Table des matières

Liste des tableaux

Liste des figures

Liste des symboles

ASIC: Application Specific Integrated Circuit

ASK: Amplitude Shift Keying

AWGN: Additive white Gaussian noise

BB: bande de base (Base Band)

BF: Basse fréquence

BPSK: Binary Phase Shift Keying

CAN: Convertisseur Analogique- Numérique (ADC : Analog to Digital Converter)

CNA: Convertisseur Numérique-Analogique (DAC: Digital to Analog Converter)

DC: Courant direct ou composante continue (Direct Current)

DDC: Convertisseur-abaisseur de fréquence (Digital Down Converter)

DDS: synthétiseur numérique directe (Direct Digital Synthesizer)

DI: Donnée d'entrée (Data Input)

DO: Donnée de sortie (Data Output)

DSSS: Spectre étalé à séquence directe (Direct Sequence Spread Spectrum)

DUC: Convertisseur-élévateur de fréquence (Digital Up Converter)

E/R: Emetteur / Récepteur

SFD: start frame delimiter

SHR: Sinchronization Header

RZ: retour zéro

NRZ: non-retour zéro

TX: transmetteur

RX: récepteur

PHR: PHY Header

FDX: Full-Duplex

FFT: Transformée de Fourier rapide (Fast Fourier Transform)

FI: Fréquence intermédiaire (IF: Intermediate Frequency)

FIFO: First In First Out

FIR: Filtre à réponse impulsionnelle finie (Finite Impulse Response Filter)

FPGA: Field Programmable Gate Array

FSK: Frequency Shift Keying

FSM: Machine à état fini (Finite State Machine)

GPP: General Purpose Processor

HDX: Half-Duplex

IEEE: Institute of Electrical and Electronics Engineers

IFFT: Transformée de Fourier rapide inverse (Inverse Fast Fourier Transform)

IP: Propriété intellectuelle (Intellectual Property (Design), Internet Protocol (Network))

E/S: entrée / sortie (I/O: Input / Output)

FPB: Filtre passe bas (LPF: Low-Pass Filter)

I/Q: In phase and Quadrature phase

ISE: Integrated Software Environment

ISM: Industriel, Scientifique and Médical

ISO: International Organization for Standardization

JTAG: Joint Test Action Group

LPT: Line Print Terminal

NCO: Oscillateur controllé numériquement (numerical controlled oscillator)

NI: National Instruments

OLN: Oscillateur Local Numérique

OOK: On Off Keying O-QPSK: Offset Quadrature Phase-shift keying

PC: Ordinateur (Personal Computer)

PLL: Boucle à verrouillage de phase (Phase Locked Loop)

PPDU: PHY protocol data units

PSK: Phase Shift Keying QPSK: Quadrature Phase Shift Keying

QAM: Quadrature Amplitude Modulation

RAM: Mémoire à accès aléatoire (Random Access Memory)

RF: Radio Fréquence

RSB : Rapport signal sur bruit (SNR: Signal-to-Noise Ratio)

SHF: Supra-haute fréquence UHF: Ultra Haute Fréquence

TSN: Traitement du signal numérique (DSP: Digital Signal Processing or DSP Processor)

USB: Universal Serial Bus

VCO: Oscillateur contrôlé par tension (Voltage Controlled Oscillator)

VHDL: Vhsic (very-high-speed integrated circuits) Hardware Description Language

WSN: Réseau de capteurs sans fil (Wireless Sensor Network)

Chapitre 1 - Introduction

1.1 Les réseaux de capteurs sans fils

Dans un premier temps, nous allons parler sur la technologie des réseaux de capteurs, qui est par définition une collecte de petits dispositifs de détection à faible coût avec calcul, stockage et communication.

Ils possèdent une large gamme d'applications potentielles à l'industrie, la science, le transport, les infrastructures civiles, et de la sécurité. Nous pouvons citer à titre d'exemple : la surveillance médicale et le contrôle des patients, détection d'intrusions et contrôle d'accès, contrôle de la pollution et découverte des catastrophes naturelles et industrielles dans des milieux hostiles.

Suivant l'historique des capteurs, les premiers capteurs inventés sont connectés par fil à un emplacement central en 1970. En 1980, les réseaux de capteurs câblés distribués sont apparus et à partir de l'année 1993, les projets sur ce domaine ont commencés tel que le projet de LWIM à UCLA et aussi le projet DARPA SENSIT: UC Berkeley, USC, Cornell (1999-2003). En 2001, le laboratoire de recherche d'Intel à Berkeley était focalisé sur les réseaux des capteurs sans fil, ainsi que le centre NSF était spécialisé sur les réseaux de détection embarqués. Dans la même période, les réseaux de capteurs sont émergés par l'industrie et les entreprises y compris Sensoria, Arbalète, Ember Corp, SensiCast. Il y avait des prototypes fabriqués par : Intel, Bosch, Motorola, General Electric,Samsung. En 2003, le standard IEEE 802.15.4 [1] est apparu.

1.2 Le standard IEEE 802.15.4

La norme la plus utilisée dans la technologie de réseaux de capteurs est la norme IEEE 802.15.4 [1]. Il existe beaucoup de protocoles de communication sans fil reposants sur cette norme dans le but est d'optimiser la communication entre les objets. Nous citons par exemple : Wireless HART, ISA-SP100, IETF IPv6 – LoWPAN, DigiMesh (réseaux maillés) et le plus répandu : ZigBee.

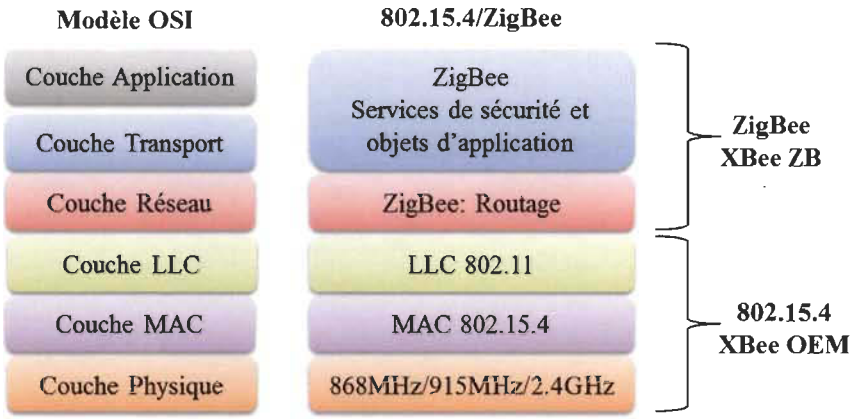


Figure 1.1 : Le modèle en couche (adaptée de [1])

Cette norme définit une couche de communication au niveau 2 (liaison des données) du modèle OSI [2]. Le but principal est de permettre la communication entre deux appareils. Elle a été créée par l'Institut d'ingénieurs en électrique et électronique (IEEE) [3]. Sa tâche essentielle est d'établir des normes de sorte que les progrès technologiques peuvent compter avec une plate-forme commune de règles à définir et compatible. Dans les dispositifs existants, ce protocole est défini à l'aide du module XBee 802.15.4 OEM [4].

Les unités d'information numériques (bits) sont gérées et organisées par la couche liaison de données pour devenir des impulsions électromagnétiques (ondes) au niveau inférieur (couche physique). Cette couche est similaire à celles d'autres connus comme la

norme 802.11 (la technologie Wifi) ou la norme 802.3 (Ethernet). Les fréquences définies dans la norme sont réparties entre 27 canaux différents divisés en trois groupes principaux:

- 868.0 - 868.6MHz → 1 canal (Europe)
- 902.0-928.0MHz → 10 canaux (États-Unis)
- 2.40-2.48GHz → 16 canaux (international)

1.3 Le protocole de communication ZigBee

Le protocole de communication ZigBee est le plus répandu pour les réseaux de capteurs. Il est basé sur le standard IEEE 802.15.4 [1].

Cette technologie définit une couche de communication au niveau 3 et dessous dans le modèle OSI. Son objectif principal est de créer une topologie de réseau (hiérarchie) afin de laisser un certain nombre de périphériques communiquer entre eux, et définit aussi les fonctions de communication supplémentaires telles que l'authentification, le cryptage et l'association. Elle a été créée par un ensemble de sociétés qui forment l'Alliance ZigBee [5]. Le module XBee ZB de Digi [6] s'inspire de ce protocole.

Trois émetteurs-récepteurs XBee différents sont utilisés pour fournir une communication dans les bandes de fréquence mentionnées précédemment:

- 868MHz → XBee 868MHz OEM [7]
- 900MHz → XBee 900MHz OEM [8]
- 2.40GHz → XBee 802.15.4 OEM [4] / XBee ZB [6]

La norme propose deux options de la couche physique basées sur la bande de fréquence. Toutes les deux reposent sur la séquence directe d'étalement du spectre (DSSS, Direct Sequence Spread Spectrum).

Tableau 1-1 : Bandes spécifiées par la norme IEEE 802.15.4 (adaptée de [1])

Bande (MHz)	Bande de fréquence (MHz)	Paramètres d'étalement		Paramètres de données		
		Taux de Chip (kchip/s)	Modulation	Débit binaire (kb/s)	Taux de symbole (ksymbole/s)	Symboles
868/915	868-868.6	300	BPSK	20	20	Binaire
	902-928	600	BPSK	40	40	Binaire
868/915 (optionnelle)	868-868.6	400	ASK	250	12.5	20 bits PSSS
	902-928	1600	ASK	250	50	5 bits PSSS
868/915 (optionnelle)	868-868.6	400	O-QPSK	100	25	16-aire Orthogonaux
	902-928	1000	O-QPSK	250	62.5	16-aire Orthogonaux
2450	2400-2483.5	2000	O-QPSK	250	62.5	16-aire Orthogonaux

Le tableau 1-1 dresse les paramètres de chaque bande définie par la norme IEEE 802.15.4. Le chip est un code symbole (PN, pseudo noise) : qui possède une fréquence beaucoup plus élevée que la fréquence des données d'entrée. Le débit de données est 250kbps à 2.45 GHz, 40kbps à 915MHz et 20kbps à 868MHz.

Le débit le plus élevé à 2.45 GHz est attribué à un schéma de modulation d'ordre supérieur. La basse fréquence fournit une portée accrue en raison de faibles pertes de propagation. Le faible taux peut être traduit en une meilleure sensibilité et une large zone de couverture. Le taux élevé signifie un débit plus élevé, une latence plus faible ou un faible rapport cyclique [1]. L'acheminement du flux de données (en émission et en réception) passé dans chaque bloc est mentionné sur la figure 1.2.

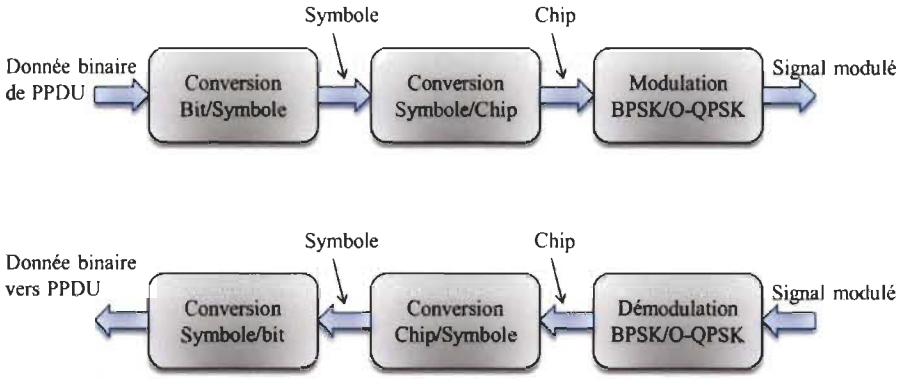


Figure 1.2 : L'acheminement du flux de données (adapté de [1])

1.4 Problématique

Nous pouvons signaler que la chaîne de communication en bande de base existante pour le protocole ZigBee est implémentée sur des circuits figés. Lorsque nous avons besoin à chaque fois de travailler sur un taux de transmission (à titre d'exemple) bien déterminé, nous serons obligé de changer le circuit au complet.

Donc nous avons un grand défi sur le fait d'implémenter une chaîne numérique en bande de base reconfigurable, pour ce protocole, sur FPGA, cette chaîne sert à :

- Configurer des paramètres dans le circuit programmable à savoir : différents taux de transmission (1 MHz, 2 MHz par exemple) ; prendre en considération les trois bandes 868 MHz, 915 MHz, 2,45 GHz ; deux schémas de modulation (QPSK et O-QPSK) ; et différentes résolutions des convertisseurs numérique-analogique (nombre de bits).
- Implémenter la technique d'étalement du spectre DSSS pour avoir un spectre large bande (à la place d'un modulateur numérique basé sur DDS).

- Effectuer un filtrage numérique reconfigurable avec la fonction d'interpolation et décimation pour différente fréquence d'échantillonnage ;
- Elle est contrôlée par une interface graphique développée sur un ordinateur ;
- Faire un test réel sur un circuit programmable FPGA et la validation par le matériel ;
- Faire un contrôle général d'émetteur/récepteur basant sur une machine à état finie afin de gérer les modes de transmission : Arrêt, boucle, émission et réception.
- Etudier la trame binaire PPDU venant de la couche MAC

L'avantage est que l'architecture à implémenter est bien définie par le standard, ce qui nous facilite sa description en VHDL bloc par bloc. Les trois bandes qui nous intéressent sont celles de trois dernières lignes du tableau précèdent (bande 868 MHz, bande 915 MHz et bande 2.45 GHz). Vu que les deux bandes (bande 868 MHz, bande 915 MHz) sont semblables au niveau de conception donc, nous étudierons seulement les deux bandes (bande 915 MHz et bande 2.45 GHz).

1.5 Objectifs spécifiques

Comme objectifs à court et long terme dans ce projet, nous pouvons les souligner de cette façon :

- Conception d'un émetteur en bande de base avec étalement du spectre (DSSS) ;
- Conception d'un émetteur récepteur en bande de base avec étalement du spectre (DSSS);

- Etude de la chaîne reconfigurable avec un étalement du spectre (DSSS) ;
- Développement d'une interface graphique en interrogeant avec la chaîne ;
- Test réel et validation sur le matériel ;
- Conception du contrôleur général des modes de transmission basant sur la machine à état finie.

1.6 Méthodologie

La première étape du travail est de faire l'état de l'art autour de notre sujet de recherche afin de clarifier les idées et d'identifier la problématique liée aux travaux antérieurs. Nous montrons les limites de la portée des travaux effectués par les prédécesseurs, ainsi, nous survolons les travaux connexes en faisant une analyse de fonctionnement; qui comprend les hypothèses projetées et aussi les démarches à suivre en terminant par une présentation sur les résultats obtenus.

L'étape suivante consiste à, détailler la partie essentielle décrite par la composition de la couche physique définit par le standard IEEE 802.15.4, ainsi que les différentes techniques utilisées. Nous parlons aussi sur les plateformes reconfigurables utilisant FPGA comme circuit pour le test et débogage.

La troisième étape s'agit de, présenter la composition de notre chaîne de communication proposée; en décrivant toute la chaîne en émission et en réception. Ainsi, elle permet d'expliquer la description et l'implémentation de la chaîne en VHDL et aussi le processus de développement qui décrit la méthodologie utilisée pour réaliser ce projet.

La dernière étape de notre projet comprend la conception des éléments de notre chaîne par programmation en VHDL, nous exposons les différents résultats des simulations obtenus et aussi les différents tests effectués avec la validation sur le matériel.

1.7 Organisation du mémoire

La structure du mémoire est décrite comme suit : le deuxième chapitre donnera une idée globale sur la composition de la couche physique définit par le standard IEEE 802.15.4, ainsi que les différentes techniques utilisées.

La troisième partie présente la chaîne de communication proposée. Nous mettrons dans l'annexe B les blocs internes de toute la chaîne d'émission et réception, en plus, nous expliquerons la description de différents blocs à implémenter en VHDL. Nous parlerons aussi sur l'interface graphique à développer pour pouvoir communiquer avec la chaîne à partir d'un ordinateur, ainsi que les éléments requis pour la validation de ce projet.

Le quatrième chapitre va exposer les résultats des simulations obtenus ainsi que les tests réels effectués durant le projet. Sachant que l'annexe D comporte une panoplie de nos travaux supplémentaires réalisés.

Enfin, ce mémoire se terminera par une brève conclusion en survolant les parties essentielles du rapport et en mentionnant quelques perspectives.

Chapitre 2 - La couche physique de IEEE 802.15.4

2.1 Introduction

Il y a quelques années, un nouveau standard est apparu qui est dédié au domaine à faible taux de transfert de données sans fil personnel (LR-WPAN). Ces réseaux sont caractérisés par le faible coût et le faible rendement de traitement. Cette norme connue sous le nom IEEE 802.15.4 et le protocole ZigBee représente le résultat des efforts du groupe de cette norme avec Alliance ZigBee. Ce groupe est responsable de la couche physique (PHY) et de la couche d'accès au support (MAC), tandis que ZigBee Alliance a été concentrée dans les couches supérieures.

Nous allons nous concentrer dans ce chapitre sur la composition de la couche physique définie par le standard IEEE 802.15.4 [9] et les différentes techniques utilisées. Ainsi, une brève introduction sera réservée pour la configuration numérique d'un système radio fréquence par un circuit programmable FPGA.

2.2 La composition de la couche physique pour la norme IEEE 802.15.4

Le standard définit 27 canaux répartis dans trois bandes de fréquences: 1 canal réservé dans la bande 868 MHz, 10 canaux dans la bande 915 MHz, et 16 canaux dans la bande 2.4 GHz.

Les fréquences des différents canaux sont données par l'équation suivante :

$$F(k) = \begin{cases} 868.3\ \text{MHz} & \text{si k} = 0; \\ 906 + 2\,(\text{k} - 1)\text{MHz} & \text{si k} = 1..,10; \\ 2405 + 5\,(\text{k} - 11)\text{MHz} & \text{si k} = 11..,26; \end{cases} \quad (2.1)$$

Où k est le numéro du canal.

2.2.1 La trame des données PPDU

Le tableau 2-1 montre les types de données de la trame PPDU dans la couche physique venant de la couche MAC.

Le champ entête de synchronisation ou SHR permet au récepteur de synchroniser et de verrouiller dans le flux binaire. L'entête couche physique ou PHR contient la longueur de la trame d'information. La charge utile de la couche physique (Payload en anglais) est fournie par les couches supérieures et elle comprend des données ou des commandes qui doivent être transmises à un autre dispositif.

La longueur du préambule de la modulation O-QPSK est de 4 octets (8 symboles): 868 MHz → 320 µs; 915 MHz →128 µs; 2.45 GHz →128 µs.

La longueur du champ début délimiteur de trame ou SFD pour la modulation O-QPSK est de 1 octet (2 symboles) pour 868 MHz; 915 MHz et 2.45 GHz.

Le temps maximal qu'un émetteur-récepteur en bande de base ait besoin pour passer de la transmission (TX) à la réception (RX) et vice-versa est de 12 périodes symbole (T_s).

Le premier bit qui sera transmis est celui le moins significatif (LSB) du champ SHR, le plus significatif (MSB) du dernier octet de la charge utile de la couche physique est transmis en dernier.

Tableau 2-1 : La trame PPDU de la couche physique (adapté de [9])

Type des données PPDU	vers LSB ← ←		Trame PPDU		→ → vers MSB
	SHR		PHR		PHY payload
	4 octets	1 octet	1 octet		Variable (de 0 octet jusqu'au 127 octets)
	Préambule	SFD	Longueur de données (7 bits)	Réservé (1 bit)	PSDU ou MPDU
Réservé	00000000 (x 4)	11100101	0000000	0	Pas des données
	00000000 (x 4)	11100101	1000000	0	1 octet des données
	00000000 (x 4)	11100101	0100000	0	2 octets des données
	00000000 (x 4)	11100101	1100000	0	3 octets des données
	00000000 (x 4)	11100101	0010000	0	4 octets des données
MPDU (acquittement)	00000000 (x 4)	11100101	1010000	0	5 octets des données
Réservé	00000000 (x 4)	11100101	0110000	0	6 octets des données
	00000000 (x 4)	11100101	1110000	0	7 octets des données
	00000000 (x 4)	11100101	0001000	0	8 octets des données
MPDU (données utiles)	00000000 (x 4)	11100101	1001000	0	9 octets des données
	:	:	:	:	:
	00000000 (x 4)	11100101	1111111	0	127 octets des données

2.2.2 Technique d'étalement du spectre par DSSS

La norme IEEE 802.15.4 utilise la méthode d'étalement du spectre en séquence directe (DSSS). En effet, les symboles sont élargis par des séquences pseudo-aléatoire PN de 32 bits (les chips) lorsque le nombre de « 0 » est égal au nombre des « 1 ». Chaque séquence PN est générée à partir du précédent par une rotation de quatre bits vers la droite dans les séquences des huit premiers symboles.

Tableau 2-2 : Pour la bande 2.45 GHz utilisant O-QPSK (adapté de [1] et [9])

Symbole de données (décimale)	Symbole de données (binaire) (b_0 b_1 b_2 b_3)	Valeurs du Chip (c_0 c_1 … c_{30} c_{31})
0	0000	11011001110000110101001000101110
1	1000	11101101100111000011010100100010
2	0100	00101110110110011100001101010010
3	1100	00100010111011011001110000110101
4	0010	01010010000101110110110011100011
5	1010	00110101001000101110110110011100
6	0110	11000011010100100010111011011001
7	1110	10011100001101010010001011101101
8	0001	10001100100101100000011101111011
9	1001	10111000110010010110000001110111
10	0101	01111011100011001001011000000111
11	1101	01110111101110001100100101100000
12	0011	00000111011110111000110010010110
13	1011	01100000011101111011100011001001
14	0111	10010110000001110111101110001100
15	1111	11001001011000000111011110111000

Les séquences des 8 derniers symboles sont générées en inversant les bits des 8 premières paires. Ainsi, à partir de la séquence correspondante au symbole zéro, nous pouvons générer tous les autres. Dans l'implémentation, les chips sont générés en basant sur le premier symbole en utilisant un tampon circulaire et chaque séquence du chip prend une certaine position du tampon. Le tableau 2-2 présente les 16 symboles définis par la norme avec sa séquence correspondante.

La séquence est convertie [une séquence chip de 2 Mcps (32 x 62.5 Kbps)] à travers un processus d'étalement de 32 fois (DSSS), c'est un type de séquence orthogonale qui est utilisée afin de fournir un taux d'erreur du paquet (PER) stable dans le système de Zigbee sans codage canal. La figure 2.1 représente l'effet d'étalement avec DSSS sur l'onde originale. Le DSSS est une technique de modulation pour la communication sans fil. Il étale les bits de données à une grande largeur de bande et occupe plus de bande passante que la bande passante d'origine du signal transmis. Toutefois, il occupe une très faible densité spectrale de puissance pour le signal. Ici, le facteur clé est la faible densité spectrale de puissance, ce qui entraîne moins d'interférences aux autres services partageant la même bande.

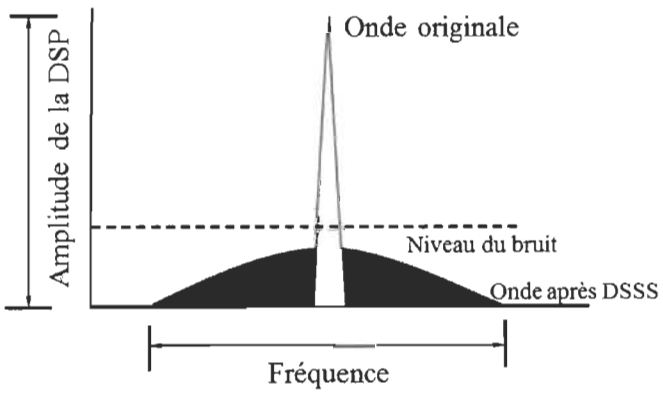


Figure 2.1 : Effet d'étalement avec DSSS (adaptée de [9])

Il y a une autre architecture pour la bande 868/915 MHz en utilisant PSSS comme technique d'étalement au lieu de DSSS avec la modulation BPSK et ASK (250 Kb/s) ou aussi une architecture qui utilise la technique DSSS avec le schéma de modulation O-QPSK (le débit binaire de 250 Kb/s), tel que nous utilisons un mappage d'un symbole à 16 séquences de 16 chips (au lieu de 32 chips pour la bande 2.45 GHz), comme le montre le tableau 2-3.

Tableau 2-3 : Pour les bandes 868/915 MHz avec O-QPSK (adaptée de [1] et [9])

Symbole de données (décimale)	Symbole de données (binaire) (b_0 b_1 b_2 b_3)	Valeurs du Chip (c_0 c_1 ... c_{15})
1	0000	0011111000100101
2	1000	0100111110001001
3	0100	0101001111100010
4	1100	1001010011111000
5	0010	0010010100111110
6	1010	1000100101001111
7	1110	1111100010010100
8	0001	0110101101110000
9	1001	0001101011011100
10	0101	0000011010110111
11	1101	1100000110101101
12	0011	0111000001101011
13	1011	1101110000011010
14	0111	1011011100000110
15	1111	1010110111000001

Pour avoir besoin plus d'informations sur les architectures de la norme IEEE 802.15.4, veuillez consulter les références [10], [11], [1], [12], [9], et [13].

2.2.3 La modulation QPSK versus O-QPSK

Le standard IEEE 802.15.4 utilise la modulation O-QPSK, elle présente plusieurs avantages par rapport QPSK: les variations de phase, sont réduites de 180° à 90°, peuvent utiliser, dans une transmission, un amplificateur non linéaire (Vous pouvez en savoir plus à ce sujet dans [15]).

La modulation QPSK, dans un flux des données d_k=d0, d1, d2,..., est composée d'impulsions bipolaires de données +1 ou -1, elle est divisée en deux composantes: une composante de phase $d_I(t)$ = d0, d2, d4,… et une composante en quadrature, $d_Q(t)$= d1, d3, d5,… dans laquelle la composante de phase est constituée par les bits pairs et la composante en quadrature par les bits impairs avec un débit binaire en moitié de $d_k(t)$.

Une onde modulée en QPSK est obtenue en utilisant les fonctions sinus et cosinus d'une onde porteuse, résultant en:

$$s(t) = \frac{1}{\sqrt{2}} d_I(t) \cos\left(2\pi f_0 t + \frac{\pi}{4}\right) + \frac{1}{\sqrt{2}} d_Q(t) \sin\left(2\pi f_0 t + \frac{\pi}{4}\right) \quad (2.2)$$

Qui peut être écrite comme: $s(t) = \frac{1}{\sqrt{2}} d_I(t) \cos(w_0 t + \theta(t))$ (2.3)

Les signaux d_I et d_Q sont modulés en amplitude par la fonction cosinus avec une amplitude de +1 ou -1, qui équivaut à un décalage de phase de cosinus ou de sinus de 0 ou π, où les deux étant orthogonaux. La somme de ces deux signaux orthogonaux résulte une

onde modulée QPSK, où la valeur de $\Theta(t)$ correspond à l'une des quatre combinaisons de d_I et d_Q : 0°, ±90°, 180°.

La technique de modulation O-QPSK peut être représentée de la même manière. La différence est dans un décalage temporel dans le signal d_Q. La durée de chaque impulsion d'origine $d_k(t)$ est de T et les deux signaux sont distincts de 2T. En QPSK, les impulsions paires et impaires sont transmises à un débit de 1/2T bits/s, elles sont alignées de telle sorte que leurs transitions coïncident. Dans O-QPSK, le signal d_Q se décale d'une période de temps T.

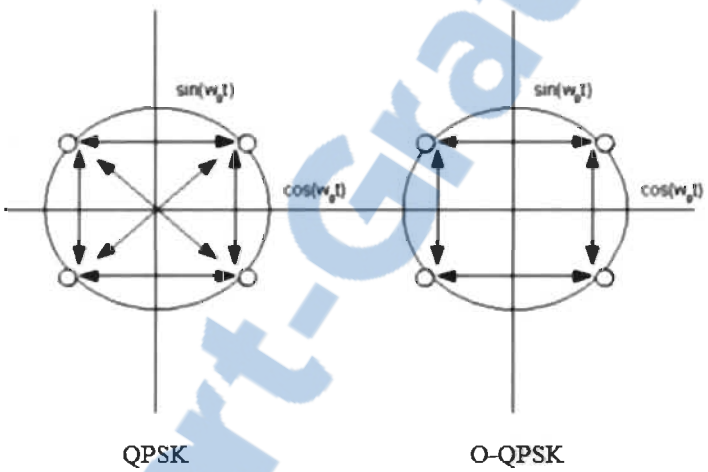


Figure 2.2 : Les changements de phase dans QPSK et O-QPSK (adaptée de [17])

En QPSK, dans chaque période de 2T, la phase peut changer. Si les deux signaux changent en même temps, nous obtenons un décalage de phase de 180° mais si nous la changeons à 90° et si nous ne la changeons pas, la phase reste la même. Le problème réside dans les changements de phase 180°: quand un signal de ce type passe à travers un filtre passe-bas, ces changements de phase se produisent avec de très grands changements d'amplitude qui est indésirable dans un système de communication.

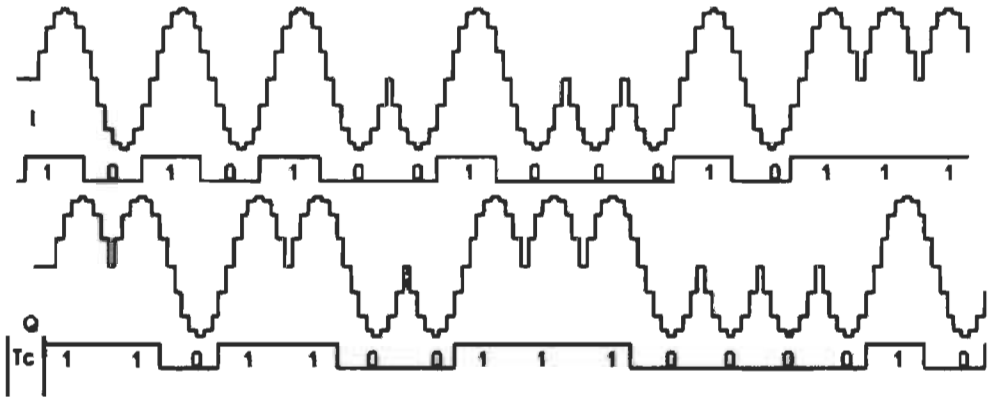


Figure 2.3 : Résultat de la modulation du symbole zéro (adaptée de [13])

En O-QPSK, il n'est pas possible que les deux signaux changent simultanément, vu que les changements de phase sont limités entre 0 et 90 ° (figure 2.2). Dans la figure 2.3, le symbole 0 du standard est modulé en O-QPSK. Vous pouvez trouver plus d'informations sur les avantages d'O-QPSK par rapport QPSK dans [16].

2.2.4 Estimation de la fréquence et correction de la phase dans le récepteur

Dans un système de communication sans fil les imperfections des oscillateurs dans l'émetteur et le récepteur conduisent à un décalage dans le support qui est détecté dans le récepteur. Ce décalage provoque une rotation de la constellation du signal qui doit être corrigé pour obtenir les données originales dans le processus de la démodulation.

Le récepteur utilise l'action combinée de deux méthodes d'estimation de fréquence et de correction de phase. D'abord, nous utilisons l'estimateur de Kay pour faire une première estimation (grossière) et une fois que nous synchronisons avec les données reçues, nous faisons une deuxième estimation « fine » en sachant les données que nous devons obtenir.

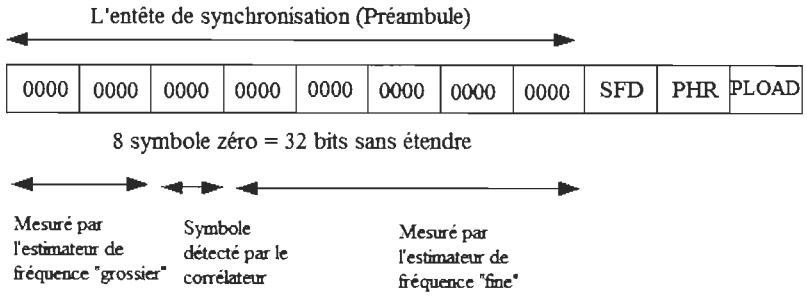


Figure 2.4 : Esquisse d'estimation et de correction de phase

L'algorithme de Kay est un estimateur de fréquence pour les sinusoïdes complexes qui est basé sur la technique de la prédiction linéaire. Dans la partie de la démodulation du récepteur, les données reçues sont affectées par un bruit de nature gaussien qui se traduit par des décalages de phase. Ces données peuvent être interprétées comme des données erronées. L'algorithme de Kay nous permet de faire une correction «aveugle» des données entrantes, selon la valeur moyenne de la différence de phases entre les échantillons mesurés.

L'implémentation d'estimateur de Kay, utilise le module CORDIC (Coordinate Rotation Digital Computer) pour estimer les phases des échantillons entrants pendant la mesure et le mode rotationnel du CORDIC; une fois que la mesure est terminée et que la correction est exécutée.

Le CORDIC permet d'implémenter des fonctions élémentaires spécifiques (trigonométrique et hyperbolique) en utilisant une configuration matérielle minimale à l'aide des opérations de déplacements simples, comparaisons, additions et soustractions.

Le CORDIC fonctionne grâce à des rotations dans le domaine complexe selon des angles constantes, jusqu'à ce que l'angle soit réduit à zéro. Ces angles sont choisis de

manière que les rotations peuvent être implémentées par des déplacements, additions et soustractions.

L'article original qui décrit l'algorithme peut être trouvé dans [20] et une généralisation de celui-ci dans [21]. Les différentes façons d'implémentation dans le matériel peuvent être trouvées dans [22].

L'estimation « fine » utilise une implémentation du CORDIC pour mesurer et faire tourner les échantillons, en contraste les mesures avec une table des phases prévues (avec les données qui doivent être reçues). Vous pouvez voir un aperçu des différentes parties du préambule qui traite chaque estimateur dans la figure précédente.

La trame PDU de la norme contient un préambule de 8 symboles zéro qui servent à synchroniser avec les données reçues. Une fois que nous détectons l'un des symboles zéro, nous ne pouvons qu'espérer pour détecter le champ SFD pour être complètement sûr que nous sommes synchronisés avec l'émetteur.

Il existe de nombreuses façons de combiner les deux estimateurs, utilisant plus ou moins une partie du préambule pour une meilleure estimation et une synchronisation avec les données reçues.

Dans ce cas, nous supposons que le corrélateur est incapable de détecter le symbole zéro qui se trouve dans la troisième position du préambule. C'est le troisième symbole zéro de l'en-tête de synchronisation. Par conséquent, l'estimateur de fréquence « grossier » effectue l'estimation durant les deux premiers symboles zéro du préambule, qui sont étendus et ils ont traversés le sous-échantillonneur, 256 chips à mesurer. Puis, l'estimateur

passe à corriger les phases des échantillons entrants suivants, en espérant que le corrélateur soit capable de détecter après le symbole zéro suivant.

Une fois que le corrélateur détecte un symbole zéro, et il sait que les échantillons suivants font partie du préambule et les données connues (symboles zéro de la norme), l'estimateur de fréquence «fin» est activé, qui estime jusqu'à ce que le préambule se termine (la comparaison des phases avec les données prévues). Lorsque l'estimateur «fin» termine l'estimation, il commence à corriger les échantillons entrants. Enfin, une fois le corrélateur détecte le champ SFD, nous sommes pleinement confiants de la synchronisation avec l'émetteur.

Pour plus d'informations sur l'estimateur de Kay ainsi que le module CORDIC, veuillez consulter l'annexe A.

2.2.5 L'indicateur de puissance du signal reçu (RSSI)

L'indicateur du système de niveau du signal ou RSSI (Received Signal Strength Indicator) peut être implémenté en utilisant le module CORDIC. Le CORDIC en mode vectoriel nous permet de convertir une valeur complexe de forme cartésienne en forme polaire (modulo, phase). Si vous vous souvenez, la formule de RSSI est donnée par :

$$RSSI = \frac{1}{N}\sum_{n=0}^{N-1}|m(n)|^2 = \frac{1}{N}\sum_{n=0}^{N-1}s_I^2(n) + s_Q^2(n) \tag{2.4}$$

Donc, nous accumulons juste le module des échantillons entrants (calculé avec le CORDIC rotationnel) pour N échantillons et à la fin, nous divisons par N échantillons. Cette division peut être implémentée par un décalage vers la droite si le nombre d'échantillons N est une puissance de 2.

2.3 Les architectures spécifiées par la norme

Dans ce qui suit, nous focalisons sur les différentes architectures définies par la norme IEEE 802.15.4 et plus particulièrement pour la bande 915 MHz et la bande 2.45 GHz.

2.3.1 L'architecture de la chaîne en bande de base pour la bande 2.45 GHz

Premièrement, nous allons parler sur la chaîne de transmission en bande de base spécifiée par la norme comme la montre la figure 2.5, elle est définie pour la bande 2.45 GHz. Pour plus des détails, veuillez consulter les références [10], [11], [1], [12], [9], et [13].

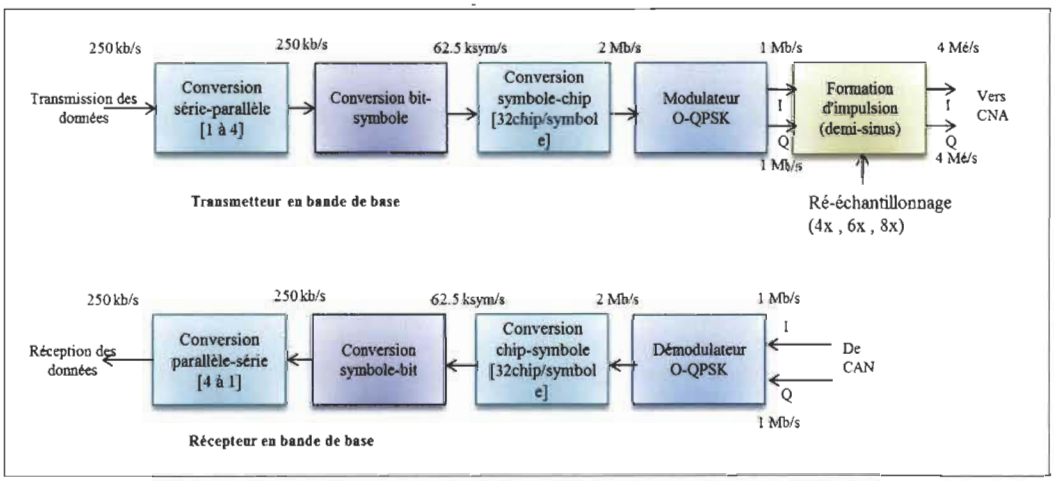


Figure 2.5 : L'architecture pour la bande 2.45 GHz (adaptée de [12])

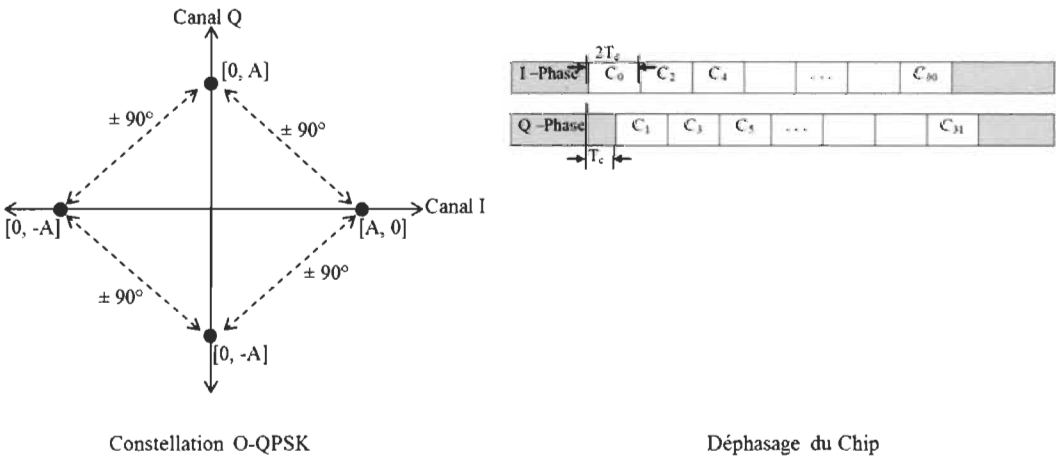
Nous voyons bien dans la figure 2.5 que l'acheminement des données se fait sur la voie en phase I et la voie en quadrature de phase Q, que ce soit pour la chaîne d'émission ou la chaîne de réception. Dans la chaîne d'émission, après la réception du paquet (les données d'entrée viennent sous forme des paquets PPDU de la couche MAC dans la pile protocolaire de l'IEEE 802.15.4), nous le passons dans un bloc qui n'est pas mentionné dans la figure. En effet, un bloc qui sert à convertir le mode de transmission du flux binaire

série au mode parallèle (1 à 4) avec un débit binaire de 250 kbps, il s'agit d'une conversion d'un flux binaire à une représentation en symbole pour avoir à la sortie un taux de 62.5 ksymbole/s. Comme le montre la figure, la chaîne d'émission commence par un bloc dédié pour la conversion (32 chip/s) ou le mappage du symbole (avec un taux de 62.5 ksymbole/s) vers une représentation en chip (le taux est de 2 Mchip/s, avec 1 Mchip/s par canal), nous suivons la chaîne par un sur-échantillonneur avec un facteur égal à 8, pour avoir en sortie un taux de 8 Mé/s sur chaque canal (I et Q). La chaîne se termine par un bloc de filtrage de mise en forme de l'impulsion que nous pouvons l'injecter dans le convertisseur numérique-analogique pour un post-traitement analogique (module RF). Pareillement pour la chaîne de réception, nous avons juste les blocs inverses. Pour plus de détails, consultez les références [7] et [8].

L'architecture est composée de :

- Convertisseur série-parallèle qui convertie les données en série vers des données en parallèle.
- Convertisseur bit-symbole qui convertie les données en parallèle (4 bits de 250 Kbps) vers un symbole (de 62.5 Ksps).
- Convertisseur symbole-chip (étalement par DSSS) qui mappe le symbole vers 16 séquences de 32 chip (Tableau 2-2). Dans l'implémentation du système de modulation O-QPSK, l'émetteur fait le mappage de bits 1 et 0 pour 0 à 15 symboles, et les symboles (de 0 à 15) sont mappés à 16 séquences chips de 32 bits (bruit pseudo-aléatoire).

- Modulateur O-QPSK : transmet une séquence chip de 1 Mcps par les canaux I et Q à travers ce processus de modulation (une séquence de 16 chips sur chaque canal) → Ce schéma de modulation possède un maximum de déphasage de ± 90°. La figure 2.6 nous montre la constellation et le décalage de phase du Chip pour la modulation O-QPSK. Chaque chip pair de la séquence PN est modulé sur la porteuse en phase (I); tandis que chaque chip impair de la séquence PN est modulé sur la porteuse en quadrature de phase (Q). Surtout, les chips sur la voie Q sont décalés de T_c (l'inverse du taux de chip) par rapport aux chips sur la voie I en créant un décalage de phase et la modulation est dite O-QPSK.



Constellation O-QPSK

Déphasage du Chip

Figure 2.6 : La modulation O-QPSK (adaptée de [9] et [12])

- La formation d'impulsion de demi-sinus qui représente un filtre de mise en forme. La figure 2.7 illustre les échantillons des séquences chip en bande de base avec le filtrage de formation d'impulsion.

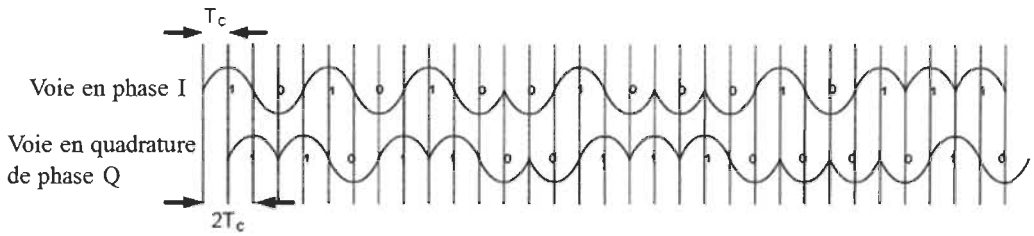


Figure 2.7 : Le filtrage de formation d'impulsion (adaptée de [9])

Le filtre de formation d'impulsion de demi-sinus change le diagramme de transition d'O-QPSK d'un carré à un cercle. Il supprime les variations d'amplitude et tourne donc O-QPSK en modulation à enveloppe constant. L'équation est utilisée pour chaque chip de bande de base par ce filtre est montrée ci-dessous, elle est décrite en détails dans [9].

$$P(t) = \begin{cases} \sin(\pi \frac{t}{2T_c}), 0 \le t \le 2T_c \\ 0, autrement \end{cases} \tag{2.5}$$

La figure 2.8 schématise les deux filtres numériques en bande de base que ce soit en émission (interpolation) ou en réception (décimation).

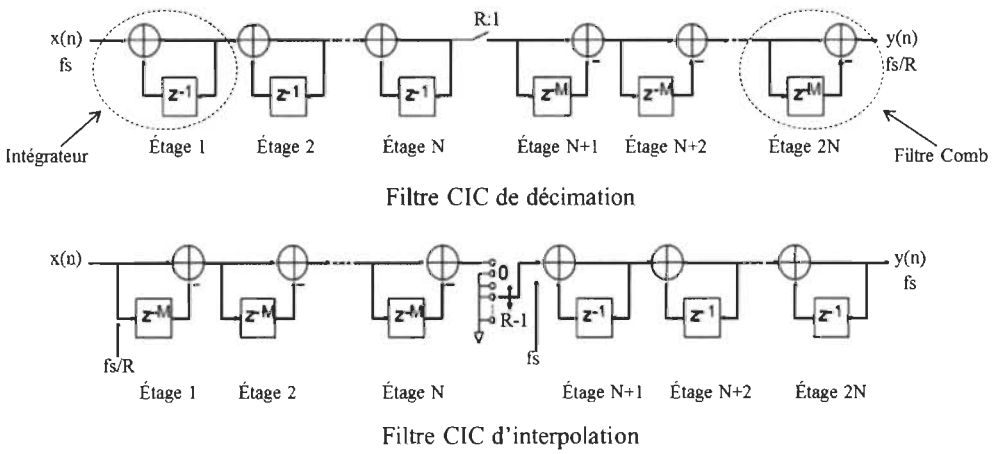


Figure 2.8 : Le filtrage numérique (adaptée de [14])

L'implémentation du filtre FIR est en forme directe. Pour obtenir la sortie, le filtre est basé seulement sur les données de l'entrée actuelle et les antérieures. La sortie du filtre peut être exprimée simplement comme la convolution du signal d'entrée avec la réponse impulsionnelle du filtre:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \tag{2.6}$$

Où M est l'ordre du filtre, h(k) sont les coefficients, x est l'entrée et y la sortie du filtre. Pour chacun des canaux I et Q, il existe un filtre séparé.

Les opérations du filtre sont limitées à l'addition, la soustraction et le déplacement étant donné que les données entrantes sont 1, -1 ou 0, oubliant la multiplication par les coefficients. L'implémentation du filtre est basée sur la référence [14].

Le filtre de réception est identique à l'émission, permettant une bonne valeur du SNR pour que le codeur RZ puisse faire la distinction entre 1 ou 0 avec plus de probabilité de succès.

- Démodulateur O-QPSK : il effectue un processus de synchronisation afin de connaitre le début de la séquence chip reçue (démodulation non cohérente).

  ➔ Les données démodulées sont traitées par l'inverse du processus de transmission (conversion chip-symbole, conversion symbole-bit, et conversion parallèle-série).

  ➔ Le taux de transmission supporté : 1 Mbps / 2 Mbps → la méthode de diminution du taux d'étalement à travers la modification du convertisseur série-parallèle, convertisseur bit-symbole, et convertisseur symbole-chip.

Par exemple : pour le taux 1 Mbps → nous avons une conversion vers 16 séquences de 8 chips, tel que la longueur de la séquence chip transmise sur les canaux I et Q, est égale à 4 (16 séquences de 4 chips sur I et 16 séquences de 4 chips sur Q).

➔ Le système peut augmenter le taux de transmission sur la même largeur de bande utilisant 8-PSK, QAM, ou un système multicode, mais, il est très complexe parce qu'il n'y a plus de séparation du pilot canal (l'erreur de la haute fréquence).

La référence [9] incite sur la conception et l'implémentation de l'émetteur / récepteur pour la norme IEEE 802.15.4 en utilisant MATLAB. Il a implémenté la séquence directe d'étalement du spectre (DSSS) utilisant la modulation (O-QPSK) avec la mise en forme de demi-impulsion sinusoïdale. La référence [11] incite sur la mise en œuvre de l'émetteur / récepteur du protocole ZigBee avec ses spécifications. Il a utilisé Matlab / Simulink pour la modélisation, sans recourir aux blocs mathématiques complexes.

Concernant la partie frontale analogique, la chaîne pour le sens montant peut contenir des filtres passe-bas intermédiaires, un étage d'amplification de puissance du signal, voire même un mélangeur des signaux, en se référant à un oscillateur local qui nous fixe la bande de fréquence (micro-onde dans notre cas). Le modulateur nous permet de transposer la fréquence en bande de base (après conversion en signal analogique) vers une autre plus haute, qui peut être du type QPSK ou O-QPSK. Cette dernière doit être autour de la fréquence de l'oscillateur local.

2.3.2 L'architecture de la chaîne en bande de base pour la bande 915 MHz

La figure 2.9 illustre l'architecture de la chaîne d'émission et réception en bande de base pour la bande 915 MHz, elle utilise la modulation BPSK et un codage différentiel.

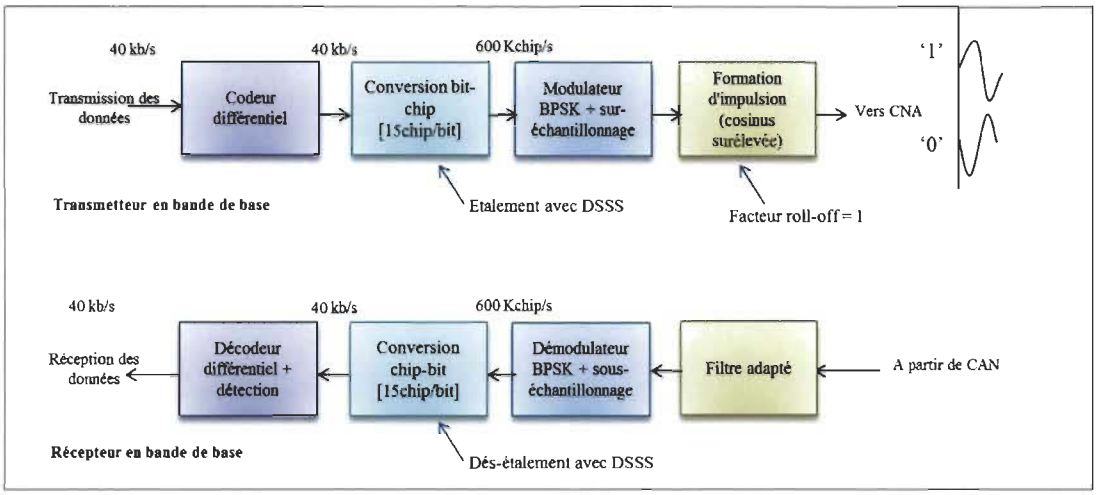


Figure 2.9 : L'architecture pour la bande 915 MHz (adaptée de [10])

Le but du codage différentiel est de résoudre le problème d'ambiguïté sur la phase au niveau du récepteur. En effet, en codage : $E_n = R_n$ xor E_{n-1} tel que : R_1 est le premier bit de données à coder et $E_0 = 0$, avec:

R_n : premier bit des données étant codé;

E_n : bit correspondant codé différentiellement;

E_{n-1} : bit précédant codé différentiellement.

Pour le décodage : $R_n = E_n$ xor E_{n-1} tel que : E_1 est le premier à décoder et $E_0 = 0$.

Le tableau 2-4 explique le mappage du bit vers 2 séquences de 15 chips utilisant un étalement par DSSS.

$T_c = T_b / N$; avec T_c : durée du chip, T_b : période du bit, N : longueur du code.

Tableau 2-4 : Pour les bandes 868/915 MHz avec BPSK (adaptée de [1] et [9])

Les bits en entrée	Valeurs du Chip (c_0 c_1 ... c_{14})
0	111101011001000
1	000010100110111

Concernant la modulation BPSK (expliquée sur la figure 2.10), elle consiste à avoir une fréquence d'échantillonnage ≥ 10 x Fréquence porteuse; à la réception : le décodeur symbole extrait le temps symbole et les valeurs du symbole du signal BPSK reçu (période symbole et seuil). La période symbole est le nombre entier de cycles d'horloges pour une fréquence d'échantillonnage choisie. Le seuil est l'amplitude relative, qui détermine la présence du symbole (0 ou 1) dans le décodeur (si le seuil diminue → la sensibilité du décodeur augmente et si le seuil augmente → la sensibilité du décodeur diminue).

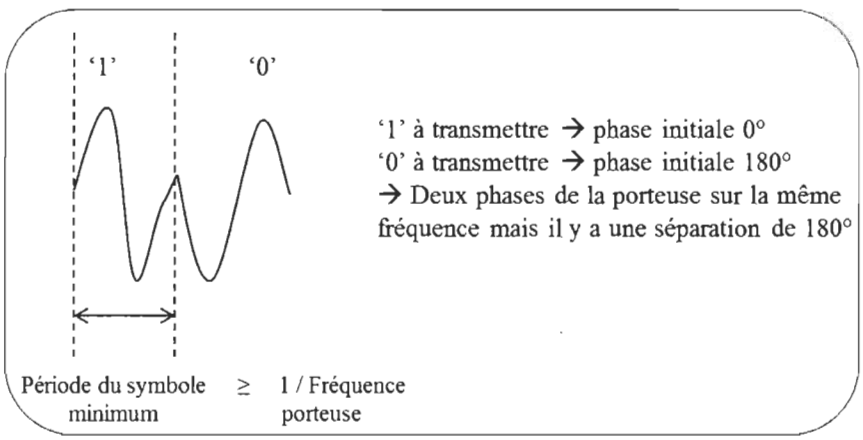


Figure 2.10 : La modulation BPSK

Le filtre de formation d'impulsion en cosinus surélevée (en supposant que le coefficient de retombée = 1) a la particularité de ne pas produire d'interférence entre symboles et d'être à support spectral borné; il est exprimé par l'équation suivante :

$$p(t)=\begin{cases}\dfrac{\sin\dfrac{\pi t}{T_c}}{\dfrac{\pi t}{T_c}}\times\dfrac{\cos\dfrac{\pi t}{T_c}}{1-\dfrac{4t^2}{T_c^2}}, t\neq 0\\ 1, t=0\end{cases} \tag{2.7}$$

2.4 La reconfiguration par le circuit programmable FPGA

Nous savons bien, que tous les éléments logiques dans les FPGA peuvent s'exécuter en parallèle. Cela inclut les multiplicateurs matériels et nous pouvons maintenant obtenir plus de 1000 d'entre eux sur un seul FPGA. Ceci est en contraste avec les processeurs TSN programmables, qui ont généralement quelques multiplicateurs et ils doivent être exploités de manière séquentielle.

Les FPGA ont désormais des interfaces série et parallèle spécialisées, pour répondre aux exigences des périphériques à haute vitesse et des bus.

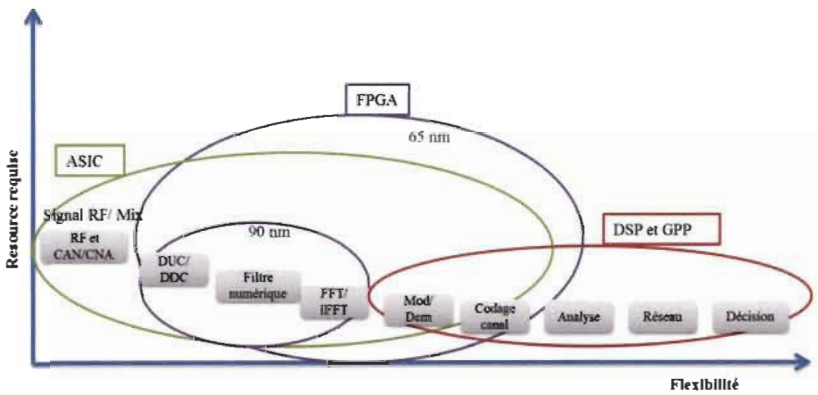


Figure 2.11 : Les tâches des applications en communication (adaptée de [9], [10] et [11])

Ce que nous entendons par intensité du processus (figure 2.11) est le degré d'opérations fortement répétitives et assez primitives. En haut à gauche, ce sont les fonctions dédiées comme des convertisseurs CAN et CNA, et aussi les DDC (les convertisseurs-abaisseurs de

fréquence numérique ou Digital Down Converter en anglais) et DUC (les convertisseurs-élévateurs de fréquence numérique ou Digital Up Converter en anglais), ces derniers nécessitent des structures matérielles spécialisées pour exécuter les opérations en temps réel.

La flexibilité concerne l'unicité ou la variabilité du traitement, et de la façon dont la fonction peut être modifiée ou adaptée pour toute application spécifique. En bas à droite ce sont des tâches telles que l'analyse et la prise de décision; qui sont très variables et souvent subjectives.

Les processeurs programmables à usage général (GPP) ou les DSP, sont généralement choisis pour ces tâches, puisque ces tâches peuvent être facilement modifiées par logiciel.

En conséquence, les FPGAs ont considérablement envahi l'espace des tâches d'application, comme indiqué par la bulle centrale dans le diagramme des tâches ci-dessus. Ils offrent des avantages du parallélisme matériel, pour traiter certaines fonctions à processus de haute intensité comme le DDC; et le bénéfice de programmation pour accueillir une partie du décodage et d'analyse des processeurs de TNS (DSP).

Ces avantages peuvent se faire au détriment de la dissipation de puissance accrue et les coûts de production plus élevés. Toutefois, ces considérations sont souvent secondaires à la performance et aux capacités de ces dispositifs remarquables.

Il y a certains paramètres à identifier qui servent à la configuration d'un système de communication, nous citons les paramètres clés (à titre d'exemple) comme suit :

- La méthode d'échantillonnage : elle décrit si le système permet un échantillonnage d'un signal complexe en I et Q (échantillonnage en quadrature),

ou tout simplement d'un signal réel. Dans le premier cas, ça nécessite deux convertisseurs CAN/CNA (ou un CAN/CNA multi canaux), et dans le second seulement un CAN/CNA est requis.

- Le dispositif d'échantillonnage : ce paramètre décrit le type de périphérique d'échantillonnage qui est utilisé dans le système. Ça peut être une carte son d'ordinateur, un CAN/CNA audio dédié, une carte d'acquisition dédiée, un CAN/CNA RF dédié, ou un ASIC dédié spécialisé au traitement frontal analogique à fréquence intermédiaire.
- Le mélangeur analogique : Certains systèmes ne sont pas entièrement numériques, car un convertisseur-abaisseur analogique de fréquence ou un convertisseur-élévateur analogique de fréquence est fait avant que l'échantillonnage s'effectue. S'il existe, ce mélange analogique peut être programmable à partir de l'ordinateur, ou parfois il n'est pas programmable (fixe). Nous pouvons rencontrer les scénarios suivants :
  - La fréquence peut être fixée par un oscillateur local OL (il n'est pas programmable par logiciel ou programmable manuellement) ;
  - Ou l'OL est programmable, et réalisé par une puce VCO et PLL (par exemple LMX2486, LMX1501A, MC145170) ;
  - Ou l'OL est programmable, et réalisé par un DDS sur un ASIC (par exemple: AD9958/59, AD9951, AD9854, AD9851) ;
  - Ou même l'OL est programmable, et réalisé par un DDS sur un FPGA (puce programmable).

- La sélection de la chaîne numérique : l'échantillonnage est accompli avec un taux élevé afin de numériser une dimension du spectre beaucoup plus large que la bande occupée par un seul canal interne. La sélection du canal est alors effectuée numériquement par le convertisseur-abaisseur numérique de fréquence qui représente une combinaison de mélange numérique, de filtrage et de décimation (ou interpolation). Ce paramètre décrit comment la sélection de la chaîne numérique est atteinte. Nous trouvons le choix de:
  - La sélection du canal numérique se fait avec un FPGA ;
  - La sélection du canal numérique se fait avec un ASIC (puce spécifique. Par exemple AD6620, AD6636, HSP50214, HSP50016) ;
  - La sélection du canal numérique se fait par un ASIC spécialisé dédié pour le front end à FI avec les deux fonctions CAN et DDC (par exemple AFEDRI8201) ;
  - La sélection du canal numérique se fait via le logiciel dans un PC ;
  - La sélection du canal numérique se fait par une puce DSP Array ;
  - Ou la sélection n'est pas possible, par exemple, lorsque la sélection du canal est faite dans le côté analogique et le côté à fréquence intermédiaire numérique de la largeur du canal souhaitée.
- Le transfert d'échantillons numériques vers l'ordinateur : il décrit la façon dont les échantillons numériques sont transférés vers l'ordinateur. Ça peut être par soit:
  - Les échantillons numériques sont transférés vers le PC via l'interface USB ;

- Les échantillons numériques sont transférés vers le PC via l'interface Ethernet ;
- Les échantillons numériques sont transférés vers le PC à travers l'interface série RS232 par exemple ;
- Ou même les échantillons numériques ne sont pas transférés à un PC mais ils sont traités localement.

- Le contrôle du système : il décrit la façon dont le système est contrôlé à distance. Ce contrôle est par exemple nécessaire pour programmer l'oscillateur local (OL), le convertisseur abaisseur numérique de fréquence (DDC), ou le convertisseur élévateur numérique de fréquence (DUC). Ça peut être par un ordinateur via l'interface série, parallèle, USB, ou Ethernet. Nous pouvons aussi le contrôler localement à travers un processeur intégré (par exemple dans un FPGA).

2.5 Conclusion

Nous avons étudié dans cette partie du rapport la composition de la couche physique définit par le standard IEEE 802.15.4, à savoir la modulation O-QPSK, la technique d'étalement du spectre DSSS, et les architectures internes pour chaque bande de fréquence. Ainsi, nous avons décrit la configuration numérique d'un système de communication par un circuit programmable FPGA.

Le chapitre suivant mettra la lumière sur notre chaîne de communication proposée; dans ce dernier, nous présenterons les éléments essentiels de cette chaîne, ainsi que la méthodologie de conception et l'explication de la description en VHDL.

Chapitre 3 - Chaîne de communication proposée

3.1 Introduction

Nous présentons dans ce chapitre la composition de notre chaîne de communication proposée. Nous présenterons les éléments essentiels de cette chaîne, ainsi que la méthodologie de conception et l'explication de la description en VHDL. L'annexe B contient une explication plus détaillée afin de comprendre l'architecture interne.

3.2 La composition de la chaîne proposée

Le but de notre travail est d'implémenter une chaîne d'émission / réception en bande de base (la couche physique du protocole de communication ZigBee). Nous jouons sur quelques paramètres de cette chaîne (comme le taux d'échantillonnage, le taux de transmission, la résolution du convertisseur) afin de la rendre flexible. Une version de nos travaux sera sur l'implémentation du contrôleur de notre chaîne basé sur la machine à état fini et l'étude de la trame binaire PDU venant de la couche MAC.

Tout d'abord, la chaîne est constituée de trois éléments essentiels : la partie analogique (étudiée pour le test au niveau de l'oscilloscope et l'analyseur du spectre pour la chaîne d'émission) qui se traduit par la chaîne de transmission RF (émetteur) et le convertisseur numérique/analogique, la partie numérique pour le traitement des signaux utilisant un FPGA, et le dernier élément comporte l'interfaçage avec l'ordinateur. L'ordinateur sert à contrôler et configurer la chaîne de communication et ainsi d'envoyer et de recevoir des

échantillons (dans notre cas, des symboles voire même une trame binaire), moyennant une communication sérielle ou même à travers une liaison par câble USB.

Notre contribution dans ce projet est de faire la description en langage VHDL d'un émetteur/récepteur en bande de base (la couche physique) reconfigurable pour la bande 915 MHz et la bande 2.45 GHz, basant sur les architectures spécifiées par le protocole Zigbee. Les deux bandes utilisent le schéma de modulation O-QPSK (voire même QPSK sans conserver le décalage entre la composante en phase et la composante en quadrature de phase que ce soit en émission et en réception), elles seront implémentées avec la technique d'étalement du spectre DSSS (consultez le tableau 2-2 du deuxième chapitre). Nous nous intéressons aussi au développement de l'interface graphique pour communiquer avec la chaîne en bande de base; ainsi que, la mise en place d'un banc de test réel. Concernant la partie frontale analogique (étude du transmetteur seulement pour la bande 2.45 GHz), nous testons la partie numérique après la conversion des signaux numériques en signaux analogiques et aussi l'analyse spectrale après la modulation des signaux en bande de base. La troisième contribution consiste à, décrire en VHDL le contrôleur du mode de transmission basé sur une machine à état fini et ainsi d'étudier la trame PDU venant de la couche MAC.

3.2.1 Le premier travail à faire

Nous implémentons la chaîne d'émission et de réception décrite dans la norme IEEE 802.15.4 [9].

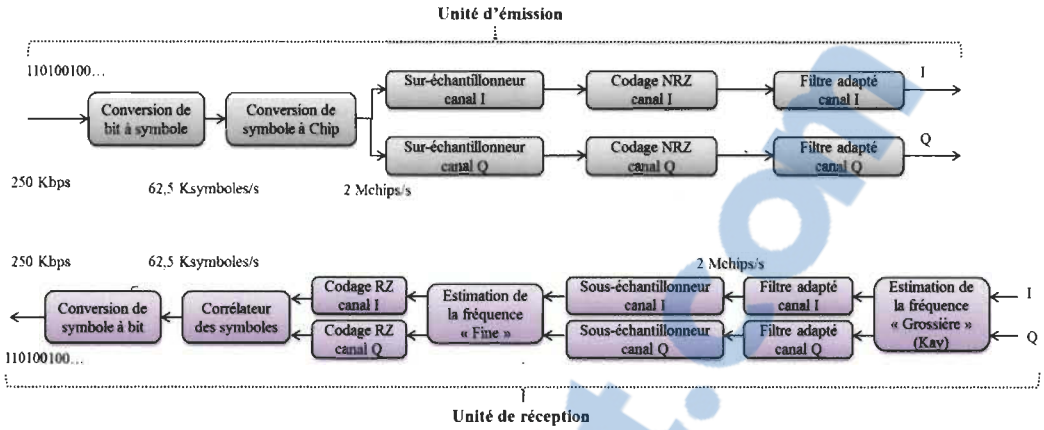


Figure 3.1 : L'architecture complète à implémenter (adaptée de [8] et [9])

La figure 3.1 représente l'architecture complète que nous allons l'implémenter au long de nos travaux, et comporte l'unité d'émission et de réception. Nous allons parler en détails sur cette architecture dans la section liée au troisième travail à faire.

Mais, dans un premier temps, (le premier travail) nous nous concentrons sur l'architecture qui envoie un seul symbole sur la chaîne d'émission et qui reçoit un seul symbole après le passage par les blocs qui forment la chaîne de réception. Donc, dans cette sous-section, nous ne traitons pas les blocs liés à la conversion de bit à symbole et à la conversion du symbole à bit.

La partie frontale analogique (en anglais, Analog Front-end) n'est pas vraiment à concevoir, nous l'étudions juste pour le test réel sur un oscilloscope multi-domaine avec le post-traitement des signaux après la chaîne numérique ainsi que l'analyseur du spectre.

La partie sur FPGA est composée de deux chaînes en émission en bande de base qui sont implémentées : la première fonctionne sur la bande 915 MHz et la deuxième fonctionne sur 2.45 GHz.

La différence de deux bandes au niveau de la conception réside dans la conversion du symbole vers chip et la conversion du chip vers symbole ; consultez le deuxième chapitre pour plus des détails.

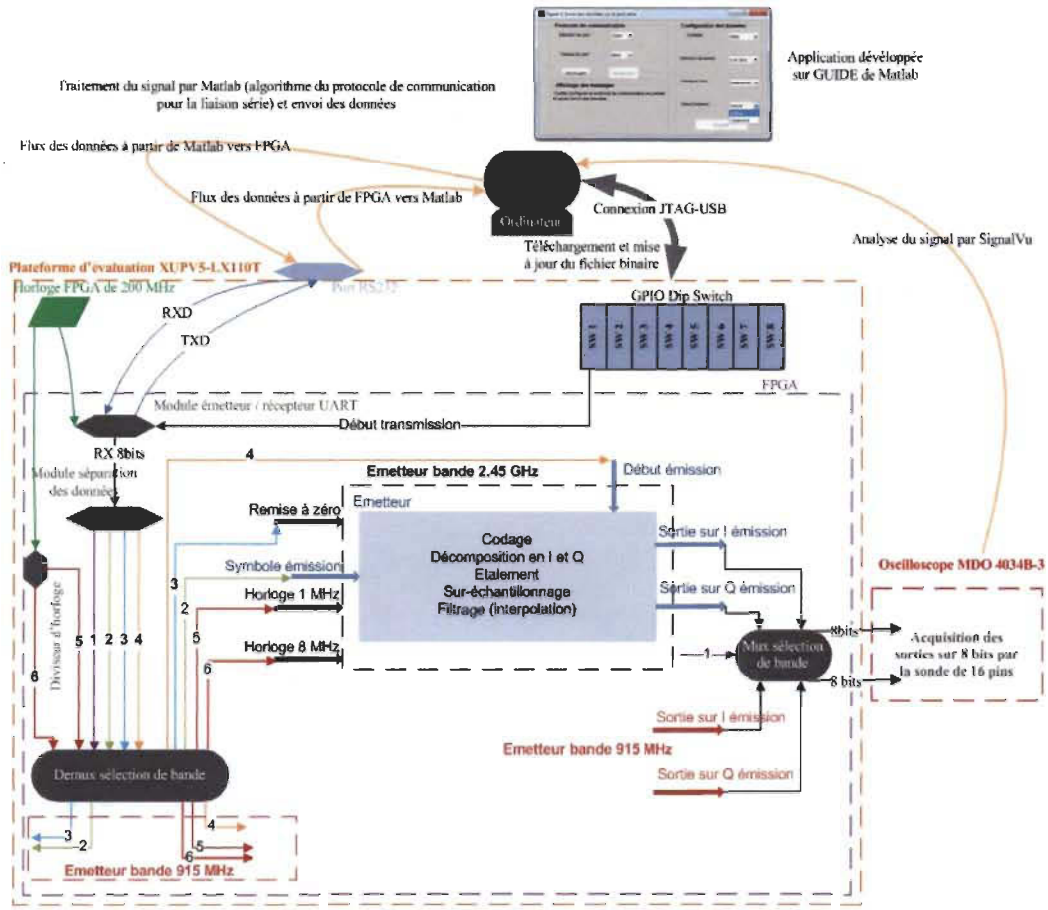


Figure 3.2 Schéma globale en émission

Donc nous faisons :

- L'acquisition des données numériques de deux sorties par l'oscilloscope MDO4034B-3 (existant à l'université) via une communication par la sonde de 16 bits;
- La communication entre l'ordinateur et la plateforme d'évaluation, en envoyant une donnée à partir d'une interface graphique sous Matlab vers un FPGA (dans notre

cas la configuration des entrées) et de recevoir une donnée (elle est testée seulement pour la boucle de retour), utilisant la liaison série via l'interface de communication UART (le développement de l'interface graphique est sur GUIDE de Matlab, il comprend l'algorithme du protocole de communication);

- La division d'horloge (description en VHDL) à partir de l'horloge FPGA 200 MHz vers les deux entrées d'horloge (8 MHz et 1 MHz);
- L'interfaçage du port série RS232 avec le FPGA via le module émetteur/récepteur UART (description en VHDL) ;
- La séparation des données venant du module UART vers les entrées à configurer (description en VHDL).
- L'adaptation des données de 4 bits à 8 bits juste pour le test de la boucle de retour (description en VHDL);

Donc tous ces modules numériques (à implémenter sur FPGA) sont décrits en VHDL.

Pour plus des détails, nous avons mis dans l'annexe B une description détaillée de chaque bloc à concevoir.

Le corrélateur de réception détecte les symboles de 4 bits du standard 802.15.4 à partir de l'entrée du codeur RZ. Il existe de nombreuses alternatives pour faire la détection des symboles. Dans notre cas, seules les données appartenant au canal Q sont utilisées, puisque nous utilisons seulement les données du canal I. Nous ne pourrions pas distinguer entre la première moitié des 16 symboles et la seconde (les séquences PN du canal I pour les huit premiers symboles sont identiques à celles des huit derniers).

Le corrélateur de réception est implémenté basant sur l'article [16] (figure 3.3). Le corrélateur comporte un registre qui stocke la séquence PN du symbole 0 pour le canal Q. Il peut générer chacune des séquences correspondantes aux 15 autres symboles (en une base similaire au générateur de chips de l'unité de l'émission).

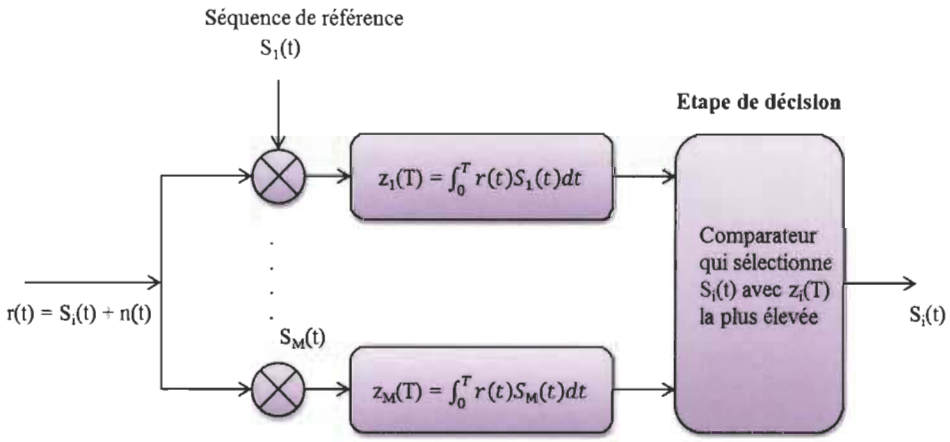


Figure 3.3 : Le corrélateur de réception (adaptée de [16])

Une fois que le corrélateur détecte l'entrée de données, chacune des séquences de référence (qui sont générées par le corrélateur) est multipliée par l'entrée (soit des données binaires. Cette opération peut être réalisée par l'opération AND. Le résultat est accumulé dans 16 registres. Lorsque les 16 bits sont analysés, le registre avec une valeur maximale 8 correspond au symbole à détecter. La valeur de détection est 8, étant donné que chacune des séquences contient le même nombre de « 1 » et de « 0 » et analyse les séquences de 16 bits.

Concernant le bloc de l'indicateur de puissance du signal reçu, nous ne l'étudions pas dans notre travail. Son idée d'implémentation est décrite dans la section 2.2.5 du chapitre 2 et nous pouvons profiter du mode vectoriel du module CORDIC qui nous permet d'obtenir la représentation polaire (module, phase).

3.2.2 Le deuxième travail à faire

Nous décrivons à ce stade, les deux applications (interface graphique) développées sous GUIDE de Matlab qui servent à envoyer des données sur le port série et aussi les recevoir sur le même port.

La première application à concevoir sert à envoyer des données sur le port série, elle comporte 3 panneaux (figure 3.4) :

- Protocole de communication : deux champs paramétrables sélection du port (COM1 … COM9) et vitesse du port (de 1200 jusqu'au 115200 baud), un bouton pour ouvrir le port avec les données des deux champs en entrée et un autre bouton pour fermer le port;
- Configuration des données : les données à configurer sont le symbole à émettre (16 combinaisons de 0000 à 1111), sélection de bande (bande 2.45 GHz ou bande 915 MHz), remise à zéro (activée ou désactivée) et enfin début émission (activé ou désactivé) puis dès que nous terminons, nous appuyons sur le bouton envoyer;
- Affichage des messages : à chaque fois que nous faisions une action sur les boutons, le message change;
- L'activation et la désactivation des boutons nous facilite les étapes pour envoyer une donnée par exemple : nous ne pouvons pas envoyer une donnée sans ouvrir le port;

Figure 3.4 : L'application développée sous GUIDE

Les fonctions dans le code source sur GUIDE de Matlab sert à :

- Ouvrir et fermer le port série;
- Paramétrage du port : les buffers d'entrée et de sortie; la taille; libération des ports;
- Manipulation et mise en forme des données : conversion d'une donnée binaire à une donnée décimale et vice versa, concaténation des données, séparation des données;
- Affichage et sauvegarde des messages;
- Envoi et acquisition d'une donnée sur le port;
- Gestion de l'application.

La deuxième application à concevoir nous permet de recevoir les données. Nous ajoutons, par rapport à l'autre application, un nouveau panneau nommé réception des données qui contient un bouton d'acquisition des données et un autre bouton pour le sauvegarde des données après avoir bien configuré les données (figure 3.5);

Il y a un nouveau champ paramétrable sous le nom de début réception (activé ou désactivé) qui est requis afin de recevoir un symbole;

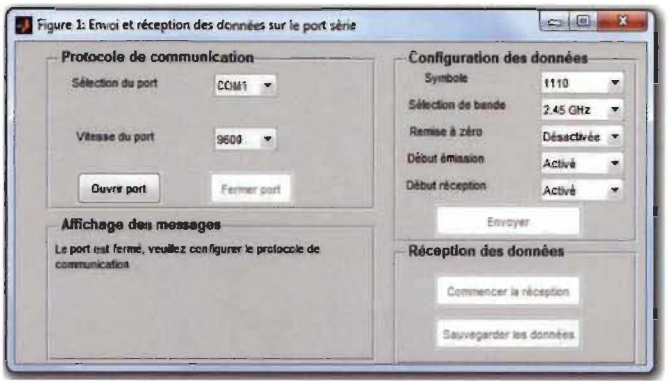


Figure 3.5 : La deuxième application développée sous GUIDE

Cette application a pour but de tester la boucle de retour sur notre chaîne de communication en bande de base.

3.2.3 Le troisième travail à faire

Le troisième travail consiste à, concevoir en VHDL le contrôleur général d'émetteur récepteur en bande de base en basant sur une machine à état fini, et ainsi, d'étudier la trame binaire PDU venant de la couche supérieure. Ici, nous envoyons un flux binaire au lieu des symboles avec un taux de 250 Kbps. Donc, nous ajoutons le bloc conversion bit à symbole et le bloc conversion symbole à bit respectivement pour la chaîne d'émission et la chaîne de réception (figure 3.1).

La chaîne de réception doit être reconçue et adaptée à la carte de la chaîne de transmission analogique RF choisie en termes d'estimation de fréquence, correction de phase et des techniques de synchronisation chip / symbole.

L'émetteur/récepteur est constitué d'un dispositif de commande général, qui gère les parties de l'émission (TX) et de la réception (RX) selon le mode de fonctionnement: arrêt, la réception, la transmission ou la boucle. Les deux parties sont traitées par deux contrôleurs indépendants qui gèrent chacun des composants de transmission et de réception

par activation et désactivation dans le bon ordre. Enfin, un module auxiliaire permet le passage des données de transmission en réception si le mode en boucle est actif (figure 3.6).

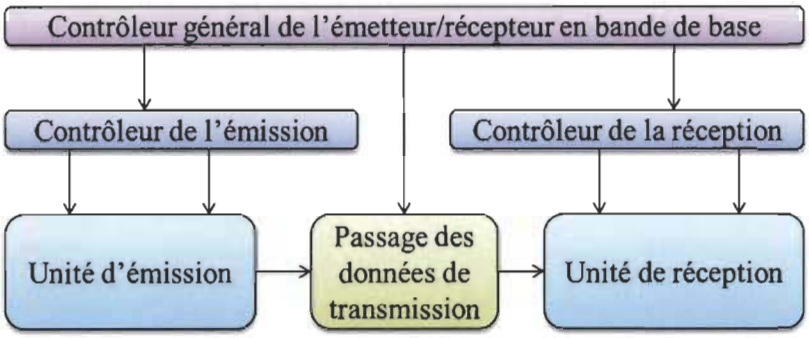


Figure 3.6 : Le contrôl de l'émission et réception

L'émetteur/récepteur peut fonctionner dans les modes suivants (figure 3.7):

- Le mode de transmission TX où la sortie des données envoyées est modulée (c.à.d. la sortie est filtrée).
- Le mode de réception RX où l'entrée est démodulée et nous obtenons les bits envoyés par l'émetteur.
- Le mode d'arrêt (IDLE en anglais) où l'émetteur/récepteur n'est pas en fonctionnement.
- Le mode boucle est un mode de test (boucle de retour) où la sortie de l'unité de l'émission est connectée à l'entrée de l'unité de la réception afin de vérifier que la réception est correcte.

Le contrôleur général de l'émetteur/récepteur vérifie le changement d'un mode à l'autre sans que le processus d'émission ou de réception soit interrompu.

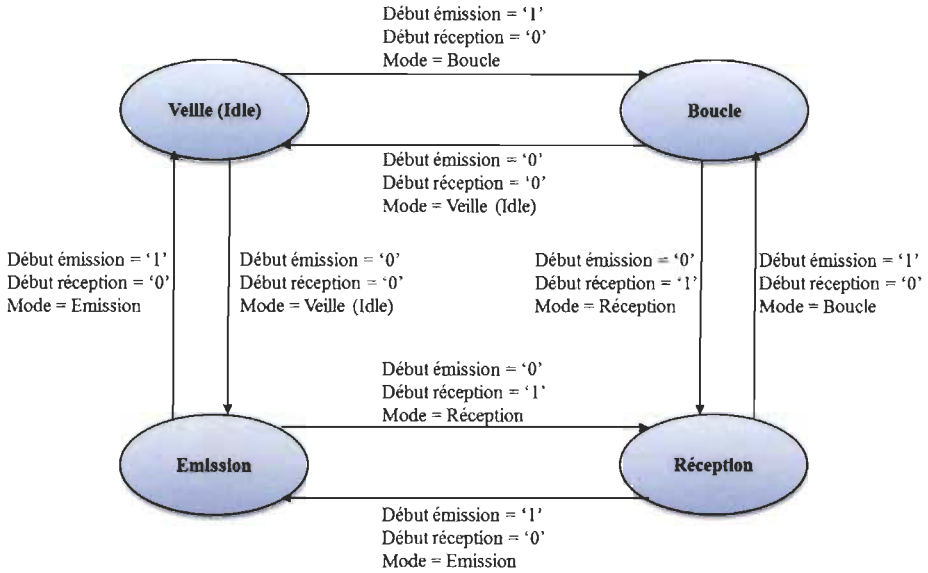


Figure 3.7 : La machine à état finie du contrôleur général

Le contrôleur général décide le passage d'un mode à un autre sans interférer le processus qui est en cours de se dérouler. Le contrôleur reçoit trois entrées, un signal indiquant le début d'émission, un autre indique le début de la réception et un troisième représente le mode à changer. Nous ne pouvons pas changer le mode veille tant que le mode de l'émission est actif ou ni le mode de l'émission tandis que la réception est en exécution.

Lorsque le mode de boucle est activé, les données de l'émetteur passent à l'unité de la réception, Cet étape des données est contrôlée par l'émetteur/récepteur à l'aide de l'unité auxiliaire de l'émetteur/récepteur TX à RX.

Le contrôleur de transmission est chargé de préparer la trame PDU et de la moduler et avant de commencer l'émission des données, il active toutes les parties constituant le processus de l'émission (figure 3.8).

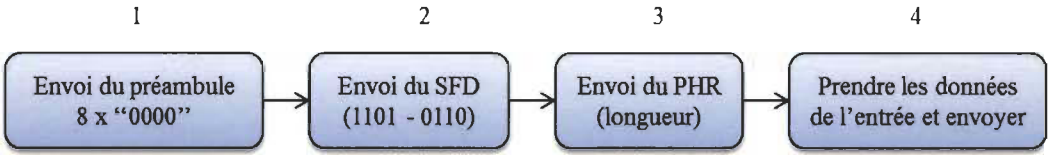


Figure 3.8 : Les étapes du contrôleur de l'émission

Le contrôleur de l'émission prépare l'en-tête de la PDU lors de la réception du signal début de l'émission. En premier lieu, l'envoi du préambule composé de 8 symboles zéro (8 x '0000') est fait l'envoi du SFD et ensuite le PHR, y compris la longueur de la charge utile (Payload). Enfin, la prise de l'entrée binaire des données, nous commençons l'émission. Pendant le processus, divers signaux du débogage sont activés indiquant que le préambule, SFD et l'entête PHR ont été envoyés.

Le contrôleur de réception s'occupe d'activer et de synchroniser chacune des parties impliquées dans le processus de la réception : correction «grossière» et «fine» pour les échantillons entrants, corrélation, sous-échantillonnage, filtrage et conversion aux bits.

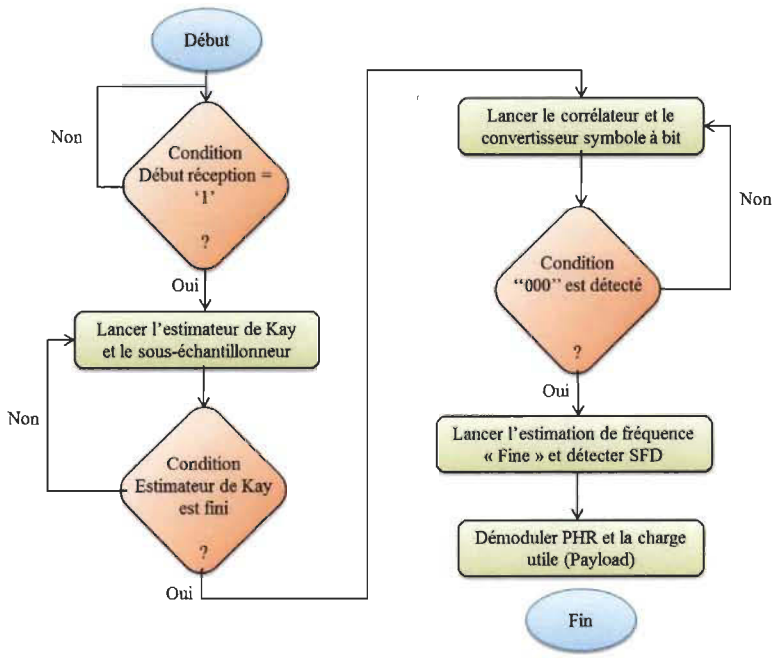


Figure 3.9 : L'organigrame du contrôleur de réception

Le contrôleur de réception (figure 3.9) est activé lorsqu'il y a des nouvelles données à recevoir (signal début réception est actif). Tout d'abord, l'estimateur de fréquence «grossier» est activé (estimateur de Kay). Lorsque l'estimateur de Kay se termine, nous sommes prêts à essayer de détecter un symbole zéro du préambule et de synchroniser avec l'émetteur. À cette étape, le corrélateur et le convertisseur de symbole à bits sont activés. Quand le corrélateur détecte un symbole zéro, il fait signal au contrôleur, et une fois il est entièrement synchronisé avec l'émetteur, l'estimation de fréquence « fine» est activée. Maintenant, l'estimation de fréquence «fine» peut faire une mesure précise, car elle a stocké les phases de données que nous nous attendons à recevoir; et nous pouvons les comparer avec l'entrée. Enfin, il fait signal lorsque le SFD est détecté et il continue à recevoir le champ PHR ainsi que la charge utile (Payload).

L'unité d'émission de la figure 3.1 est responsable de recevoir un train binaire à moduler, avec un taux de 250 kbps. En premier lieu, la conversion de bit à symbole se produit après la conversion à travers une séquence PN. Enfin, les données sont codées en NRZ, elles sont passées à travers un facteur de sur-échantillonneur 7 et elles sont converties en une forme de demi-sinus par le filtre adapté défini par la norme.

La structure comporte :

- La conversion de bit à symbole : convertie l'entrée binaire en un symbole de 4 bits à une vitesse de 250 kbps.
- La conversion de symbole à Chip : chacun des symboles est converti en une séquence PN de 32 Chips (16 Chips par canal) à un taux de 2 Mchips/s.

- Le sur-échantillonnage et le codage NRZ : cette unité insère sept zéros entre chaque bit et converti chaque valeur en NRZ (valeurs 1 et -1), valeurs en complément à 2 codées sur 2 bits.
- Le filtrage adapté : est un filtre FIR à réponse impulsionnelle de demi-sinus définit dans la norme IEEE 802.15.4 pour la bande ISM.

L'unité de réception de la figure 3.1 commence par l'estimation de la fréquence « grossière » en utilisant l'estimateur de Kay. Puis les données sont traitées à la forme originale, les faisant passer par un facteur de sous-échantillonneur 7, un filtre de réception identique à l'émission pour augmenter le SNR, coder les données en RZ et synchroniser la réception de données conjointement par l'estimation « fine » et le corrélateur.

La structure est composé de :

- L'estimateur de la fréquence « grossier » (Kay) : l'estimateur de fréquence «grossier» utilise l'estimateur de Kay. Tout d'abord, l'estimation est faite pour un certain laps de temps au bout duquel les échantillons entrants sont corrigés.
- Le filtre adapté : le filtre de réception augmente le SNR. C'est le même filtre utilisé dans l'émetteur et définit par la norme IEEE 802.15.4.
- Le sous-échantillonneur : sous-échantillonner de facteur 7 pour obtenir des données telles qu'elles étaient à l'origine.
- L'estimation de la fréquence « fine ».
- Le codeur RZ : convertit les données codées en NRZ (1, -1) sur 2 bits en complément à 2 en format RZ (0,1).

- Le corrélateur du symbole : c'est le corrélateur de réception capable de détecter les symboles.
- Le convertisseur du symbole à bit : l'unité convertit les symboles de 4 bits vers le flux binaire d'origine pour un taux de 250 kbps.

Les différents blocs à concevoir en VHDL (pour chaque unité) sont décrits en détails dans l'annexe B.

3.3 Le flot de conception

Le flot de conception à suivre comporte les étapes suivantes :

- La synthèse : la génération du Netlist (schématique RTL et technologie), l'estimation des ressources matérielles (par l'outil ISE Design Suite);
- L'implémentation de l'architecture (par l'outil ISE Design Suite):
  - Placement relatif et estimation du routage (PlanAhead) → Floorplanning
  - Gestion des contraintes des entrées / sorties → fichier « .ucf »;
  - Compilation du circuit : la transformation ou le mappage, la disposition ou le placement, et le routage (PlanAhead);
- Vérification finale :
  - La simulation temporelle du design (création du testbench) : ModelSim;
  - La configuration (BitStream) : la génération, le téléchargement (iMPACT);
  - La validation : vérification réelle soit à travers un analyseur logique (dans notre cas l'oscilloscope) ou un analyseur logique intégré ILA utilisant

Chipscope pro Analyser [30]. Un oscilloscope est peut être utilisé ainsi qu'un analyseur du spectre.

Nous utilisons aussi le logiciel RealTerm pour la capture des données sur le port RS232 et Matlab GUIDE pour le développement de l'application (interface graphique).

3.4 Le choix du matériel

A ce stade, nous toucherons le côté matériel choisi en vue de constituer toute la chaîne de communication. Pour la partie numérique, nous avons adopté une plateforme d'évaluation (XUPV5-LX110T ou l'équivalent ML505) chez Xilinx. Elle embarque un circuit FPGA de la famille Virtex 5 (boitier FF1136) et elle possède plusieurs ports de transfert des données (Série,USB, Ethernet). Elle est cadencée par une horloge de 125 MHz (veuillez consulter l'annexe C). Pour plus de détails sur la configuration matérielle et logicielle, veuillez consulter la référence [24].

La programmation de cette carte ou le téléchargement de l'architecture numérique se fait par le fameux JTAG-USB et la connexion avec le terminal est réalisée via RS232.

Un câble null modem est requis pour connecter la plateforme au port série d'un ordinateur parce que le port série RS-232 mâle DB-9 (veuillez consulter l'annexe C) est câblé comme un dispositif hôte (DCE), et ça nécessite aussi un convertisseur USB-RS232 pour travailler avec le port série.

Un équipement de test externe pour le débogage dans le circuit FPGA tel qu'un oscilloscope à domaines mixtes ou analyseur logique (dans notre cas: MDO4034B-3 avec la sonde de 16 bits);

Nous avons pensé à mettre un banc de test réel pour le transmetteur vu que dans le laboratoire hyperfréquence et RF du centre de recherche C2T3 nous disposons:

- ✓ Le double convertisseur numérique-analogique de 10 bits (pour I et Q) avec un taux de 125 MEPS, il dispose de deux filtres de réjection d'horloge de type Butterworth à 6 pôles et il est caractérisé par : bande passante maximale: +/- 13 MHz à ±0.4dB ondulation, rejet d'horloge d'échantillonnage à 40 MHz > 84 dBc, tension analogique de sortie (connecteurs SMA femelles): 1 $V_{crête\text{-}à\text{-}crête}$ avec 0.85V DC bias, il est de type unipolaire c.-à-d. il accepte des valeurs numériques en format non signé (donc les valeurs doivent être entre 0 et 1023) et enfin une alimentation de 5 V, pour plus des détails, veuillez consulter la référence [32];
- ✓ Le générateur du signal RF AT-N9310A de 9kHz-3GHz (option: 001 Add Analog IQ input capability) : il travaille comme un modulateur en I/Q (y compris QPSK), il dispose de deux entrées analogiques (avec une tension pleine échelle de 0.5 V_{rms} c.à.d. 1.4142 $V_{crête\text{-}à\text{-}crête}$ au maximum ou autrement un signal analogique d'amplitude 1 V) et une sortie RF, pour plus des détails, veuillez consulter la référence [31];
- ✓ L'analyseur du signal EXA AT-N9010AEP (il visualise le spectre du signal RF) avec l'option: 004 EXA express – fréquence 26.5 GHz, préamplificateur 7.0 GHz et un atténuateur électronique 3.6 GHz, pour plus des détails, veuillez consulter la référence [33];

3.5 Les tests réels à faire

Afin de valider le fonctionnement de la chaîne, nous faisons une série de test. Il y a une façon de vérifier si nos sorties I et Q en émission sont conformes à la norme, le fait de respecter la période du bit de données (4 µs pour une fréquence de 250 kHz), la période du symbole (16 µs pour une fréquence de 1 MHz par voie) et la période d'un chip (1 µs) en plus le respect du décalage entre I et Q pour appliquer la modulation O-QPSK (0.5 µs) et enfin la vérification de la séquence d'étalement chip de 32 bits (16 chips par voie) suivant le symbole envoyé à travers le tableau 2-2 dans le chapitre 2 pour le mappage de symbole vers chip.

Nous faisons un test de l'application d'envoi d'une donnée passant par les modules implémentés sur FPGA (module UART et module séparation des données) ainsi en émission de la donnée. Nous vérifions les sorties qui sont connectées à l'oscilloscope par l'intermédiaire d'une sonde de 16 pins (comme le montre la figure 3.11). L'image de la figure suivante montre comment analyser les signaux numériques sur l'analyseur logique intégré et l'oscilloscope multi-domaine.

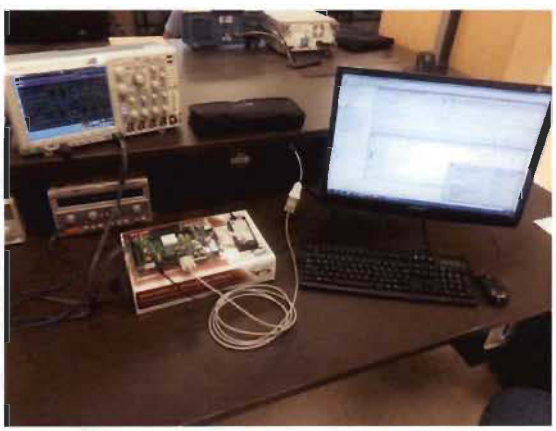


Figure 3.10 : Premier essai avec l'analyseur logique

Donc nous vérifions les signaux visualisés par l'oscilloscope MDO4034B-3 (existant à l'université) à chaque fois que nous faisons une modification sur les données à envoyer par l'interface graphique. La visualisation des résultats se fait sur l'application d'envoi, sur l'outil Chipscope Pro à travers l'analyseur logique intégré et sur l'oscilloscope.

Lorsque nous parlons d'un émetteur / récepteur, nous faisons toujours un test de la boucle de retour (deuxième test). L'idée est d'envoyer une donnée par l'application et nous vérifions si nous avons la même donnée à la réception de l'interface graphique.

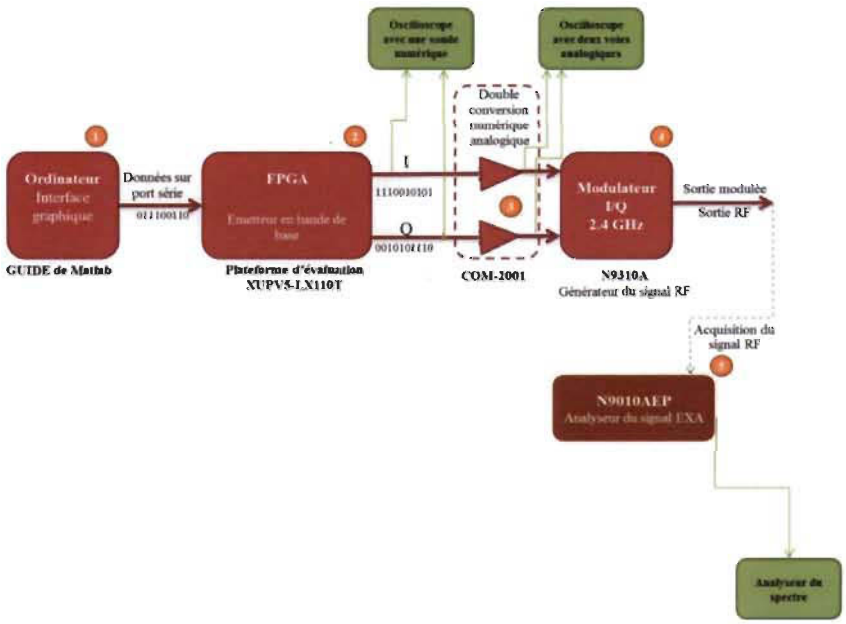


Figure 3.11 : Banc de test global

Comme le montre la figure 3.11, nous allons mettre un banc de test global et nous allons vérifier l'acheminement des données. Mais, avant de commencer à vérifier les résultats, nous procédons à la configuration du matériel comme suit :

- Pour la plateforme d'évaluation, nous fixons la force de conduction maximale (24 mA) de nos sorties globales (I et Q ainsi que le signal d'horloge de 8 MHz) au niveau d'assignement des pins, en plus, les blocs internes de la chaîne sont

synchronisés sur le front descendant de l'horloge afin d'assurer la stabilité des bits. La carte est configurée pour donner une tension maximale sur chaque pin de sortie 3.3 V afin de représenter un « 1 logique »;

- Pour le convertisseur numérique-analogique (il requiert une tension d'alimentation unipolaire +5V à partir du bloc d'alimentation Agilent E3631A [37]). Nous tenons compte de la masse commune avec la plateforme d'évaluation et puisqu'il est de type unipolaire donc nous devons adapter les valeurs numériques (8 coefficients du filtre pour représenter un demi-sinus) des sorties I et Q (sachant qu'elles sont signées) par des valeurs non signées entre 0000000000 (0 en décimal) et 1111111111 (1023 en décimal), tel que l'origine est 0111111111 (511 en décimal), et que la sortie du convertisseur est inversée. Cela dit, il considère 0 décimal comme valeur maximale du sinus et aussi 1023 décimal représente la valeur minimale du sinus. Donc nous appliquons la formule suivante pour trouver les seize valeurs : $512 \times \sin((n \times \pi) / 8) + 511$; n=0..15. La plage de tension pour les entrées numériques est de 0-3.3V (qui est la même sur les pins de la carte d'évaluation).

  N.B : suivant l'architecture interne de la carte du convertisseur (la structure interne fournit un signal de sortie en courant), il est suivi par un circuit de conversion courant-tension pour fournir en sortie une tension comprise entre 0 et 1.65 V. Après nous trouvons un filtre anti-repliement passe-bas Butterworth (en cascade à 6 pôles et de type single-ended) à base des amplificateurs opérationnels avec une fréquence de coupure de 13 MHz. Suivant la formule qui

nous permet de calculer le recouvrement de ce filtre, la relation entre la fréquence d'échantillonnage F_e et la fréquence de coupure F_c est :

$F_e = [1 + [(1/0.00707)^2]^{1/12}] \times F_c$ donc pour $F_c = 13$ MHz → $F_e \sim 40$ MHz

- Pour le générateur du signal RF, nous configurons la modulation en I/Q externe avec une porteuse de 2.405 GHz et une puissance du signal de – 3 dBm (raccord avec le convertisseur par un câble BNC-SMA);

3.6 Conclusion

Au long de ce chapitre, nous avons énuméré les spécifications et les besoins techniques pour la réalisation de la chaîne de communication. Nous avons détaillé notre méthodologie étudiée dans le but de faciliter la compréhension. Nous allons voir dans le chapitre qui suit, toute la contribution liée à la chaîne que ce soit à travers l'étude ou la conception en terminant par la visualisation des différents résultats obtenus.

Chapitre 4 - Résultats de simulation et de test

4.1 Introduction

Dans ce chapitre, nous nous intéressons à la présentation des différentes simulations effectuées en suivant le flot de conception décrit dans le chapitre précédent. Ainsi, nous visualisons les résultats obtenus par les tests réels sachant qu'il y a une partie des travaux supplémentaires dans l'annexe D.

4.2 Conception d'un émetteur en bande de base

A ce stade, nous commençons à présenter seulement les résultats de la conception de la chaîne d'émission en bande de base (sans et avec diviseur d'horloge).

4.2.1 Emetteur en bande de base sans diviseur d'horloge

Ici, nous implémentons le composant chaîne émission de la figure 3.1. Nous allons le tester sous ModelSim et sous l'outil Xilinx ISE.

Supposant que notre message à émettre : [110010010101011110111010] ; Avec : S_1 = 1100, S_2 = 1001, S_3 = 0101, S_4 = 0111, S_5 = 1011, S_6 = 1010 sont les symboles à émettre sous forme de 4 bits, ils sont envoyés sur les deux canaux (en phase I et en quadrature de phase Q) afin de les moduler par le schéma de modulation O-QPSK (consultez le deuxième chapitre).

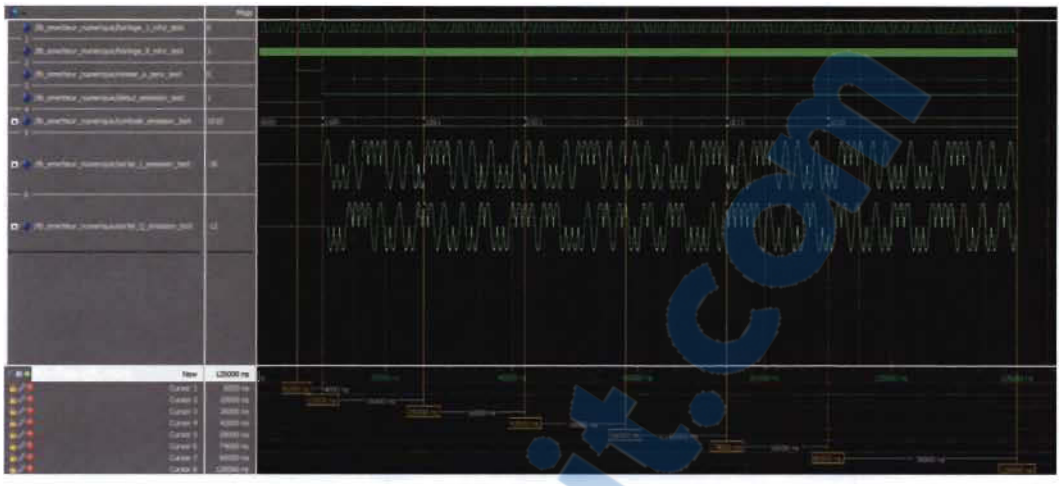


Figure 4.1 : Simulation de l'émetteur sous ModelSim

Comme le montre la figure 4.1, nous avons une remise à zéro pendant une période de 6 microsecondes, après la désactivation de cette entrée pour une fraction de 4 microsecondes. Puis, nous commençons d'envoyer le premier symbole (sur 4 bits) '1100' avec l'activation de l'émission pour une durée de 16 microsecondes. Ensuite, nous avons un deuxième symbole '1001' pendant la même période (16 microsecondes), après un troisième symbole qui arrive '0101' (pour 16 microsecondes), le symbole qui suit est '0111' (pour 16 microsecondes), puis un autre symbole '1011' (pour 16 microsecondes) et enfin le dernier symbole est '1010' (pour 16 microsecondes). Le temps de simulation sous ModelSim est 120 microsecondes.

Tableau 4-1 : Ressources occupées de l'émetteur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	92	69120	0 %
Nombre de LUTs	175	69120	0 %
Nombre de paires LUT-FF	39	228	17 %
Nombre de blocs E/S	24	640	3 %
Nombre de BUFGs/BUFGCTRS	2	32	6 %

Le tableau 4-1 récapitule les ressources occupées par l'émetteur en bande de base sans diviseur d'horloge. Nous voyons bien qu'il occupe les entrées / sorties du circuit (deux entrées d'horloge de 1 MHz et 8 MHz, une entrée remise à zéro, une entrée début émission, quatre entrées pour symbole émission et seize sorties pour la sortie sur la voie I et pour la sortie sur la voie Q).

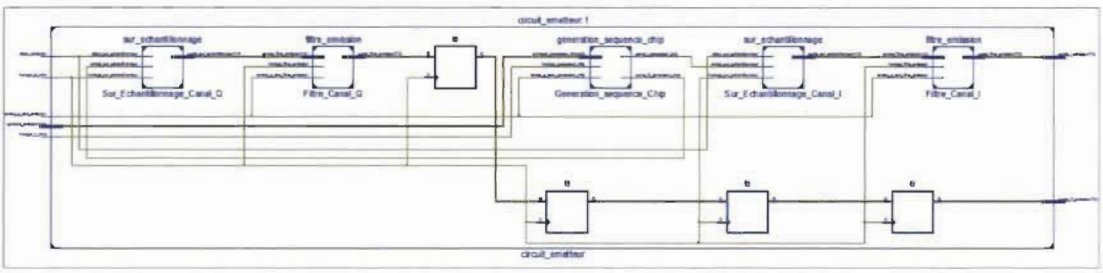


Figure 4.2 : Le schématique de l'émetteur en bande de base

La figure 4.2 représente notre circuit sous forme d'un schématique obtenu au niveau de la synthèse utilisant l'outil Xilinx ISE. Nous voyons bien les différents blocs inclus dans l'architecture.

4.2.2 Emetteur en bande de base avec diviseur d'horloge

La nécessité de générer les deux horloges (1 MHz et 8 MHz), à partir de l'horloge FPGA à 200 MHz (c'est l'horloge globale utilisée durant tous les tests) est pour faire fonctionner notre circuit (test réel sur Chipscope Pro et sur l'oscilloscope). Dans l'annexe D, vous trouverez la partie consacrée aux résultats de conception du diviseur d'horloge.

Dans ce qui suit, nous faisons la même chose que précédemment en ajoutant le diviseur d'horloge à notre émetteur afin de tester le circuit sur Chipscope Pro.

La figure 4.3 représente la simulation sous ModelSim de l'émetteur en bande de base en tenant compte du diviseur d'horloge. Ce résultat nous montre le bon fonctionnement de ce module.

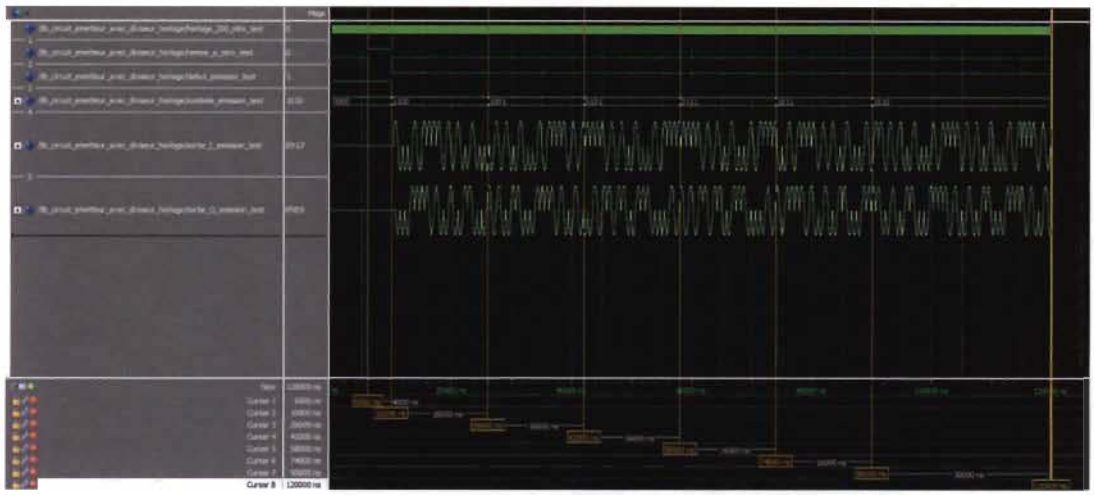


Figure 4.3 : Simulation de l'émetteur avec diviseur sous ModelSim

Le tableau 4-2 contient l'estimation des ressources occupées par l'émetteur en bande de base avec l'ajout de l'entrée d'horloge de 200 MHz à la place de deux horloges précédentes (1 MHz et 8 MHz).

Tableau 4-2 : Ressources occupées de l'émetteur avec le diviseur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	134	69120	0 %
Nombre de LUTs	247	69120	0 %
Nombre de paires LUT-FF	79	302	26 %
Nombre de blocs E/S	23	640	3 %
Nombre de BUFGs/BUFGCTRS	3	32	9 %

La figure 4.4 illustre le schéma du circuit d'émetteur implémenté en VHDL qui est connecté avec le diviseur d'horloge.

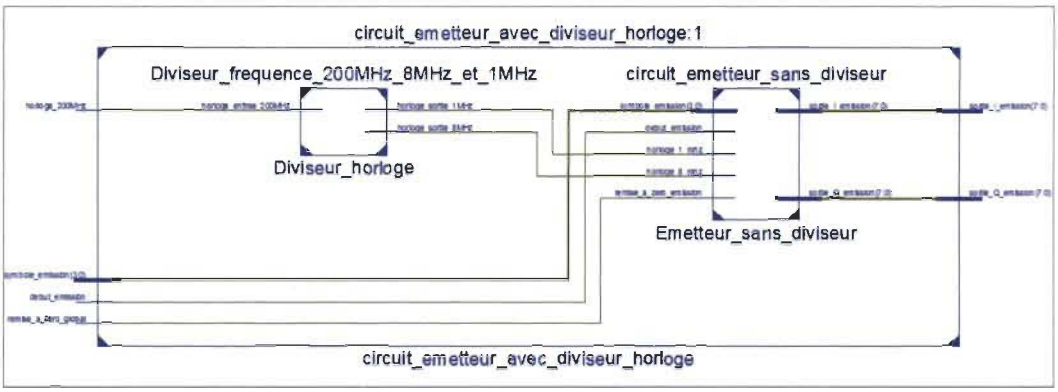


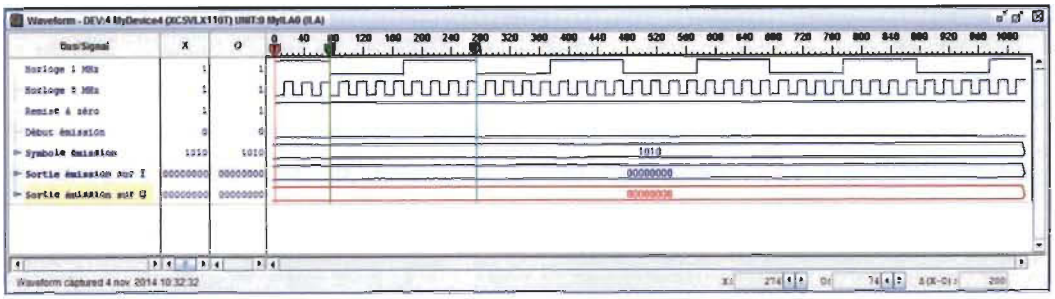
Figure 4.4 : Le schématique de l'émetteur avec le diviseur

Le tableau 4-3 comporte les ports de détection (ou Trigger ports en anglais) qui sont introduits durant la configuration du fichier de connexion (ajouté au projet sous Xilinx ISE) afin de tester les entrées sorties du circuit à implémenter sous Chipscope Pro. Il est obligé de choisir une seule horloge globale qui est dans notre cas l'horloge 200 MHz (celle du FPGA).

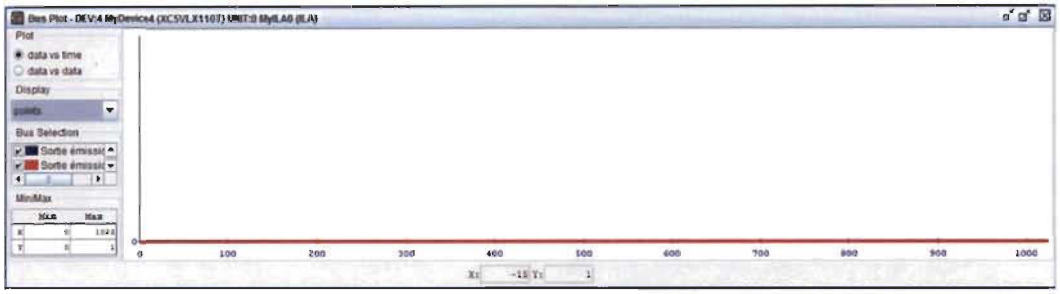
Tableau 4-3 : Les ports à détecter pour l'émetteur avec diviseur

Entrée / sortie	Port
Entrée horloge 200 MHz	Port horloge
Sortie horloge 1 MHz	Port détection 0
Sortie horloge 8 MHz	Port détection 1
Entrée remise à zéro	Port détection 2
Entrée début émission	Port détection 3
Entrée symbole émission [0 à 3]	Ports détection 4
Sortie sur la voie I [0 à 7]	Ports détection 5
Sortie sur la voie Q [0 à 7]	Ports détection 6

La figure 4.5 présente la simulation des différentes entrées / sorties de l'émetteur en bande de base. Nous avons fait une comparaison des échantillons des sorties sur la voie I et la voie Q avec ceux qui sont prélevés du simulateur ModelSim. Nous avons obtenu les mêmes résultats pour le même symbole envoyé.



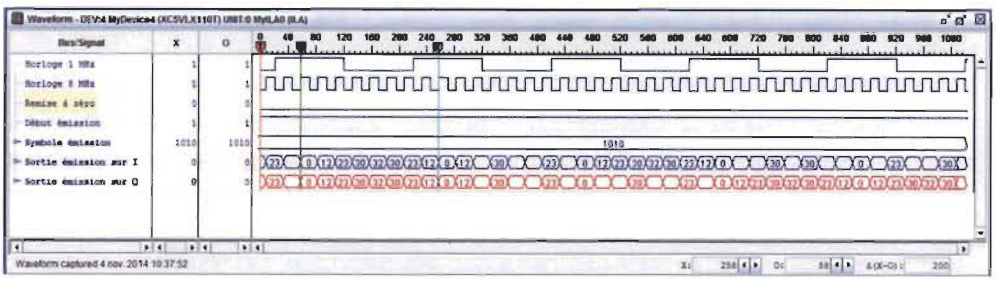
(a) La forme d'onde



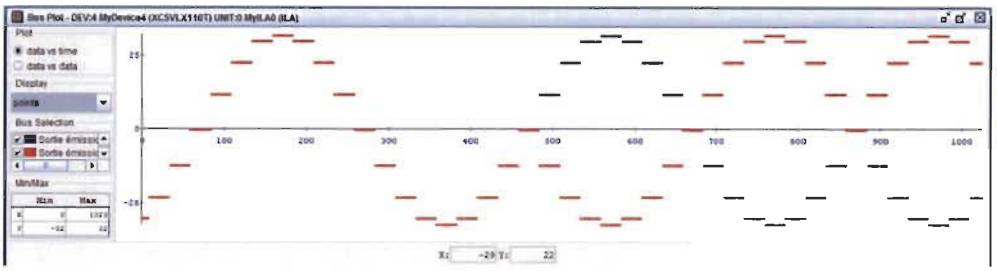
(b) Tracé du bus

Figure 4.5 : Lors d'une remise à zéro

Nous remarquons bien (figure 4.5) lors d'une remise à zéro les sorties sont toujours à zéro que ce soit pour la forme d'onde (en (a)) ou même le tracé du bus (en (b)) avec un affichage en point.



(a) La forme d'onde



(b) Tracé du bus

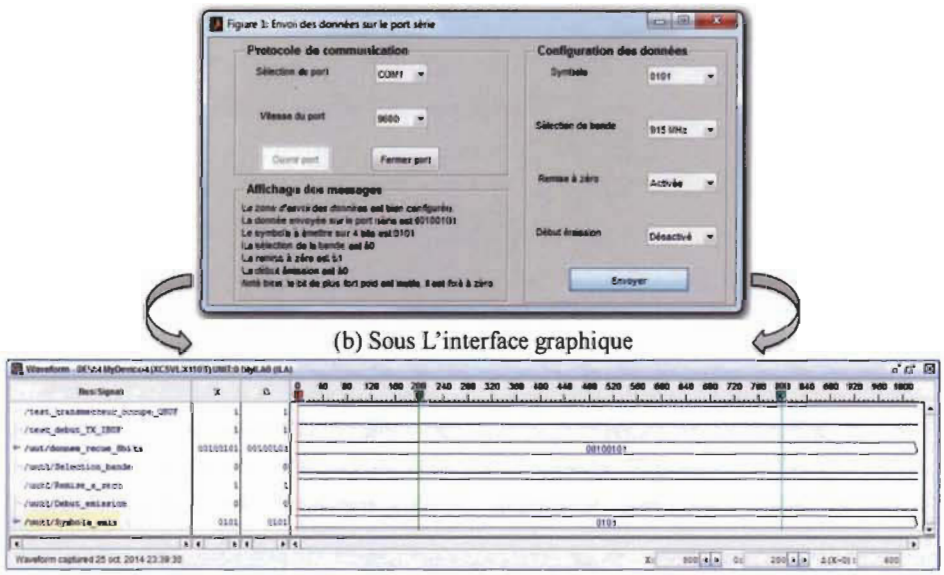
Figure 4.6 : Lors de l'envoi d'un symbole

D'après la figure 4.6, nous voyons bien le bon fonctionnement du circuit implémenté, dans (a) les sorties sont affichées en nombre décimal signés pour pouvoir les représenter au tracé du bus (en (b)) tel que le signal en bleu représente la sortie sur I émission et ce qui est en rouge correspond à la sortie sur Q en émission (l'affichage est en mode point) et tout le tracé comprend 1024 points.

Dans cette partie, les entrées sont introduites par les Dip Switchs de la carte pour le test sous Chipscope Pro et nous n'avons pas fait le test sur les LEDs à cause d'insuffisance des LEDs afin de prendre en compte toutes les sorties du circuit (seize sorties).

Dans l'annexe D, vous trouverez les résultats de conception d'un émetteur à double bande.

Nous faisons un test de l'application d'envoi d'une donnée passant par les deux modules implémentés sur FPGA (module UART et module séparation des données) ainsi nous vérifions les sorties qui sont connectées aux LEDs utilisateurs sur la plateforme.



(b) Sous L'interface graphique

(b) Sous Chipscope Pro

Figure 4.7 : Test d'envoi de la donnée

D'après ce test (figure 4.7) nous sommes sûr que l'application fonctionne très bien en comparant les données envoyées par l'application (zone affichage des messages en (a)) avec les données reçues à l'intérieur de FPGA (suivant l'analyseur logique intégré en (b)) et même nous avons vérifié le changement d'état sur les LEDs utilisateurs.

4.3 Conception d'un émetteur / récepteur en bande de base

Ici, nous implémentons le circuit émetteur récepteur de la figure 3.2. Nous allons le tester sous ModelSim et sous l'outil Xilinx ISE.

4.3.1 Émetteur-récepteur sans connexion de deux chaînes

La première tâche à accomplir est de décrire l'émetteur récepteur en bande de base sans faire la connexion entre la chaîne d'émission et la chaîne de réception. Dans la figure 3.1, l'entrée sur I réception et la sortie sur I émission ne sont pas connectées et aussi l'entrée sur Q réception et la sortie sur Q émission ne sont pas connectées;

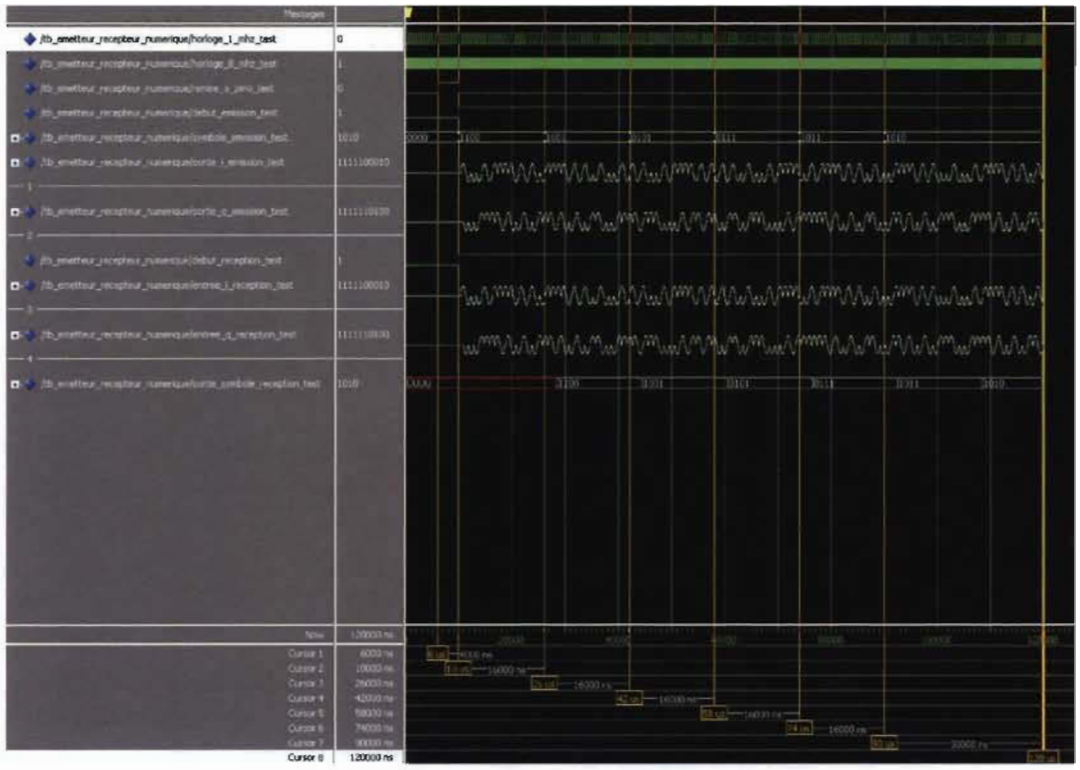


Figure 4.8 : Simulation de l'émetteur récepteur sous ModelSim

La figure 4.8 nous confirme le bon fonctionnement de l'émetteur récepteur en bande de base à travers l'envoi de six symboles à l'émission et nous avons reçu les mêmes symboles (les deux chaînes sont connectées durant le test).

Tableau 4-4 : Ressources occupées de l'émetteur / récepteur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	377	69120	0 %
Nombre de LUTs	1040	69120	1 %
Nombre de paires LUT-FF	77	1340	5 %
Nombre de blocs E/S	37	640	5 %
Nombre de BUFGs/BUFGCTRS	2	32	6 %
Nombre des DSP48Es	7	64	10 %

Le tableau 4-4 rassemble les ressources occupées par l'émetteur récepteur en bande de base. Nous voyons bien que notre circuit dispose beaucoup des entrées / sorties (37 blocs) en plus, il consomme plus de mémoire.

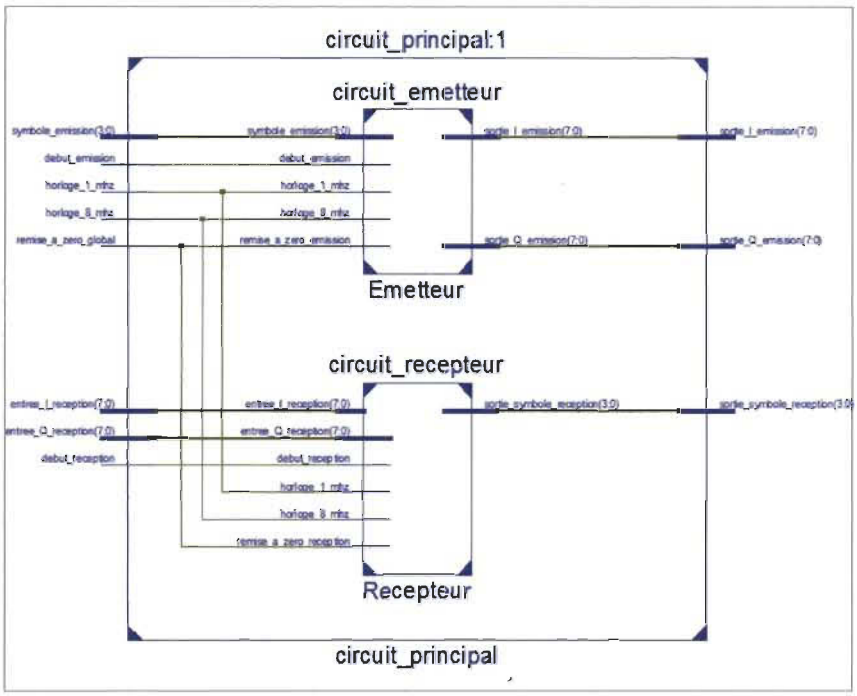


Figure 4.9 : Le schématique de l'émetteur récepteur en bande de base

La figure 4.9 reproduit le schéma global de l'émetteur récepteur en bande de base sans connecter les deux chaînes.

4.3.2 Avec connexion de deux chaînes et avec le diviseur d'horloge

Dans cette sous-section, nous faisons la connexion de deux chaînes et nous ajoutons le diviseur d'horloge afin de tester l'architecture sur l'outil Chipscope Pro.

La figure 4.10 représente la simulation de l'émetteur / récepteur en bande de base en tenant compte de la connexion entre les deux chaînes et aussi en le connectant avec un

diviseur d'horloge. A la réception, nous avons reçu les mêmes symboles transmis par l'émetteur ce qui nous prouve le bon fonctionnement.

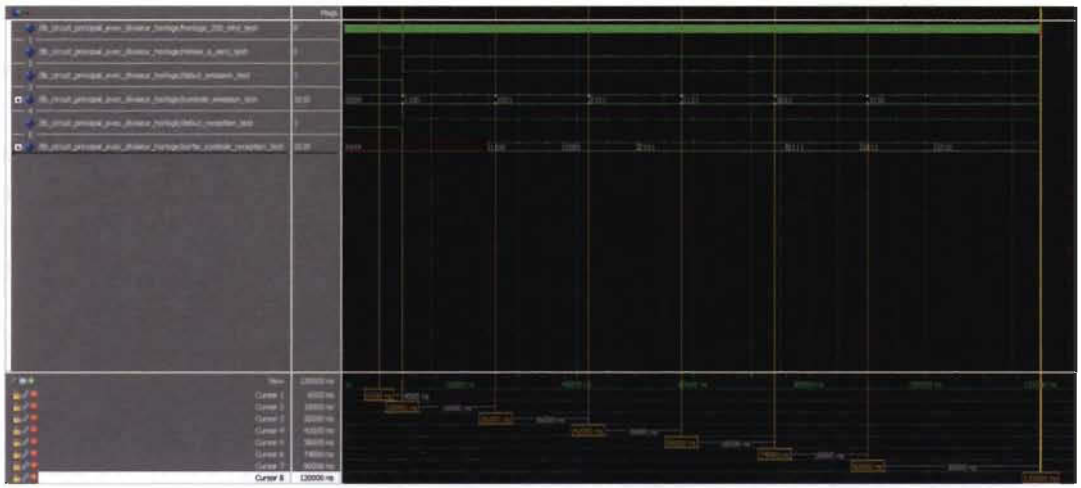


Figure 4.10 : Simulation de l'E/R avec connexion et avec diviseur sous ModelSim

Le tableau 4-5 contient les ports de détection qui sont entrés durant la configuration du fichier de connexion (ajouté au projet sous Xilinx ISE) afin de tester les entrées sorties du circuit à implémenter sous Chipscope Pro. Nous avons choisi l'horloge globale de 200 MHz (celle de l'FPGA). Nos entrées sont placées sur les pins de GPIO Dip switch et nos sorties sont connectées aux pins de GPIO LED.

Pour plus des détails, consultez le fichier des contraintes d'utilisateur de notre plateforme d'évaluation [29].

Tableau 4-5 : Test des ports sur la carte

Port	Localisation	Bank	Fonction
Horloge global	L19	4	CLK 200 MHz FPGA
Remise à zéro	U25	15	GPIO Dip switch 1
Début émission	AG27	21	GPIO Dip switch 2
Début réception	AF25	21	GPIO Dip switch 3
Symbole émission [0]	AF26	21	GPIO Dip switch 4
Symbole émission [1]	AE27	21	GPIO Dip switch 5
Symbole émission [2]	AE26	21	GPIO Dip switch 6
Symbole émission [3]	AC25	21	GPIO Dip switch 7
Symbole réception [0]	H18	3	GPIO Led 0
Symbole réception [1]	L18	3	GPIO Led 1
Symbole réception [2]	G15	3	GPIO Led 2
Symbole réception [3]	AD26	21	GPIO Led 3

Le tableau 4-6 résume les ressources occupées de l'émetteur récepteur en bande de base avec la connexion et avec le diviseur d'horloge. Nous avons ajouté l'entrée d'horloge de 200 MHz à la place de deux horloges précédentes (1 MHz et 8 MHz). Pour cela, le circuit possède 12 blocs d'entrée / sortie.

Tableau 4-6 : Ressources occupées de l'E/R avec connexion et avec diviseur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	377	69120	0 %
Nombre de LUTs	1039	69120	1 %
Nombre de paires LUT-FF	104	1312	7 %
Nombre de blocs E/S	12	640	1 %
Nombre de BUFGs/BUFGCTRS	3	32	9 %
Nombre des DSP48Es	7	64	10 %

La figure 4.11 représente le schématique du circuit d'émetteur-récepteur en bande de base implémenté en VHDL qui est connecté avec le diviseur d'horloge.

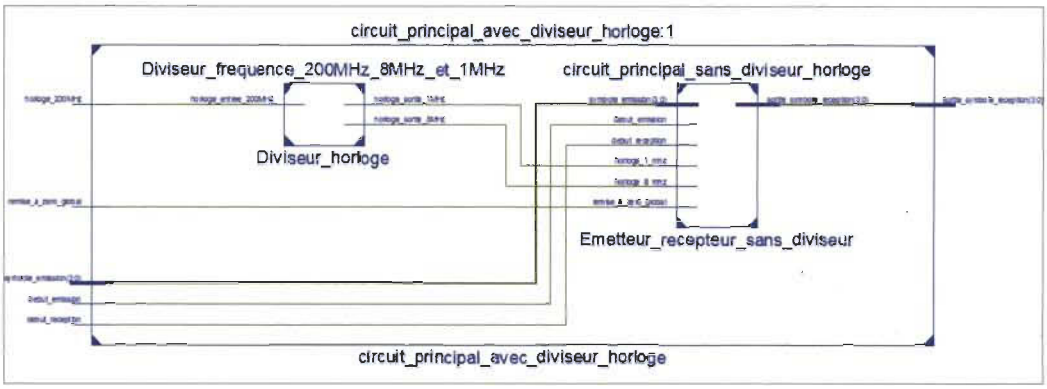


Figure 4.11 : Le schématique de l'E/R avec connexion et avec diviseur

Le tableau 4-7 dresse les ports de détection qui sont introduits durant la configuration du fichier de connexion (ajouté au projet sous Xilinx ISE) afin de tester les entrées / sorties du circuit à implémenter sous Chipscope Pro. Ce fichier comporte la connexion avec le port d'horloge globale (200 MHz) et aussi sept ports de détection dans le but de les afficher durant la simulation réelle.

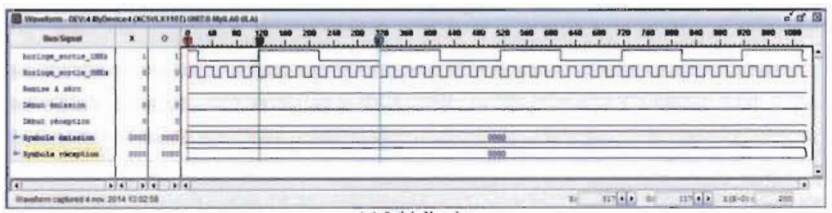
Tableau 4-7 : Les ports à détecter pour l'E/R avec diviseur

Entrée / sortie	Port
Entrée horloge 200 MHz	Port horloge
Sortie horloge 1 MHz	Port détection 0
Sortie horloge 8 MHz	Port détection 1
Entrée remise à zéro	Port détection 2
Entrée début émission	Port détection 3
Entrée début réception	Ports détection 4
Entrée symbole émission [0 à 3]	Ports détection 5
Sortie symbole réception [0 à 3]	Ports détection 6

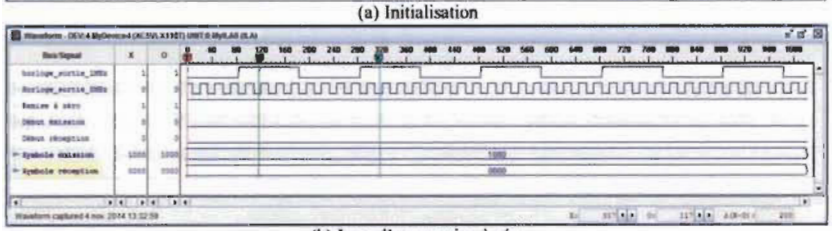
La figure 4.12 nous montre le test sur les entrées de l'émetteur récepteur en bande de base. En effet, nous avons initialisé nos entrées et nos sorties dans le simulateur (en (a)),

après, nous avons activé l'entrée remise à zéro pour voir son influence (en (b)) en jouant sur Dip switch 1. Enfin, nous avons entré notre symbole (1000) à émettre en fixant les Dip switches correspondants sur la carte et en désactivant la remise à zéro donc en (c). L'envoi et la réception du premier symbole (1100), et en (d), l'envoi et la réception du deuxième symbole (1111).

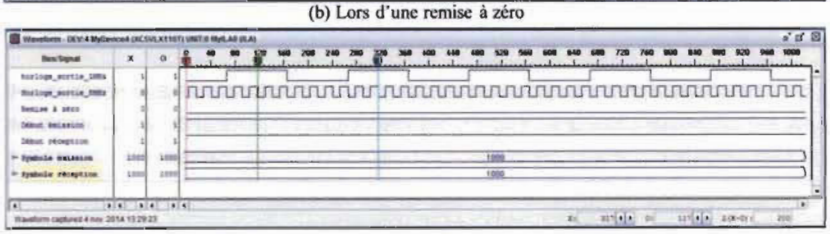
Ici, nous pouvons transmettre n'importe quel symbole en l'introduisant sur les Dip switches adéquats et nous obtenons le même symbole à la réception en le vérifiant à travers les LEDs. Bien sûr, il est nécessaire d'activer les switches liés aux entrées début émission et début réception.



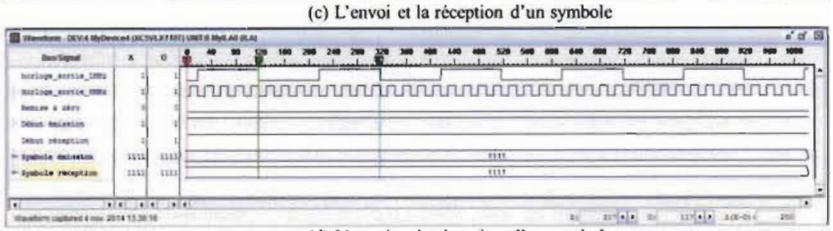
(a) Initialisation



(b) Lors d'une remise à zéro



(c) L'envoi et la réception d'un symbole



(d) L'envoi et la réception d'un symbole

Figure 4.12 : Les contrôles sur les entrées

4.3.3 Emetteur / récepteur en bande de base à double bande

A ce stade, nous allons présenter les résultats obtenus au niveau de la conception d'un émetteur / récepteur en bande de base avec la sélection de la bande, sachant que les deux chaînes (émission et réception) sont connectées et aussi le diviseur d'horloge est implémenté pour que nous puissions les tester sous l'outil Chipscope Pro et même à travers les switches et les LEDs sur la carte. L'architecture de la figure 4.13 est la même que celle décrite précédemment et elle est capable de supporter les deux bandes de fréquences (bande 915 MHz et bande 2.45 GHz).

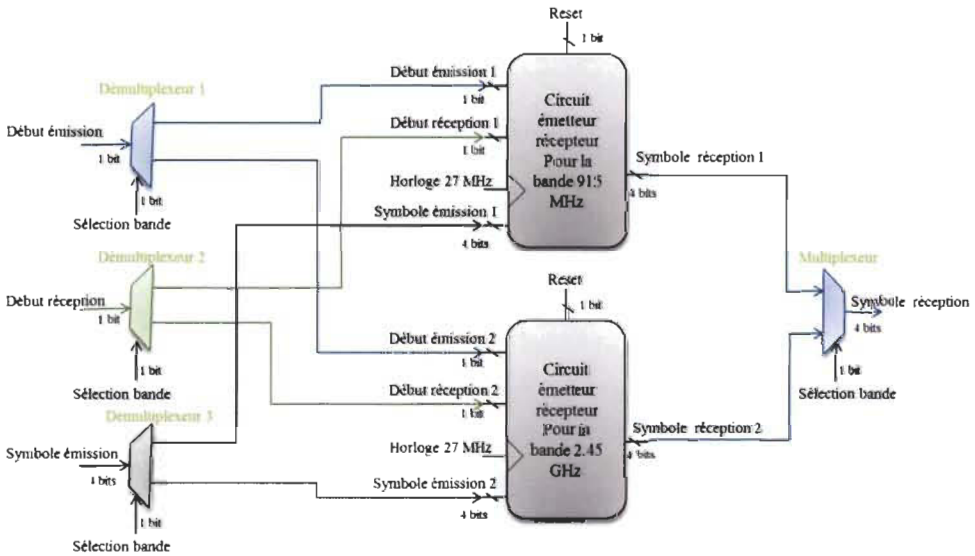


Figure 4.13 : Architecture de deux E/R avec sélection de la bande utilisée

Dans la figure, nous avons un multiplexeur 2 vers 1 (sur 4 bits) afin de sélectionner un seul symbole réception parmi les deux, un démultiplexeur 1 vers 2 (sur 1 bit) sert à choisir une seule entrée début émission, un deuxième démultiplexeur 1 vers 2 (sur 1 bit) pour sélectionner une seule entrée début réception, et enfin, un dernier démultiplexeur 1 vers 2 (sur 4 bits) afin de choisir une seule entrée symbole émission parmi les deux. L'entrée

sélection bande, qui va passer soit la bande 915 MHz ou la bande 2.45 GHz, est commune pour tous les démultiplexeurs et le multiplexeur.

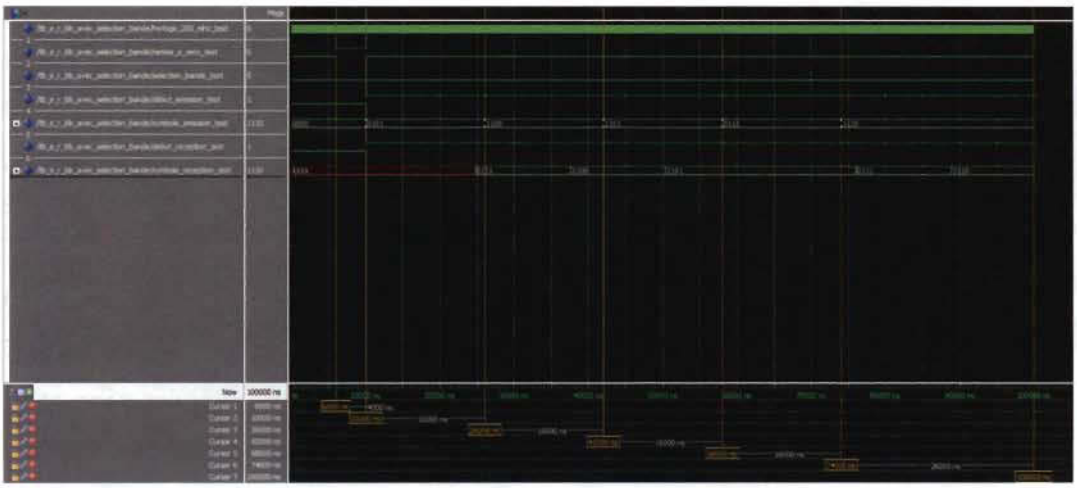


Figure 4.14 : Simulation de deux E/R avec sélection de la première bande

La figure 4.14 représente la simulation sous ModelSim de deux émetteurs / récepteurs avec la sélection de la première bande (bande 915 MHz). Nous avons bien reçu les symboles émis.

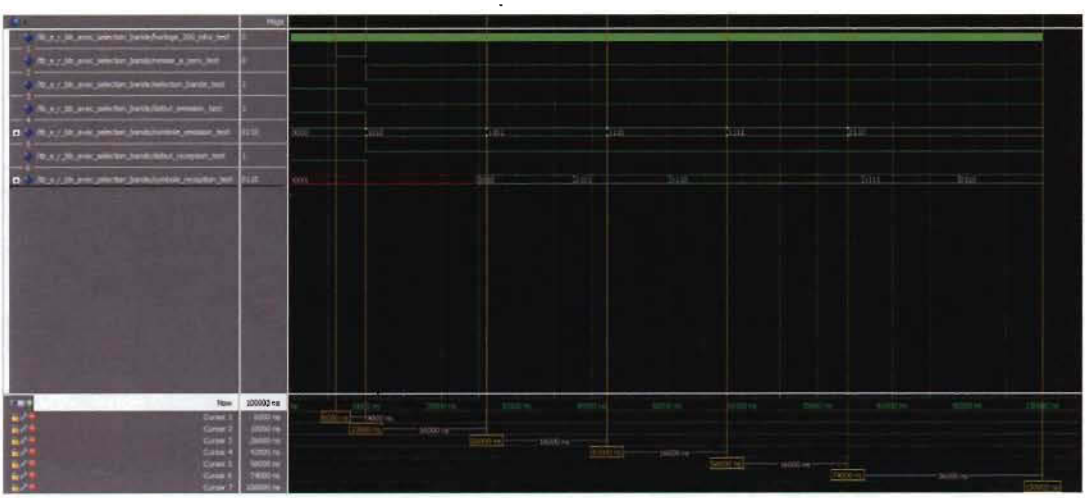


Figure 4.15 : Simulation de deux E/R avec sélection de la deuxième bande

La figure 4.15 contient la simulation sous ModelSim de deux émetteurs / récepteurs avec la sélection de la deuxième bande (bande 2.45 GHz). Nous avons bien reçu les symboles transmis.

Tableau 4-8 : Ressources occupées de deux E/R avec sélection de la bande

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	724	69120	1 %
Nombre de LUTs	2071	69120	2 %
Nombre de paires LUT-FF	164	2631	6 %
Nombre de blocs E/S	13	640	2 %
Nombre de BUFGs/BUFGCTRS	4	32	12 %
Nombre des DSP48Es	14	64	21 %

Le tableau 4-8 récapitule les ressources occupées de deux émetteurs / récepteurs avec la sélection de la bande. D'après l'estimation, nous avons 13 blocs d'entrée / sortie qui sont suffisantes pour le test sur la carte.

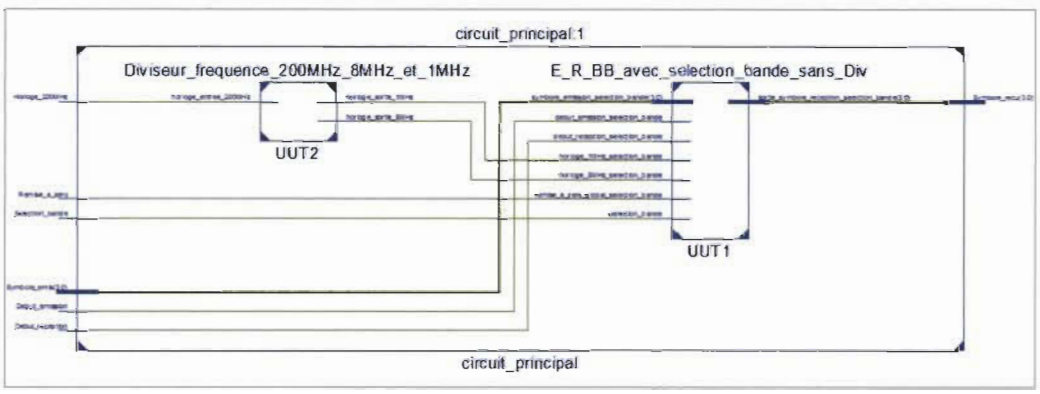


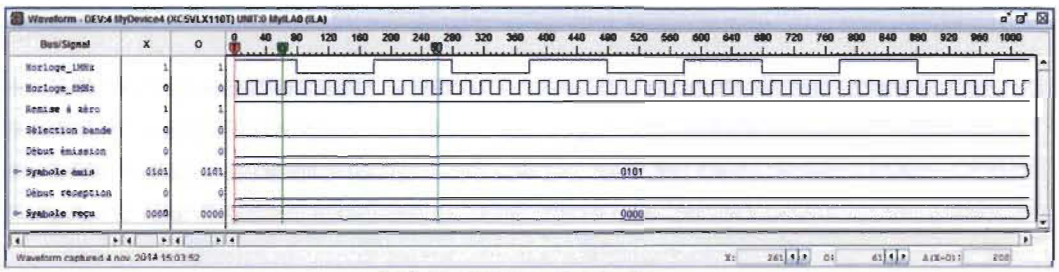
Figure 4.16 : Le schématique de deux E/R avec sélection de la bande

La figure 4.16 décrit le schématique de deux émetteurs / récepteurs en bande de base avec la sélection de la bande y compris le diviseur d'horloge.

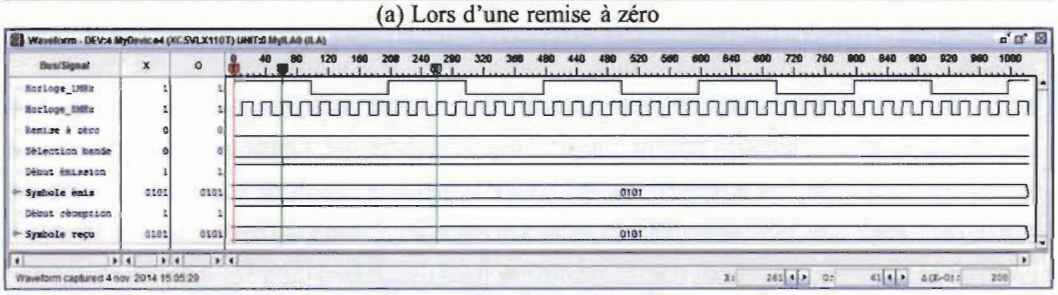
Au niveau du test, nous avons configuré les différents ports de détection durant l'ajout du fichier de connexion au projet Xilinx ISE. Nous avons ajouté au tableau 4-7 (des entrées

/ sorties) l'entrée « sélection bande » qui est localisée sur le pin « AC24 » et associée au GPIO Dip switch 8.

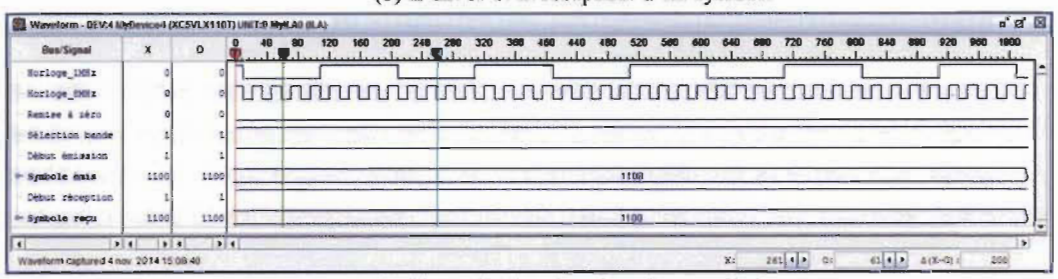
La figure 4.17 nous montre le test et la simulation réelle de deux émetteurs / récepteurs en bande de base avec la sélection de la bande. En effet, nous avons activé la remise à zéro et donc les sorties dans le simulateur sont à zéro (en (a)), après, nous avons désactivé l'entrée remise à zéro en jouant sur Dip switch 1, bien sûr, il est nécessaire d'activer les switches liés aux entrées début émission et début réception, et aussi nous jouons sur le Dip switch 8 afin de sélectionner la bande.



(a) Lors d'une remise à zéro



(b) L'envoi et la réception d'un symbole



(c) L'envoi et la réception d'un symbole

Figure 4.17 : Test sous l'outil Chipscope Pro

En (b) de la même figure, nous avons entré notre symbole (0101) à émettre en fixant les Dip switches correspondants sur la carte, et en (c) nous avons introduit un deuxième symbole (1100). Nous pouvons transmettre n'importe quel symbole en l'introduisant sur les Dip switches adéquats, nous obtenons le même symbole à la réception en le vérifiant à travers les Leds.

4.4 Test de la boucle de retour pour l'application

Avant de présenter les résultats liés au test de la boucle de retour de l'application d'envoi et de réception, nous commençons par le module adaptation de 4 bits à 8 bits.

4.4.1 Le module adaptation de 4bits à 8bits

Ce module a pour objectif d'adapter les données sur 4 bits venant de la sortie symbole reçu de l'émetteur / récepteur en bande de base vers l'entrée sur 8 bits du module UART.

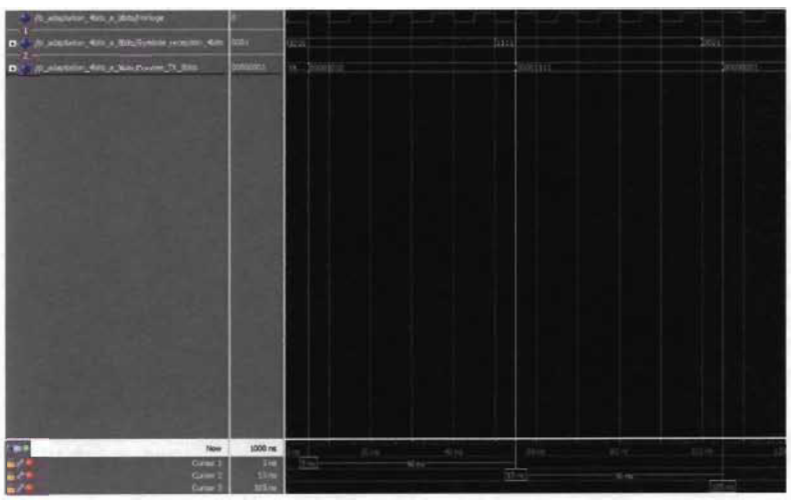


Figure 4.18 : Simulation du module adaptation 4 bits à 8 bits

D'après la simulation sous ModelSim (dans la figure 4.18), la sortie est synchronisée avec l'entrée, donc nous commençons à recevoir la donnée en sortie au front montant suivant.

4.4.2 Test de la boucle de retour avec l'interface graphique

Nous faisons un test de l'application d'envoi et de réception d'une donnée passant par les modules implémentés sur FPGA (module UART, module séparation des données et module adaptation de 4 bits à 8 bits) ainsi en émission de la donnée, nous vérifions les sorties qui sont connectées aux LEDs utilisateurs sur la plateforme et en réception, nous jouons sur les Dip switchs avec la vérification de la donnée reçue dans l'application.

Au moment de la configuration du fichier de connexion pour l'analyseur logique intégré, nous avons introduit le port d'horloge de 200 MHz ainsi que neuf ports de détection pour les visualiser sous Chipscope Pro qui représentent les entrées sorties du module (dans la figure précédente) implémenté à l'exception de la donnée reçue RXD et la donnée envoi TXD.

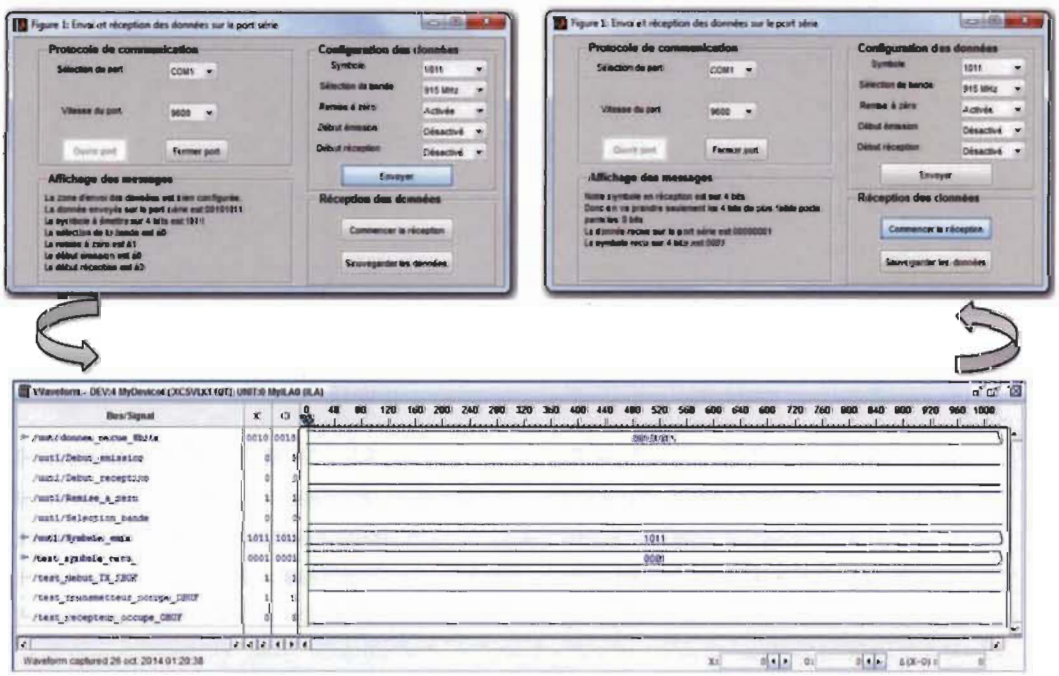
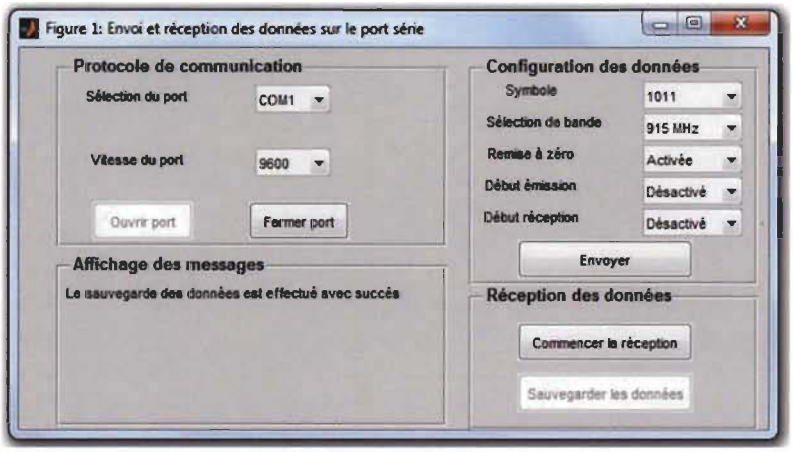


Figure 4.19 : Résultats de test obtenus

D'après les résultats obtenus (sur la figure 4.19), les fonctionnalités d'émission et de réception des données par l'application, en les comparant avec celles trouvées par l'analyseur logique intégré, fonctionnent.



(a) Après l'appui sur sauvegarder les données

Workspace

Name	Min	Value	Max
donnee_decimale_emise	43	43	43
donnee_sur_8bits_emise		'00101011'	
donnee_sur_8bits_recue		'00000001'	
symbole_emis		'1011'	
symbole_sur_4bits_recu		'0001'	
valeur_decimale_recue	1	1	1

(b) Le contenu de notre fichier

Figure 4.20 : La sauvegarde des données émises et reçues

Nous avons fait une sauvegarde (figure 4.20) de six valeurs dans fichier « données sauvegardées.mat » enregistré sous l'arborescence de l'espace du travail. Si nous vérifions le fichier, nous trouvons les mêmes données envoyées et reçues par l'application; sachant que 43 et 1 sont les valeurs décimales qui correspondent respectivement à 00101011 et 00000001.

4.5 La chaîne d'émission sur l'oscilloscope avec le contrôle par l'application

Nous exposons dans cette partie, les résultats liés au test réel et simulation de la chaîne d'émission en bande de base sur l'oscilloscope MDO4034B-3 ainsi que le contrôle (des entrées de l'émetteur) effectué par l'interface graphique.

Tableau 4-9 Ressources occupées par le module de test

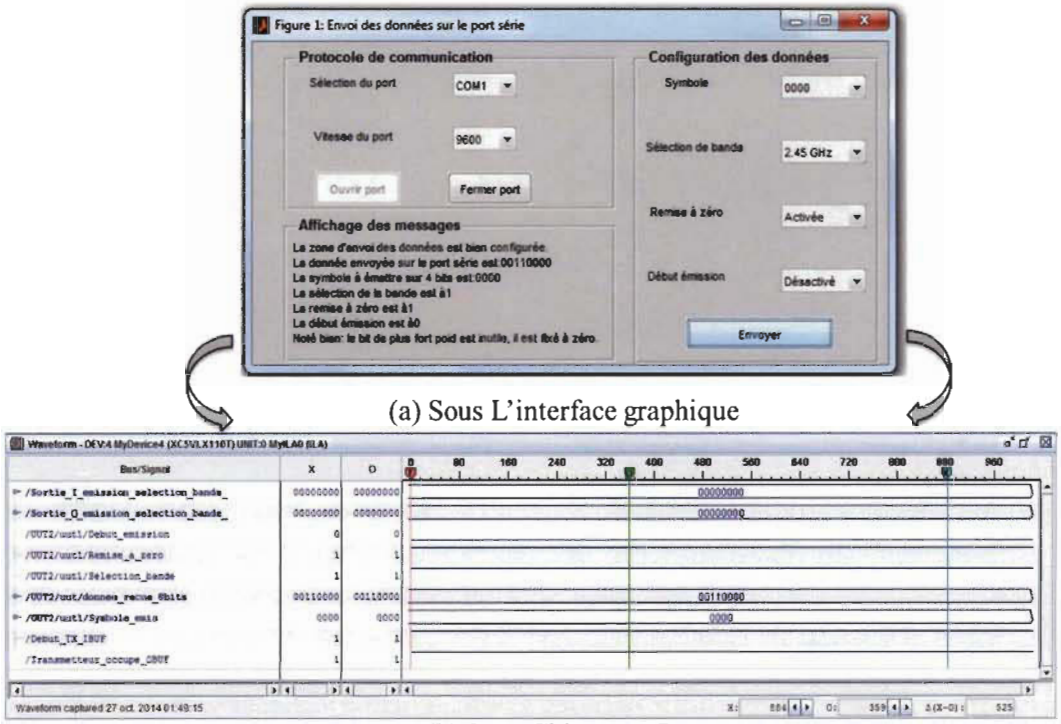
Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	306	69120	0 %
Nombre de LUTs	515	69120	0 %
Nombre de paires LUT-FF	175	646	27 %
Nombre de blocs E/S	27	640	4 %
Nombre de BUFGs/BUFGCTRS	3	32	9 %

Le tableau 4-9 dresse les ressources occupées de tout le module implémenté à tester. Il nous confirme que le module consomme treize blocs d'entrée / sortie qui sont suffisantes pour le test sur la carte.

Au niveau du test, nous avons configuré (dans le fichier de connexion) neuf ports de détection pour les vérifier sur Chipscope Pro qui sont : sortie I émission, sortie Q émission, symbole émis, début émission, remise à zéro, sélection de bande, début TX, transmetteur occupé, et donnée reçue sur 8 bits.

Concernant le fichier des contraintes utilisateurs, nous avons attribué les ports donnée envoi TXD et donnée reçue RXD respectivement sur les pins de FPGA AG20 et AG15 (correspondent aux entrée et sortie du port série sur la plateforme d'évaluation). Le port sortie I émission (0 à 7) est respectivement sur : H33, H34, G32, J32, L33, P34, AA34, Y34 et aussi le port sortie Q émission (0 à 7) est respectivement sur : AH34, AG32, AK34, AJ32, AL34, AM33, AM32, AN33 (correspondent aux entrées de l'interface auxiliaire sur

la plateforme d'évaluation ou en anglais XGI Expansion Headers). Pour plus des détails, veuillez consulter la référence [27].



(a) Sous L'interface graphique

(b) Sous Chipscope Pro

Figure 4.21 : Lors d'une remise à zéro

Nous faisons un test de l'application d'envoi d'une donnée (nous nous intéressons ici seulement par l'envoi de la donnée) passant par les modules implémentés sur FPGA (module UART et module séparation des données), ainsi à l'émission de la donnée, nous vérifions les sorties qui sont sondées par l'intermédiaire d'analyseur logique intégré qui représentent les signaux visualisés par l'outil Chipscope Pro, et nous pouvons modifier au besoin les données à envoyer par l'interface graphique.

Lors de l'activation de la remise à zéro (en (a) de la figure 4.21), nos sorties sont réinitialisées en (b) de la même figure.

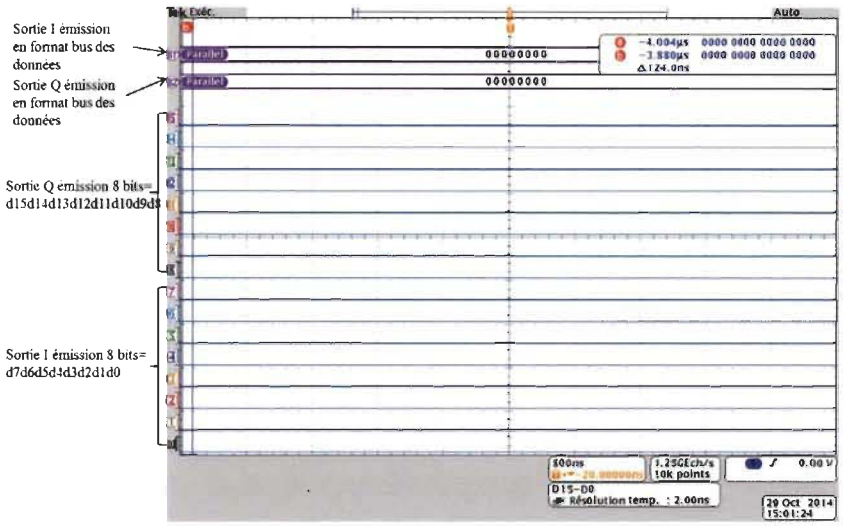


Figure 4.22 : Résultats de visualisation sur l'oscilloscope

La figure 4.22 nous confirme le fonctionnement de l'architecture implémentée lors de la remise à zéro et nous voyons bien que tous les signaux numériques sont mis à zéro. En effet, les données représentées allant de d0 (en bas) vers d7 correspondent aux 8 bits de la sortie émission sur I et les données restantes (c.-à-d. de d8 jusqu'au d15) représentent les 8 bits de la sortie émission sur Q. Pour mieux voir les sorties, nous avons créé deux bus de données parallèles b1 (pour la première sortie) et b2 (pour la deuxième sortie).

Après la désactivation de la remise à zéro, nous pouvons envoyer n'importe quel symbole avec l'activation du début émission et le choix de la bande, comme le montre la figure 4.23 en (a). Nous avons envoyé le symbole 0110 avec la sélection de la bande 2.45 GHz et ainsi les données sont visualisées par Chipscope Pro en (b). En fait, nous avons fait un zoom sur l'affichage des résultats pour voir clairement les 8 bits de chaque sortie.

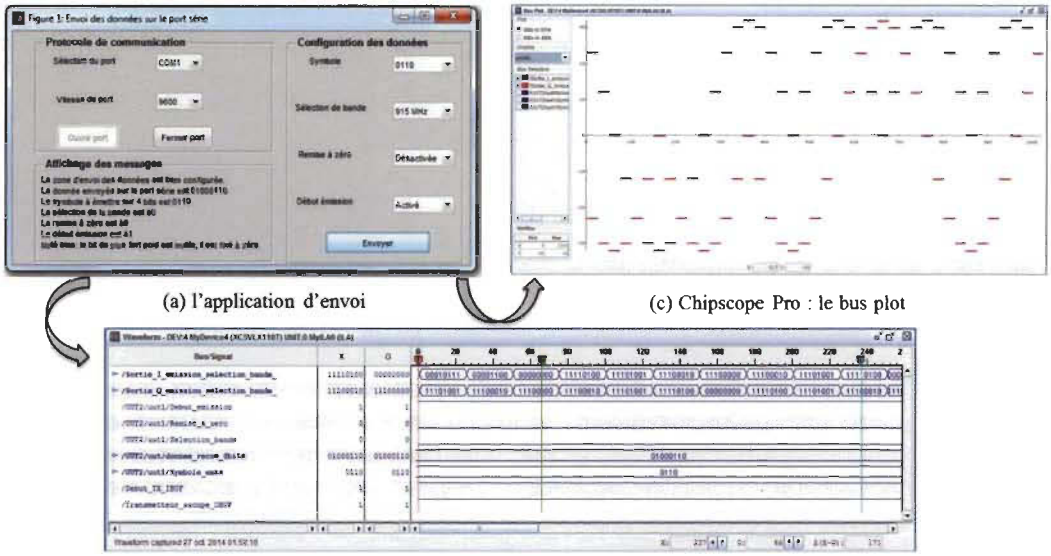


Figure 4.23 : Lors de l'envoi d'un symbole

Nous avons aussi les résultats du bus plot en (c) qui nous permet de visualiser les sorties après les avoir converties en analogique avec des valeurs calculées en nombres décimaux signés. Ici les deux sorties sont corrélées et la visualisation se fait par des points.

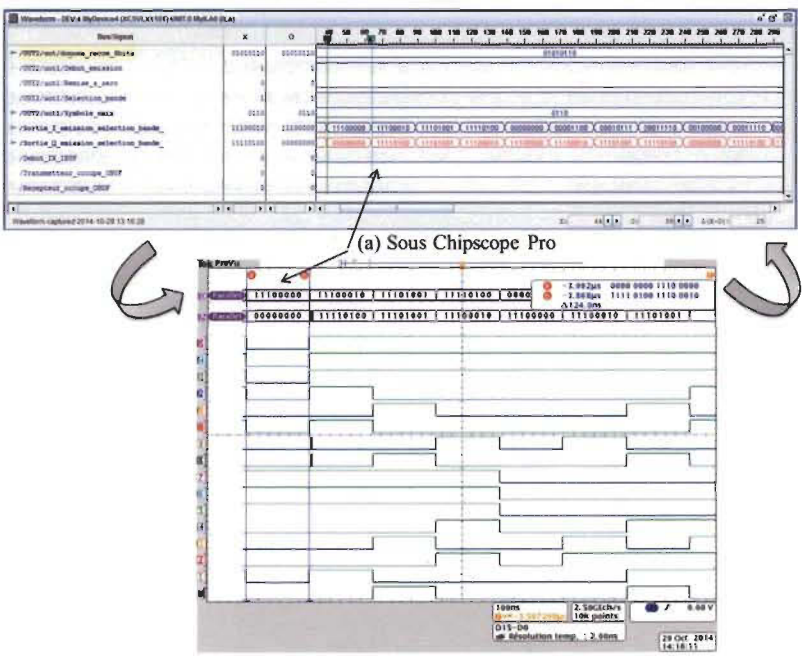


Figure 4.24 : Comparaison des résultats

La visualisation des signaux sur l'oscilloscope est très dense due à sa configuration (2.5 Giga Echantillons/s et 1 Méga points) et nous pouvons vérifier seulement les curseurs. En effet, les curseurs a et b nous permettent de vérifier les sorties à un moment bien déterminé.

La visualisation des signaux de la figure 4.24 est en mode fenêtrage c.à.d. nous avons zoomé l'affichage de signaux. Si nous comparons les signaux sous Chipscope Pro en (a) et ceux de l'oscilloscope en (b), nous pouvons conclure que l'architecture fonctionne bien.

4.6 Test de la boucle de retour de la chaîne d'émission / réception

A ce niveau, nous présentons les résultats obtenus à travers le test de la boucle de retour. Les deux sorties de l'émetteur en bande de base sont injectées dans les deux entrées du récepteur. Ainsi nous vérifions le symbole transmis et le symbole reçu par l'application d'envoi et de réception sur le port série.

Le tableau 4-10 affiche les ressources nécessaires pour l'implémentation de l'architecture liée au test de la boucle de retour. Nous remarquons bien que le circuit occupe assez des unités de traitement et un peu de mémoire.

Tableau 4-10 : Ressources occupées par l'architecture

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	794	69120	1 %
Nombre de LUTs	2058	69120	2 %
Nombre de paires LUT-FF	217	2635	8 %
Nombre de blocs E/S	6	640	0 %
Nombre de BUFGs/BUFGCTRS	3	32	9 %
Nombre des DSP48E	14	64	21 %

Afin de voir les signaux sur l'analyseur logique intégré par Chipscope Pro, nous avons configuré le fichier de connexion qui attribue neuf ports de détection : début TX, symbole

émis, symbole reçu, sélection de bande, remise à zéro, début émission, début réception, transmetteur occupé, et récepteur occupé.

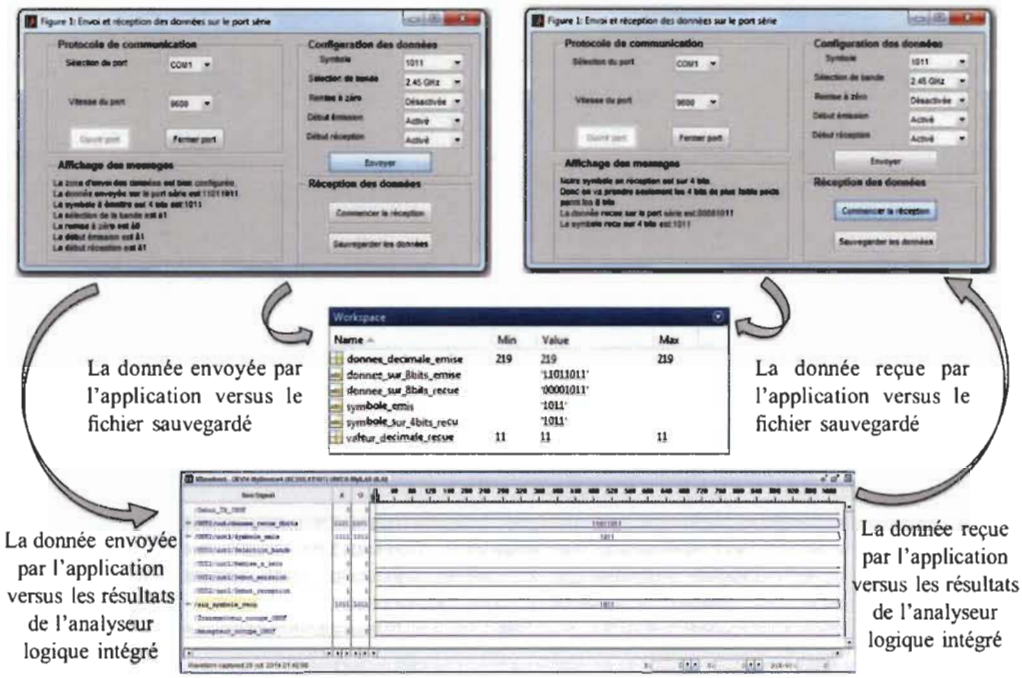


Figure 4.25 : Résultats du test de la boucle de retour

Comme le montre la figure 4.25, le symbole émis par l'interface graphique est le même qui est reçu (après la configuration des entrées de contrôle de l'émetteur / récepteur en bande de base) en vérifiant les résultats sur l'application, dans le fichier sauvegardé et sur Chipscope Pro.

Dans cette partie, nous n'avons pas présenté tous les résultats liés à la chaîne de communication reconfigurable par contre cette chaîne est capable de travailler avec : n'importe quel taux d'échantillonnage (en modifiant l'horloge 8 MHz, les coefficients du filtre et le facteur d'interpolation), différents taux de transmission (en jouant sur l'horloge 1 MHz), les deux bandes 915MHz ou 2.45 GHz, deux modulations possibles (QPSK et O-QPSK suivant que nous programmons avec ou sans décalage entre I et Q), et aussi nous

pouvons jouer sur le nombre de bits de deux sorties (pour l'adapter à la résolution du convertisseur numérique-analogique).

4.7 Banc de test global

La figure 4.26 nous montre le banc de test global installé avec tous les équipements nécessaires.

Figure 4.26 : Banc de test global

Nous avons essayé de visualiser les deux signaux analogiques en bande de base (sorties I et Q du convertisseur) sur les entrées d'oscilloscope numérique MSO-X3054A avec un couplage CC. Nous avons constaté qu'il y a un décalage de tension de sortie de 0.85 V dû au DC bias. Pour enlever ce décalage il faut insérer un condensateur série sur chaque voie. Dans notre cas nous avons raccordé chaque sortie avec un DC Bloc Agilent N9399C [35]. A travers l'oscilloscope, nous avons mesuré la fréquence de notre signal utile qui est de 500 kHz. Donc nous respectons bien le théorème de Shannon en comparant avec la fréquence d'échantillonnage 8 MHz, l'amplitude du signal est presque 1.45 $V_{crête\text{-}crête}$, le symbole se répète chaque 16 us et enfin le décalage entre I et Q est 0.5 us. Nous avons remarqué aussi que les signaux sont bruités et avons mesuré la fréquence de ce bruit qui est de l'ordre de 8

MHz qui représente la fréquence d'échantillonnage, vu que la fréquence de coupure du filtre passe bas Butterworth est de 13 MHz (figure 4.27). Sachant que le signal en I est représenté par la voie 1 en jaune et le signal en Q est représenté par la voie 2 en vert.

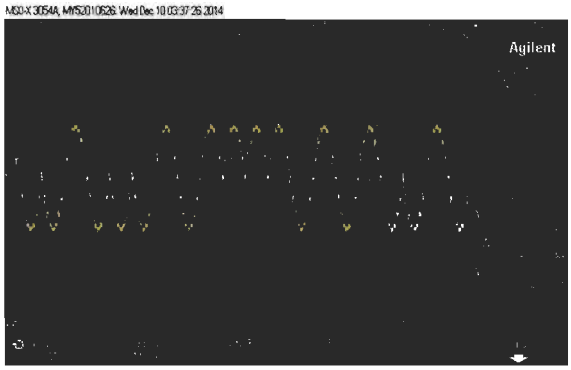


Figure 4.27 : Les sorties du CNA avec une fréquence de 8 MHz

Donc pour remédier à ce problème, nous avons deux solutions : soit nous insérons un autre filtre passe bas RC pour filtrer cette fréquence (par exemple avec une fréquence de coupure de 3 MHz par un simple condensateur de 1 nF vu que l'impédance des câbles est 50 Ω), soit nous augmentons la fréquence d'échantillonnage au-delà de 13 MHz (20 MHz par exemple) en modifiant le coefficient d'interpolation et aussi les coefficients du filtre (40 coefficients pour construire un sinus au lieu de 16). Ces coefficients sont choisis en appliquant cette formule : 511 x sin((n x π)/20)+511 ; n=0...39.

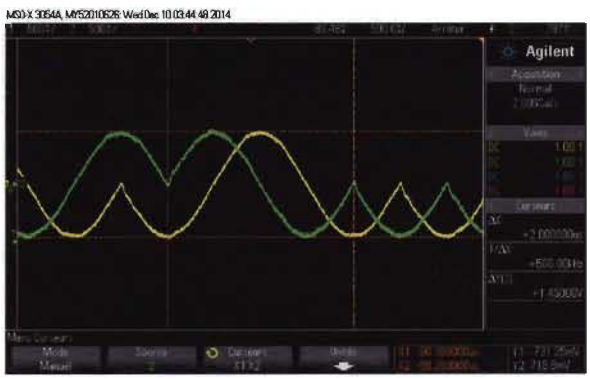


Figure 4.28 : Les sorties du CNA avec une fréquence de 20 MHz

Concernant la deuxième solution, pour assurer le décalage entre I et Q de la modulation O-QPSK (qui est 0.5 us), nous devons fixer le coefficient de retard qui doit être un nombre entier pour la boucle sur la fréquence 20 MHz et aussi le rapport du diviseur d'horloge entre 200 MHz et 20 MHz (200/20=10). Suivant la figure précédente, le décalage est de l'ordre de 0.5 us avec 10 itérations (x 50ns) de la boucle sur 20 MHz. Nous remarquons que les deux signaux contiennent encore un peu de bruit (la fréquence 20 MHz).

Afin d'éviter le problème de recouvrement avec la fréquence de coupure du filtre Butterworth, nous augmentons encore la fréquence d'échantillonnage à 50 MHz avec 100 coefficients pour le filtre d'interpolation qui sont choisis par la formule suivante : 511 * Sin ((n * π)/50) + 511 pour n=0..99.

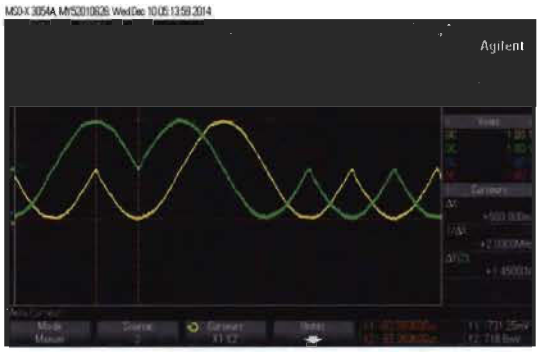
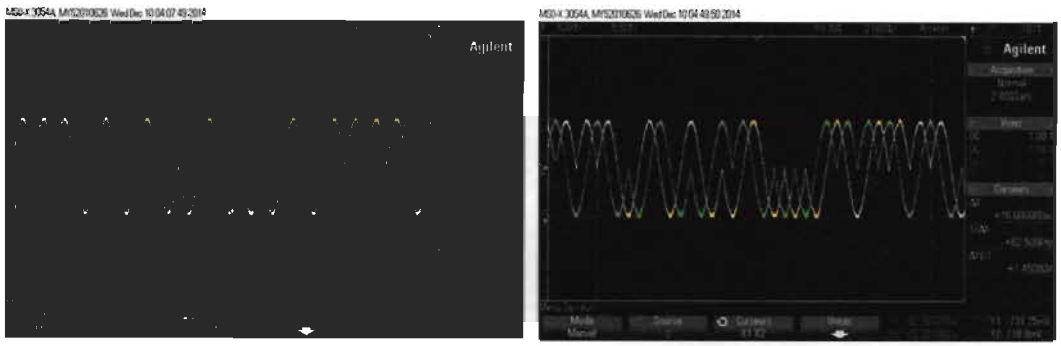


Figure 4.29 : Les sorties du CNA avec une fréquence de 50 MHz

Comme le montre la figure 4.29, les deux signaux analogiques en bande de base sont filtrés, malgré la liaison entre les cartes par des fils de transmission qui cause toujours des problèmes d'interconnexion dus principalement au rayonnement ou un type de couplage qui se produit. De ce fait, au niveau de la conception des interfaces des données, nous trouvons toujours l'effet du courant de retour (par un plan de masse) en se basant sur la norme de compatibilité électromagnétique afin de remédier à ce problème de perturbation et de perte d'information.



(a) pour le symbole 0000 (b) pour le symbole 1000

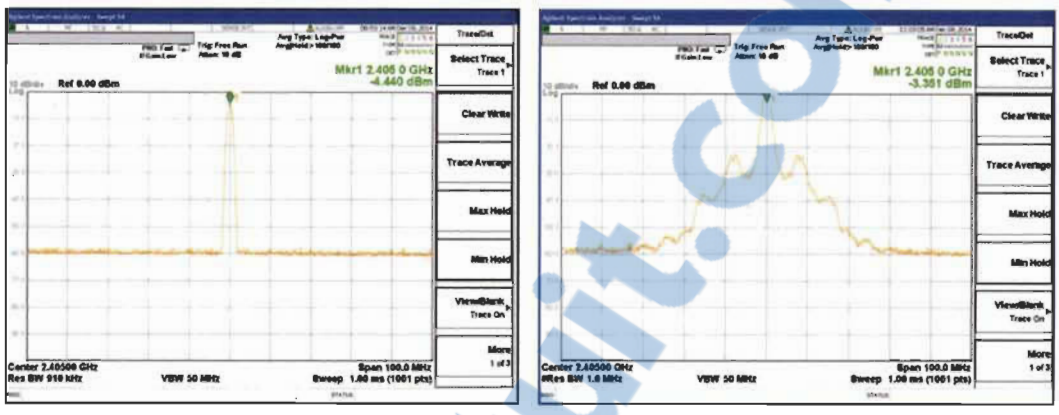
Figure 4.30 : La visualisation des sorties sur l'oscilloscope

Il y a un moyen de vérifier les symboles transmis en comparant la séquence du chip de 32 bit avec le tableau 1-2 dressé dans le deuxième chapitre. Donc suivant la figure 4.30, la visualisation des sorties sur l'oscilloscope avec deux combinaisons du symbole (0000 et 1000) sachant que la séquence reste la même pour 0000 jusqu'au 0111 il y a juste un décalage de 4 bits, pareillement pour 1000 jusqu'au 1111, la séquence reste la même avec un décalage de 4 bits. Ces résultats nous confirment le bon fonctionnement de l'émetteur en bande de base avec une certaine amélioration sur les signaux obtenus à savoir le filtrage dans le but d'assurer un spectre du signal RF filtré.

La figure 4.31 représente les quatre cas de visualisation sur l'analyseur du spectre RF (ils sont pris par un maintien maximal ou max hold en anglais) tel qu'en (a), nous voyons seulement le spectre de la porteuse (sans entrées I et Q) crée par le générateur du signal RF et nous remarquons bien que la bande est étroite. Elle est focalisée sur la fréquence 2.405 GHz avec une puissance de l'ordre de -4.4dBm.

En (b), nous voyons bien que le spectre est étalé (dans le cas de 8 MHz) dû à la séquence d'étalement du chip par la modulation DSSS pour le protocole de communication

ZigBee large bande (consultez la figure 2.1 du deuxième chapitre), en plus de l'apparition du lobe principal et les lobes secondaires.



(a) Le spectre de la porteuse (b) Le spectre du signal pour 8 MHz

(c) Le spectre du signal pour 20 MHz (d) Le spectre du signal pour 50 MHz

Figure 4.31 : Résultats d'analyseur du spectre

En (c), le spectre du signal RF est filtré par rapport à l'image en (b) grâce à l'augmentation de la fréquence d'échantillonnage à 20 MHz mais nous voyons encore deux lobes en petite amplitude. Lorsque que nous avons augmenté encore la fréquence d'échantillonnage (à 50 MHz qui résout le problème de recouvrement), nous remarquons bien que le spectre est bien filtré en (d). Suivant la norme et le protocole ZigBee, le spectre du signal RF est acceptable après l'élimination de tout bruit par un filtrage.

A travers le même outil, nous avons fait des mesures de la résolution de la bande passante qui est de 1 MHz et la densité spectrale de puissance qui est -67 dBm/Hz.

4.8 Test d'envoi d'une trame PPDU

Dans ce qui suit nous allons procéder à deux tests réels afin de vérifier le bon fonctionnement lors de l'envoi d'une trame PPDU. Sachant que, nous avons mis dans l'annexe D, un complément des résultats pour ces tests.

4.8.1 Test de la chaine d'émission

Le premier test consiste à envoyer une trame PPDU basant sur le tableau 2-1. En effet, nous créons une interface graphique sur GUIDE de Matlab et à partir de cette interface, nous construisons la trame PPDU et nous configurons la chaîne.

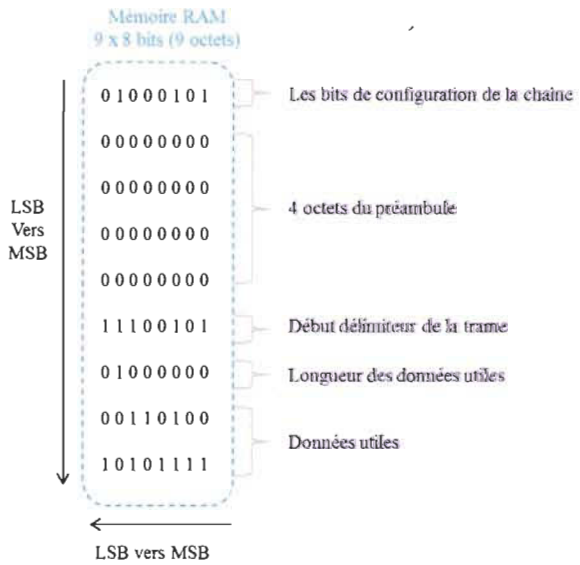


Figure 4.32 : Les données existantes dans la mémoire RAM

Les données (9 octets) sont envoyées sur le port série et elles sont sauvegardées dans une mémoire RAM 9 x 8 bits (nous allons utiliser seulement 9 lignes parmi 16)

implémentée dans la plateforme FPGA (décrite en VHDL). Les données sauvegardées dans la RAM sont organisées de la façon présentée par la figure 4.32. Pour plus des détails, consultez la dernière section de l'annexe D.

La mémoire RAM va acheminer ses données vers les entrées de configuration et l'entrée principale (la trame PPDU) de la chaîne d'émission en bande de base.

Nous envoyons une trame PPDU avec le type des données « Réservé », les données utiles se composent de 2 octets. Ce type est décrit dans le tableau 2-1 du deuxième chapitre et la trame comporte (64 bits ou 8 octets) les champs suivants :

- Préambule (32 bits) : 0000 0000 0000 0000 0000 0000 0000 0000
- Début délimiteur de trame (8 bits) : 1110 0101
- Longueur des données (7 bits et 1 bit supplémentaire) : 0100 0000
- Données utiles (16 bits) : 1010 1111 0011 0100 (à titre d'exemple)

Sachant que le bit envoyé (de la trame) en premier est le bit le moins significatif du préambule (bit0), le dernier bit envoyé est le bit le plus significatif du champ « données utiles » (bit15).

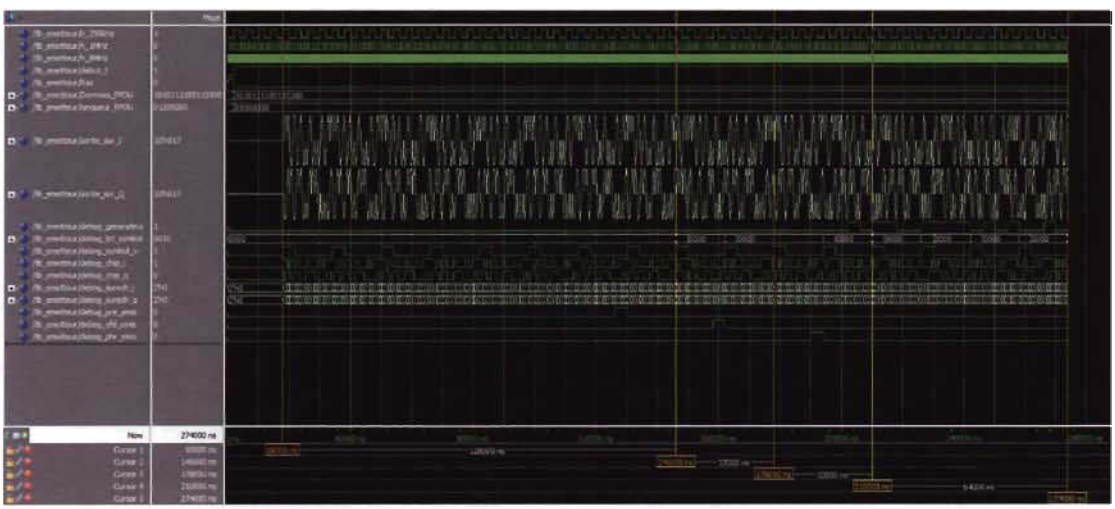


Figure 4.33 : La simulation de la trame PPDU

La simulation sous ModelSim (de la figure 4.33) assure le bon fonctionnement de l'émetteur :

- Le temps de simulation total est de 274 us ;
- La période qui sépare le curseur1 et le curseur2 est de 128 us et représente le préambule avec 8 symboles zéros (chaque symbole sur 16 us)
- Le temps séparant le curseur2 et le curseur3 est de 32 us représente le champ « début délimiteur de trame » avec 2 symboles ;
- La troisième durée est 32 us et correspond au champ « longueur des données » avec 2 symboles ;
- La dernière période (64 us) sert au dernier champ de la trame PPDU « données utiles » constitué de 4 symboles.

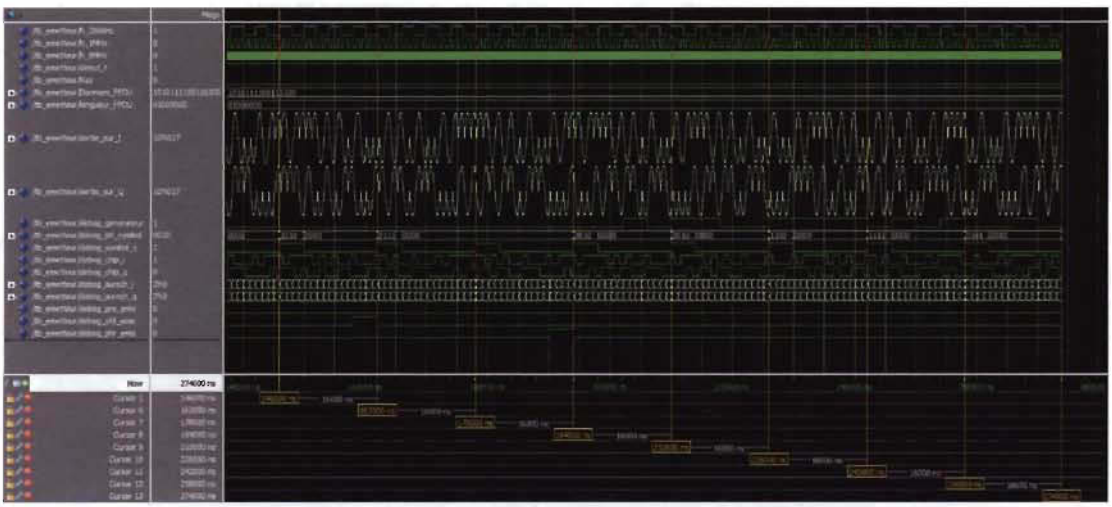


Figure 4.34 : Une vue zoomée sur les données

La figure ci-dessus affiche une vue zoomée sur le données de la trame en partant du deuxième champ (début délimiteur de trame). Nous voyons bien que les symboles envoyés en ordre (16 us de chaque) : 1010 → 0111 → 0000 → 0010 → 0010 → 1100 → 1111 → 0101. Les symboles sont inversés au niveau de l'affichage vu que nous commençons toujours à envoyer le bit le moins significatif de chaque champ.

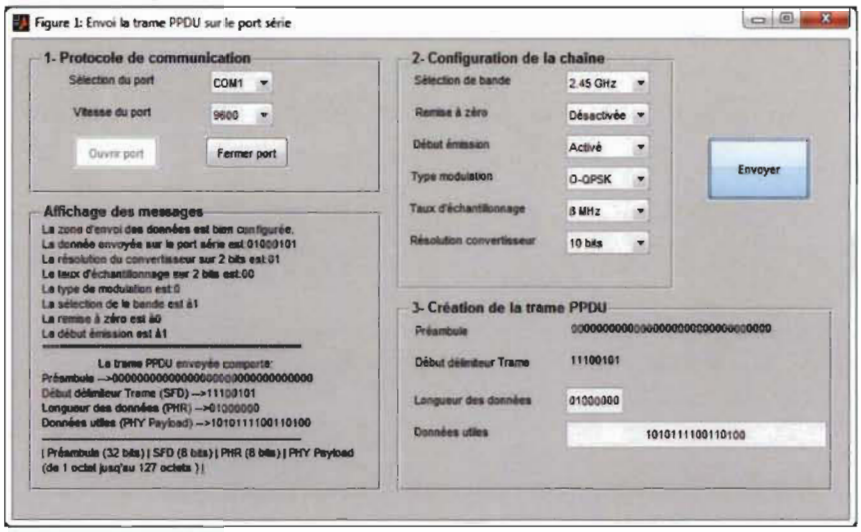


Figure 4.35 : L'interface d'envoi d'une trame PPDU

La figure 4.35 représente l'interface complète pour envoyer une trame PPDU sur le port série. Elle est développée sous GUIDE de Matlab. Par rapport à l'ancienne interface, nous avons ajouté un nouveau panneau pour la création de la trame. Les deux champs : préambule et début délimiteur de la trame sont toujours les mêmes. Par contre, nous pouvons varier les deux autres champs : longueur et données utiles.

Dans le panneau « configuration de la chaîne », nous avons ajouté d'autres paramètres (type de modulation, taux d'échantillonnage et résolution du convertisseur) afin d'avoir plus de flexibilité de la chaîne.

La figure 4.36 nous montre les résultats de l'émetteur en bande de base sur Chipscope Pro. En (a), nous avons le symbole « 1100 » qui fait partie de nos données utiles et nous respectons bien la période prévue pour chaque symbole. Nous sommes limités au niveau de l'affichage des résultats vu que la profondeur maximale de la fenêtre de l'échantillon est 1024, sachant que la période 4 us représente une profondeur de 800. Donc nous ne pouvons pas voir tous les symboles pour un seul déclenchement.

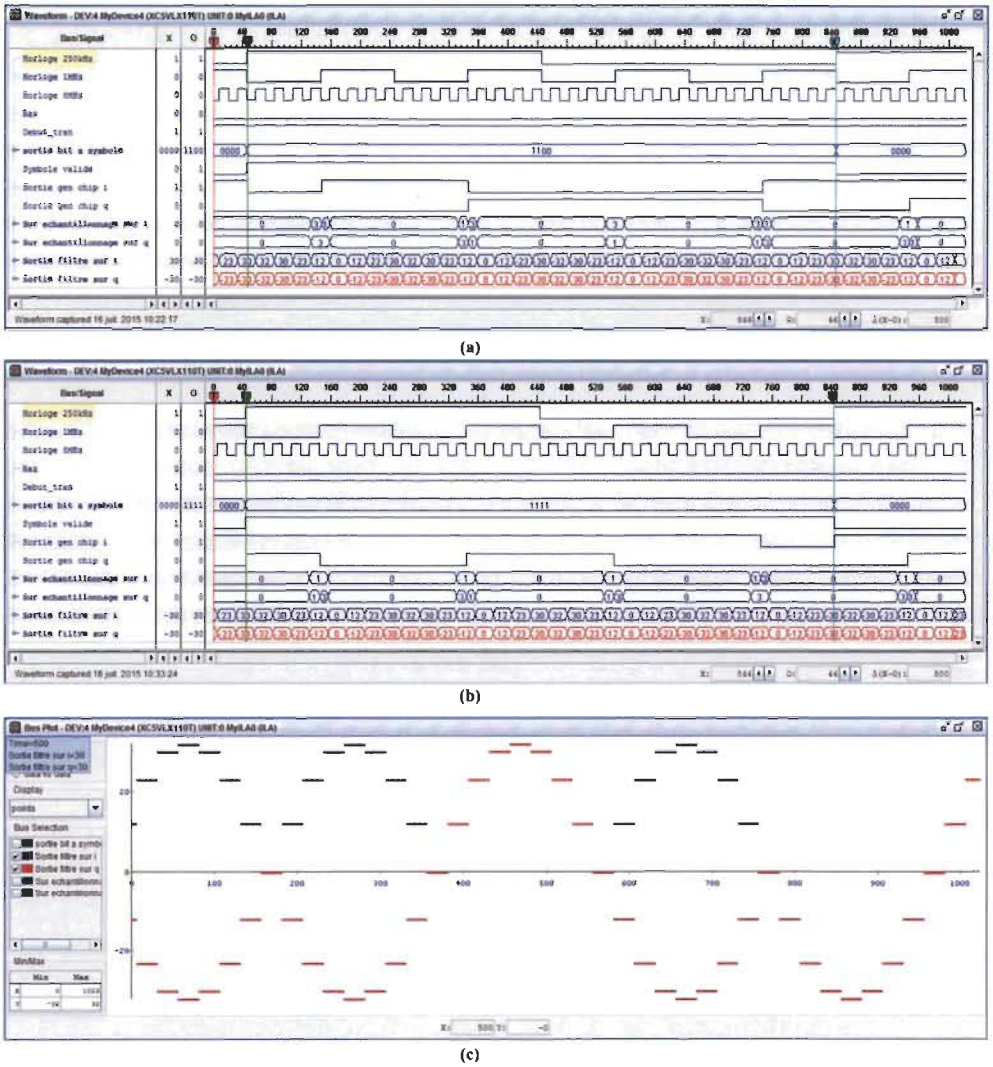


Figure 4.36 : Résultats de l'émetteur sur Chipscope Pro

Mais, le mode d'exécution répétitif nous permet de vérifier l'acheminement des données durant le test. Le deuxième déclenchement en (b) nous affiche le symbole suivant « 1111 ». En (c), nous voyons les deux sorties de l'émetteur (sortie filtre sur I et sortie filtre sur Q) durant le dernier symbole.

4.8.2 Test de la trame sur l'émetteur / récepteur

Ici, nous testons l'envoi de la trame sur l'émetteur/récepteur avec le contrôleur. La simulation de la figure suivante montre que la trame est bien reçue si nous la comparons

avec celle de l'émission en respectant la période de chaque symbole. Les données sont affichées en ordre (partant du champ début délimiteur de trame jusqu'au dernier symbole des données utiles) et en format hexadécimal sur 4 bits.

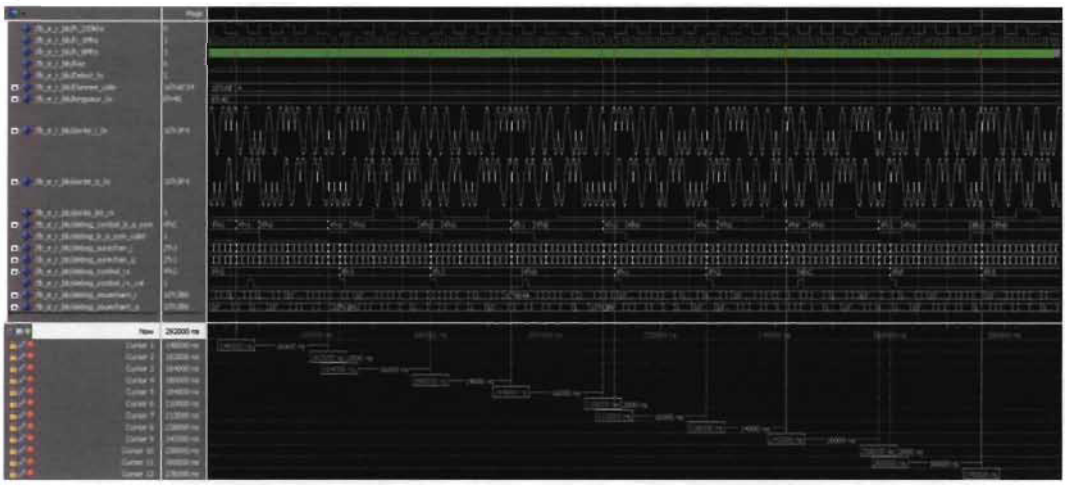


Figure 4.37 : Simulation de l'E/R sous Modelsim

Au niveau du test réel, nous pouvons ajouter le paramètre « mode de transmission » dans le panneau « configuration de la chaîne » de l'interface : mode arrêt 00, mode boucle 01, mode émission 10 et mode réception 11. Mais, au long de cette partie, nous faisons seulement un test du mode boucle.

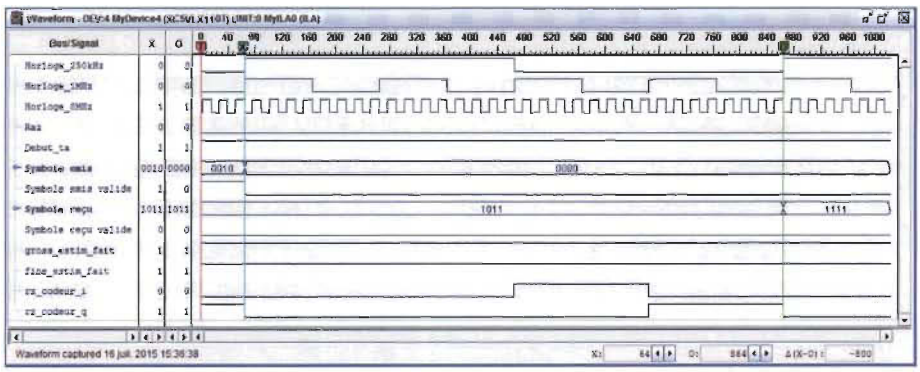


Figure 4.38 : Test de l'E/R sur Chipscope Pro

Nous avons vérifié à travers le mode d'exécution répétitif, l'acheminement des données de la trame dans l'émetteur et le récepteur. La figure précédente représente un test d'un seul déclenchement. Donc il est normal que nous voyions la non-coïncidence des données en émission et en réception, vu qu'il y a toujours un décalage entre le symbole émis et le symbole reçu de 18 us (plus que 1024 de profondeur). Veuillez consulter l'annexe D, pour avoir plus des détails.

4.9 Conclusion

Dans ce chapitre, nous avons présenté toute la contribution sur la conception de la chaîne de communication. Nous avons exposé les simulations effectuées sur le simulateur ModelSim, en plus, nous avons visualisé les résultats obtenus par la synthèse sous l'outil Xilinx ISE. Enfin, nous avons installé un banc de test réel comprenant la plateforme, le convertisseur numérique-analogique, le modulateur en I/Q, l'analyseur du spectre et utilisant l'outil Chipscope Pro ainsi que l'interface graphique sur GUIDE de Matlab. Dans l'annexe D vous trouverez une partie de nos travaux qui a pour but d'étudier la chaîne de communication numérique avec la modulation BPSK ainsi que la partie consacrée aux résultats obtenus au niveau de l'étude du convertisseur numérique analogique Sigma-Delta.

Chapitre 5 - Conclusion

Premièrement, nous avons étudié la composition de la couche physique définit par le standard IEEE 802.15.4, à savoir la modulation O-QPSK, la technique d'étalement du spectre DSSS et les architectures internes pour chaque bande de fréquence ISM. Ainsi, nous avons décrit la configuration numérique d'un système de communication par un circuit programmable FPGA. Deuxièmement, nous avons exposé les différentes parties à étudier ou à concevoir pour notre projet, ainsi, nous avons énuméré les spécifications et les besoins techniques (matériels et logiciels) pour la réalisation de ce projet. Nous avons détaillé notre méthodologie étudiée dans le but de faciliter la compréhension. Finalement, nous avons présenté les simulations effectuées sur le simulateur ModelSim, en plus, nous avons visualisé les résultats obtenus par la synthèse sous l'outil Xilinx ISE. Enfin, nous avons installé un banc de test réel du transmetteur comprenant la plateforme, le convertisseur numérique-analogique, le modulateur en I/Q, l'analyseur du spectre avec l'utilisation de l'outil Chipscope Pro et l'interface graphique sur GUIDE de Matlab. Sachant qu'il y a une partie de nos travaux étudiés dans l'annexe D.

Nous avons obtenu des bons résultats que ce soit au niveau de la conception et les simulations ou même par le test réel et aussi durant le test à travers la visualisation des signaux numérique sur l'analyseur logique, la vérification des sorties I et Q analogiques en bande de base (les deux sorties du convertisseur numérique-analogique) par l'oscilloscope numérique, l'analyse du spectre du signal RF à la sortie du générateur du signal RF.

Comme perspective, nous pouvons fournir les améliorations suivantes au modèle développé:

Implémentation du système de RSSI : comme décrit dans le chapitre 2, l'indicateur de niveau de signal ou d'un système RSSI (Received Signal Strength Indicator) est une estimation de la puissance du signal reçu dans la bande passante d'un canal donné. Nous pouvons calculer cette valeur en mesurant l'énergie des échantillons entrants et en l'accumulant pendant une période de temps. La puissance est calculée en divisant l'énergie accumulée par le nombre N d'échantillons testés.

Le RSSI peut être mis en œuvre par le CORDIC en mode vectoriel qui nous permet de convertir une valeur complexe de la forme cartésienne en forme polaire (modulo, phase). Ainsi, nous avons juste à accumuler les modules d'échantillons entrants (calculés avec le CORDIC rotationnel) pour N échantillons et à la fin, les diviser par N échantillons. Cette division peut être implémentée par un décalage vers la droite si le nombre d'échantillons N est une puissance de 2. En outre, la CCA ou Clear Channel Assessment et l'indicateur de la qualité de liaison (LQI) peuvent être implémentés à partir du RSSI.

Améliorations du contrôleur d'émission / réception : La nécessité d'un composant extérieur pour contrôler les deux signaux (Début_tx, Début_rx) peut être éliminée en utilisant le champ de la longueur de PDU de ZigBee. Lorsque vous lancez l'unité d'émission, un compteur s'initialise avec la valeur du champ de longueur et le processus ne prend pas fin tant que le compteur n'est pas égal à 0. De même, dans le processus de la réception, une fois que le champ de longueur est détecté, un autre compteur serait initialisé et ainsi, diminuant chaque bit reçu, nous pourrons contrôler le total du paquet reçu sans nécessité de contrôler avec une unité de contrôle externe.

Références

[1] IEEE, Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks, IEEE Standard for Information Technology, 2006.

[2] Modèle OSI : http://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI

[3] L'institut des ingénieurs en électrique et électronique http://www.ieee.org/index.html

[4] Module Xbee série 1 : http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#overview

[5] ZigBee Alliance. ZigBee Specification. 2008: http://www.zigbee.org/

[6] Module Xbee Zb: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#overview

[7] Module Xbee 868 MHz: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-pro-868#overview

[8] Module Xbee 900 MHz: http://www.digi.com/products/wireless-wired-embedded-solutions/rf-modules/xbee-rf-modules/xbee-proprietary-rf-modules/xbee-pro-900hp

[9] Mémoire, 'IMPLEMENTING PHYSICAL LAYER (PHY) OF IEEE 802.15.4G STANDARD WITH DIRECT SEQUENCE SPREAD SPECTRUM (DSSS) USING OFFSET QUADRATURE PHASE SHIFT KEYING (O-QPSK)', 2012, disponible sur : http://sdsu-dspace.calstate.edu/bitstream/handle/10211.10/1905/Panchal_Karan.pdf?sequence=1

[10] Zigbee Tutorial : http://www.rfwireless-world.com/Tutorials/Zigbee_tutorial.html

[11] Mémoire, 'DESIGN OF ZIGBEE TRANSCEIVER FOR IEEE 802.15.4 USING MATLAB/SIMULINK', 2011, disponible sur : http://ethesis.nitrkl.ac.in/2816/1/209EC1106_thesis.pdf

[12] ZIGBEE COMMUNICATION APPARATUS AND METHOD FOR HIGH-SPEED TRANSMISSION AND RECEPTION, disponible sur: http://www.faqs.org/patents/app/20100202564

[13] 2.4 GHZ IEEE 802.15.4/ZIGBEE® RF TRANSCEIVER: http://www.ti.com/lit/ds/symlink/cc2520.pdf

[14] Volnei A. Pedroni. Circuit Design with VHDL. MIT Press, Cambridge, Massachusetts, London, England, 2004.

[15] B. Razavi. RF Microelectronics Prentice Hall, Upper Saddle River, NJ, 1998.

[16] Bernard Sklar. Digital Communications. Fundamentals and Applications. Prentice Hall, Upper Saddle River, NJ, 2001.

[17] Agilent, "Digital Modulation in Communications Systems - An Introductions", Agilent Technologies, Inc., 2001.

[18] Steven Kay. A Fast and Accurate Single Frequency Estimator. IEEE Trans. Acoust. Speech Signal Process. v37 i12. 1987-1990.

[19] P.J. Kootsookos. A Review of the Frequency Estimation and Tracking Problems. Systems Engineering Department, Australian National University, 1993.

[20] Jack E. Volder. The CORDIC Trigonometric Computing Technique, IRE Transactions on Electronic Computers. 1959.

[21] J. S. Walther. The Story of Unified CORDIC, VLSI Signal Processing 25, 107. 2000.

[22] Andraka, Ray. A survey of CORDIC algorithms for FPGA based computers.

[23] HomeRF Archive. http://www.cazitech.com/HomeRF Archives.htm.

[24] Xilinx, "ML505 Overview Setup", http://www.xilinx.com/products/boards/ml505/ml505_12.1/docs/ml505_overview_setup.pdf

[25] Xilinx ML505 Evaluation Platform Documentation: http://www.xilinx.com/products/boards/ml505/docs.htm

[26] ML505 Reference Designs: http://www.xilinx.com/products/boards/ml505/reference_designs.htm

[27] ML505: http://www.xilinx.com/support/documentation/ml505.htm

[28] Xilinx University Program XUPV5-LX110T Development Platform: http://www.xilinx.com/products/boards-and-kits/XUPV5-LX110T.htm

[29] Fichier des contraintes d'utilisateur: http://www.xilinx.com/univ/xupv5-lx110t/design_files/master_xupv5-lx110t.ucf

[30] Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/ug750.pdf

[31] Le générateur du signal RF N9310A: http://www.keysight.com/en/pd-748073-pn N9310A/rf-signal-generator-9-khz-3-ghz?cc=US&lc=eng

[32] Le double convertisseur numérique-analogique 10 bits (en quadrature): http://comblock.com/com2001.html

[33] Analyseur du signal EXA N9010 AEP : http://www.keysight.com/en/pd-1933439-pn-N9010AEP/EXA-Signal-Analyzer-Express-Configuration-Distributors-in-CN-KR?cc=US&lc=eng

[34] L'outil d'analyse du signal Vectorielle (VSA) AT 89601A : http://www.keysight.com/en/pd-1254738-pn-89601A/vector-signal-analysis-vector-modulation-analysis?cc=US&lc=e

[35] N9399C DC Block, 700 kHz to 26.5 GHz: http://www.keysight.com/en/pd-820242-pn-N9399C/dc-block-700-khz-to-265-ghz?cc=US&lc=eng

[36] Introduction aux convertisseurs Sigma Delta: http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html

[37] Agilent E3631A, Triple output DC power supply: http://www.keysight.com/en/pd-836433-pn-E3631A/80w-triple-output-power-supply-6v-5a-25v-1a?cc=US&lc=eng

Annexe A – L'algorithme de Kay et le module CORDIC

Lors de la conception, un système numérique doit trouver un compromis entre la précision de la représentation numérique du signal et les exigences matérielles d'implémentation. Une grande précision est essentielle pour minimiser les erreurs de phase de la démodulation, cependant, ça complique considérablement l'implémentation, ce qui augmente les coûts de la réalisation et la consommation d'énergie qui peut devenir inacceptable si l'objectif est la conception de dispositif à faible coût.

Dans le domaine numérique, le signal n'est pas représenté comme une variation continue dans le temps d'une grandeur électrique (courant ou tension), mais comme une succession d'échantillons, de laquelle, est associée une valeur numérique qui représente l'intensité du signal à l'instant d'échantillonnage. Cette valeur est obtenue après un processus de conversion du domaine analogique au numérique effectué en utilisant des circuits convertisseurs. Bien entendu, le processus de conversion n'est pas parfait et il entraîne une erreur, qui dépend du nombre de bits disponibles pour représenter un échantillon. Par conséquent, le choix du nombre de bits approprié, pour la représentation des échantillons des signaux, est crucial afin d'obtenir une réalisation qui réduit les erreurs de la démodulation avec la non nécessité d'une matérielle trop compliquée.

A.1 L'algorithme de Kay

Nous sommes intéressé à utiliser la version non pondérée de l'algorithme (pour plus d'informations, veuillez consulter le document original de Steven Kay dans [18]) et un retard de mesure entre les échantillons D = 1.

L'estimateur de Kay est donné par :

$$\hat{\omega} = \frac{1}{D} \sum_{t=0}^{N-2} w_t \angle(x_t * x_{t+D}) \tag{A.1}$$

Où N est le nombre d'échantillons utilisé dans l'estimation et D le retard de mesure entre les échantillons. La fonction des pondérations pour la version généralisée de l'estimateur de Kay est $\frac{1}{N-1}$, de sorte que l'estimateur possède la forme suivante:

$$\hat{\omega_0} = \frac{1}{N-1} \sum_{t=0}^{N-2} w_t \angle(x_t * x_{t+D}) = \frac{1}{N-1} \sum_{t=0}^{N-2} \angle(x_{t+1}) - \angle(x_t) \tag{A.2}$$

L'idée sur laquelle repose l'estimateur est que la fréquence est donnée par le décalage de phase entre le temps, c.-à-d. $w = \frac{\theta_D - \theta_0}{D}$ où w_D est la phase à l'instant D et w_0 à l'instant actuel et D le retard.

Le choix du paramètre D dépend du SNR. Un délai plus long équivalent à une estimation généralement meilleure et plus haut seuil de SNR. Dans notre cas, comme nous optimisons la zone et la consommation, nous faisons l'estimation à un faible SNR. Noté que l'estimateur dépend de beaucoup échantillons pour obtenir une bonne estimation. Ainsi, le paramètre D choisi est 1.

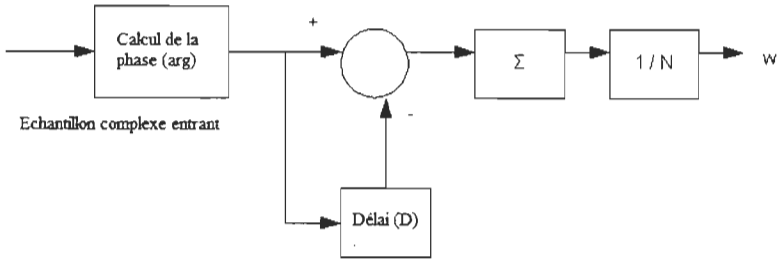


Figure A.1 : Diagrame des blocs de l'estimateur de Kay sans poids (adaptée de [18])

Le schéma synoptique de l'algorithme représenté sur la figure A.1. Le tableau suivant montre le résultat d'algorithme de Kay pour 8 échantillons complexes, où la division par le nombre d'échantillons est approximativement un décalage à droite (dans ce cas 3).

Tableau A-1 : Exemple d'estimation avec l'algorithme de Kay sur 8 échantillons (adapté de [18])

It.	I	Q	Θ_{t+1}	Θ_t	Σ	>> 3
1	0000001100	0000001100	0000011001	0000000000	0000011001	
2	0000010111	0000010111	0000011001	0000011001	0000011001	
3	0000011110	0000011110	0000011001	0000011001	0000011001	
4	0000100000	0000100000	0000011001	0000011001	0000011001	
5	0000011110	0000011110	0000011001	0000011001	0000011001	
6	0000001100	0000001100	0000011001	0000011001	0000011001	
7	0000000000	0000000000	0000000000	0000011001	0000000000	
8	1111110100	0000001100	0001001011	0000000000	0001001011	0000001001

Plus d'informations sur d'autres estimateurs de fréquence (y compris Kay) et différentes comparaisons de la performance, peuvent être trouvées dans [19].

A.2 Démonstration mathématique pour le module CORDIC

Le module CORDIC fait tourner une valeur complexe dans le plan, transforme un vecteur (xi, yi) à un autre (xj; yj) par un angle Θ (Figure A.2).

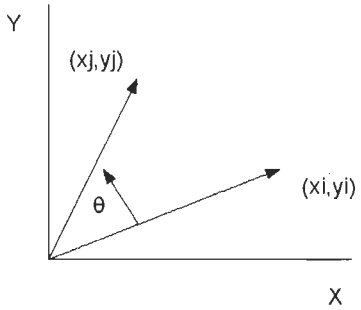


Figure A.2 : Rotation d'une valeur complexe par Θ

Une rotation dans le plan complexe peut être exprimée sous une forme matricielle :

$$\begin{bmatrix} X_j \\ Y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{A.3}$$

L'angle de rotation Θ peut fonctionner à petits pas de façon itérative, où à chaque itération s'exécute une rotation. Un pas est défini comme suit :

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{A.4}$$

Nous pouvons passer de quatre produits à trois en éliminant cos Θ_n

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{A.5}$$

Le reste des produits peut être éliminé en choisissant les angles des itérations de sorte que la tangente est une puissance de deux (la multiplication ou la division par une puissance de deux peut être implémentée avec un décalage).

L'angle pour chaque itération est exprimé comme :

$$\theta_n = \arctan\frac{1}{2^n} \tag{A.6}$$

La somme des angles de chaque itération est égale à l'angle Θ:

$$\sum_{n=0}^{\infty} S_n\theta_n = \theta \quad \text{Ou} \quad S_n = \{-1,+1\} \text{ et } \tan\theta_n = S_n 2^{-n}$$

Remplaçant : $$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \quad \text{(A.7)}$$

Ainsi, l'algorithme a été réduit à des déplacements et des additions, à l'exception du coefficient cos Θ_n qui peut être appliqué à la fin. Nous pouvons rapprocher la valeur du coefficient comme :

$$\cos\theta_n = \cos\left[\arctan\left(\frac{1}{2^n}\right)\right] \quad \text{(A.8)}$$

Et pour toutes les valeurs de n, nous multiplions les résultats, nous obtenons :

$$K = \prod_{n=0}^{\infty} \cos\left[\arctan\left(\frac{1}{2^n}\right)\right] \approx 0{,}607253 \quad \text{(A.9)}$$

Que nous considérons comme constante K. Maintenant, le module CORDIC peut être exprimé comme :

$$X_j = K(X_i \cos\theta - Y_i \sin\theta)$$
$$Y_j = K(Y_i \cos\theta + X_i \sin\theta) \quad \text{(A.10)}$$

Nous introduisons une nouvelle variable Z, qui représente la partie de l'angle de rotation qui n'est pas encore mis en rotation. Si à chaque rotation, S_n vaut la valeur -1 si Z_n est inférieur à zéro ou +1 si elle est supérieure en réduisant successivement la valeur de Z au voisinage de zéro par des valeurs de arc tangente déjà calculées (par exemple dans un tableau).

Nous obtenons le mode rotationnel du CORDIC (figure A.3) :

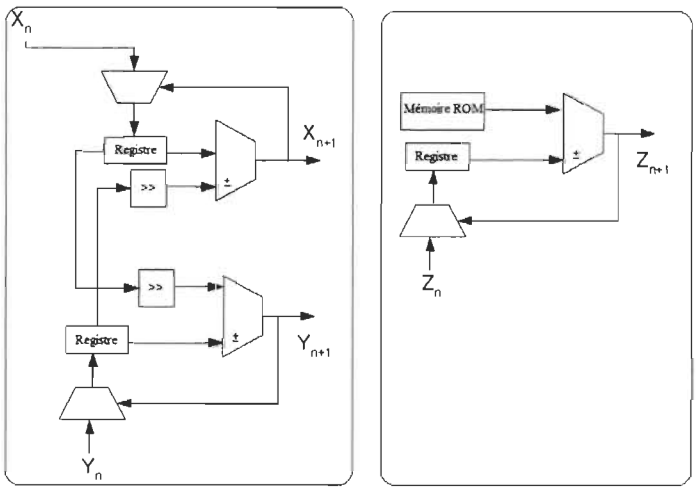


Figure A.3 : Structure d'implémentation itérative du CORDIC

Pour n = 0 jusqu'à ∞

Si $Z_n \geq 0$

$$X_{n+1} = X_n - \frac{Y_n}{2^n}$$
$$Y_{n+1} = Y_n + \frac{X_n}{2^n}$$
$$Z_{n+1} = Z_n - \arctan\left[\frac{1}{2^n}\right]$$

Sinon

$$X_{n+1} = X_n + \frac{Y_n}{2^n}$$
$$Y_{n+1} = Y_n - \frac{X_n}{2^n}$$
$$Z_{n+1} = Z_n + \arctan\left[\frac{1}{2^n}\right]$$

Application de la fin de la constante K.

Si au lieu de réduire Z à zéro, nous réduisons la valeur de Y, nous faisons face au mode vectoriel du CORDIC; qui nous permet de, calculer le module et la phase d'une valeur complexe : le module apparaît dans la valeur de X, puisque nous avons tourné le complexe d'origine sur l'axe des abscisses, la phase apparaît dans Z comme le cumul des angles additionnés ou soustraits selon la valeur de Y dans le processus.

Annexe B –Description des blocs internes

B.1 La description des blocs internes de l'émetteur / récepteur en bande de base

Dans ce qui suit, nous allons détailler la description en VHDL de chaque bloc interne de l'émetteur / récepteur en bande de base.

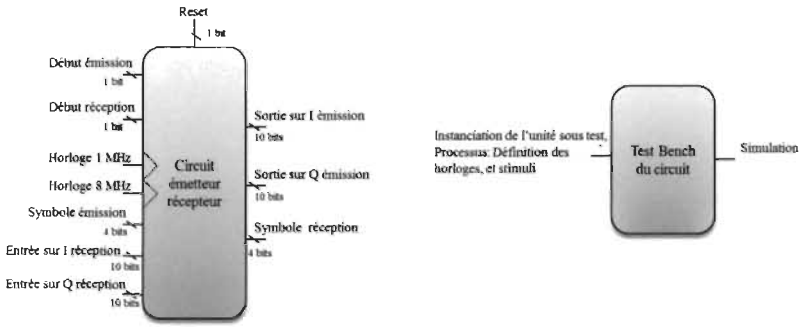


Figure B.1 : Le circuit global et son testbench

La figure B.1 illustre L'architecture globale de notre circuit complet pour une seule bande et un seul taux de transmission (en émission et en réception ou E/R) ainsi que son fichier du test pour visualiser les simulations, en effet, l'architecture comporte deux composants : composant chaîne émission et composant chaîne réception.

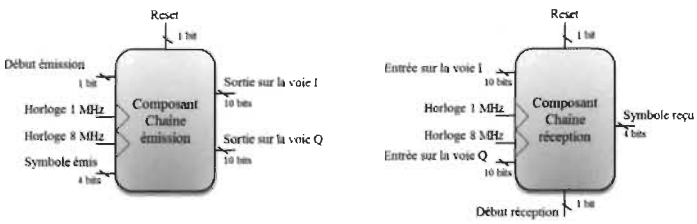


Figure B.2 : Les deux composants inclus dans le circuit global

Dans la figure précédente, nous voyons le composant qui englobe toute la chaîne d'émission, l'architecture de ce composant comporte : le composant Génération Chip (décomposition en I et Q), deux composants Sur-échantillonneur (pour les voies I et Q), et deux composants Filtre émission (pour les deux composantes I et Q). Nous trouvons aussi le composant qui rassemble toute la chaîne de réception, son architecture inclut deux composants Codage RZ (pour I et Q), un composant corrélateur symbole, deux composants Sous-échantillonneur (pour I et Q), et deux composants Filtre réception (dans les deux voies I et Q). Les deux horloges (1 MHz et 8 MHz) mentionnées sur la figure montrent le taux de transmission et le taux d'échantillonnage dû au sous-échantillonnage et au sur-échantillonnage.

Le bloc de génération Chip convertit chacun des symboles en une séquence PN de Chips. De là, nous avons deux flux de données distincts, l'un pour le canal I et l'autre pour le canal Q. Pour chaque symbole, nous obtiendrons deux séquences de 16 bits (32 Chips au total sans séparer les deux canaux avec un taux de 2 Mchips/s), résultant la séparation des bits paires et impaires (bits paires appartenant au canal I et les impairs au canal Q). Chaque séquence est formée par le même nombre des uns que des zéros, et toutes les séquences peuvent être générées à partir du symbole zéro faisant des rotations de 4 bits. Les séquences des 8 derniers symboles sont générées en inversant les bits pairs des séquences des 8 premiers symboles.

Nous pouvons penser à deux alternatives d'implémentation du générateur de séquence PN, l'un basé sur la simplicité et l'autre tente de réduire la surface de circuit :

- *Générateur de séquence PN basé sur une ROM* : nous pouvons stocker les 16 séquences correspondantes aux 16 symboles dans une mémoire ROM, et en

fonction de l'entrée (selon le symbole entrant), nous enlevons les Chips qui correspondent à la fréquence. L'implémentation est simple mais nous augmentons également la zone du circuit.

- *Générateur de séquence PN basé sur des tampons (buffers) circulaires* : toutes les séquences peuvent être obtenues moyennant de rotations à partir de la séquence du premier symbole (symbole zéro), nous chargeons deux tampons circulaires des séquences appartenant à la voie I et Q du symbole zéro. Selon le symbole entrant qui prend une position du tampon déterminée (correspondante à ce symbole) comme sortie, et en roulant les données, nous obtenons la séquence ciblée.

Où chacun des ports:

- *chip gen raz:* entrée de remise à zéro asynchrone du circuit.
- *chip gen horloge:* horloge de 1 MHz (taux de 2MChips/s entre les deux canaux).
- *chip gen debut:* signal indiquant que l'entrée du symbole est active.
- *chip gen sym valid:* Signal indiquant l'entrée d'un nouveau symbole.
- *chip gen sym:* entrée des symboles.
- *chip gen sortie i:* sortie de Chips correspondent au canal I.
- *chip gen sortie q:* sortie de Chips correspondent au canal Q.

Le codeur RZ fait la discrimination entre 0 ou 1 selon l'échantillon entrant. L'unité de réception dispose de deux codeurs RZ, un pour chaque canal. Ces ports sont :

- *entree cod rz i:* l'entrée des données du codeur RZ pour le canal I.

- *entree cod rz j:* l'entrée des données du codeur RZ pour le canal Q.
- *sortie cod rz i:* sortie des données du codeur RZ pour le canal I.
- *sortie cod rz j:* sortie des données du codeur RZ pour le canal Q.

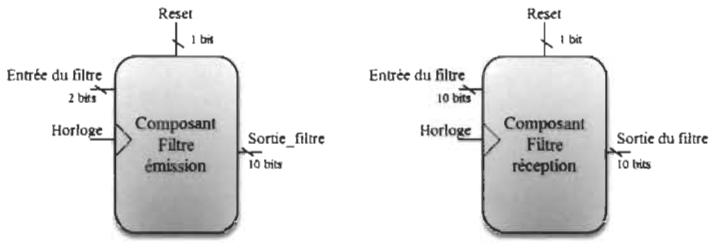


Figure B.3 : Les deux filtres (en émission et en réception)

Concernant le filtrage (figure B.3), nous avons implémenté le filtre de forme d'impulsion de demi-sinus comme structure du filtre RIF à huit coefficients [67]. Ses coefficients sont représentés sur 10 bits en complément à 2 (5 pour la partie entière et 5 pour la partie fractionnaire). Donc pour représenter un sinus, nous appliquons cette formule suivante : 32 x sin ((n*π)/8); n= 0..15, pour plus des détails sur le filtrage, consultez l'annexe précédent.

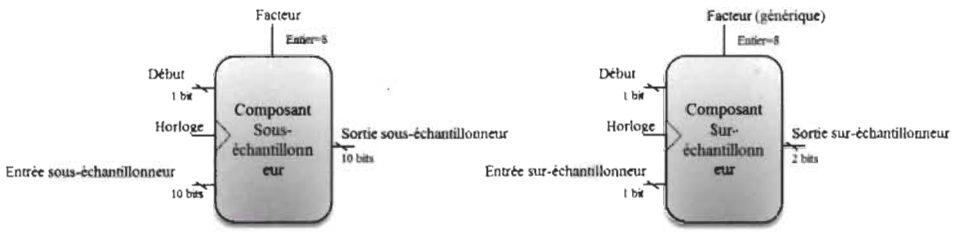


Figure B.4 : Le sur-échantillonneur et le sous-échantillonneur

Le but du sur-échantillonneur est d'insérer une série de zéros (7 dans notre cas) entre les impulsions entrantes, de sorte que les réponses du filtre FIR ne se chevauchent pas,

évitant ainsi les interférences entre symboles. L'unité de l'émission utilise deux sur-échantillonneurs, un par canal. Les ports sont :

- *sur_ech sortie:* sortie du sur-échantillonneur.
- *sur_ech entree:* entrée des données à sur-échantillonner.
- *sur_ech horloge:* horloge de 8 MHz.
- *sur_ech debut:* signal d'entrée qui indique de nouvelles données à traiter.

Le codeur NRZ convertit une entrée RZ en NRZ sur 2 bits, utilisant le complément à 2. Le codeur NRZ est implémenté au sein du sur-échantillonneur. Donc si l'entrée 0, la sortie vaut 11 et si l'entrée 1, la sortie vaut 01.

A l'émission, le canal Q est retardé par le demi de période symbole parce que le modulateur O-QPSK exige ce retard (dans notre cas 0.5 us). En pratique, nous faisons une boucle de 5 itérations sur l'horloge 8 MHz. Mais, si nous ne conservons pas ce retard nous parlons donc de la modulation QPSK.

Les ports pour le corrélateur de réception sont :

- *corr debut*: entrée qui active le corrélateur.
- *corr horloge:* entrée d'horloge du corrélateur.
- *corr raz:* entrée de remise à zéro asynchrone.
- *corr entree q:* entrée des données du corrélateur pour le canal Q.
- *corr sym valid:* sortie indiquant qu'un nouveau symbole est détecté.
- *corr sym:* sortie des symboles détectés par le corrélateur.

A la réception, Le signal début de corrélation du symbole est retardé par 2 us avant le début de détection des symboles, le canal I est retardé afin d'effectuer la démodulation au même temps que le canal Q (retardé par l'émetteur) arrive. Mais, si nous voulons implémenter la modulation QPSK, nous n'avons pas besoin de créer ce retard à la réception.

Le filtre d'émission est en liaison avec le sur-échantillonneur afin d'augmenter le taux d'échantillonnage en variant le facteur d'interpolation (par exemple, les facteurs : x8, x10, x12) ainsi que les coefficients du filtre.

Le filtre de réception est en liaison avec le sous-échantillonneur afin de réduire le taux d'échantillonnage en variant le facteur de décimation (par exemple, les facteurs : /8, /10, /12) et aussi les coefficients du filtre.

Le sous-échantillonneur élimine les zéros insérés dans le processus de modulation par le sur-échantillonneur. Deux sous-échantillonneurs dans l'unité de réception, un pour chaque canal.

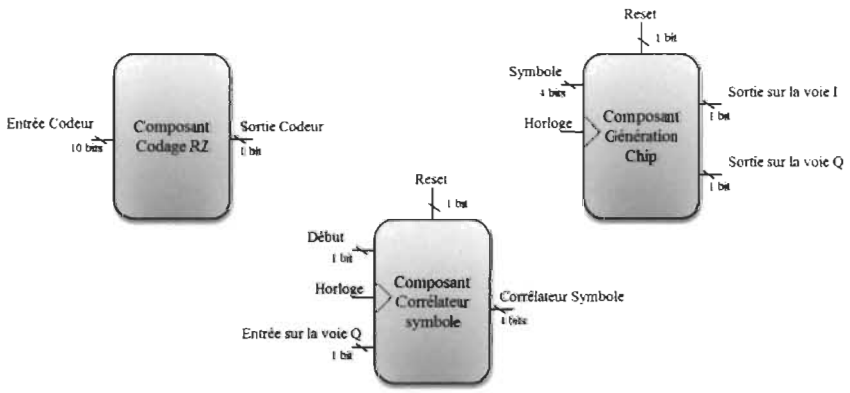


Figure B.5 : Le reste des composants inclus dans la description

Les trois composants restants (de la figure B.5) qui sont inclus dans la description comportent : le composant codage RZ (c'est un convertisseur du codage bipolaire NRZ à un codage unipolaire RZ afin de, recevoir nos symboles sur 4 bits avec 2 bits pour chaque canal). Le composant corrélateur symbole (sert à la synchronisation au moment de démodulation à la réception en plus de la détection du symbole reçu), et le composant génération Chip (c'est ici se fait la conversion du symbole au chip en plus de passer les chips paires sur le canal I et les chips impaires sur le canal Q en appliquant la technique d'étalement du spectre DSSS).

B.2 La description du diviseur d'horloge

L'horloge interne sur FPGA, est de 200 MHz et il est demandé de travailler en 1 MHz et 8 MHz. Pour le cas de 1 MHz, nous allons utiliser un compteur modulo 200 car 200MHz / 1 MHz = 200, tant que le compteur n'est pas égal à 199 c.à.d. de 0 à 199, nous incrémentons ce dernier à chaque front montant d'horloge à 200 MHz et à chaque fois que nous arrivons à 199, nous appliquons le signal inverse de celui obtenu à la fin du comptage précédent;

Nous faisons la même chose pour l'horloge de 8 MHz par un compteur modulo 25 car 200MHz / 8 MHz = 25, et il va aller jusqu'à 24.

B.3 La description de l'interface série ou le module émetteur/récepteur (UART)

Ce module joue le rôle d'une interface pour le port série RS232, afin de connecter un FPGA à un PC (le FPGA est relié uniquement à des broches de données TX et RX du port série).

Il sert à convertir la donnée série en une donnée parallèle, et la donnée parallèle en une donnée série. La donnée série, transférée à ce module, est placée sur un bus de sortie après que ce module la convertit en information parallèle. Ce bus peut ensuite être utilisé en tant qu'entrée logique à un circuit logique programmable FPGA à titre d'exemple. La donnée qui en résulte peut alors être renvoyée par le même module pour trouver en sortie une information série.

Ce module contient deux parties : une partie en réception qui gère la conversion série-parallèle, tandis que la partie en émission fait la conversion parallèle à série (N.B : émission c.à.d. de ce module vers le port RS232 et réception c.à.d. du port RS232 vers ce module).

B.4 La description du module séparation des données

Ce module sert à séparer les données (sur 8 bits) venant du module émetteur/récepteur UART en symbole émission de 4 bits, sélection bande sur 1 bit, remise à zéro sur 1 bit, début émission sur 1 bit et le bit de plus fort poids de Donnée reçue est à éliminer.

Exemple: Donnée reçue = 01011100 correspond à:

- ✓ Symbole émission [3..0] = Donnée reçue [3..0] = 1100
- ✓ Sélection bande = Donnée reçue [4] = 1
- ✓ Remise à zéro = Donnée reçue [5] = 0
- ✓ Début émission = Donnée reçue [6] = 1

B.5 La description du module adaptation de 4bits à 8bits

Ce module nous permet d'adapter le signal issu du symbole réception sur 4 bits vers le signal « donnée_TX_8bits » de 8 bits; donc il va utiliser seulement les 4 premiers bits de plus faible poids parmi les 8 bits, les 4 bits restants sont mis à 0;

Il est nécessaire pour l'acquisition des données sur le port série, par exemple: si notre symbole réception = 1001, donc le signal « donnée_TX_8bits » = 00001001.

B.6 La description du contrôleur d'émetteur / récepteur en bande de base

Cette partie, est spécifiée à la description de différents ports liés à chaque bloc du contrôleur d'émetteur/récepteur en bande de base.

B.6.1 Les ports du contrôleur général

Le fonctionnement de l'émetteur/récepteur nécessite des ports, ils sont inclus dans la spécification et le code utilisé (débogage); pour vérifier que toutes les unités fonctionnent correctement.

Les ports élémentaires de l'E/R :

- ✓ Horloge 1 de l'E/R : l'entrée d'horloge de 250 kHz pour synchroniser la transmission des données binaires de l'E/R (250 kbps).
- ✓ Horloge 2 du de l'E/R : l'entrée d'horloge de 1 MHz utilisée le générateur de Chip (1 Mchips par canal).
- ✓ Horloge 3 du de l'E/R : l'entrée d'horloge de 8 MHz utilisée par le sur-échantillonneur et le filtre FIR à réponse impulsionnelle de demi-sinus
- ✓ Remise à zéro de l'E/R : une remise à zéro asynchrone

- ✓ Mode de l'E/R : un port d'entrée pour indiquer le mode de fonctionnement de l'E/R, une combinaison sur 2 bits (IDLE 00, Boucle 01, TX 10, RX 11).
- ✓ Début transmission de l'E/R : un port d'entrée pour indiquer le début de la transmission.
- ✓ Entrée bit transmission de l'E/R : le port du signal d'entrée binaire à transmettre.
- ✓ Longueur transmission de l'E/R : l'entrée de valeur de longueur de la PDU à la transmission.
- ✓ Sortie transmission sur I de l'E/R : la sortie des données de l'émetteur à transmettre correspondant au canal I.
- ✓ Sortie transmission sur Q de l'E/R : la sortie des données de l'émetteur à transmettre correspondant au canal Q.
- ✓ Entrée réception sur I de l'E/R : l'entrée des données reçues de l'émetteur correspondant au canal I.
- ✓ Entrée réception sur Q de l'E/R : l'entrée des données reçues de l'émetteur correspondant au canal Q.
- ✓ Début réception de l'E/R : un port d'entrée pour indiquer le début de la réception.
- ✓ Sortie bit réception de l'E/R : un port de sortie des bits de réception du récepteur.

- ✓ SFD détecté à la réception de l'E/R : un port de sortie qui indique que le SFD de l'en-tête de la PDU a été détectée.

Les ports de débogage de l'E/R :

- ✓ Débogage bit à symbole en transmission de l'E/R : la sortie du débogage de la sortie du convertisseur bit à symbole.
- ✓ Débogage bit à symbole en transmission valide : la sortie du débogage indiquant que le convertisseur bit à symbole génère un symbole correctement.
- ✓ Débogage la sortie I de génération du chip en transmission : sortie du convertisseur symbole à chip sur le canal I.
- ✓ Débogage la sortie Q de génération du chip en transmission : sortie du convertisseur symbole à chip sur le canal Q.
- ✓ Débogage la sortie I du sur-échantillonneur en transmission : sortie du sur-échantillonneur sur le canal I.
- ✓ Débogage la sortie Q du sur-échantillonneur en transmission : sortie du sur-échantillonneur sur le canal Q.
- ✓ Débogage du contrôleur de l'émission du préambule en transmission : un port de sortie qui indique que le préambule a été envoyé par l'unité de transmission.
- ✓ Débogage du contrôleur de l'émission du SFD en transmission : un port de sortie qui indique que l'en-tête SFD a été envoyé par l'unité de transmission.
- ✓ Débogage du contrôleur de l'émission du PHR en transmission : un port de sortie qui indique que le paquet PHR a été envoyé par l'unité de transmission.

- ✓ Débogage du symbole en réception : sortie de symboles détectés par le corrélateur de réception de l'E/R .
- ✓ Débogage du symbole valide en réception : sortie du corrélateur de réception pour indiquer qu'un nouveau symbole a été détecté correctement.
- ✓ Débogage de l'estimation de la fréquence grossière faite en réception : un port de sortie qui indique que l'estimation « grossière » a terminé la mesure de l'entrée et elle commence à corriger les échantillons entrants.
- ✓ Débogage de l'estimation de la fréquence « fine » faite en réception : un port de sortie qui indique que l'estimation "fine" a terminé la mesure de l'entrée et elle commence à corriger les échantillons entrants.
- ✓ Débogage de la sortie I du sous-échantillonneur en réception : un port de sortie qui prend la sortie du sous-échantillonneur correspondant au canal I.
- ✓ Débogage de la sortie Q du sous-échantillonneur en réception : un port de sortie qui prend la sortie du sous-échantillonneur correspondant au canal Q.
- ✓ Débogage de l'estimateur de la fréquence « grossier » sur I en réception : sortie de l'estimateur grossier correspondant au canal I.
- ✓ Débogage de l'estimateur de la fréquence « grossier » sur Q en réception : sortie de l'estimateur grossier correspondant au canal Q.
- ✓ Débogage de l'estimateur de la fréquence « fin » sur I en réception : sortie de l'estimateur fin correspondant au canal I.

- ✓ Débogage de l'estimateur de la fréquence « fin » sur Q en réception : sortie de l'estimateur fin correspondant au canal Q.
- ✓ Débogage du codeur RZ sur I en réception : sortie du codeur RZ en réception correspondant canal I.
- ✓ Débogage du codeur RZ sur Q en réception : sortie du codeur RZ en réception correspondant canal Q.
- ✓ Débogage du filtre sur I en réception : sortie du filtre FIR en réception correspondant au canal I.
- ✓ Débogage du filtre sur Q en réception : sortie du filtre FIR en réception correspondant au canal Q.

B.6.2 Les ports du contrôleur d'émission

Le contrôleur d'émission possède les ports suivants :

- *contr début tx:* signal d'entrée indiquant le début de transmission.
- *contr remise à zéro:* signal de remise à zéro asynchrone.
- *contr horloge:* entrée d'horloge du contrôleur.
- *contr entrée bit:* entrée du flux binaire des données à moduler.
- *contr longueur:* entrée du paramètre longueur de la PDU.
- *contr sortie bit:* sortie des données à moduler.
- *contr bit à symbole act:* signal d'activation du convertisseur bit à symbole.
- *contr génér chip act:* signal d'activation du convertisseur symbole à Chip.

- *contr sur_ech act:* signal d'activation du sur-échantillonneur et du codeur NRZ.
- *contr debog pream em:* signal de sortie indiquant que le préambule est envoyé.
- *contr debog SFD em:* signal de sortie qui indique que le champ SFD est envoyé.
- *contr debog PHR em:* signal de sortie qui indique que le champ PHR est envoyé

B.6.2.1 Convertisseur de bit à symbole

Cette partie de l'unité d'émission reçoit l'entrée binaire à moduler d'un taux 250 kbps, et la convertit en groupes de 4 bits, symboles (la norme définit 16 symboles codés sur 4 bits), à un taux de 62,5 ksymboles/s. Où chaque port fait ce qui suit:

- *bit à sym debut:* entrée indiquant que le convertisseur devrait commencer à obtenir les symboles d'entrée.
- *bit à sym entre bit:* l'entrée bianire.
- *bit à sym horloge:* horloge à 250 KHz pour synchroniser avec l'entrée binaire.
- *bit à sym valid:* Indique un symbole valide est détecté et il est prêt à la sortie.
- *bit à sym sortie sym:* la sortie des symboles obtenus.

B.6.3 Les ports du contrôleur de réception

Les ports du contrôleur de réception sont :

- *contr debut rx: s*ignal de début de la réception.
- *contr sym valid:* le corrélateur utilise ce port pour aviser le contrôleur qu'un nouveau symbole est détecté.

- *contr sym corr:* le corrélateur utilise ce port pour aviser la contrôleur qu'un symbole est détecté (nous cherchons le premier symbole zéro du préambule détecté pour synchroniser avec l'émetteur).
- *contr estim freq gros fait:* l'estimateur de fréquence « grossier » utilise ce port pour avertir le contrôleur de terminer la mesure et de commencer la correction de la phase des échantillons entrants.
- *contr estim freq gros act:* pour activer l'estimateur de fréquence « grossier ».
- *contr estim freq fin act:* sortie pour activer l'estimateur de fréquence « fin ».
- *contr sous_ech act:* sortie pour activer le sous-échantillonneur.
- *contr corr act:* sortie pour activer le corrélateur de la réception.
- *contr sym à bit act:* sortie pour activer le convertisseur du symbole à bit.
- *contr SFD detec:* sortie pour indiquer que le SFD est détecté.

B.6.3.1 Estimateur de la fréquence « grossier »

Ce bloc comporte les ports suivants :

- *estim freq gros horloge:* horloge pour synchroniser avec les données entrantes.
- *estim freq gros raz:* remise à zéro asynchrone.
- *estim freq gros act:* signal d'activation pour activer l'estimation.
- *estim freq gros i:* entrée des données correspondante au canal I.
- *estim freq gros q:* entrée des données correspondante au canal Q.

- *estim freq gros estim fait:* signal de sortie indiquant que l'estimation est terminée et commence la correction de phase d'échantillons entrants.
- *estim freq gros ij:* sortie des données correspondante au canal I.
- *estim freq gros qj:* sortie des données correspondante au canal Q.

B.6.3.2 Estimation de la fréquence « fine »

Ces ports sont :

- *estim freq fin horloge:* entrée d'horloge de l'estimateur.
- *estim freq fin raz:* entrée de remise à zéro de l'estimateur.
- *estim freq fin act:* activation de l'estimateur pour commencer l'estimation.
- *estim freq fin i:* entrée des données du canal I.
- *estim freq fin q:* entrée des données du canal Q.
- *estim freq fin estim fait:* sortie indiquant que l'estimation est terminée et commence la correction des phases des échantillons entrants suivants.
- *estim freq fin ij:* sortie des données du canal I.
- *estim freq fin qj:* sortie des données du canal Q.

B.6.3.3 Convertisseur de symbole à bit

La dernière structure de l'unité de réception convertit les symboles à 4 bits vers le flux binaire d'origine envoyé par émetteur avec le taux de 250 kbps. Ces ports sont :

- *sym à bit debut:* signal d'entrée indiquant le début d'arrivée des symboles.
- *sym à bit horloge:* entrée d'horloge du convertisseur de 250 kbps.

- *sym à bit sym valid:* signal d'entrée qui indique l'arrivée d'un nouveau symbole sur le convertisseur est valide.
- *sym à bit sym:* entrée des symboles.
- *sym à bit sortie:* flux binaire à 250 kbps (données démodulées complètement).

B.6.4 L'unité auxiliaire de passage des données dans le modem

Cette unité comporte les ports suivants :

- *E/R tx i:* entrée des données modulées du canal I.
- *E/R tx q:* entrée des données modulées du canal Q.
- *E/R rx i:* sortie des données modulées du canal I à l'unité de réception.
- *E/R rx q:* sortie des données modulées du canal Q à l'unité de réception.

Annexe C – Le matériel utilisé

C-1. La plateforme d'évaluation

La figure C.1 représente notre plateforme d'évaluation, utilisée pour le test et la simulation réelle de l'architecture implémentée. Elle comporte beaucoup d'interfaces, dans notre cas, nous avons utilisé seulement GPIO Dip SW8, GPIO LEDs, le port série COM1 et XGI Expansion Headers. Nous avons fixé aussi le mode de configuration à travers Mode Pins DIP Switch (fixé à 00010101 afin de nous permettre de télécharger le fichier de configuration).

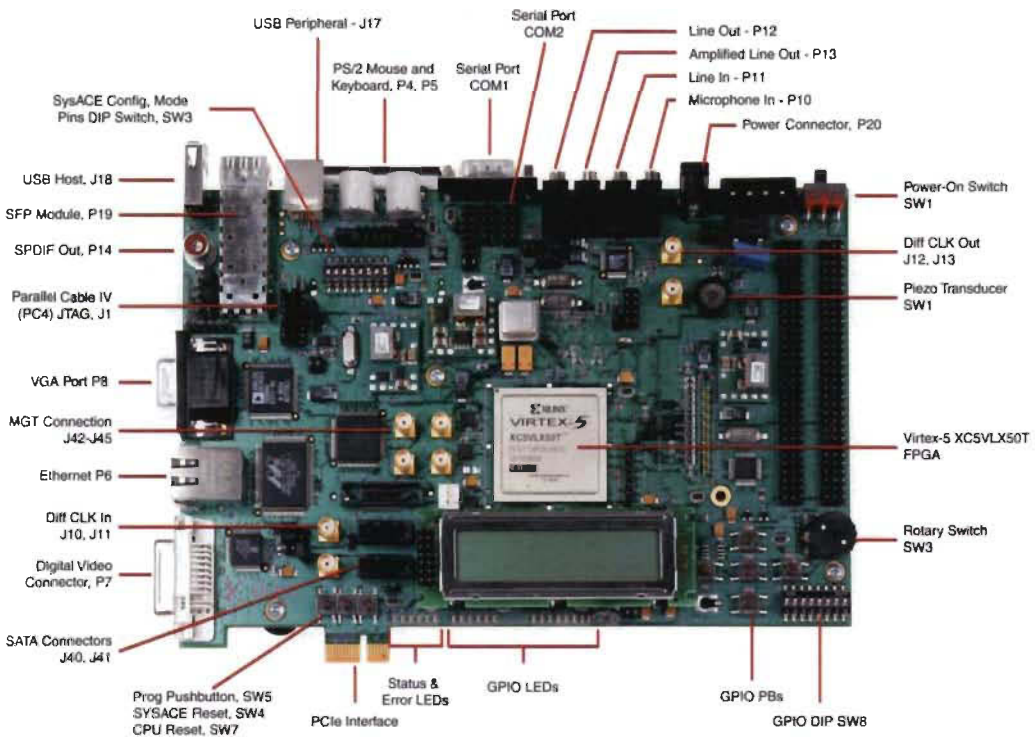


Figure C.1 : La plateforme d'évaluation XILINX XUPU5-LX110T ([25])

Le tableau C-1 contient les différentes horloges qui existent sur la plateforme d'évaluation. Dans notre cas, nous avons utilisé le port CLK_FPGA_P de 200 MHz qui représente l'horloge globale de notre architecture implémentée.

Tableau C-1 : Liste des horloges sur FPGA

Port	Localisation	Bank
CLK_27MHz_FPGA	AG18	4
CLK_33MHz_FPGA	AH17	4
User_CLK	AH15	4
CLK_FPGA_N	K19	3
CLK_FPGA_P	L19	3
CLKBUF_Q0_N	H3	116
CLKBUF_Q0_P	H4	116
CLKBUF_Q1_N	J19	3
CLKBUF_Q1_P	K18	3

Pour plus des détails, consultez le fichier des contraintes d'utilisateur de notre plateforme d'évaluation [29].

Les éléments matériels qui nous intéressent sur la plateforme d'évaluation sont les suivants : GPIO DIP Switches (tableau C-1), GPIO User LEDs (tableau C-2), XGI Expansion Headers (tableau C-3) et Serial port DCE RS232.

Tableau C-2 : GPIO User LEDs ([29])

Port	Localisation
GPIO LED 0	H18
GPIO LED 1	L18
GPIO LED 2	G15
GPIO LED 3	AD26
GPIO LED 4	G16
GPIO LED 5	AD25
GPIO LED 6	AD24
GPIO LED 7	AE24

Tableau C-3: GPIO DIP Switches ([29])

Port	Localisation
GPIO DIP SWITCH 0	U25
GPIO DIP SWITCH 1	AG27
GPIO DIP SWITCH 2	AF25
GPIO DIP SWITCH 3	AF26
GPIO DIP SWITCH 4	AE27
GPIO DIP SWITCH 5	AE26
GPIO DIP SWITCH 6	AC25
GPIO DIP SWITCH 7	AC24

Tableau C-4: XGI Expansion Headers ([29])

J6 Pin	Localisation	J6 Pin	Localisation
2	H33	34	W32
4	F34	36	AH34
6	H34	38	AE32
8	G33	40	AG32
10	G32	42	AH32
12	H32	44	AK34
14	J32	46	AK33
16	J34	48	AJ32
18	L33	50	AK32
20	M32	52	AL34
22	P34	54	AL33
24	N34	56	AM33
26	AA34	58	AJ34
28	AD32	60	AM32
30	Y34	62	AN34
32	Y32	64	AN33

Pour plus des détails, consultez le schématique de notre plateforme d'évaluation [25] ou même les références [26], [27], et [28].

C-2. Survole sur la liaison série

Une interface RS-232 présente les caractéristiques suivantes:

- Elle utilise un connecteur 9 broches « DB-9 »; avec seulement 3 fils, vous pouvez envoyer et recevoir des données: broche 2: RxD (réception de données), broche 3: TxD (transmission de données), et broche 5: GND (terre).
- Permet une communication bidirectionnelle full-duplex (le PC peut envoyer et recevoir des données en même temps).
- Peut communiquer à une vitesse maximale d'environ 10KBytes / s.
- Les données sont généralement envoyées par blocs de 8 bits (on appelle ça un octet) et elles sont sérialisées : le LSB (bit de données 0) est envoyé en premier, puis le bit de données 1, ... et le MSB (bit de données 7) est le dernier.
    - Les valeurs communes de vitesse sont les suivantes: 1200 bauds, 9600 bauds, 38400 bauds, 115200 (généralement la plus rapide), par exemple, à 115200 bauds, chaque bit dure (1/115200) = 8.7µs. Si nous transmettons des données de 8-bits, ça dure 8 x 8.7µs = 69µs. Mais chaque octet nécessite un bit de départ et un bit d'arrêt supplémentaires, de sorte que nous avons réellement besoin de 10 x 8.7µs = 87µs. Cela se traduit par une vitesse maximale de 11.5KBytes par seconde.

Annexe D – Autres résultats de simulation et de test

D-1. Résultats de la conception du diviseur d'horloge

La deuxième horloge générée (figure D.1) dépend du taux d'échantillonnage choisi au niveau de la conception (par exemple : une fréquence d'échantillonnage de 8 MHz, 10 MHz, ou 12 MHz), dans notre cas, nous avons opté à l'horloge de 8 MHz afin d'avoir un taux de 8 Mé/s.

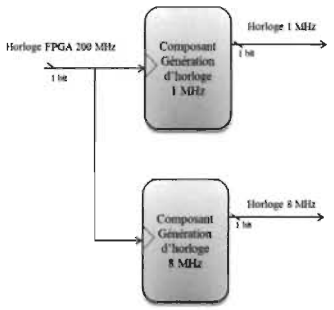


Figure D.1 : Génération d'horloge

La figure D.2 montre la simulation du générateur d'horloge obtenue utilisant ModelSim, d'après la simulation, le diviseur fonctionne très bien à travers la vérification de la période de chaque horloge (elle est calculée par les six curseurs) :

- Horloge 200 MHz: la période entre curseur 1 et curseur 2 est 5 ns
- Horloge 8 MHz: la période entre curseur 3 et curseur 4 est 125 ns
- Horloge 1 MHz: la demi période entre curseur 5 et curseur 6 est 500 ns x 2 = 1 us (période)

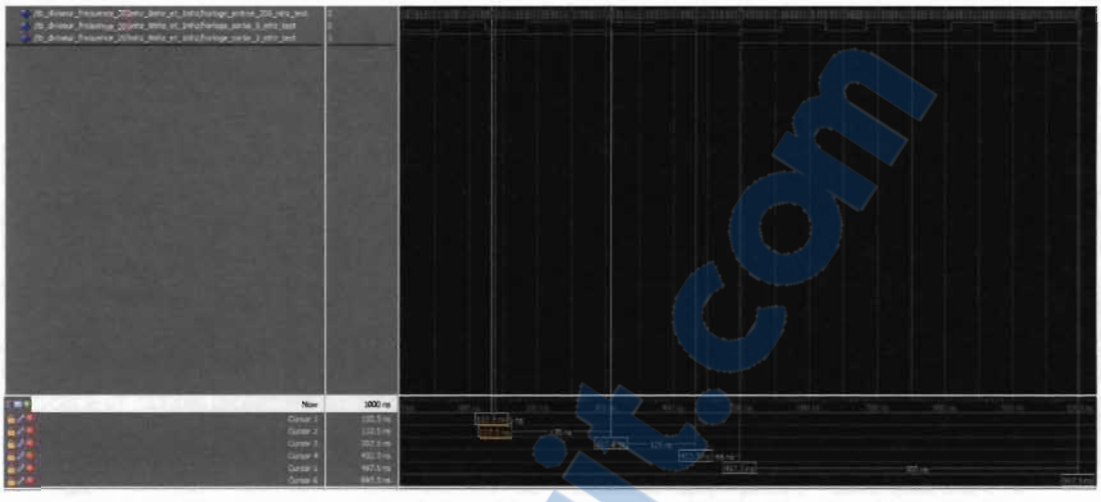


Figure D.2 : Simulation du générateur d'horloge sous ModelSim

Le tableau D.1 résume les ressources occupées par le générateur d'horloge, nous voyons bien qu'il occupe les entrées / sorties du circuit (une entrée de 200 MHz, deux sorties de 1 MHZ et 8 MHz).

Tableau D.1 : Ressources occupées du générateur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	42	69120	0 %
Nombre de LUTs	98	69120	0 %
Nombre de paires LUT-FF	41	99	41 %
Nombre de blocs E/S	3	640	0 %
Nombre de BUFGs/BUFGCTRS	1	32	3 %

La figure D.3 dresse notre circuit sous forme d'un schématique obtenu au niveau de la synthèse utilisant l'outil Xilinx ISE.

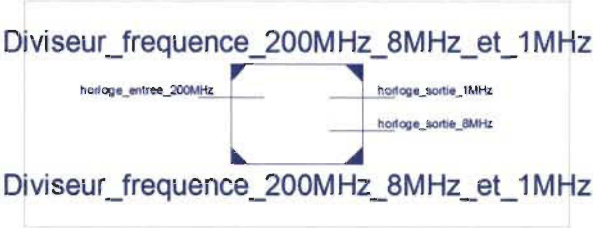


Figure D.3 : Le schématique (RTL) pour l'horloge

Après l'ajout du fichier de connexion pour l'analyseur logique intégré dans notre projet de description sous l'outil Xilinx ISE, en effet, l'unité d'analyseur comporte un port d'horloge globale (celui de l'FPGA, dans notre cas : 200 MHz) et deux ports de détection (à l'écoute) qui sont les deux sorties d'horloges.

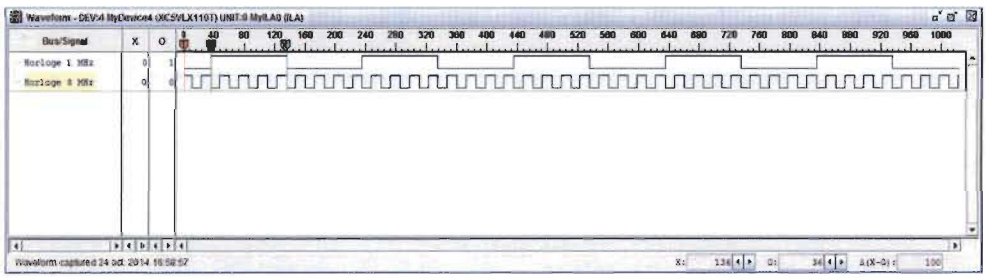


Figure D.4 : Simulation réelle du générateur d'horloge

La figure D.4 représente la simulation réelle de deux horloges sur l'outil Chipscope Pro après le téléchargement du fichier de configuration. Ça nous confirme le bon fonctionnement du générateur d'horloge, en effet, pour l'horloge 1 MHz : la demi période entre le curseur X et le curseur O est Δ=100 x 2 = 200 (période). Donc la période en seconde est : $\Delta t = 2 * \Delta / 200$ MHz = 1 us (la même chose pour 8 MHz : 2*Δ=25).

D.2 Emetteur en bande de base à double bande

Dans cette sous-section, nous présenterons les résultats obtenus de la description d'émetteur en bande de base à double bande c.-à-d. nous prenons en compte de la bande 915 MHz et la bande 2.45 GHz.

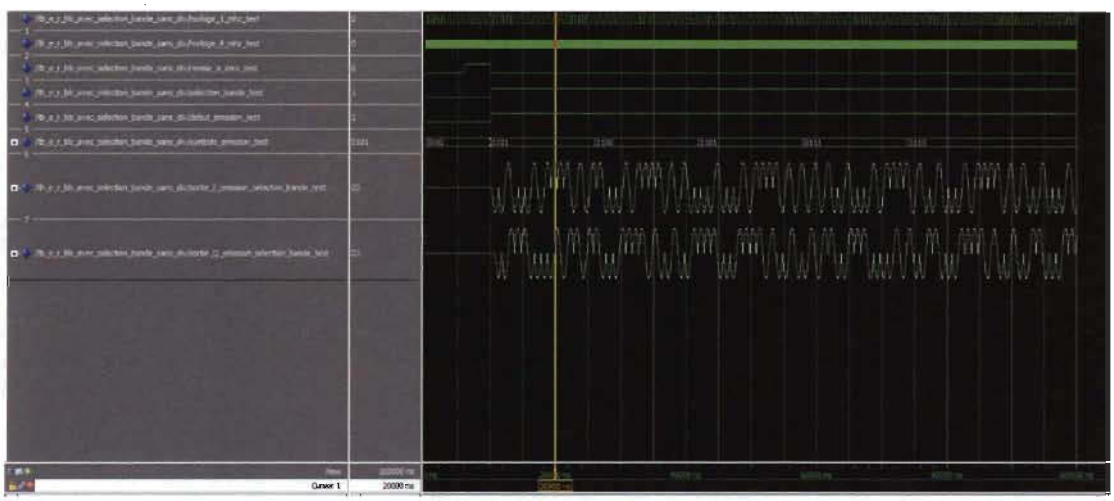


Figure D.5 : Simulation sous ModelSim

La simulation par ModelSim (figure D.5) montre le bon fonctionnement de la description de l'émetteur en bande de base en sélectionnant la bande concernée; chaque symbole émis correspond à des ondulations durant toute la période symbole et nous avons choisi la bande 2.45 GHz au niveau de la simulation.

Tableau D.2 : Ressources occupées par l'émetteur à double bande

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	240	69120	0 %
Nombre de LUTs	441	69120	0 %
Nombre de paires LUT-FF	127	554	22 %
Nombre de blocs E/S	24	640	3 %
Nombre de BUFGs/BUFGCTRS	4	32	12 %

Le tableau D.2 nous dresse les ressources occupées au niveau de la synthèse par l'émetteur en bande de base à double bande.

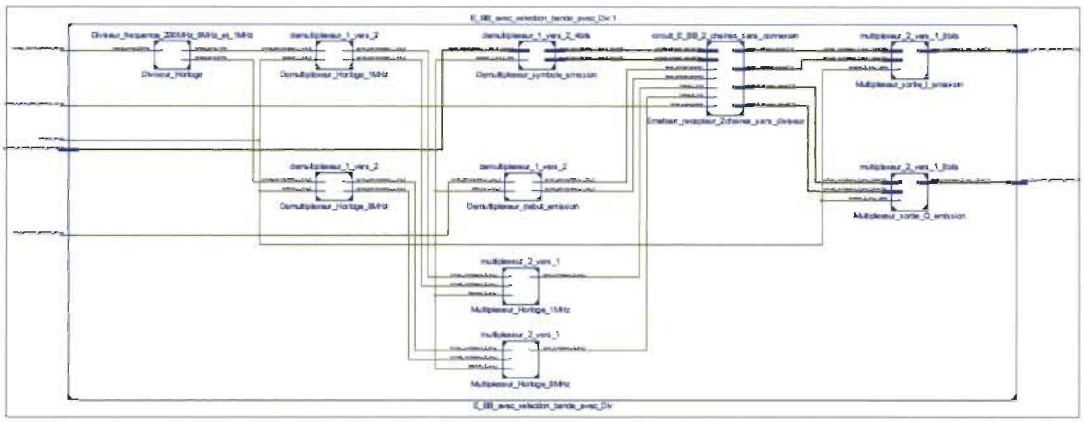


Figure D.6 : Le schématique de l'émetteur à double bande

Le schéma précédent représente les circuits internes de l'émetteur à double bande décrits en VHDL, ce qui nous confirme l'interconnexion des blocs dans le circuit.

D.3 L'interface série ou l'émetteur / récepteur UART

Dans cette partie, nous allons exposer les résultats obtenus par la description en VHDL de l'émetteur / récepteur UART.

Tableau D.3 : Ressources occupées par UART

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	58	69120	0 %
Nombre de LUTs	105	69120	0 %
Nombre de paires LUT-FF	57	106	53 %
Nombre de blocs E/S	22	640	3 %
Nombre de BUFGs/BUFGCTRS	1	32	3 %

Le tableau D.3 nous dresse le ressources estimées au niveau de la synthèse pour le module émetteur / récepteur UART, il consomme plus de LUT-FF (plus que moitié des ressources disponibles).

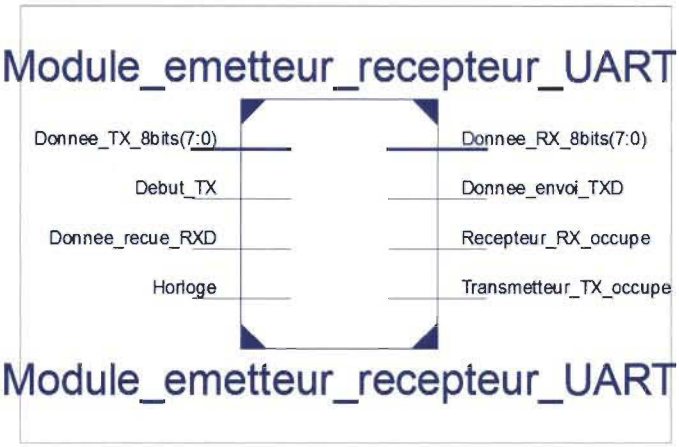


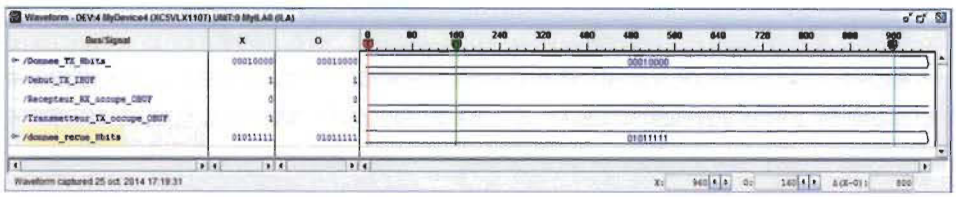
Figure D.7 : Le shématique du module émetteur / récepteur UART

La figure D.7 comporte le bloc interne décrit en VHDL du module émetteur / récepteur UART avec ses entrées / sorties.

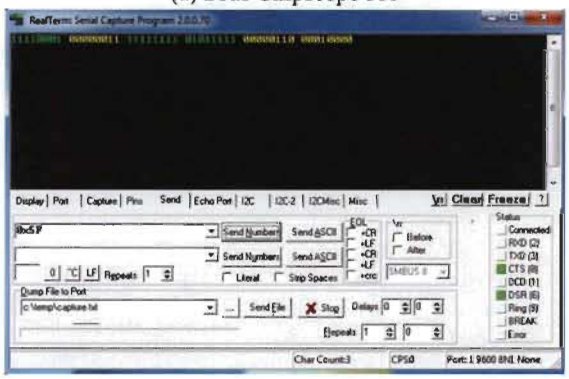
Au niveau de test réel, nous avons fixé la données_RX_8bits sur les LEDs de la plateforme d'évaluation ainsi que la donnée_TX_8bits sur les Dip Switchs. L'entrée et la sortie principale de ce module sont liées directement avec les broches TX et RX du port série. Les sorties TX_occupé et RX_occupé sont connectées sur des LEDs pour indiquer ses occupations.

Au niveau du fichier de connexion pour l'analyseur logique intégré, nous avons configuré 200 MHz comme port d'horloge principale et cinq autres ports de détection pour les visualiser sur Chipscope Pro (port 1 : donnée_TX_8bits; port 2 : donnée_RX_8bits; port 3 : début_TX; port 4 : TX_occupé; et port 5 : RX_occupé).

Nous faisons un test réel du module UART par un logiciel de capture sur le port série en envoyant d'une part des données vers les LEDs et acquérant des données prélevées des états des Dip Switchs.



(a) Sous Chipscope Pro



(b) Sous RealTerm

Figure D.8 : Visualisation des résultats

La figure D.8 nous visualise les résultats issus de l'outil Chipscope Pro (en (a)) et de l'outil RealTerm (en (b)) ; en effet ; la dernière donnée envoyée par RealTerm ((la donnée émise par ce logiciel est toujours en vert et la donnée reçue est en jaune) est 01011111 qui est la même affichée dans (a), de même, la dernière donnée reçue en (b) est 00010000 qui correspond bien à la donnée affichée en (a).

Il y a un autre moyen de test du module émetteur / récepteur UART implémenté par la boucle de retour c.-à-d. nous avons injecté (dans la description) la sortie donnée_RX_8bits dans l'entrée donnée_TX_8bits comme le montre la figure D.9.

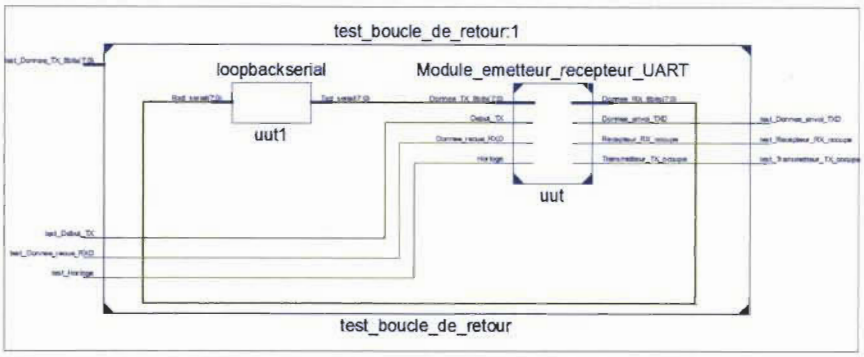


Figure D.9 : Le schématique du module avec la boucle de retour

Donc, nous procédons au test réel, le fait d'envoyer et de recevoir la même donnée moyennant un test de la boucle de retour par l'outil RealTerm (figure D.10).

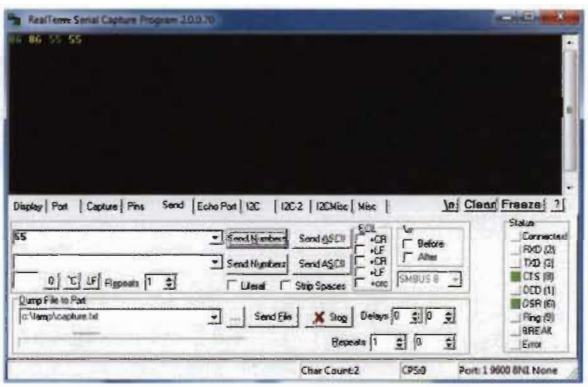


Figure D.10 : Résultat sur l'outil RealTerm

D'après RealTerm, le module implémenté fonctionne bien par :

- La première donnée envoyée est 86 qui est la même en retour (à la réception) ;
- La dernière donnée envoyée est 55 qui est la même en retour (à la réception);

N.B : L'émission du module UART correspond à la réception par RealTerm et la réception du module UART correspond à l'envoi par RealTerm.

D.4 Le module séparation des données

A ce stade, nous produirons les résultats obtenus par la description du module séparation des données.

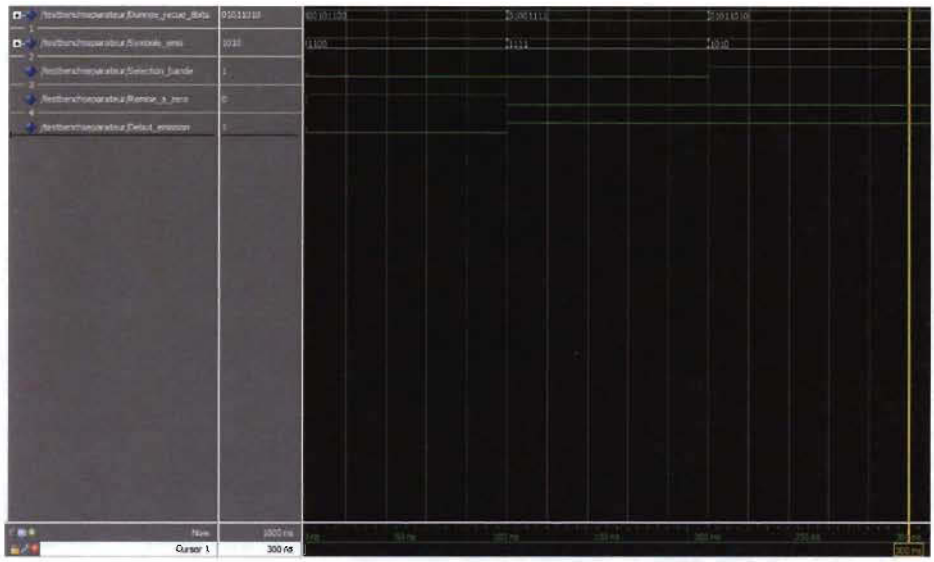


Figure D.11 : La simulation sur ModelSim

La simulation sur ModelSim (figure D.11) nous montre le bon fonctionnement de ce module et pour chaque donnée reçue, les sorties reçoivent les bits concernés.

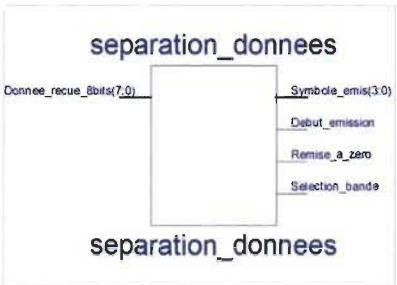


Figure D.12 : Le schématique du module séparation des données

La figure D.12 représente le circuit interne décrit en VHDL sachant que l'entrée « Donnée_reçue_8bits » est liée avec les Dip switchs ainsi que les sorties sont connectées aux Leds utilisateurs sur la plateforme.

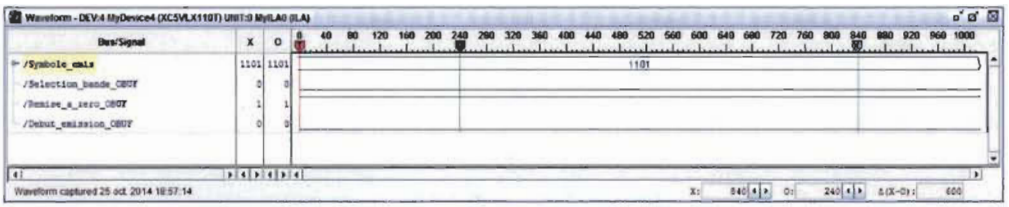
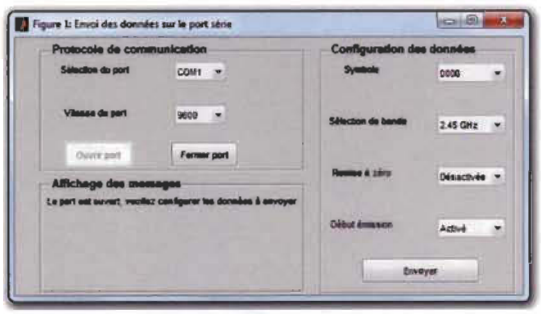


Figure D.13 : Résultat sur Chipscope Pro

Après l'attribution des ports de détection dans le fichier de connexion, nous avons vérifié l'état des Leds en sortie à chaque fois nous activons ou nous désactivons les Dip switchs que ce soit sur la plateforme ou même sur Chipscope Pro (figure D.13).

D.5 Test avec l'application d'envoi des données sur le port série

A ce niveau, nous montrerons les résultats obtenus par le test de l'interface graphique afin d'envoyer des données sur le port série.



(a) Lors d'ouverture du port

(b) Lors de la fermeture du port

Figure D.14 : Fonctionalités de l'interface graphique

La figure D.14 illustre la capacité de l'interface graphique développée sous GUIDE de Matlab tel qu'en (a) nous avons appuyé sur le bouton ouvrir port et nous avons fermé le port en (b).

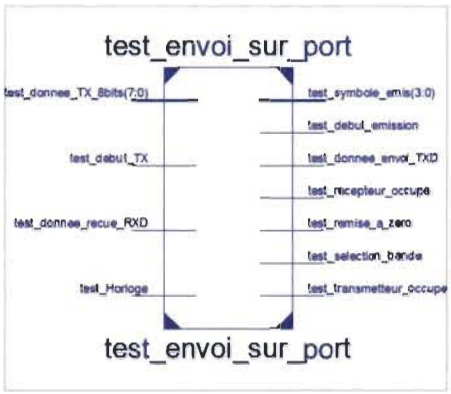


Figure D.15 : Le schématique du module de test d'envoi sur le port

La figure précédente, correspond au circuit interne du module de test d'envoi sur le port série, ses entrées / sorties sont attribuées ensuite aux ports de détection configurés dans le fichier de connexion.

D.6 Étude de la chaîne en modulation BPSK

Dans cette partie nous nous intéresserons en premier lieu à la modélisation de la chaine en BPSK avec étalement du spectre sous Matlab Script; pour atteindre les objectifs suivants : voir le principe de fonctionnement du système sans codage canal; vérifier l'acheminement de l'information dans chaque bloc compris dans le système (étalement, modulation, démodulation, dés-étalement); et analyser la performance du système : le taux d'erreur binaire (en variant le rapport signal sur bruit).

Les étapes du code source en Matlab script se résument en huit points essentiels :

1. Génération des données aléatoires pour le message d'information binaire de longueur 100 ;
2. Génération de la séquence binaire à bruit pseudo aléatoire (PN) de longueur 15 ;
3. Etalement du spectre utilisant DSSS : la fonction ou-exclusive entre le message d'information et la séquence PN ;
4. Modulation en BPSK du message d'information étalé avec la fréquence d'échantillonnage = 3 * la fréquence porteuse ;

- Analyse de performance (TEB) en variant le rapport signal sur bruit RSB (une boucle) :

5. Ajout du canal AWGN au signal à émettre (en fonction du RSB) ;
6. Démodulation du signal reçu avec filtrage ;
7. Corrélation avec la séquence pseudo-aléatoire et dés-étalement du signal reçu ;
8. Décision et calcul d'erreurs

La figure D.16 nous montre les simulations sous Matlab script de la chaîne complète en BPSK, en effet, en (a) le message d'information (génération aléatoire) de longueur 100; en (b) la séquence à bruit pseudo aléatoire de longueur 15 ; en (c) le message étalé avec DSSS de longueur 1500 (100*15); en (d) le message modulé en BPSK à émettre; en (e) le message données reçu bruité (AWGN) pour RSB = 7; en (f) le message reçu démodulé et filtré pour RSB = 7; en (g) le message reçu dés-étalé pour RSB = 7 → Conforme au message d'information; et en (h) l'analyse de performance : Taux d'erreurs binaire (pour un canal AWGN vs Théorique).

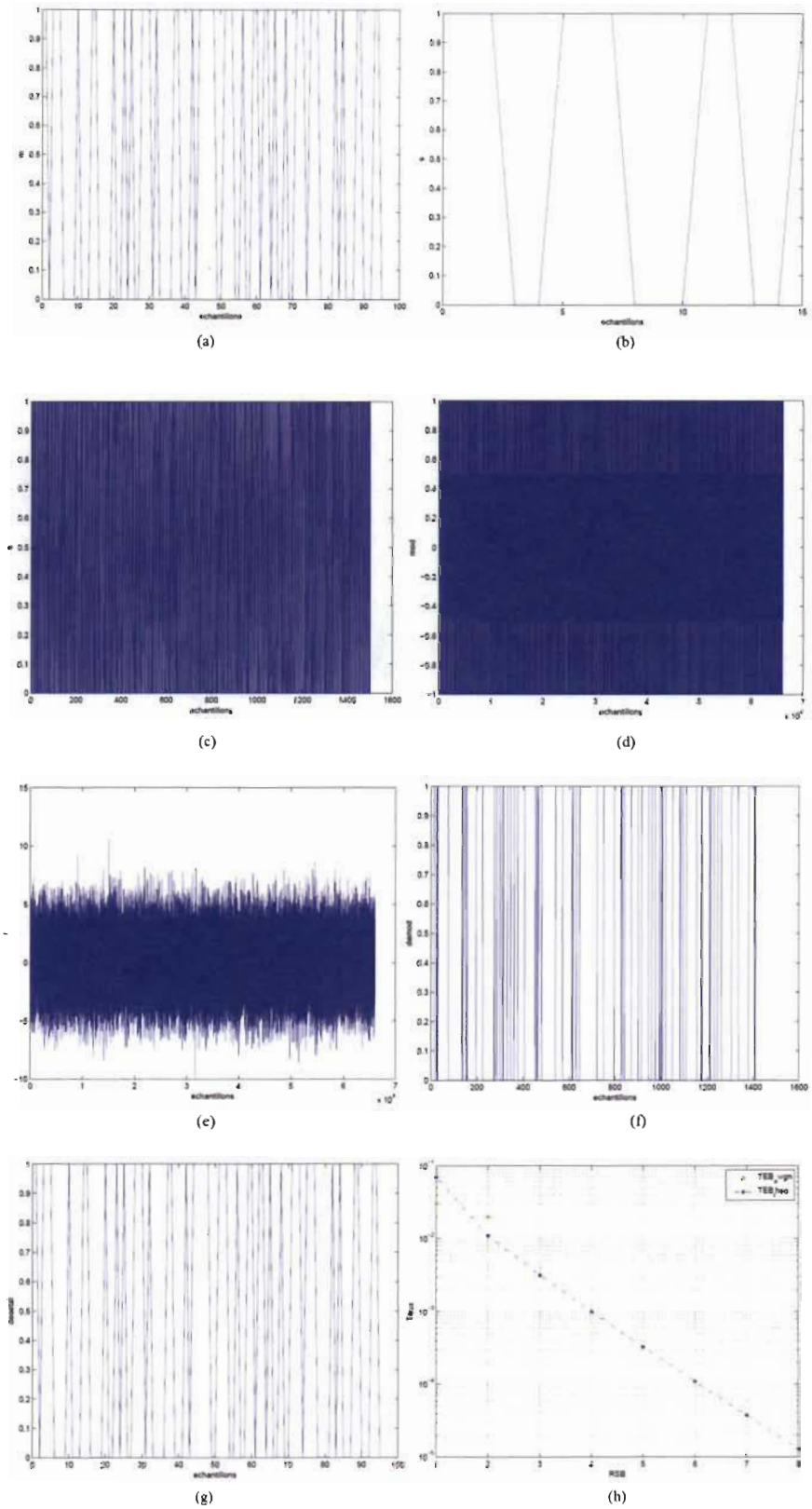


Figure D.16 : La modélisation de la chaine en BPSK

La deuxième partie dans cette section est consacrée à l'implémentation du codeur / décodeur différentiel en VHDL.

Premièrement, nous avons introduit les équations correspondantes sous Matlab Script afin de les tester le fonctionnement et nous avons obtenus les résultats sur la figure D.17.

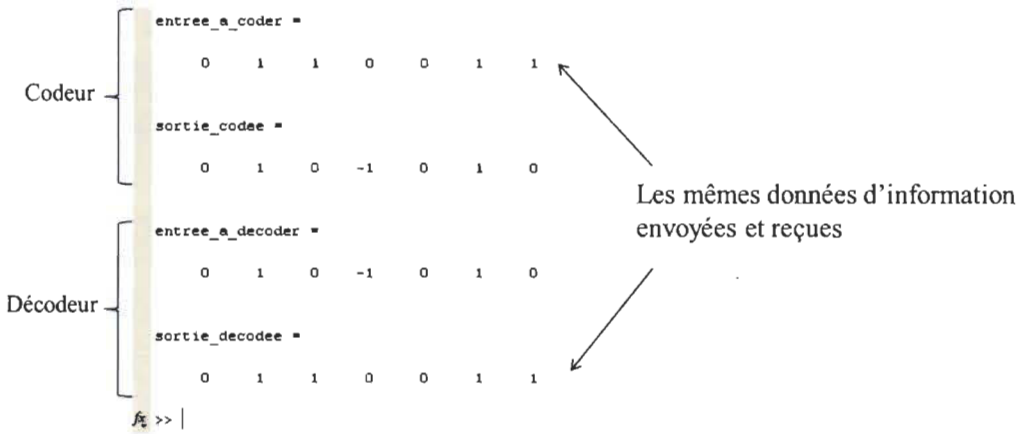


Figure D.17 : Les résultats obtenus sous Matlab script

Nous remarquons bien que nous obtenons les mêmes valeurs injectées dans le codeur et la sortie du décodeur.

Deuxièmement, nous décrivons l'architecture du codeur et du décodeur en VHDL comme le montre la figure D.18 par les entrées / sorties de deux blocs.

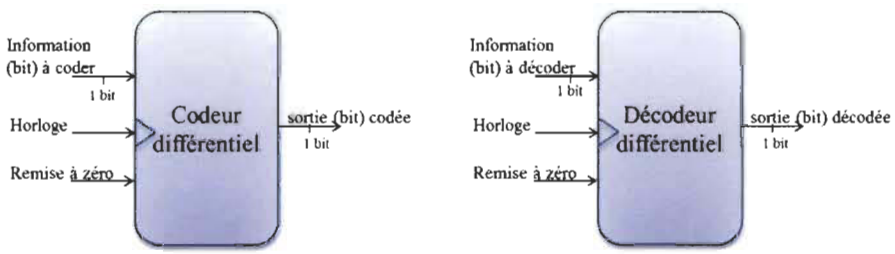


Figure D.18 : Les entrées / sorties de deux blocs

La description de deux blocs contient des opérations d'Ou-exclusif et des blocs de mémorisation qui sont construit par les bascules D.

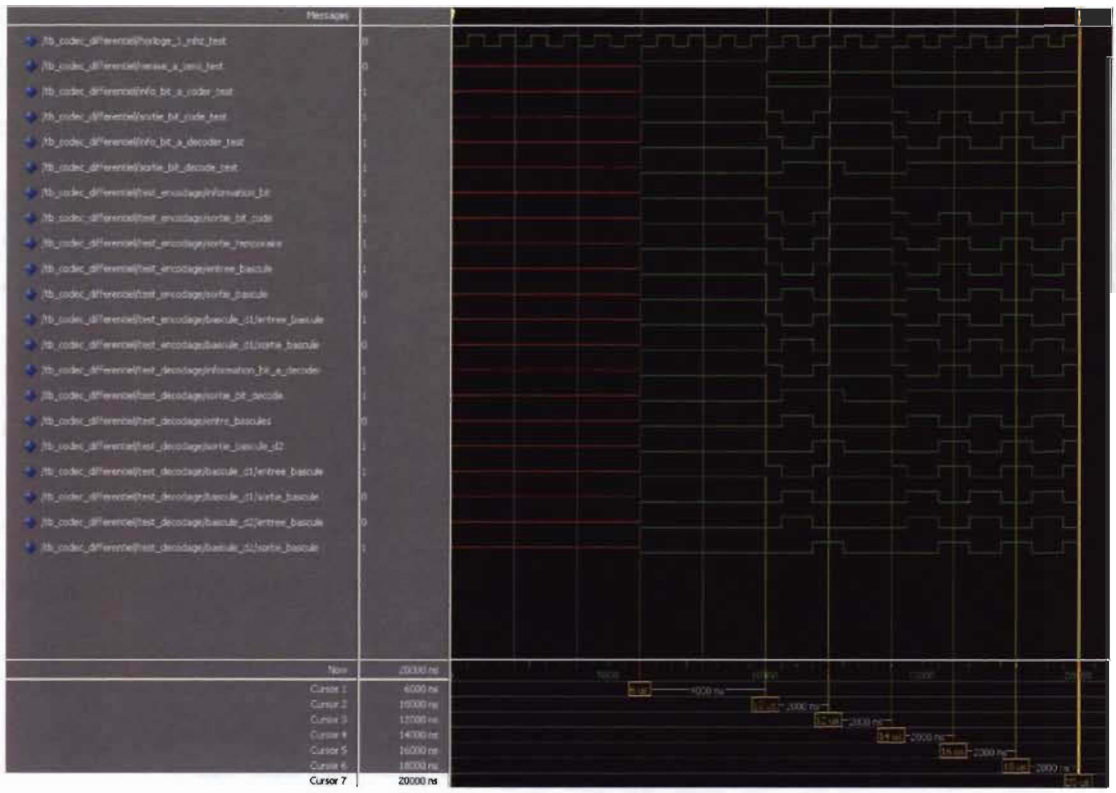


Figure D.19 : Les résultats de simulation obtenus sous ModelSim

Dans le test, nous avons injecté le message d'information à l'entrée du codeur : 0 1 1 0 0 1 1 et nous avons obtenu le même message à la sortie du décodeur, sachant que nous avons fait la liaison entre la sortie du codeur et l'entrée du décodeur;

Depuis la simulation sous ModelSim, nous remarquons que nous avons un certain retard de résultat, il est dû à l'utilisation de mémorisation du bit précédant à travers les bascules D, donc à chaque front montant d'horloge nous avons un nouveau résultat;

Le tableau D-4 dresse les ressources utilisées par le codeur / décodeur différentiel au niveau de l'implémentation.

Tableau D-4 : Les ressources occupées par le codeur / décodeur

Ressources	Occupées	Disponibles	Utilisation
Nombre de registres	2	69120	0 %
Nombre de LUTs	2	69120	0 %
Nombre de paires LUT-FF	0	4	0 %
Nombre de blocs E/S	4	640	0 %
Nombre de BUFGs/BUFGCTRS	1	32	3 %

La figure ci-dessous représente le circuit interne au niveau RTL qui comporte les deux blocs décrits.

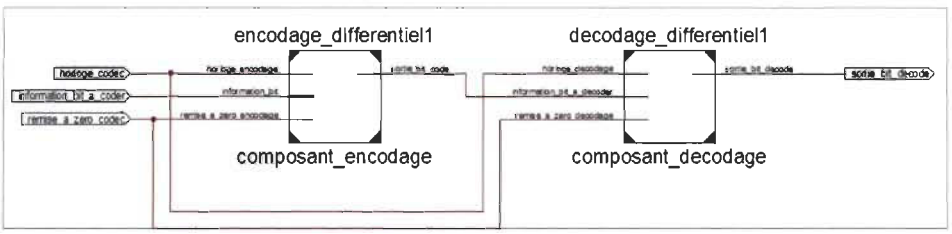


Figure D.20 : Le schématique du codeur / décodeur au niveau RTL

Après avoir créé le fichier des contraintes en associant les entrées / sorties de blocs aux pins adéquats ; et après l'ajout du fichier de connexion pour le test réel (le port d'horloge globale et les ports de détection), nous avons testé notre architecture sous l'outil Chipscope Pro. Nous avons obtenu les résultats de test représentés sur la figure D.21 tel qu'en (a) nous avons activé la remise à zéro pour les bascules (la sortie bit code est la sortie du codeur). En (b) nous avons désactivé la remise à zéro ainsi que l'envoi du premier bit ('1') sur l'information bit à coder, et la réception du premier bit ('1') sur la sortie bit décode ; en (c) l'envoi du deuxième bit ('0') sur l'information bit a coder, et la réception du deuxième bit ('0') sur la sortie bit décode ; et ainsi de suite en (d).

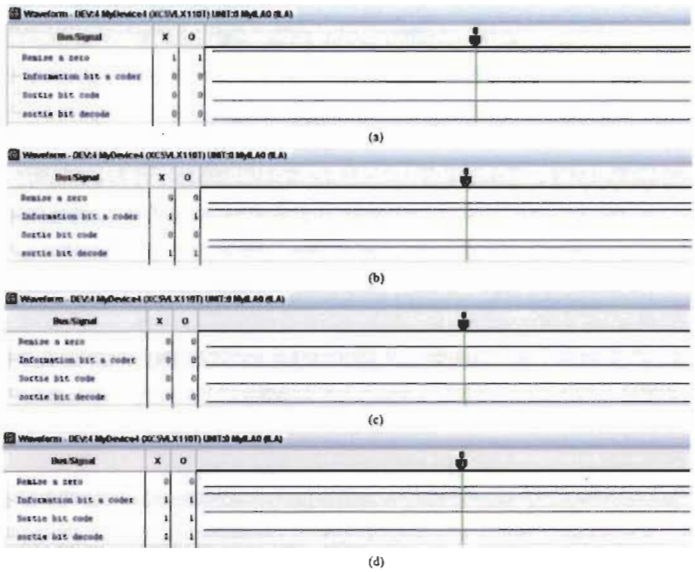


Figure D.21 : Test sous Chipscope Pro

La troisième partie étudiée, est chargée d'implémenter le modulateur BPSK en VHDL afin de connaitre le principe de fonctionnement de ce bloc. Son principe consiste à générer des données binaires (bloc générateur des données, synchronisé avec le bloc modulateur), injecter ces données (séries) dans l'émetteur, et visualiser les échantillons (sur 10 bits) sur la sortie. Par contre, le code VHDL n'est pas synthétisable dû à l'utilisation de la bibliothèque 'real math' afin de construire la porteuse (fonction sinus) → donc il y en a plus d'implémentation sur FPGA.

Le bloc générateur des données comporte quatre registres, afin de, générer une séquence pseudo aléatoire de longueur 15. Le modulateur BPSK sert à manipuler les réels et à convertir les valeurs réelles vers des valeurs binaires → création de la porteuse → création du signal modulé suivant la donnée injectée dans le modulateur.

Figure D.22 : Résultats obtenus du modulateur BPSK

Suivant la figure D.22, nous constatons bien que notre modulateur en BPSK fonctionne très bien au point de vue simulation comme il est expliqué dans l'annexe A.

D-7. Résultats de la conversion numérique analogique

L'idée est, d'envoyer des symboles via la liaison série, à partir de, l'application développée sous GUIDE de Matlab vers l'émetteur en bande de base. Il est connecté à travers ses sorties (sortie émission sur I et sortie émission sur Q) avec les entrées numériques (entrée 1 modulateur et entrée 2 modulateur) du convertisseur numérique-

analogique (qui est basé sur un double modulateur Sigma-Delta décrit en VHDL). Pour plus des détails sur le modulateur, consultez la référence [36].

Figure D.23 : Schéma synoptique

Les sorties du modulateur sont sur 1 bit (donc soit 0 ou 1) en valeurs signés. Elles sont connectées avec les entrées de deux filtres RC (réalisés sur la platine d'essai) afin d'obtenir en sortie les signaux analogiques. Les deux signaux analogiques sont sondés par deux sondes analogiques pour pouvoir les visualiser sur l'oscilloscope.

Le filtre RC a les propriétés suivantes: U_e = 2.5 V; R=2.2 kΩ; C= 0.1 nF; τ=0.22 µs; F_c= 750 kHz et donc ΔN=-3dB ➔ U_s = 2.5*0.707 = 1.76 V.

La figure D.24 présente le schématique au niveau RTL qui comporte les blocs décrits en VHDL avec les interconnexions.

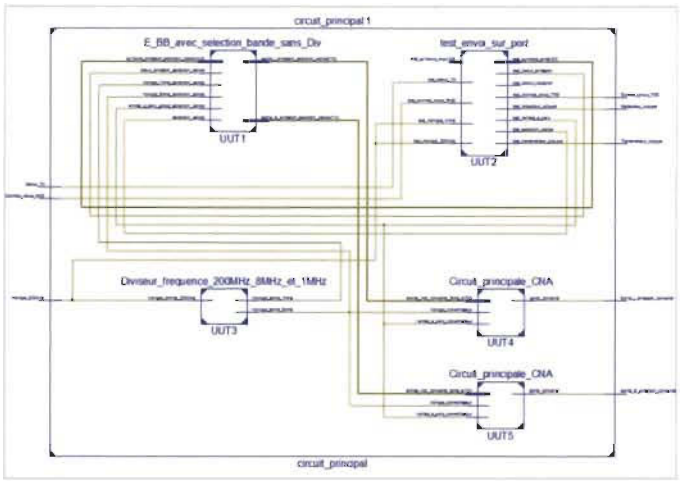


Figure D.24 : Le circuit niveau RTL

Nous avons associé, dans le fichier des contraintes, les deux sorties principales (sur 1 bit chaque) les pins H33 et AN33 du FPGA.

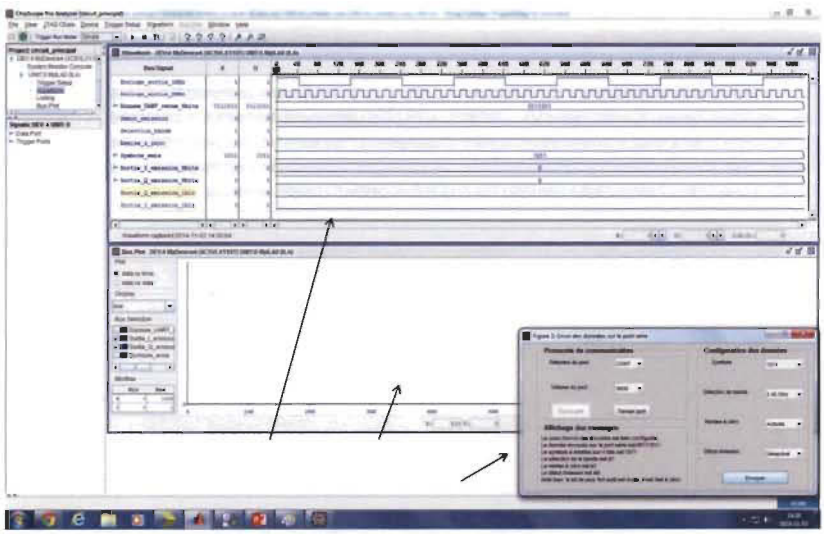


Figure D.25 : Lors d'une remise à zéro

D'après la visualisation de la forme d'onde, et le bus plot par Chipscope Pro (figure D.25), ainsi que, l'interface graphique, nous vérifions l'acheminement des données envoyées (par l'interface) sur l'analyseur logique intégré.

D'après la visualisation sous MDO4034B-3, Nous remarquons bien que les deux signaux analogiques (signal 1 est la sortie du premier filtre et signal 2 est la sortie du deuxième filtre) sont à zéro (lors d'une remise à zéro).

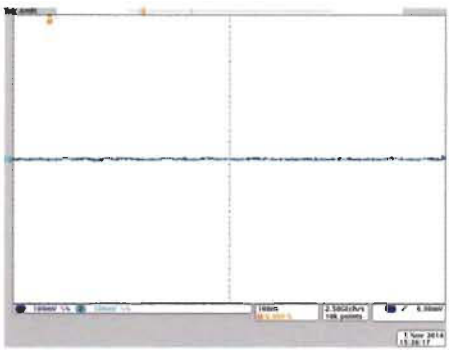


Figure D.26 : La visualisation sur l'oscilloscope

D'après la visualisation de la forme d'onde et le bus plot par Chipscope Pro ainsi que l'interface graphique, les signaux des sorties converties ressemblent à un codage qui nous donne après la décision des valeurs signées (par exemple : -32, 0, 12).

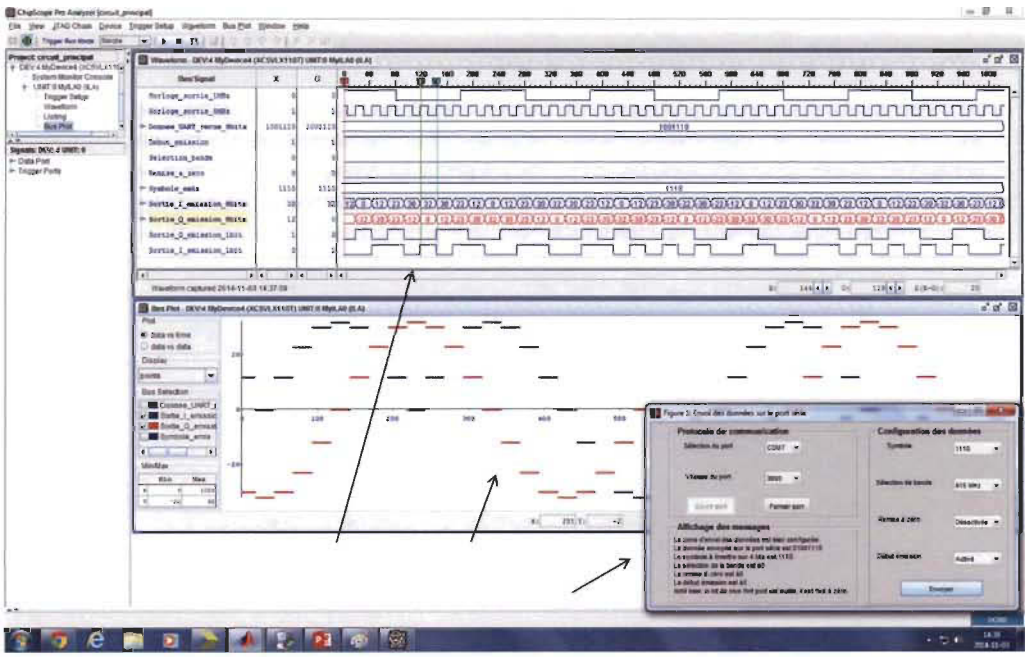


Figure D.27 : Lors de l'envoi d'un symbole

D'après la visualisation sous MDO4034B-3, Nous remarquons bien que les deux signaux analogiques (signal 1 est la sortie du premier filtre et signal 2 est la sortie du deuxième filtre) ont des formes d'onde sinusoïdales. Les résultats sont plus ou moins bons dû à la précision de la décision (au niveau du flux de bits) du modulateur Sigma-Delta.

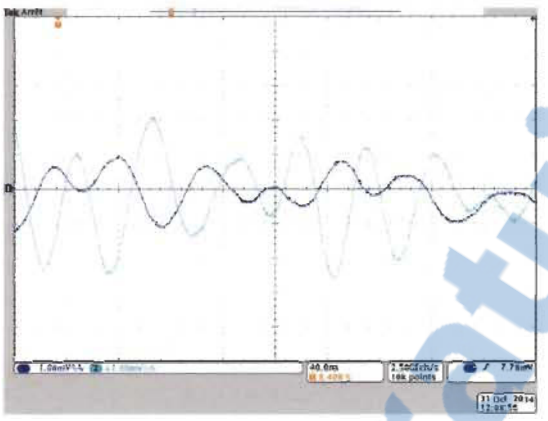


Figure D.28 : La visualisation en envoyant un symbole

La figure ci-dessous comporte la photo du banc du test de notre transmetteur avec un modulateur Sigma-Delta.

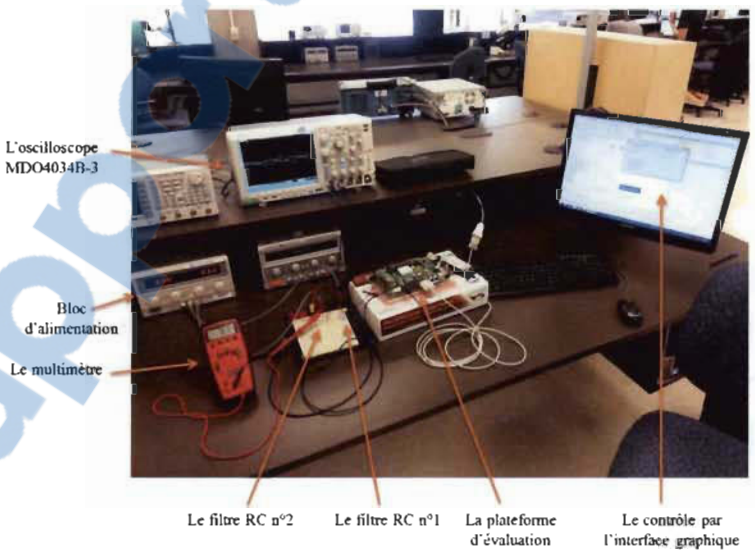


Figure D.29 : Photo d'installation

D-8. Complément des résultats de test d'envoi d'une trame PPDU

Nous continuons de présenter le reste des résultats liés au test d'envoi d'une trame PPDU que ce soit pour l'émission ou l'émission/réception.

Tout d'abord, nous commençons par la mémoire RAM. Elle est décrite en VHDL, elle estcapable de mémoriser 16 octets des données (16 x 8 bits). Dans notre cas, nous avons besoin juste de 9 octets pour enregistrer la trame PPDU et les données de configuration, à partir de l'interface graphique.

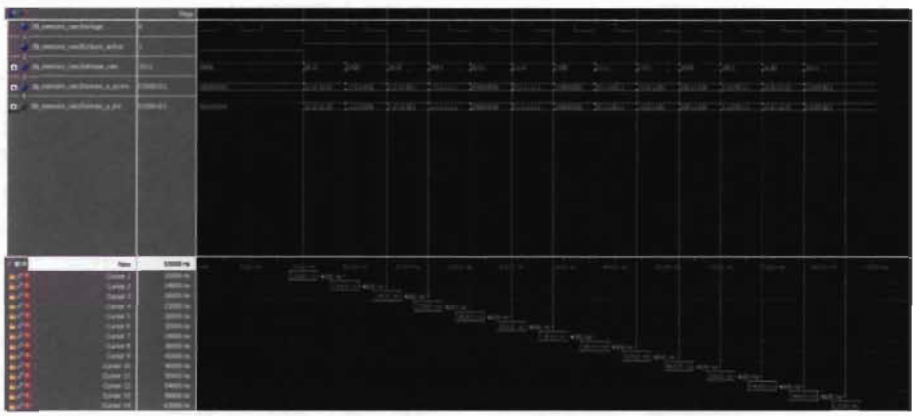


Figure D.30 : Simulation de la mémoire

La figure D.30 représente la simulation sous Modelsim de la mémoire RAM. Pour écrire une donnée dans cette mémoire, nous avons besoin d'activer l'écriture et de spécifier une adresse parmi 16 (les adresses commence de 0000 jusqu'au 1111).

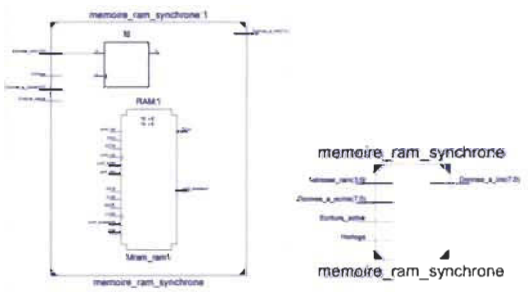


Figure D.31 : Le circuit interne de la RAM

Pour lire une donnée de la mémoire, il faut préciser une adresse, par exemple, nous avons la donnée « 11100011 » pour l'adresse « 1001 ». La figure D.31 décrit le circuit interne de la mémoire RAM.

La figure suivante illustre les résultats de test sur l'outil Chipscope Pro pour l'envoi de la trame dans l'émetteur. En (a), nous avons le symbole « 0010 » qui correspond à la donnée utile de la trame. Pareillement en (b), nous trouvons le symbole « 0101 ».

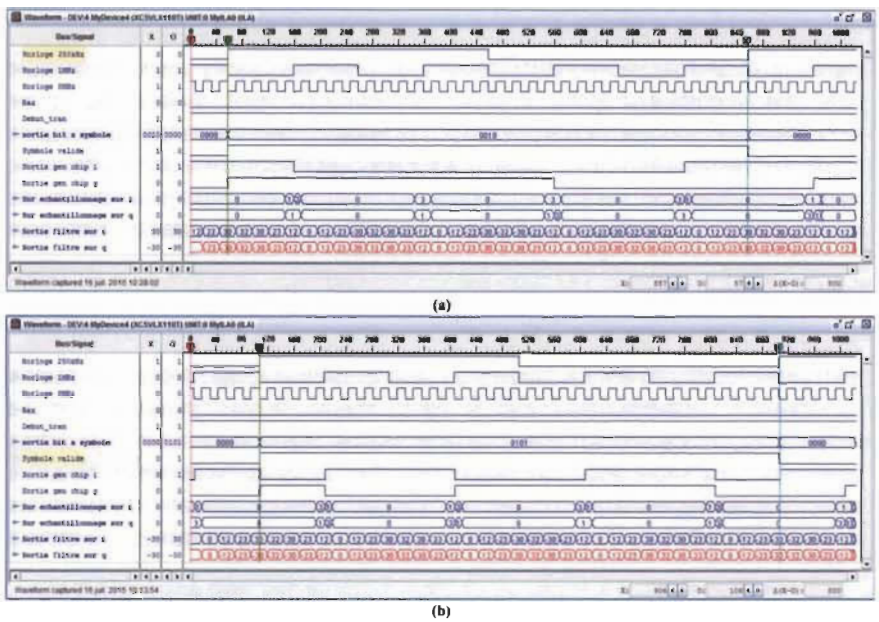
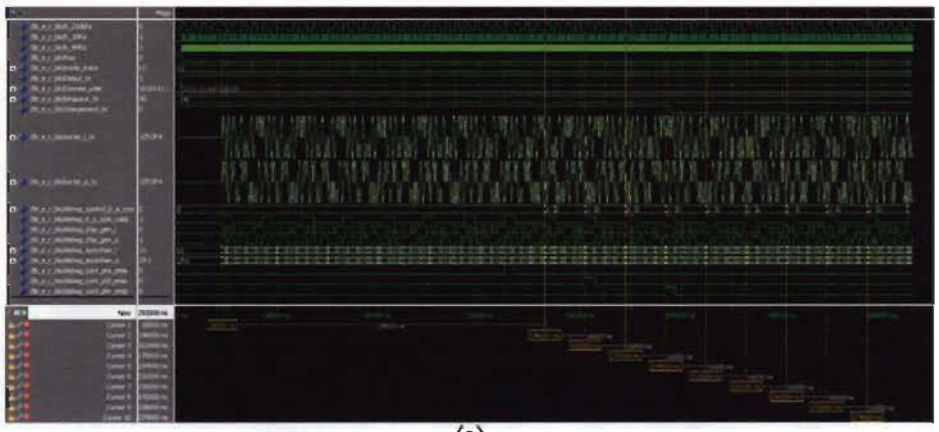
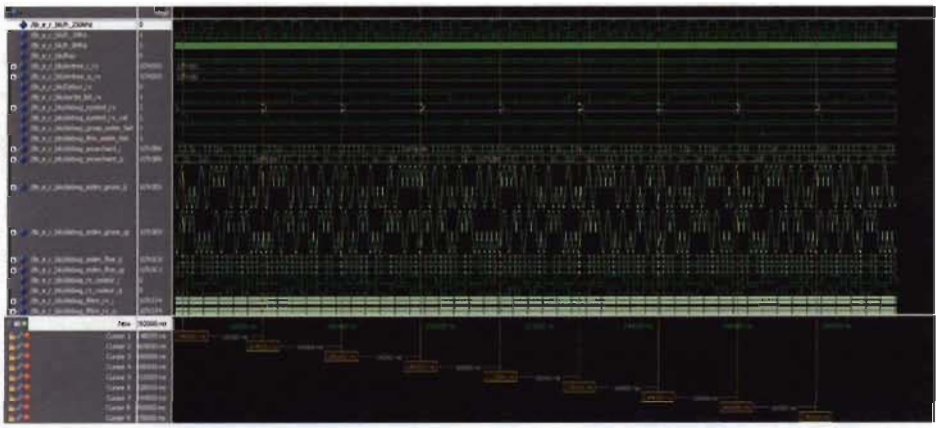


Figure D.32 : Résultat 1 de l'émetteur sur Chipscope Pro

La figure D.33 correspond à la simulation sur Modelsim de l'E/R. En (a), le résultat affiche toute la trame PPDU envoyée en commençant par le préambule et en terminant par les données utiles. Nous respectons bien la période de chaque symbole. Nous remarquons bien que les données reçues sont les mêmes par rapport aux données envoyées avec un certain décalage. En (b), nous avons les signaux liés à l'unité de réception.



(a)



(b)

Figure D.33 : Simulation de l'E/R sous Modelsim

La dernière figure représente le résultat de test sur Chipscope Pro de l'envoi de la trame dans l'E/R. Nous voyons bien que les deux données ne sont pas les mêmes vu qu'il y a un décalage de 18 us.

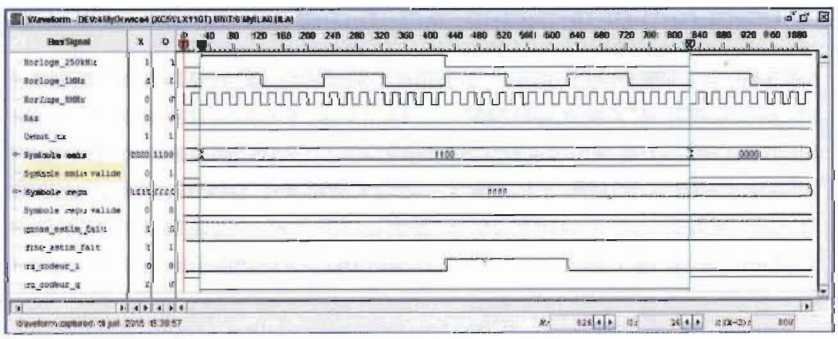


Figure D.34 : Résultat de l'E/R sur Chipscope Pro