REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département Electronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : Electronique

Spécialité : Industries Electroniques

Par

- ***Rania SAOUDI***
- ***Moussa Abderrahim MEHIRIS***

Intitulé

Implémentation d'un système de communication à base de la technique OFDM sur une carte FPGA

Soutenu le : 16/06/2022

Devant le Jury composé de :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M. Khaled ROUABAH	*Pr*	*Président*	*Univ-BBA*
M. Idris MESSAOUDENE	*MCA*	*Encadreur*	*Univ-BBA*
M. Seif Eddine AZOUG	*MCB*	*Examinateur*	*Univ-BBA*

Année Universitaire 2021/2022


بسم الله الرحمن الرحيم

Remerciements

Nous rendons nos profondes gratitudes tout d'abord à Allah Tout-Puissant, l'Omnipotent et le Bienveillant, qui nous aide et nous a donné la santé, la force et la patience d'achever ce modeste travail.

En préambule à ce mémoire, nous tenons à remercier vivement notre encadrant Dr. Idris MESSAOUDENE, de nous avoir pris en charge et bien encadré, fait profiter de son expérience scientifique durant ce travail. On le remercie sincèrement pour son aide précieuse, son encouragement, ses judicieux conseils, sa patience et sa disponibilité tout au Long de notre réalisation de ce mémoire.

Nous adressons tous nos remerciements les plus chères au président du jury et l'examinateur pour l'honneur qu'ils nous font en jugeant ce travail.

Nous désirons aussi remercier toutes les membres du département d'Electronique de l'université de Bordj-Bou-Arreridj, que ce soit des enseignants ou cadres administratifs qui ont contribué de près ou loin à l'élaboration et l'avancement du notre projet fin d'étude.

Dédicace

*D'abord je tiens à remercier « **ALLAH** » tout puissant qui m'a aidé, qui m'a donné la force et le courage pour réaliser ce mémoire. Et puis je dédie ce modeste travail à :*

Mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

Mes chers frères et sœurs pour leur appui, leur encouragement permanents et leur soutien moral.

*Tous les membres de la famille **Saoudi** et la famille **Mohamadi** pour leurs conseils ainsi que leur soutien inconditionnel tout au long de mon parcours universitaire.*

*Mon binôme **Moussa Abderrahim**, pour son soutien moral, sa patience et sa compréhension tout au long de ce projet.*

Sans oublier mes enseignants du département d'électronique ainsi que mes collègues de la promotion MCIL d'université BBA.

Saoudi Rania

Dédicaces

Je dédie ce travail à :

Mes chers parents, surtout ma mère, pour tous ses sacrifices, son amour, sa tendresse, son grand soutien et ses prières tout au long de mes études ;

Mes chères sœurs pour leurs encouragements permanents, et leur soutien moral.

Mes chers frères ***Abderrahmane***, ***Hichem*** *et* ***Oussama*** *pour leurs appuis et leurs encouragements.*

Toute ma famille ***Mehiris*** *et* ***Kacimi*** *pour leurs soutiennes même de loin tout au long de ma carrière universitaire.*

Mon binom ***S.Rania***.

Mes chers amies ***B.Boubaker***, ***R.Mohammed El Amin***, ***H.Mohammed El Amin***, ***B.Tadj Eddin*** *et tous mes collèges de l'université.*

Mehiris Moussa Abderrahi

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

3G	3rd Generation
4G	4th Generation
AMBA	Advanced Microcontroller Bus Architecture
APU	Application Processing Unit
ARM	Advanced RISC Machine
AXI	Advanced extensible Interface
AWGN	Additive White Gaussian Noise
CAN	Controller Area Network
CP	Cyclic Prefix
CLB	Configurable Logic Blocks
DAB	Digital Audio Broadcasting
DDR	Double Data Rate
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DSP	Digital Signal Processor
DVB	Digital Video Broadcasting
FDM	Frequency Division Multiplexing
FF	Flip Flop
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GigeE	Ethernet (RGMII)
HDMI	High Definition Multimedia Interface
I2C	Inter-Integreted Circuit bus
ICI	Inter Carrier Interference
IEEE	Institute of Electrical and Electronics Engineers
IDFT	Inverse Discreet Fourier Transform
IFFT	Inverse Fast Fourier Transform
IOB	Input/Output Blocks
ISI	Inter-Symbol Interference
JTAG	Joint Test Action Group
LAN	Local Area Network
LED	Ligth-Emitting Diode
LSB	Least Significant Bit
LTE	Long-Term Evolution
LUT	Look Up Table
MMC	Multi-Carrier Modulation
MMU	Memory Management Unit
MOI	Mutiplexed Input /Output

MSB	Most Significant Bit
OCM	On Chip Memory
OFDM	Orthogonal Frequency Division Multiplexing
RAM	Random Access Memory
SD	Secure Digital
SMC	Static Memory Controller
SPI	Serial Peripheries Interface
SCU	Snoop Control Unit
SoC	System On Chip
SNR	Signal to Noise Ratio
SRAM	Static Random Access Memory
TEB	Taux Erreur Binaire
PAPR	Peak-to-Average Power Ratio
PL	Programmable Logic
Pmod	Peripheral Module interface
PS	Processing System.
QAM	Quadrature Amplitude Modulation
Quad-SPI	Quad Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UWB	Ultra Wide Band
WLAN	Wireless Local Area Network
WiMAX	Worldwide Interoperability for Microwave Access
XADC	Xilinx Analog to Digital Converter

Introduction générale

Introduction générale

Les communications numériques envahissent la quasi-totalité des domaines d'activité, la demande de systèmes de transmission assurant des très hauts débits avec une haute qualité de service ne cesse de croître. Ceci a motivé la recherche de nouveaux modes de transmissions capables de supporter des transmissions à large bande. Un prometteur candidat pour le développement de ces systèmes est le multiplexage par répartition orthogonale de la fréquence OFDM (Orthogonal Frequency Division Multiplexing) en raison de son efficacité spectrale élevée et de sa meilleure robustesse contre les canaux d'évanouissement à trajets multiples de Rayleigh sans fil.

En télécommunications, l'OFDM est une technologie de transmission consiste en un type de modulation à porteuses multiples basées sur un procédé de codage de signaux numériques par répartition en fréquences orthogonales. Son principe est de répartir un flux de données à transmettre sur un canal de transmission à large bande sur plusieurs sous-canaux à bande étroites, chacun étant modulé à un faible débit de données. Ceci explique la nomenclature de transmission qui est utilisée lorsqu'on parle des systèmes OFDM.

Notre projet de fin d'étude est axé sur l'implémentation d'un système de communication sans fil OFDM sur une carte FPGA du fabricant Xilinx Pynq-z2 en raison de leur flexibilité et de leur évolutivité. Donc, deux objectifs sont identifiés pour ce travail. Le premier objectif est de se familiariser avec les systèmes de communication sans fil, et plus particulièrement OFDM, et leur principe de fonctionnement. Le second objectif est d'approfondir nos connaissances théoriques et pratiques sur les cartes FPGA qui vont être manipulées dans ce travail.

Le présent manuscrit est structuré en trois chapitres :

- Le premier chapitre sera consacré à fournir des informations générales sur les systèmes de communication sans fil OFDM, à discuter ces caractéristiques et leur principe de fonctionnement et examiner les structures de base de la chaîne de transmission des technologies OFDM et OFDM avec CP et de mentionner certains de ses avantages et inconvénients par rapport aux autres techniques.
- Le deuxième chapitre sera dédié à la présentation générale de la famille Zynq AP SOC puis expliquera les principales composantes internes constituant la carte FPGA Xilinx PYNQ- Z2 et l'environnement de synthèse adopté dans son conception matérielle.

- Le troisième chapitre portera sur l'étude pratique de notre projet. En commençant par la partie simulation software afin de suivre et de vérifier les différentes réponses du système conçu avant de passer à la partie pratique hardware en vérifiant les problèmes de chaque constituant de la chaine de transmission OFDM. Enfin, on va entamer la mise en œuvre finale où on présentera les résultats des différents tests effectués à travers les trois méthodes différentes d'implémentation.

Finalement, les conclusions des différents travaux réalisés dans ce projet ainsi que quelques perspectives seront présentées dans la conclusion générale.

Chapitre Ⅰ :

Généralités sur la modulation OFDM

1.1. Introduction

Récemment, la demande croissante des systèmes de transmission garantissant des débits très élevés dans les communications numériques a motivé la recherche des nouveaux modes de transmissions capables de supporter des transmissions à large bande. Le premier candidat pour le développement de ces systèmes est la modulation par répartition orthogonale en fréquence OFDM .Cette technique de multiplexage en fréquence existe depuis longtemps mais l'intérêt actuel est axé sur l'amélioration apportée pour augmenter l'efficacité spectrale et la robustesse optimale vis-à-vis des multi trajets.

C'est pourquoi dans ce chapitre, on définit les systèmes OFDM et on présente leur principe de base, puis on reviendra en détail sur leur chaîne de transmission : l'émission, canal de transmission et la réception. Enfin, on citera quelques avantages et inconvénients de ce type de modulation.

1.2. Historique

La modulation multi-porteuse a été introduite à la fin des années 1950. Sa première utilisation était dans des systèmes de communications hautes fréquences militaires pour des applications radio HF. Parallèlement, le concept d'utilisation de la transmission des données parallèles et du FDM (Frequency Division Multiplexing) a été décrit au début des années 1960.

Dix ans plus tard, le schéma d'un modem de la modulation multi-porteuse a été simplifié avec l'utilisation de la Transformée de Fourier discrète ; TFDI (Transformée de Fourier discrète Inverse) à l'émission et la TFD (Transformée de Fourier discrète) au niveau du récepteur, ce qui rend facile son implémentation numérique [1].

En 1995, la technique OFDM a été développée dans les domaines industriels civils tels que le projet de radiodiffusion numérique DAB (Digital Audio Broadcasting) et le DVB (Digital Video Broadcasting).

Entre 1999 et 2001, les normes WLAN's (Wireless Local Area Network), telle que IEEE 802.11a/g nommé Wi-Fi (Wireless Fidelity) et HiperLAN (High Performance radio LAN), ont adoptés la modulation multiporteuse OFDM comme une spécification principale de leur couche physique.

En 2005, une amélioration de la technologie Wi-Fi par le procédé de modulation OFDM a été adoptée par l'alliance WiMedia pour les communications à très haut débit et à courte portée basées sur la technologie UWB (Ultra Wide Band).

Enfin, plus récemment l'OFDM se trouve dans plusieurs standards comme ceux des communications filaires ou sans fil mobiles et même les communications optiques. On notera les réseaux mobiles, WiMAX (Worldwide Interoperability for Microwave Access), les standards LTE (Long-Term Evolution) et la 4G (4th Generation).

1.3. Modulation Multiporteuse

Les technologies de modulation multi-porteuse ou MCM (Multi-Carrier Modulation) font actuellement partie des technologies qui contribuent à améliorer les performances des systèmes de transmission.

Le principe d'un système de modulation multi-porteuse est de transmettre des données sur plusieurs fréquences porteuses. Les fréquences sont choisies de manière à ce que l'espacement fréquentiel entre deux porteuses successives soit identique pour toutes les porteuses [2]. Lorsque le multiplexage fréquentiel obtenu est orthogonal en temps et en fréquence on parle de modulation OFDM

1.4. Modulation OFDM

La technique multi-porteuse OFDM est une technique de modulation basée sur un processus de codage des signaux numériques par répartition en fréquences orthogonales. L'idée de base de l'OFDM est de partager un signal à large bande en multiples signaux à bandes étroites. Ces signaux à bandes étroites sont mathématiquement orthogonaux, c'est-à-dire que l'information est transmise parallèlement en utilisant N recouvrements fréquentiels orthogonaux des bandes étroites.

Ainsi, les N données qui étaient auparavant transmises consécutivement à un débit élevé de $1/Td$, vont être, émises simultanément sur N sous-canaux fréquentiels élémentaires à bas débit $1/Ts$ [2] comme le montre la figure 1.1[3] :

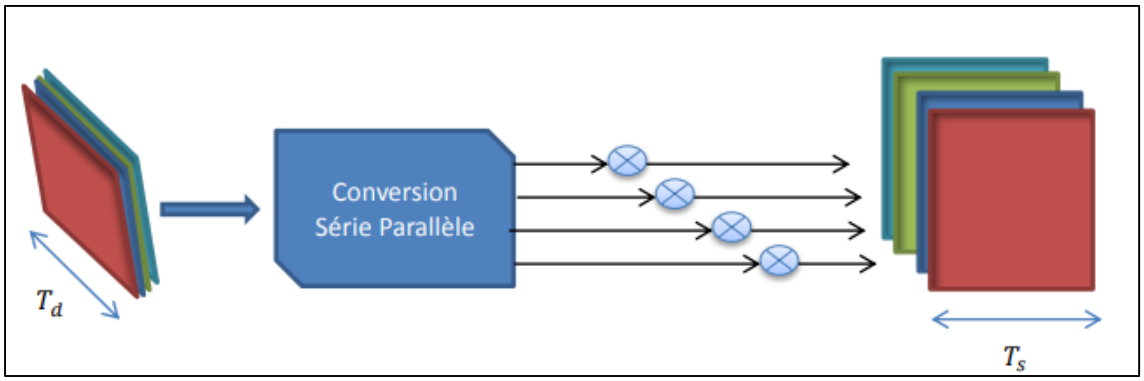


Figure1. 1. Principe de la modulation OFDM.

1.5. Notions d'orthogonalité

Pour assurer une efficacité spectrale optimale, les porteuses doivent respecter la contrainte d'orthogonalité qui permet de transmettre parfaitement plusieurs signaux sur un canal commun et les détecter, sans interférences [4] à la fois dans les deux domaines temporel et fréquentiel comme l'illustre la figure 1.2 [2]

1.5.1. Orthogonalité temporelle

Dans le domaine temporel, les signaux OFDM sont constitués d'une somme de N sinusoïdes de fréquences respectives f_k (0, 1,2,…N-1) transmises pendant une durée T_s (Voir l'exemple d'un signal OFDM avec quatre sinusoïdes illustré dans la figure 1.2.a), la fréquence f_k est donnée par :

$$f_k = f_0 + {}^{k}/_{T_s} \qquad \textbf{(1.1)}$$

L'équation (1.2) montre un ensemble de sinusoïdes représentant les sous porteuses pour un signal OFDM réel non modulé [2].

$$S_k(t) = \begin{cases} \sin(2\pi f_k t), & si\ 0 < t < T_s \\ 0 \quad , & Ailleurs \end{cases} \qquad \textbf{(1.2)}$$

On dit que les sous-porteuses, définies par l'expression (1.2), sont orthogonaux sur l'intervalle [0, T_s] si elles répondent aux conditions de l'équation (1.3) [4].

$$\int_0^{T_s} \psi_p(t)\psi_q^*(t)dt = \int_0^{T_s} e^{j2\pi f_p t} e^{-j2\pi f_q t} dt = \begin{cases} 0, & si\ p \neq q \\ 1, & si\ p = q \end{cases} \qquad \textbf{(1.3)}$$

1.5.2. Orthogonalité fréquentielle

La propriété de l'orthogonalité du signal OFDM peut être vérifiée en visualisant son spectre. Dans le domaine fréquentiel, chaque sous porteuse a une réponse fréquentielle dont l'enveloppe spectrale est un sinus cardinal qui s'annule aux fréquences $f_k - {}^{1}/_{T_s}$ et $f_k + {}^{1}/_{T_s}$ (Voir la figure 1.2.b).

L'orthogonalité dans le domaine fréquentiel est réalisée puisque le maximum de chaque sous-porteuse correspond à un "zéro" des autres. Cette condition permet ainsi d'avoir une occupation spectrale idéale et d'éviter les interférences entre sous-porteuses [2].

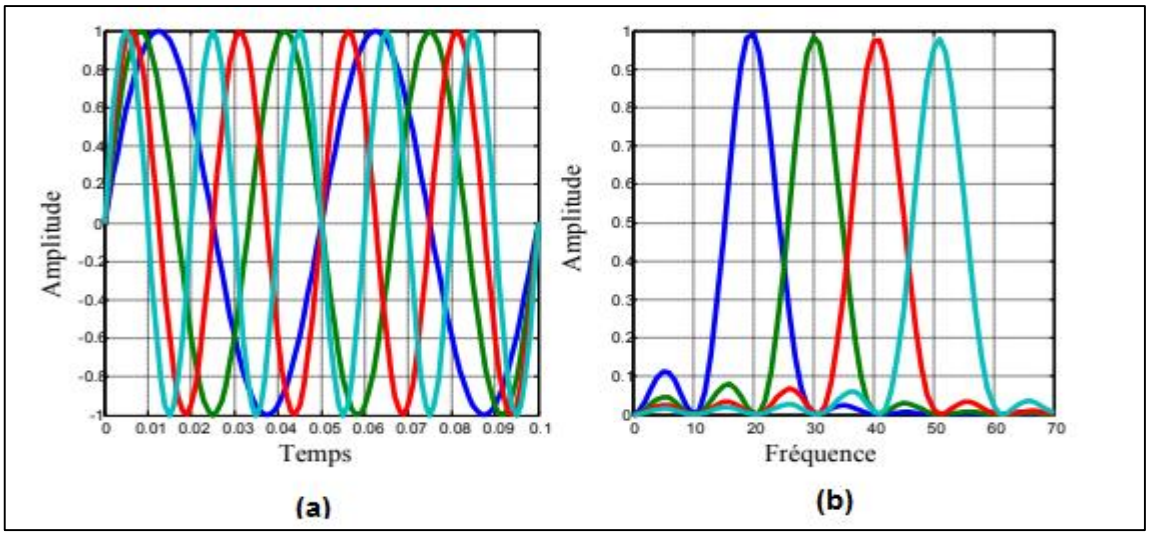


Figure1. 2. L'orthogonalité dans le domaine; (a) temporel, (b) fréquentiel.

1.6. Modèle analogique d'un système OFDM

L'émetteur OFDM module l'information binaire à transmettre en des symboles QAM. Ces symboles sont ensuite divisés en N flux parallèles. Chaque flux de symbole va moduler une sous porteuse différente. Les N symboles sont regroupés de sorte que la séquence de N symboles C_0, C_1,…, C_{N-1} constitue un symbole OFDM. La description mathématique du modulateur OFDM peut s'exprimer comme suit [5] :

$$S_j(t) = \sum_{k=0}^{N-1} C_k e^{2i\pi\left(f_0+\frac{k}{T_s}\right)t} \qquad \textbf{(1.4)}$$

Où :

C_k : sont les éléments binaires définis par une constellation souvent de modulation QAM à 4, 16, 64, 2^q états avec q : le nombre de bit occupé par la modulation QAM.

N : nombre de porteuses dans un symbole OFDM.

Ts : la durée de transmission d'un symbole binaire *C*.

On peut représenter un modulateur OFDM par le schéma bloc illustré dans la figure 1.3 [5] :

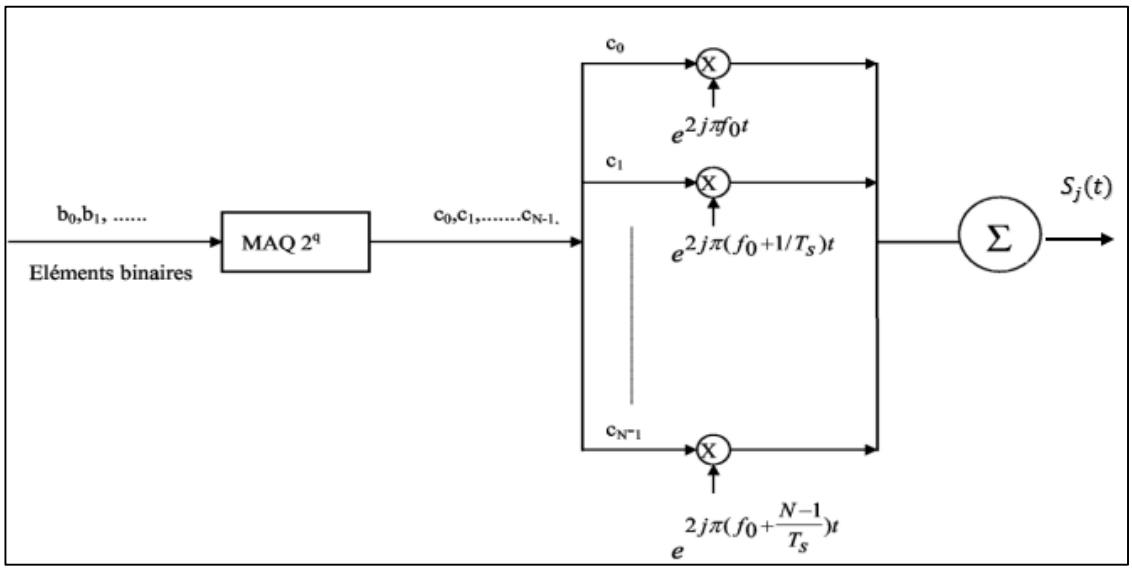


Figure1. 3. Schéma de principe de la modulation OFDM.

Au niveau du récepteur, on élimine d'abord l'effet du canal et du bruit sur les informations transmises, puis on reconstruit les symboles QAM émis depuis la bande de base, où le symbole OFDM a parvenu au récepteur. La description mathématique du démodulateur OFDM peut s'exprimer comme suit [5] :

$$y(t) = \sum_{k=0}^{N-1} C_k H_k(t) \mathrm{e}^{2\mathrm{i}\pi\left(\mathrm{f}_0+\frac{\mathrm{k}}{\mathrm{T_s}}\right)\mathrm{t}} \qquad \textbf{(1.5)}$$

Où :

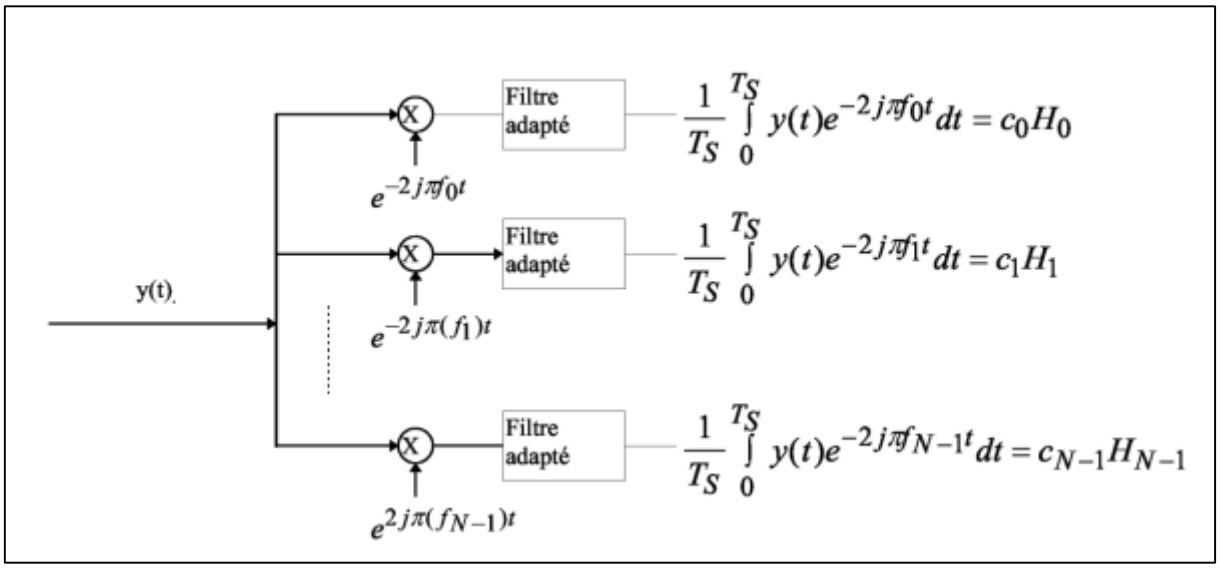
C_k : sont les éléments binaires définis par une constellation souvent de modulation QAM à 4, 16, 64, 2^q états.

N : nombre de porteuses dans un symbole OFDM.

$H_k(t)$ est la fonction de transfert du canal autour de la fréquence f_k et à l'instant *t*.

Ts : la durée de transmission d'un symbole binaire *C*.

On peut représenter un démodulateur OFDM par le schéma bloc de la figure 1.4 [5] :



- **Figure1. 4.** Schéma de principe de la démodulation OFDM.

1.7. Modèle Numérique d'un système OFDM

La mise en œuvre d'un système OFDM analogique, tel que décrit dans la section précédente, est très complexe car elle nécessite d'utiliser N filtres analogiques parfaitement orthogonaux en parallèle. Ces filtres sont pratiquement très difficiles à réaliser et leur installation est très coûteuse. C'est la raison pour laquelle l'OFDM a été proposée dans les domaines du traitement numérique du signal. Ce dernier a simplifié le problème à l'aide d'une implémentation numérique en utilisant les algorithmes FFT (Fast Fourier Transform) et IFFT (Inverse Fast Fourier Transform) [6].

1.7.1. Description mathématique d'un modulateur OFDM numérique

L'expression numerique d' un symbole OFDM composé de N sous porteuses sur une période T_s est de la forme :

$$S(t) = e^{2i\pi f_0 t} \sum_{k=0}^{N-1} C_k e^{2i\pi \frac{k}{T_S} t} \quad \textbf{(1.6)}$$

En discrétisant ce signal et en le ramenant en bande de base pour l'étude numérique, on obtient une sortie S_n sous la forme :

$$S_n = \sum_{k=0}^{N-1} C_k e^{2i\pi \frac{k}{N} n} \quad \textbf{(1.7)}$$

Notons que les S_n sont donc obtenus par une transformée de Fourier discrète inverse des C_k. En choisissant le nombre de porteuses N tel que $N = 2^n$, le calcul de la transformée de Fourier inverse discrète se simplifie par l'utilisation d'une simple IFFT.

1.7.2. Description mathématique d'un démodulateur OFDM numérique

Le signal en bande de base parvenant au récepteur durant la période T_s s'écrit sous la forme [5] :

$$Z_n = Z(\frac{nT_s}{N}) = \sum_{k=0}^{N-1} C_k H_k e^{2i\pi\frac{k}{N}n} \quad \textbf{(1.8)}$$

On voit que Zn est la Transformée de Fourier discrète inverse de$C_k H_k$, la démodulation consiste donc à effectuer une Transformée de Fourier directe discrète. Le nombre de porteuses est choisi pour $N = 2^n$. L'intérêt de cette discrétisation est qu'on peut réaliser ces transformées de Fourier à l'aide d'algorithmes de FFT.

1.7.3. Chaine de transmission d'un système OFDM classique

Le synoptique du système OFDM classique à temps discret est illustré dans la figure 1.5 :

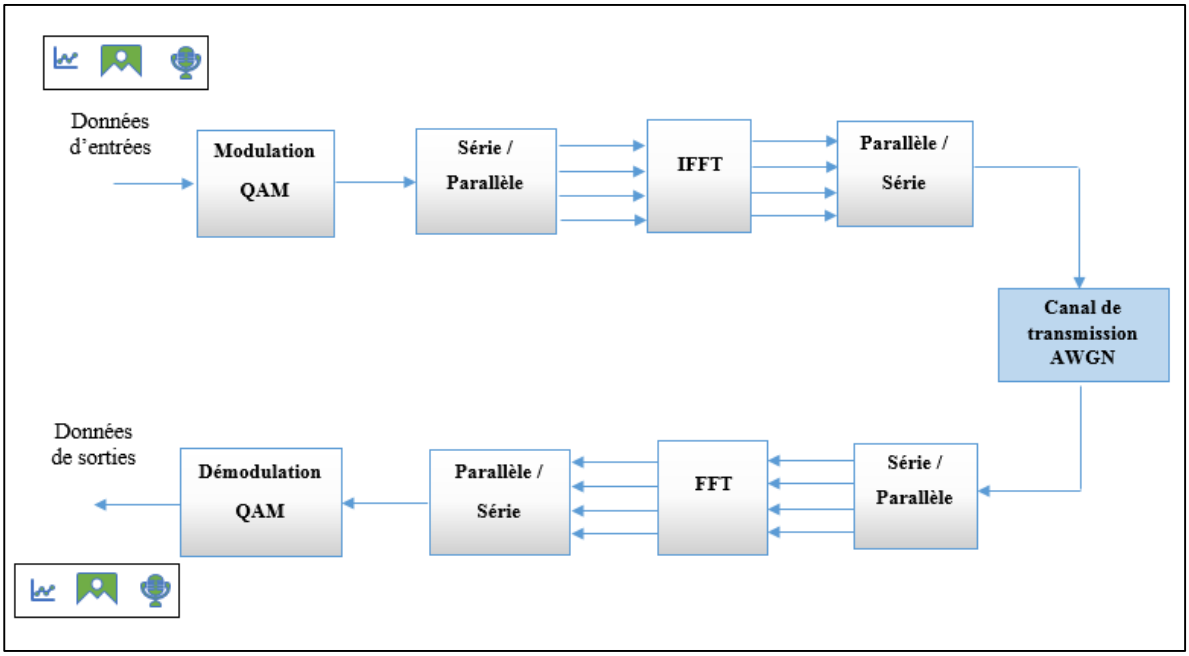


Figure1. 5. Schéma bloc d'un système OFDM classique.

Dans l'émetteur, les données binaires série provenant de la source de données (une image ou un audio ou bien une vidéo) sont mappées par une modulation QAM pour générer des nombres complexe. Ensuite, ces symboles séries seront convertis en données parallèles à l'aide d'un convertisseur série-parallèle (S/P). Plus tard, les symboles parallèles modulés sont convertis en des signaux temporels via le bloc IFFT. Ces données parallèles sont ensuite converties en données série à l'aide d'un convertisseur parallèle-série (P/S) afin d'obtenir enfin le symbole OFDM, que l'on transmet à travers un canal.

Au niveau du récepteur, les processus inverses seront effectués de sorte que les signaux séries reçus sont convertis en parallèles via le convertisseur série/ parallèle Le signal est transformé ensuite, du domaine temporel vers le domaine fréquentiel par le bloc FFT. Les données parallèles sortant du bloc FFT sont converties en des données série via un bloc parallèle/série. Enfin, le décodage des bits est effectué par une démodulation QAM et les données sont récupérées.

1.7.4. Chaine de transmission d'un système OFDM avec CP

Jusqu'à présent, nous avons toujours supposé une synchronisation parfaite entre l'émetteur et le récepteur. Dans un canal d'évanouissement par trajets multiples sélectif en fréquence, les désadaptations de synchronisation sont généralement d'ordre significatif que les impulsions de base du signal OFDM d'origine et les versions retardées du signal ne sont plus orthogonales. Cette propriété due au chevauchement des composantes spectrales provoquant des interférences entre les symboles. Il existe cependant une technique simple qui modifie le signal d'émission afin que l'orthogonalité soit en quelque sorte préservée en présence de composantes de signal à trajets multiples, comme illustré dans la figure 1.6.

D'après le schéma bloc du modèle amélioré de l'OFDM-CP ci-dessous, on constate que l'idée est très similaire au modèle OFDM classique que nous avons abordé dans la section précédente, dans lequel il n'y a que deux modules supplémentaires. Du côté de l'émetteur, une quantité de données appelée cyclique préfixe est ajoutée aux symboles parallèles qui sont traités par le bloc IFFT via le module d'insertion du CP. Ce dernier sera supprimé avant le bloc FFT de l'autre côté du récepteur à travers le module d'élimination du CP.

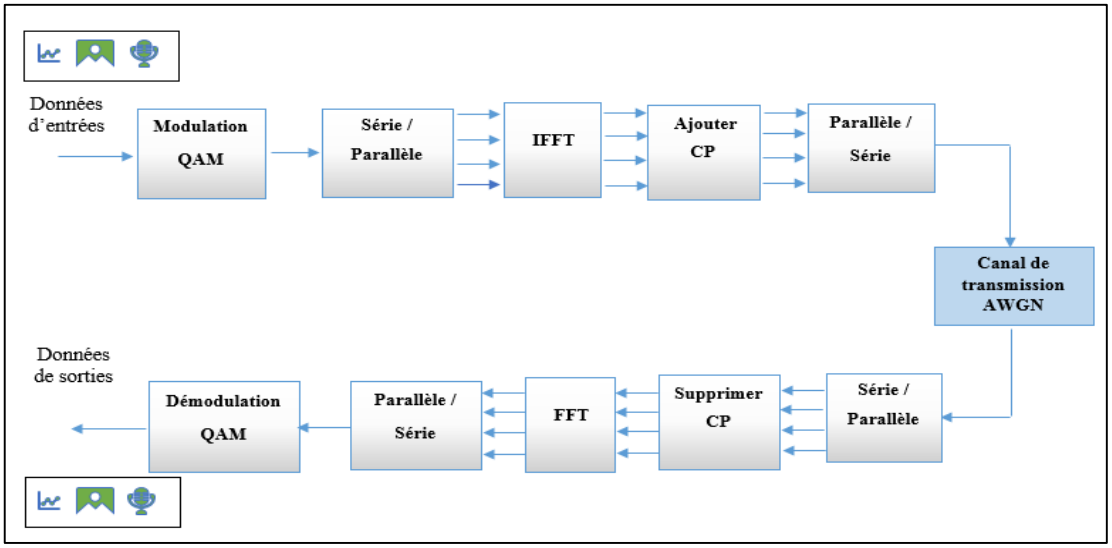


Figure1. 6. Schéma bloc d'un système OFDM avec CP.

1.8. Description des différents blocs d'une chaine de transmission OFDM-CP

La chaine de transmission OFDM-CP englobe un ensemble des blocs fonctionnels, on se limite seulement à présenter les principaux blocs qui entrent dans la composition de la chaine et qui sont :

1.8.1. Modulation d'amplitude en quadrature

La modulation d'amplitude en quadrature est aussi connue par son abréviation anglaise QAM (Quadrature Amplitude Modulation**)**. Cette modulation résulte de la combinaison de deux fréquence porteuse sinusoïdales déphasées l'une de l'autre de 90° (d'où le nom de quadrature) et la sortie résultante consiste à la fois en variations d'amplitude et de phase [6]. La modulation QAM est capable de supporter des débits de transmission de données numériques plus élevés que les schémas modulés en amplitude et les schémas modulés en phase [7].

Lors de l'utilisation de QAM, les points de la constellation sont disposés dans une grille carrée avec des espacements verticaux et horizontaux égaux. Les formes les plus courantes de QAM utilisent donc une constellation avec un nombre de points égal à 4, 16, 64. . . . etc, comme l'illustre la figure 1.7.

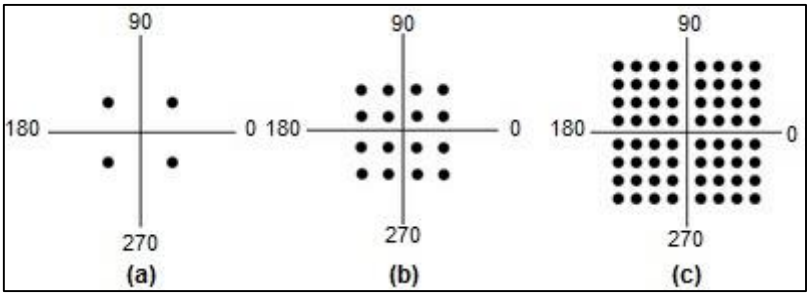


Figure1. 7. Constellation des modulations ; MAQ 4 (a),16 (b) ,64(c).

1.8.2. Conversion série - parallèle / parallèle - série

Les données à transmettre prennent généralement la forme d'un flux de données série, il est donc nécessaire d'intégrer un convertisseur série- parallèle et parallèle-série, dont la conversion série parallèle illustrée dans de la figure 1.8.a, permet de convertir l'information numérique de plusieurs bits existants pendant des intervalles de temps très courts, en une information numérique où tous les bits sont disponibles en même temps. Tandis que la conversion parallèle - série est le processus inverse qui permet de transformer une information numérique de plusieurs bits disponibles simultanément, en une information numérique, transmise sur un seul fil, et pour laquelle chaque bit est présent pendant un temps défini (voir la figure 1.8.b).

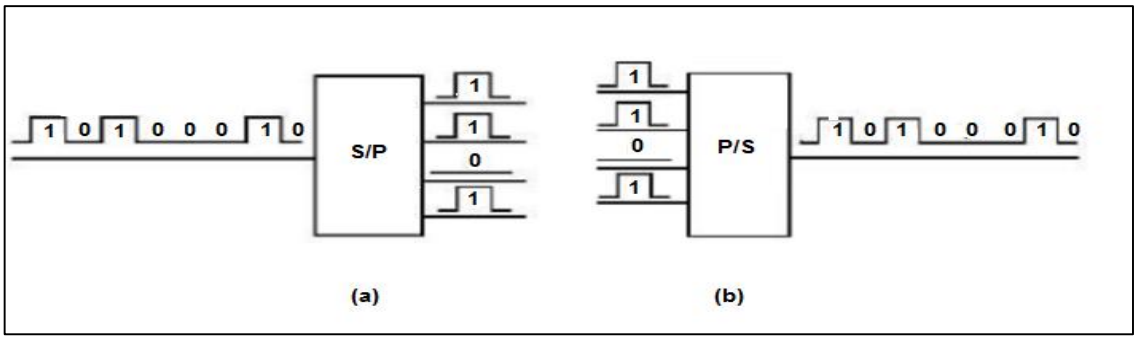


Figure1. 8. Convertisseur série- parallèle (a) / parallèle- série (b).

1.8.3. Transformé de fourrier rapide FFT et Transformé de fourrier rapide inverse IFFT

La Transformée de Fourier Discrète (DFT) et la Transformée de Fourier Discrète Inverse (IDFT) sont couramment utilisées en particulier dans les systèmes de communication numérique utilisant le principe de modulation multiporteuse. Dans de tels systèmes, un IDFT est calculé au côté de l'émetteur et un DFT au côté du récepteur. Pour réduire les opérations mathématiques utilisées dans le calcul de DFT et IDFT, on utilise l'algorithme de transformée

de Fourier rapide FFT et IFFT qui correspond respectivement à DFT et IDFT. Ces algorithmes sont couramment utilisés en traitement numérique du signal pour transformer des données discrètes du domaine temporel vers le domaine fréquentiel ou vice versa comme le montre la figure 1.9.

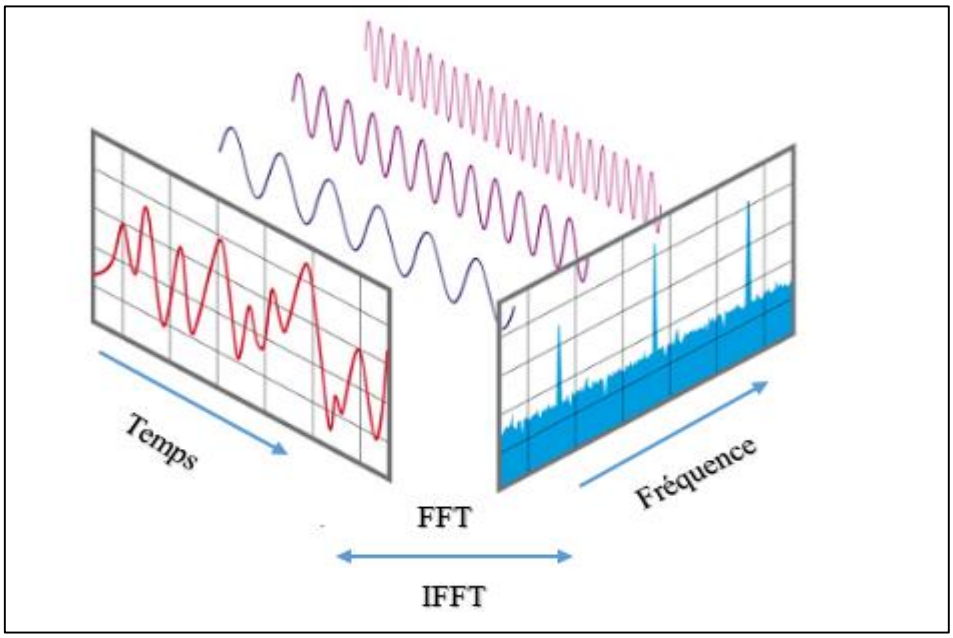


Figure1. 9. Transformée de fourrier FFT et Transformée de fourrier inverse IFFT.

1.8.4. Préfixe cyclique

Le préfixe cyclique est un bon moyen de lutter contre les interférences entre symbole (ISI) consiste à étendre la durée du symbole OFDM en copiant les derniers échantillons du symbole OFDM et en les plaçant à son début. La figure 1.10 donne une vue sur le principe d'insertion du préfixe cyclique.

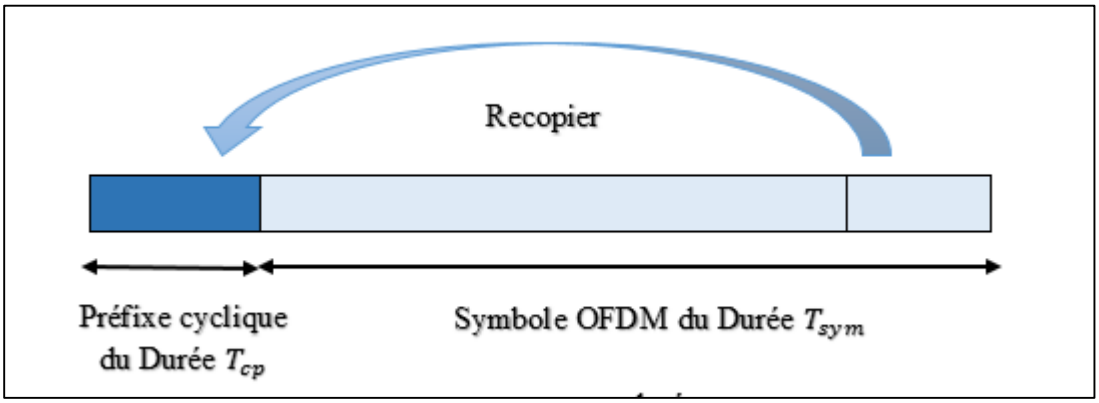


Figure1. 10. Insertion d'intervalle de garde.

Comme le montre la figure1.10 , Le T_{cp} est la longueur du préfixe que nous ajoutons au symbole initial T_{sym}, nous obtenons le symbole OFDM totale pour la longueur de :

$$T_{Totale} = T_{cp} + T_{sym} \quad \textbf{(1.9)}$$

Grâce à cette extension, la période du symbole est plus longue. A noter que bien que la période soit plus longue, cela n'a pas une incidence sur le spectre fréquentiel du signal dont l'orthogonalité est maintenue et les interférences éliminées [6].

1.9. Avantages et inconvénients de l'OFDM

La modulation multiporteuse, en particulier l'OFDM, a été appliquée avec succès à une variété d'applications de communication numérique, ces dernières années. Cette technique présente des avantages et des inconvénients.

1.9.1. Les avantages de l'OFDM

OFDM a été utilisée dans plusieurs systèmes sans fils hauts débits en raison de nombreux avantages qu'elle offre, notamment [1]:

- Efficacité spectrale élevée par son orthogonalité en permettant le chevauchement entre des N sous porteuses, ce qui garantit une utilisation optimale de la bande de fréquence allouée.
- Mise en œuvre efficace à l'aide de la transformation de Fourier rapide (FFT) et la transformation de Fourier rapide inverse (IFFT).
- L'utilisation des systèmes OFDM permet de minimiser l'effet des ISI (Inter-Symbol Interference) sur les canaux à trajets multiples, en ajoutant un intervalle de garde, ce qui garantit l'augmentation de la robustesse du signal.

1.9.2. Les inconvénients de L'OFDM

Malgré ces caractéristiques attrayantes, le système OFDM n'est pas parfait, il présente également certains des inconvénients qu'il faut prendre en considération en faisant la conception. Ces inconvénients sont brièvement énumérés ci-dessous [1] :

- Sensible au l'effet Doppler (Interférences entre les sous porteuses).
- Facteur de crête (PAPR ; 'Peak-to-Average Power Ratio') élevé, nécessitant un circuit d'émetteur linéaire, qui souffre d'une faible efficacité de puissance.

1.10. Conclusion

Ce chapitre présente un aperçu sur le concept de la modulation multi-porteuse. Au cours de ce dernier, on a introduit les différents mécanismes des modulations à porteuses multiples, en particulier les concepts de la modulation OFDM, où on a constaté que bien qu'elle soit résistante aux canaux sélectifs en fréquence, elle présente certaines limites d'utilisation. On décrit dans le prochain chapitre la plate-forme de développement FPGA Xilinx sur laquelle on va implémenter le système OFDM.

Chapitre Ⅱ :

Etude des circuits logiques programmables FPGA

2.1. Introduction

Suite à la révolution technologique du XXe siècle et l'implantation de l'électronique dans tous les domaines de la vie, l'évolution des circuits électroniques était très perceptible et le besoin des circuits puissants et spécialisés pour des applications bien précises a été apparu où ces applications nécessitent que ces circuits soient programmables. Cela a conduit à l'émergence d'une famille des circuits programmable comme les FPGA (Field Programmable Gate Array).

Ainsi, au cours de ce chapitre on va présenter un bref rappel sur les circuits logiques programmables FPGA notamment du constructeur Xilinx. Puis, on va donner un aperçu général sur la famille Zynq où on abordera leur infrastructure et leurs composants de base. Enfin, on examinera de plus près le cadre open source Pynq où on discutera de son concept général et de ses éléments de bases.

2.2. Circuits logiques programmables FPGA Xilinx

Les FPGAs font partie de la famille des composants électriquement programmables. Ils sont des composants entièrement reconfigurables constitués de plusieurs PALs (Programmable Array Logic) élémentaires reliés entre eux par une zone d'interconnexion et regroupés pour former des blocs logiques.

Le fabricant Xilinx est l'un des pionniers dans le domaine des FPGAs où les performances et les capacités de ses dispositifs programmables couvrent des niveaux modestes à extrêmement élevés. Les circuits FPGA du Xilinx, structurés sous forme de matrices, utilisent deux types de cellules de base, les cellules logiques configurables appelées CLB (Configurable Logic Blocks), et les cellules d'entrées/sorties appelées IOB (Input/Output Blocks). Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions programmable [8] (voir la figure 2.1).

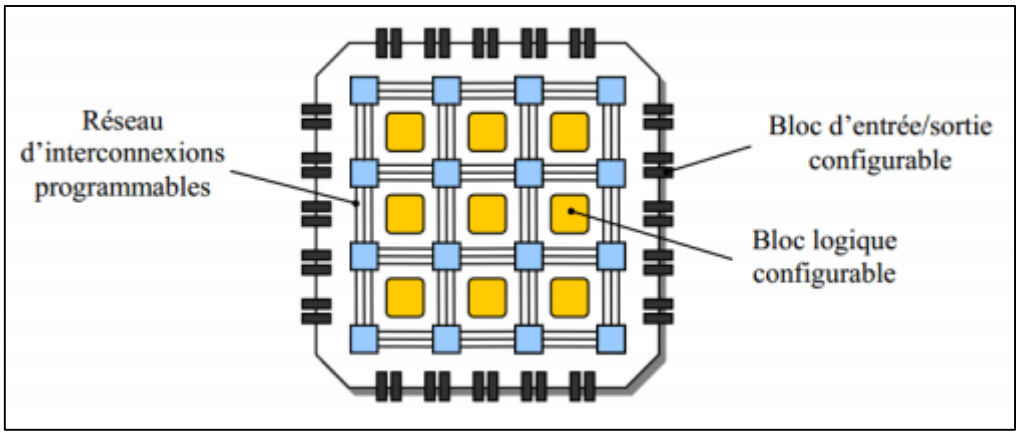


Figure2. 1. Structure globale d'une FPGA du fabricant Xilinx.

2.3. Description de la famille Zynq-7000

Zynq-7000 est une famille de FPGA de la série XILINX-7 basée sur l'architecture du Xilinx (All Programmable System-on-Chip (AP SoC)).Elle offre la flexibilité et l'évolutivité d'un FPGA, tout en offrant les performances, la puissance et la facilité d'utilisation associées aux ASIC (Application Specific Integrated Circuit) et ASSP (Application Specific Standard Parts).

La gamme d'appareils de la famille Zynq-7000 AP SoC permet aux concepteurs de cibler des applications sensibles aux coûts ainsi que des applications hautes performances à partir d'une plate-forme unique à l'aide d'outils standards de l'industrie. Chaque appareil de la famille Zynq-7000 contient le même PS (Processing System) mais les ressources PL (logique programmable) et les entrées sorties se varient d'un appareil à un autre en fonction du périphérique [9].

En conséquence, les dispositifs Zynq-7000 AP SoC sont capables de servir un large éventail d'applications, notamment : caméra de diffusion, commande de moteurs industriels, de réseaux industriels et de vision industrielle, caméra intelligente, Base radio et LTE, diagnostic et imagerie médicale, imprimantes multifonctions, équipement vidéo et vision nocturne, voiture intelligente, etc....[9].

2.4. Architecture de la famille Zynq-7000 AP SoC

Dans cette section, on abordera brièvement l'architecture et les fonctionnalités de la famille Zynq-7000 AP Soc. L'architecture de base d'un dispositif Zynq-7000 AP SoC, illustrée schématiquement dans la figure 2.2, combine un système de traitement basé sur un processeur double cœur ARM Cortex-A9 avec une logique programmable basée sur une FPGA. Les deux parties sont connectées via un certain nombre d'interfaces AXI (Advanced eXtensible Interface).

D'après cette architecture, on constate que le Zynq-7000 AP SoC se compose des principaux blocs fonctionnels suivants :

- Système de traitement (PS).
  - Unité de processeur d'application (APU).
  - Interfaces mémoires.
  - Périphériques d'E/S.
- Logique programmable (PL).
- Interconnexion entre PS et PL.

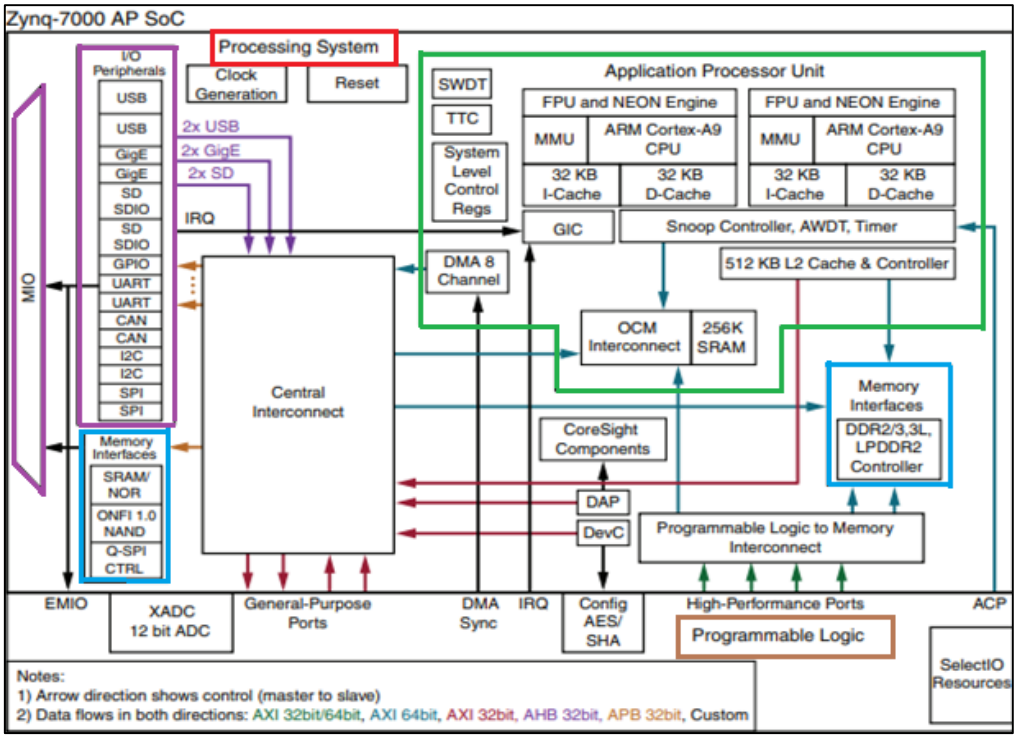


Figure2. 2. Architecture de la famille Zynq-7000 AP SoC.

2.4.1. Système de traitement

Le système de traitement (Processing System PS), indiqué par la section rouge de la figure 2.2, comprend un processeur double noyau ARM cortex A9 ainsi que d'autres composants et périphériques décrits ci-dessous.

a) Unité de processeur d'application

APU (Application Processing Unit) indiqué par la section vert de la figure 2.2, est un ensemble de ressources de traitement associé. Il est constitué principalement de deux cœurs de processeur ARM (Advanced RISC Machine) chacun associé à plusieurs unités de traitement, notamment [9]:

- NEON™ de 128 b également appelé Advanced SIMD (Single Instruction Multiple Data) ou encore MPE (Media Processing Engine), littéralement « moteur de calcul de médias » est une unité de calcul de type SIMD, apporte une accélération majeure des algorithmes DSP (Digital Signal Processing) et multimédia au processeur ARM principal.
- FPU (Floating Point Unit) est un coprocesseur arithmétique dédié aux opérations sur les nombres réels.
- MMU (Memory Management Unit) axée pour translater les adresses virtuelles et physiques.
- Mémoire cache L1 de niveau 1 de 32 KB pour les données et les instructions.
- Mémoire cache L2 de niveau 2 de 512 KB pour les données et les instructions.
- Mémoire SRAM (Static Random Access Memory) de 256KB on OCM (On Chip Memory).
- SNOOP (Snoop Control Unit (SCU)) forme une passerelle entre les cœurs ARM et la mémoire cache de niveau 2 et la mémoire OCM pour maintenir la cohérence de L1 et L2.
- Contrôleur DMA (Direct Memory Access) quatre canaux pour PS (copie de mémoire vers/depuis n'importe quelle mémoire du système) et quatre pour PL (Programmable Logic) « mémoire vers PL, PL vers mémoire ».
- GIC (General interrupt controller) prend les interruptions des périphériques, les hiérarchise et les transmet au cœur de processeur.
- SWD (Serial Wire Debug) processeur implémente le débogage série matériel pris en charge le port de débogage JTAG.

- TTC (Triple Timer Counter) fournit une ressource de synchronisation beaucoup plus flexible. Il peut être utilisé comme minuteries ou pour produire des formes d'onde sur les broches EMIO (Extended Multiplexed Input/Output) ou MOI (Multiplexed Input/Output) du Zynq SoC. Sa source d'horloge sélectionnée via le registre de contrôle d'horloge situé en dessous.

b) Interfaces mémoires

Les interfaces mémoires intègrent plusieurs technologies mémoire aussi bien avec la partie PL qu'avec le monde extérieur, notamment [9] :

- Contrôleur de mémoire dynamique multiprotocole qui prend en charge la norme de 16 bits ou 32 bits vers les mémoires DDR3, DDR3L, DDR2, LPDDR-2.
- Contrôleur de mémoire statique (Static Memory Controller SMC) 8 bits avec prise en charge jusqu'à 64 Mo contient les périphériques d'amorçages principaux tels le contrôleur NAND E/S de 8/16 bits et le contrôleur parallèle SRAM/NOR de largeur du bus de données de 8 bits.
- Contrôleur Quad-SPI (Quad Serial Peripheral Interface) sur la norme SPI (Serial Peripheries Interface) utilise un quadruple (bus de données à quatre bits). Cela augmente le débit de données afin qu'il soit plus rapide qu'une interface SPI standard.

c) Périphériques d'entrées sorties

La communication entre le système de traitement (PS) et les interfaces des périphériques externes est assurée par un bus d'entrées sorties multiplexé MIO. Le tableau 2.1 récapitule toutes les interfaces externes avec une description de chaque périphérique [8] :

Tableau2. 1. Liste des interfaces périphériques d'E/S de connectivité générale disponibles pour le MOI.

Les interfaces I/O	**Description**
UART	Interface de modem de données à bas débit pour la communication série, souvent utilisé pour les connexions Terminal à un PC hôte.
SPI	Norme pour les communications série basées sur une interface à 4 broches. Peut être utilisé en mode maître ou esclave.
CAN	Contrôleur d'interface de bus conforme aux normes ISO 118980-1, CAN 2.0A et 2.0B.

I2C	Conforme à la spécification de bus I2C, version 2. prend en charge les modes maîtres et esclave.
GPIO	Des entrées sorties d'usage général, il y a 3 banques de GPIO, chacune de 26 bits.
GigE(x4)	Périphérique MAC Ethernet, prenant en charge les modes 10 Mbps, 100 Mbps et 1 Gbps.
USB	Conforme à l'USB 2.0 double rôle peut être utilisé comme un USB hôte ou contrôleur de périphérique USB utilisant le même matériel (OTG).
SD	Interface avec la carte mémoire SD prend en charge SD 2.0.

2.4.2. Logique programmable (PL)

La partie PL indiqué par la section orange de la figure 2.2, est basée sur un FPGA série 7 du type ARTIX ou KINTEX suivant le modèle de ZYNQ, possédant toutes les fonctionnalités et les avantages évoqués dans cette section. Elle se compose principalement de [9]:

- Des blocs logiques configurables (CLB) composés d'une tranche contenant les ressources nécessaires de mise en œuvre des circuits logiques séquentiels et combinatoires. Chaque élément se compose d'une table de recherche à 6 entrées LUT (look-up tables).
- Des matrices de commutation pour interconnecter les CLB entre eux.
- Des blocs Entrée-sorties IOBs (Input/output Blocks) supportant une tension de 1.2V à 3.3V.
- Des blocs RAM (Random Access Memory) de 36 Ko à double accès configurable en double 18 Ko.
- Des blocs de traitement du signal numérique DSP48E1 constituent des nombreux multiplicateur / des accumulateurs matériel qui offrent un traitement du signal efficace et à grande vitesse.
- Deux convertisseurs analogique-numérique (XADC) de 12 bits pouvant prendre en charge un taux d'échantillonnage de 1 Msps sur le signal.

Les appareils Zynq APSoC peuvent inclure un ensemble des blocs IP durs ou IP intégrés, au sein du PL. Il existe actuellement cinq blocs Hard IP, à savoir : PCI Express, Interlaken, Ethernet 100G, codec vidéo (VCU) et moniteur système Xilinx (SYSMON). Les interfaces PCI Express, Interlaken et Ethernet 100G fonctionnent dans le PL et disposent de nombreux paramètres configurables pour contrôler la connectivité à un périphérique externe. L'unité VCU, trouvé uniquement dans PL n'a pas de connexion directe avec PS, fournit un encodage

et un décodage vidéo multi-standard. Tandis, le bloc SYSMON permet l'observation du PL pour les questions relatives à la sûreté, à la sécurité et à son environnement physique [9].

2.4.3. Interconnexion entre PS et PL

La communication d'instructions et de données entre le PS et le PL est réalisée à l'aide d'interfaces matérielles dédiées. Il existe de nombreux types d'interfaces ; dont le plus courant est le bus AXI de la norme ouverte Arm AMBA [9]. Ce protocole est actuellement à sa quatrième révision sous le nom AXI4. Les appareils Zynq -7000 utilisent AXI4 à la fois dans le PS et le PL, ils fournissent également un moyen de communication entre les deux sections du dispositif comme le montre la figure 2.3. Il existe trois types de formats pour le bus AXI4, chacun avec un protocole de bus différent, qui sont résumés ci-dessous [8] :

- **AXI4 :** Ce protocole est destiné aux connexions nécessitant des liens mappés en mémoire entre les éléments de traitement et les blocs IP. Il est adapté au transfert de grandes quantités d'informations depuis /vers la mémoire principale.
- **AXI-Lite :** Ce protocole est généralement utilisé pour une communication à faible bande passante avec des registres de contrôle de blocs IP et des éléments de traitement.
- **AXI4-Stream :** Ce protocole prend en charge le streaming de données point à point afin de transférer des données à haute vitesse. Il est particulièrement utile pour le traitement du signal dans les applications vidéo, de communication et de mise en réseau.

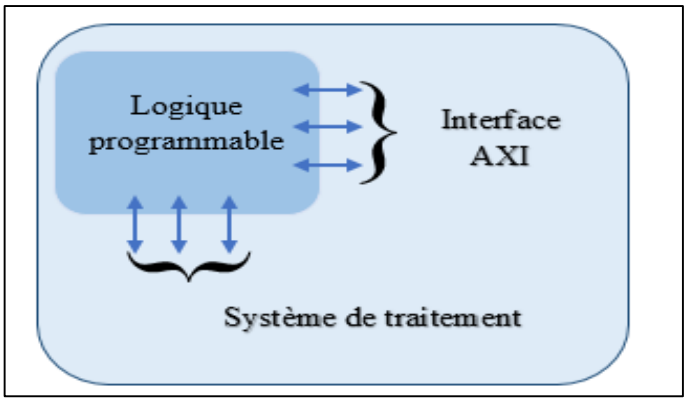


Figure2. 3. Interfaçage entre les deux parties PS et PL de la famille Zynq.

2.5. Description du cadre open source PYNQ

PYNQ (prononcé “pink”) est une infrastructure open-source permet aux programmeurs d'exploiter les capacités des systèmes sur puce entièrement programmables (APSoC) Zynq de Xilinx, sans avoir à concevoir des circuits logiques programmables. En outre, le nom ‘ PYNQ ‘ est dérivé de (Productivité Python pour Zynq). l'APSoC est alors programmé en Python et le code est développé et testé directement sur la carte PYNQ [8].

Ce cadre open source peut aider les utilisateurs à réaliser une variété d'applications intégrées à haute performance, y compris: mise en œuvre de matériel parallèle, traitement vidéo à haute fréquence d'images, l'accélération matérielle algorithme, traitement du signal en temps réel, cryptage, drones et bien plus encore [8].

2.6. Concept général du PYNQ

Le cadre open source de PYNQ permet aux programmeurs de travailler directement avec les FPGA et de profiter des capacités d'accélération logique programmable avec Python. La vue d'ensemble de PYNQ est assez simple, elle se devise en trois couches fonctionnelles notamment :

- Couches supérieure (Applications).
- Couches intermédiaires (Software).
- Couche inférieure (Hardware).

La figure 2.4 illustre le concept général du framework PYNQ et montre comment ses trois couches sont liées à celles du système embarqué basé sur Zynq. Les trois couches sont brièvement expliquées ci-dessous [9]:

2.6.1. Couche supérieure (Applications)

La couche supérieure du PYNQ est facilitée par un ou plusieurs notebooks Jupyter. En bref, le Jupyter est une évolution du projet open source IPython conçu pour fonctionner sur n'importe quelle plate-forme informatique et système d'exploitation. Cet objectif peut être atteint en adoptant une architecture Web, qui n'a rien à voir avec le navigateur. PYNQ intègre l'infrastructure de bloc-notes Jupyter pour exécuter directement le noyau sur le processeur ARM du périphérique Zynq. Le serveur Web accède au noyau via un ensemble d'outils basés sur le navigateur qui fournissent des tableaux de bord, des terminaux bash, des éditeurs de code et des

blocs-notes Jupyter. Dans un notebook Jupyter PYNQ, le développeur doit créer ses propres fonctionnalités personnalisées en écrivant son propre code Python et en le réutilisant de manière sélective via de nombreuses bibliothèques Python open source disponibles, y compris des packages tels que NumPy, Matplotlib, OpenCV.

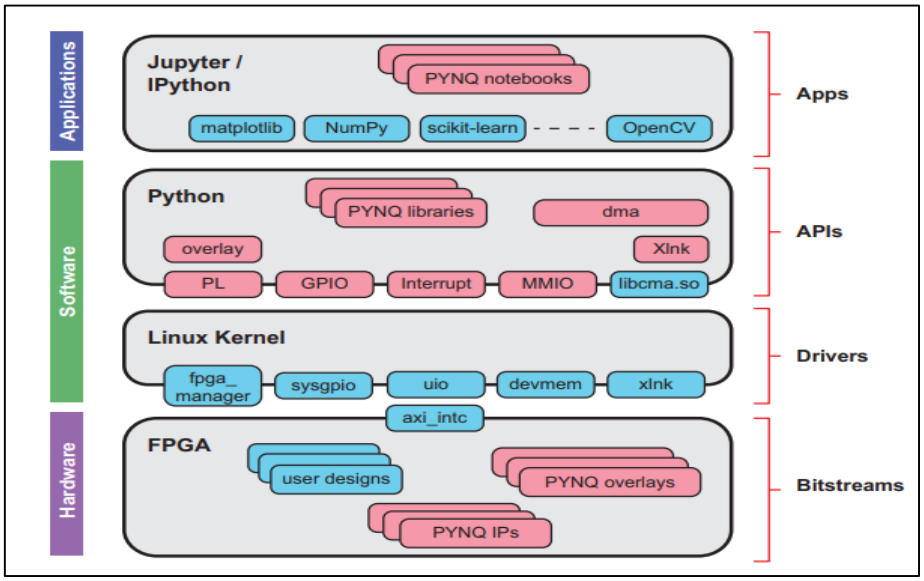


Figure2. 4. Concept général du cadre open source PYNQ.

2.6.2. Couches intermédiaires (Software)

Les couches intermédiaires du PYNQ se composent du logiciel Python, du système d'exploitation et des pilotes logiciels de bas niveau. Dans la couche intermédiaire supérieure, le framework PYNQ comprend des bibliothèques Python et des API pour interagir avec divers éléments des systèmes basés sur Zynq. Par exemple, il existe des API Python pour télécharger des superpositions (fichiers bitstream) sur le PL, pour communiquer avec les ressources GPIO et gérer les interruptions générées dans le PL. Il existe également des API Python pour les transferts mappés en mémoire (MMIO) et les transferts DMA.

Dans la couche intermédiaire inférieure, le framework PYNQ comprend un système d'exploitation basé sur Linux, des chargeurs de démarrage pour lancer le démarrage du système et un serveur Web pour héberger les ordinateurs portables Jupyter. Elle comprend également un ensemble de pilotes pour interagir avec les éléments du système matériel Zynq.

2.6.3. Couche inférieure (Hardware)

La couche inférieure du PYNQ représente une conception de système matériel, sur lequel il sera créé sur Vivado à l'aide d'IP Integrator et des outils de conception associés, puis générée dans un fichier bitstream (*.bit). Le fichier bitstream est transféré sur la carte mémoire insérée dans la carte cible. Le processus de programmation du système matériel sur le PL peut alors être lancé directement à partir d'un bloc-notes Jupyter susmentionné sur la couche supérieure de tel concept (fonctionnant sur le PS).

En tenant compte de tous ces composants réutilisables, on peut définir le terme 'bibliothèques matérielles'. Il s'agit d'un terme générique faisant référence à l'ensemble d'adresses IP et de superpositions disponibles dans le cadre du PYNQ pour une réutilisation flexible. Les détails des conceptions matérielles et leurs fonctionnalités peuvent être résumés en Python via une interface de programmation d'application (API) ou les bibliothèques matérielles peuvent être considérées comme similaires aux bibliothèques logicielles.

2.7. Carte PYNQ-Z2

Il existe quatre types de carte de développement FPGA en améliorant la productivité de la conception de systèmes Zynq pris en charge (PYNQ-Z1, PYNQ-Z2, ZCU104 et ZCU111) [9]. Actuellement, la carte PYNQ-Z2, illustrée sur la figure 2.5, est la plus populaire et la plus accessible. Elle prend le cœur ZYNQ XC7Z020 FPGA, utilisant les avantages de la logique programmable et du processeur Arm de ZYNQ, pour construire un système intégré puissant constitué généralement de [10] :

- Source d'alimentation sélectionnée via le commutateur J9 en le mettant en mode USB ou mode du régulateur d'alimentation externe REG (External power REGulator/Battery). Elle prend en charge 7VDC à 15VDC et (12V recommandé). Les LEDs rouges s'allumeront immédiatement pour confirmer que la carte est sous tension.
- En outre la carte pynq-Z2 peut être alimentée par le port Micro-USB. Ce dernier prend en charge l'émetteur-récepteur asynchrone universel USB-UART (attaché au connecteur J8 PROG UART) et un port de programmation et débugge JTAG (Joint Test Action Group).
- Port USB Host agit comme un contrôleur du système USB qui permet de connecter différents périphériques USB tels qu'un lecteur flash, un clavier ou une souris USB.

- Port Ethernet sert à connecter le PYNQ à l'Internet via un câble Ethernet (RJ-45), cette méthode est plus facile à mettre en place que le Wifi. Il peut fournir un accès Internet plus rapide (10/100/1000 Mbps).

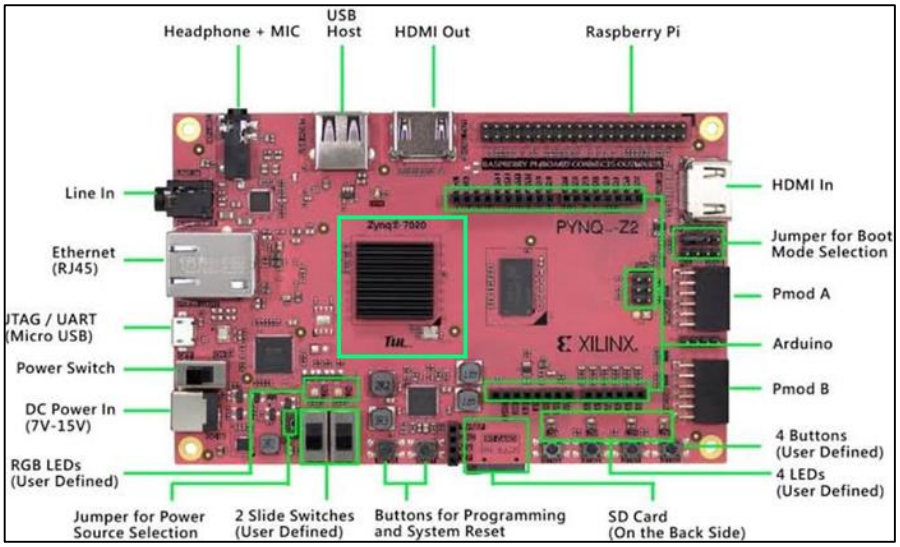


Figure2. 5. Composants de base de la carte PYNQ-Z2.

- Deux ports HDMI ; un port HDMI comme entrée et l'autre port comme sortie, en fournissant une interface vidéo numérique au Pynq (relié avec des cartes Vidéos comme l'ordinateur, télévisions ...).
- Connecteur Raspberry Pi à 40 broches avec 28 broches connectées à l'appareil Zynq. Ces broches de pilotage axées pour contrôler les périphériques connectés à une interface RPi compatibles aux broches PL. Les broches RPi peuvent également être utilisées comme broches à usage général pour se connecter au matériel personnalisé à l'aide de fils.
- Deux ports Pmod sur le PYNQ-Z2 (étiquetés A et B), chaque port Pmod à 12 broches fournit deux signaux VCC de 3,3 V (broches 6 et 12), deux signaux de masse GND (broches 5 et 11) et huit signaux logiques.
- Un connecteur de blindage Arduino à 26 broches connectées à la partie PL du Zynq. Les broches peuvent être utilisées comme GPIO. Six des broches Arduino (étiquetées A0-A5) peuvent également être utilisées comme entrées analogiques asymétriques avec une plage d'entrée de 0V-3,3V, et 14 autres (étiquetées AR0-AR13) peuvent être utilisées comme entrées analogiques différentielles.

- Flash SPI quadruple (Quad SPI) de vitesses de bus jusqu'à 104 MHz, prenant en charge les taux de configuration Zynq à 100 MHz alimentés par 3,3 V. Il est utilisé pour initialiser et démarrer le sous-système PS ainsi que pour configurer le sous-système PL.
- Source audio peut être sélectionnée, soit line-in ou MIC+ HP, et l'audio entrant dans la carte peut être soit enregistré dans un fichier, soit lu sur la sortie casque.
- Une SD (minimum 8 Go recommandé) utilisée pour les applications nécessitent un stockage de mémoire externe non volatile dont laquelle le système d'exploitation sera installé.
- La carte PYNQ-Z2 comprend deux LED tricolores (LED4 et LED5) se compose de trois LED internes rouge, vert et bleu. Les signaux d'entrée vers les LED RGB internes sont pilotés par le Zynq PL via un transistor, qui inverse les signaux. Elle contient également deux commutateurs DIP (Dual In-line Package) « SW0, SW1 », quatre boutons poussoirs (BTN0, BTN1, BTN2, BTN3) et quatre LED individuelles connectées au PL (LED0, LED1, LED2, LED3).
- Bouton-poussoir SRST qui réinitialise la carte de manière externe, en effaçant tout le contenu de la mémoire PS, y compris l'OCM. ainsi que le PL.
- Bouton-poussoir PROG qui réinitialise seulement la partie PL. De ce fait, elle restera non configurée jusqu'à ce qu'elle soit reprogrammée par le processeur ou via JTAG.

2.8. Conclusion

Dans ce chapitre, on a passé en revue les circuits logiques programmables FPGA, en particulier du fabricant Xilinx, où on a examiné également la famille zynq soc 7000, ses caractéristiques techniques et ses interfaces. Ensuite, on a introduit le cadre open source pour PYNQ, en expliquant la motivation de cette méthodologie de développement basée sur Python en tant qu'amélioration de la productivité pour la conception de systèmes Zynq. Les différents composants du framework PYNQ ont été expliqués, y compris le concept de superpositions matérielles et l'utilisation des blocs-notes Jupyter comme interface utilisateur interactive. Cela va permettre d'aborder la partie simulation et implémentation de notre système sur le prochain chapitre.

Chapitre Ⅲ :
Implémentation et résultats de simulation

3.1. Introduction

Dans ce chapitre, on va aborder la phase d'implémentation matérielle d'un système OFDM. Cette tâche n'est pas trivial et elle requiert des outils de simulation et de calcul assez précis et capable de produire des résultats dans un lapse de temps raisonnable.

Le présent chapitre est alors divisé en quatre volets. Le premier volet présente l'environnement logiciel utilisé dans ce projet. Le deuxième volet présente les différents algorithmes implémentés pour la gestion du système ainsi que les résultats des simulations des implémentations effectuées. Le troisième volet propose les différents outils de test des performances du système mis en œuvre. Ainsi, le dernier volet fournit une comparaison des performances du système OFDM implémenté avec ceux des architecteurs rapportés dans la littérature.

3.2. Environnement logiciel

Avant de commencer la description de notre système, on va tout d'abord présenter les logiciels de conception et de simulation utilisés lors de la réalisation de ce travail.

3.2.1. Vivado Design Suite

Vivado Design Suite est une suite logicielle produite par la société technologique américaine Xilinx et qui offre plusieurs façons d'accomplir les tâches impliquées dans la conception, la mise en œuvre et la vérification des appareils Xilinx. À cette fin, Xilinx Vivado vise à synthétiser et analyser des conceptions en langage de description matérielle (HDL) afin de fournir des modules supplémentaires pour le développement des systèmes sur puce et des outils de synthèse de haut niveau. Les fonctionnalités d'analyse et la vérification de conception incluent la programmation, le débogage, la simulation logique, la planification des E/S, la visualisation de la logique de conception, la modification des résultats de mise en œuvre [11].

3.2.2. Xilinx System Generator

Xilinx System Generator est un outil de conception complémentaire à Simulink qui permet aux concepteurs de concevoir, simuler et développer des systèmes DSP hautes performances, non seulement les algorithmes DSP peuvent être conçus et simulés, mais un code HDL synthétisable peut également être généré à l'aide du codeur HDL pour les FPGA Xilinx [12]. Cet outil de conception se compose d'un ensemble de blocs Xilinx, en tant que bibliothèque

Simulink qui peut être mappée directement dans le matériel FPGA cible. Par conséquent, Xilinx System Generator est considéré comme un outil de haut niveau pour la conception de systèmes DSP en fournissant la modélisation du système et la génération automatique de code HDL à partir de MATLAB ou Simulink [13].

3.2.3. Jupyter Notebook

Jupyter Notebook est le dernier environnement de développement interactif basé sur le Web pour les blocs-notes. Il offre une expérience simple, rationalisée et centrée sur les documents avec un effort raisonnable. En bref, son interface flexible permet aux développeurs de configurer et d'organiser des flux de travail en science de données, en informatique, en journalisme informatique et en apprentissage automatique. Jupyter permet aux utilisateurs de créer des documents interactifs, appelés " notebooks ", qui sont servis via un navigateur Web standard. Ces blocs-notes peuvent contenir une variété de contenus différents, y compris du code exécutable en direct et visualisations, ainsi que la documentation textuelle, graphique et mathématique, et ils sont organisés en cellules [9].

3.3. Implémentation hardware

Dans cette section, on va présenter les principaux algorithmes qu'on a conçus pour la gestion optimale de notre système OFDM classique et améliore OFDM-CP. Ces algorithmes sont donnés sous forme de schémas blocs.

3.3.1. OFDM Classique

Dans cette section, on va présenter un aperçu de chaque bloc utilisé pour concevoir l'émetteur et le récepteur OFDM. L'algorithme et les fonctionnalités de ce système sont décrits brièvement ci-dessous.

3.3.1.1. Module de Codage QAM

Concernant le codage, on a choisi une modulation QAM 16 avec une constellation similaire à celle de MATLAB, les quatre bits (Code gray) en série des données numériques d'origine sont codés sur l'axe de phase I et de quadrature Q afin de constituer un symbole complexe QAM, comme illustré sur la figure 3.1.

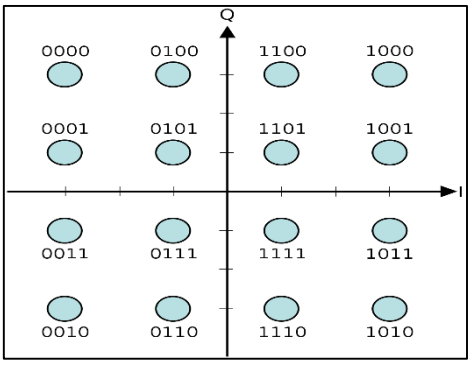


Figure3. 1. Constellation du codage QAM 16.

Comme indiquer sur la figure 3.2, le modulateur QAM est implémenté en fonction de données d'entrée séries de 32 bits qui seront mappées à chaque cycle d'horloge à l'aide de l''instruction « case » qui permet d'effectuer une suite de tests d'égalité consécutifs pour toutes les cas de la constellation. Ce bloc génère des sorties séquentielles complexes dont les parties réelles et imaginaires représentées séparément en virgule fixe de type $\boldsymbol{Q_{16,16}}$. Ces deux parties prennent seulement les 16 bits entière vu absence du partie fractionnaire dans de la constellation.

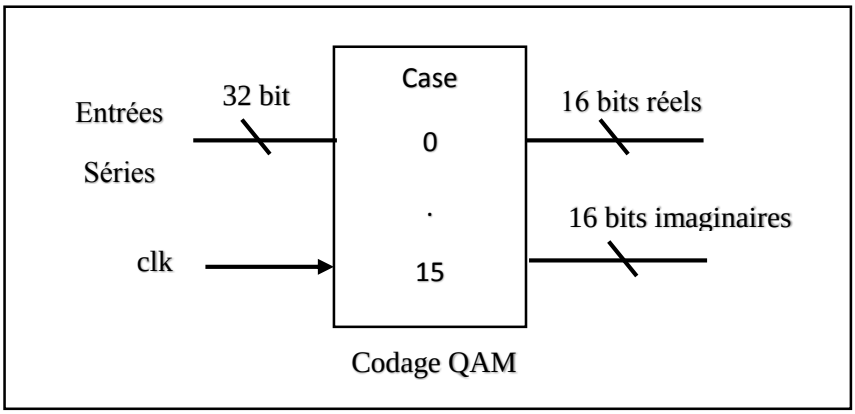


Figure3. 2. Implémentation de la modulation QAM 16.

3.3.1.2. Module SIPO (Serial Input to Parallel Output)

Le registre à décalage d'entrée série vers sortie parallèle (SIPO) est une sorte de circuits logiques qui prennent des données d'entrée numériques séries de 16 bits et les convertit en données numériques parallèles. Ce module fonctionne à l'aide d'un compteur qui compte les données réelles et imaginaires 16 fois et les accumule dans des registres réels et imaginaires pour être prêts pour les modules IFFT et FFT successivement. La figure 3.3 décrit le schéma fonctionnel du module SIPO.

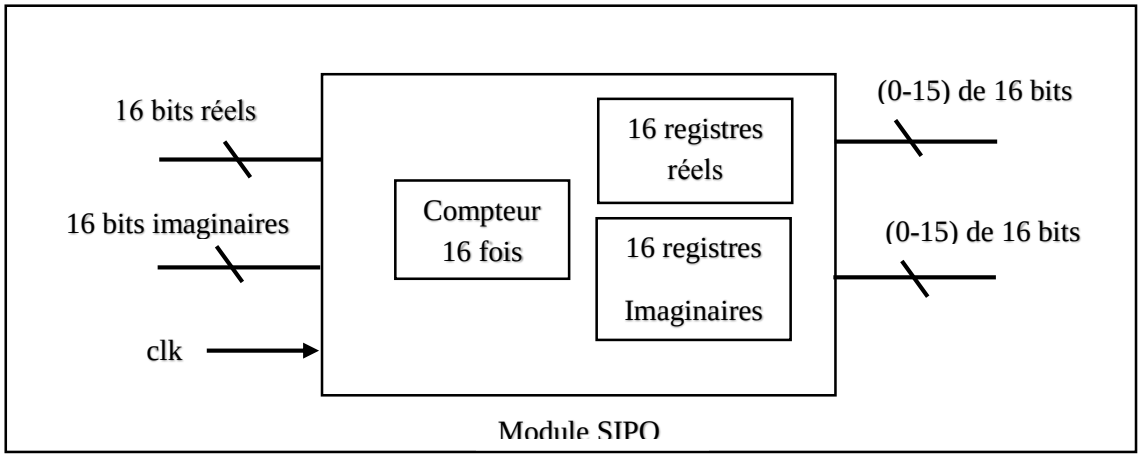


Figure3. 3. Implémentation du module série vers parallèle.

3.3.1.3. Modules IFFT/FFT

L'algorithme de la transformation de Fourier rapide direct et inverse peut être programmé de multiples façons. On va détailler une version particulière, appelée "algorithme papillon" en utilisant la technique Radix-2 DIT (Decimation-in-time). Pour mieux comprendre les subtilités de cet algorithme, prenant un exemple de N=16 illustré dans figure 3.4.

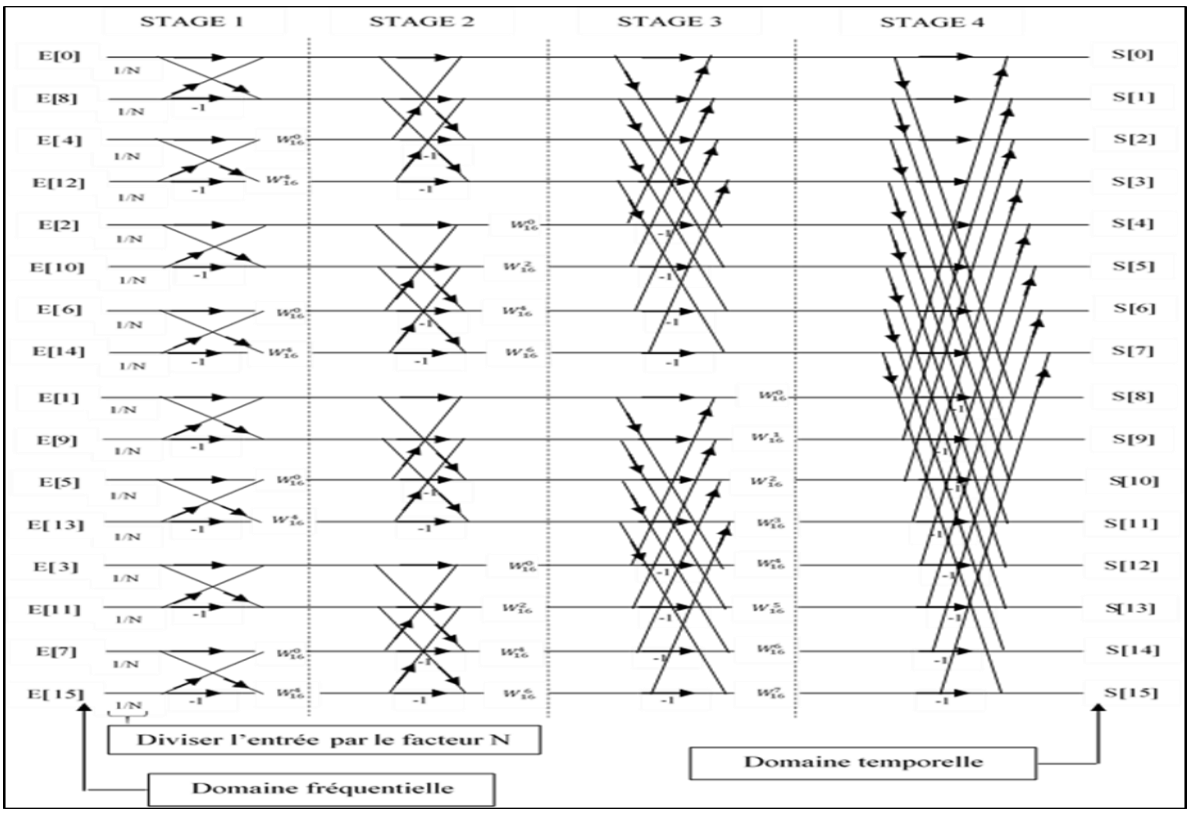


Figure3. 4. Graphique de flux IFFT à 16 points.

.

À partir de la figure 3.4, E(0) à E(15) de la position gauche de l'organigramme sont indiqués en tant que des variables d'entrée fréquentiels multipliés par un facteur de 1/N pour le calcul IFFT et S(0) à S(15) sur la position droite de l'organigramme sont désignées comme des variables de sortie temporels. L'opération d'addition sera exécutée lorsque deux flèches convergentes vers le haut et l'opération de soustraction sera exécutée lorsque deux flèches convergentes vers le bas tandis que la multiplication sera effectuée lorsque la flèche finira par un facteur de type W_{16}^{i}.

Les modifications apportées du module FFT par rapport au module IFFT sont les entrées ne sont pas multipliées par un facteur de 1/N et les facteurs de rotation sont remplacés par leurs conjugués complexes.

Le nom " algorithme papillon " vient d'une certaine similitude entre son graphe et le papillon (voir la figure 3.5). Chaque papillon se compose de deux entrées (A, B) et donne deux sorties (C, D), W est le facteur de rotation. Pour calculer les sorties, les équations données ci-dessous sont utilisées :

$$C = (A + B) * W \quad \textbf{(3.1)}$$

$$D = (A - B) * W \quad \textbf{(3.2)}$$

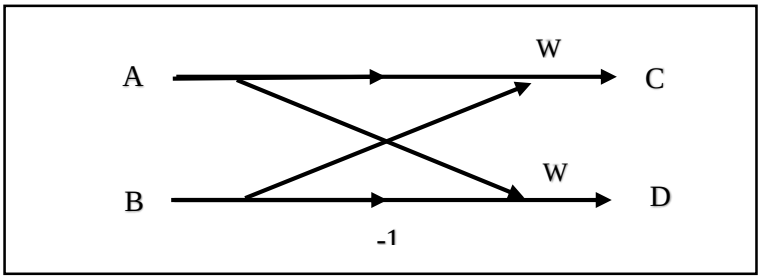


Figure3. 5. Unité papillon Radix 2.

Le facteur de rotation W dépend entièrement du nombre de points IFFT utilisés pour le calcul. Les valeurs de ce facteur de l'IFFT à 16 points ont été calculées selon l'équation (3.3) avec des valeurs positive de i. Ensuite les résultats obtenues sont converties en forme virgule fixe de type $\boldsymbol{Q_{16,16}}$ et positionnés de manière appropriée dans le tableau 3.1.

$$W_{16}^{i} = \cos(\frac{2\pi i}{16}) - j\sin(\frac{2\pi i}{16}) \quad \textbf{(3.3)}$$

Tableau3. 1. Tableau des facteurs de rotation pour IFFT (N=16).

Partie réel		**Partie imaginaire**	
$\cos(0) = 1$	00010000	$\sin(0) = 0$	00000000
$\cos(\frac{\pi}{8}) = 0.923$	0000EC83	$\sin(\frac{\pi}{8}) = 0.382$	000061F8
$\cos(\frac{\pi}{4}) = 0.707$	0000B505	$\sin(\frac{\pi}{4}) = 0.707$	0000B505
$\cos(\frac{3\pi}{8}) = 0.382$	000061F8	$\sin(\frac{3\pi}{8}) = 0.923$	0000EC83
$\cos(\frac{\pi}{2}) = 0$	00000000	$\sin(\frac{\pi}{2}) = 1$	00010000
$\cos(\frac{5\pi}{8}) = - 0.382$	FFFF9E08	$\sin(\frac{5\pi}{8}) = 0.923$	0000EC83
$\cos(\frac{6\pi}{8}) = - 0.707$	FFFF4AFB	$\sin(\frac{6\pi}{8}) = 0.707$	0000B505
$\cos(\frac{7\pi}{8}) = - 0.923$	FFFF137D	$\sin(\frac{7\pi}{8}) = 0.382$	000061F8

Les valeurs de facteur de rotation W pour FFT à 16 points sont calculées de même méthode que la précédente sauf qu'il suffit d'inverser le signe de i de son équation. Le tableau 3.2 aidera à mieux comprendre le changement.

Tableau3. 2. Tableau des facteurs de rotation pour FFT (N=16).

Partie réel		**Partie imaginaire**	
$\cos(0) = 1$	00010000	$\sin(0) = 0$	00000000
$\cos(-\frac{\pi}{8}) = 0.923$	0000EC83	$\sin(-\frac{\pi}{8}) = - 0.382$	FFFF9E08
$\cos(-\frac{\pi}{4}) = 0.707$	0000B505	$\sin(-\frac{\pi}{4}) = - 0.707$	FFFF4AFB
$\cos(-\frac{3\pi}{8}) = 0.382$	000061F8	$\sin(-\frac{3\pi}{8}) = - 0.923$	FFFF137D
$\cos(-\frac{\pi}{2}) = 0$	00000000	$\sin(-\frac{\pi}{2}) = -1$	FFFF0000
$\cos(-\frac{5\pi}{8}) = 0.382$	000061F8	$\sin(-\frac{5\pi}{8}) = - 0.923$	FFFF137D
$\cos(-\frac{6\pi}{8}) = 0.707$	0000B505	$\sin(-\frac{6\pi}{8}) = - 0.707$	FFFF4AFB
$\text{scos}(-\frac{7\pi}{8}) = 0.923$	0000EC83	$\sin(-\frac{7\pi}{8}) = - 0.382$	FFFF9E08

L'implémentation pas à pas des modules de base IFFT et FFT à 16 points illustré dans la figure 3.6 , cette dernier est identique pour les deux modules dont le symbole OFDM est généré

en implémentant l'algorithme IFFT qui a été abordé ci-dessus puis le récupère en implémentant également l'algorithme opposé FFT, chaque stage mentionné dans le schéma bloc de l'implémentation des modules suscitées contient huit papillon chacun de ces papillon prend des entrées parallèles réelles et imaginaires de 16 bits. En obtenant enfin des sorties parallèles réelles et imaginaires de 32 bits car elles contiennent à la fois les deux parties entier et fraction.

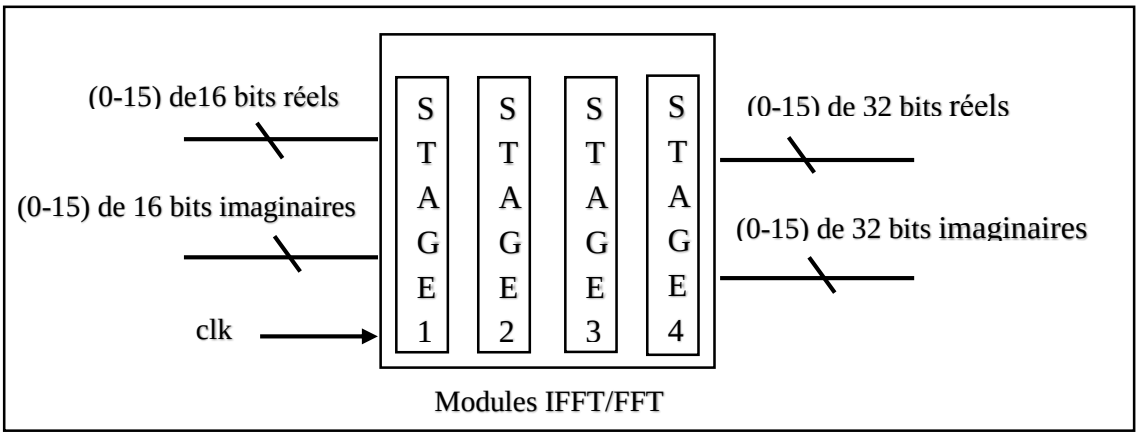


Figure3. 6. Implémentation du module IFFT/FFT à 16 points.

3.3.1.4. Module PISO (Parallel Input Serial Output)

Le registre à décalage d'entrée parallèle vers sortie série effectue une opération inverse de SIPO. Ces circuits numériques prennent les données d'entrée numériques parallèles et les changent simultanément à la forme série effectuée à l'aide d'un autre compteur qui compte 16 fois, mais dans ce cas, à chaque incrément de ce compteur, la valeur série est libérée. Ce type de module est toujours placé avant que les données ne soient transmises sur le canal de transmission via le convertisseur numérique-analogique. La figure 3.7 montre le schéma fonctionnel du module convertisseur parallèle-série.

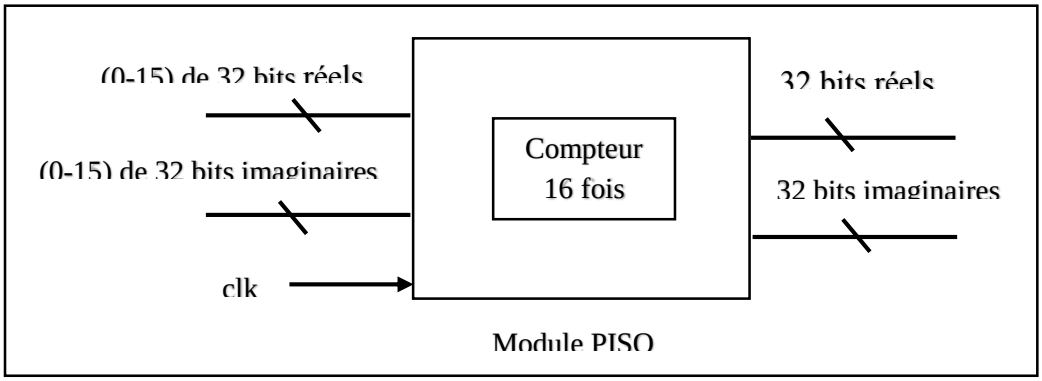


Figure3. 7. Implémentation du module parallèle vers série.

3.3.1.5. Module de décodage QAM16

Les signaux entrants dans ce module sont démappés selon le schéma de mappage QAM utilisé au niveau de l'émetteur. La mise en œuvre de ce module consiste à prendre des seuils de décision mentionnés dans la figure 3.8 afin de récupérer l'information transmise.

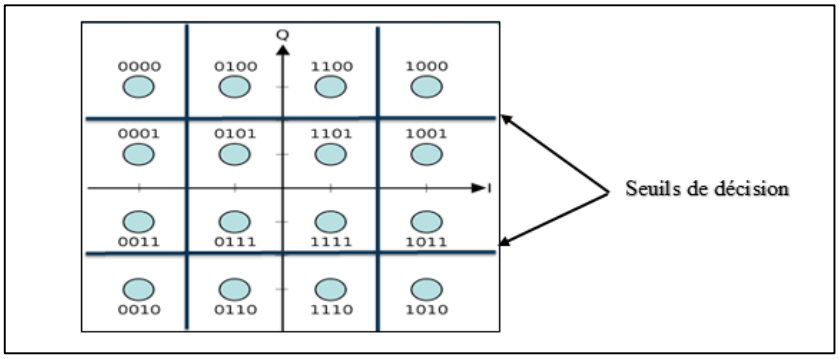


Figure3.8. Démodulation QAM 16.

L'implémentation du décodage illustrée dans la figure 3.8 est basée sur une possession de conditions bien précises affectées aux entrées réelles et imaginaires de 32 bits, la sortie libérée dans ce module est de 32 bits pour l'utilisation des AXI DMA (voir la figure 3.9).

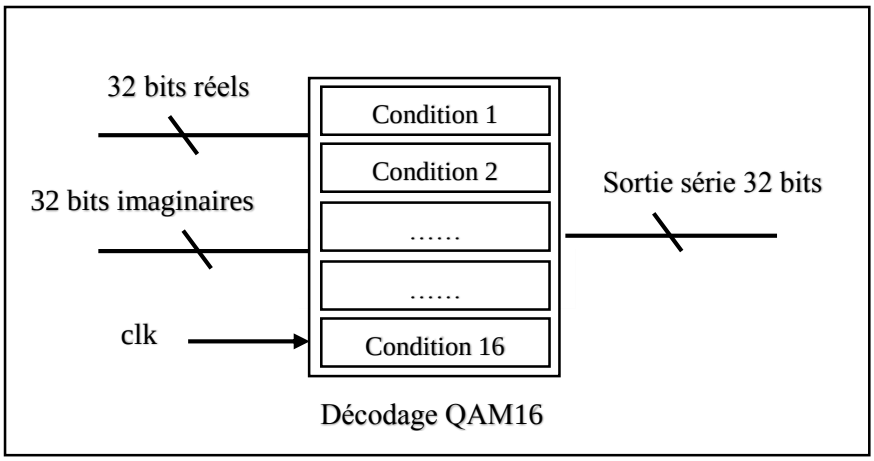


Figure3.9. Implémentation de la démodulation QAM 16.

Pour bien expliquer la procédure d'effectuer ces conditions, on distingue trois types de conditions présentées dans le tableau 3.3 avec un exemple illustratif pour chacune des conditions.

Tableau3. 3. Tableau des conditions de démodulation QAM 16.

Type (nbr conditions)	Constellation correspondantes	Exemple
Type 01 : 2 conditions	0, 2, 8,10	Si ((réel ≤ -2) et (imaginaire ≥ 2)) pour 0
Type 02 : 3 conditions	1, 3, 4, 6, 9, 11, 12, 14	Si ((réel ≤ -2) et (imaginaire ≥ 0) et (imaginaire < 2)) pour 1
Type 03 : 4 conditions	5, 7, 13, 15	Si ((réel ≤ 0) et (imaginaire ≥ 0) et (réel > -2) et (imaginaire < 2)) pour 5

3.3.2. OFDM CP

L'implémentation de l'OFDM-CP est similaire à celle implémentée en OFDM classique avec les mêmes blocs utilisés avec l'ajout de deux modules. L'un au niveau de l'émetteur qui sert à ajouter un préfixe cycle CP et l'autre à la réception afin d'éliminer ce CP.

3.3.2.1. Module d'insertion du CP

Du coté émetteur, le symbole OFDM est etendu de manière cyclique afin d'empêcher les interférences intersymboles (ISI) dues à l'effet du multi trajets lorsque le signal OFDM est transmis dans un canal dispersif. L'implémentation de ce processus est effectuée en ajoutant quatre échantillons nuls au début de du symbole OFDM, comme le montre la figure 3.10. Les 16 entrées parallèles réelles et imaginaires sont affectées aux 16 dernières sorties parallèles réelles et imaginaires (4-19) de sorte que le CP est inséré alors dans les quatre premières sorties (0-3).

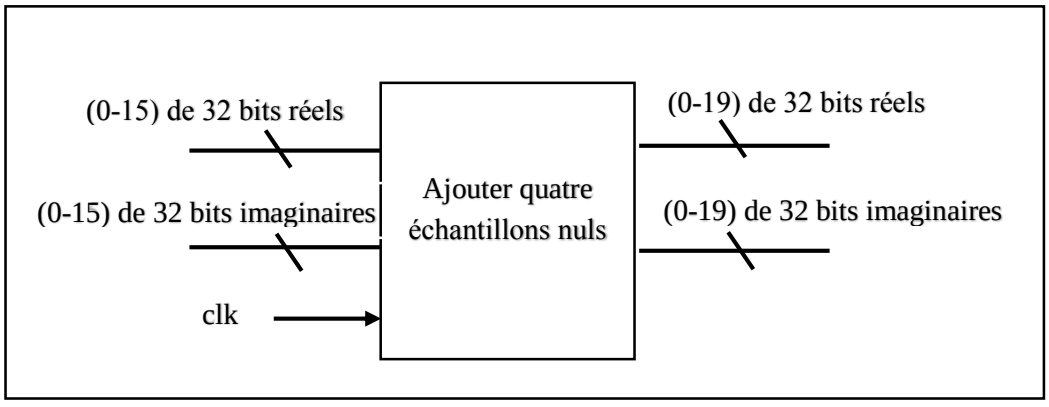


Figure3.10. Implémentation du module additive CP.

3.3.2.2. Module de suppression du CP

De l'autre côté, le symbole OFDM étendu au niveau de l'émetteur est périodiquement rétréci au niveau de récepteur avant le module FFT, ce qui assure un calcul de démodulation OFDM bien précis. La mise en œuvre de ce module est la procedure oppposée effectuée en éliminant les quatre premiers échantillons nuls du début du symbole OFDM, comme l'indique la figure 3.11. Les 16 dernières entrées parallèles réelles et imaginaires (4-19) sont affectées aux 16 sorties parallèles réelles et imaginaires de sorte que les quatre échantillons nuls CP d'entrée (0-3) sont supprimés.

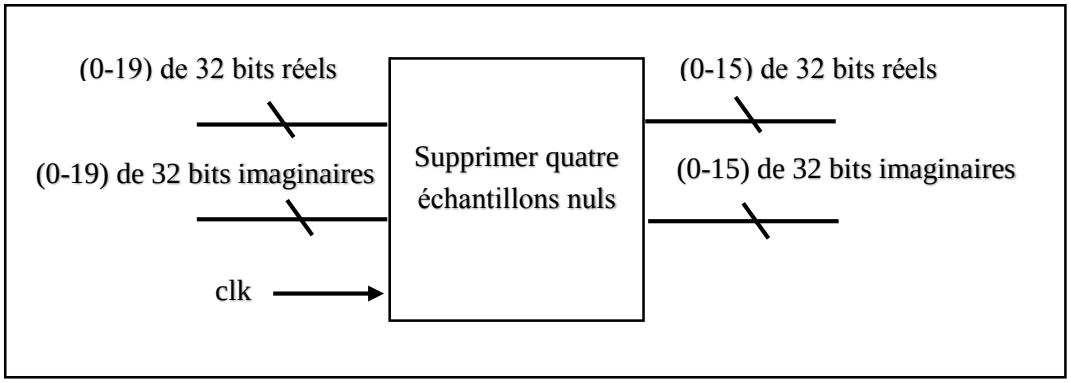


Figure3.11. Implémentation du module suppression CP.

3.4. Résultats de simulation et consommation de l'implémentation hardware

Les modules précédemment mis en œuvre sont simulés à l'aide du logiciel Vivado Xilinx et leurs résultats de simulation et consommation du FPGA sont rassemblés de manière exhaustive dans cette section. On a spécifiquement choisi ce logiciel de simulation afin de visualiser le bon déroulement des systèmes OFDM classique et amélioré (OFDM-CP).

3.4.1. Module de codage QAM 16

La première simulation est celle de la modulation QAM 16 dont le déroulement est décrit dans la sous-section 3.1.1, où on définit les parties réelles et imaginaires de chaque entrée de 0 jusqu'à 15. Les sorties de ce module sont illustrées par l'exemple de la simulation présenté dans la figure 3.12.

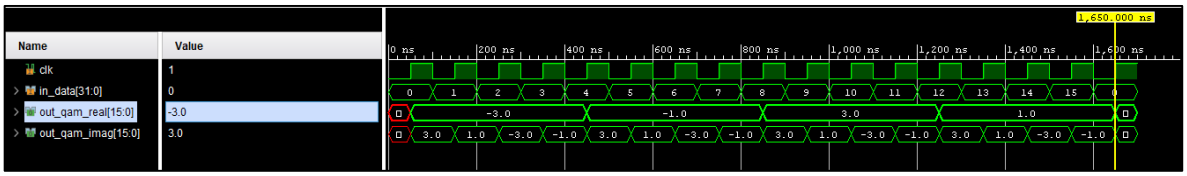


Figure3.12. Simulation du module de codage QAM 16.

La consommation du FPGA pour l'implémentation du module de codage QAM 16 est illustrée dans le tableau de la figure 3.13. D'après ce tableau, on note que ce module utilise 5 bascules Flip Flop sur 106400 (0.01 %), 9 LUT sur 53200 (0.02 %), 0 BUFG sur 32 (0.00 %) et 0 DSP48E1 sur 220 (0.00%).

Ressource	Estimation	Disponible	Utilisation
FF	5	106400	0,01%
LUT	9	53200	0,02%
BUFG	0	32	0.00%
DSP	0	220	0.00%

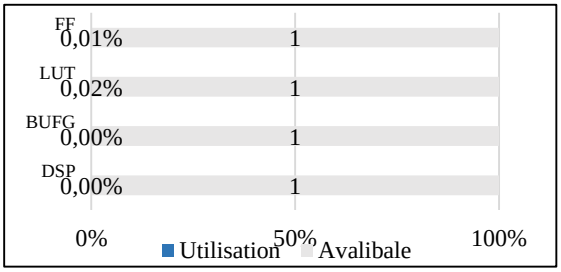


Figure3.13. Consommation du FPGA pour l'implémentation du module de codage QAM 16.

3.4.2. Module SIPO

Cette partie est réservée à la simulation de l'unité SIPO dont la progression est décrite dans la sous-section 3.1.2. Pour une séquence d'entrée de 16 éléments pré-modifiée par un codeur QAM 16, la sortie du SIPO est illustrée par l'exemple suivant. Ici, les données d'entrée sont au format série réels **'-3, -3, -3, -3, -1, -1, -1, -1, 3, 3, 3, 3, 1, 1, 1'** et imaginaires **'3, 1, -3, -1, 3, 1, -3, -1, 3, 1, -3, -1, 3, 1, -3, -1'.** La conversion est déclenchée lorsque l'horloge est égale à 1. Les sorties réels et imaginaires de SIPO après le traitement sont illustrés respectivement dans la figure 3.14 pour la partie réelle et dans la figure.3.15 pour la partie imaginaire.

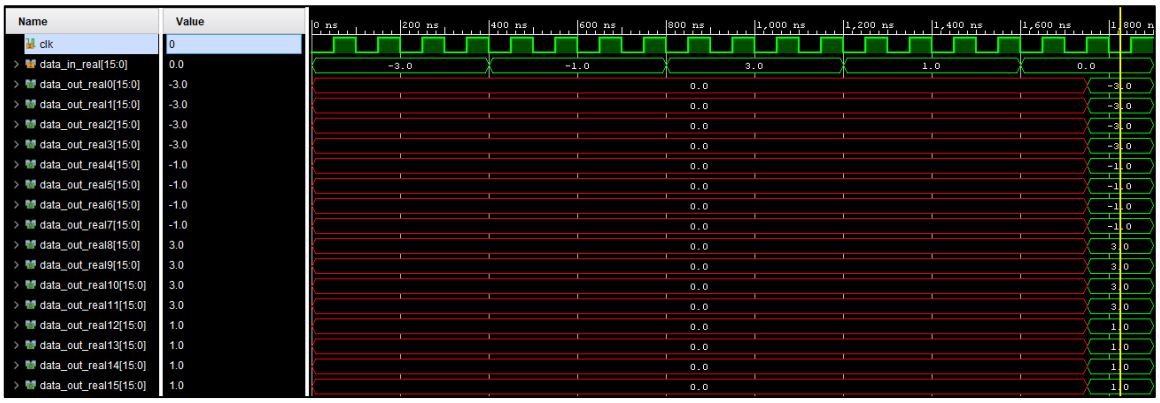


Figure3.14. Simulation de la partie réelle du module SIPO.

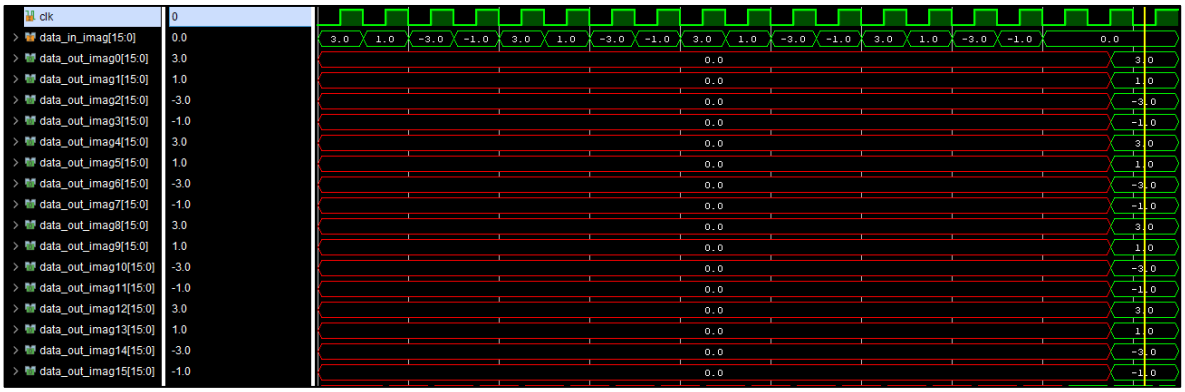


Figure3.15. Simulation de la partie imaginaire du module SIPO.

Le rapport de consommation du FPGA de la conception du module SIPO est fourni dans le tableau de la figure 3.16. Étant donné que ce module utilise 1064 bascules Flip Flop sur 106400 (1 %), 7 LUT sur 53200 (0.01 %), 0 BUFG sur 32 (0.00 %) et 0 DSP48E1 sur 220 (0.00%).

Ressource	Estimation	Disponible	Utilisation
FF	1064	106400	1,00%
LUT	7	53200	0,01%
BUFG	0	32	0.00%
DSP	0	220	0.00%

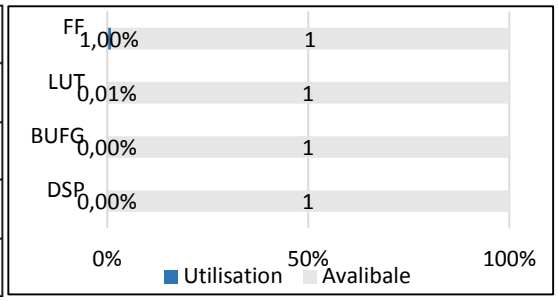


Figure3. 16. Consommation du FPGA pour l'implémentation du module SIPO.

3.4.3. Module IFFT

La simulation suivante est celle du module IFFT dont le déroulement est décrit dans la sous-section 3.1.4. Pour une séquence d'entrée de 16 éléments de 32 bits, les sorties du module IFFT sont illustrées par l'exemple suivant. Donc, chacune des valeurs réelles pré-modifiées par le module SIPO est affectée à l'entrée réelle du module IFFT et de la même manière pour les parties imaginaires. Les sorties de l'IFFT correspondantes aux données entrées sont décrites respectivement dans la figure 3.17 pour la partie réelle et dans la figure 3.18 pour la partie imaginaire.

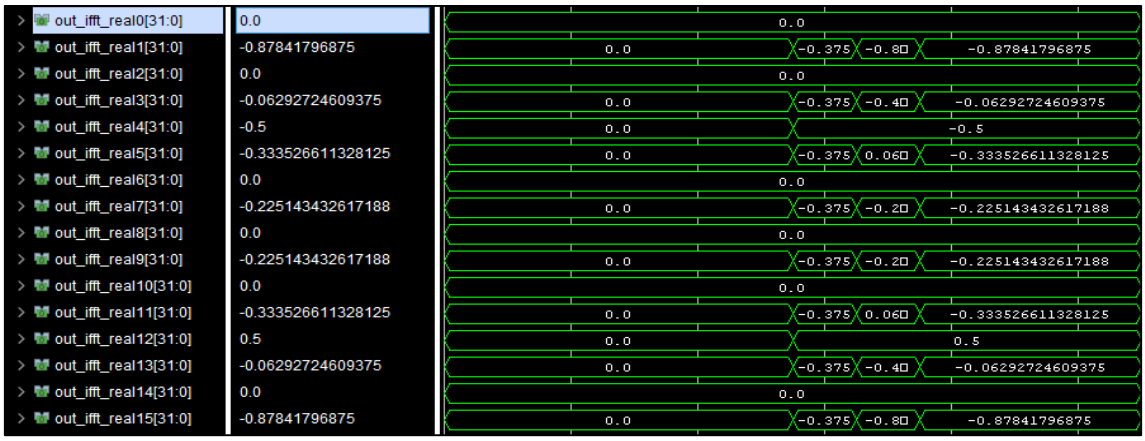


Figure3. 17. Simulation de la partie réelle du module IFFT.

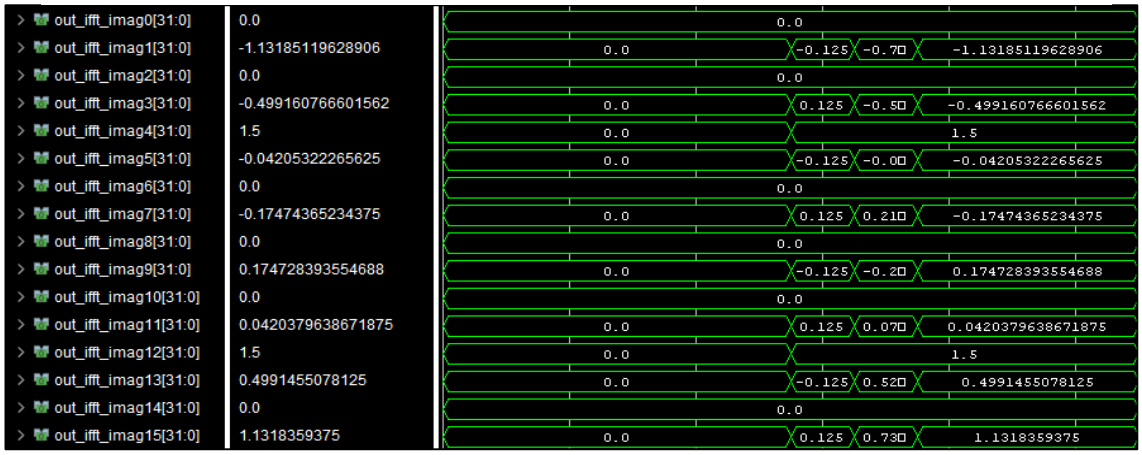


Figure3.18. Simulation de la partie imaginaire du module IFFT

Le résumé de l'utilisation de FPGA et le rapport de consommation de la conception du module IFFT sont fournis dans le tableau de la figure 3.19. Avec 4.80 % des bascules et 18,48 % des LUT sont consommés dans cette conception.

Ressource	Estimation	Disponible	Utilisation
FF	5104	106400	4,80%
LTU	9831	53200	18,48%
BUFG	0	32	0.00%
DSP	0	220	0.00%

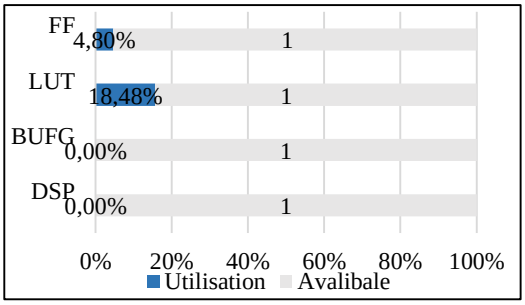


Figure3.19. Consommation du FPGA pour l'implémentation du module IFFT.

3.4.4. Module d'insertion du CP

La simulation suivante est celle du module d'insertion du CP dont le déroulement est décrit dans la sous-section 3.2.1. Pour une séquence d'entrée de 16 éléments de 32 bits, les sorties du ce module sont illustrées par l'exemple de la figure 3.21. On ajoute quatre échantillons nuls au début du symbole OFDM pour les parties réel et imaginaire. Les 16 dernières sorties sont les 16 sorties prétraitées par le module IFFT, les sorties correspondantes sont décrites sur la figure 3.20 pour la partie réelle et sur la figure.3.21 pour la partie imaginaire.

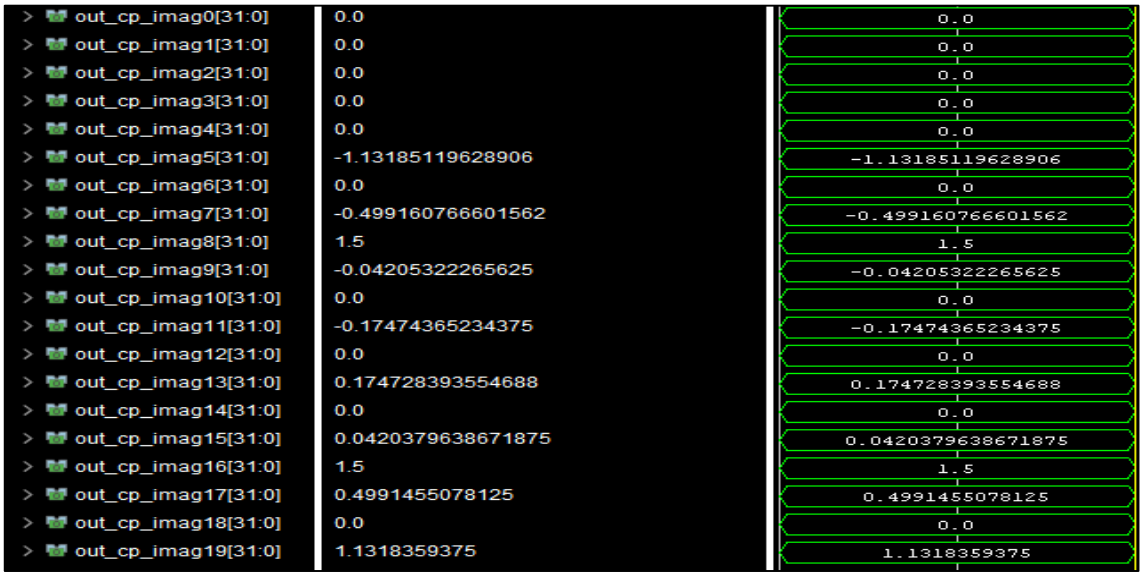


Figure3.20. Simulation de la partie réelle du module d'insertion du CP.

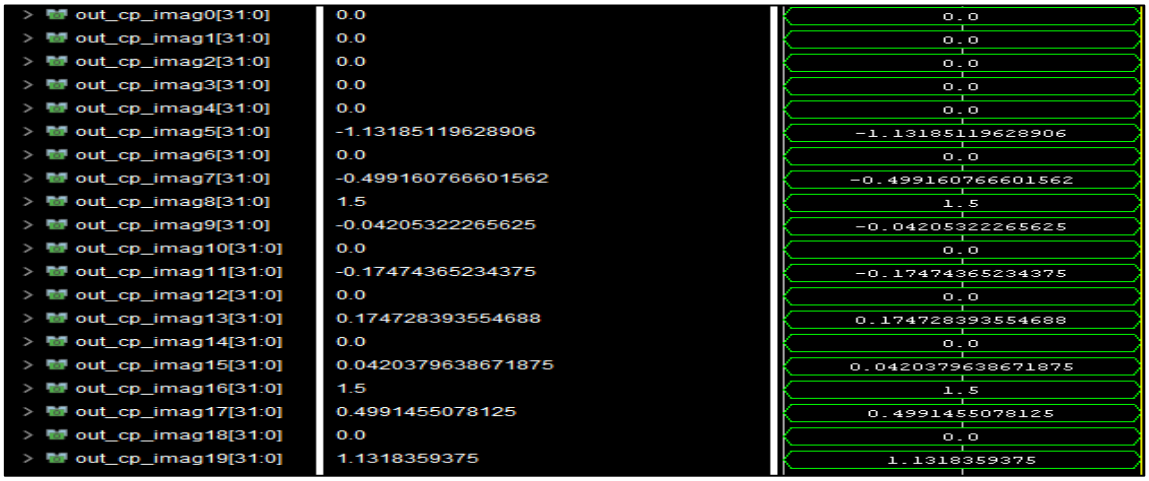


Figure3.21. Simulation de la partie imaginaire du module d'insertion du CP.

Le résumé de l'utilisation de FPGA et le rapport de consommation de la conception du module d'insertion du CP sont donnés dans le tableau 3.22. Ce module ne consomme aucune bascule FF, aucun bloc LTU, aucun bloc DSP ou BUFG.

Ressource	Estimation	Disponible	Utilisation
FF	0	106400	0,00%
LUT	0	53200	0,00%
BUFG	0	32	0,00%
DSP	0	220	0,00%

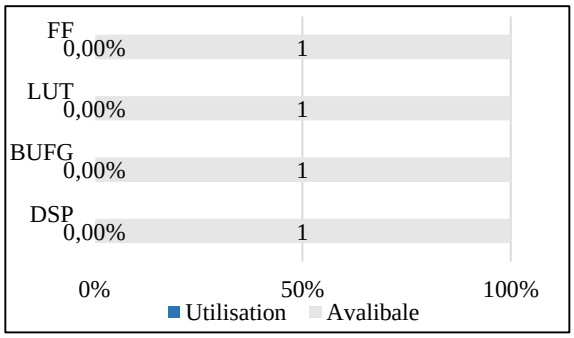


Figure3.22. Consommation du FPGA pour l'implémentation du module d'insertion du CP.

3.4.5. Module PISO

Dans cette partie, on va présenter la simulation du module PISO dont la progression est décrite dans la sous-section 3.1.3. Pour une séquence d'entrée de 16 éléments de 32 bits, la sortie du module PISO est illustrée par l'exemple des figures suivantes. L'entrée du module PISO est donnée sous forme de valeurs parallèles réelles et imaginaires. Les sorties de PISO après le traitement sont données dans la figure 3.23 pour la partie réelle et dans la figure 3.24 pour la partie imaginaire.

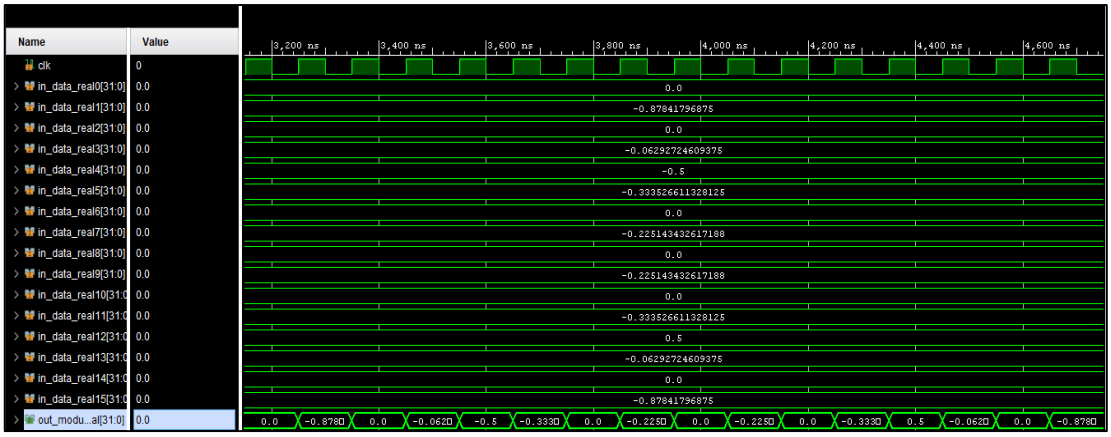


Figure3.23. Simulation de la partie réelle du module PISO.

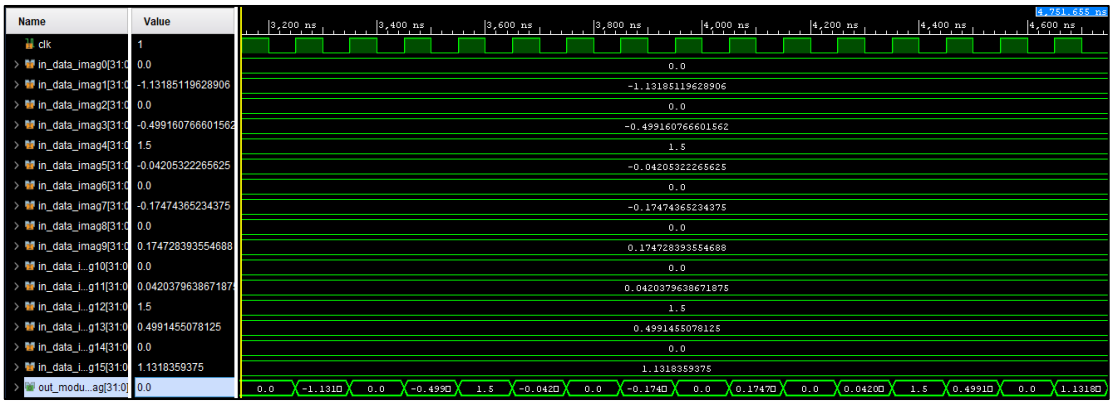


Figure3.24. Simulation de la partie imaginaire du module PISO.

Le rapport de consommation de FPGA de la conception du module PISO est mentionné dans le tableau suivant. D'après ce tableau, on note ce module utilise 1110 bascules Flip- Flop sur 106400 (1,04 %), 279 LUT sur 53200 (0.52 %), 0 BUFG sur 32 (0.00 %) et 0 DSP48E1 sur 220 (0.00%).

Ressource	Estimation	Disponible	Utilisation
FF	1110	106400	1,04%
LTU	279	53200	0,52%
BUFG	0	32	0.00%
DSP	0	220	0.00%

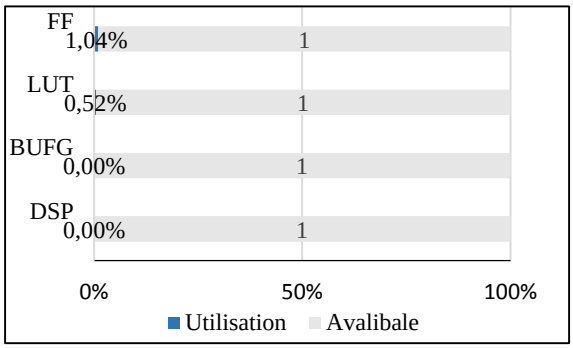


Figure3.25. Consommation du FPGA pour l'implémentation du module PISO.

3.4.6. Module d'élimination du CP

La simulation du module d'élimination du CP sera discutée dans cette sous-section. Pour une séquence d'entrée de 20 éléments de 32 bits, les sorties du ce module sont illustrées par l'exemple suivant. Ici, on supprime les quatre échantillons nuls réels et imaginaires qui ont été ajoutés au début du symbole OFDM simulé dans le module précédent. Les sorties de ce module correspondantes aux entrées données sont décrites dans la figure 3.26 pour la partie réelle et du partie imaginaire dans la figure 3.27.

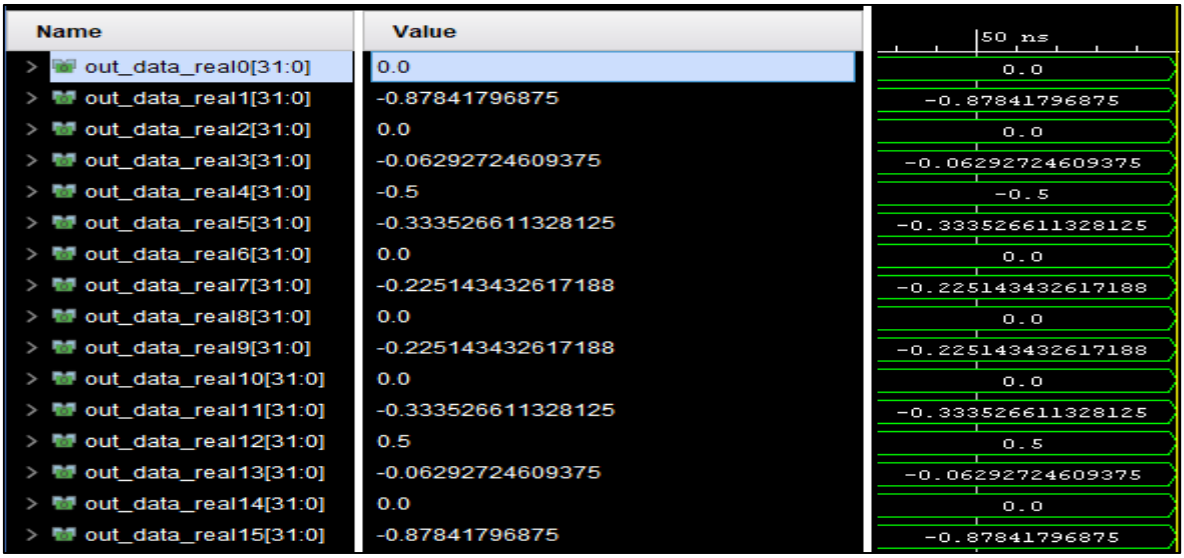


Figure3.26. Simulation de la partie réelle du module d'élimination du CP.

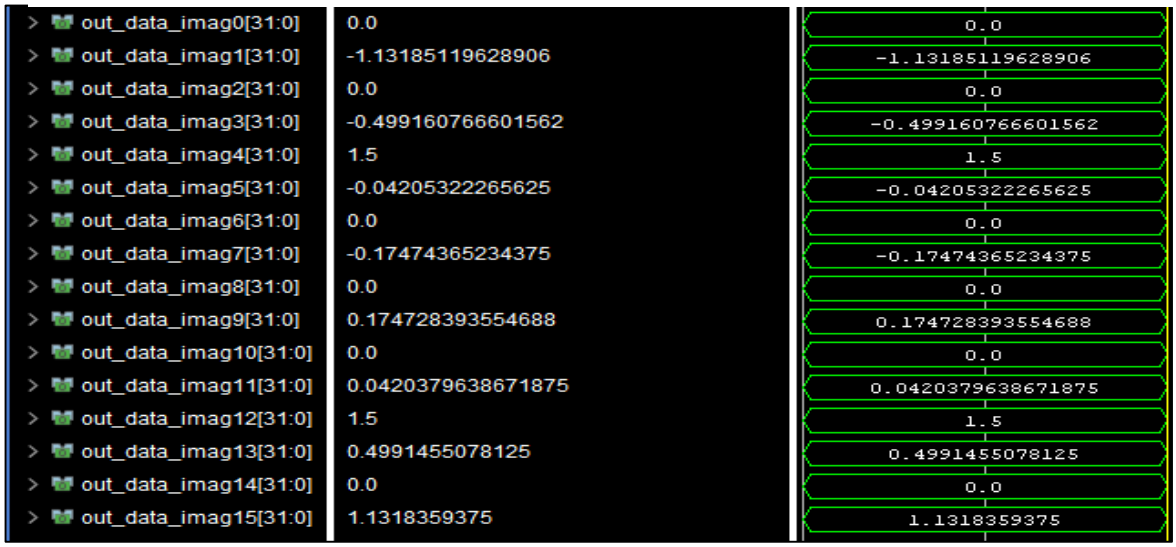


Figure3.27. Simulation de la partie imaginaire du module d'élimination du CP.

Le taux d'utilisation et de consommation du FPGA pour la conception du module d'élimination du CP est similaire à celui consommé par le module d'insertion du CP (voir la figure 3.22).

3.4.7. Module FFT

Après la simulation du module, on note qu'une séquence d'entrée de 16 éléments de 32 bits, les sorties du module FFT sont illustrées par l'exemple suivant. Chacune des valeurs réelles pré-modifiées par le module SIPO sont affectées aux entrées réelles du module FFT et de même

façon pour les valeurs imaginaires. Les sorties FFT correspondantes aux données d'entrée pour les parties réelles et imaginaires sont respectivement présentées sur les figures 3.28 et 3.29.

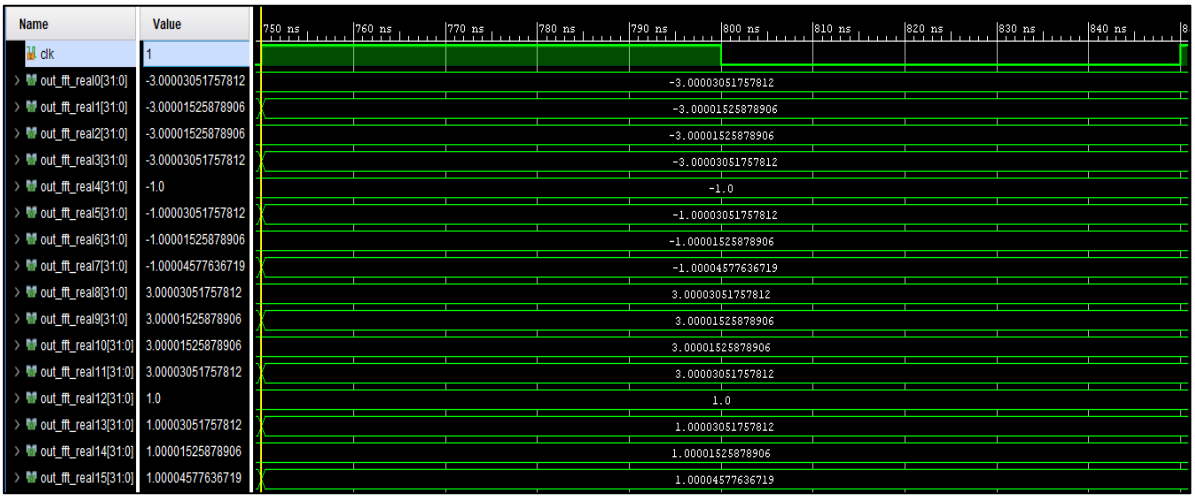


Figure3.28. Simulation de la partie réelle du module FFT.

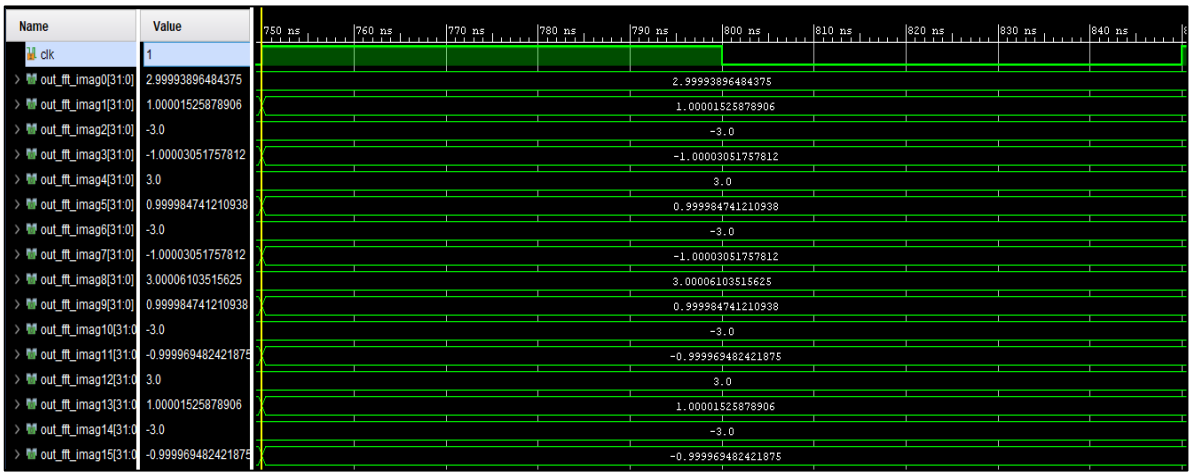


Figure3.29. Simulation de la partie imaginaire du module FFT.

Le taux d'utilisation et de consommation du FPGA pour la conception du module FFT est listé dans le tableau de la figure 3.30. Pour ce module, la consommation représente 6.38 % des bascules et 20.18 % des LUT.

Ressource	Estimation	Disponible	Utilisation
FF	6784	106400	6,38%
LUT	10735	53200	20,18%
BUFG	0	32	0,00%
DSP	0	220	0,00%

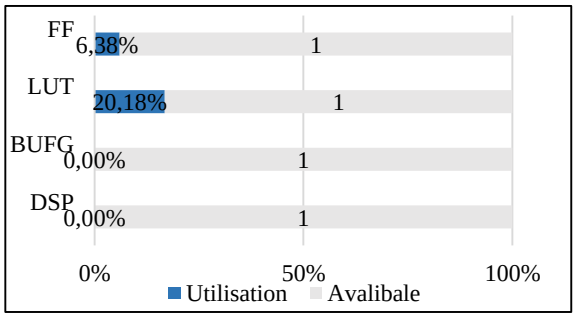


Figure3.30. Consommation du FPGA pour l'implémentation du module FFT.

3.4.8. Module de décodage QAM 16

Le dernier module simulé est celui de la démodulation QAM 16 dont le déroulement est décrit dans la sous-section 3.1.1. Alors, pour une séquence d'entrée de 16 éléments de 32 bits, les sorties du ce module sont données dans l'exemple suivant. Chacune des valeurs réelles et imaginaires pré-ajustées par le module PISO sont décodées selon un ensemble de seuils de décision. Les sorties du décodeur QAM16 correspondant à l'entrée spécifiée sont indiquées sur la figure 3.31.

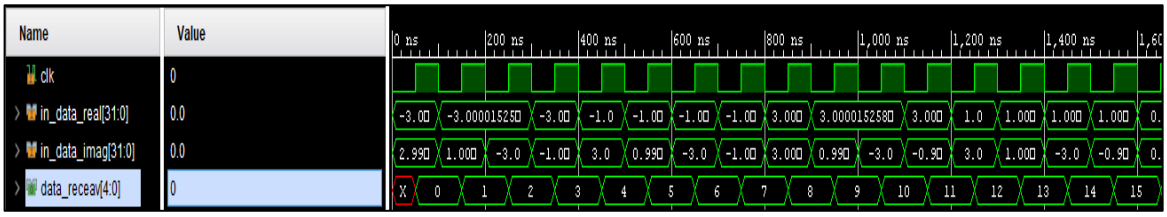


Figure3.31. Simulation du module de décodage QAM 16.

Le résumé de l'utilisation de FPGA et le rapport de consommation de décodeur QAM 16 sont fournis dans le tableau de la figure 3.32. Étant donné que ce module utilise 4 bascules Flip-Flop sur 106400 (0.01 %), 70 LUT sur 53200 (0.13 %), 0 BUFG sur 32 (0.00 %) et 0 DSP48E1 sur 220 (0.00%).

Ressource	Estimation	Disponible	Utilisation
FF	4	106400	0,01%
LUT	70	53200	0,13%
BUFG	0	32	0,00%
DSP	0	220	0,00%

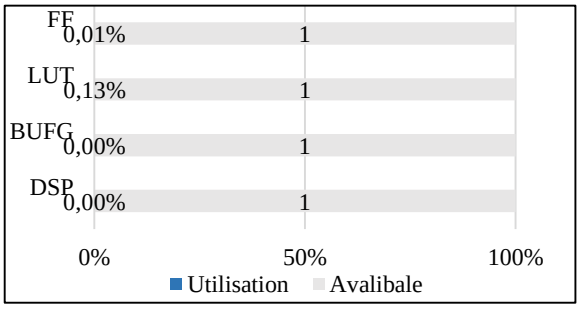


Figure3.32. Consommation du FPGA pour l'implémentation du module de décodage QAM16.

3.4.9. Module OFDM global

Enfin, après avoir simulé les différents blocs implémentés constituant le modulateur et le démodulateur OFDM, on peut maintenant collecter tous ces blocs pour construire un module global représentant l'émetteur et le récepteur afin de calculer le temps nécessaire à la tâche complète et de tester le bon fonctionnement de notre système implémenté de manière exhaustive.

La figure 3.33 montre la séquence des niveaux qu'on veut transmettre ou qui peut être prise comme entrée du module OFDM global.

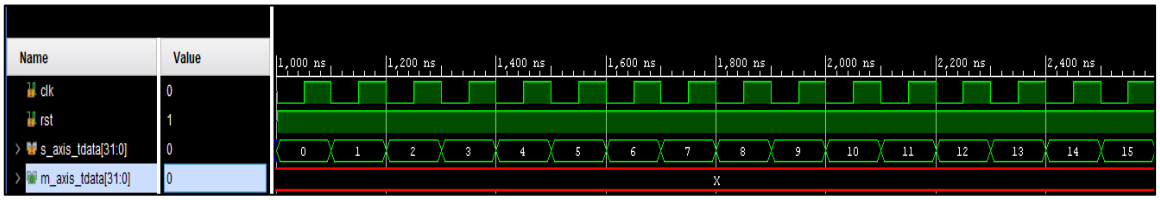


Figure3.33. L'entrée série du module globale OFDM.

Après un décalage de 62 cycles d'horloge, on remarque que la séquence de données reçue est identique à celle émet comme le montre la figure 3.34.

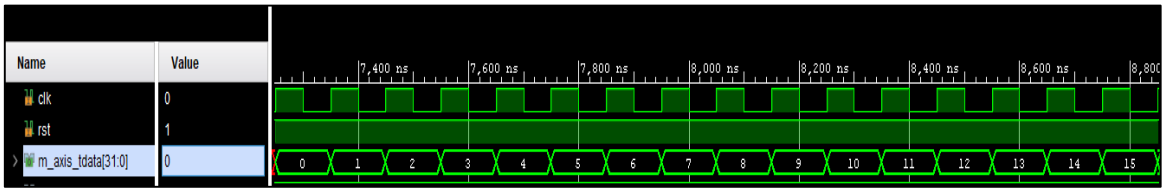


Figure3.34. La sortie série du module globale OFDM.

Finalement, la consommation totale de ce module dans l'FPGA et plus précisément dans le xc7z020clg400-1 est mentionnée dans le tableau de la figure 3.35. On remarque que ce module utilise en total 13172 bascules Flip-Flop sur 106400 (12,38%), 21158 LUT sur 53200 (39,77%), 0 BUFG sur 32(0.00 %) et 0 DSP48E1 sur 220 (0.00%).

Ressource	Estimation	Disponible	Utilisation
FF	13172	106400	12,38%
LUT	21158	53200	39,77%
BUFG	0	32	0,00%
DSP	0	220	0,00%

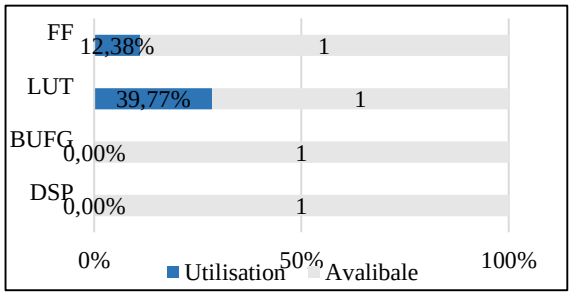


Figure3.35. Consommation du FPGA pour l'implémentation du module OFDM globale.

3.5. Design hardware et application software

Dans cette section, on va voir une autre utilisation du module OFDM global mise en œuvre dans la section précédente afin de valider ses performances, en implémentant une conception matérielle en créant un design hardware qui permet de basculer les données vers et depuis la partie processeur, puis en utilisant cette conception dans une application software.

3.5.1. Conception matérielle

La conception matérielle « design hardware » de notre application est basée sur trois blocs de base " zynq processing system, axi-dma et le bloc IP du module OFDM global. Comme l'on a évoqué précédemment dans le deuxième chapitre, l'interconnexion entre la partie du processeur (PS) et la partie logique (PL) est de type AXI4 Stream. Elle oblige donc à crier un bloc IP avec un interfaçage Stream et un bloc axi-dma car le processeur n'a pas d'interface Stream. Alors la création du design hardware se déroule en deux étapes :

3.5.1.1. Création du bloc IP pour le module d'OFDM global

La première étape concerne la création d'un bloc IP pour le module OFDM global afin d'utiliser ce dernier pour créer un bloc design. Tout d'abord, il est important de considérer ce

bloc comme "Stream interface IP". La figure 3.36 montre le bloc IP qu'on a créé et leurs ports d'interfaçage Stream.

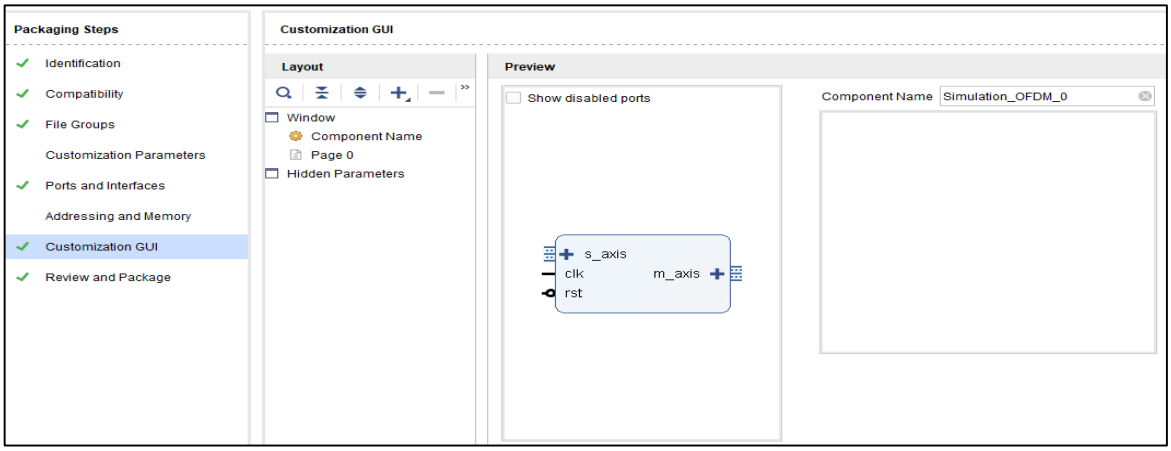


Figure3. 36. Création du bloc IP du module OFDM global.

3.5.1.2. Création du design hardware

La deuxième étape est axée sur la création du design hardware qui comporte trois blocs de base qui sont responsables au transférer les données comme l'illustre la figure 3.37.

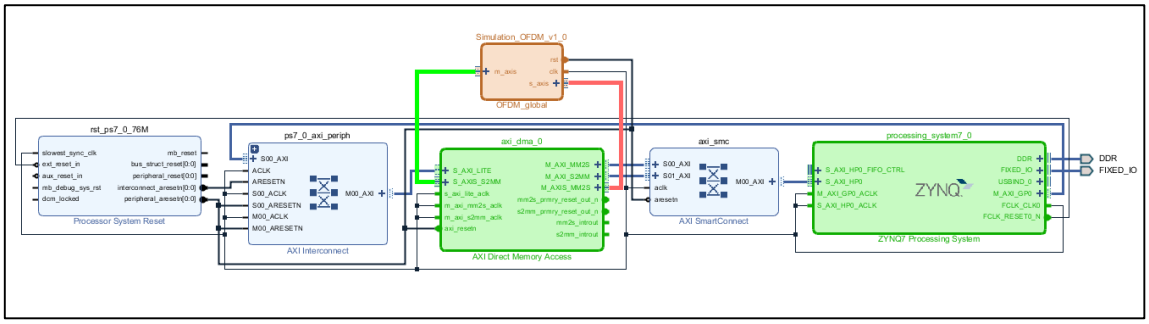


Figure3.37. Design Hardware de la modulation OFDM.

Le premier bloc est le bloc IP du module OFDM global, qui est encadré en marron au milieu de la figure 3.37. Ce bloc est responsable de calculer les algorithmes de modulation et démodulation OFDM et il est directement connecté au bloc situé juste en dessous "axi-dma" via les jonctions rouge et verte. Les deux blocs encadrés en vert sont alloués à l'échange des données. Le premier est l'axi-dma qui est le cerveau du design, donc il est responsable de pousser les données dans les deux sens (du processeur vers le bloc OFDM global et prend les données résultants du bloc OFDM global vers le processeur). Ce bloc a une signification très importante dans cette conception car il est impossible de connecter directement la partie processeur à la partie logique à cause d'utilisation du protocole AXI4-Stream.Tandis que le

dernier bloc est le système processeur "zynq processing system" avec les configurations suivantes : une horloge de 50 Mhz et une activation du bus HP (High Performance) pour l'utilisation de la mémoire DDR3.

Après la validation du design, un fichier de flux binaire (fichier Bitstream) a été créé pour être utilisé dans l'application software.

3.5.2. Application software

Dans cette application software, on a utilisé l'environnement de développement Jupyter dans le langage de programmation Python, les deux derniers travaillent spécifiquement avec le FPGA SOC. La donnée utilisée dans l'application est une image de type **grayscale** au format **bmp** avec des dimensions de **512×512**. Un fichier **.bmp** est un fichier bitmap, c'est-à-dire un fichier d'image graphique stockant les pixels sous forme de tableau de points. Alors que les images de types **grayscale** sont des images en 256 couleurs codées sur 8 bits pour chaque pixel. Notre bloc implémenté a une seule entrée série codée sur 4 bits (voir la partie de codage QAM16). Ainsi, l'image qu'on a utilisée doit être prétraitée par la conversion du tableau de deux dimensions en un tableau d'une seule dimension. En notant que chaque chaîne de données doit avoir 4 bits de pixel. Afin de bien comprendre cette application, nous illustrons un organigramme montrant toutes les étapes qu'elle a parcourues dans notre application.

3.5.2.1. Organigramme de programme implémenté pour l'application software

L'application software passe par des étapes de programmation essentielle pour transférer une image de type **grayscale**. Ces étapes sont résumées dans l'organigramme de la figure 3.38.

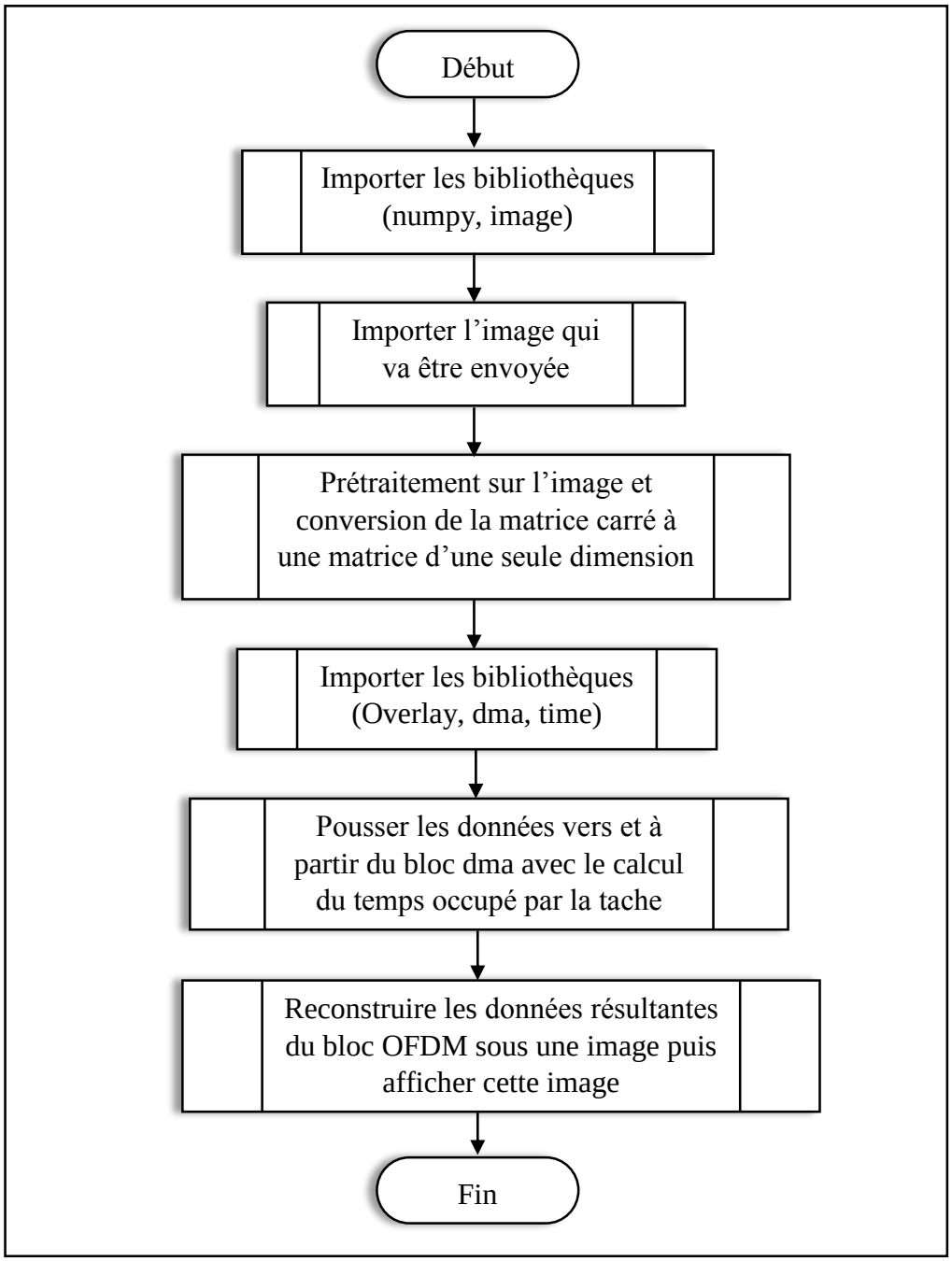


Figure3.38. Organigramme du programme principal pour l'application software.

D'après la figure 3.38, on remarque que l'organigramme du programme principal pour l'application software est composé de six parties qui sont brièvement décrites ci-dessous :

- **Importation des bibliothèques :** inclut les bibliothèques telles que la bibliothèque numpy qu'on l'utilisera essentiellement pour manipuler des tableaux et les matrices ainsi

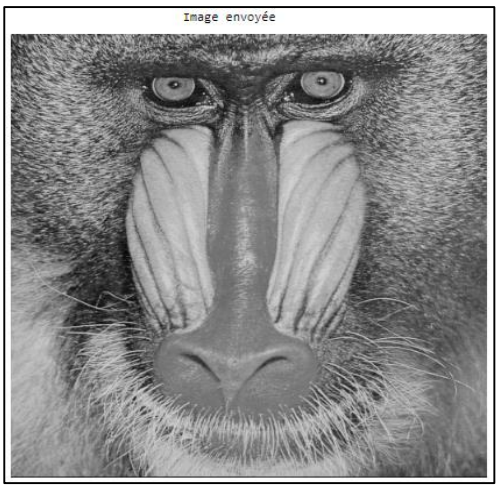
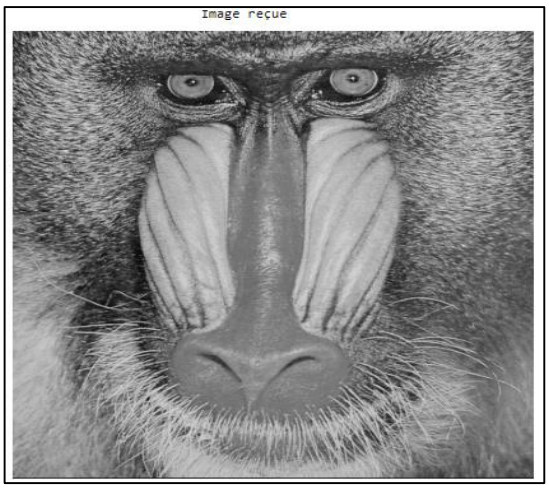
que la bibliothèque image qui est conçue pour offrir un accès rapide aux données contenues dans une image.

- **Importation de l'image :** permet d'importer l'image de test via la bibliothèque mentionnée ci-dessus.
- **Prétraitements sur l'image :** l'image de test est soumise à un certain nombre de traitements, notamment la conversion d'une matrice en tableau, la division de l'image de deux parties, une partie MSB et une partie LSB de 4 bits.
- **Bibliothèques (Overlay, dma, time) :** comprend les bibliothèques utilisées pour manipuler le design hardware, pousser les données vers et depuis le processeur et calculer le temps d'exécution.
- **Poussage des données :** permet aux données d'être poussées vers et depuis le bloc dma en calculant le temps occupé par la tâche.
- **Reconstruction de données:** englobe toutes les traitements inverses affectés pour reconstruire approximativement le pixel original.

3.5.2.2. Résultats d'application

Dans cette partie, on présente les différents résultats des étapes de programmation de notre application qui sont résumés dans le tableau 3.4.

Tableau3. 4.Résultats d'application avec comparaison.

L'image envoyée:	L'image reçue :
	

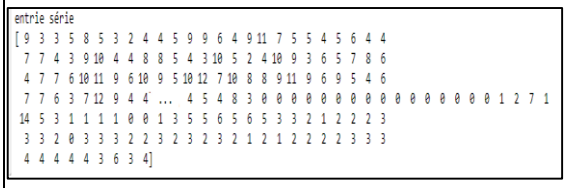
La matrice carrée de l'image envoyée :

```
squar matrix of input image:
[[145  55  48 ..., 112 153 177]
 [116 101  39 ..., 129 148 125]
 [ 76 114  46 ..., 100  88  81]
 ...,
 [140 145 140 ...,  82  89  78]
 [155 140 131 ...,  78  79  69]
 [ 10  11  14 ...,   6   3   4]]
```

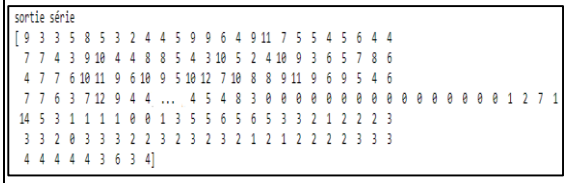
La matrice carrée de l'image reçue :

```
squar matrix of receav image
[[145  55  48 ..., 112 153 177]
 [116 101  39 ..., 129 148 125]
 [ 76 114  46 ..., 100  88  81]
 ...,
 [140 145 140 ...,  82  89  78]
 [155 140 131 ...,  78  79  69]
 [ 10  11  14 ...,   6   3   4]]
```

L'entrée série du modulateur OFDM :



La sortie série du démodulateur OFDM :



Le temps d'exécution hardware :

```
Temps d exécution Hardware  0.0013468265533447266
```

D'après les résultats présentés dans le tableau comparatif ci-dessus, on constate que les résultats de la sortie reçue sont identiques à ceux de l'entrée transmise après suppression du décalage provoqué par le bloc OFDM (78 cycles d'horloge).

3.6. Application sur les blocs d'implémentation sous System Generator

Comme on a vu dans la section précédente, on ne peut pas séparer les blocs modulateur et démodulateur en raison des retards causés par le FPGA.

Afin de tester les performances notre système implémenté, on a proposé une application compatible avec l'utilisation du System Generator. Ce qui permet de simuler le canal de transmission réel vu du manque des modules de transmission réel **"AD9361"** en raison de leurs prix élevés (enivrent de 18 000,00 DA pour l'unité). System Generator permet à Vivado et Matlab de se communiquer entre eux. Cette application consiste donc à importer des blocs réalisés sous Vivado (codage _QAM16, IFFT, SIPO,…etc.) pour les utiliser dans des modèles Simulink. L'utilisation des bibliothèques de Simulink permet de simuler un canal de transmission et même d'observer l'effet de ce canal sur le système implémenté.

Le type de données d'entrée de test utilisé dans cette dernière application est exactement identique à celui de la précédente. Le chemin de données suivi dans l'application pour transmettre l'image est indiqué sur la figure 3.49.

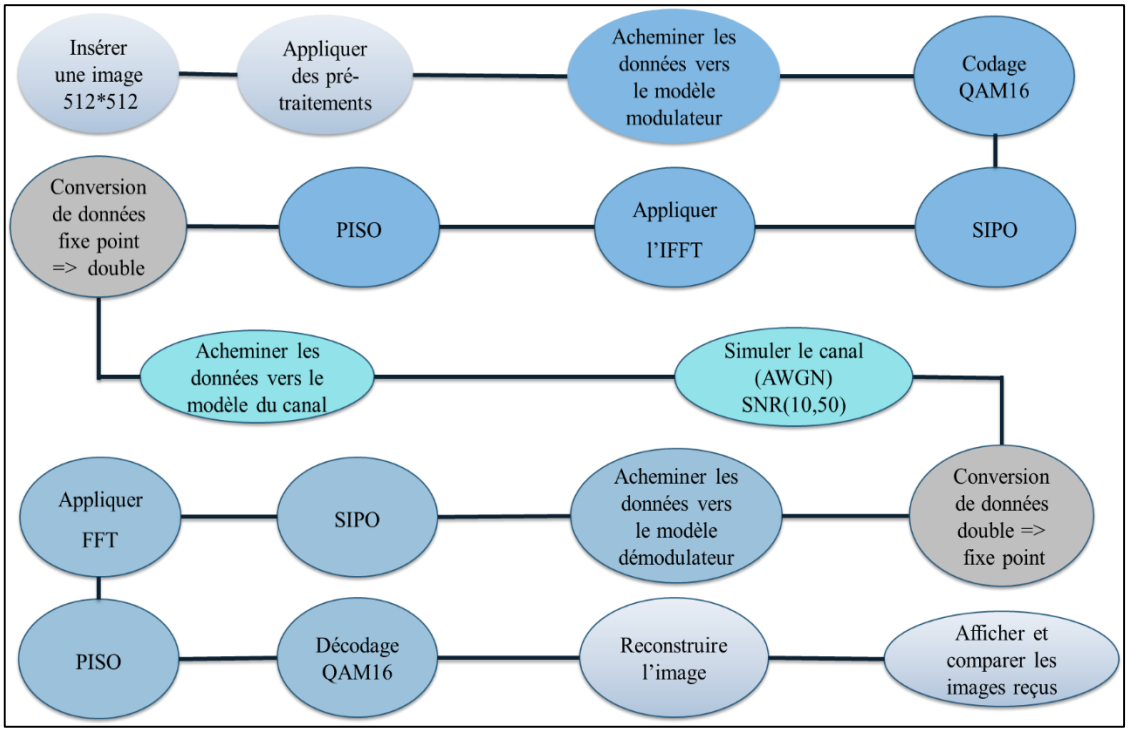


Figure3.39. Schéma global d'application sous System Generator.

À partir du schéma illustré sur la figure 3.39, on peut déduire trois parties principales de cette application notamment : la modulation OFDM, le canal de transmission (AWGN) et la démodulation OFDM. De plus, on a également deux autres parties intermédiaires pour la conversion des données. Les étapes restantes sont des traitements avant la modulation pour extraire les données série d'une image **grayscale** et après démodulation afin de reconstituer les données série résultantes en une image.

3.6.1. Etapes de l'application et résultats de simulation

Afin de bien expliquer les méthodes et les configurations évoquées dans cette dernière application, on abordera tout d'abord brièvement les huit (08) étapes constituant l'application, ensuite on discutera les résultats issus des différentes simulations :

a. Insertion d'image d'entrée

L'image utilisée pour la transmission est la même que celle utilisée dans l'application software, elle est donc de type **grayscale**, avec des dimensions de 512 x 512 sous l'extension **bmp** mentionnée dans la figure 3.40.

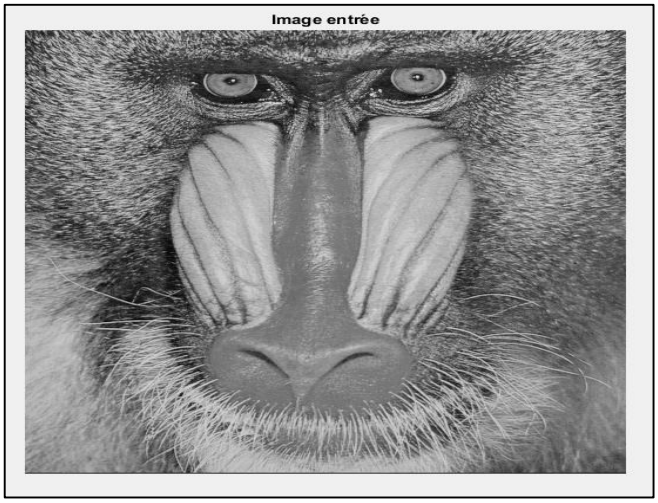


Figure3.40. L'image d'entrée pour la transmission.

.

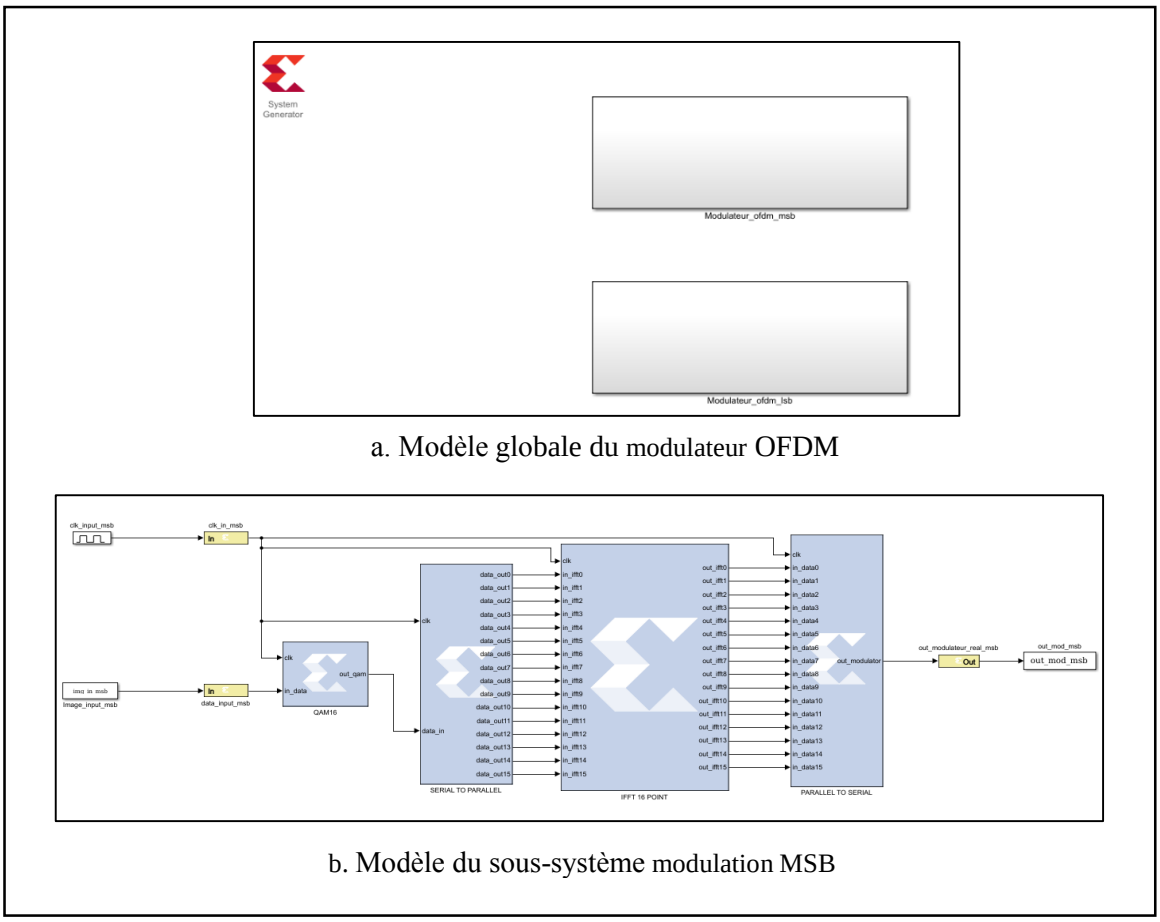
b. Prétraitements

Comme on a déjà mentionné dans l'application software, l'image de type **grayscale** a des pixels de 256 couleurs, donc le codage est effectué sur 8 bits, cela exige la décomposition de chaque pixel en deux parties de 4bits **MSB** et 4bits **LSB**. Après avoir eu des données d'entrées de 4 bits, il est nécessaire de reformuler la matrice d'image observée à partir du fichier **bmp** en un vecteur unidimensionnel pour entrer les données vectorielles de manière séquentielle, et enfin diriger les résultats du prétraitement vers le modulateur OFDM.

c. Modèle Simulink de la modulation OFDM

La bibliothèque Xilinx ajoutée au Simulink contient un bloc **"Blck Box"** qui est utilisé pour importer des fichiers avec une extension .v (hdl). Le modèle de modulateur OFDM se compose principalement de quatre blocs de modulation OFDM (codage _QAM16, SIPO, IFFT_16point, PISO) implémentés sous Vivado. On a également un générateur d'impulsion (période de 2 secondes) et des blocs qui acheminent les résultats vers et depuis l'espace de travail MATLAB.

Le modèle de la modulation OFDM global montré sur la figure 3.41.a contient deux sous-systèmes identiques MSB et LSB ayant chacun 4 bits de pixel. La figure 3.41.b représente un sous-système parmi ces deux sous-systèmes (modulateur OFDM MSB).



a. Modèle globale du modulateur OFDM

b. Modèle du sous-système modulation MSB

Figure3.41. Modèle Simulink (System Generator) de la modulation OFDM.

d. Première conversion des données

Avant de simuler le canal de transmission, on doit convertir les données de modulateur OFDM résultantes. La première étape consiste à convertir les données à virgule Fix $\boldsymbol{Q_{16,16}}$ en une présentation à virgule flottante de double précision. Cette conversion est très importante car le bloc AWGN utilisé dans le canal est piloté à l'aide d'une présentation à virgule flottante.

e. Canal de Transmission

Pour tester les performances du système d'OFDM implémenté, on a ajouté un canal de transmission caractérisé par un bruit Gaussien blanc additif (Additive White Gaussian Noise

AWGN). Le bruit gaussien blanc additif (AWGN) est un modèle de bruit simple représentant le mouvement des électrons à l'extrémité du récepteur RF.

Une fois les données converties en présentation à virgule flottante, elles sont transmises au deuxième modèle Simulink du canal AWGN. Dans le modèle Simulink illustré sur la figure 3.42, on a utilisé deux canaux AWGN puisque on a deux sorties du modulateur OFDM (MSB et LSB).

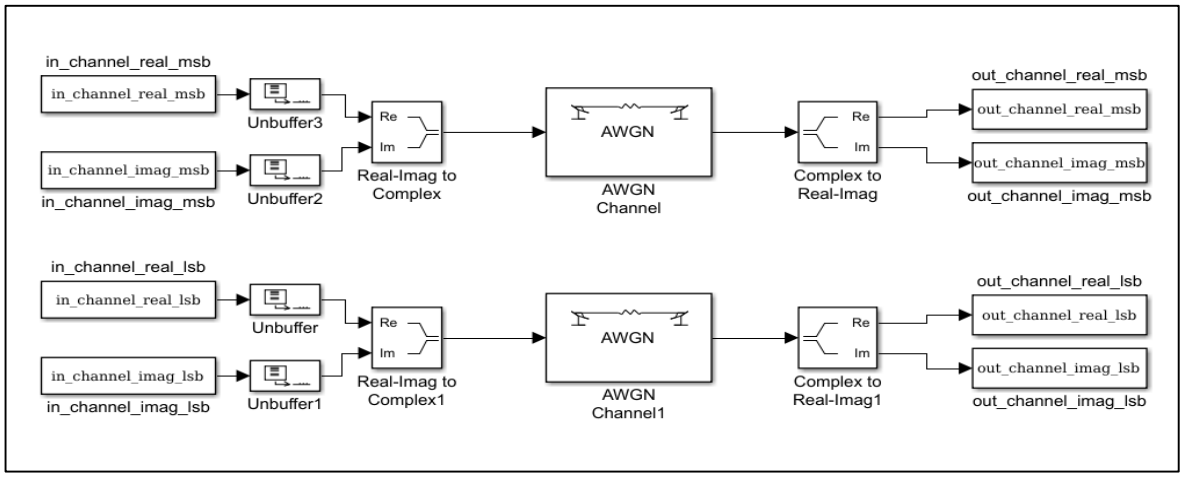


Figure3.42. Modèle Simulink du canal AWGN.

Ce bruit additif provoque des erreurs au niveau du décodage lors de la réception, en comparant les données binaires reçues avec les données binaires d'origine. Le taux d'erreur binaire (TEB) peut être calculé en fonction du rapport signal sur bruit (Signal to Noise Ratio SNR) en décibels (dB).

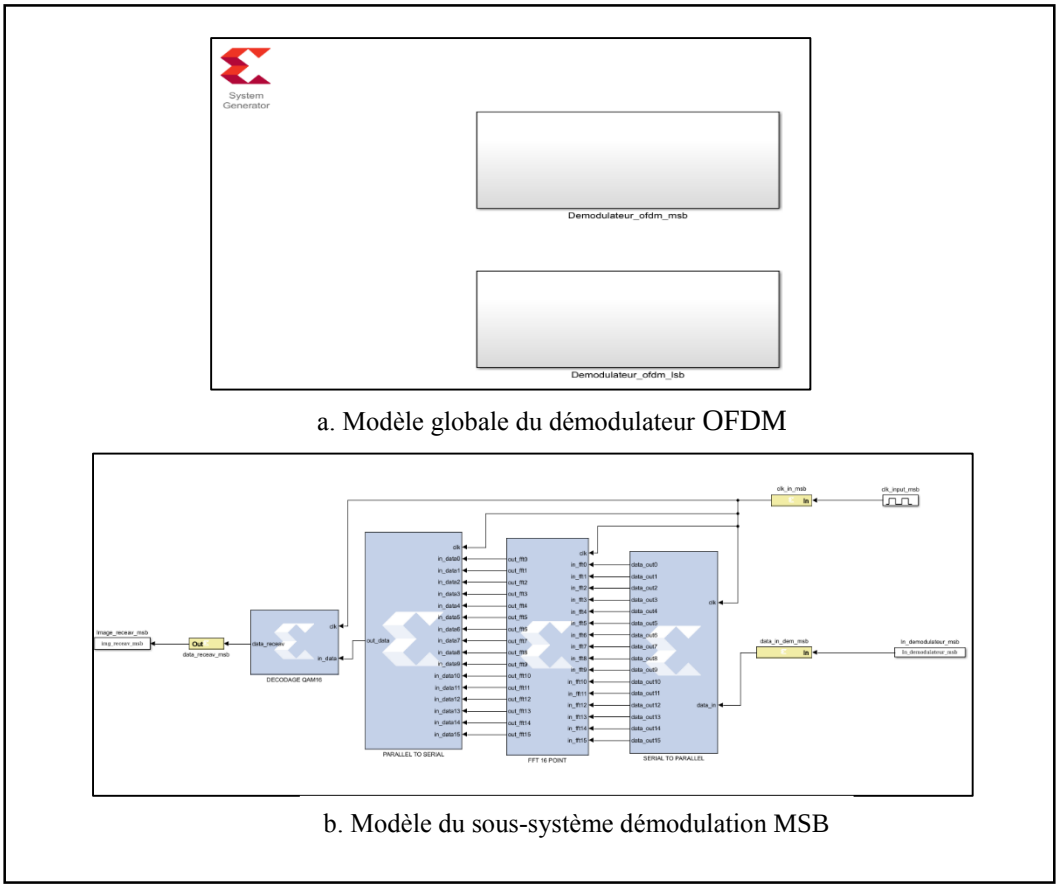
f. Deuxième conversion des données

Pour acheminer les données hors du canal vers le bloc démodulateur OFDM, les données doivent être converties à nouveau en une présentation à virgule fixe $\boldsymbol{Q_{16,16}}$. Ce processus est l'opération inverse de la première conversion.

g. Modèle Simulink de la démodulation OFDM

Le modèle de la démodulation est également comprend principalement quatre blocs y compris ; SIPO, FFT, PISO, DEC_QAM16). Ce modèle est similaire au modèle de modulation mais de manière inverse, c'est-à-dire qu'il contient les algorithmes inverses par rapport au modèle de modulation OFDM. La période d'horloge de 2 secondes est encore maintenue pour assurer la synchronisation entre les différents blocs.

Le modèle de la démodulation OFDM global est représenté dans la figure 3.43.a. Ce modèle comprend également deux sous-systèmes identiques MSB et LSB ayant chacun 4 bits de pixel. La figure 3.43.b montre un sous-système (celui de MSB) parmi ces deux sous-systèmes.



a. Modèle globale du démodulateur OFDM

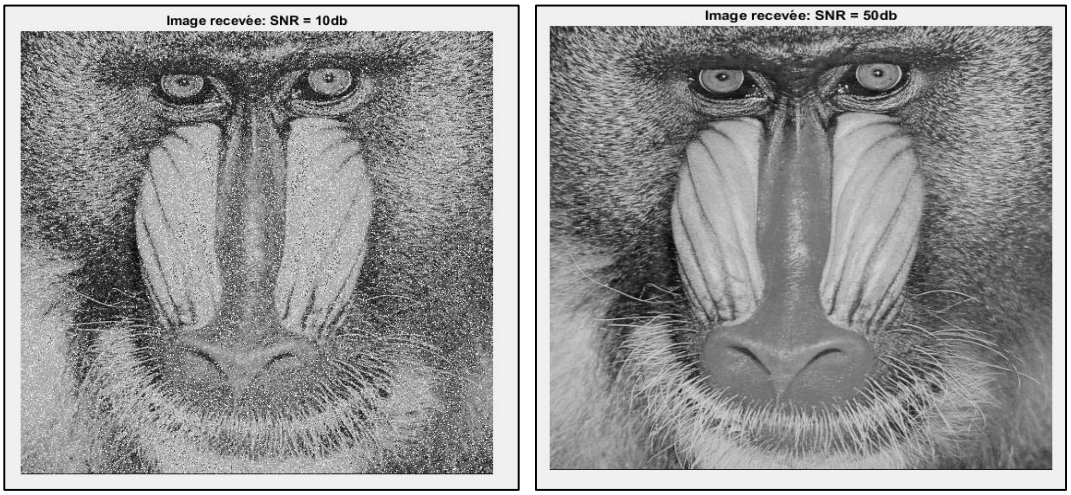
b. Modèle du sous-système démodulation MSB

Figure3.43. Modèle Simulink de la démodulation OFDM.

h. Reconstruction des données reçues et comparaison des résultats

Dans cette dernière partie de notre application, on a reçu les résultats de démodulation sous forme de vecteur unidimensionnel. Pour visualiser l'image résultante, on regroupe d'abord les pixels sous forme de 8 bits, c'est-à-dire qu'on combine 4 bits MSB et LSB, puis on reformule le vecteur dans une matrice carrée. Enfin, on montre l'image et on compare les différents résultats en fonction du critère TEB et SNR.

Pour visualiser clairement l'effet du canal de transmission AWGN sur l'image de test, on illustre sur la figure 3.44 deux images résultantes pour deux valeurs SNR différentes (partie (a) pour SNR = 10 dB et partie (b) pour SNR = 50 dB).



a. Image résultante avec SNR = 10dB b. Image résultante avec SNR = 50dB

Figure3.44. Comparaison des images résultant sou l'effet du canal AWGN.

Le rapport signal sur bruit (SNR ou S/N) est un indicateur de qualité du canal de transmission. Il est défini comme le rapport des puissances entre la partie du signal qui représente l'information et la partie du bruit. Le bruit est alors le rapport du signal sur le SNR. La variation du bruit se change relativement inverse avec la variation du SNR dont l'augmentation de la valeur SNR à mener une diminution de bruit sur les informations reçues. Les images a et b présentées dans la figure 3.44 montrent clairement l'effet du canal AWGN sur les données transmises selon la relation inverse obtenus précédemment (image a : $SNR \downarrow \equiv bruit \uparrow$, image b : $\uparrow \equiv bruit \downarrow$).

Enfin, on calcule et en trace le taux d'erreur binaire (TEB) en fonction de la variation du rapport signal sur bruit (SNR). Les courbes du TEB en fonction du SNR résultantes implémentée et celle calculée de la théorie sont illustrées dans la figure suivante.

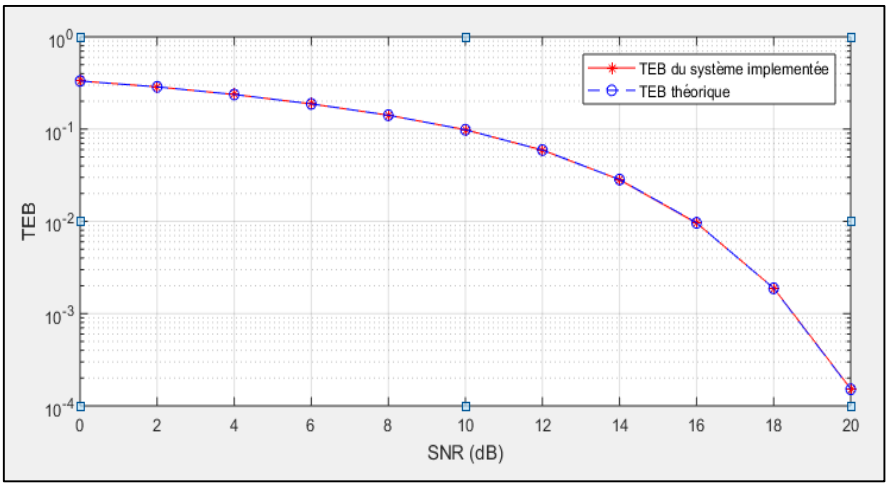


Figure3.45. TEB en fonction du SNR pour codage QAM16 et modulation d'IFFT à 16 point modulation OFDM.

D'après les courbes de cette figure, on constate que les résultats de notre implémentation sont identiques à ceux de la théorie. Ces courbes confirment explicitement à nouveau la relation entre le taux d'erreur binaire (BER) et le rapport signal sur bruit (SNR).

3.7. Comparaison des résultats issus de notre implémentation avec ceux rapportés dans la littérature

Dans cette section, on propose une étude comparative des résultats de notre application OFDM deux autres travaux réalisés dans les références [13] et [14]. Cette comparaison est résumée dans le tableau 3.5 et elle est effectuée en terme de plusieurs paramètres et caractéristiques notamment ; technique multi-porteuse utilisée, architecture, type de codage, type de canal, type de carte FPGA et application software.

Tableau3. 5. Comparaison des différentes applications et implémentations.

Architecture du système	**OFDM techniques**	**Architecture et application sou logiciel**	**Système implémentée**	**Nombre de point FFT**	**Codage**	**Canal**	**Application Software**	**FPGA**
Architecture 2012[13]	OFDM, OFDM CP	System Generator	Non	512	QPSK	AWGN, Multi-Trajet	Non	Virtex5
Architecture 2018 [14]	FBMC	Xilinx ISE	Oui	8	OQAM	Non	Non	Spartan3, Virtex5, Artix7

Notre Architecture	OFDM, OFDM CP	Vivado, System Generator, Jupyter	Oui	16	QAM16	AWGN	Oui	Pynq-Z2

Selon le tableau précèdent, on a considérablement apporté une amélioration dans la mise en œuvre du système OFDM et OFDM-CP sous la carte de développement FPGA. Chaque travail a adopté un type de FPGA spécifique en fonction de l'application souhaitée. Pour notre cas, on a choisi FPGA Pynq-Z2 car elle met en évidence l'application software qui distingue notre système des autres travaux réalisés dans la littérature. On a implémenté cette technique via trois logiciel vivado, System Generator et Jupyter nootbook par rapport aux autres travaux qui ont utilisés un seul logiciel. On a également augmenté le débit de transmission en utilisant une modulation QAM de 16 niveaux par rapport aux modulations utilisés dans les deux travaux (QPSK et OQAM). Malheureusement, le système a été testé seulement sur un canal de transmission AWGN.

3.8. Conclusion

Au cours de ce chapitre, on a décrit les trois sections principales constituantes la mise en œuvre du système OFDM sur la carte FPGA. Tout d'abord, la première section a été consacrée à la simulation des différents systèmes de transmission OFDM ainsi que la version amélioré OFDM avec préfixe cyclique CP dans le cadre de simulation "vivado ". Ensuite, la section suivante a été réservée à l'application software proposée afin d'analyse les performances du système OFDM implémenté dans l'outil de développement "Jupyter Notbook ". La troisième section contient l'évaluation de notre système implémenté en présence du canal de transmission AWGN via le "system generator ". En comparant son effet sur l'image puis en traçant la courbe du TEB en fonction SNR. Finalement, une étude comparative de notre système avec ceux qui existent dans la littérature, a été abordée.

Pour conclure, on souligne que la partie implémentation de la modulation multiporteuse OFDM sur la plate-forme FPGA PynqZ2 est la partie la plus importante. Ainsi, le cahier de charge a été respecté avec succès.

Conclusion générale

Conclusion générale

L’intérêt majeur de ce projet est de mettre en évidence les divers aspects de la couche physique d’un futur système de communication OFDM sur une carte FPGA du fabricant Xilinx, où nous nous sommes mis dans une réelle situation d’implémentation pratique ce qui nous a permis de toucher plusieurs domaines dont le domaine de l’électronique et de la télécommunication.

Dans le cadre de ce projet nous avons étudié et analysé deux systèmes notamment OFDM classique et améliorée OFDM-CP. L'idée principale de la modélisation de deux systèmes différents était de choisir le système le plus robuste à l'évanouissement sélectif en fréquence. Préalablement à cela, nous nous sommes attelé à présenter un état de l’art du domaine de la modulation OFDM ainsi que celui des techniques d’élimination des interférences entre les symboles (ISI).

Parallèlement à l’analyse des deux systèmes sus citées, les modèles de l'OFDM classique et améliorée OFDM-CP ont été élaborés et implantés numériquement grâce aux algorithmes IFFT et FFT rapides en utilisant le code HDL et synthétisés dans Vivado Xilinx. Chaque module constituant l’émetteur et le récepteur a été testé à l'aide du logiciel Vivado et de l'outil de développement Jupyter Notebook. Cela permet de s'assurer que les modules conçus fonctionnaient efficacement lorsqu’ils étaient exécutés dans la plateforme matérielle FPGA.

Malheureusement, la transmission réelle n’est pas possible grâce aux coûts élevés des cartes Analog Device qui permettent d’effectuer ce type de transmission. Nous avons donc rectifié la situation et pris d'autres moyens similaires à la transmission réelle en validant l'effet du canal de transmission AWGN sur le système OFDM implémenté. La validation du système a été effectuée par une simulation matérielle via Xilinx System Generator.

En perspectives, nous estimons que ce travail a été d’une grande utilité pour nous et qu’il sera bénéfique pour les chercheurs. On espère que cette étude soutiendra une éventuelle amélioration telle l’optimisation des codes HDL en réduisant la consommation du FPGA ainsi que la mise en œuvre d'une transmission réelle via les modules Analog Device. Nous suggérons également d’implémenter les modulations avancées tels que les modulations OFDM filtrées connus sous son acronyme anglais FBMC (Filtre Bank Multicarrier) pour combattre les interférences entre les sous porteuse dues au l’effet doppler sur la carte FPGA PYNQ- Z2.

Bibliographie

[1] Liu H., Li G. "OFDM-Based Broadband Wireless Networks-Wiley-Interscience". 1st edition (November 4, 2005).

[2] Khaled ROUABAH, cours " Systeme de Comunication Numerique ", Université de Bordj Bou Arréridj département d'electronique, 2019.

[3] BOUMEDIENE Fatima Zohra, " Limites des Performances de la Technique Multi-porteuse « OFDM » dans les systèmes Radio-mobiles ", thèse de doctorat, Université Djillali Liabes de Sidi- Bel-Abbes, 2019.

[4] H. Schulze and C. Lüders, "Theory and applications of OFDM and CDMA: Wideband wireless communications", John Wiley & Sons, 2005.

[5] GRUYER Pierre, PAILLARD Simon,"Modelisation d'un modulateur et demodulateur OFDM ". 15 décembre 2005.

[6] Ahmad R. S. Bahai, Burton R. Saltzberg, Mustafa Ergen "Multi-Carrier Digital Comm OFDM-Springer " (2004).

[7] T. T. Ha, "Theory and design of digital communication systems", Cambridge University Press, 2010.

[8] Louise H. Crockett, David Northcote, Craig Ramsay, Fraser D. Robinson, Robert W. "Stewart Exploring Zynq® MPSoC With PYNQ and Machine Learning Applications", University of Strathclyde Glasgow, Scotland, UK. April 20190

[9] "Zynq-7000 All Programmable SoC Technical Reference Manual UG585 (v1.7) " February 11, 2014.

[10]" PYNQ-Z2 Reference Manual v1.1", guide de l'utilisateur. 28 Oct 2019.

[11]" Vivado Design Suite Design Flows Overview UG892 ", guide de l'utilisateur. (v2018.2) Juin 6, 2018 UG892 (v2019.1) Mai 22, 2019.

[12]"System Generator for DSP User Guide UG640 ", guide de l'utilisateur. October 16, 2012.

[13] Ahmed Almajdoob, "Design and FPGA Implementation of an OFDM System Based on 3GPP LTE Standard over Multipath Fading Channel", thèse de Master, University Montréal, Québec, Canada, November 2012.

[14] Ramesha M, "FPGA implementation of FBMC transceiver for 5G technologies", thèse de doctorat, university GITAM, January 2018.

Abstract

With recent developments in digital wireless communications, the need for high data transmission speed has arisen, the promising solution is the orthogonal frequency division multiplexing modulation OFDM which offers high spectral efficiency and optimum robustness to interference between symbols (ISI) versus multipath channels.

The objective of this thesis is to implement a communication system based on the OFDM technique by using an FPGA board which offers a high performance and low cost solution for the implementation of digital communication systems. Our approach differs from existed researches that use a single method of implementation, however the present work adopt three different methods for the hardware implementation.

Résumé

Avec les développements récents des communications numériques sans fil, le besoin d'une vitesse de transmission de données élevée est apparu, la prometteuse solution est la modulation de multiplexage par répartition en fréquence orthogonale OFDM qui offre une efficacité spectrale élevée et une robustesse optimale aux interférences entre symboles (ISI) vis-à-vis les canaux à trajets multiples.

L'objectif de ce mémoire est d'implémenter un système de communication basé sur la technique OFDM sur une carte FPGA qui offre une solution de hautes performances et à faible coût pour la mise en œuvre des systèmes de communication numérique. Notre approche se distingue des autres recherches existantes qui ont utilisées seule méthode d'implémentation. En revanche, ce travail adopte trois méthodes différentes pour l'implémentation hardware.

ملخص:

مع التطورات الأخيرة في الاتصالات اللاسلكية الرقمية، ظهرت الحاجة إلى سرعة نقل بيانات عالية التدفق، والحل الواعد هو تعديل الإرسال المتعدد بتقسيم التردد المتعامد OFDM الذي يوفر كفاءة طيفية عالية ومتانة مثالية للتداخل بين الرموز (ISI) مقابل القنوات متعددة المسارات.

الهدف من هذه الأطروحة هو تنفيذ نظام اتصالات يعتمد على تقنية OFDM على لوحة FPGA والتي تقدم أداءً عاليًا وحلاً منخفض التكلفة لتنفيذ والتحقق من أنظمة الاتصالات الرقمية. تختلف هذه المقاربة عن الأبحاث الأخرى المتاحة التي تستعمل طريقة واحدة للتنفيذ مقارنة بمشروعنا الذي استعمل ثلاث طرق مختلفة.