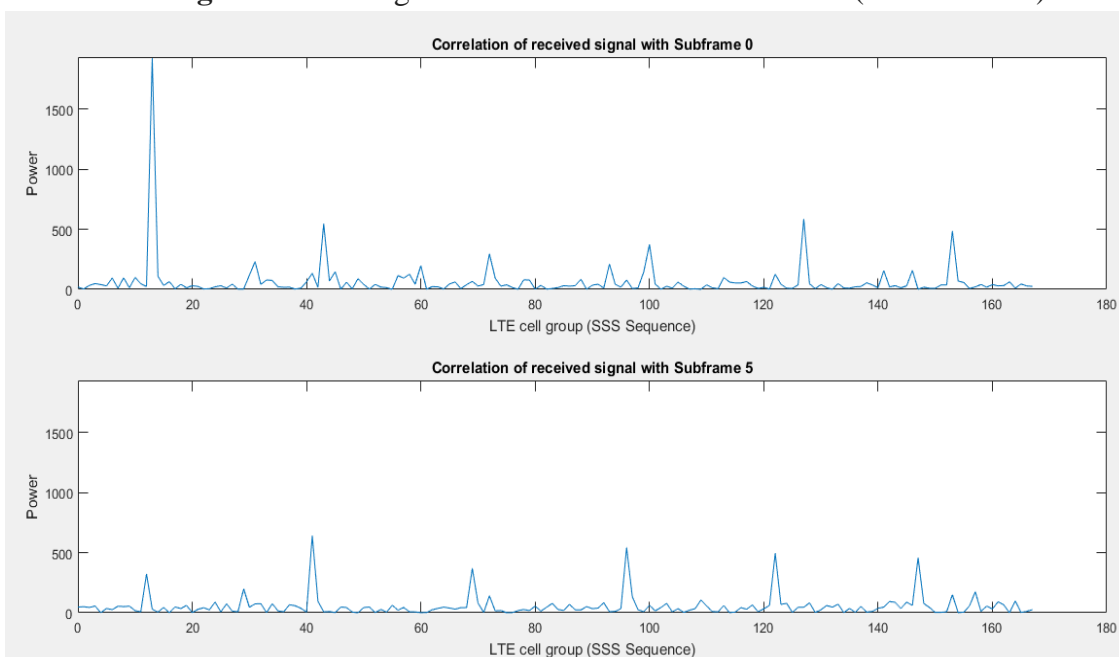


Figure 3.42 : Diagramme de corrélation croisée SSS (BW=15MHZ)

Dans le cas de paramètre bande passante égale a 15MHZ le Taux d'erreur (RMS) a diminuée a 0.06%

6. BW= 20 Mhz

Avec le paramètre bande passante égale a 20 (BW=20MHZ) :

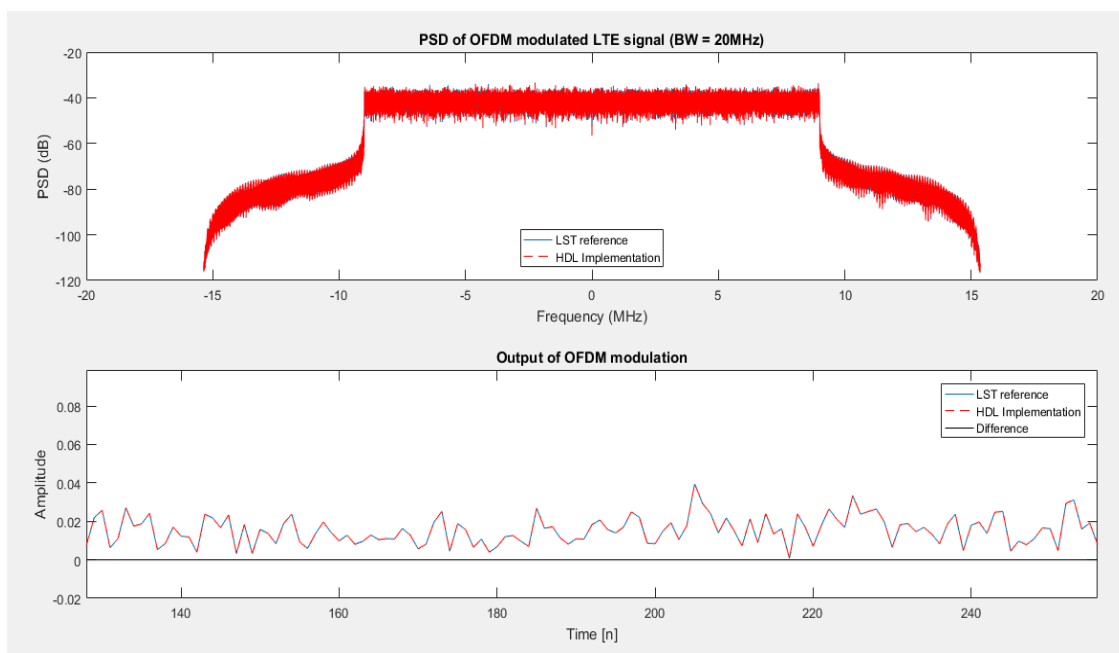


Figure 3.43 : Densité spectrale de puissance (PSD) et sortie de du Modulateur OFDM(BW=20MHZ)

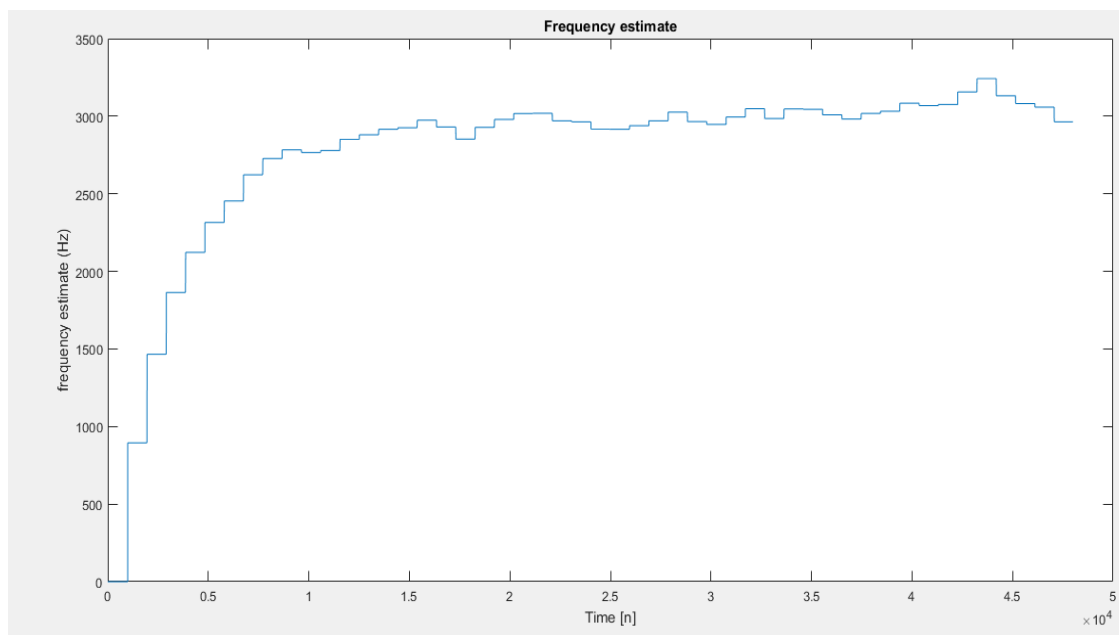


Figure 3.44 : Graphique d'estimation de fréquence (BW=20MHZ)

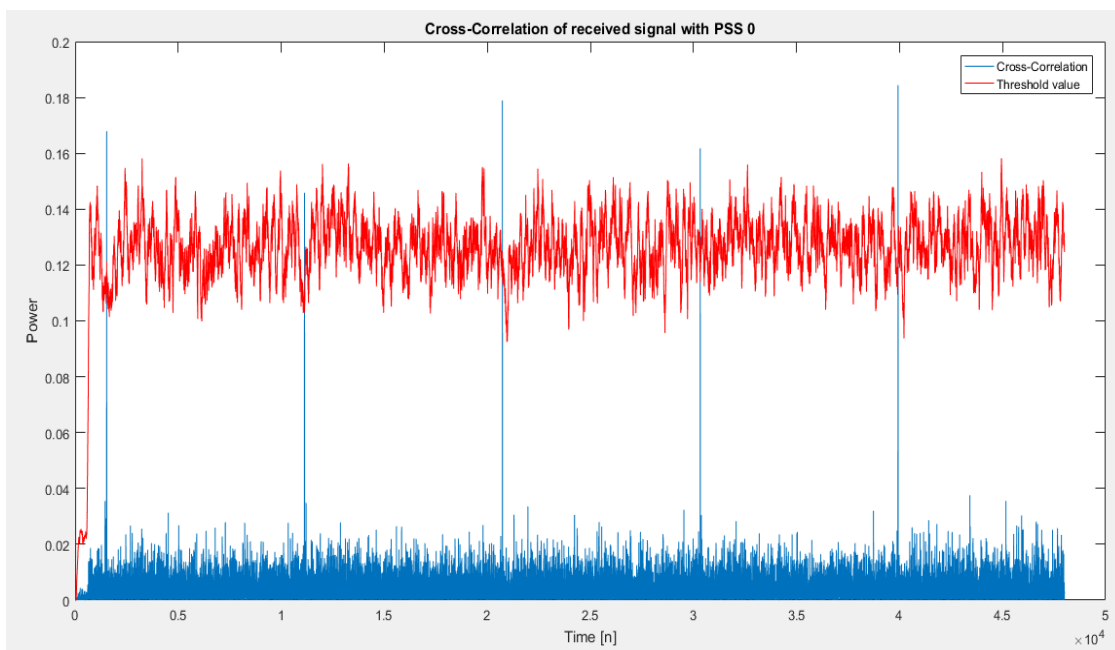


Figure 3.45 : Diagramme de corrélation croisée PSS (BW=20MHZ)

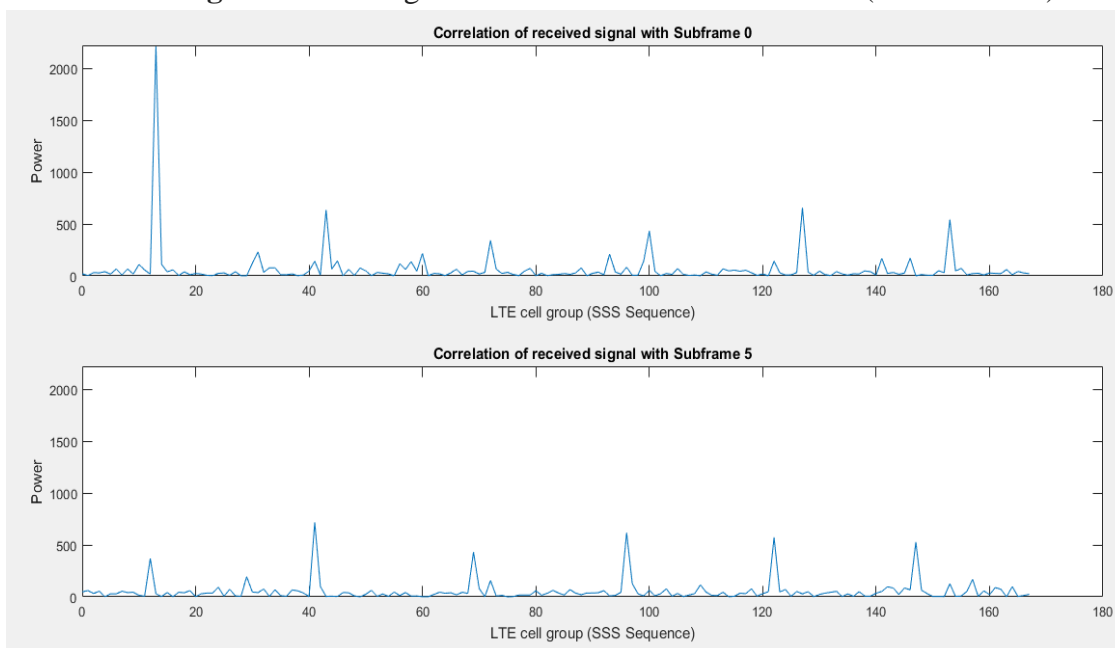


Figure 3.46 : Diagramme de corrélation croisée SSS (BW=20MHZ)

Dans le cas de paramètre bande passante égale a 20MHZ le Taux d'erreur a diminuée a 0.013%.

On observe qu'avec l'augmentation de la bande passante le taux d'erreur se diminue relativement avec cette dernière parce que plus on augmente la bande passante plus l'interférence entre symboles et entre canal adjacent se diminue.

Pour cela on peut dire que le meilleur résultat obtenu est celui le dernier avec une bande passante égale à 20MHz et ça à donner comme un taux d'erreur égale à 0.013%

3.9 Génération du code HDL du model modulateur /Détecteur OFDM LTE :

A L'aide de s fonctionnalité du Matlab/Simulink et leur interfaçage avec Xilinx pour la génération du code VHDL

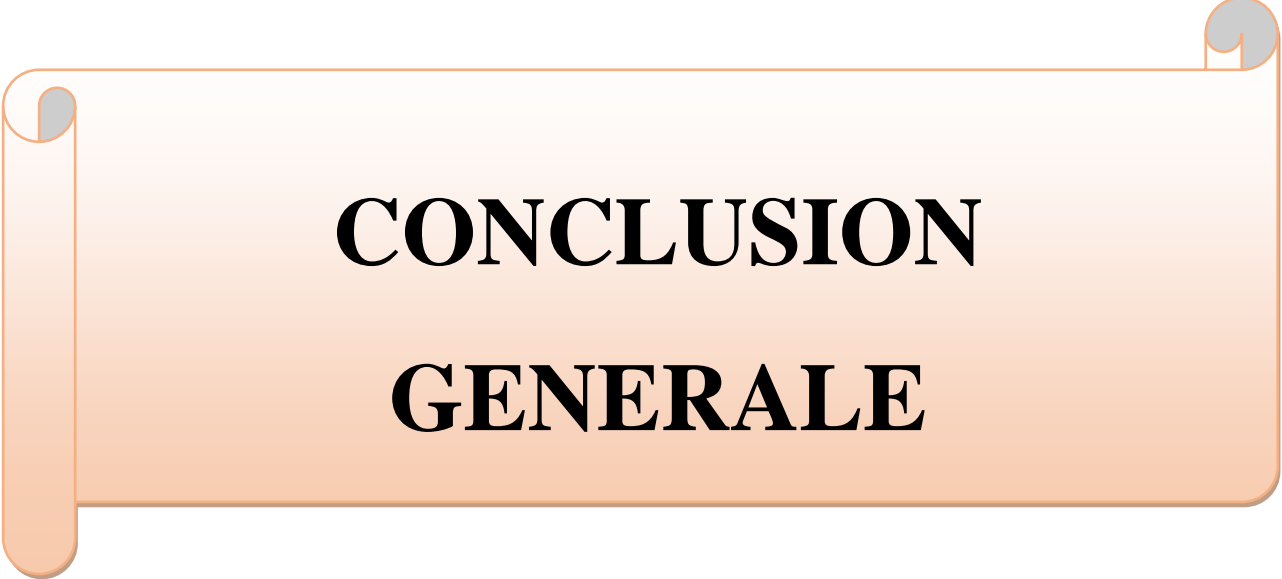
Le code VHDL générer est mentionné en détails dans l'annexe 1 pour le modulateur OFDM LTE et dans l'annexe 2 pour le détecteur OFDM LTE.

3.10 Conclusion

Dans ce chapitre, nous avons étudié et simulé l'implémentation HDL d'un modulateur et d'un détecteur OFDM LTE ainsi l'effet des paramètres de bande passante, SNR et atténuation du canal sur les performances d'un système de transmission OFDM LTE. L'objectif a été réalisé avec l'obtention d'une erreur négligeable entre l'implémentation HDL et signal de référence, la variation des paramètres de chaîne de transmission implique que la chaîne fonctionne correctement dans des environnements LOS et NLOS.

Les valeurs obtenues ont été évaluées en utilisant plusieurs paramètres de la chaîne de transmission mentionnée ci-dessus.

Nous avons aussi généré de code Vhdl du modulateur et du détecteur pour une future implémentation Matériels sur des cartes FPGA en utilisant la fonction FPGA in the loop.



**CONCLUSION
GENERALE**

Conclusion Générale

En communication mobile sans fils, la transmission des données et le processus de transmission sont affectés par les interférences et les évanouissements par trajets multiples. Cela réduit son efficacité spectrale et diminue le débit global. Le préfixe cyclique (CP) est utilisé comme des bandes de garde. L'amplitude d'un tel préfixe est de l'ordre de l'étalement du retard du signal. Il existe deux longueurs différentes pour un CP (4.7u sec) et un CP (16.67u sec), les FPGA via l'implémentation HDL peuvent être utilisés avec les systèmes OFDM LTE. L'utilisation de la transformée de Fourier rapide convertit le signal du domaine temporel en un signal du domaine fréquentiel, tandis que la transformée de Fourier rapide inverse effectue l'opération inverse avec un temps de traitement réduits a cause de complexité d'implémenttion HDL qui réduit les operation de calcule.

L'utilisation des parametres suivantes dans la structure d'implémentation optimise les resultat de calcule et minimise le taux d'erreurs entre le signal de sortie de l'implémentation HDL et celui du signal de reference.

- Le SNR prévu est de 10 dB avec variation d'étude canal de 5dB à 20dB.
- Bande passante de differente gamme 1.4, 3, 5, 10, 15,20 Mhz.
- Une parfaite synchronisation temporelle.
- Une parfaite synchronisation fréquentielle.
- Modulation M-QAM.
- Le canal est considéré comme invariant dans le temps
- Le rapoort de gain canal a été exploite de 0.2 a 0.9

Le resulats a confirmé qu'avec un SNR de 10 et un gain de 0.9 on obtient des resultat acceptable pour l'implémentation du système OFDM LTE en HDL

Le système generateur a été utilisé pour la conversion de la structure HDL Simulink en code code VHDL prete a l'utilisé pour FPGA via le système XILINX ISE pour une implémentation matériels rapide et efficace.

Le développement de la communication OFDM LTE 4G conduira à la technologie 5G et les opérateurs de télécommunications se concentrent sur la recherche et l'innovation pour la connectivité de prochaine génération. Ces systèmes fourniront aux utilisateurs des opérations à faible latence et un débit et une évolutivité accrus.



REFERENCES

[1] D. Roque et C. Siclet, « Performances of Weighted Cyclic Prefix OFDM with Low-Complexity Equalization », *IEEE Communications Letters*, vol. 17, no 3, 2013, p. 439-442 (DOI 10.1109/LCOMM.2013.011513.121997)

[3] Annick Le Glaunec ,Modulations Multiporteuses

[4] Elhadji Mansour Fall, "Conception D'un amplificateur RF Agile En CMOS Pour Les Futures Générations De La Téléphonie Mobile ", Université Québec, Mars 2012.

[5] T. S. Rappaport, *Wireless communications: principles and practice*: Prentice Hall PTR, 2002.

[6]: P. Duhamel and M. Vetterli. Fast fourier transforms: A tutorial review and a state of the art. *Signal Processing*, 19(4) :259-299, Apr. 1990.

[7]: Marceau Coupechoux Philippe Martins Vers les systemes radiomobilesde 4e generation De l'UMTS au LTE"

[8] : Farhi Nabila et Helaimia Souhaila , Etude et Simulation d'une Transmission de Type OFDM Pour Les Communications Sans Fil. 2016.

[9] : [COM11] comelec.enst.fr/hdl/hdl_histhorique.html. (2011). Consulté le juin 2011, sur comelec.enst.fr: <http://comelec.enst.fr/hdl.html>

[10]: S. J.E. Wilton. Architectures and Algorithms for Field Programmable Gate Arrays with Embedded Memory . Ph.D Thesis, University of Toronto, Canada, 1997.

[11] (2011). Récupéré sur [xilinx.com/](http://www.xilinx.com/): <http://www.xilinx.com>

[12] <http://www.univ-bpclermont.fr/FORMATIONS/Master/meam/cme/cme02.pdf> [archive] [PDF] p. 30-47

[13] : François Verdier, « Les circuits FPGA Concepts de base, architecture et applications », Université de Cergy-Pontoise Laboratoire ETIS - UMR CNRS 8051.

[14] : <https://www.xilinx.com/products/design-tools/coregen.html>

[15] Mohamed Bendada « Implantation d'algorithme de filtrage numérique sur FPGA (réseau de portes programmables) » Université Farhat Abbas de Sétif Algérie- Master électronique.

[16] Smith, G. R. (2010). *FPGA 101*. New York: ELSEVIER.

[16] D. J. Smith, "VHDL and Verilog compared and contrasted-plus modeled example written in VHDL, Verilog and C," in Proceedings of 33rd Design Automation Conference, 3-7 June 1996, New York, NY, USA, 1996, pp. 771-6

[6] Yvon Sosthène Yameogo ,"Etudes de Nouvelles Techniques D'estimation et D'égalisation de Canal Adaptées au Système SC-FDMA", Thèse doctorat ,Université de Rennes 2011.

[9]<http://www.univbpclermont.fr/FORMATIONS/Master/meam/cme/cme02.pdf> [archive] [PDF] p. 30-47

[13] <https://exponenta.ru/hdl-coder>

[15] [DONALD G. BAELEY : Design for Embedded Image Processing on FPGAs



ANNEXES

ANNEXE 1

Fichier : LTE_Modulator_HDL.vhd

```
1  FileName:hdl_prj\hdlsrc\hdlcoder_lteofdm_modDetect\LTE_Modulator_HDL.vhd
2  -- Created: 2020-10-03 19:28:05
3  -- Generated by MATLAB 9.0 and HDL Coder 3.8
4  -- Rate and ClockingDetails
5  -- Model base rate: 3.25521e-08
6  -- Target subsystem base rate: 3.25521e-08
7  -- ClockEnableSample Time
8  -- ce_out      3.25521e-08
9  -- Output Signal      ClockEnableSample Time
10 -- dataRequestce_out  3.25521e-08
11 -- dataOut_rece_out   3.25521e-08
12 -- dataOut_imce_out   3.25521e-08
13 -- validOutce_out    3.25521e-08
14 -- Module: LTE_Modulator_HDL
15 -- Source Path: hdlcoder_lteofdm_modDetect/LTE_Modulator_HDL
16 -- HierarchyLevel: 0
17 LIBRARY IEEE;
18 USE IEEE.std_logic_1164.ALL;
19 USE IEEE.numeric_std.ALL;
20 ENTITY LTE_Modulator_HDL IS
21 PORT(clk          : IN  std_logic;
22      reset         : IN  std_logic;
23      clk_enable    : IN  std_logic;
24      dataIn_re     : IN  std_logic_vector(15 DOWNTO 0); -- sfix16_En14
25      dataIn_im     : IN  std_logic_vector(15 DOWNTO 0); -- sfix16_En14
26      ce_out        : OUT std_logic;
27      dataRequest   : OUT std_logic;
28      dataOut_re    : OUT std_logic_vector(22 DOWNTO 0); -- sfix23_En21
29      dataOut_im    : OUT std_logic_vector(22 DOWNTO 0); -- sfix23_En21
30      validOut      : OUT std_logic
31      );
32 END LTE_Modulator_HDL;
33 ARCHITECTURE rtl OF LTE_Modulator_HDL IS
34 -- Component Declarations
35 COMPONENT OFDM_Symbol_Mapping
36 PORT(clk          : IN  std_logic;
37      reset         : IN  std_logic;
```

```

38 enb          : IN  std_logic;
39 dataIn_re   : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
40 dataIn_im   : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
41 dataRequest      : OUT std_logic;
42 dataOut_re  : OUT std_logic_vector(15 DOWNT0 0); -- sfix16_En14
43 dataOut_im  : OUT std_logic_vector(15 DOWNT0 0); -- sfix16_En14
44 validOut    : OUT std_logic
45 );
46 END COMPONENT;
47 COMPONENT FFT_Shift
48 PORT(clk    : IN  std_logic;
49 reset     : IN  std_logic;
50 enb       : IN  std_logic;
51 dataIn_re : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
52 dataIn_im: IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
53 validIn   : IN  std_logic;
54 dataOut_re: OUT std_logic_vector(15 DOWNT0 0); -- sfix16_En14
55 dataOut_im : OUT std_logic_vector(15 DOWNT0 0); -- sfix16_En14
56 validOut  : OUT std_logic
57 );
58 END COMPONENT;
59 COMPONENT IFFT
60 PORT(clk          : IN  std_logic;
61      reset        : IN  std_logic;
62 enb              : IN  std_logic;
63 dataIn_re   : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
64 dataIn_im   : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En14
65 validIn     : IN  std_logic;
66 dataOut_re: OUT std_logic_vector(22 DOWNT0 0); -- sfix23_En21
67 dataOut_im : OUT std_logic_vector(22 DOWNT0 0); -- sfix23_En21
68 validOut   : OUT std_logic
69 );
70 END COMPONENT;
71 COMPONENT CP_Extension_Windowing
72 PORT(clk    : IN  std_logic;
73      reset  : IN  std_logic;
74 enb       : IN  std_logic;
75 dataIn_re : IN  std_logic_vector(22 DOWNT0 0); -- sfix23_En21
76 dataIn_im : IN  std_logic_vector(22 DOWNT0 0); -- sfix23_En21
77 validIn   : IN  std_logic;

```



```

78 dataOut_re : OUT std_logic_vector(22 DOWNT0 0); -- sfix23_En21
79 dataOut_im : OUT std_logic_vector(22 DOWNT0 0); -- sfix23_En21
80 validOut      : OUT std_logic
81      );
82 END COMPONENT;
83 COMPONENT Filtering
84 PORT(clk          : IN std_logic;
85      reset        : IN std_logic;
86      enb          : IN std_logic;
87      dataIn_re    : IN std_logic_vector(22 DOWNT0 0); -- sfix23_En21
88      dataIn_im    : IN std_logic_vector(22 DOWNT0 0); -- sfix23_En21
89      dataOut_re   : OUT std_logic_vector(22 DOWNT0 0); -- sfix23_En21
90      dataOut_im   : OUT std_logic_vector(22 DOWNT0 0) -- sfix23_En21
91      );
92 END COMPONENT;
93 -- Component Configuration Statements
94 FOR ALL : OFDM_Symbol_Mapping
95   USE ENTITY work.OFDM_Symbol_Mapping(rtl);
96 FOR ALL : FFT_Shift
97   USE ENTITY work.FFT_Shift(rtl);
98 FOR ALL : IFFT
99   USE ENTITY work.IFFT(rtl);
100 FOR ALL : CP_Extension_Windowing
101   USE ENTITY work.CP_Extension_Windowing(rtl);
102 FOR ALL : Filtering
103   USE ENTITY work.Filtering(rtl);
104 -- Signals
105 SIGNAL enb          : std_logic;
106 SIGNAL dataIn_re_signed : signed(15 DOWNT0 0); -- sfix16_En14
107 SIGNAL dataIn_im_signed : signed(15 DOWNT0 0); -- sfix16_En14
108 SIGNAL Delay2_out1_re : signed(15 DOWNT0 0); -- sfix16_En14
109 SIGNAL Delay2_out1_im : signed(15 DOWNT0 0); -- sfix16_En14
110 SIGNAL OFDM_Symbol_Mapping_out1 : std_logic;
111 SIGNAL OFDM_Symbol_Mapping_out2_re : std_logic_vector(15 DOWNT0 0); -- ufix16
112 SIGNAL OFDM_Symbol_Mapping_out2_im : std_logic_vector(15 DOWNT0 0); -- ufix16
113 SIGNAL OFDM_Symbol_Mapping_out3 : std_logic;
114 SIGNAL delayMatch_reg : std_logic_vector(0 TO 9); -- ufix1 [10]
115 SIGNAL Delay4_out1 : std_logic;
116 SIGNAL FFT_Shift_out1_re : std_logic_vector(15 DOWNT0 0); -- ufix16
117 SIGNAL FFT_Shift_out1_im : std_logic_vector(15 DOWNT0 0); -- ufix16

```

```

118 SIGNAL FFT_Shift_out2          : std_logic;
119 SIGNAL IFFT_out1_re: std_logic_vector(22 DOWNT0 0); -- ufix23
120 SIGNAL IFFT_out1_im : std_logic_vector(22 DOWNT0 0); -- ufix23
121 SIGNAL IFFT_out2   : std_logic;
122 SIGNAL CP_Extension_Windowing_out1_re : std_logic_vector(22 DOWNT0 0); -- ufix23
123 SIGNAL CP_Extension_Windowing_out1_im : std_logic_vector(22 DOWNT0 0); -- ufix23
124 SIGNAL CP_Extension_Windowing_out2   : std_logic;
125 SIGNAL Filtering_out1_re : std_logic_vector(22 DOWNT0 0); -- ufix23
126 SIGNAL Filtering_out1_im : std_logic_vector(22 DOWNT0 0); -- ufix23
127 SIGNAL Filtering_out1_re_signed : signed(22 DOWNT0 0); -- sfix23_En21
128 SIGNAL Filtering_out1_im_signed : signed(22 DOWNT0 0); -- sfix23_En21
129 SIGNAL Delay3_out1_re: signed(22 DOWNT0 0); -- sfix23_En21
130 SIGNAL Delay3_out1_im : signed(22 DOWNT0 0); -- sfix23_En21
131 SIGNAL delayMatch1_reg : std_logic_vector(0 TO 11); -- ufix1 [12]
132 SIGNAL Delay6_out1          : std_logic;
133 BEGIN
134 -- <S3>/OFDM Symbol Mapping
135 u_OFDM_Symbol_Mapping : OFDM_Symbol_Mapping
136   PORT MAP(clk =>clk,
137            reset => reset,
138            enb =>clk_enable,
139            dataIn_re =>std_logic_vector(Delay2_out1_re), -- sfix16_En14
140            dataIn_im =>std_logic_vector(Delay2_out1_im), -- sfix16_En14
141            dataRequest => OFDM_Symbol_Mapping_out1,
142            dataOut_re => OFDM_Symbol_Mapping_out2_re, -- sfix16_En14
143            dataOut_im => OFDM_Symbol_Mapping_out2_im, -- sfix16_En14
144            validOut => OFDM_Symbol_Mapping_out3
145            );
146 -- <S3>/FFT Shift
147 u_FFT_Shift : FFT_Shift
148   PORT MAP(clk =>clk,
149            reset => reset,
150            enb =>clk_enable,
151            dataIn_re => OFDM_Symbol_Mapping_out2_re, -- sfix16_En14
152            dataIn_im => OFDM_Symbol_Mapping_out2_im, -- sfix16_En14
153            validIn => OFDM_Symbol_Mapping_out3,
154            dataOut_re => FFT_Shift_out1_re, -- sfix16_En14
155            dataOut_im => FFT_Shift_out1_im, -- sfix16_En14
156            validOut => FFT_Shift_out2
157            );

```

```

158 -- <S3>/IFFT
159 u_IFFT : IFFT
160   PORT MAP(clk =>clk,
161           reset => reset,
162           enb =>clk_enable,
163           dataIn_re => FFT_Shift_out1_re, -- sfix16_En14
164           dataIn_im => FFT_Shift_out1_im, -- sfix16_En14
165           validIn => FFT_Shift_out2,
166           dataOut_re => IFFT_out1_re, -- sfix23_En21
167           dataOut_im => IFFT_out1_im, -- sfix23_En21
168           validOut => IFFT_out2
169           );
170 -- <S3>/CP_Extension_Windowing
171 u_CP_Extension_Windowing : CP_Extension_Windowing
172   PORT MAP(clk =>clk,
173           reset => reset,
174           enb =>clk_enable,
175           dataIn_re => IFFT_out1_re, -- sfix23_En21
176           dataIn_im => IFFT_out1_im, -- sfix23_En21
177           validIn => IFFT_out2,
178           dataOut_re => CP_Extension_Windowing_out1_re, -- sfix23_En21
179           dataOut_im => CP_Extension_Windowing_out1_im, -- sfix23_En21
180           validOut => CP_Extension_Windowing_out2
181           );
182 -- <S3>/Filtering
183 u_Filtering : Filtering
184   PORT MAP(clk =>clk,
185           reset => reset,
186           enb =>clk_enable,
187           dataIn_re => CP_Extension_Windowing_out1_re, -- sfix23_En21
188           dataIn_im => CP_Extension_Windowing_out1_im, -- sfix23_En21
189           dataOut_re => Filtering_out1_re, -- sfix23_En21
190           dataOut_im => Filtering_out1_im -- sfix23_En21
191           );
192   dataIn_re_signed<= signed(dataIn_re);
193   dataIn_im_signed<= signed(dataIn_im);
194   enb<= clk_enable;
196 -- <S3>/Delay2
197   Delay2_process : PROCESS (clk)
198   BEGIN

```

```

199   IF clk'EVENT AND clk = '1' THEN
200     IF reset = '1' THEN
201       Delay2_out1_re <= to_signed(16#0000#, 16);
202       Delay2_out1_im <= to_signed(16#0000#, 16);
203     ELSIF enb = '1' THEN
204       Delay2_out1_re <= dataIn_re_signed;
205       Delay2_out1_im <= dataIn_im_signed;
206     END IF;
207   END IF;
208 END PROCESS Delay2_process;
209 -- <S3>/Delay4
210 delayMatch_process : PROCESS (clk)
211 BEGIN
212   IF clk'EVENT AND clk = '1' THEN
213     IF reset = '1' THEN
214       delayMatch_reg<= (OTHERS => '0');
215     ELSIF enb = '1' THEN
216       delayMatch_reg(0) <= OFDM_Symbol_Mapping_out1;
217       delayMatch_reg(1 TO 9) <= delayMatch_reg(0 TO 8);
218     END IF;
219   END IF;
220 END PROCESS delayMatch_process;
221 Delay4_out1 <= delayMatch_reg(9);
222 Filtering_out1_re_signed <= signed(Filtering_out1_re);
223 Filtering_out1_im_signed <= signed(Filtering_out1_im);
224 -- <S3>/Delay3
225 Delay3_process : PROCESS (clk)
226 BEGIN
227   IF clk'EVENT AND clk = '1' THEN
228     IF reset = '1' THEN
229       Delay3_out1_re <= to_signed(16#000000#, 23);
230       Delay3_out1_im <= to_signed(16#000000#, 23);
231     ELSIF enb = '1' THEN
232       Delay3_out1_re <= Filtering_out1_re_signed;
233       Delay3_out1_im <= Filtering_out1_im_signed;
234     END IF;
235   END IF;
236 END PROCESS Delay3_process;
237 dataOut_re<= std_logic_vector(Delay3_out1_re);
238 dataOut_im<= std_logic_vector(Delay3_out1_im);

```

```
239 -- <S3>/Delay6
240 -- <S3>/Delay5
241 delayMatch1_process : PROCESS (clk)
242 BEGIN
243   IF clk'EVENT AND clk = '1' THEN
244     IF reset = '1' THEN
245       delayMatch1_reg <= (OTHERS => '0');
246     ELSIF enb = '1' THEN
247       delayMatch1_reg(0) <= CP_Extension_Windowing_out2;
248     delayMatch1_reg(1 TO 11) <= delayMatch1_reg(0 TO 10);
249     END IF;
250   END IF;
251 END PROCESS delayMatch1_process;
252 Delay6_out1 <= delayMatch1_reg(11);
253 ce_out <= clk_enable;
254 dataRequest <= Delay4_out1;
255 validOut <= Delay6_out1;
256 END rtl;
```

ANNEXE 2

Fichier : LTE_Detector_HDL.vhd

```
1 -- File Name:
2 hdl_prj\hdlsrc\hdlcoder_lteofdm_modDetect\LTE_Detector_HDL.vhd
3 -- Created: 2020-10-03 19:06:29
4 -- Generated by MATLAB 9.0 and HDL Coder 3.8
5
6 -- Rate and ClockingDetails
7 -- Model base rate: 8.13802e-09
8 Target subsystem base rate: 1.30208e-07
9 -- ClockEnableSample Time
10 -- ce_out      5.20833e-07
11
12 Output Signal      ClockEnableSample Time
13 -- dataOutce_out  5.20833e-07
14 -- validOutce_out 5.20833e-07
15 peakCorrValce_out 5.20833e-07
16 -- primCellIDce_out 5.20833e-07
17 -- offset          ce_out  5.20833e-07
18 -- XcorrOutce_out  5.20833e-07
19 -- thresholdOutce_out 5.20833e-07
20 -- secCellIDce_out  5.20833e-07
21 -- cellIDce_out    5.20833e-07
22 -- freqEstce_out   5.20833e-07
23 -- CPCorrMagce_out 5.20833e-07
24 -- Module: LTE_Detector_HDL
25 -- Module: LTE_Detector_HDL
26 -- HierarchyLevel: 0
27 LIBRARY IEEE;
28 USE IEEE.std_logic_1164.ALL;
29 USE IEEE.numeric_std.ALL;
30 USE work.LTE_Detector_HDL_pkg.ALL;
31 ENTITY LTE_Detector_HDL IS
32 PORT(clk          : IN  std_logic;
33      reset         : IN  std_logic;
34      clk_enable    : IN  std_logic;
35      dataIn_re     : IN  std_logic_vector(15 DOWNTO 0); -- sfix16_En14
36      dataIn_im     : IN  std_logic_vector(15 DOWNTO 0); -- sfix16_En14
37      ce_out        : OUT std_logic;
38      dataOut       : OUT std_logic_vector(35 DOWNTO 0); -- sfix36_En8
39      validOut      : OUT std_logic;
```

```

37 peakCorrVal: OUT std_logic_vector(31 DOWNTO 0); -- sfix32_En20
38 primCellID : OUT std_logic_vector(15 DOWNTO 0); -- uint16
39 offset : OUT std_logic_vector(14 DOWNTO 0); -- ufix15
40 XcorrOut : OUT std_logic_vector(31 DOWNTO 0); -- sfix32_En20
41 thresholdOut :OUT std_logic_vector(31 DOWNTO 0); -- sfix32_En20
42 secCellID: OUT std_logic_vector(7 DOWNTO 0); -- uint8
43 cellID : OUT std_logic_vector(8 DOWNTO 0); -- ufix9
44 freqEst : OUT std_logic_vector(26 DOWNTO 0); -- sfix27_En27
45 CPCorrMag : OUT std_logic_vector(24 DOWNTO 0) -- sfix25_En23
46 );
47 END LTE_Detector_HDL;
48 rtl OF LTE_Detector_HDL IS
49 -- Component Declarations
50 COMPONENT LTE_Detector_HDL_tc
51 PORT(clk : IN std_logic;
52 reset : IN std_logic;
53 clk_enable : IN std_logic;
54 enb_64_16_1 : OUT std_logic;
55 enb_1_16_0 : OUT std_logic;
56 enb_1_16_1 : OUT std_logic
57 );
58 END COMPONENT;
59 COMPONENT Frequency_Estimation
60 PORT(clk : IN std_logic;
61 reset : IN std_logic;
62 enb_1_16_0 : IN std_logic;
63 data_re : IN std_logic_vector(15 DOWNTO 0); -- sfix16_En14
64 data_im : IN std_logic_vector(15 DOWNTO 0); -- sfix16_En14
65 freqEst : OUT std_logic_vector(26 DOWNTO 0); -- sfix27_En27
66 CPCorrMag : OUT std_logic_vector(24 DOWNTO 0) -- sfix25_En23
67 );
68 END COMPONENT;
69 COMPONENT Frequency_Correction
70 PORT(clk : IN std_logic;
71 reset : IN std_logic;
72 enb_1_16_0 : IN std_logic;
73 dataIn_re : IN std_logic_vector(15 DOWNTO 0); -- sfix16_En14
74 dataIn_im : IN std_logic_vector(15 DOWNTO 0); -- sfix16_En14
75 freq : IN std_logic_vector(26 DOWNTO 0); -- sfix27_En27
76 dataOut_re : OUT std_logic_vector(15 DOWNTO 0); -- sfix16_En15

```

```

77     dataOut_im : OUT std_logic_vector(15 DOWNT0 0) -- sfix16_En15
78 );
79 END COMPONENT;
80 COMPONENT PSS_Detection
81 PORT(clk          : IN  std_logic;
82      reset         : IN  std_logic;
83      enb_1_16_0    : IN  std_logic;
84      enb_64_16_1   : IN  std_logic;
85      dataIn_re : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
86      dataIn_im : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
87      Detected    : OUT  std_logic;
88      Val         : OUT  std_logic_vector(31 DOWNT0 0); -- sfix32_En20
89      CellID      : OUT  std_logic_vector(15 DOWNT0 0); -- uint16
90      Offset      : OUT  std_logic_vector(14 DOWNT0 0); -- ufix15
91      XcorrOut    : OUT  std_logic_vector(31 DOWNT0 0); -- sfix32_En20
92      thresholdOut : OUT  std_logic_vector(31 DOWNT0 0) -- sfix32_En20
93 );
94 END COMPONENT;
95 COMPONENT Timing_Adjustment
96 PORT(clk          : IN  std_logic;
97      reset         : IN  std_logic;
98      enb_1_16_0    : IN  std_logic;
99      dataIn_re : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
100     dataIn_im : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
101     PSSDetected : IN  std_logic;
102     dataOut_re : OUT  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
103     dataOut_im : OUT  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
104     validOut   : OUT  std_logic
105 );
106 END COMPONENT;
107 COMPONENT FFT
108 PORT(clk          : IN  std_logic;
109      reset         : IN  std_logic;
110     enb_1_16_0    : IN  std_logic;
111     dataIn_re : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
112     dataIn_im : IN  std_logic_vector(15 DOWNT0 0); -- sfix16_En15
113     validIn    : IN  std_logic;
114     dataOut_re : OUT  std_logic_vector(17 DOWNT0 0); -- sfix18_En10
115     dataOut_im : OUT  std_logic_vector(17 DOWNT0 0); -- sfix18_En10
116     validOut   : OUT  std_logic

```



```

117 );
118 END COMPONENT;
119 COMPONENT SSS_Detection
120 PORT(clk          : IN  std_logic;
121      reset        : IN  std_logic;
122      enb_1_16_0   : IN  std_logic;
123      receiveSamples_re: IN std_logic_vector(17 DOWNTO 0); -- sfix18_En10
124      receiveSamples_im :IN std_logic_vector(17 DOWNTO 0); -- sfix18_En10
125      enb          : IN  std_logic;
126      pCellID      : IN  std_logic_vector(15 DOWNTO 0); -- uint16
127      dataOut      : OUT std_logic_vector(35 DOWNTO 0); -- sfix36_En8
128      currentCellGroup : OUT std_logic_vector(7 DOWNTO 0); -- uint8
129      positionInGroup : OUT std_logic_vector(15 DOWNTO 0); -- uint16
130      validOut     : OUT std_logic
131 );
132 END COMPONENT;
133 COMPONENT Determine_Cell_ID
134 PORT(clk          : IN  std_logic;
135      reset        : IN  std_logic;
136      enb_1_16_0   : IN  std_logic;
137      SSSDotProd   : IN  std_logic_vector(35 DOWNTO 0); -- sfix36_En8
138      currentCellGroup : IN  std_logic_vector(7 DOWNTO 0); -- uint8
139      positionInGroup : IN  std_logic_vector(15 DOWNTO 0); -- uint16
140      SSSDotProdOut : OUT std_logic_vector(35 DOWNTO 0); -- sfix36_En8
141      cellGroup     : OUT std_logic_vector(7 DOWNTO 0); -- uint8
142      cellID        : OUT std_logic_vector(8 DOWNTO 0) -- ufix9
143 );
144 END COMPONENT;
145 -- Component Configuration Statements
146 FOR ALL : LTE_Detector_HDL_tc
147 USE ENTITY work.LTE_Detector_HDL_tc(rtl);
148 FOR ALL : Frequency_Estimation
149 USE ENTITY work.Frequency_Estimation(rtl);
150 FOR ALL : Frequency_Correction
151 USE ENTITY work.Frequency_Correction(rtl);
152 FOR ALL : PSS_Detection
153 USE ENTITY work.PSS_Detection(rtl);
154 FOR ALL : Timing_Adjustment
155 USE ENTITY work.Timing_Adjustment(rtl);
156 FOR ALL : FFT

```

```

157 USE ENTITY work.FFT(rtl);
158 FOR ALL : SSS_Detection
159 USE ENTITY work.SSS_Detection(rtl);
160 FOR ALL : Determine_Cell_ID
161 USE ENTITY work.Determine_Cell_ID(rtl);
162 – Signals
163 SIGNAL enb_1_16_0          : std_logic;
164 SIGNAL enb_64_16_1        : std_logic;
165 enb_1_16_1                : std_logic;
166 SIGNAL Frequency_Estimation_out1 : std_logic_vector(26 DOWNT0 0); -- ufix27
167 SIGNAL Frequency_Estimation_out2 : std_logic_vector(24 DOWNT0 0); -- ufix25
168 SIGNAL Frequency_Correction_out1_re : std_logic_vector(15 DOWNT0 0); -- ufix16
169 SIGNAL Frequency_Correction_out1_im : std_logic_vector(15 DOWNT0 0); -- ufix16
170 SIGNAL Frequency_Correction_out1_re_signed : signed(15 DOWNT0 0); -- sfix16_En15
171 SIGNAL Frequency_Correction_out1_im_signed : signed(15 DOWNT0 0); -- sfix16_En15
172 SIGNAL Delay9_reg_re : vector_of_signed16(0 TO 29);-- sfix16_En15 [30]
173 SIGNAL Delay9_reg_im : vector_of_signed16(0 TO 29);-- sfix16_En15 [30]
174 SIGNAL Delay9_out1_re : signed(15 DOWNT0 0); -- sfix16_En15
175 SIGNAL Delay9_out1_im : signed(15 DOWNT0 0); -- sfix16_En15
176 SIGNAL PSS_Detection_out1 : std_logic;
177 SIGNAL PSS_Detection_out2 : std_logic_vector(31 DOWNT0 0); -- ufix32
178 SIGNAL PSS_Detection_out3 : std_logic_vector(15 DOWNT0 0); -- ufix16
179 SIGNAL PSS_Detection_out4 : std_logic_vector(14 DOWNT0 0); -- ufix15
180 SIGNAL PSS_Detection_out5 : std_logic_vector(31 DOWNT0 0); -- ufix32
181 SIGNAL PSS_Detection_out6 : std_logic_vector(31 DOWNT0 0); -- ufix32
182 SIGNAL Timing_Adjustment_out1_re : std_logic_vector(15 DOWNT0 0); -- ufix16
183 SIGNAL Timing_Adjustment_out1_im : std_logic_vector(15 DOWNT0 0); -- ufix16
184 SIGNAL Timing_Adjustment_out2 : std_logic;
185 SIGNAL FFT_out1_re : std_logic_vector(17 DOWNT0 0); -- ufix18
186 SIGNAL FFT_out1_im : std_logic_vector(17 DOWNT0 0); -- ufix18
187 SIGNAL FFT_out2 : std_logic;
188 SIGNAL PSS_Detection_out3_unsigned : unsigned(15 DOWNT0 0); -- uint16
189 SIGNAL Delay5_out1 : unsigned(15 DOWNT0 0); -- uint16
190 SIGNAL SSS_Detection_out1 : std_logic_vector(35 DOWNT0 0); -- ufix36
191 SIGNAL SSS_Detection_out2 : std_logic_vector(7 DOWNT0 0); -- ufix8
192 SIGNAL SSS_Detection_out3 : std_logic_vector(15 DOWNT0 0); -- ufix16
193 SIGNAL SSS_Detection_out4 : std_logic;
194 SIGNAL Determine_Cell_ID_out1 : std_logic_vector(35 DOWNT0 0); -- ufix36
195 SIGNAL Determine_Cell_ID_out2 : std_logic_vector(7 DOWNT0 0); -- ufix8
196 SIGNAL Determine_Cell_ID_out3 : std_logic_vector(8 DOWNT0 0); -- ufix9

```

```

197 SIGNAL Determine_Cell_ID_out1_signed : signed(35 DOWNT0 0); -- sfix36_En8
198 SIGNAL Delay3_out1 : signed(35 DOWNT0 0); -- sfix36_En8
199 SIGNAL Delay2_reg : std_logic_vector(0 TO 1); -- ufix1 [2]
200 SIGNAL Delay2_out1 : std_logic;
201 SIGNAL PSS_Detection_out2_signed : signed(31 DOWNT0 0); -- sfix32_En20
202 SIGNAL Delay4_out1 : signed(31 DOWNT0 0); -- sfix32_En20
203 SIGNAL PSS_Detection_out4_unsigned : unsigned(14 DOWNT0 0); -- ufix15
204 SIGNAL Delay6_out1 : unsigned(14 DOWNT0 0); -- ufix15
205 SIGNAL PSS_Detection_out5_signed : signed(31 DOWNT0 0); -- sfix32_En20
206 SIGNAL Delay7_out1 : signed(31 DOWNT0 0); -- sfix32_En20
207 SIGNAL PSS_Detection_out6_signed : signed(31 DOWNT0 0); -- sfix32_En20
208 SIGNAL delayMatch_reg : vector_of_signed32(0 TO 2); -- sfix32 [3]
209 SIGNAL Delay8_out1 : signed(31 DOWNT0 0); -- sfix32_En20
210 SIGNAL Determine_Cell_ID_out2_unsigned : unsigned(7 DOWNT0 0); -- uint8
211 SIGNAL Delay11_out1 : unsigned(7 DOWNT0 0); -- uint8*
212 SIGNAL Determine_Cell_ID_out3_unsigned : unsigned(8 DOWNT0 0); -- ufix9
213 SIGNAL Delay10_out1 : unsigned(8 DOWNT0 0); -- ufix9
214 SIGNAL Frequency_Estimation_out1_signed : signed(26 DOWNT0 0); -- sfix27_En27
215 SIGNAL delayMatch1_reg : vector_of_signed27(0 TO 4); -- sfix27 [5]
216 SIGNAL Frequency_Estimation_out1_1: signed(26 DOWNT0 0); -- sfix27_En27
217 SIGNAL Frequency_Estimation_out2_signed : signed(24 DOWNT0 0); -- sfix25_En23
218 SIGNAL delayMatch2_reg : vector_of_signed25(0 TO 4); -- sfix25 [5]
219 SIGNAL Frequency_Estimation_out2_1: signed(24 DOWNT0 0); -- sfix25_En23
220 BEGIN
221 u_LTE_Detector_HDL_tc : LTE_Detector_HDL_tc
222 PORT MAP(clk =>clk,
223 reset => reset,
224 clk_enable =>clk_enable,
225 enb_64_16_1 => enb_64_16_1,
226 enb_1_16_0 => enb_1_16_0,
227 enb_1_16_1 => enb_1_16_1
228 );
229 -- <S2>/Frequency_Estimation
230 u_Frequency_Estimation : Frequency_Estimation
231 PORT MAP(clk =>clk,
232 reset => reset,
233 enb_1_16_0 => enb_1_16_0,
234 data_re =>dataIn_re, -- sfix16_En14
235 data_im =>dataIn_im, -- sfix16_En14
236 freqEst => Frequency_Estimation_out1, -- sfix27_En27

```

```

237   CPCorrMag => Frequency_Estimation_out2 -- sfix25_En23
238   );
239   -- <S2>/Frequency_Correction
240   u_Frequency_Correction : Frequency_Correction
241   PORT MAP(clk =>clk,
242   reset => reset,
243   enb_1_16_0 => enb_1_16_0,
244   dataIn_re =>dataIn_re, -- sfix16_En14
245   dataIn_im =>dataIn_im, -- sfix16_En14
246   freq => Frequency_Estimation_out1, -- sfix27_En27
247   dataOut_re => Frequency_Correction_out1_re, -- sfix16_En15
248   dataOut_im => Frequency_Correction_out1_im -- sfix16_En15
249   );
250   -- <S2>/PSS_Detection
251   u_PSS_Detection : PSS_Detection
252   PORT MAP(clk =>clk,
253   reset => reset,
254   enb_1_16_0 => enb_1_16_0,
255   enb_64_16_1 => enb_64_16_1,
256   dataIn_re => Frequency_Correction_out1_re, -- sfix16_En15
257   dataIn_im => Frequency_Correction_out1_im, -- sfix16_En15
258   Detected => PSS_Detection_out1,
259   Val => PSS_Detection_out2, -- sfix32_En20
260   CellID => PSS_Detection_out3, -- uint16
261   Offset => PSS_Detection_out4, -- ufix15
262   XcorrOut => PSS_Detection_out5, -- sfix32_En20
263   thresholdOut => PSS_Detection_out6 -- sfix32_En20
264   );
265   -- <S2>/Timing_Adjustment
266   u_Timing_Adjustment : Timing_Adjustment
267   PORT MAP(clk =>clk,
268   reset => reset,
269   enb_1_16_0 => enb_1_16_0,
270   dataIn_re =>std_logic_vector(Delay9_out1_re), -- sfix16_En15
271   dataIn_im =>std_logic_vector(Delay9_out1_im), -- sfix16_En15
272   PSSDetected => PSS_Detection_out1,
273   dataOut_re => Timing_Adjustment_out1_re, -- sfix16_En15
274   dataOut_im => Timing_Adjustment_out1_im, -- sfix16_En15
275   validOut => Timing_Adjustment_out2);
276   -- <S2>/FFT

```

```

277     u_FFT : FFT
278     PORT MAP(clk =>clk,
279     reset => reset,
280     enb_1_16_0 => enb_1_16_0,
281     dataIn_re => Timing_Adjustment_out1_re, -- sfix16_En15
282     dataIn_im => Timing_Adjustment_out1_im, -- sfix16_En15
283     validIn => Timing_Adjustment_out2,
284     dataOut_re => FFT_out1_re, -- sfix18_En10
285     dataOut_im => FFT_out1_im, -- sfix18_En10
286     validOut => FFT_out2
287 );
288 -- <S2>/SSS_Detection
289 u_SSS_Detection : SSS_Detection
290 PORT MAP(clk =>clk,
291 reset => reset,
292 enb_1_16_0 => enb_1_16_0,
293 receiveSamples_re => FFT_out1_re, -- sfix18_En10
294 receiveSamples_im => FFT_out1_im, -- sfix18_En10
295 enb => FFT_out2,
296 pCellID =>std_logic_vector(Delay5_out1), -- uint16
297 dataOut => SSS_Detection_out1, -- sfix36_En8
298 currentCellGroup => SSS_Detection_out2, -- uint8
299 positionInGroup => SSS_Detection_out3, -- uint16
300 validOut => SSS_Detection_out4
301 );
302 -- <S2>/Determine_Cell_ID
303 u_Determine_Cell_ID : Determine_Cell_ID
304 PORT MAP(clk =>clk,
305 reset => reset,
306 enb_1_16_0 => enb_1_16_0,
307 SSSDotProd => SSS_Detection_out1, -- sfix36_En8
308 currentCellGroup => SSS_Detection_out2, -- uint8
309 positionInGroup => SSS_Detection_out3, -- uint16
310 SSSDotProdOut => Determine_Cell_ID_out1, -- sfix36_En8
311 cellGroup => Determine_Cell_ID_out2, -- uint8
312 cellID => Determine_Cell_ID_out3 -- ufix9
313 );
314 Frequency_Correction_out1_re_signed <= signed(Frequency_Correction_out1_re);
315 Frequency_Correction_out1_im_signed <= signed(Frequency_Correction_out1_im);
316 -- <S2>/Delay9

```

```

317 Delay9_process : PROCESS (clk)
318 BEGIN
319 IF clk'EVENT AND clk = '1' THEN
320 IF reset = '1' THEN
321 Delay9_reg_re <= (OTHERS =>to_signed(16#0000#, 16));
322 Delay9_reg_im <= (OTHERS =>to_signed(16#0000#, 16));
323 ELSIF enb_1_16_0 = '1' THEN
324 Delay9_reg_im(0) <= Frequency_Correction_out1_im_signed;
325 Delay9_reg_im(1 TO 29) <= Delay9_reg_im(0 TO 28);
326 Delay9_reg_re(0) <= Frequency_Correction_out1_re_signed;
327 Delay9_reg_re(1 TO 29) <= Delay9_reg_re(0 TO 28);
328 END IF ;
329 END IF ;
330 END PROCESS Delay9_process ;
331 Delay9_out1_re <= Delay9_reg_re(29);
332 Delay9_out1_im <= Delay9_reg_im(29);
333 PSS_Detection_out3_unsigned <= unsigned(PSS_Detection_out3);
334 -- <S2>/Delay5
335 Delay5_process : PROCESS (clk)
336 BEGIN
337 IF clk'EVENT AND clk = '1' THEN
338 IF reset = '1' THEN
339 Delay5_out1 <= to_unsigned (16#0000#, 16);
340 ELSIF enb_1_16_0 = '1' THEN
341 Delay5_out1 <= PSS_Detection_out3_unsigned;
342 END IF;
343 END IF;
344 END PROCESS Delay5_process;
345 Determine_Cell_ID_out1_signed <= signed(Determine_Cell_ID_out1);
346 -- <S2>/Delay3
347 Delay3_process : PROCESS (clk)
348 BEGIN
349 IF clk'EVENT AND clk = '1' THEN
350 IF reset = '1' THEN
351 Delay3_out1 <= to_signed(0, 36);
352 ELSIF enb_1_16_0 = '1' THEN
353 Delay3_out1 <= Determine_Cell_ID_out1_signed;
354 END IF;
355 END IF;
356 END PROCESS Delay3_process ;

```

```

357     dataOut<= std_logic_vector(Delay3_out1);
358     -- <S2>/Delay2
359     -- <S2>/Delay1
360     Delay2_process : PROCESS (clk)
361     BEGIN
362     IF clk'EVENT AND clk = '1' THEN
363     IF reset = '1' THEN
364     Delay2_reg <= (OTHERS => '0');
365     ELSIF enb_1_16_0 = '1' THEN
366     Delay2_reg(0) <= SSS_Detection_out4;
367     Delay2_reg(1) <= Delay2_reg(0);
368     END IF;
369     END IF;
370     END PROCESS Delay2_process;
371     Delay2_out1 <= Delay2_reg(1);
372     PSS_Detection_out2_signed <= signed(PSS_Detection_out2);
373     -- <S2>/Delay4
374     Delay4_process : PROCESS (clk)
375     BEGIN
376     IF clk'EVENT AND clk = '1' THEN
377     IF reset = '1' THEN
378     Delay4_out1 <= to_signed(0, 32);
379     ELSIF enb_1_16_0 = '1' THEN
380     Delay4_out1 <= PSS_Detection_out2_signed;
381     END IF;
382     END IF;
383     END PROCESS Delay4_process;
384     peakCorrVal<= std_logic_vector(Delay4_out1);
385     primCellID<= std_logic_vector(Delay5_out1);
386     PSS_Detection_out4_unsigned <= unsigned(PSS_Detection_out4);
387     -- <S2>/Delay6
388     Delay6_process : PROCESS (clk)
389     BEGIN
390     IF clk'EVENT AND clk = '1' THEN
391     IF reset = '1' THEN
392     Delay6_out1 <= to_unsigned(16#0000#, 15);
393     ELSIF enb_1_16_0 = '1' THEN
394     Delay6_out1 <= PSS_Detection_out4_unsigned;
395     END IF;
396     END IF;

```

```

397     END PROCESS Delay6_process;
398     offset <= std_logic_vector(Delay6_out1);
399     .0PSS_Detection_out5_signed <= signed(PSS_Detection_out5);
400     -- <S2>/Delay7
401     Delay7_process : PROCESS (clk)
402     BEGIN
403     IF clk'EVENT AND clk = '1' THEN
404     IF reset = '1' THEN
405     Delay7_out1 <= to_signed(0, 32);
406     ELSIF enb_1_16_0 = '1' THEN
407     Delay7_out1 <= PSS_Detection_out5_signed;
408     END IF;
409     END IF;
410     END PROCESS Delay7_process;
411     XcorrOut<= std_logic_vector(Delay7_out1);
412     PSS_Detection_out6_signed <= signed(PSS_Detection_out6);
413     -- <S2>/Delay8
414     delayMatch_process : PROCESS (clk)
415     BEGIN
416     IF clk'EVENT AND clk = '1' THEN
417     IF reset = '1' THEN
418     delayMatch_reg<= (OTHERS =>to_signed(0, 32));
419     ELSIF enb_1_16_0 = '1' THEN
420     delayMatch_reg(0) <= PSS_Detection_out6_signed;
421     delayMatch_reg(1 TO 2) <= delayMatch_reg(0 TO 1);
422     END IF;
423     END IF;
424     END PROCESS delayMatch_process;
425     Delay8_out1 <= delayMatch_reg(2);
426     thresholdOut<= std_logic_vector(Delay8_out1);
427     Determine_Cell_ID_out2_unsigned <= nsigned(Determine_Cell_ID_out2);
428     -- <S2>/Delay11
429     Delay11_process : PROCESS (clk)
430     BEGIN
431     IF clk'EVENT AND clk = '1' THEN
432     IF reset = '1' THEN
433     Delay11_out1 <= to_unsigned(16#00#, 8);
434     ELSIF enb_1_16_0 = '1' THEN
435     Delay11_out1 <= Determine_Cell_ID_out2_unsigned;
436     END IF;

```



```

437     END IF;
438     END PROCESS Delay11_process;
439     secCellID<= std_logic_vector(Delay11_out1);
440     Determine_Cell_ID_out3_unsigned <= unsigned(Determine_Cell_ID_out3);
441     -- <S2>/Delay10
442     Delay10_process : PROCESS (clk)
443     BEGIN
444     IF clk'EVENT AND clk = '1' THEN
445     IF reset = '1' THEN
446     Delay10_out1 <= to_unsigned(16#000#, 9);
447     ELSIF enb_1_16_0 = '1' THEN
448     Delay10_out1 <= Determine_Cell_ID_out3_unsigned;
449     END IF;
450     END IF;
451     END PROCESS Delay10_process;
452     cellID<= std_logic_vector(Delay10_out1);
453     Frequency_Estimation_out1_signed <= signed(Frequency_Estimation_out1);
454     delayMatch1_process : PROCESS (clk)
455     BEGIN
456     IF clk'EVENT AND clk = '1' THEN
457     IF reset = '1' THEN
458     delayMatch1_reg <= (OTHERS =>to_signed(16#0000000#, 27));
459     ELSIF enb_1_16_0 = '1' THEN
460     delayMatch1_reg(0) <= Frequency_Estimation_out1_signed;
461     delayMatch1_reg(1 TO 4) <= delayMatch1_reg(0 TO 3);
462     END IF;
463     END IF;
464     END PROCESS delayMatch1_process;
465     Frequency_Estimation_out1_1 <= delayMatch1_reg(4);
466     freqEst<= std_logic_vector(Frequency_Estimation_out1_1);
467     Frequency_Estimation_out2_signed <= signed(Frequency_Estimation_out2);
468     delayMatch2_process : PROCESS (clk)
469     BEGIN
470     IF clk'EVENT AND clk = '1' THEN
471     IF reset = '1' THEN
472     delayMatch2_reg <= (OTHERS =>to_signed(16#0000000#, 25));
473     ELSIF enb_1_16_0 = '1' THEN
474     delayMatch2_reg(0) <= Frequency_Estimation_out2_signed;
475     delayMatch2_reg(1 TO 4) <= delayMatch2_reg(0 TO 3);
476     END IF;

```

```
477     END IF;
478     END PROCESS delayMatch2_process;
479     Frequency_Estimation_out2_1 <= delayMatch2_reg(4);
480     CPCorrMag<= std_logic_vector(Frequency_Estimation_out2_1);
481     ce_out<= enb_1_16_1;
482     validOut<= Delay2_out1;
483     END rtl;
```

ملخص

OFDM (تعدد الإرسال المتعامد بتقسيم التردد) هو معيار الإرسال اللاسلكي الأكثر استخدامًا في العالم في العديد من الأنظمة مثل ADSL و DAB و DVB-T2 و DVB-S2 و Wifi و Wimax و LTE (4G) و NR (5G.) يتمتع OFDM بمزايا الكفاءة الطيفية العالية والمتانة المتأصلة ضد خبو القناة ، لذا فهو مستخدم في نظام LTE يقدم هذا العمل جهاز إرسال واستقبال باستخدام تخطيطي لمغير OFDM وكاشف عبر قناة AWGN. يتم تنفيذ جهاز الإرسال والاستقبال بلغة HDL على Matlab / Simulink. تتم مقارنة نتائج تطبيق HDL مع إشارة مرجعية ثم دراسة التباين في عرض النطاق الترددي و SNR وتأخير القناة بعد إنشاء رمز خط HDL ، يكون التنفيذ جاهزًا للتنفيذ في الأجهزة على FPGA (مصنوفة البوابة القابلة للبرمجة الميدانية) عبر نظام الجيل Xilinx .

RESUME

L'OFDM (Orthogonal Frequency-Division Multiplexing) est la transmission sans fil standard la plus utilisée au monde dans plusieurs systemes tel que L'ADSL, DAB, DVB-T2, DVB-S2, Wifi, Wimax, LTE (4G) et NR (5G.) L'OFDM présente les avantages d'une efficacité spectrale élevée et d'une robustesse inhérente contre les évanouissements de canal, de sorte qu'il est utilisé dans le système LTE. Ce travail présente un émetteur-récepteur en utilisant un schéma d'un modulateur et d'un détecteur OFDM via un canal AWGN. L'émetteur-récepteur est implémenté en langage HDL sur Matlab/Simulink. Les résultats d'implémentation HDL sont comparer avec un signal de référence ensuite l'etude de variation de la bande passante, SNR et Delay canal. Apres la generation de code line HDL, l'implémentation réaliser est prete une implementation en matériels sur FPGA (Field Programmable Gate Array) via le système de génération Xilinx.

ABSTRACT

LTE is the most widely used standard wireless transmission in the world such as ADSL, DAB, DVB-T2, DVB-S2, Wifi, Wimax, LTE 4G and NR 5G. OFDM (Orthogonal Frequency-Division Multiplexing) has the advantages of high spectral efficiency and inherent robustness against channel fading, so it is used in LTE system. This work presents a transceiver using a schematic of a Modulator and OFDM Detector via an AWGN channel. The transceiver is implemented over HDL on matlab / Simulink. The results of HDL implementation are compared with a reference signal then the study of variation in bandwidth, SNR and channel delay. After the generation of HDL line code, the implementation is ready for implementation in hardware on FPGA (Field Programmable Gate Array) via the Xilinx generation system.