REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr. YAHIA FARES de Médéa



Faculté des Sciences et de la Technologie

Département du Génie électrique et de l'Informatique

Mémoire de Projet de Fin d'Etude pour L'Obtention du Diplôme de Master

En Génie Electrique

Option: Systèmes de Télécommunications

Thème :

Implémentation d'un Codeur /décodeur Espace-temps d'Alamouti sur FPGA

Proposé et dirigé par :

Mr. DALI Mohamed

Réalisé par :

M^elle^ FERHAT TALEB Amina

M^elle^ LAKHDARI Nassiba

Promos: 2012 - 2013

REMERCIMENT

On remercie vivement monsieur **DALI Mohamed** d'avoir proposé et dirige ce travail de si près avec une qualité remarquable; on a grandement apprécié ses compétences et qualités scientifiques ainsi que son intelligence et son intuition sans oublier ses indéniables qualités humaines à travers un suivi régulier, une grande disponibilité et une attention minutieuse accordés à l'avancement des travaux; on lui présente toute notre gratitude.

On saisit cette occasion pour remercier **l'ensemble des enseignants** du **département d'électronique** ayant consacré leurs vies sur la voie noble de l'enseignement. On exprime aussi nos sincères remerciements aux **membres de jury.**

Que ceux qui se sentent oubliés trouvent ici notre profonde gratitude et nos chaleureux remerciements pour leur concours dans l'accomplissement de ce travail.

DÉDICACE

JE TIENS À DÉDIER CE TRAVAIL:

A ma très chère mère
Tu es l'exemple de dévouement qui n'a pas cessé de m'encourager et de prier pour moi.
Puisse Dieu, le tout puissant, te préserve et t'accorde santé, longue vie et bonheur.
A mon père.
A mes très chères sœurs : **Khaoula, Khadidja, Soumaya.**
Vous vous êtes dépensés pour moi sans conter. En reconnaissances de tous les sacrifices consentis par tous et chacun pour me permettre d'atteindre cette étape de ma vie.

A mes grands-parents, mes oncles, tantes, cousins et cousines affectueuses reconnaissances.
A mes Amies intime **Yasmine**, **Hafidha** et **Meriem.**
A mes amis; **Fatima, Hichem, Basma, Asmaa,Selma, Souhila, Amel, Fouziya, Amina, Amine,** et à leurs familles.

Je vous remercie tous, merci d'être toujours près de moi.
A toute la promotion du master 2012/2013.
A mes camarades de la faculté des sciences et de la technologie de l'université et à leurs familles.

AMINA.

A cœur vaillant rien d'impossible
A conscience tranquille tout est accessible

Quand il y a la soif d'apprendre
Tout vient à point à qui sait attendre

Quand il y a le souci de réaliser un dessein
Tout devient facile pour arriver à nos fins

Malgré les obstacles qui s'opposent
En dépit des difficultés qui s'interposent

Les études sont avant tout
Notre unique et seul atout

Ils représentent la lumière de notre existence
L'étoile brillante de notre réjouissance

Comme un vol de gerfauts hors du charnier natal
Nous partons ivres d'un rêve héroïque et brutal

Espérant des lendemains épiques
Un avenir glorieux et magique

Souhaitant que le fruit de nos efforts fournis
Jour et nuit, nous mènera vers le bonheur fleuri

Aujourd'hui, ici rassemblés auprès des jurys,
Nous prions dieu que cette soutenance
Fera signe de persévérance
Et que nous serions enchantés
Par notre travail honoré

Je dédie ce travail à

Ma très chère mère,

Aucune dédicace ne saurait être assez éloquente

Pour exprimer ce que tu mérites pour tous les sacrifices

Que tu n'as cessé de me donner depuis ma naissance,

Durant mon enfance et même à l'âge adulte.

Mon très cher père,

Aucune dédicace ne saurait exprimer l'amour,

L'estime, le dévouement et le respect que j'ai pour vous.

Rien au monde ne vaut les efforts fournis jour et

Nuit pour mon éducation et mon bien être.

A mes très chers frères

En témoignage de l'attachement, de l'amour et de

L'affection que je porte pour vous.

Je vous dédie ce travail avec tous mes vœux de

Bonheur, de santé et de réussite.

A ma très chère sœur

Je te souhaite un avenir plein de joie, de bonheur, de

Réussite et de sérénité.

Je t'exprime à travers ce travail mes sentiments de

Sororité et d'amour.

A mes chers oncles et tantes et leurs familles

Vous avez toujours été présents pour les bons conseils.

Votre affection et votre soutien m'ont été d'un grand

Secours au long de ma vie professionnelle et personnelle.

Veuillez trouver dans ce modeste travail

Ma reconnaissance pour tous vos efforts.

A ma grand-mère

Je vous dédie ce travail en témoignage de mon profond

Amour. Puisse Dieu, le tout puissant, vous préserver et

Vous accordez santé, longue vie et bonheur.

A tous les membres de ma famille, petits et grands

Veuillez trouver dans ce modeste travail l'expression de mon affection

A mes chères ami (e)s :

Hadjer, Abou beker, Touba, Essam, Fairouz, Hannane, Khadidja, Amel

Je ne peux trouver les mots justes et sincères pour vous

Exprimer mon affection et mes pensées, vous êtes pour moi des

Frères, sœurs et des amis sur qui je peux compter.

En témoignage de l'amitié qui nous uni et des souvenirs de

Tous les moments que nous avons passé ensemble, je vous dédie

Ce travail et je vous souhaite une vie pleine de santé et de

Bonheur.

Nassiba

الملخص

الهدف من هذه المذكرة هو تصميم ,محاكاة وتنفيذ تشفير/ فك التشفير المكاني و الزماني ل Alamouti علي دارة قابلة لإعادة البرمجة من نوع FPGA و ذلك ابتداءا بملخص حول الاتصالات الرقمية و نظام متعدد المداخل و المخارج MIMO و دارة قابلة لإعادة البرمجة,ثم قدمنا التصميم باستخدام البرنامج Système Générateur de XILINX. ليأتي بعدها مرحلة محاكاة للتصميم بغرض التحقق من التشغيل السليم ,وأخيرا مرحلة التنفيذ علي الدارة FPGA

الكلمات المفتاحية : نظام متعدد المداخل و المخارج, التشفير المكاني و الزماني, الدارة القابلة لإعادة البرمجة المحاكاة .

Résumé

Ce travail a pour objectif la conception la simulation et l'implémentation sur FPGA d'un système de codage/décodage espace-temps d'Alamouti. Nous avons commencé par aborder des généralités sur les communications numériques, les systèmes MIMO et, les FPGAs, après nous avons présenté la conception on utilisant le logiciel Système Générateur de XILINX. Ensuite nous avons fait une simulation sur la conception afin de vérifier son bon fonctionnement et, finalement l'étape d'implémentation sur le circuit FPGA.

Mots clés : MIMO, codage spatio-temporelle, FPGA, Simulation,

Abstract

This work presents the design, simulation and FPGA implementation of the Alamouti space-time coding / decoding. We begin with a general approach on digital communications, MIMO systems, and FPGAs, after we presented our design using the XILINX System Generator tools. Then we have made a simulation on the design in order to check its correct functioning and finally the implementation of the FPGA.

Key words: MIMO, space-time coding, FPGA, Simulation,

SOMMAIRE

INTRODUCTION GENERALE…… Page 1

CHAPITRE I : *Généralités sur la communication numérique*

I.1Introduction …… Page 3
I.2 Les éléments d'une chaine de communication numérique …… Page 3
I.3 paramètres d'évaluation d'un système de communication…… Page 5
I.4 La modulation numérique…… Page 6
I.4.1 Le principe de modulation …… Page 6
I.4.2 types de modulations …… Page 8
I.4.2.1 Modulation par déplacement d'amplitude (ASK) …… Page 8
I.4.2.2Modulation par déplacement de fréquence (FSK) …… Page 11
I.4.2.3Modulation par Déplacement de Phase (PSK)…… Page 12
I.4.2.4 Modulation d'amplitude sur deux porteuses en quadrature (QAM)… Page 15
I.5 Le canal de transmission…… Page 17
I.5.1 Canal a Bruit blanc gaussien additif (AWGN)…… Page 17
I.5.2 Canal avec évanouissement …… Page 18
I.5.2.1 Le canal de communications a trajet multiples …… Page 18
I.5.2.2 Propriétés des distributions de Rayleigh et de Rice …… Page 20
I.5.2.2.1 Distribution de Rayleigh…… Page 20
I.5.2.2.2 Distribution de Rice …… Page 21
I.6 Notion de diversité …… Page 22
1.6.1 Diversité temporelle …… Page 22
I.6.2 Diversité de fréquence…… Page 23
I.6.3 Diversité spatiale …… Page 23
I.7 Codage espace-temps pour le système MIMO …… Page 24
I.7.1 Système de transmission MIMO…… Page 24
I.7.2 Capacité des systèmes MIMO …… Page 25
I.7.3 Gain des systèmes MIMO …… Page 26
I.7.3.1Gain de diversité spatiale…… Page 26
I.7.3.2 Gain de multiplexage…… Page 26
I.7.4 Codage spatio-temporels…… Page 26
I.7.4.1 Codage spatio-temporel en treillis STTC …… Page 27

I.7.4.2 Codage spatio-temporel en bloc STBC …… Page 29
I.7.4.2.1 Le cas d'une seule antenne de réception…… Page 29
I.7.4.2.2Cas de deux antennes en réception…… Page 31
I.8 Conclusion …… Page 32

CHAPITRE II : *Généralités sur les circuits FPGA*

II.1Introduction ……... Page 33
II.2 Les circuits FPGA ……. Page 33
II.3 Description des circuits FPGA de XILINX …… Page 33
II.3.1 Les CLB (configurable logic bloc)……. Page 34
II.3.2 Les IOB (input output bloc)……... Page 36
II.3.3 Les interconnexions…….. Page 38
II.3.4 Evolution des composants logiques FPGA de XILINIX ……... Page 40
II. Les atouts des circuits FPGA ……..… Page 43
II.5 étapes d'implémentation d'une application sur un FPGA ……. Page 44
II.5.1 Spécification ……... Page 45
II.5.2 Vérification ……. Page 46
II.5.3 Implémentation …….. Page 46
II.5.4 Downloding ou Programing……... Page 46
II.6 Technique de Programmation des FPGA de Xilinx……..…. Page 47
II .7 La plateforme de développement ATLYS Spartan 6…… Page 48
II.8 Conclusion ……... Page 49

CHAPITRE III : *Simulation et Implémentation*

III.1 Introduction ……. Page 50
III.2 Schéma de l'émetteur …… Page 51
III.2.1 Codeur d'Alamouti pour la modulation BPSK…… Page 52
III.2.2 Codeur pour la modulation QPSK ……. Page 54
III.3 Canal de transmission ……. Page 58
III.4 Récepteur à base d'un décodeur d'Alamouti ……... Page 59
III.4.1 Décodeur d'Alamouti ……... Page 60
III.4.1.1 Cas d'une seule antenne de réception ……. Page 60
III.4.1.2 Cas de deux antennes de réception ……. Page 62
III.4.2 Demapping QPSK …… Page 64
III.5 Mesure du BER ……. Page 65
III.6 Réduction des ressources utilisées (Etude de l'influence de nombres de bits)…… Page 67
III.7 Interface graphique de l'environnent ISE 12. 3 …… Page 68

III.8 Synthèse et implémentation …… Page 68
III.8.1 Créer le projet …….. Page 68
III.8.2 Compilation (compile)…….. Page 68
III.8.3 Synthèse et optimisation …….. Page 69
III.8.4 Placement-routage…… Page 79
III.8.5 Programmation …… **Page 79**
III.9 Conclusion …… Page 80
CONCLUSION GENERALE …… Page 81

LISTE DES FIGURES

Figure I.1	Les éléments de base d'un système de communication numérique	Page 03
Figure I.2	Schéma d'un modulateur numérique	Page 07
Figure I.3	Représentation graphique d'un seul symbole complexe	Page 08
Figure I.4	Représentation graphique d'une constellation numérique	Page 08
Figure I.5	Modulation OOK	Page 09
Figure I.6	Constellation de modulation OOK	Page 09
Figure I.7	La constellation de la modulation ASK à M états	Page 10
Figure I.8	Modulation FSK	Page 12
Figure I.9	Modulation BPSK	Page 14
Figure I.10	Constellation de la modulation de phase BPSK	Page 14
Figure I.11	Constellation de la modulation de phase QPSK	Page 15
Figure I.12	La constellation de la QAM-16 et de la QAM-64	Page 16
Figure I.13	Représentation d'une distribution gaussienne	Page 18
Figure I.14	Effet doppler dans un canal a trajet multiples	Page 19
Figure I.15	Densité de probabilité de distribution de Rayleigh	Page 20
Figure I.16	Fonction de densité de probabilité d'une distribution de Rice pour Différentes valeurs de K et avec σ=1	Page 22
Figure I.17	Principe de diversité temporelle	Page 23
Figure I.18	Principe de la diversité fréquentielle	Page 23
Figure I.19	Système de transmission sans fil MIMO	Page 25
Figure I.20	Une chaine de transmission utilisant le codage espace-temps	Page 27
Figure I.21	Exemple de code ST en Treillis à 8 états	Page 28
Figure I.22	Schéma d'Alamouti avec une deux antennes émettrices et une antenne réceptrice	Page 29
Figure I.23	Schéma d'Alamouti avec deux antennes émettrices et deux antennes Réceptrices	Page 31
Figure II.1	Architecture interne d'un FPGA	Page 34
Figure II.2	Cellule logique (CLB)	Page 35
Figure II.3	Input Output Block (IOB)	Page 37
Figure II.4	Connexions à usage général	Page38
Figure II.5	Les interconnexions directes	Page 39
Figure II.6	Les longues lignes	Page 40
Figure II.7	Les étapes d'implémentation d'une application sur un circuit FPGA	Page 45
Figure II.8	Vue externe de plateforme Spartan 6	Page 49
Figure III.1	Flot de conception.	Page 50
Figure III.2	Schéma fonctionnel de codeur/décodeur d'Alamouti.	Page 51
Figure III.3	Schéma fonctionnel de l'émetteur avec codage d'Alamouti	Page 52
Figure III.4	Codeur Alamouti par la modulation BPSK	Page 53
Figure III.5	Les signaux transmis sur les antennes	Page 54
Figure III.6	Diagramme de constellation QPSK	Page 54
Figure III.7	Mapping de la modulation QPSK	Page 55

Figure III.8 Résultats de la simulation du mapping Page 55
Figure III.9 Codeur Alamouti pour la modulation QSK Page 56
Figure III.10 Les séquences réelles des signaux transmis sur les antennes Page 57
Figure III.11 Les séquences imaginaires des signaux transmis sur les antennes Page 57
Figure III.12 Model du canal de transmission de Rayleigh. Page 58
Figure III.13 La constellation des deux antennes Page 58
Figure III.14 Canal de Rayleigh Page 59
Figure III.15 Schéma d'un récepteur d'Alamouti Page 59
Figure III.16 Schéma global d'un système de transmission MISO Page 61
Figure III.17 Schéma global d'un système de transmission MIMO Page 63
Figure III.18 Le demapping QPSK Page 64
Figure III.19 La séquence reçue Page 64
Figure III.20 calcul de BER Page 65
Figure III.21 Le BER en fonction de du système conçu. Page 65
Figure III.22 Projet Navigator du logiciel ISE 12.3 Page 68
Figure III.23 Compilation du projet Page 69
Figure III.24 Schéma RTL équivalent d'un système MISO sous Xilinx ISE 12.3 Page 70
Figure III.25 Vue d'ensemble du fichier RTL Schématique d'un émetteur d'Alamouti Page 72
Figure III.26 Schéma RTL détaillé du mapping Page 73
Figure III.27 Schéma RTL détaillé d'un codeur Page 74
Figure III.28 Schéma RTL équivalent d'un récepteur d'Alamouti sous Xilinx ISE 12.3 Page 76
Figure III.29 Schéma RTL équivalent d'un décodeur d'Alamouti Page 77
Figure III.30 Schéma RTL équivalent du demapping Page 78
Figure III.31 Génération du bit Stream et identification de la cible FPGA. Page 79
Figure III.32 Circuit implémenté sur Spartan 6 Page 80

LISTE DES TABLES

Tableau I.1 Valeurs de l'alphabet de la modulation ASK Page 10
Tableau I.2 Les différentes valeurs en fonction du symbole à transmettre Page 14
Tableau I.3 Transmission dans le temps Page 30
Tableau I.4 Signaux reçus aux niveaux des antennes réceptrices Page 31
Tableau I.5 Canaux entre antennes émettrices et antennes réceptrices Page 31
Tableau II.1 Évolutions de la série Virtex Page 41
Tableau II.2 Évolutions de la série Spartan Page 42
Tableau III.1 Estimation des ressources Page 66
Tableau III.2 Estimation des ressources Page 67
Tableau III.3 Rapport d'utilisation des ressources Page 71
Tableau III.4 L'estimation des ressources de l'émetteur d'Alamouti Page 71
Tableau III.5 L'estimation des ressources d'un récepteur d'Alamouti Page 75

LISTE DES ACRONYMES

Abréviation	**Signification**
ACIA	Asynchrones Communication Interface Adapter
ASIC	Application Spécific Integrated Circuit
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussien Noise
ATA	Advanced Technology Attachment
BER	Bit Error Rate
BDTI	Berkeley Design Technology, Inc
BPSK	Binary Phase Shift Keying
BGA	Ball Grid Arra
CLB	Configurable Logic Bloc
CDMA-2000	Code Division Multiple Access
CPLD	Complex Programmable Logic Device
CMOS	Complementary Metal-Oxide Semiconductor
CLK	Clock
DSP	Digital Signal Processor
DPRAM	Dual Port Random Access Memory
DDR	Double Data Rate
DCM	DECOM Control Memory
DLL	Delay Lock Loop
EDK	Embedded Development Kit
ERFC	Complementary Error Function
FPGA	Field Programmable Gate Arrays
FSK	Frequency Shift Keying
JTAG	Joint Text Action Groupe
HDL	Hardware Description Language
HID	High-Intensity Discharge
HDMI	High Definition Multimedia Interface
HTML	Hypertext Markup Language
IOB	Input output Bloc
ISE	Integrated Software Environment
LCA	logic cells arrays
LCI	Large Scale Integration
LUT	Lookup Table

MIC	Modulation par Impulsion et Codage
MIMO	Multiple Input Multiple Output
MISO	Multiple Input single Output
MSI	Médium Scale Integration
MOS	Metal-Oxide Semiconductor
NRE	National Real Estate
OOK	On Off Keying
PSK	Phase Shift Keying
OEM	Onde Electro Magnétique
PLD	programmable logic device
PCI	Peripheral Component Interconnect
PLL	Phase Locked Loop
QAM	Quadrature Amplitude modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register Tansfert Level
SATA	Serial Advanced Technology Attachment
SNR	Signal Noise Ratio
SIMO	Single Input Multiple Output
SISO	Single Input Single Output
SSI	Small Scale Intégration
STBC	Space Time Bloc Coding
STC	Space Time Coding
STTC	Space Time Trellis Coding
SRAM	Static Random Access Memory
SROM	Serial Read Only Memory
TTL	Transistor Logic
TCL	Tool command language
USB	Universel Serial Bus
UART	Universal Asynchronous Receiver-Transmitter
VHDL	Very High Density Logic
WCDMA	Wideband Code Division Multiple Access

NOTATION

D	Débit binaire
T_b	Temps binaire
B	bande d'un signal
S	puissance du signal
C	taux de transmission de l'information
A	l'amplitude
F	la fréquence
φ_{ri}	La phase
Eb	L'énergie émise par bit
N_0	La densité spectrale de puissance de bruit
σ_n	La variance
Δϕ	La variation de fréquence
K	la distribution de Rice
P_e	Probabilité d'erreur par bit
T_c	Temps de cohérence
Bc	la bande de cohérence du canal
Nt	nombre d'antennes à l'émission
Nr	nombre d'antennes à la réception
S^H	transposé conjugué d'une matriceS_n
T	période de transmission
H	matrice du canal
I_n	matrice identité de taille n
t_{ij}	Un temps de délai

Introduction générale

Introduction générale

L'électronique représente un très large marché très lucratif et un domaine très dynamique riche en nouvelles découvertes et applications. Un domaine qui se caractérise par la tendance à la miniaturisation et la simplification dans le but de fabrication de systèmes compacts, légers, économiques, ergonomiques et fiables.

Cette course industrielle et scientifique a permis de pousser plus loin ce progrès et l'amélioration des systèmes en intégrant plus de transistors dans une surface de semi-conducteur la plus petite que possible. Ce qui permet l'apparition de circuits intégrés de plus en plus denses et performants tel que : microprocesseurs, ASICs, et FPGAs.

Les circuits FPGAs (Field Programmable Gate Array) sont les circuits qui ont permis de pousser encore plus loin le progrès technologique en électronique. En effet, le degré de maturité des FPGAs fait d'eux aujourd'hui une solution pour remplacer les ASICs (circuits intégrés à application spécifique) et les processeurs personnalisés dans les applications de codage et de traitement de signaux.

Réduire les couts, augmenter les performances et la fiabilité et réduire l'encombrement des composants sur circuit, sont des atouts conjugués dans un FPGA. Ces atouts justifient le succès de ces composants dans tous les domaines d'applications de l'électronique. L'aéronautique, l'aérospatial, l'industrie, les télécommunications, les appareils médicaux, l'électroménager…etc. Tous les domaines trouvent le bon compromis en intégrant les FPGAs.

L'application des FPGAs pour les systèmes de télécommunication est très bénéfique. Les caractéristiques principales d'un système de transmission peuvent être optimisées sans se soucier d'influencer un autre paramètre. En ce qui concerne la partie matérielle, la modification du circuit n'implique pas la dépense dans un matériel supplémentaire, car il y aura seulement une exploitation des ressources internes à l'FPGA.

L'une des caractéristiques majeures des communications sans fil demeure les environnements dans lesquels se propagent les ondes électromagnétiques. Ces environnements sont plus souvent de types multi trajets à cause des phénomènes liés à la propagation de ces ondes. Ce qui entraine le phénomène d'évanouissement des signaux reçue dégradant ainsi la qualité des communications. D'autre part, la précision, la rapidité, la fréquence et le bruit sont les principales caractéristiques du canal de transmission dans une chaine de transmission.

Les chercheurs ont trouvé que les systèmes MIMO (Multiple Input Multiple Output) ont une meilleure performance sous les canaux à évanouissement que les systèmes SISO (Single Input Single Output). L'utilisation d'antennes multiples est une technique très efficace pour lutter contre les évanouissements du canal. Cette technique appelée codage espace-temps STC permet d'utiliser la diversité spatiale et temporelle pour réduire l'effet du canal et rendre la communication plus fiable.

Introduction générale

Notre travail s'inscrit dans le cadre de l'implémentation d'une technique de codage espace-temps simple sur un circuit FPGA. Le mémoire se divise en trois chapitres organisés de la manière qui suit :

Le premier chapitre présente des généralités sur la communication numérique, on décrivant le principe de fonctionnement d'une chaine de transmission numérique, ce chapitre traite aussi les modulations numériques ASK, PSK, FSK, QAM et les différents modèles de canaux utilisés. Nous donnons ainsi un aperçu sur les systèmes MIMO et les techniques de codage espace-temps.

Le deuxième chapitre a pour objectif de se familiariser avec les circuits FPGA commençant par une présentation de l'architecture interne des FPGA de chez XILINX ou nous expliquons les différents blocs constituant le circuit. Ensuite nous décrivons les étapes et phases essentielles pour le développement d'une application à base de ce circuit finalisant par les avantages des circuits FPGAs.

Le troisième chapitre se divise en deux parties : partie simulation qui est consacré à la présentation de la conception d'un système de transmission à base d'un codeur/décodeur espace-temps d'Alamouti en utilisant le logiciel System Generator de Xilinx et le Simulink de Matlab ainsi les résultats de simulation de notre conception. Partie implémentation de notre conception sur un FPGA de type SPARTAN 6 de la compagnie XILINX avec une description pas à pas du processus d'implémentation effectué en utilisant le logiciel ISE fournie à cet effet par le fabricant avec des aperçues graphique du circuit implémenté.

En conclusion, nous donnerons un récapitulatif des différentes parties de notre présent travail et nous terminerons sur des perspectives de recherche et développement du système de transmission ayant pour but l'amélioration des performances du système.

Chapitre I

I.1 Introduction :

Dans ce chapitre ; Nous présentons les principes de base qui sont à la base de l'analyse et de la conception des systèmes de communication numérique, le sujet des communications numériques implique la transmission d'information à partir d'une source qui produit l'information à une ou plusieurs destinations. Une importance particulière dans l'analyse et la conception des systèmes de communication doit être donnée aux caractéristiques des canaux physiques par lesquels l'information est transmise, car c'est les caractéristiques du canal qui affectent généralement la conception des blocs fonctionnels de base.

I.2 Les éléments d'une chaine de communication numérique :

La figure I.1 montre un schéma synoptique d'un exemple d'un système de communication numérique.

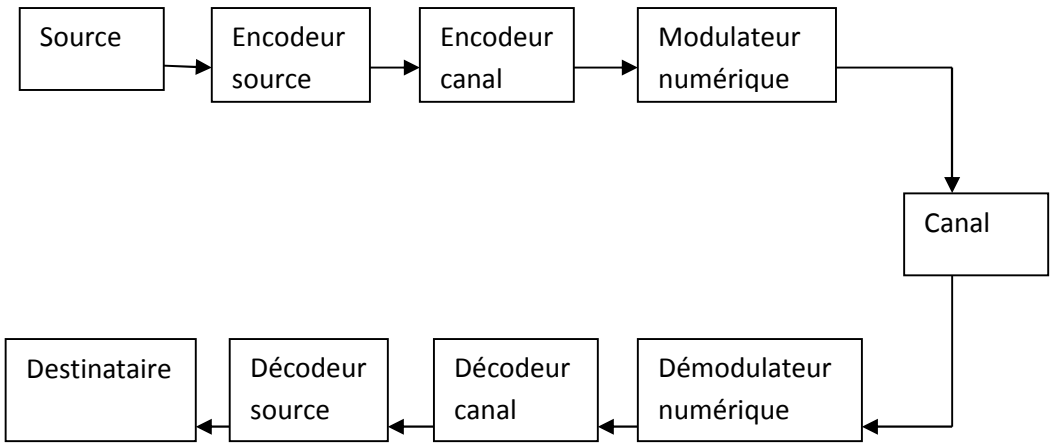


Figure I.1 : Les éléments de base d'un système de communication numérique.

Source : La source transmis un message numérique constitué d'une suite d'éléments binaires. En groupant sous forme de blocs de n éléments indépendants. On obtient un alphabet de M=2^n symboles. Dans les cas fréquent ou cet alphabet est constitué uniquement de deux valeurs le symbole est dit binaire.

La source est caractériser par son débit définit comme étant le nombre d'éléments binaires émis par unité de temps $D = \frac{1}{T_b}$ ou T_b est le temps séparant l'émission de deux éléments consécutifs.

Encodeur de source : c'est une technique qui permet de compresser l'information à transmettre en réduisant le plus que possible toute forme de redondance.

Exemple : la modulation par impulsion et codage (***MIC***) permet de coder la parole sur huit bit.

Codeur/Décodeur de canal : les erreurs qui se produisent durant la communication sont dues aux bruits et aux interférences produites par le canal de transmission. Pour y remédier, un codage correcteur d'erreur est utilisé, des bits de redondances sont ajoutés d'une manière contrôlée : ces bits sont exploités par le récepteur pour détecter les erreurs et ainsi décoder l'information utile qui est associé au démodulateur, connaissant le code utilisé par le codeur canal et la redondance contenue dans les données reçues il minimise le taux d'erreurs entre le message émis et le message décodé pour reconstituer la séquence transmise.

Modulateur numérique: le modulateur serf d'interface au canal de communication il a pour rôle d'adapter le spectre au canal sur lequel il sera transmis. Etant donné qu'en pratique tous les canaux sont capables de transmettre des signaux électriques il est donc nécessaire d'associer au message numérique une représentation physique sous forme d'un signal électrique. Le modulateur associe à chaque mot de n élément binaires un signal $\boldsymbol{S_i(t)}$ $avec$ i=1,2….M de durée $T = nT_b$ choisis parmi M=2^n signaux.

Le canal : le canal de transmission est un support physique de l'information, il peut être filaire (câble coaxial ou fibre optique….) ou l'espace libre qui utilise la propagation d'une onde électromagnétique dans l'atmosphère. Ce milieu est généralement réservé aux transmissions par satellite ou par faisceaux hertziens.

Les caractéristiques physiques des voies de transmission peuvent varier considérablement, et les bons modèles de canal sont critiques à la conception des systèmes de communication efficaces.

Démodulateur numérique: le démodulateur ramène le signal en bande de base. Il traite la forme d'onde transmise et la réduit en une séquence de nombres représentant les estimés des symboles transmis. Il réalise également l'opération de détection.

Décodeur de source : le décodeur connaissant la méthode de codage source recrée un signal numérique qui puisse être compris par le destinataire. Les rôles du codeur et du décodeur de source sont de diminuer le débit de transmission pour augmenter la capacité effective du système de transmission.

La différence entre le signal original et celui reconstruit est une mesure de la qualité du

système de communication. **[1, 2, 3]**

I.3 Paramètres d'évaluation d'un système de communication:

Les paramètres fondamentaux qui commandent le taux et la qualité de la transmission de l'information sont la largeur de bande B de canal et la puissance S du signal.

La largeur de bande d'un canal est la gamme des fréquences par lesquelles elle peut transmettre un signal sans le modifier. Par exemple, si un canal peut transmettre un signal dont les composants de fréquence occupent une gamme de 0 jusqu'à un maximum de 5000 Hz la largeur de bande B de canal est de 5 kHz.

Pour comprendre le rôle de B, considérer la possibilité d'augmenter la vitesse d'information transmise par la compression de temps du signal. Si un signal est comprimé à temps par un facteur de 2, il peut être transmis dans la moitié du temps, et la vitesse de la transmission est doublée. Compression par un facteur de 2, cependant, causes le signal a « agiter » deux fois aussi rapidement, impliquant que les fréquences de ses composants sont doublées. Pour transmettre ce signal comprimé sans déformation, la largeur de bande de canal doit également être doublée. Ainsi, le taux de transmission de l'information est directement proportionnel à B. Plus généralement, si un canal de la largeur de bande B peut transmettre N impulsions par seconde, puis pour transmettre KN impulsions par seconde nous avons besoin d'un canal de largeur de bande KB. Donc, le nombre d'impulsions par seconde qui peut être transmis à travers un canal est directement proportionnel à sa largeur de bande B.

La puissance S du signal joue un double rôle dans la transmission de l'information. Premièrement, S est lié à la qualité de la transmission, **si** la puissance du signal est grande, la puissance de bruit de canal est petite, et l'information est reçue avec moins d'incertitude. Un grand rapport signal/bruit (SNR) permet également la transmission d'information à travers une longue distance ; quoiqu'un certain minimum SNR soit nécessaire pour la communication.

Le deuxième rôle de la puissance de signal est très important Pour maintenir les données de transmission, on peut réduire B si on augmente **S**, ou on peut réduire **S** si on augmente B.

Dans le cas binaire, la puissance transmise de signal est réduite au coût de la largeur de bande accrue. L'augmentation de B permet une réduction significative de S. La limitation imposée à la communication par la largeur de bande de canal et le SNR est nettement accentué par l'équation de Shannon.

$$C = B \log_2 (1 + SNR) \text{ bit/s} \qquad (I.1)$$

Ici C est le taux de transmission de l'information par seconde. Ce taux **C** (connu sous le nom de la capacité de canal) est le nombre maximum des symboles binaires qui peuvent être transmis par second avec une probabilité d'erreur très faible. En d'autres termes, un canal peut transmettre $B \log_2 (1 + SNR)$ bit/s

Il est impossible de transmettre un taux plus haut que ceci sans encourir des erreurs. L'équation de Shannon met en évidence clairement la limitation sur le taux de communication imposé par B et SNR. **[4]**

I.4 La modulation numérique :

I.4.1 Le principe de modulation :

Les signaux de bande de base produits par de divers émetteurs d'informations ne sont pas toujours appropriés à la transmission directe à travers un canal donné. Ces signaux sont habituellement modifiés pour faciliter leur transmission. Ce processus de conversion est connu comme modulation.

Dans ce processus, le signal de bande de base est utilisé comme moyen pour modifier un ou plusieurs paramètres d'une onde porteuse.

L'onde porteuse est un signal sinusoïdal m(t) ayant la forme :

$$m(t) = A\cos(w_0 t + \varphi_0) \qquad (I.2)$$

Où :

A : est l'amplitude

f : est la fréquence ou f=$w_0/2\pi$

φ_0 : est la phase initiale.

Le signal modulant, obtenu après codage, est un signal en bande de base, éventuellement complexe, qui s'écrit sous la forme :

$$c(t) = \sum c_k \, g(t - kT) \qquad (I.3)$$

Avec : $c_k = a_k + jb_k$

La fonction g(t) est une forme d'onde qui est prise en considération dans l'intervalle [0, T] puisque t doit vérifier la relation : kT<= t < (k+1) T.

Dans les modulations ASK, FSK et, PSK, la modulation transforme ce signal en un signal modulé m(t) tel que :

$$m(t) = R\left[\sum c_k(t) e^{j(w_0 t + \varphi_0)}\right] \qquad (I.4)$$

Si les $c_k(t)$ sont réels ($b_k(t) = 0$), la modulation est dite unidimensionnelle, et s'ils sont

complexes la modulation est dite bidimensionnelle.

Le signal modulé s'écrit aussi plus simplement :

$$m(t) = \sum a_k(t)\cos(w_0 t + \varphi_0) - \sum b_k(t)\sin(w_0 t + \varphi_0) \quad \text{(I.5)}$$

Ou encore : $m(t) = a(t)\cos(w_0 t + \varphi_0) - b(t)\sin(w_0 t + \varphi_0)$ (I.6)

Le signal $a(t) = \sum a_k(t)$ module en amplitude la porteuse en phase $\cos(w_0 t + \varphi_0)$

Le signal $b(t) = \sum b_k(t)$ module en amplitude la porteuse en quadrature $\sin(w_0 t + \varphi_0)$

Dans la plupart des cas les signaux élémentaires $a_k(t)$ et $b_k(t)$ sont identiques à un coefficient près et ils utilisent la même forme d'impulsion g(t) appelée aussi "formant".

$$a_k(t) = a_k g(t - kT) \quad et \quad b_k(t) = b_k g(t - kT) \quad \text{(I.7)}$$

Les deux signaux a(t) et b(t) sont aussi appelés "trains modulants" et s'écrivent :

$$a(t) = \sum a_k g(t - kT) \quad et \quad b(t) = \sum b_k g(t - kT) \quad \text{(I.8)}$$

Les symboles a_k et b_k prennent respectivement leurs valeurs dans l'alphabet $(a_1, a_2, \ldots a_m)$ et dans l'alphabet $(b_1, b_2, \ldots b_m)$.

Le schéma d'un modulateur numérique est représenté dans la figure I.2

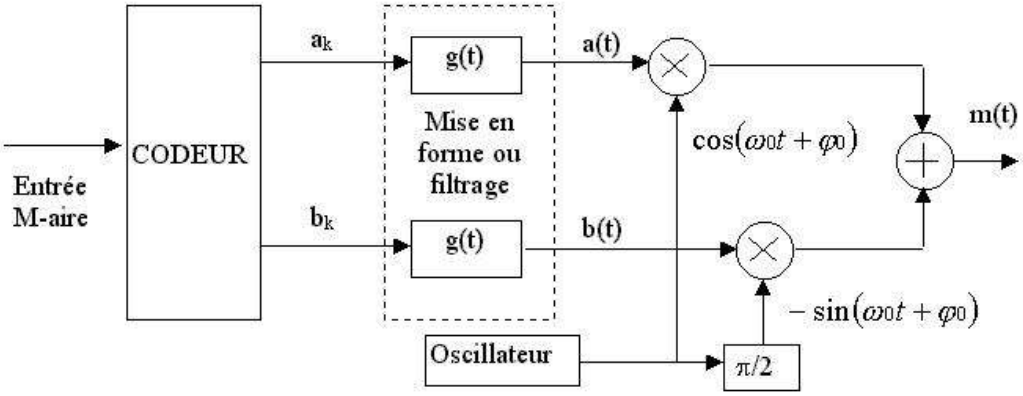


Figure I.2 : schéma d'un modulateur numérique.

Une représentation dans le plan complexe qui fait correspondre à chaque signal élémentaire un point $c_k = A_k + jB_k$ permet de différencier chaque type de modulation. La figure suivante représente graphiquement un symbole C_k :

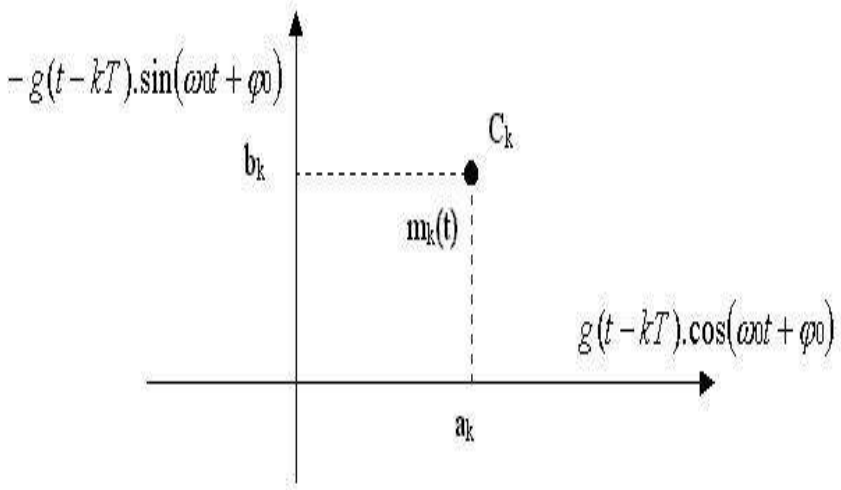


Figure I.3 : représentation graphique d'un seul symbole complexe C_k .

L'ensemble de ces points associés aux symboles porte le nom de constellation. [5]

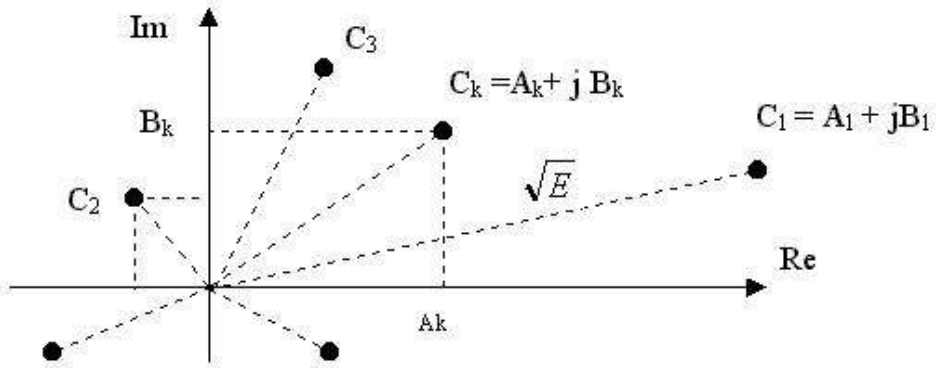


Figure I.4 : Représentation graphique d'une constellation numérique.

I.4.2 types de modulations :

I.4.2.1 Modulation par déplacement d'amplitude (ASK):

Les Modulations par Déplacement d'Amplitude sont aussi souvent appelées par leur abréviation anglaise : ASK pour "Amplitude Shift Keying".

Dans ce cas, la modulation ne s'effectue que sur la porteuse en phase $\cos(w_0 t + \varphi_0)$.

Il n'y a pas de porteuse en quadrature. Cette modulation est parfois dite mono dimensionnelle. Le signal modulé s'écrit alors:

$$m(t) = \sum a_k g(t - kT)\cos(w_0 t + \varphi_0)$$

La forme de l'onde g(t) est rectangulaire, de durée T et d'amplitude égale à 1 si t

appartient à l'intervalle [0, T] et égale à 0 ailleurs.

Rappelons que le symbole a_k prend sa valeur dans l'alphabet $(a_1, a_2, \ldots a_m)$ .Autrement dit, cet alphabet met en évidence les M=2^n amplitudes possibles du signal, la valeur n désignant les groupements de n bits ou symboles à émettre. Les changements d'amplitude de la porteuse se produiront au rythme R de la transmission des symboles.

Un exemple de modulation d'amplitude est la modulation (binaire) par tout ou rien encore appelée par son abréviation anglaise : OOK pour "On Off Keying".

Dans ce cas, un seul bit est transmis par période T, et par conséquent n=1 et M=2. Le symbole a_k prend sa valeur dans l'alphabet (0, 1). On observe donc sur un chronogramme des extinctions de porteuse quand $a_k = 0$ ce qui est présenté par la figure suivante.

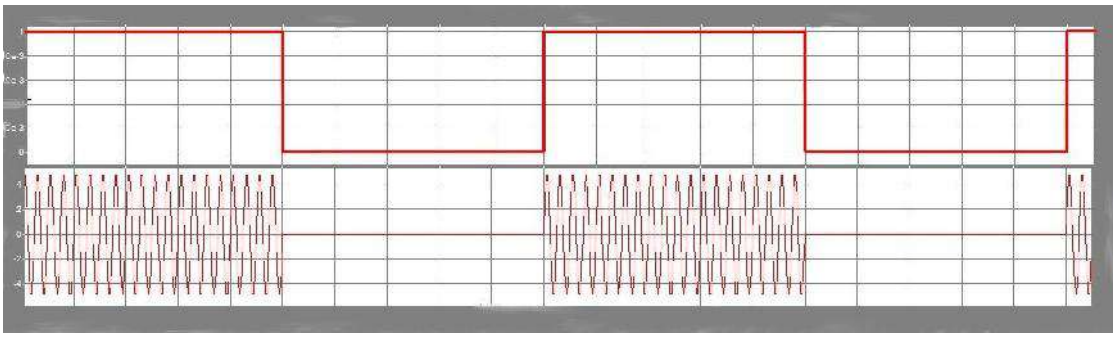


Figure I.5: modulation OOK.



Figure I.6 : constellation de modulation OOK.

Un autre exemple de la modulation ASK est ASK symétrique

On a toujours M=2^namplitudes possibles du signal, mais ici les valeurs de l'alphabet sont représentées telles que :$A_i = (2i - M - 1)a_0$ avec i=0, 1, 2,3….. (M-1)

Suivant les valeurs de n on obtient le tableau suivant :

n	M	Valeurs de l'alphabet
1	2	-a0, a0
2	4	-3a0, -a0, a0, 3a0
3	8	-7a0, -5a0, -3a0, -a0, a0, 3a0, 5a0, 7a0

Tableau I.1 : valeurs de l'alphabet de la modulation ASK.

La constellation de la modulation à M états symétriques est donnée dans la figure1.7 en prenant les valeurs 2, 4 et 8.

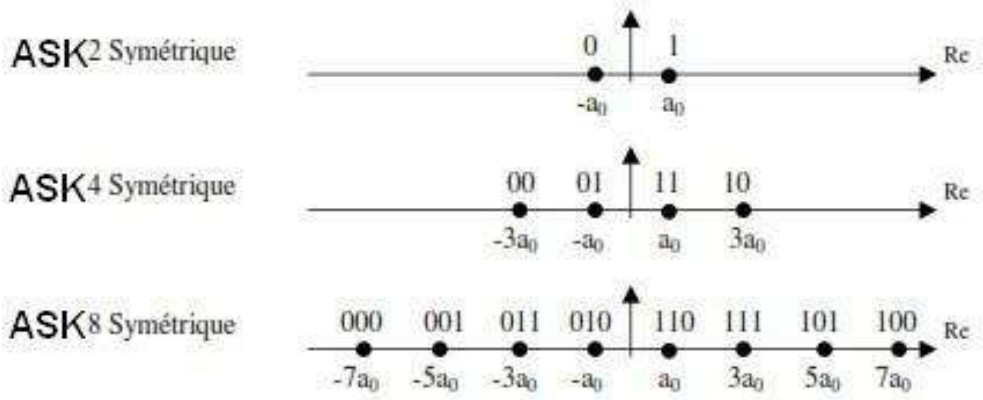


Figure I.7 : La constellation de la modulation ASK à M états.

Pour pouvoir comparer les différentes modulations entre elles, il est d'usage d'exprimer la probabilité d'erreur on fonction du rapport $\frac{Eb}{N_0}$ dans lequel :

Eb: Représente l'énergie émise par bit.

N_0 : représente la densité spectrale de puissance de bruit.

Ou la probabilité d'erreur par symbole est donnée par l'équation I.9:

$$P_s(e) = \frac{M-1}{M} erfc\left(\sqrt{\frac{3log_2 M}{M^2-1}\frac{Eb}{N_0}}\right) \qquad (I.9)$$

On peut alors constater que pour conserver une probabilité d'erreur par symbole constante lorsque M augmente, il faut aussi augmenter le rapport $\frac{Eb}{N_0}$ Autrement dit, il faut augmenter l'énergie émise par bit.

La modulation ASK est simple à réaliser mais est assez peu employé pour M>2 car ses performances sont moins bonnes que celles d'autres modulations, notamment pour sa résistance au bruit**. [5]**

1.4.2.2Modulation par déplacement de fréquence (FSK):

Les Modulations par Déplacement de Fréquence sont aussi souvent appelées par leur abréviation anglaise : FSK pour "Frequency Shift Keying".

Le signal modulé m(t) s'écrire :

$$m(t) = Re[e^{j\emptyset t} e^{j(w_0 t + \varphi_0)}] \quad \text{(I.10)}$$

Une propriété de la modulation par déplacement de fréquence est d'avoir une enveloppe constante : $e^{j\emptyset t} = cst$

L'expression du signal modulé par déplacement de fréquence s'écrit aussi plus simplement en prenant $\varphi_0 = 0$ comme :

$$m(t) = \cos(w_0 t + \emptyset(t)) = \cos(2\pi f_0 t + \emptyset(t)) \quad \text{(I.11)}$$

C'est la dérivée de la phase $\emptyset(t)$ qui est reliée de façon simple (linéaire) à la valeur des symboles, le tout constituant une relation non linéaire.

La fréquence instantanée f(t) du signal m(t) est obtenue par dérivation de la phase $2\pi f_0 t + \emptyset(t)$ par rapport au temps :

$$f(t) = f_0 + \frac{1}{2\pi}\frac{d\emptyset}{dt} \quad \text{(I.12)}$$

Où

f_0 : représente la fréquence centrale.

$\frac{1}{2\pi}\frac{d\emptyset}{dt}$: Représente la déviation de fréquence par rapport à la fréquence f_0 .

Appelons Δf la différence de la fréquence instantanée correspondant à l'émission de deux symboles adjacents ; Et soit a_k un symbole appartenant à l'ensemble {± 1, ± 3,… ± (M-1)}.

La déviation de fréquence s'écrit alors, suivant la valeur à transmettre :

$$\frac{1}{2\pi}\frac{d\emptyset}{dt} = \frac{\Delta f}{2}\sum a_k g(t - kT)$$

$$\frac{d\emptyset}{dt} = \pi\Delta f \sum a_k g(t - kT) \quad \text{(I.13)}$$

Où g(t) est l'impulsion rectangulaire de durée T.

La phase étant l'intégrale de la fréquence, on obtient après intégration de l'expression précédente et pour t appartenant à l'intervalle [kT, (k+1) T] :

$$\emptyset(t) = \pi\Delta f a_k g(t - kT) + \vartheta \quad \text{(I.14)}$$

Où $\vartheta = \emptyset(kt)$ est une constante.

Cette expression montre que la phase varie linéairement sur l'intervalle [KT, (k+1) T] et que cette variation est de: $\pi\Delta f a_k$

En reportant l'expression $\emptyset(t)$ dans la relation (1.12) on obtient :

La fréquence instantanée : $f(t) = f_0 + \frac{\Delta f}{2} a_k$ (I.15)

L'expression du signal modulé : $m(t) = \cos(2\pi \left(f_0 + \frac{\Delta f}{2} a_k\right) t)$ (I.16)

On peut aussi définir l'indice de modulation $\mu = \Delta f T$ qui conditionne la forme de la densité spectrale du signal modulé. **[5]**

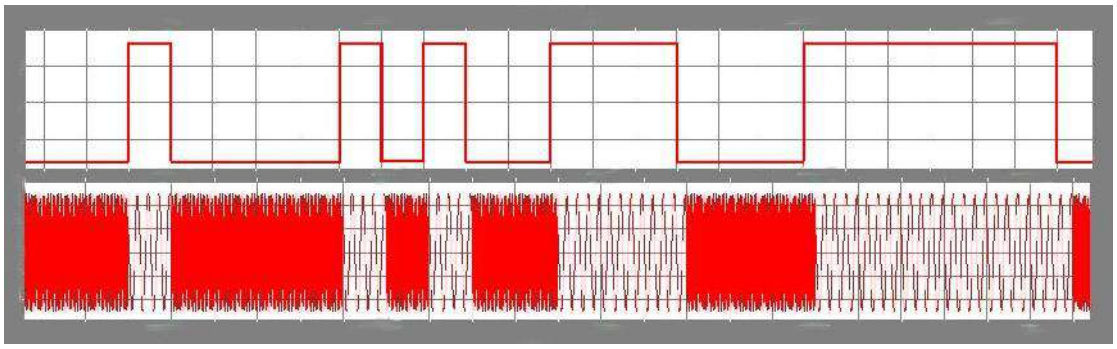


Figure I.8 : modulation FSK.

1.4.2.3Modulation par Déplacement de Phase (PSK):

Les Modulations par Déplacement de Phase sont aussi souvent appelées par leur abréviation anglaise : PSK pour "Phase Shift Keying".

Reprenons l'expression générale d'une modulation numérique présenter dans l'équation (1.4)

Les signaux élémentaires $a_k(t)$et $b_k(t)$ utilisent la même forme d'onde g(t)qui est ici une impulsion rectangulaire de durée T et d'amplitude égale à A si t appartient à l'intervalle [0, T [et égale à 0 ailleurs.

On a toujours : $a_k(t) = a_k g(t - kT)$ et $b_k(t) = b_k g(t - kT)$

Soit : $c_k(t) = (a_k + jb_k)g(t - kT) = c_k g(t - kT)$ (I.17)

Dans le cas présent, les symboles C_k sont répartis sur un cercle, et par conséquent

$c_k = a_k + jb_k = e^{j\varphi k}$ D'où : $a_k = \cos(\varphi_k)$ et $b_k = \sin(\varphi_k)$

Et : $a_k(t) = \cos(\varphi_k)g(t - kT)$ et $b_k(t) = \sin(\varphi_k)g(t - kT)$ (I.18)

On pourrait imaginer plusieurs PSK-M pour la même valeur de M où les symboles seraient disposés de façon quelconque sur le cercle. Pour améliorer les performances par rapport au bruit, on impose aux symboles d'être répartis régulièrement sur le cercle (il sera ainsi plus facile de les discerner en moyenne). L'ensemble des phases possibles se traduit alors par les expressions suivantes :

Lorsque M > 2 $\varphi_k = \frac{\pi}{M} + \frac{k2\pi}{M}$

Lorsque M = 2 $\varphi = 0$ou π

Les symboles C_k prennent leurs valeurs dans un alphabet de M >2 éléments $e^{j\varphi k}$où φ_k est défini avec k = 0,1,…M-1.

On peut aussi considérer que $a_k$et b_k prennent simultanément leurs valeurs dans l'alphabet $\cos(\varphi_k)$ et sin (φ_k)

Le signal modulé devient :

$$m(t) = Re[\sum e^{j\varphi k}\, g(t-kT)e^{j(w_0 t+\varphi_0)}]$$

$$m(t) = Re[\sum g\,(t-kT)e^{j(w_0 t+\varphi_0+\varphi k)}] \qquad \text{(I.19)}$$

Soit, plus simplement, en ne considérant que l'intervalle de temps [kT, (k+1) T] :

$$m(t) = Re\left[Ae^{j(w_0 t+\varphi_0+\varphi k)}\right] = Acos(w_0 t+\varphi_0+\varphi_k)$$

$$m(t) = Acos(w_0 t+\varphi_0)\cos(\varphi_k) - Asin(w_0 t+\varphi_0)\sin(\varphi_k) \qquad \text{(I.20)}$$

Cette dernière expression montre que la phase de la porteuse est modulée par l'argument $\varphi_k$de chaque symbole ce qui explique le nom donné à la PSK. Remarquons aussi que la porteuse en phase $cos(w_0 t+\varphi_0)$est modulée en amplitude par le signal $Acos(\varphi_k)$ et que la porteuse en quadrature $\sin(w_0 t+\varphi_0)$ est modulée en amplitude par le signal $A(\sin(\varphi_k))$.

L'expression de la PSK montre qu'il s'agit d'une modulation à enveloppe constante ; l'enveloppe étant le module de l'enveloppe complexe. Cette propriété est intéressante pour des transmissions sur des canaux non linéaires, ce qui fait de la modulation PSK un outil de choix par exemple pour les transmissions par satellites. L'intérêt d'avoir un signal modulé à enveloppe constante est que cela permet d'employer les amplificateurs dans leur zone de meilleur rendement qui correspond souvent à un mode de fonctionnement non linéaire.

On prend l'exemple de la modulation PSK-2 encore appelée par son abréviation anglaise : BPSK pour "Binary Phase shift Keying".C'est une modulation binaire (un seul bit est transmis par période T) : n=1, M=2 et $\varphi_k = 0\ ou\ \pi$

Le symbole $c_k = e^{j\varphi_k}$ prend donc sa valeur dans l'alphabet {-1, 1}.

Ici, la modulation ne s'effectue que sur la porteuse en phase $cos(w_0 t+\varphi_0)$. C'est une modulation mono dimensionnelle. Le signal modulé s'écrit alors pour t appartenant à l'intervalle [0, T] :

$$m(t) = \pm Acos(w_0 t+\varphi_0) \qquad \text{(I.21)}$$

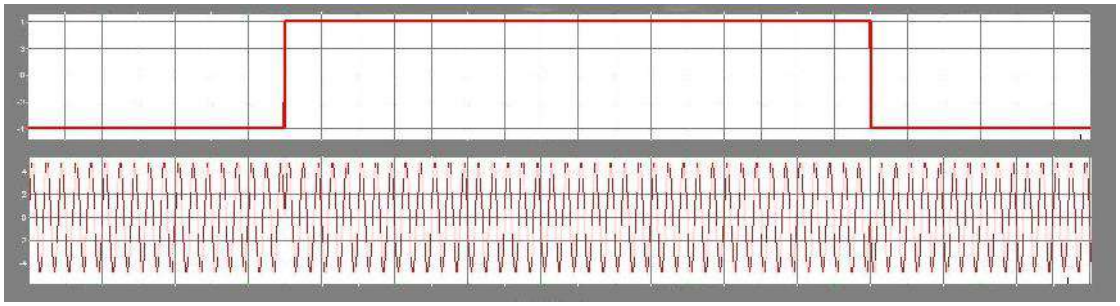


Figure I.9 : modulation BPSK.

La constellation BPSK est représentée dans la figure I.10. On remarque que cette modulation est strictement identique à la modulation ASK-2 symétrique.

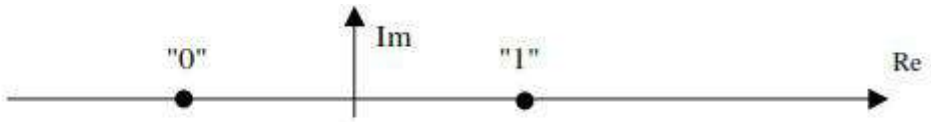


Figure I.10 : constellation de la modulation de phase BPSK.

Un autre exemple de modulation PSK-M est la modulation PSK-4 encore appelée par son abréviation anglaise : QPSK pour "Quadrature Phase shift Keying".

C'est une modulation d'amplitude à deux niveaux sur chacune des porteuses en quadrature.

Dans ce cas : n=2, M=4 et $\varphi_k = \frac{\pi}{4} + \frac{K\pi}{2}$

Bit pair	Bit impair	Symbole	φ^k	a^k	b^k	$a^k\sqrt{2}$	$b^k\sqrt{2}$
0	0	00	$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1	1
1	0	01	$3\frac{\pi}{4}$	$-\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	-1	1
1	1	11	$5\frac{\pi}{4}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	-1	-1
0	1	10	$7\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	1	-1

Tableau I.2 : Les différentes valeurs en fonction du symbole à transmettre.

La constellation QPSK est représentée dans la figure I.11. Elle montre que l'affectation des bits aux points de la constellation se fait en général selon un codage de Gray.

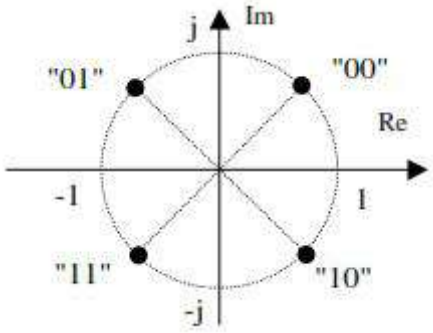


Figure I.11: Constellation de la modulation de phase QPSK

L'augmentation de M réduit la distance entre symboles adjacents sur la constellation et cela dégrade naturellement les performances. **[5]**

Il est possible de comparer les PSK entre elles, en utilisant la probabilité d'erreur par symbole Pe en fonction du rapport $\frac{Eb}{N_0}$ la probabilité d'erreur par symbole est donnée par la relation

$$P_S(e) = erfc(\sqrt{\log M \frac{E_b}{N0}} sin\frac{\pi}{M}) \qquad (I.22)$$

I.4.2.4 Modulation d'amplitude sur deux porteuses en quadrature (QAM):

Les modulations d'amplitude sur deux porteuses en quadrature (QAM) sont aussi appelées par leur abréviation anglaise : QAM pour "Quadrature Amplitude Modulation". C'est une modulation dite bidimensionnelle.

La modulation ASK et PSK ne constituent pas une solution satisfaisante pour utiliser efficacement l'énergie émise lorsque le nombre de points M est grand. En effet, dans la modulation ASK les points de la constellation sont sur une droite, et dans la PSK les points sont sur un cercle. Or, la probabilité d'erreur est fonction de la distance minimale entre les points de la constellation, et la meilleure modulation est celle qui maximise cette distance pour une puissance moyenne donnée. Un choix plus rationnel est alors une modulation qui répartit les points uniformément dans le plan. Pour faire cela, nous avons vu que le signal modulé m(t) s'écrire comme montrer dans l'équation (I.5) et (I.7)

Le signal modulé m(t) est donc la somme de deux porteuses en quadrature, modulées en amplitude par les deux signaux a(t) et b(t).

Les symboles $a_k$et $b_k$prennent respectivement leurs valeurs dans deux alphabets à M éléments $(a_1,a_2 ,\ldots a_m)$ et $(b_1,b_2,\ldots b_m)$.donnant ainsi naissance à une modulation possédant un nombre E = M^2états. Chaque état est donc représenté par un couple (a_k,b_k) ou ce qui revient au même par un symbole complexe $c_k = a_k + jb_k$.

Dans le cas particulier mais très fréquent où M peut s'écrire M = 2^n , alors les a_k représentent un mot de n bits et les b_k représentent aussi un mot de n bits. Le symbole complexe $c_k = a_k + jb_k$ peut par conséquent représenter un mot de $2.n$ bits. L'intérêt de cette configuration est que le signal m(t)est alors obtenu par une combinaison de deux porteuses en quadrature modulées en amplitude par des symboles $a_k$et b_k indépendants.

De plus, les symboles $a_k$et b_k prennent très souvent leurs valeurs dans un même alphabet à M éléments.

Par exemple, la QAM-16 est construite à partir de symboles $a_k$et b_k qui prennent leurs valeurs dans l'alphabet {±d, ±3d} où d est une constante donnée. Une représentation de la constellation de cette modulation est donnée dans la figure I.12. La QAM-16 a été souvent utilisée, notamment pour la transmission sur ligne téléphonique du RTC (à 9600 bit/s) et pour les faisceaux hertziens à grande capacité (140 Mbits/s).

Plus généralement lorsque les symboles $a_k$et b_k prennent leurs valeurs dans l'alphabet {±d, ±3d, ±5d,…, ± (M-1) d} avec M = 2^n on obtient une modulation à 2^{2n} états et une constellation avec un contour carré dont font partie la QAM-4, La QAM-16, la QAM-64 et la QAM-256. **[5]**

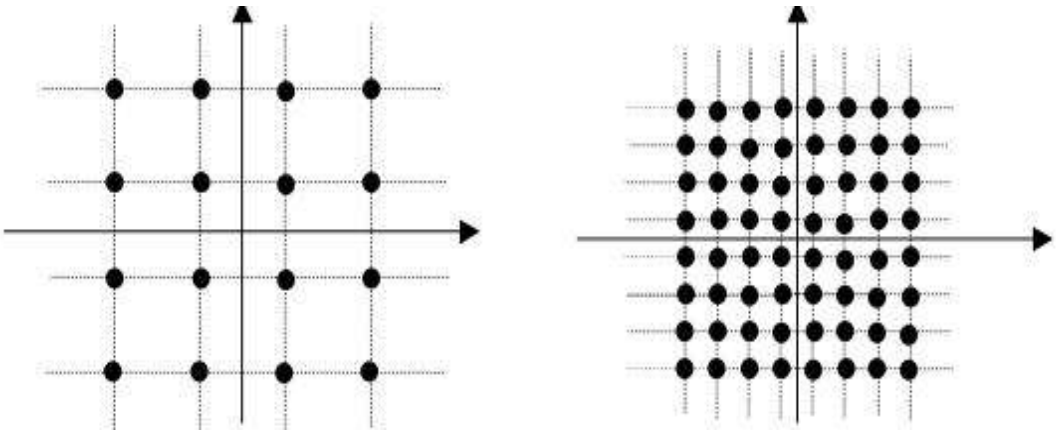


Figure I.12: La constellation de la QAM-16 et de la QAM-64.

I.5 Le canal de transmission :

Un canal de communication est un médium physique qui est utilisé pour la transmission d'un signal à partir d'un transmetteur jusqu'au récepteur. Il est également possible de procéder via des ondes électromagnétiques dans les communications sans-fil.

Quel que soit le mode de transmission, une constante demeure, c'est que des perturbations aléatoires non prévisibles affectent le signal transmis avant sa réception.

On utilise les processus stochastiques pour faire de modélisation d'un canal de communications dû aux incertitudes inhérentes au processus. Un des problèmes communs est la présence de bruits additifs ; ils ont diverses causes dont le bruit thermique des composantes constituant l'appareil récepteur par exemple.

L'effet du bruit peut être atténué par l'augmentation de puissance du signal transmis. Toutefois, certaines limitations peuvent l'empêcher. La bande passante disponible est également une limitation majeure, non seulement due aux composantes physiques, mais par les restrictions législatives et la demande très élevée pour cette ressource limitée. **[6]**

I.5.1 Canal a Bruit blanc gaussien additif (AWGN) :

Sans doute le plus simple des modèles, nous avons le modèle de canal avec bruit blanc gaussien additif. Dans cette situation, le signal reçu r(t) est la résultante du signal original s(t) avec l'ajout de bruits modélisés par une fonction de densité de probabilité gaussienne :

$$r(t) = s(t) + n(t) \qquad \text{(I.23)}$$

Un bruit blanc Gaussien est un processus aléatoire stationnaire puisqu'il est indépendant du signal transmis. La spécificité d'un bruit blanc réside dans l'uniformité de sa densité spectrale de puissance qui vaut N_0 sur toute la bande de fréquences. Du fait de sa largeur de bande théoriquement infinie, il est difficile d'exprimer le bruit blanc, c'est pourquoi l'on suppose que le bruit sommé au signal d'entrée du récepteur a été filtré par un filtre idéal, de largeur de bande B très grande devant la bande utile. Le bruit blanc peut être rendu Gaussien par une distribution gaussienne. Ainsi, le bruit doit suivre la loi normale, ou loi de Gauss, régie par l'équation suivante :

$$p(r) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r-s)^2}{2\sigma^2}} \qquad \text{(I.24)}$$

La figure suivante représente la probabilité d'un bruit blanc gaussien. **[7][8]**

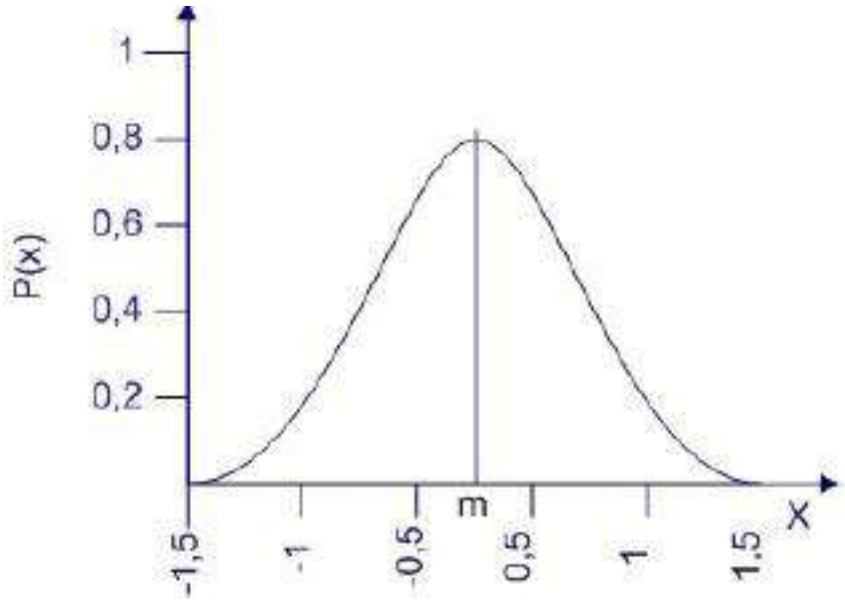


Figure I.13 : Représentation d'une distribution gaussienne.

I.5.2 Canal avec évanouissement :

Une version d'un modèle de canal ayant seulement l'évanouissement qui affecte le signal est aussi possible. Ce modèle est décrit par l'équation r (t) = h (t; td) ∗ s (t)

I.5.2.1 Le canal de communications a trajet multiples :

Lorsque le récepteur est immobile et lorsque rien dans l'environnement ne bouge, le récepteur reçoit plusieurs versions décalées dans le temps du même signal. Suivant la longueur du trajet emprunte par chacune de ces versions, les signaux arrivent avec un délai ou un déphasage qui leur est propre. C'est la superposition de ces signaux déphasés entre eux qui engendre des additions constructives ou destructives suivant la position du récepteur mobile. Ce canal statique spatialement devient dynamique pour le récepteur si ce dernier se déplace, En se déplaçant, les longueurs des trajets changent de manière continue et le déphasage de chaque trajet également. L'usager en mouvement subit une succession d'affaiblissements rapides suivant sa position dans l'espace.

Nous pouvons donc relier les variations de la longueur des trajets au déplacement du récepteur. Le taux de variation de phase est relie a l'effet Doppler associe au mouvement du récepteur mobile. Intuitivement, si le récepteur est immobile, il ne va jamais rencontrer d'affaiblissement et plus le récepteur se déplace vite, plus les phases changent rapidement et avec les chances de rencontrer des affaiblissements rapides durant un laps de temps fixe. La Figure I.14 permet de mieux comprendre le phénomène.

Prenons un signal non modulé, émis par une source lointaine. Un obstacle diffuse ce signal émis dans plusieurs directions. Le récepteur mobile, d'abord à la position 1, reçoit ce signal réfléchi par l'obstacle. En se déplaçant à la vitesse v durant la dure Δt, le récepteur mobile parcourt la distance d et se rend en position 2. La longueur de trajet supplémentaire empruntée par le signal quand le récepteur mobile est à la position 2 est Δl=d.cos(α)

Cette variation de phase est une fonction croissante de la vitesse de déplacement comme nous pouvions le penser de manière intuitive. La variation de fréquence apparente est donnée par : $\Delta\emptyset = -\frac{2\pi\Delta l}{\lambda} = -\frac{2\pi v\Delta t}{\lambda}cos\alpha$ (I.25)

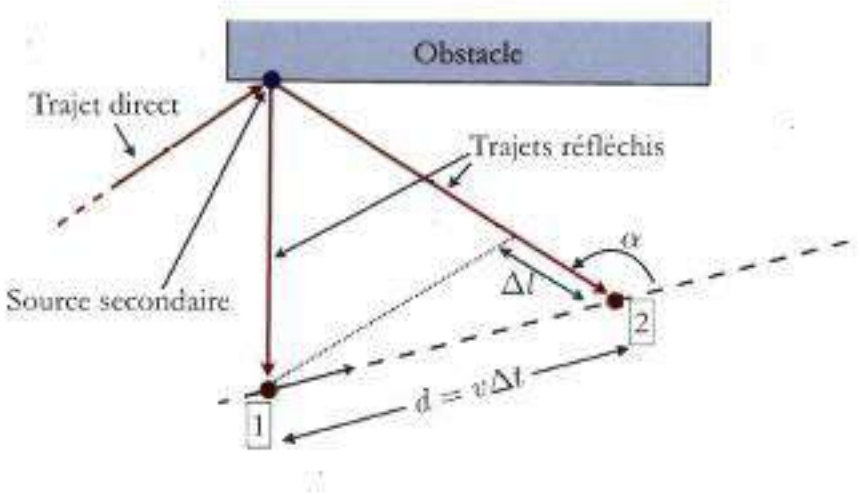


Figure I.14 : Effet doppler dans un canal a trajet multiples.

En pratique, il est très difficile de prévoir la direction d'arrivée (l'angle α) de chaque onde. C'est ici que la méthode analytique touche à ses limites. Devant la complexité du problème, les amplitudes ainsi que les phases seront considérées comme des variables aléatoires et les méthodes statistiques permettront de modéliser le phénomène de manière statistique**. [9]**

I.5.2.2 Propriétés des distributions de Rayleigh et de Rice :

I.5.2.2.1 Distribution de Rayleigh :

La loi de Rayleigh est une loi de probabilité à densité de la norme d'un vecteur gaussien bidimensionnel dont les coordonnées sont indépendantes, centrées et de même variance.

La fonction de densité d'une distribution de Rayleigh est donnée à l'équation I.26 Sa représentation graphique permet de remarquer qu'elle ne possède pas de symétrie comme la loi normale par rapport à sa moyenne.

$$p(r) = \frac{r}{\sigma^2}\exp(-\frac{r^2}{2\sigma^2}) \qquad \text{(I.26)}$$

Ou :

σ^2:Represente la variance

La moyenne de la distribution est donnée par :

$$E\{r\} = \sigma\sqrt{\frac{\pi}{2}} = 1.25\sigma \qquad \text{(I.27)}$$

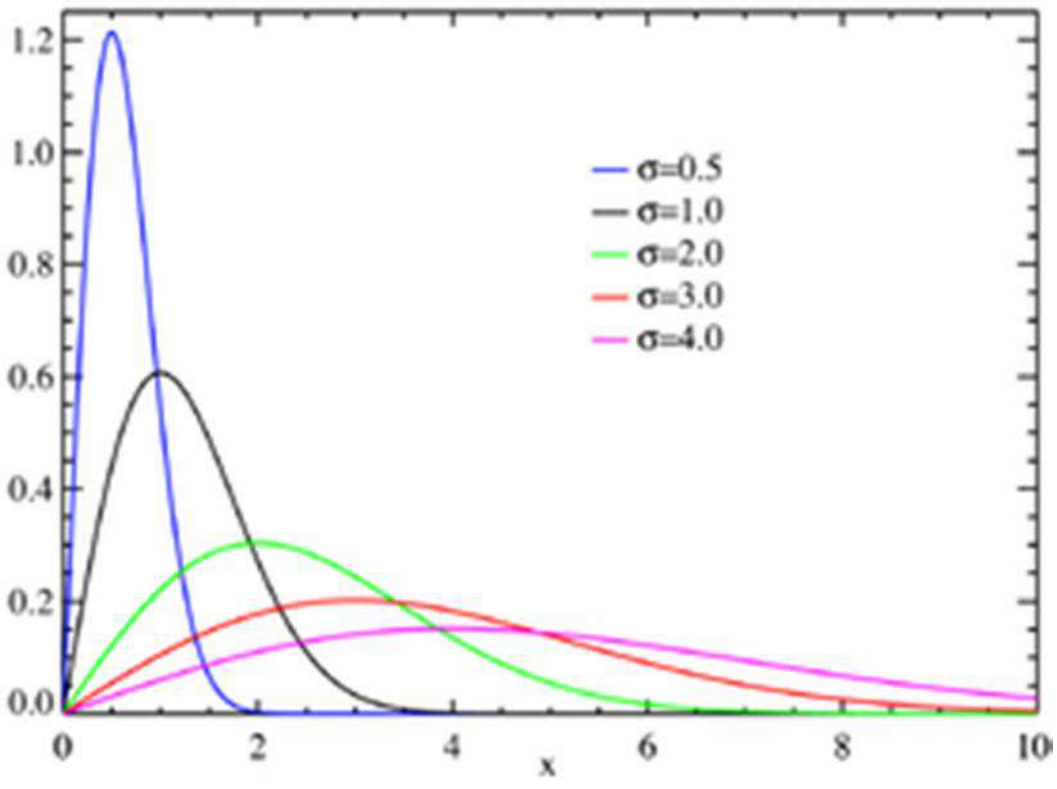


Figure I.15 : Densité de probabilité de distribution de Rayleigh.

La moyenne quadratique s'exprime comme :

$$E\{r^2\} = 2\sigma^2 \qquad \text{(I.28)}$$

La connaissance de la moyenne de la distribution de Rayleigh suffit pour caractériser la distribution de Rayleigh. Donc, sachant que le canal suit une distribution de Rayleigh, il suffit de quelques mesures pour estimer la moyenne et caractériser complètement le canal. **[9]**

I.5.2.2.2 Distribution de Rice :

Nous nous sommes concentres jusqu'à présent sur de distributions de Rayleigh parce qu'elles décrivent la situation la plus courante dans un canal de communications mobiles. La plupart du temps, il n'existe pas de ligne de vue directe entre l'émetteur et le récepteur et les trajets multiples ont des puissances voisines (aucun trajet dominant).

Nous pouvons tout de même considérée le cas où il y aurait une ligne de vue de l'émetteur ou plus généralement le cas où une des ondes incidentes serait sensiblement plus puissante que les autres. Ce type de canal a été étudié par Rice.

Intuitivement, nous pouvons nous attendre à ce que la moyenne de l'amplitude soit plus élevée et que les affaiblissements soient plus rares qu'avec un canal de Rayleigh. La densité de probabilité d'une distribution de Rice est donnée par :

$$p(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2+s_d^2}{2\sigma^2}\right) I_0\left(\frac{rs_d}{\sigma^2}\right) \qquad pour\ s \geq 0 \qquad \text{(I.29)}$$

Où s_d désigne l'amplitude du signal dominant, et I_0 la est la fonction de Bessel modifiée du premier type à l'ordre zéro. Nous remarquons que pour s_d nul, la distribution de Rice équivaut à une distribution de Rayleigh. Le paramètre K est souvent utilisé pour décrire la distribution de Rice, ce paramètre peut s'interpréter comme le rapport entre la puissance du signal dominant et la puissance des multi trajets.

$$K(db) = 10\log\left(\frac{s_d^2}{E\{s^2\}}\right) = 10\log\left(\frac{s_d^2}{2\sigma^2}\right) \qquad \text{(I.30)}$$

La Figure 1.16 montre l'allure de la distribution de Rice pour différentes valeurs de K. Lorsque K devient grand, la moyenne de la distribution est voisine de s_d. **[9]**

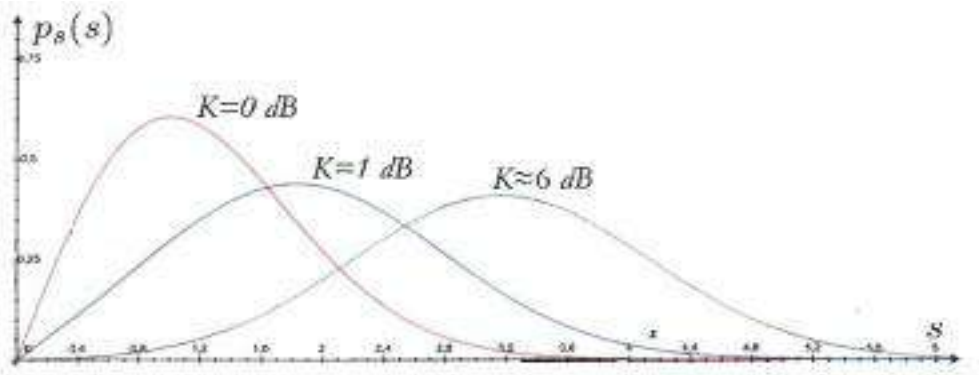


Figure I.16 : Fonction de densité de probabilité d'une distribution de Rice pour différentes valeurs de K et avec σ=1

I.6 Notion de diversité :

La diversité est une technique utilisée pour combattre l'évanouissement. Le principe sous-jacent est de transmettre plusieurs répliques de la même information sur plusieurs canaux ayant des puissances comparables et des évanouissements indépendants, et donc, il est fort probable qu'au moins un, ou plus, des signaux reçus ne soit pas atténué à un moment donné, rendant possible une transmission de bonne qualité. Deux principales catégories de diversité existent, la catégorie explicite, et la catégorie implicite. La première catégorie consiste à transmettre d'une manière ou une autre, plusieurs répliques du même message. La seconde catégorie, quant à elle, consiste à n'envoyer qu'une copie, mais à compter sur des concepts tels que le multi parcours pour obtenir plusieurs versions du signal envoyé. En pratique, les déférents "canaux" ne sont pas totalement dé corrélés, et donc, une diversité parfaite n'est pas toujours accessible. Pour la diversité spatiale, par exemple, il faut une distance, entre les antennes, qui soit plus grande que la distance cohérente. De même, pour la diversité fréquentielle, une séparation fréquentielle supérieure à la cohérence de la bande passante est nécessaire. Lors de présence de plusieurs antennes à la réception, une combinaison des signaux reçus est nécessaire, pour cela, plusieurs méthodes existent qui sont : la diversité temporelle, la diversité fréquentielle et la diversité spatial**. [6]**

I.6.1 Diversité temporelle :

L'information est répétée dans le temps à intervalle plus grand que le temps de cohérence du canal Tc (le temps où le canal peut être considéré comme constant c'est la durée

minimale séparant deux évanouissements selon un certain degré de dé corrélation).

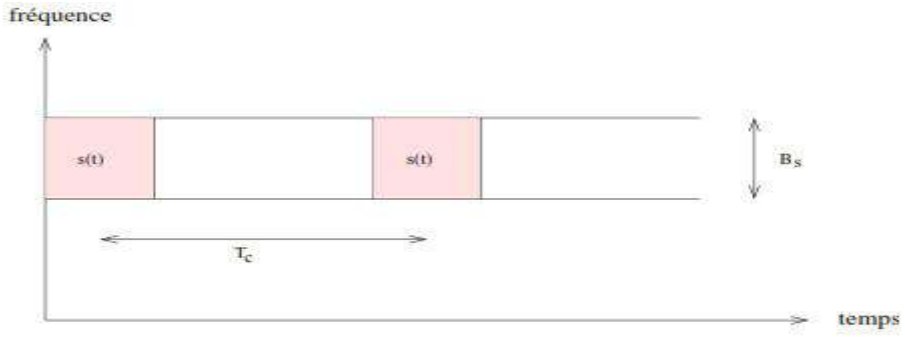


Figure I.17 : Principe de diversité temporelle.

Le désavantage de la diversité temporelle est évidemment le retard induit par la diversité, ce qui réduit le débit de transmission.

I.6.2 Diversité de fréquence :

Le même signal est retransmis sur plusieurs fréquences porteuses. L'écart fréquentiel est supérieur à la bande de cohérence du canal Bc (ou l'espace fréquentiel minimal entre deux fréquences affectée d'évanouissements indépendants).

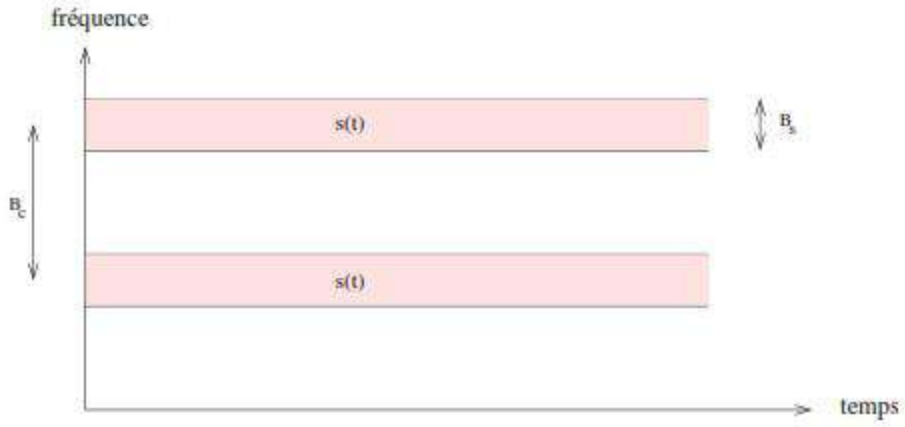


Figure I.18 : principe de la diversité fréquentielle.

On a le même désavantage de la diversité de fréquence que celui de la diversité temporelle. On perd l'efficacité spectrale qui est liée au débit de transmission.

I.6.3 Diversité spatiale :

Dans ce cas, le récepteur traite des versions indépendantes du signal, obtenues à partir de capteurs judicieusement disposés. La distance minimale à respecter entre capteurs de

l'ordre d'une fois la longueur d'onde du signal pour le mobile. Elle peut atteindre plus de dix fois la longueur d'onde pour une station de base de façon à garantir un évanouissement indépendant. Contrairement à la diversité temporelle et de fréquence, la diversité spatiale ne réduit pas le débit de transmission mais elle requiert plus de complexité et d'espace. Il est possible d'utiliser des antennes à l'émission, à la réception ou à la fois à l'émetteur et au récepteur.

Les systèmes exploitant la diversité spatiale en réception sont appelés système SIMO ("Single-Input Multiple-Output"). Les systèmes exploitant la diversité Spatiale en émission sont appelés système MISO ("Multiple-Input Single-Output").

La troisième structure est l'utilisation de la diversité spatiale à la fois à l'émission et à la réception nommé système MIMO ("Multiple-Input Multiple-Output"). Les systèmes MIMO nous permettent non seulement de réduire l'effet de fading mais atteindre aussi les bornes de capacité largement supérieures à celles données par d'autres structures à condition que l'émetteur et le récepteur soient dans un milieu riche en diffuseurs. **[10] [11]**

I.7 Codage espace-temps pour le système MIMO :

I.7.1 Système de transmission MIMO :

Les systèmes multi-antennes à l'émission et à la réception (Multi input multi output MIMO en anglais) permettent théoriquement d'accroître la capacité des liens de communications sans fil par rapport aux systèmes composés d'une seule antenne à l'émission et à la réception. En faisant l'hypothèse que les trajets entre chaque antenne d'émission et de réception sont indépendants, Foshini et Telatar [11] ont démontré que la capacité théorique du canal MIMO avec Nt antennes à l'émission et Nr antennes à la réception croit linéairement avec min(Nt, Nr). Les systèmes MIMO sont l'un des principaux axes de développement pour augmenter les débits des communications sans fil, nous assistons à un très rapide développement de cette technologie avec des applications dans les réseaux locaux sans fil et les réseaux de communication de 3^{eme} ou 4^{eme} génération.

Les systèmes MIMO présentent des avantages majeurs par rapport aux systèmes SISO :

1) grâce à l'apport de la diversité spatiale ils permettent d'améliorer la qualité du lien en s'affranchissant des évanouissements des canaux.

2) par multiplexage spatial, ils permettent d'augmenter le débit d'information sans augmenter la bande passante ou la puissance transmise.

Le principe de base des systèmes MIMO consiste donc à combiner les signaux judicieusement tant à l'émission qu'à la réception pour exploiter la diversité spatiale et donc réduire les effets des évanouissements ou pour augmenter le débit de transmission**. [12]**

I.7.2 Capacité des systèmes MIMO :

Soit un système MIMO composé de Nt antennes à l'émission et de Nr antennes à la réception comme illustré sur la figure 1.19.

Le signal reçu yj à chaque instant sur la j-ième antenne de réception est la somme des symboles bruités issus des Nt signaux transmis :

$$y_j = \sum_{i=1}^{N_t} h_{ji} c_i + n_j \quad \text{I.31}$$

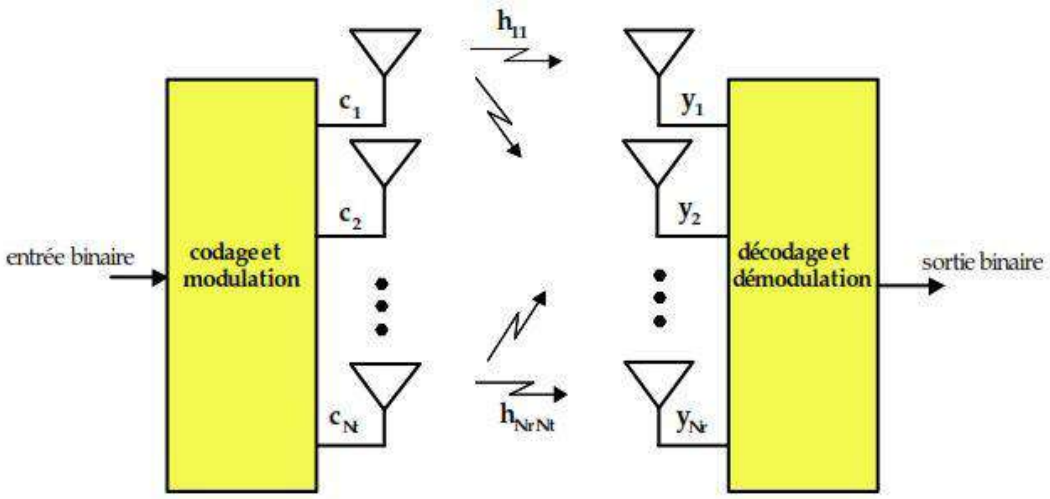


Figure I.19 : Système de transmission sans fil MIMO.

Où h_{ji} est le gain du canal entre l'antenne d'émission et l'antenne de réception, n_j est le bruit additif qui est modélisé par des échantillons indépendants et suivant une loi gaussienne centrée de variance $\frac{N_0}{2}$ par dimension réelle. Le rapport signal à bruit (SNR) par antenne de réception est défini par :

$$SNR = \frac{E_s}{N_0} \quad \text{(I.32)}$$

L'équation I.31 peut s'écrire sous une forme matricielle comme suit:

$$\mathbf{y} = \boldsymbol{H}_c + \mathbf{n} \quad \text{(I.33)}$$

Où **y** et **n** sont respectivement le vecteur de réception et le vecteur de bruit de dimension$(N_r . 1)$.

Soit la matrice du canal MIMO **H** de dimension $N_r . N_t$ suivante :

$$\boldsymbol{H} = \begin{Bmatrix} h_{11} & \cdots & h_{1\,N_t} \\ \vdots & \ddots & \vdots \\ h_{N_r 1} & \cdots & h_{N_r N_t} \end{Bmatrix} (I.34)$$

La capacité des systèmes MIMO est la suivante :

$$C(SNR, N_t, N_r) = \log_2(\det\left(\mathbf{I}_{nr} + \frac{SNR}{N_t}\mathbf{H}\boldsymbol{H}^*\right))\text{bps/Hz} \quad (I.35)$$

Où $\mathbf{I}_{nr}$ est la matrice identité de dimension nr.

I.7.3 Gain des systèmes MIMO :

I.7.3.1 Gain de diversité spatiale:

Dans un canal de transmission sans fil, la puissance reçue varie dans le temps, en fréquence et dans l'espace. La diversité est alors utilisée pour combattre ces évanouissements. L'idée principale consiste à utiliser à la réception plusieurs répliques du signal. Plus le nombre de répliques augmente, plus la probabilité que toutes les répliques subissent simultanément un évanouissement diminue. Classiquement, dans les systèmes SISO les diversités temporelle et fréquentielle sont exploitées par le codage de canal associé à une fonction d'entrelacement. Avec les systèmes MIMO, on dispose d'une nouvelle forme de diversité : la diversité spatiale. L'ordre de diversité est en fonction du nombre d'antennes à l'émission et à la réception.

En pratique, pour mettre à profit cette diversité, il est nécessaire d'utiliser un code spatio-temporel**. [12]**

I.7.3.2 Gain de multiplexage:

Lorsque les évanouissements des différents canaux sont indépendants, on peut montrer que la matrice de canal est de rang plein avec une grande probabilité. Il est alors possible de voir le canal MIMO comme un ensemble de canaux SISO en parallèle. En transmettant des flux d'information dans chacun de ces canaux, il est possible d'augmenter le débit d'information. Il en résulte un gain dit de multiplexage. **[12]**

I.7.4 Codage spatio-temporels :

Les codes spatio-temporel offrent à la fois de la diversité et du gain de codage tout en améliorant l'efficacité spectrale.

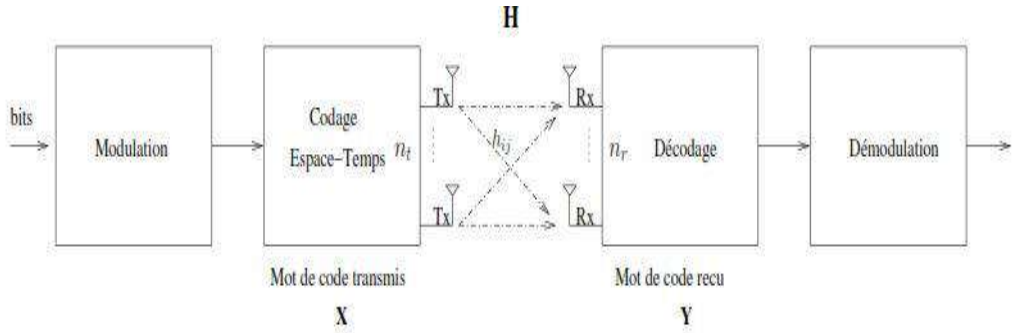


Figure I.20 : Une chaine de transmission utilisant le codage espace-temps.

Les codes spatio-temporels se divisent en deux familles: les codes spatio-temporels en treillis (STTC) et les codes spatio-temporel en bloc (STBC).

Les codes STTC sont une généralisation des modulations codées en treillis aux canaux MIMO. Bien que les performances obtenues par ces codes soient excellentes, la complexité du décodage est exponentielle avec le rendement. **[12]**

I.7.4.1 Codage spatio-temporel en treillis STTC :

Les premiers codes ST en Treillis ont été construits par Tarokh et al. Dans [14]. C'est une généralisation des codes en Treillis classiques (canal Gaussien) pour les systèmes MIMO.

Dans un code ST en Treillis les symboles à transmettre à chaque instant, par toutes les antennes émettrices représentent une transition dans le Treillis. Au début du codage d'une trame le codeur doit être à l'état 0. En fonction de l'état du codeur et des bits entrants, une transition est choisie à chaque instant t. Si l'étiquette de cette branche est $s_t^1, s_t^2, s_t^3 \ldots . s_t^n$ cela veut dire qu'à l'instant t, l'antenne j va transmettre le symbole s_t^j . La figure I.21 illustre un exemple de code en Treillis, pour nt= 2, utilisant une modulation 8-PSK. **[13]**

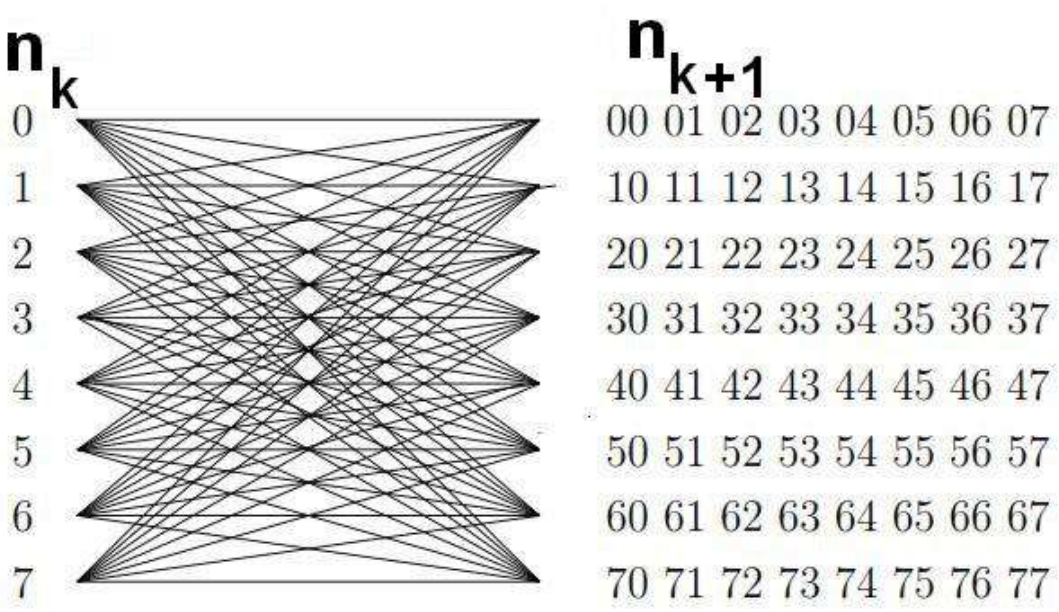


Figure I.21 : Exemple de code ST en Treillis à 8 états.

Le fonctionnement du codeur est relativement simple, et peut être résumé comme suit:

- $n_K$représente l'état du treillis à l'instant k et par conséquent l'état suivant est noté n_{K+1}
- Considérons que le treillis est à l'état initial n_K =0.
- L'état suivant du treillis dépend des bits d'information à coder. Ainsi, si la valeur décimale à coder est 1 alors l'état suivant prend les deux bits 01.
- Les symboles à droite du treillis sont les codes associés à chaque doublet d'éléments binaires entrants.
- Ces symboles sont alors mis en forme par la 8-PSK avant l'émission par leurs antennes respectives. **[11]**

Soit la trame {0, 1, 5, 7, 6,4} de symboles 8-PSK à transmettre, le mot de code s'écrit

$$\text{alors } \begin{Bmatrix} 0 & 0 & 1 & 5 & 7 & 6 \\ 0 & 1 & 5 & 7 & 6 & 4 \end{Bmatrix}$$

Le décodage d'un code en Treillis se fait par l'algorithme de Viterbi. Cet algorithme minimise une métrique additive sur tous les chemins dans le Treillis. La complexité de cet algorithme est exponentielle en fonction du nombre d'états du codeur, ce qui rend les codes en Treillis peu pratiques.

I.7.4.2 Codage spatio-temporel en bloc STBC :

Du point de vue historique, le premier schéma STBC et le plus simple a été proposé par S.M.Alamouti **[14]**. Il a présenté comme l'illustre les figures I.22 et I.23 deux systèmes de transmissions dotés d'une diversité de deux antennes en émission en considérant les cas d'un et de deux antennes en réception. Ces résultats sont très utilisés et cette technique est déjà introduite dans les standards de réseaux comme c'est le cas avec le W –CDMA et le CDMA-2000. On s'est servi de son schéma comme référence pour procéder à la généralisation aux cas beaucoup plus complexes de plusieurs antennes et même pour

On peut définir la technique d'Alamouti à partir de fonctions suivantes : la séquence de transmission et encodage, schème de combinaison au récepteur afin de faciliter la détection des symboles transmis avec une base du maximum de vraisemblance. [**14]**

I.7.4.2.1 Le cas d'une seule antenne de réception:

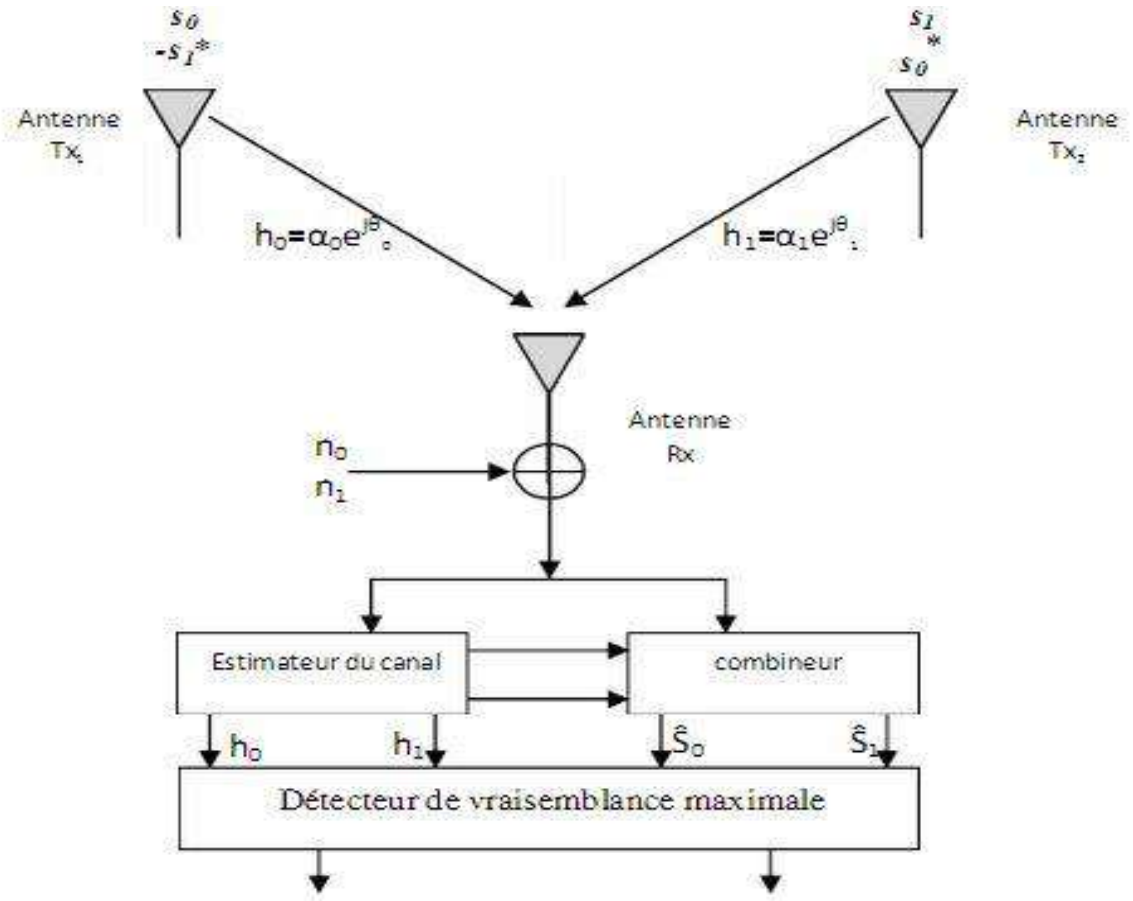


Figure I.22 : Schéma d'Alamouti avec deux antennes émettrices et une antenne réceptrice.

Séquence de transmission et encodage :

Supposant qu'on veut transmettre la séquence $\{S_0, S_1, S_2, \ldots\ldots S_n$. Dans le cas ordinaire, le symbole S_0 est transmis durant le premier intervalle de temps, S_1 durant le second intervalle, puis S_2 et ainsi de suite; Mais, dans le cas du code d'Alamouti les symboles sont groupés deux à deux et transmis simultanément par les deux antennes à un intervalle de temps donné. Durant le premier intervalle, les symboles S_0, S_1 sont simultanément transmis de la premiere et de la deuxième antenne. Dans le second intervalle c'est $-S_1^*$ et S_0^* qui sont transmis; Dans le troisième intervalle S_2, S_3 et dans le quatrième $-S_3^*$ et S_2^* et ainsi de suite; Notant que en groupant les symboles ainsi, en a toujours besoin d'un double intervalle de temps pour l'émission de deux symbole. Autrement dit, on conserve toujours le même débit que dans le cas d'une seule antenne.

	Antenne T	Antenne T
Temps t		
Temps t+T		

Tableau I.3: transmission dans le temps

Schéma de combinaison au récepteur :

Les signaux reçus sont :

$$r_0 = h_0 s_0 + h_1 s_1 + n_0$$
$$r_1 = -h_0 s_1^* + h_1 s_0^* + n_1 \quad (I.36)$$

Où r_0 et r_1 sont les signaux reçus aux temps **t** et **t+T**, n_0 et n_1 étant les variables aléatoires représentant le bruit et l'interférence au récepteur.

Avant que les signaux reçus soient envoyés au décodeur, ils sont combinés comme suit :

$$\widetilde{s_0} = h_0^* r_0 + h_1 r_1^*$$
$$\widetilde{s_1} = h_1^* r_0 - h_0 r_1^* \quad (I.37)$$

Et substituant (1.36) dans (1.37) on trouve :

$$\widetilde{s_0} = (h_0^2 + h_1^2) s_0 + h_0^* n_0 + h_1 n_1^*$$
$$\widetilde{s_1} = (h_0^2 + h_1^2) s_1^* + h_1^* n_0 - h_0 n_1^* \quad (I.38)$$

Les valeurs estimées $\widetilde{s_0}$ et $\widetilde{s_1}$ sont alors envoyés à un décodeur de maximum de vraisemblance pour estimer les symboles transmis S_0, S_1, respectivement. [**14**

I.7.4.2.2Cas de deux antennes en réception :

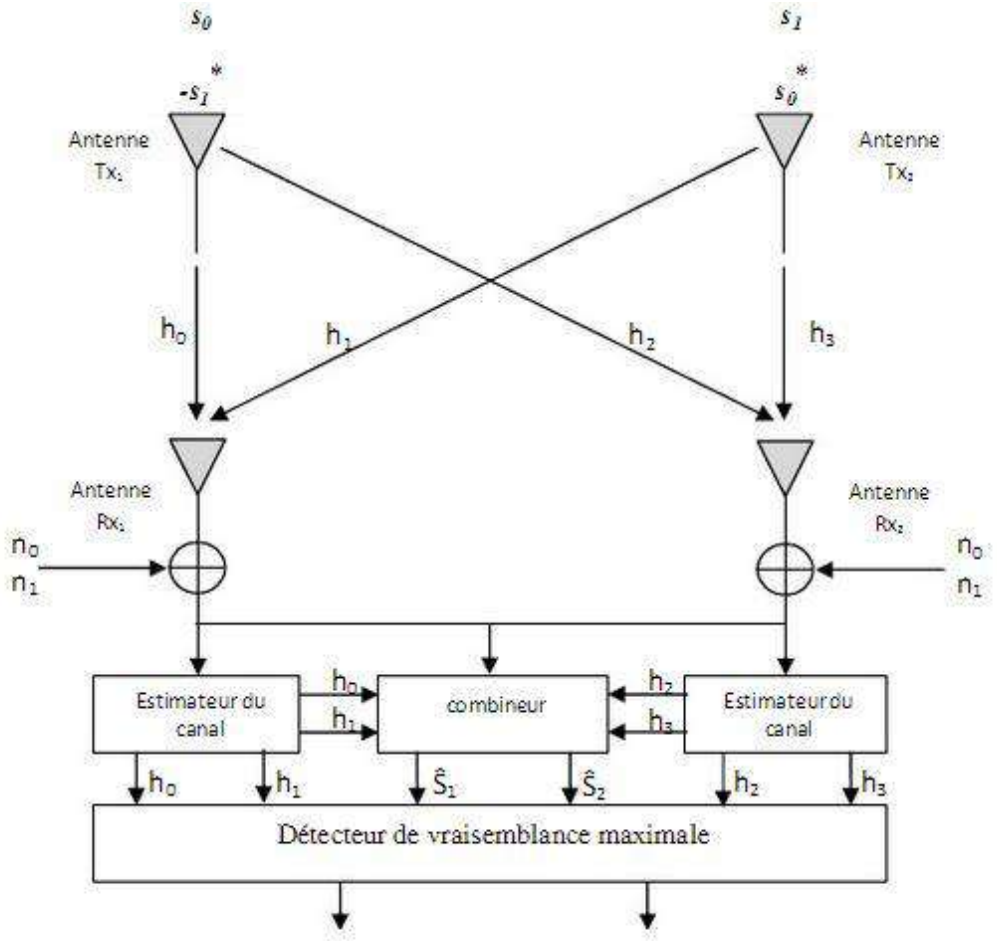


Figure I.23 : Schéma d'Alamouti avec deux antennes émettrices et deux antennes réceptrices

	Antenne R	Antenne R
Temps t		
Temps t+T		

Tableau I.4: Signaux reçus aux niveaux des antennes réceptrices.

	Antenne R	Antenne R
Antenne T	h_0	h_2
Antenne T	h_1	h_3

Tableau I.5: Canaux entre antennes émettrices et antennes réceptrices.

Le schéma consiste à la transmission de 2 symboles différentsS_0, S_1, respectivement à l'antenne 1 et 2 pendant la première période symbole. Pendant la deuxième période symbole, les symboles$-S_1^*$ et S_0^* sont transmis par les antennes 1 et 2 respectivement. La situation est résumée à la figure I.23.

Le codage et la séquence de symbole transmis reste le même que dans le cas d'un seul récepteur présentés dans le tableau I.3 les signaux reçus aux niveaux des deux antennes réceptrices sont présentés dans le tableau I.4 et dans le tableau I.5 sont représentés les canaux entre antennes émettrices et réceptrices. [14]

On en déduit le modèle suivant pour le signal reçu

$$\begin{aligned} r_0 &= h_0 s_0 + h_1 s_1 + n_0 \\ r_1 &= -h_0 s_1^* + h_1 s_0^* + n_1 \\ r_2 &= h_2 s_0 + h_3 s_1 + n_2 \\ r_3 &= -h_2 s_1^* + h_3 s_0^* + n_3 \end{aligned} \quad \text{(I.39)}$$

Et les signaux combinés sont [14] :

$$\begin{aligned} \tilde{s_0} &= h_0^* r_0 + h_1 r_1^* + h_2 r_2^* + h_3 r_3^* \\ \tilde{s_1} &= h_1^* r_0 - h_0 r_1^* + h_3 r_2^* - h_2 r_3^* \end{aligned} \quad \text{(I.40)}$$

Qui, après la substitution deviennent :

$$\begin{aligned} \tilde{s_0} &= (h_0^2 + h_1^2 + h_2^2 + h_3^2) s_0 + h_0^* n_0 + h_1 n_1^* + h_2^* n_2 + h_3 n_3^* \\ \tilde{s_1} &= (h_0^2 + h_1^2 + h_2^2 + h_3^2) s_1^* + h_1^* n_0 - h_0 n_1^* + h_3^* n_2 - h_2 n_3^* \end{aligned} \quad (I.41)$$

I.8 Conclusion:

Dans les systèmes de communications traditionnels, il n'existe qu'une antenne à l'émission et une antenne à la réception. Or les futurs services de communications mobiles sans fils demandent plus de transmissions de données (augmentation de la capacité de transmission). Ainsi pour augmenter la capacité et satisfaire ces demandes, les bandes passantes de ces systèmes et les puissances à transmettre ont été largement augmentés. Mais l'utilisation de plusieurs antennes à l'émission et à la réception (Système MIMO) permettait d'augmenter le débit de transmission des données et l'amélioration de la fiabilité de la transmission et cela sans augmenter ni la bande passante de l'antenne réceptrice, ni la puissance du signal à l'émission. La différence majeure avec les systèmes exploitant le domaine spatial sur un seul des deux sites, provient du fait que les systèmes MIMO peuvent exploiter bénéfiquement les trajets multiples alors que les autres systèmes cherchent à lutter contre les trajets multiples et leurs effets sur le système.

Chapitre II

II.1Introduction :

L'électronique moderne se tourne de plus en plus vers le numérique qui présente de nombreux avantages sur l'analogique : grande insensibilité aux parasites, reconfigurable, facilité de stockage de l'information etc... .

Aujourd'hui les techniques de traitement numérique occupent une place majeure dans tous les systèmes électroniques modernes grand public, professionnels ou de défense. De plus, les techniques de réalisation de circuits spécifiques, tant dans les aspects matériels (composants reprogrammables circuits pré caractérisés et bibliothèques de macro fonctions) que dans les aspects logiciels (placement-routage, synthèse logique) font désormais de la microélectronique l'une des bases indispensables pour la réalisation de systèmes numériques performants.

II.2 Les circuits FPGA (Field Programmable Gate Array):

Les FPGA (Field Programmable Gate Arrays) sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court.

Le progrès de ces technologies permet de faire des composants toujours plus rapides et à plus haute intégration, ce qui permet de programmer des applications importantes. Cette technologie permet d'implanter un grand nombre d'applications et offre une solution d'implantation matérielle à faible coût pour des compagnies de taille modeste pour qui, le coût de développement d'un circuit intégré spécifique implique un trop lourd investissement. **[15]**

II.3 Description des circuits FPGA de XILINX :

L'architecture, retenue par Xilinx, se présente sous forme de deux couches :

- une couche appelée circuit configurable,
- une couche réseau mémoire SRAM.

La couche dite 'circuit configurable' est constituée d'une matrice de blocs logiques configurables CLB permettant de réaliser des fonctions combinatoires et des fonctions séquentielles .

Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées/sorties IOB dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs (figure II.1).

La programmation du circuit FPGA appelé aussi LCA (logic cells arrays) consistera par le biais de l'application d'un potentiel adéquat sur la grille de certains transistors à effet de champ à interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux. Ces potentiels sont tout simplement mémorisés dans le réseau mémoire SRAM. **[16, 17]**

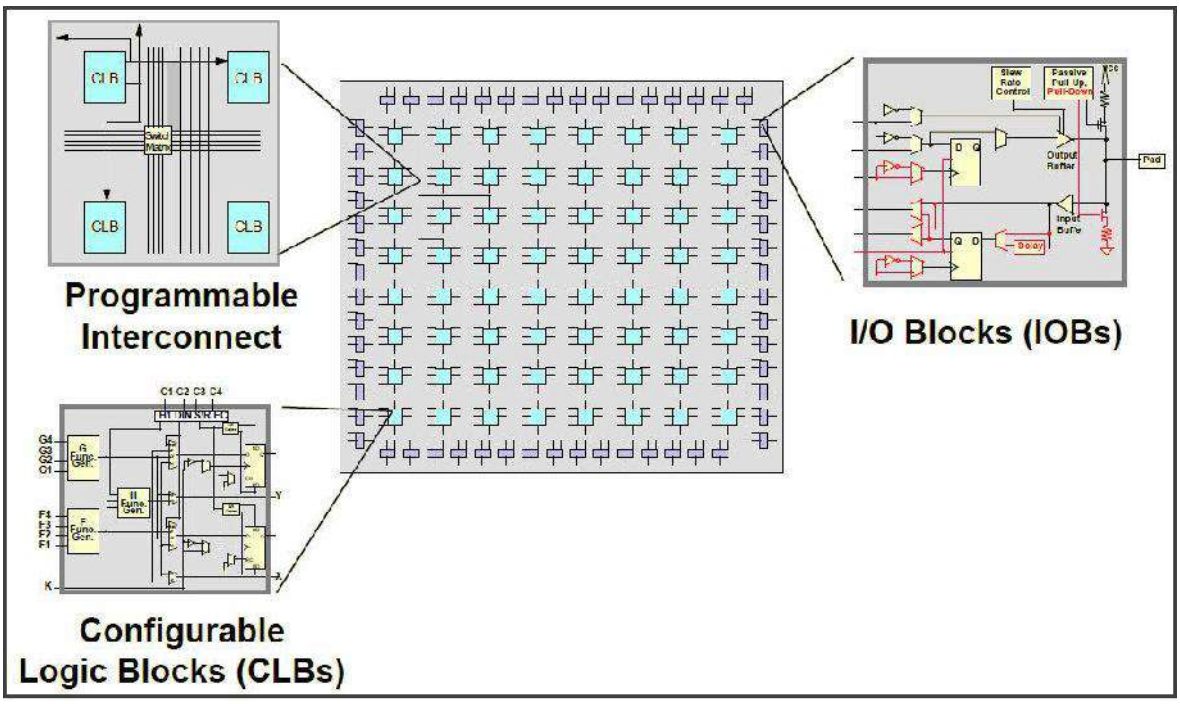


Figure II.1 : Architecture interne d'un FPGA.

II.3.1 Les CLB (configurable logic bloc):

Les blocs logiques configurables sont les éléments déterminants des performances du FPGA. Chaque bloc est composé d'un bloc de logique combinatoire composé de deux générateurs de fonctions à quatre entrées et d'un bloc de mémorisation synchronisation composé de deux bascules D. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB. La figure II.2, nous montre le schéma d'un CLB.

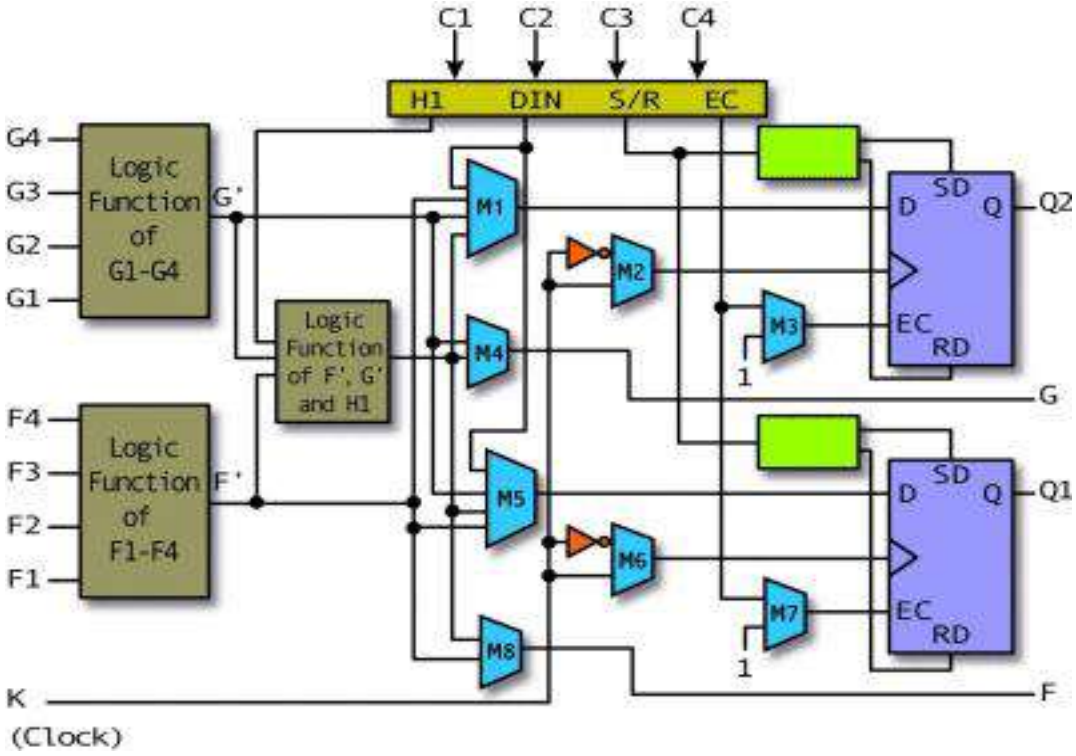


Figure II.2 : Cellule logique (CLB).

Voyons d'abord le bloc logique combinatoire qui possède deux générateurs de fonctions F' et G' à quatre entrées indépendantes (F1...F4, G1...G4), lesquelles offrent aux concepteurs une flexibilité de développement importante car la majorité des fonctions aléatoires à concevoir n'excède pas quatre variables. Les deux fonctions sont générées à partir d'une table de vérité câblée inscrite dans une zone mémoire, rendant ainsi les délais de propagation pour chaque générateur de fonction indépendants de celle à réaliser. Une troisième fonction H' est réalisée à partir des sorties F' et G' et d'une troisième variable d'entrée H1 sortant d'un bloc composé de quatre signaux de contrôle H1, DIN, S/R, Ec. Les signaux des générateurs de fonction peuvent sortir du CLB, soit par la sortie G, pour les fonctions H' et G', soit F pour les fonctions F' et H'. Ainsi un CLB peut être utilisé pour réaliser

- deux fonctions indépendantes à quatre entrées indépendantes
- ou une seule fonction à cinq variables
- ou deux fonctions, une à quatre variables et une autre à cinq variables.

L'intégration de fonctions à nombreuses variables diminue le nombre de CLB nécessaires, les délais de propagation des signaux et par conséquent augmente la densité et la vitesse du circuit. Les sorties de ces blocs logiques peuvent être appliquées à des bascules au nombre de

deux ou directement à la sortie du CLB (sorties G et F). Chaque bascule présente deux modes de fonctionnement : un mode 'flip-flop' avec comme donnée à mémoriser, soit l'une des fonctions F', G', H' soit l'entrée directe DIN. La donnée peut être mémorisée sur un front montant ou descendant de l'horloge (CLK). Les sorties de ces deux bascules correspondent aux sorties du CLB Q1 et Q2. Un mode dit de " verrouillage " exploite une entrée S/R qui peut être programmée soit en mode SET, mise à 1 de la bascule, soit en Reset, mise à zéro de la bascule. Ces deux entrées coexistent avec une autre entrée laquelle n'est pas représentée sur la figure II.2 appelée le global Set/Reset. Cette entrée initialise le circuit FPGA à chaque mise sous tension, à chaque configuration, en commandant toutes les bascules au même instant soit à '1', soit à '0'. Elle agit également lors d'un niveau actif sur le fil RESET lequel peut être connecté à n'importe quelle entrée du circuit FPGA.

Un mode optionnel des CLB est la configuration en mémoire RAM de 16x2bits ou 32x1bit. Les entrées F1 à F4 et G1 à G4 deviennent des lignes d'adresses sélectionnant une cellule mémoire particulière. La fonctionnalité des signaux de contrôle est modifiée dans cette configuration, les lignes H1, DIN et S/R deviennent respectivement les deux données D0, D1 (RAM 16x2bits) d'entrée et le signal de validation d'écriture WE. Le contenu de la cellule mémoire (D0 et D1) est accessible aux sorties des générateurs de fonctions F' et G'. Ces données peuvent sortir du CLB à travers ses sorties G et F ou alors en passant par les deux bascules. [16, 17]

II.3.2 Les IOB (input output bloc):

La figure II.3 présente la structure de ce bloc. Ces blocs entrée/sortie permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé (haute impédance). **[16, 17]**

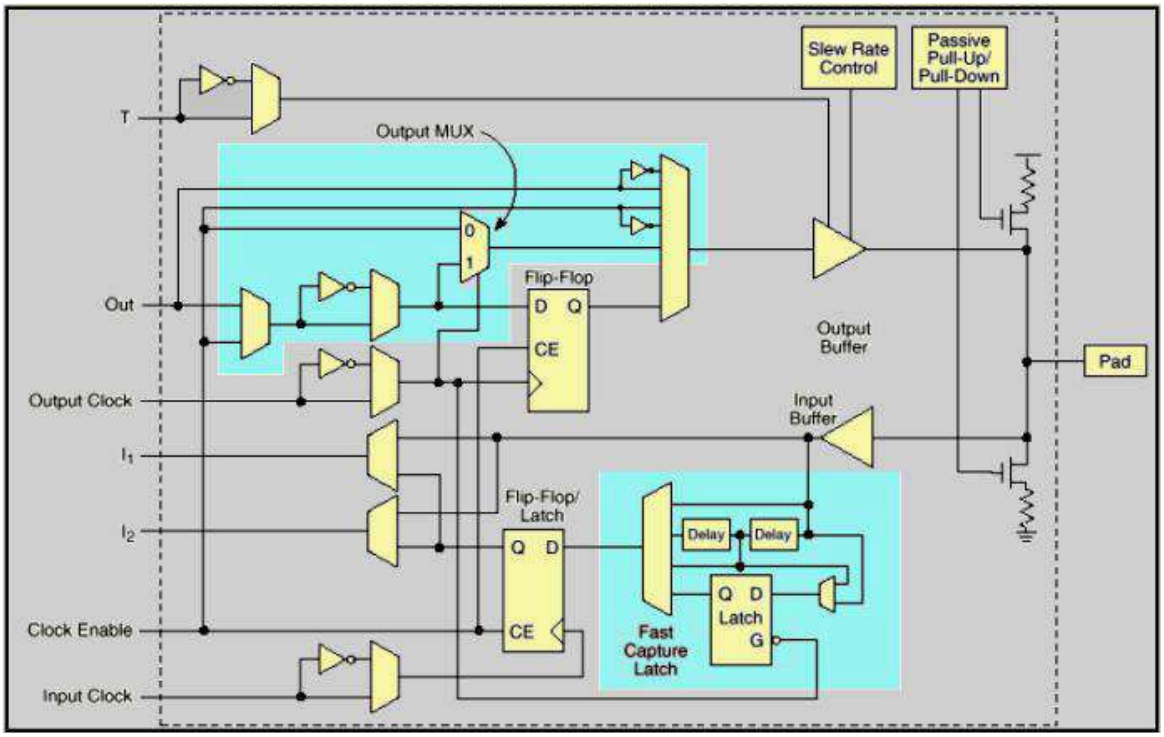


Figure II.3: Input Output Block (IOB).

- **Configuration en entrée:**

Le signal d'entrée traverse un buffer qui selon sa programmation peut détecter soit des seuils TTL ou soit des seuils CMOS. Il peut être routé directement sur une entrée directe de la logique du circuit FPGA ou sur une entrée synchronisée. Cette synchronisation est réalisée à l'aide d'une bascule de type D, le changement d'état peut se faire sur un front montant ou descendant. De plus, cette entrée peut être retardée de quelques nanosecondes pour compenser le retard pris par le signal d'horloge lors de son passage par l'amplificateur. Le choix de la configuration de l'entrée s'effectue grâce à un multiplexeur (program controlled multiplexer). Un bit positionné dans une case mémoire commande ce dernier. **[16, 17]**

- **Configuration en sortie:**

Nous distinguons les possibilités suivantes :

- Inversion ou non du signal avant son application à l'IOB.
- Synchronisation du signal sur des fronts montants ou descendants d'horloge,
- Mise en place des résistances Pull-Up et Pull-Down dans le but de limiter la consommation des entrées sorties inutilisées,
- Signaux en logique trois états ou deux états. Le contrôle de mise en haute impédance et la réalisation des lignes bidirectionnelles sont commandés par le signal de commande Out Enable lequel peut être inversé ou non. Chaque sortie peut délivrer un

courant de 12mA. Ainsi toutes ces possibilités permettent au concepteur de connecter au mieux une architecture avec les périphériques extérieurs. **[16, 17]**

II.3.3 Les interconnexions:

Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé. Pour parvenir à cet objectif, Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons. Nous disposons:

- interconnexions à usage général.
- interconnexions directes.
- de longues lignes. **[16, 17]**

Les interconnexions à usage général:

Ce système fonctionne en une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes de CLB et de l'IOB. **[16, 17]**

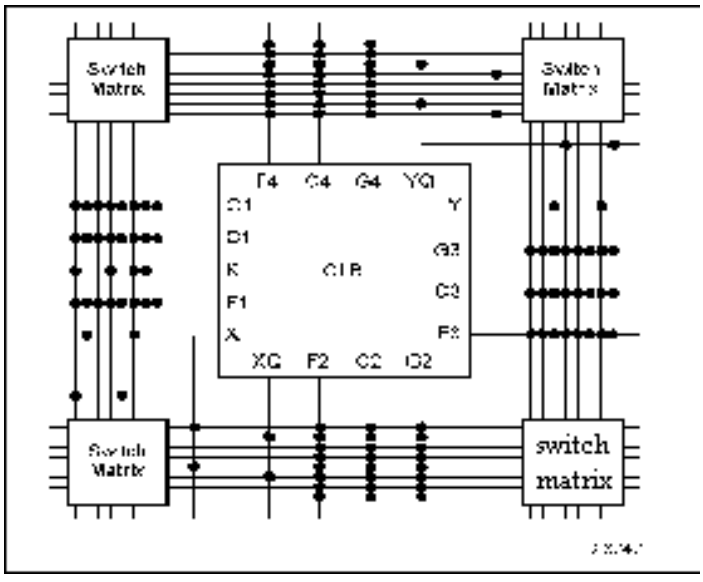


Figure II.4 : Connexions à usage général

Des aiguilleurs appelés aussi matrices de commutation sont situés à chaque intersection. Leur rôle est de raccorder les segments entre eux selon diverses configurations, ils assurent ainsi la communication des signaux d'une voie sur l'autre. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre. Pour éviter que les signaux traversant les grandes lignes ne soient affaiblis, nous trouvons généralement des buffers implantés en haut et à droite de chaque matrice de commutation.

Interconnexions directes:

Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en termes de vitesse et d'occupation du circuit. De plus, il est possible de connecter directement certaines entrées d'un CLB aux sorties d'un autre. **[16, 17].**

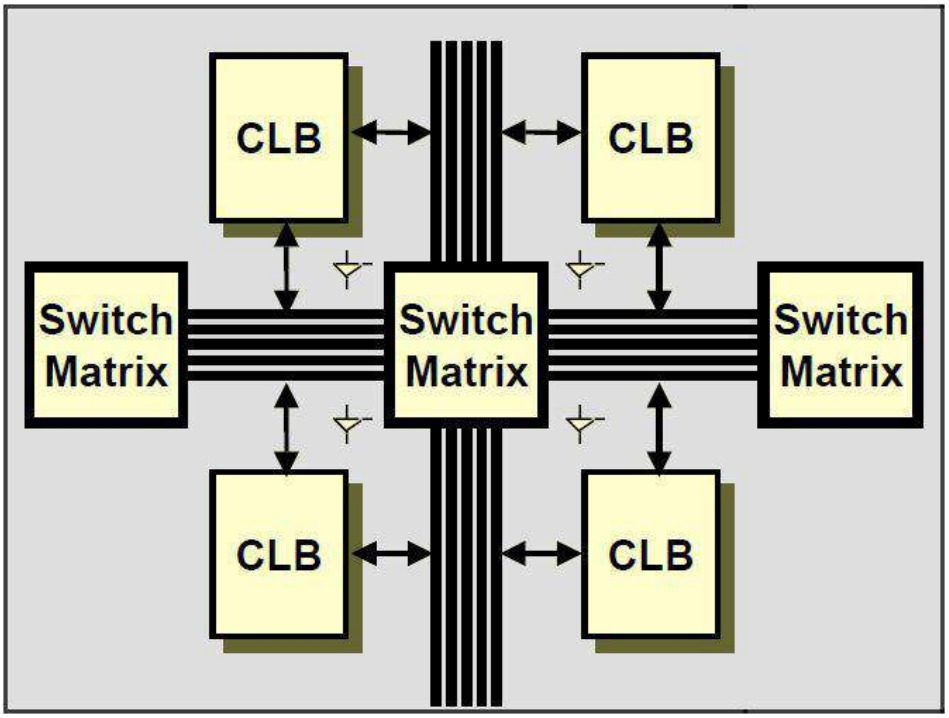
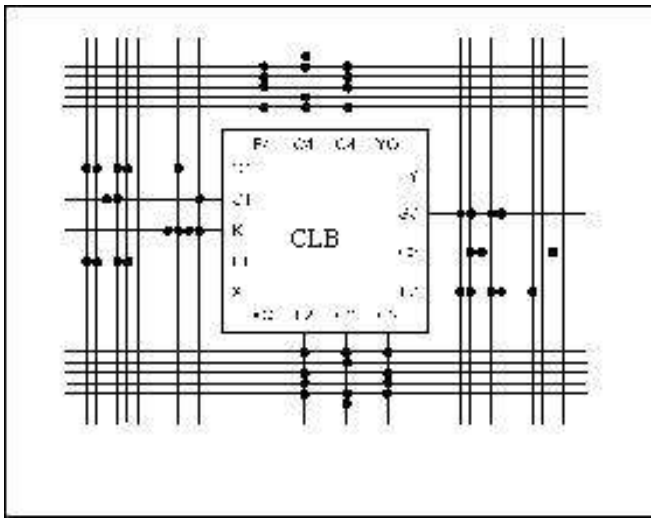


Figure II.5 : Les interconnexions directes.

Les longues lignes:

Les longues lignes sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion. **[16, 17]**



FigureII.6 : Les longues lignes.

Performances des interconnexions:

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions à usage général, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connectique de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ils possèdent une faible résistance mais une capacité importante. De plus, si on utilise un aiguilleur, sa résistance s'ajoute à celle existante. **[16, 17].**

II.3.4 Evolution des composants logiques FPGA de XILINIX :

FPGA est ciblé pour des applications à forte valeur ajoutée :

- Réseaux et télécommunications (routage de dorsale, stations de base cellulaires, ...)
- Vidéonumérique (effets visuels, encodage, ...)
- Traitement numérique du signal
- Émulation d'ASIC
- Modulation / Démodulation implémentée de manière logicielle (Software Defined Radio)

La première génération des Virtex a introduit l'intégration de mémoire double-port DPRAM et de boucles à verrouillage de délai (DLL) directement au sein de la matrice du FPGA.

Les générations suivantes ont apporté, en plus d'une augmentation des performances et des capacités logiques, des multiplieurs dédiés, puis un processeur PowerPC et transceivers série multi-gigabit (permettant de supporter directement le Gigabit Ethernet, et plus récemment le PCI-Express ainsi que le Serial ATA). Les multiplieurs dédiés sont ensuite devenus des blocs DSP dédiés à part entière

Nom	L'année	Process de fabrication (nm, couche, V)	Nouveautés introduites	Transceivers	PowerPC	Remarques
Virtex	**1998**	**220; 5 ; 2,5**	**DLL, DPRAM**	**Non**	**Non**	**Modèle basé sur le XC4000**
Virtex-E	**1999**	**180 ; 6;1,8**	**Support LVDS**	**Non**	**Non**	
Virtex-E EM	**2000**	**180 ; 6 ; 1,8**	**Quantité de DPRAM en hausse**	**Non**	**Non**	
Virtex-II	**2001**	**150; 8 ; 1,5**	**Multiplieurs câblés**	**Non**	**Non**	
Virtex-II Pro	**2002**	**130 ; 9 ; 1,5**	**Transceivers ; PowerPC**	**3.125 Gbit/s**	**Oui**	
Virtex-II Pro X	**2003**	**130 ; 9 ; 1,5**	**Transceivers 10 Gbit/s[3]**	**4.25+ Gbit/s**		**Production arrêtée;**
Virtex-4 LX	**2004**	**90 ; 11;1,2**	**Blocs DSP câblés**	**Non**	**Non**	**Plus de ressources logiques que le Virtex-4 SX**
Virtex-4 SX	**2004**	**90; 11;1,2**	**Blocs DSP câblés**	**Non**	**Non**	**Plus de blocs DSP que Virtex-4 LX**
Virtex-4 FX	**2004**	**90 ; 11 ; 1,2**	**Blocs DSP câblés**	**6.125 Gbit/s**	**Oui**	**Transceivers susceptibles aux variations PVT**

Virtex-5 LX	**2006**	**65 ; 12 ; 10**	**LUT à 6 entrées**	**Non**	**Non**	**Plus de ressources logiques que le Virtex-5 SXT**
Virtex-5 LXT	**2007**	**65 ; 12;1,0**	**LUT à 6 entrées**	**3.125 Gbit/s**	**Non**	**Plus de ressources logiques que Virtex-5 SXT**
Virtex-5 SXT	**2007**	**65; 12 ; 1,0**	**LUT à 6 entrées**	**3.125 Gbit/s**	**Non**	**Plus de blocs DSP que Virtex-5 LX/LXT**
Virtex-5 FXT	**2008**	**65 ; 12 ; 1,0**	**LUT à 6 entrées et GTX**	**6.5 Gbit/s**	**Oui**	**Même chose que LXT et avec 2 cœurs** **PowerPC440**
Virtex-6 LX et LXT	**2009**	**40**		**6.5 Gbit/s**	**Non**	
Virtex-6 SXT	**2009**	**40**		**6.5 Gbit/s**	**Non**	**Optimisé pour les applications de** traitement
Virtex-7	**2010**	**28**		**13.1 Gbit/s**		

Tableau II.1 : Évolutions de la série Virtex.

Nom complet	Introduction	Référence	Process de fabrication	Remarques
Spartan		XCSxx-	5 V	
Spartan-XL	1998	XCSxxXL-	3,3 V	
Spartan-II	2000	XC2Sxxx-	2,5 V	
Spartan-IIE	2000	XC2SxxxE-	1,8 V	Basé sur le Virtex-E
Spartan-3	2003	XC3Sxxxx-		Basé sur le Virtex-II
Spartan-6	02/02/ 2009	XC6SLXxxx(T)-	1,2 V, 45nm, 9 couches de métal	
Spartan-6 XA	02 /03/2010	XA6SLXxxx(T)-	1,2 V, 45nm,	Automotive-grade

			9 couches de métal	FPGAs
Spartan-6Q	08/11/2010	XQ6SLXxxx(T)-	1,2 V, 45nm, 9 couches de métal	Defense-grade FPGA

Tableau II.2 : Évolutions de la série Spartan.

II.4 Les atouts des circuits FPGA :

Performances : Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. BDTI, une importante société d'analyse et de « benchmarking », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application.

Temps de mise sur le marché : Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, on peut tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé. Par la suite, on peut apporter les éventuelles modifications nécessaires au circuit FPGA, en quelques heures au lieu de quelques semaines. Le matériel actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cœurs de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux.

Coût : Les coûts de développement des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC se justifie largement pour les grands fabricants, par exemple, qui peuvent livrer des circuits par milliers. Cependant, la plupart des utilisateurs finaux ont

besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la préconception d'un ASIC.

Fiabilité : Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche.

Maintenance à long terme : Comme nous l'avons vu, les circuits FPGA sont évolutifs et nous épargnent donc la dépense de temps et d'argent qu'implique la préconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, on peut y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit. **[22]**

II.5 Les étapes d'implémentation d'une application sur un FPGA :

Pour implémenter une application sur un FPGA, quatre étapes importantes sont à suivre tel illustré par la figure II.7

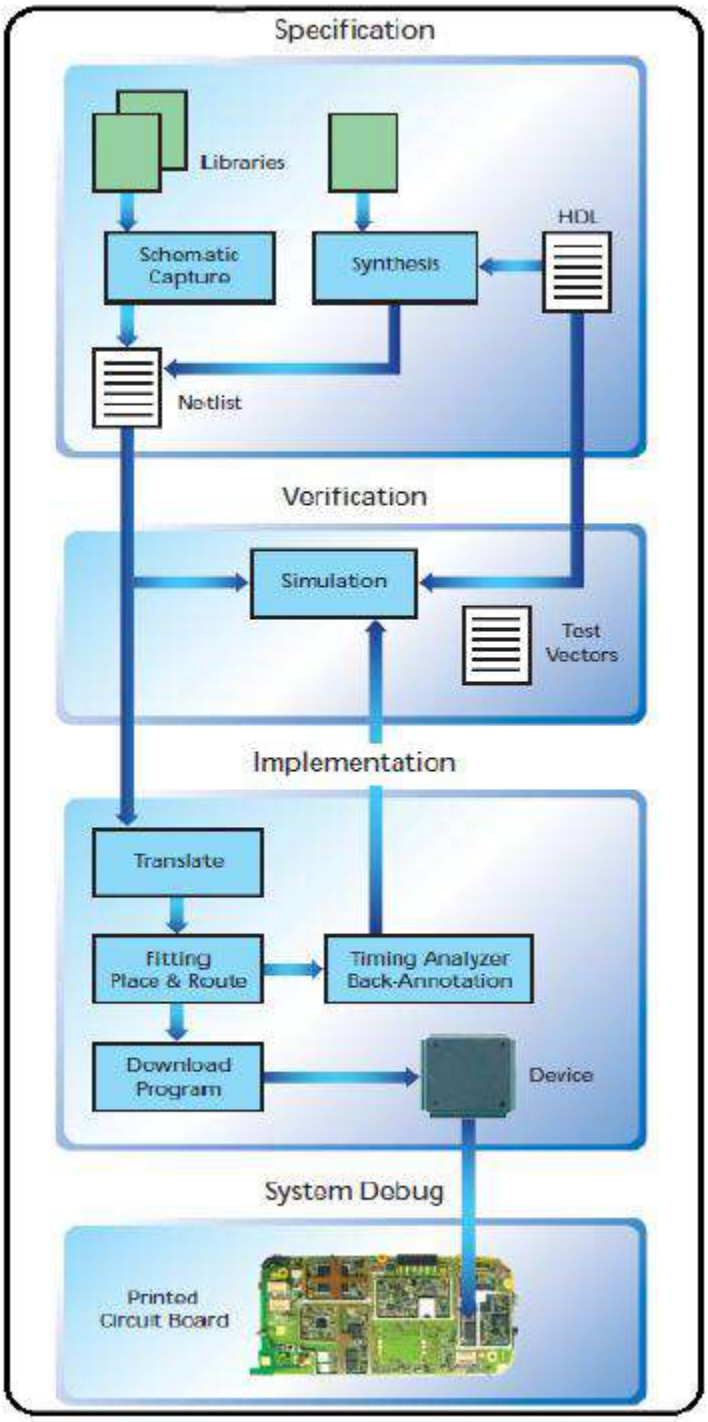


Figure II.7: Les étapes d'implémentation d'une application sur un circuit FPGA.

II.5.1 Spécification :

Le terme spécification est un terme qui regroupe les trois modes (schématique, diagrammes d'états ou HDL) de saisie d'un circuit électronique. La spécification est synthétisée pour générer un fichier appelé NETLIST qui décrit les interconnexions entre les registres.

II.5.2 Vérification :

La vérification du design est une étape parallèle ou le concepteur observe le comportement du code et observe s'il se comporte tel qu'il est supposé. Un simulateur simule le circuit à condition de lui fournir les vecteurs de test. Les vecteurs de test peuvent se présenter sous plusieurs formes, la plus courante est TEST BENCHS rédigés dans un langage de description matériel ou sous forme de lignes de code TCL pour entrer les instructions au simulateur. En appliquant les vecteurs de test sur le code, le simulateur fournit les sorties du circuit.

II.5.3 Implémentation :

Une fois le Netlist (les interconnexions entre les portes logiques) décrit la conception en utilisant les portes logiques en tenant compte du fabricant et d'une famille de composant bien définie. Une fois la vérification est terminée, le circuit est implémenté sur le composant en spécifiant les références exactes de celui-ci à savoir : le boitier, la fréquence de travail et les autres options spécifiques à chaque composant. Cette étape se termine par un rapport de tous les sous-programmes exécutés, les avertissements, les erreurs, les entrées /sorties utilisés, des données qui permettent de savoir si le composant choisit est le mieux adapté pour l'application ciblée.

Cette étape se décompose à son tour en sous-étapes que voici :

Fitting : c'est le dimensionnement de la conception en fonction des ressources internes du composant cible

Place and Route : les sous-programmes de placement et routage sont exécutés après compilation du code.

Place : c'est le processus de sélection des modules ou blocks logiques où les portes logiques seront placées.

Route : le routage est l'interconnexion physique entre les différents blocks logiques.

II.5.4 Downloding ou Programing :

Après que le fichier de programmation est généré l'étape suivante consiste à le charger sur le FPGA ciblé par l'application. Les FPGA de type SRAM nécessite une mémoire de type PROM pour retenir le programme même après extinction de la carte électronique comportant le FPGA, au démarrage il y a chargement ou Downloding des interconnexions sur le FPGA. Dans le cas des FPGA à anti fusibles, le programme peut être chargé ou Programing par l'intermédiaire d'un programmateur ou par technique de programmation sur circuit JTAG.

II.6 Techniques de Programmation des FPGA de Xilinx:

La configuration du circuit est mémorisée sur la couche réseau SRAM et stockée dans une ROM externe. Un dispositif interne permet à chaque mise sous tension de charger la SRAM interne à partir de la ROM. Ainsi on conçoit aisément qu'un même circuit puisse être exploité successivement avec des ROM différentes puisque sa programmation interne n'est jamais définitive. On voit toute la partie que l'on peut tirer de cette souplesse en particulier lors d'une phase de mise au point. Une erreur n'est pas rédhibitoire, mais peut aisément être réparée.

Les circuits FPGA ne possèdent pas de programme résident. A chaque mise sous tension, il est nécessaire de les configurer. Leur configuration permet d'établir des interconnexions entre les CLB et IOB. Pour cela, ils disposent d'une RAM interne dans laquelle sera écrit le fichier de configuration. Le format des données du fichier de configuration est produit automatiquement par le logiciel de développement sous forme d'un ensemble de bits organisés en champs de données. Le FPGA dispose de quatre modes de chargement et de trois broches M0, M1, M2 lesquelles définissent les différents modes. Ces modes définissent les différentes méthodes pour envoyer le fichier de configuration vers le circuit FPGA, selon deux approches complémentaires :

- configuration automatique, le circuit FPGA est autonome,
- configuration externe, l'intervention d'un opérateur est nécessaire.
- M0 M1 M2 Mode sélectionné

0 0 0 maître série
0 0 1 maître parallèle bas
0 1 1 maître parallèle haut
1 0 1 périphérique
1 1 1 esclave série

- **Mode maître série:**

  Le mode maître série utilise une mémoire à accès série de type registre à décalage commercialisée par le fabricant Xilinx. Le programme est préalablement chargé par le système de développement utilisé pour le circuit FPGA. Le FPGA génère tous les signaux de dialogue nécessaires pour la copie du contenu de la PROM dans sa RAM interne, lorsque la copie est terminée, il bascule le signal DONE pour le signaler au circuit. Une seule PROM peut configurer plusieurs circuits FPGA avec la même configuration ou plusieurs PROM peuvent configurer plusieurs FPGA en chaîne où le premier des circuits FPGA est le maître et

génère l'horloge. Les données en provenance des PROM sont envoyées aux autres circuits FPGA par la sortie DOUT de ce premier.

- On dispose aussi d'un **mode maître-parallèle** où le FPGA est relié en parallèle à une EPROM classique, de même qu'un mode passif **type périphérique** dans lequel le FPGA est considéré comme un périphérique de µP et peut-être configuré à partir de celui-ci.
- **Mode esclave:**

  Dans ce mode, le programme de configuration peut être envoyé à partir d'un PC, d'une station de travail ou à partir d'un autre circuit FPGA. Le circuit FPGA maître peut être interfacé à une mémoire en mode parallèle ou série sans apporter aucune modification au niveau du câblage des circuits FPGA esclaves. C'est souvent le mode exploité pour la mise au point d'une configuration. **[19, 17, 16]**

II .7 La plateforme de développement ATLYS Spartan 6:

• Package BGA (à matrice de billes) 324 broches, Xilinx Spartan-6 LX45 FPGA
• Ports USB2 intégrés pour programmation et transfert de données
• Port USB-UART et USB-HID
• DDR2 de 128 Mo avec données étendues sur 16 bits
• Livré avec un bloc d'alimentation de 20 W et un câble USB

Le Kit de développement de réseaux logiques programmables (FPGA) Atlys Spartan 6 de Digilent est une plateforme de développement de circuits numériques complète et prête à l'emploi basée sur un réseau de portes programmables (FPGA) Xilinx Spartan 6 LX45. Le grand nombre de périphériques haut de gamme intégrés, tels que l'Ethernet à 1 Gb, la vidéo HDMI, une barrette mémoire de 128 Mo, des ports audio et USB, fait de la carte Atlys un support idéal pour les systèmes numériques complets conçus sur des processeurs embarqués, tels que le MicroBlaze de Xilinx.

Le Spartan-6 LX45 est optimisé pour des circuits logiques de haute performance et offre :
• 6 822 slices (tranches) comprenant chacune quatre LUT (table de correspondance) à 6 entrées et huit bascules
• Bloc de mémoire RAM de 2,1 Mbits
• 4 tuiles d'horloge (8 DCM et 4 PLL)
• 6 boucles à verrouillage de phase

• 58 slices DSP

• Vitesses d'horloge de plus de 500 MHz

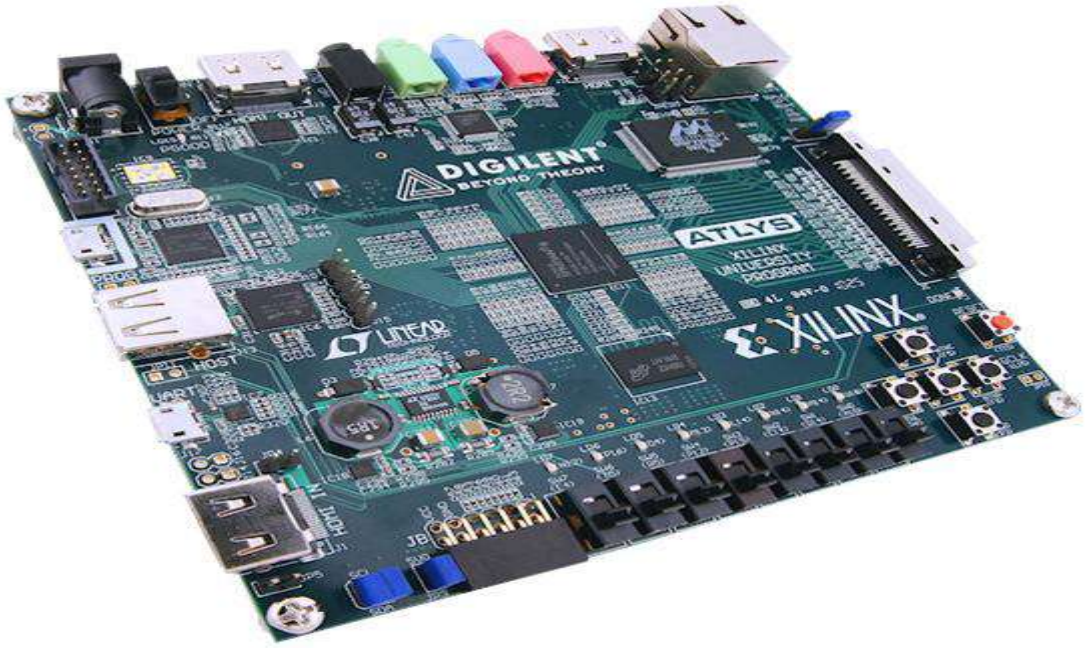


Figure II.8 : Vue externe de plateforme ATLYS Spartan 6

II.8 Conclusion :

Dans ce chapitre, nous avons présenté les circuits de type FPGA dont nous précisons ici les principaux avantages :

- ✓ Le premier argument est la souplesse de programmation qui permet l'emploi conjoint d'outils de schématique aussi bien que l'exploitation d'un langage de haut niveau tel VHDL
- ✓ Le second argument est évidemment la nouvelle possibilité de reconfiguration dynamique partielle ou totale d'un circuit ce qui permet d'une part, une meilleure exploitation du composant, une réduction de surface de silicium employé et donc du coût, et d'autre part, une évolutivité assurant la possibilité de couvrir à terme des besoins nouveaux sans nécessairement repenser l'architecture dans sa totalité
- ✓ Notons enfin que ces circuits n'ont pas vocation à concurrencer les super calculateurs, mais plutôt à offrir une alternative en fonction de critères comme l'encombrement, les performances et le prix, et sont de ce fait bien adaptés à des applications de qualité dans le domaine des systèmes ambulatoires ou nomades.

Chapitre III

III.1 Introduction :

Dans ce chapitre nous allons présenter la conception ainsi que les résultats de la simulation et l'implémentation d'un codeur/décodeur d'Alamouti.

Pour la conception et la simulation nous avons utilisé l'outil « System Generator » compatible avec l'environnement Simulink. Ensuite nous avons convertie notre design sous langage VHDL afin de compléter les différentes étapes d'implémentation. La figure III.1 illustre le flot de conception suivi :

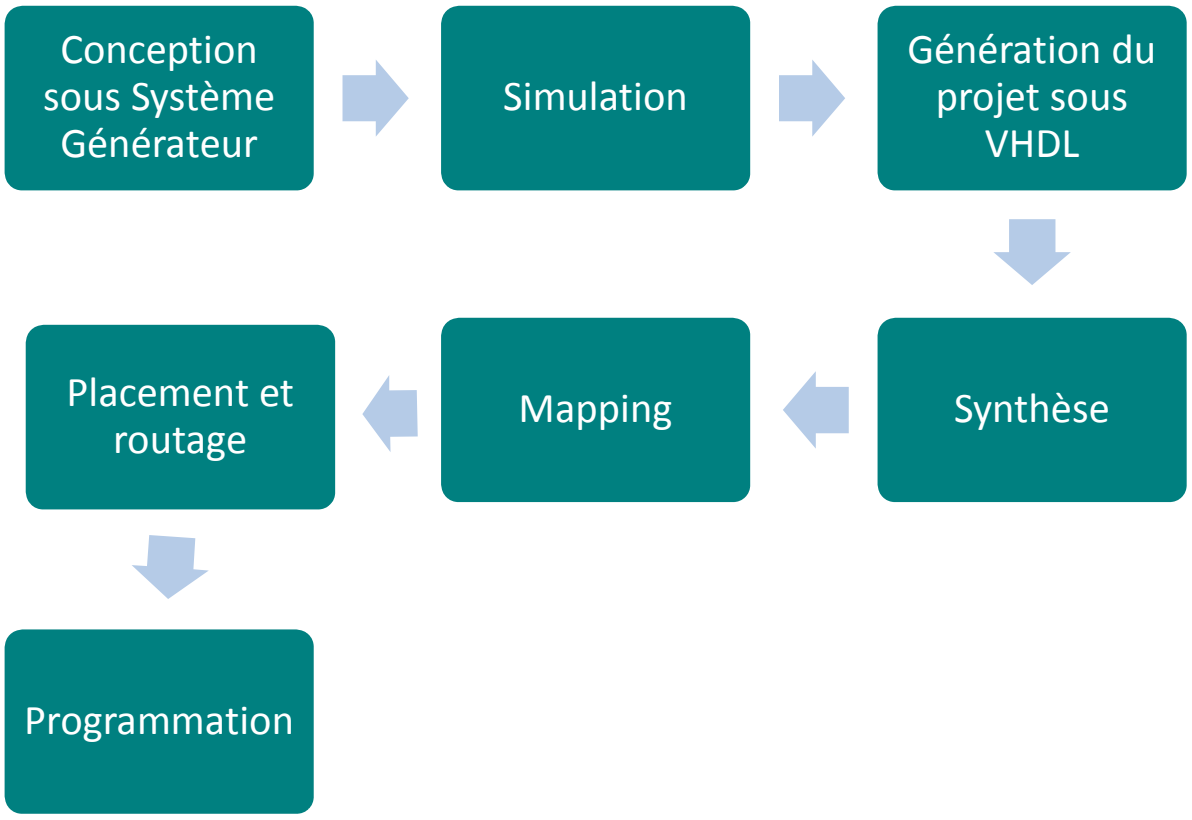
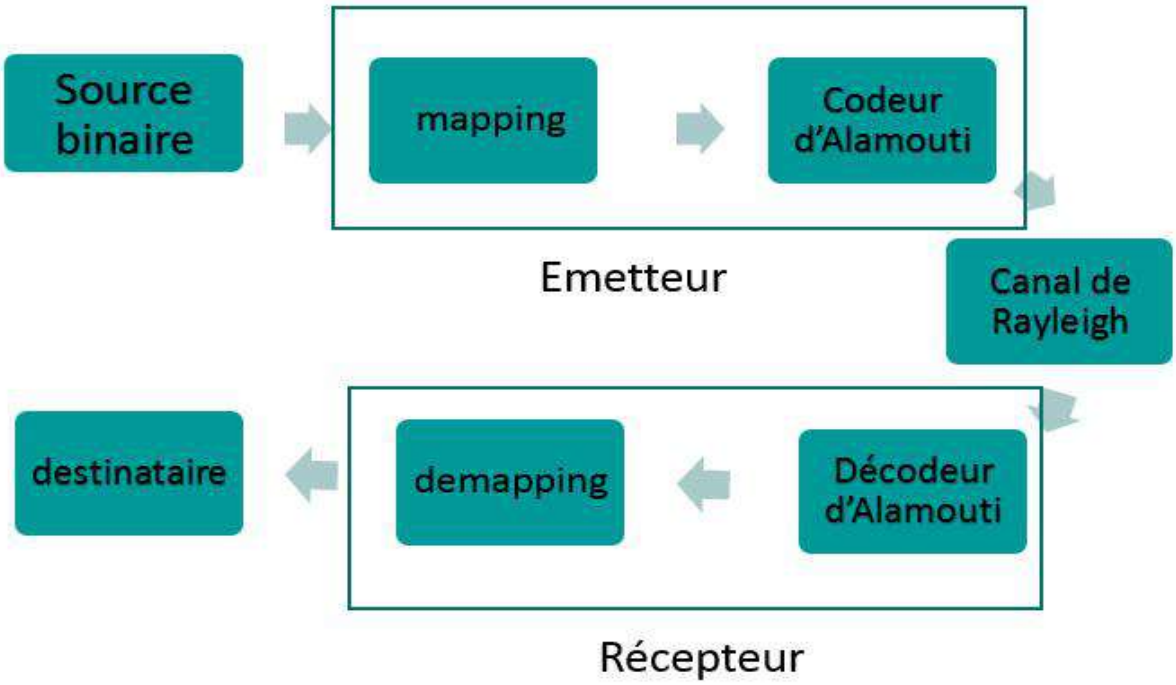


Figure III.1: flot de conception.

Le synoptique général de notre conception et donné comme suivant :



La figure III.2 : Schéma fonctionnel de codeur/décodeur d'Alamouti.

Pour le mapping nous avons examiné deux types de modulations qui sont QPSK et BPSK.

Pour le décodage d'Alamouti nous avons traité le cas d'une seule antenne de réception (système MISO) ainsi que le cas de deux antennes de réception (système MIMO).

III.2 Schéma de l'émetteur :

La figure III.3 montre le schéma fonctionnel de l'émetteur d'Alamouti pour les deux modulations BPSK et QPSK qui se composent de deux parties essentielles le mapping et l'encodeur d'Alamouti. Ces parties se composent de plusieurs blocs qui seront détaillés dans le reste de ce chapitre.

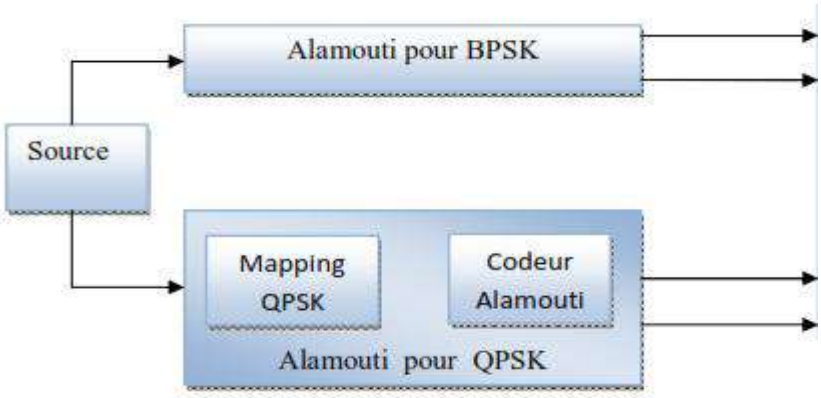


Figure III.3 : Schéma fonctionnel de l'émetteur avec codage d'Alamouti

D'abord, pour le cas d'une modulation QPSK l'information binaire issue de la source est convertie en symboles à quatre états possibles (deux bits par symbole) en utilisant le bloc **serial to parallel**. C'est l'étape du mapping. Ce bloc produit des symboles QPSK, qui seront passés à l'encodeur d'Alamouti.

Nous constatons que pour le cas de la modulation BPSK le bloc du mapping n'est pas utilisé par ce que c'est une modulation binaire. Nous pouvons donc utiliser directement les bits issus de la source pour réaliser le codage d'Alamouti où le codage s'effectue sur des nombres réels (uniquement la composante en phase).

III.2.1 Codeur d'Alamouti pour la modulation BPSK:

Le codeur d'Alamouti génère les trois symboles ($x_1, x_2,$ [illegible]) formant les données à envoyer alternativement.

Les données codées à l'aide de l'encodeur sont reparties selon leur voie. Une voie sera transmise par la première antenne d'émission T_{x1} .L'autre sera transmise sur la deuxième antenne d'émission T_{x2}. Pendant la première période les symboles notés x_1 sont envoyé à travers T_{x1} alors que x_2 sont envoyés sur T_{x2}.

Pendant la deuxième période les symboles [illegible] sont envoyé à travers T_{x1} et les symboles notés x_1 sont envoyé à traversT_{x2}.

Les deux symboles sont envoyés pendant 2 périodes mais les deux symboles sont envoyés avec une diversité spatiale et une diversité temporelle.

Comme exemple nous avons utilisé une source d'information de type :**Repeating séquence staire** qui génère la séquence [1 1 1 1 1 0 0 1 1 0 1 0].

Apres l'utilisation du codage d'Alamouti qui est donné par la matrice :

$$\begin{Bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{Bmatrix}$$

A l'instant t_i on trouve :

Sur tx_1 : x_1 = [1 1 1 0 1 1]

Sur tx_2 : x_2 = [1 1 0 1 0 0]

A l'instant t_i+1 on trouve :

Sur tx_1 : $\overline{x_2}$ = [0 0 1 0 1 1]

Sur tx_2 : x_1 = [1 1 1 0 1 1]

On ajoute des multiplexeurs pour obtenir les séquences à envoyer sur T_{x1} et T_{x2} ou le premier **TDM** permet le multiplexage entre x_1 et $\overline{x_2}$ ce qui donne la séquence envoyé sur T_{x1} .Le deuxième **TDM** permet le multiplexage entre x_2 et x_1 qui donne la séquence envoyé sur T_{x2}.

La séquence envoyée sur la première antenne est T_{x1} : [1 0 1 0 1 1 0 0 1 1 1 1]

La séquence envoyée sur la deuxième antenne est T_{x2} : [1 1 1 1 0 1 1 0 0 1 0 1]

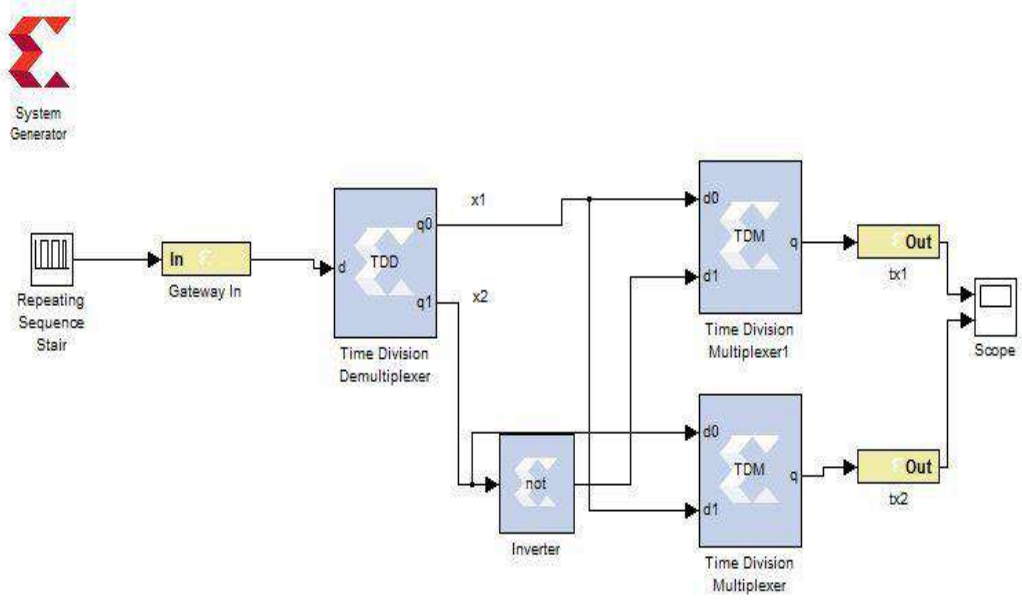


Figure III.4 : Codeur d'Alamouti pour la modulation BPSK.

La simulation nous a donné les résultats suivants :

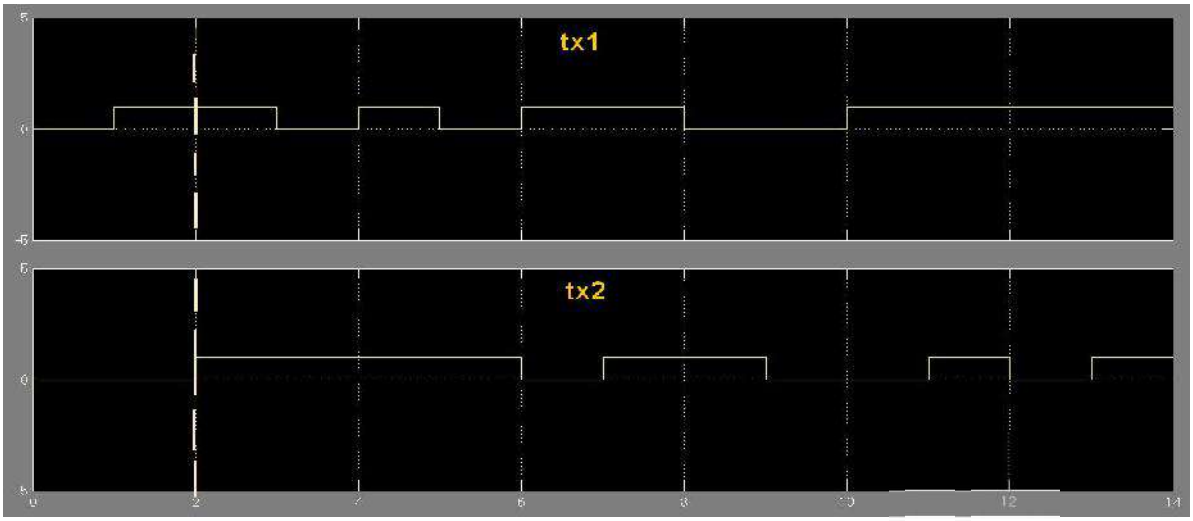


Figure III.5 : Les signaux transmis sur les antennes T_{x1} et T_{x2}.

On remarque que les signaux obtenus par la simulation sont correctes à l'exception des deux premiers bits, bit de retard dû au retard initial du **TDD**.

III.2.2 Codeur pour la modulation QPSK :

Opération de mapping :

Dans ce cas l'émetteur est composé de deux blocs : le bloc du mapping, et le bloc de codage d'Alamouti.

Pour le mapping QPSK nous avons utilisé un diagramme de constellation à quatre points, à équidistance autour d'un cercle. Avec quatre phases, QPSK peut coder deux bits par symbole, avec le code de Gray.

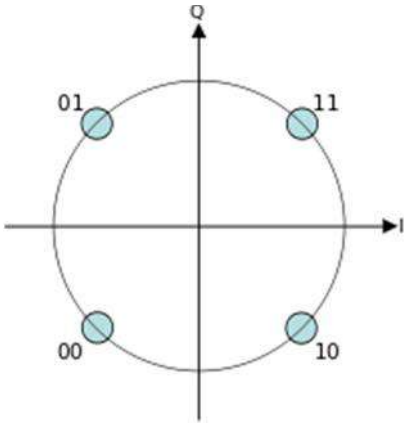


Figure III.6 : Diagramme de constellation QPSK.

Deux valeurs sont possibles pour la partie réelle (composantes en phase) et deux valeurs pour la partie imaginaire (composantes en quadrature). Pour le codage de la partie réelle et la partie imaginaire nous avons utilisé le montage présenté dans la **figure III.7**

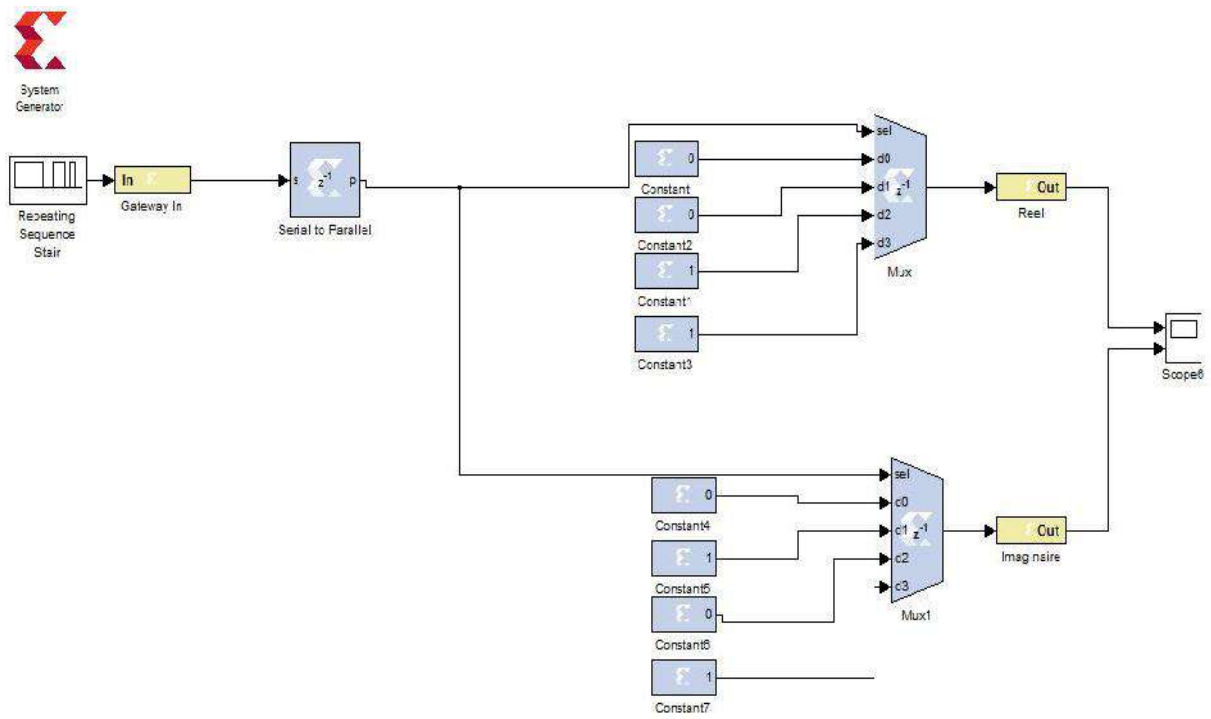


Figure III.7 : Mapping de la modulation QPSK.

On gardant toujours la séquence d'entrée du paragraphe III.2.1

Théoriquement on obtient six (6) symboles complexes dont leurs parties sont :

La séquence de la partie réelle : [1 1 1 0 1 1]

La séquence de la partie imaginaire : [1 1 0 1 0 0]

Apres la simulation on a obtenu les résultats suivants :



Figure III.8 : Résultats de la simulation du mapping.

On remarque que les résultats sont compatibles avec la théorie mais avec deux premiers bits supplémentaires liés au retard du circuit et on remarque que la période d'échantillonnage a doublé du a l'utilisation du bloc **serial to parallel**.

Codage d'Alamouti :

On a utilisé deux démultiplexeurs pour générer les deux partie réel $x_1$et $x_2$et les deux partie imaginairex_1, x_2.

Et pour générer les deux séquences en phase et en quadrature qui seront envoyé sur les antennes d'émission T_{x1}et T_{x2}on a utilisé des multiplexeurs. Le schéma suivant montre les blocs utilisés.

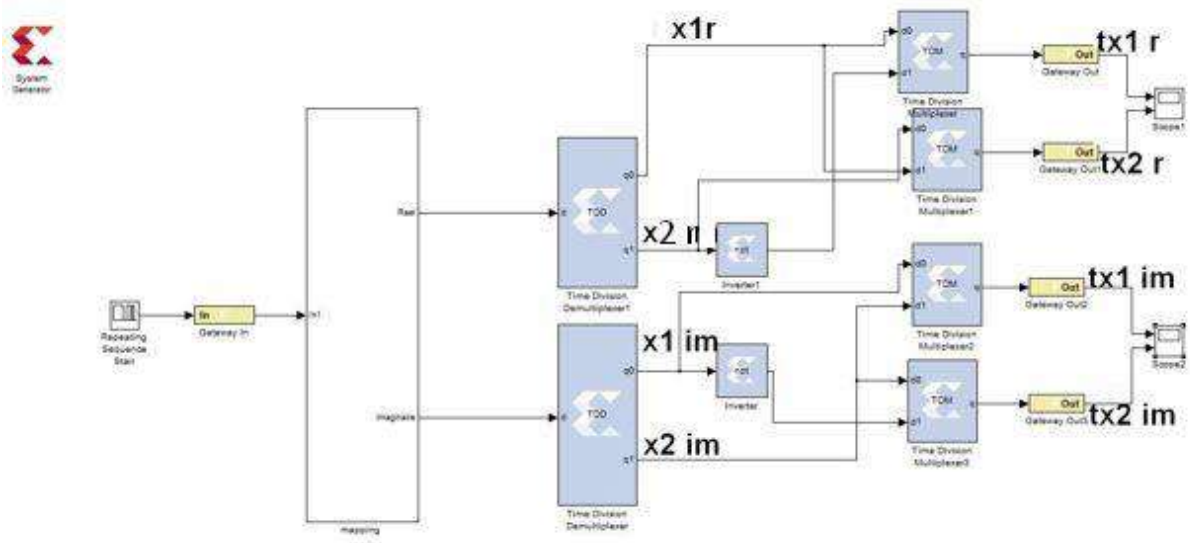


Figure III.9 : Codeur d'Alamouti pour la modulation QPSK.

Le calcul théorique est le suivant :

- ✓ Pour la partie réelle : [1 1 1 0 1 1]

La composante en phase qui sera envoyé sur la première antenne T_{x1}est : [1 0 1 1 1 0]

La composante en phase qui sera envoyé sur la deuxième antenne T_{x2}est : [1 1 0 1 1 1]

- ✓ Pour la partie imaginaire : [1 1 0 1 0 0]

La composante en quadrature qui sera envoyé sur la première antenne T_{x1}: [11 0 1 0 0]

La composante en quadrature qui sera envoyé sur la deuxième antenne T_{x2}: [10 1 1 0 1]

La simulation a donnée :

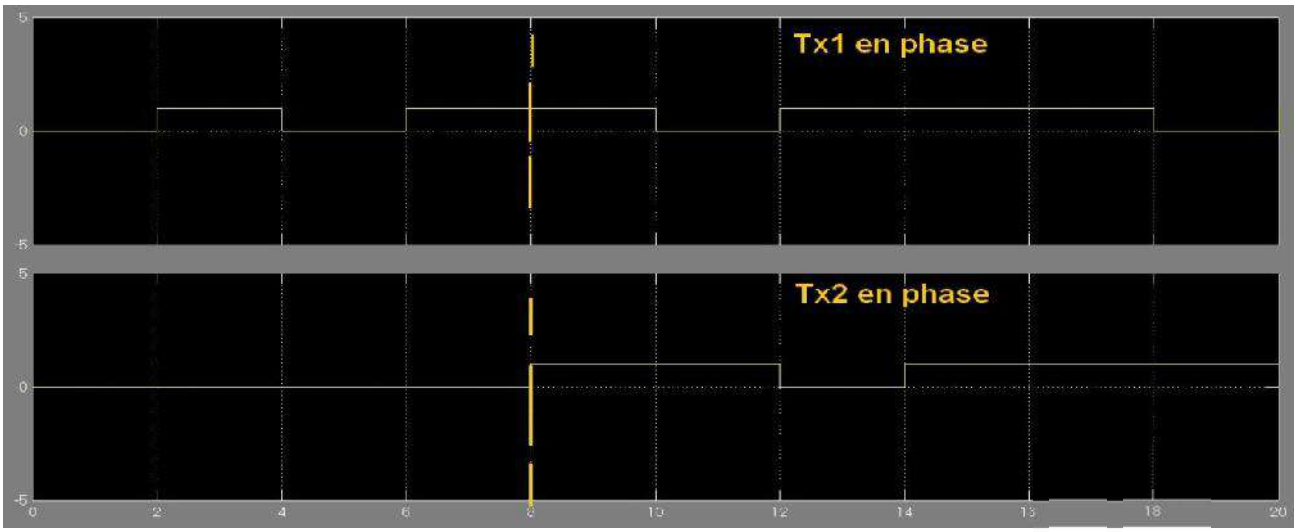


Figure III.10 : Les séquences réelles des signaux transmis sur les antennes T_{x1}et T_{x2}.

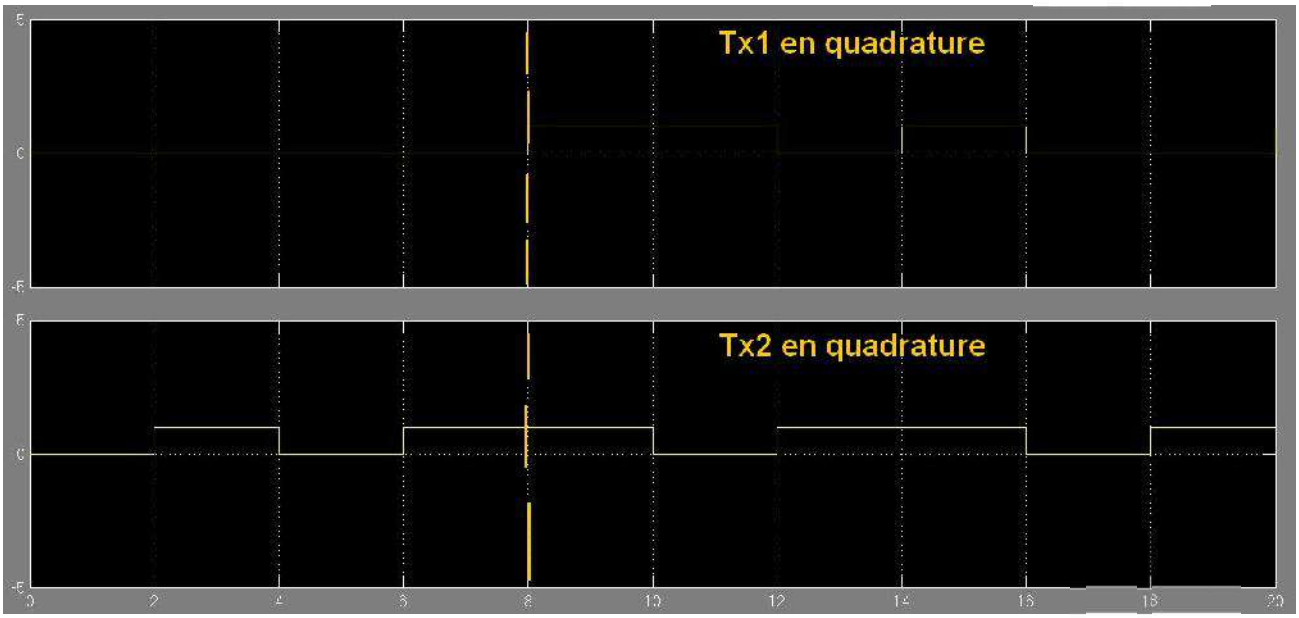


Figure III.11 : Les séquences imaginaires des signaux transmis sur les antennes T_{x1}etT_{x2}.

On remarque que les résultats obtenus coïncident avec la théorie mais avec quatre bit supplémentaires du au retard des circuits utilisés.

III.3 Canal de transmission :

Pour la simulation du canal nous avons utilisé un modèle de Rayleigh avec un bruit blanc additif (AWGN) selon le schéma de la **figure III.12**

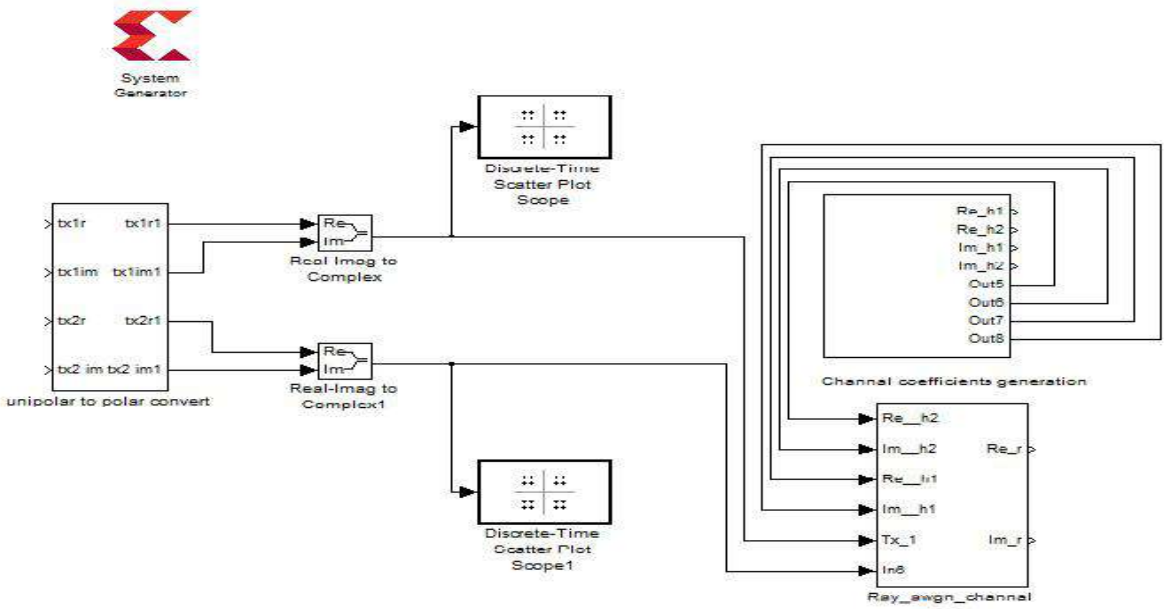


Figure III.12 : model du canal de transmission de Rayleigh.

Pour l'obtention des deux signaux complexes à envoyer sur T_{x1} et T_{x2}, nous avons d'abord utilisé quatre bloc **Unipolar to Bipolar Converter** pour faire la conversion de l'unipolaire ver le bipolaire des deux parties réel T_{x1}, T_{x2} et les deux parties imaginaire T_{x1}, T_{x2}. Puis on rajoute deux blocs qui font le passage du réel imaginaire ver le complexe. À l'aide de **Discrete-Time Scatter Plot Scope** on peut voir le résultat de la constellation QPSK.

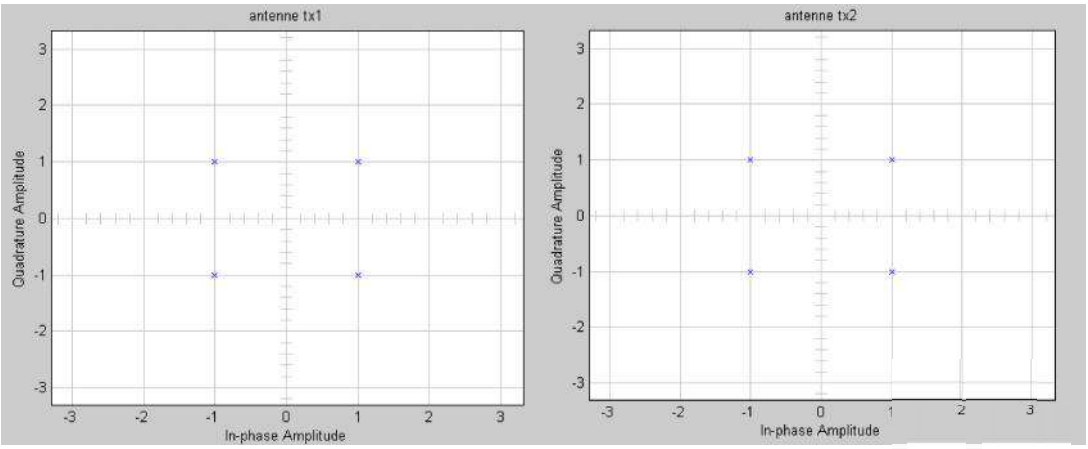


Figure III.13 : La constellation des deux signaux à envoyer sur T_{x1} et T_{x2}.

Les deux signaux complexes T_{x1} et T_{x2} sont d'abord multipliés par les coefficients complexes du canal de Rayleigh qui sont générés par le bloc « Channel Coefficients Generation » puis on leurs ajoute un bruit blanc additif (AWGN).

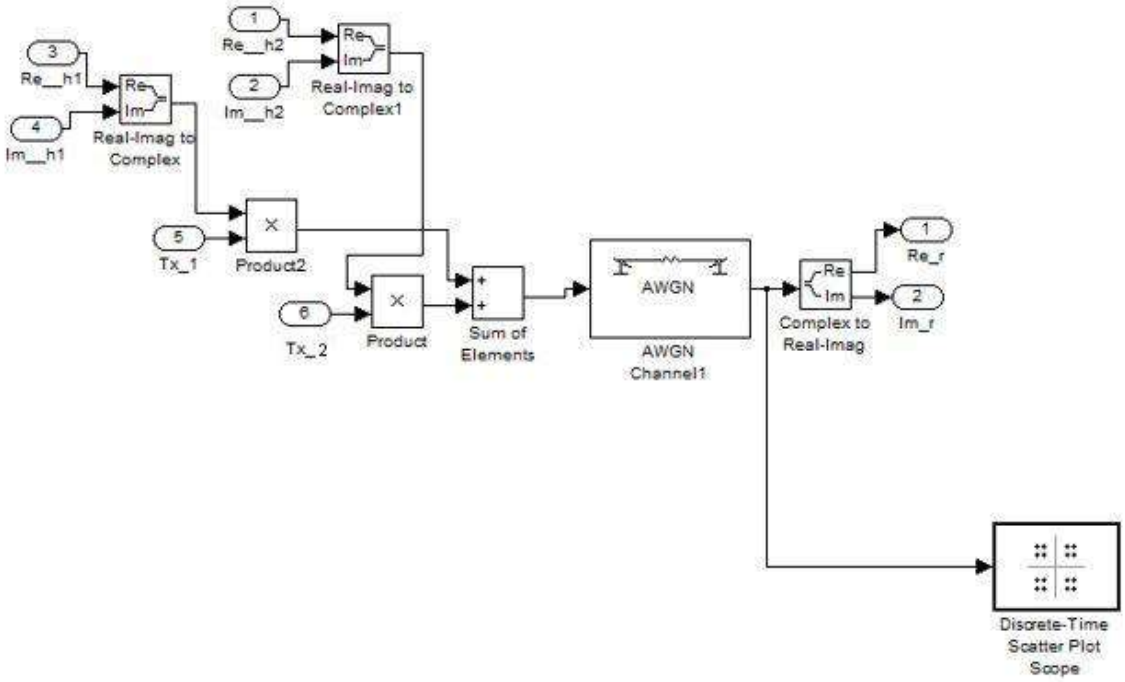


Figure III.14 : Canal de Rayleigh.

III.4 Récepteur à base d'un décodeur d'Alamouti :

La figure III.15 montre le schéma de principe du récepteur qui se compose de deux sous-blocs : le décodage d'Alamouti et le Demapping. Le signal reçu sur l'antenne est introduit dans le décodeur d'Alamouti, où le signal est décodé en fonction de la diversité de la technique d'Alamouti. Après le décodage, le signal est ensuite introduit dans le bloc de demapping pour récupérer le train de bits binaire d'origine.



Figure III.15 : Schéma d'un récepteur d'Alamouti.

III.4.1 Décodeur d'Alamouti :

III.4.1.1 Cas d'une seule antenne de réception :

Le décodage d'Alamouti est basé sur l'estimation des deux séquences $\widetilde{x_1}$ et $\widetilde{x_2}$ selon :

$$\widetilde{x_1} = h_1^* r_1 + h_2 r_2^* \quad \text{(III.1)}$$

$$\widetilde{x_2} = h_2^* r_1 - h_1 r_2^* \quad \text{(III.2)}$$

Ce qui donne :

$$\text{Re}\widetilde{x_1} = \text{Re_h}_1 . Re_\text{r}_1 + Re_\text{h}_2 . Re_\text{r}_2 + Im_\text{h}_1 . Im_r_1 + Im_h_2 . Im_r_2 \quad \text{(III.3)}$$

$$\text{Im}\ \widetilde{x_1} = Re_h_1 . Im_r_1 - Re_h_2 . Im_r_2 - Im_h_1 . Re_r_1 + Im_h_2 . Re_r_2 \quad \text{(III.4)}$$

$$\text{Re}\ \widetilde{x_2} = Re_{h_2} . Re_{\text{r}_1} - Re_h_1 . Re_r_2 + Im_{h_2} . Im_{\text{r}_1} - Im_h_1 . Im_r_2 \quad \text{(III.5)}$$

$$\text{Im}\ \widetilde{x_2} = Re_h_2 . Im_r_1 + Re_h_1 . Im_r_2 - Im_h_2 . Re_r_1 - Im_h_1 . Re_r_2 \quad \text{(III.6)}$$

Pour réaliser ces équations on a utilisé un ensemble de 16 multiplieurs, 6 additionneurs et 6 soustracteurs. Le schéma global de notre travail est le suivant :

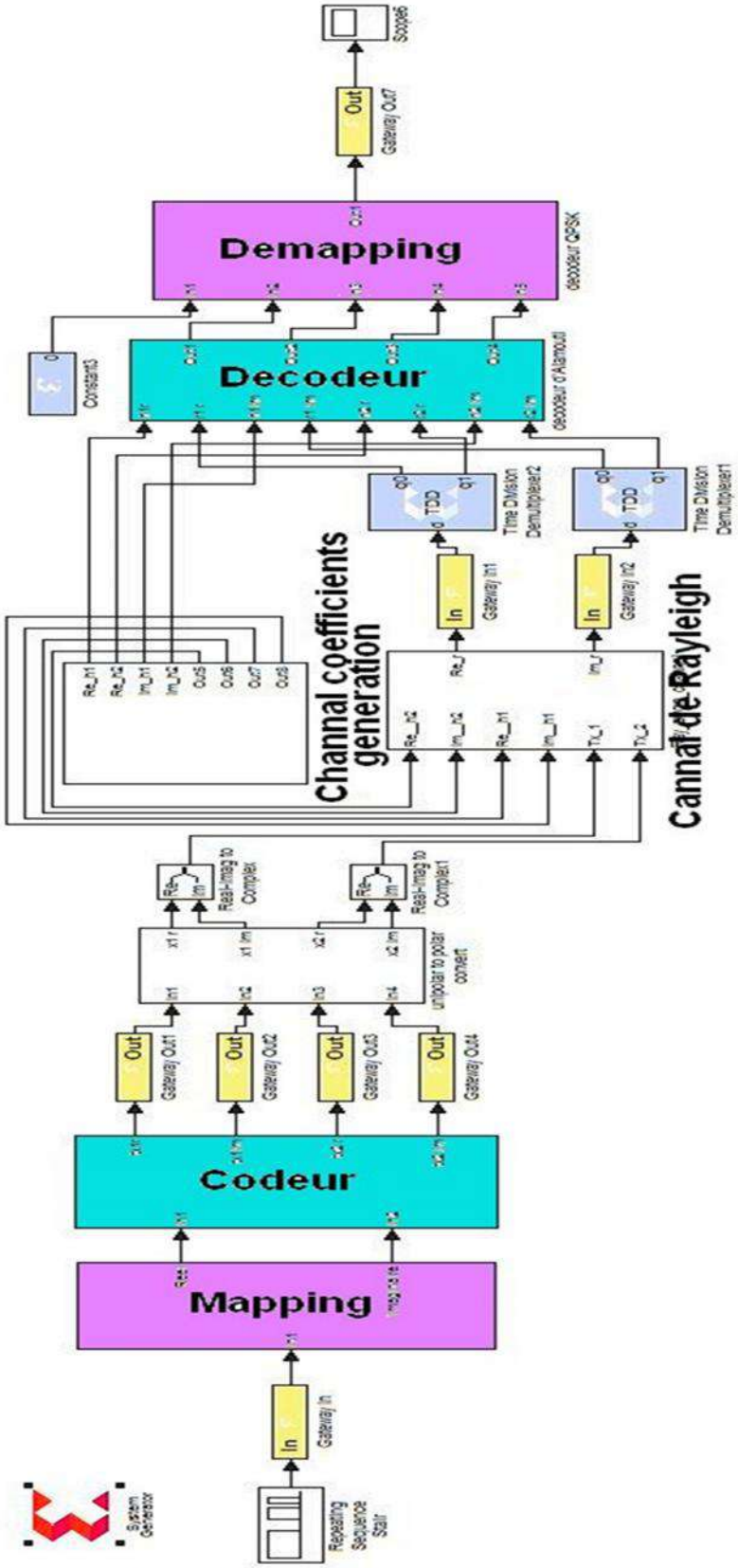


Figure III.16 : Schéma global d'un système de transmission MISO.

III.4.1.2 Cas de deux antennes de réception :

L'estimation des deux séquences $\widetilde{x_1}$ et $\widetilde{x_2}$ est donnée par :

$$\widetilde{x_1} = h_1^* r_1 + h_2 r_2^* + h_3{}^* r_3 + h_4 r_4^* \quad \text{(III.7)}$$

$$\widetilde{x_2} = h_2^* r_1 - h_1 r_2^* + h_4^* r_3 - h_3 r_4^* \quad \text{(III.8)}$$

Ce qui donne :

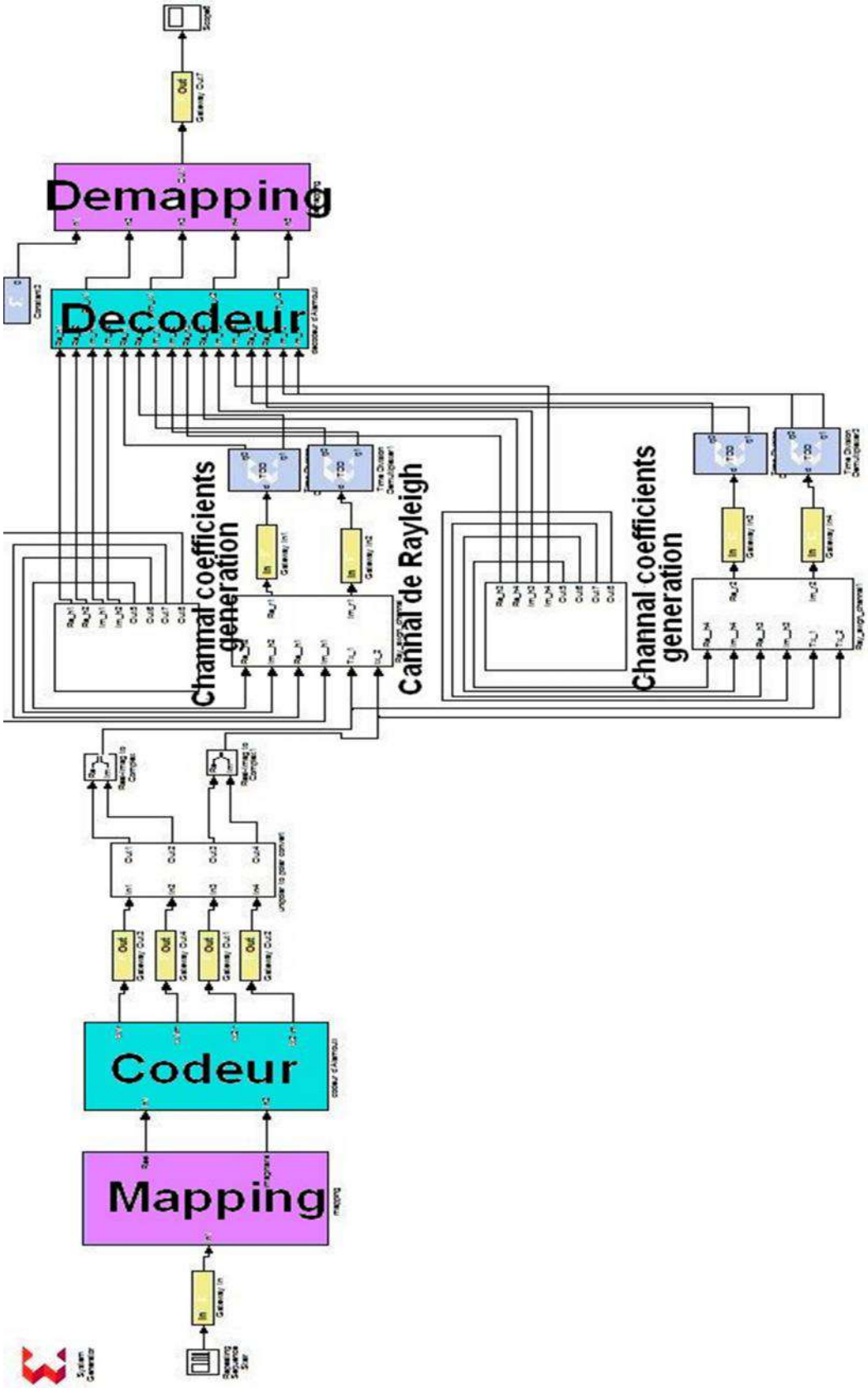
$$\mathrm{Re}\widetilde{x_1} = \mathrm{Re_h}_1 . Re_\mathrm{r}_1 + Im_h_1 . Im_r_1 + Re_\mathrm{h}_2 . Re_\mathrm{r}_2 + Im_h_2 . Im_r_2 + Re_\mathrm{h}_3 . Re_\mathrm{r}_3 + Im_h_3 . Im_r_3 + Re_\mathrm{h}_4 . Re_\mathrm{r}_4 + Im_h_4 . Im_r_4 \quad \text{(III.9)}$$

$$\mathrm{Im}\widetilde{x_1} = Re_h_1 . Im_\mathrm{r}_1 - Re_\mathrm{h}_2 . Im_r_2 - Im_h_1 . Re_\mathrm{r}_1 + Im_h_2 . Re_\mathrm{r}_2 + Re_\mathrm{h}_3 . Im_r_3 - Re_h_4 . Im_r_4 - Im_h_3 . Re_r_3 + Im_h_4 . Re_r_4 \quad \text{(III.10)}$$

$$\mathrm{Re}\widetilde{x_2} = Re_h_2 . Re_r_1 - Re_h_1 . Re_r_2 + Im_h_2 . Im_r_1 - Im_h_1 . Im_r_2 + Re_h_4 . Re_r_3 - Re_h_3 . Re_r_4 + Im_h_4 . Im_r_3 - Im_h_3 . Im_r_4 \quad \text{(III.11)}$$

$$\mathrm{Im}\widetilde{x_2} = Re_h_2 . Im_r_1 + Re_h_1 . Im_r_2 - Im_h_2 . Re_r_1 - Im_h_1 . Re_r_2 + Re_h_4 . Im_r_3 + Re_h_3 . Im_r_4 - Im_h_4 . Re_r_3 - Im_h_3 . Re_r_4 \quad \text{(III.12)}$$

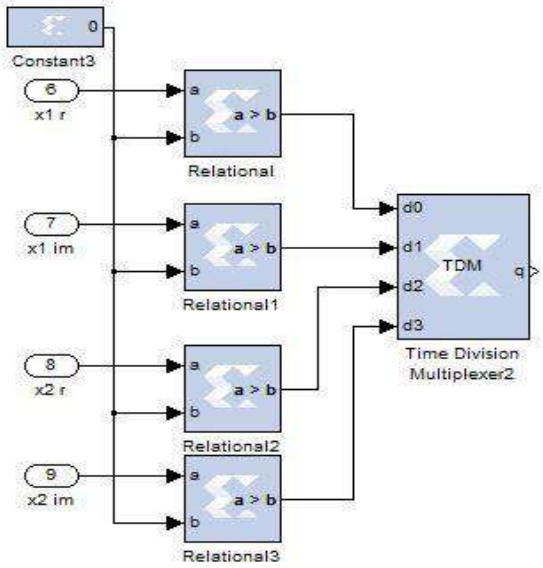
Pour réaliser ces équations on a utilisé un ensemble de 32 multiplieurs, 12 additionneurs et12 soustracteurs. Le schéma global de notre travail est le suivant :



FigureIII.17 : Schéma global d'un système de transmission MIMO.

III.4.2 Demapping QPSK :

La figure III.18 illustre l'étape du demapping qui transforme les signaux afin de restituer le train de bits binaire. Si le symbole est supérieur à 0 on le considère comme un 1 sinon c'est un 0. On a ajouté un multiplexeur temporel pour reconstruire la séquence final.



FigureIII.18 : Le demapping QPSK.

La simulation a donné :

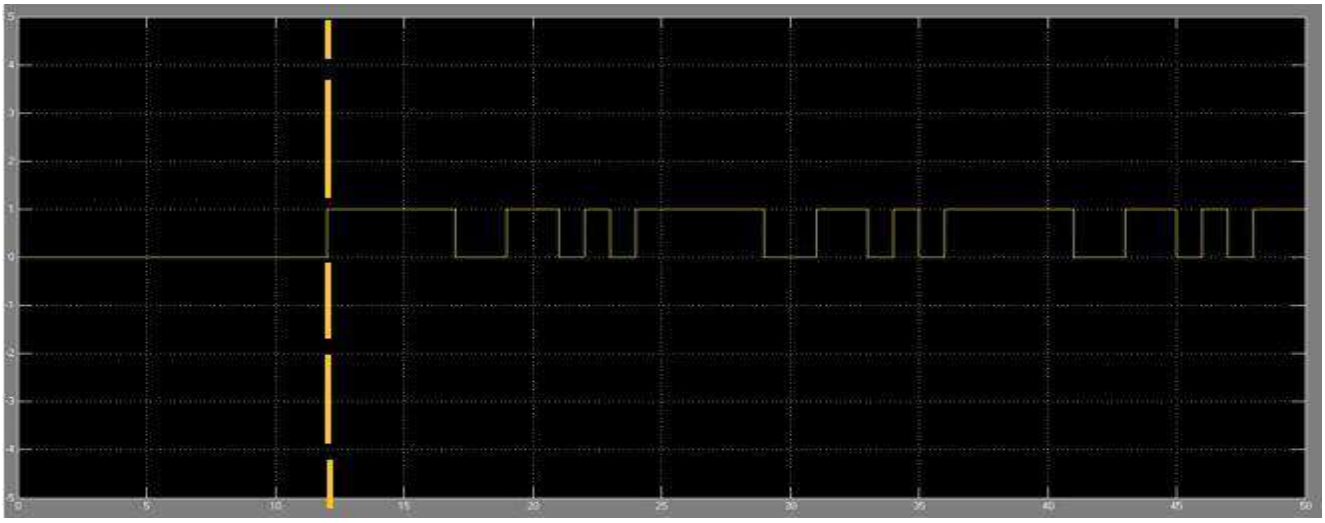
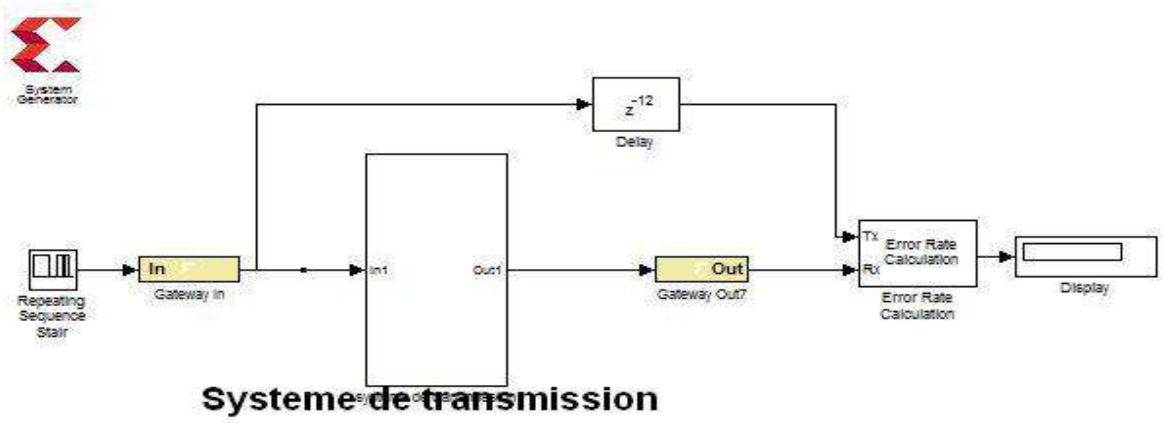


Figure III.19: La séquence reçue.

On remarque que la séquence obtenu est la même que la séquence envoyé mais avec douze période horloge de retard.

III.5 Mesure du BER :

Le BER (Bite Error Rate) est défini comme le rapport du nombre total de bits décodés erronés et nombre total de bits transmis. Dans les systèmes de communication numérique le BER apparaît en raison des erreurs de conception et la mise en œuvre ou d'autres sources extrinsèques d'erreur comme le bruit. Les simulations ont été effectuées dans deux cas le cas d'une antenne de réception et de deux antennes de réception. La figure suivante illustre le principe de mesure du BER:



FigureIII.20 : calcul de BER

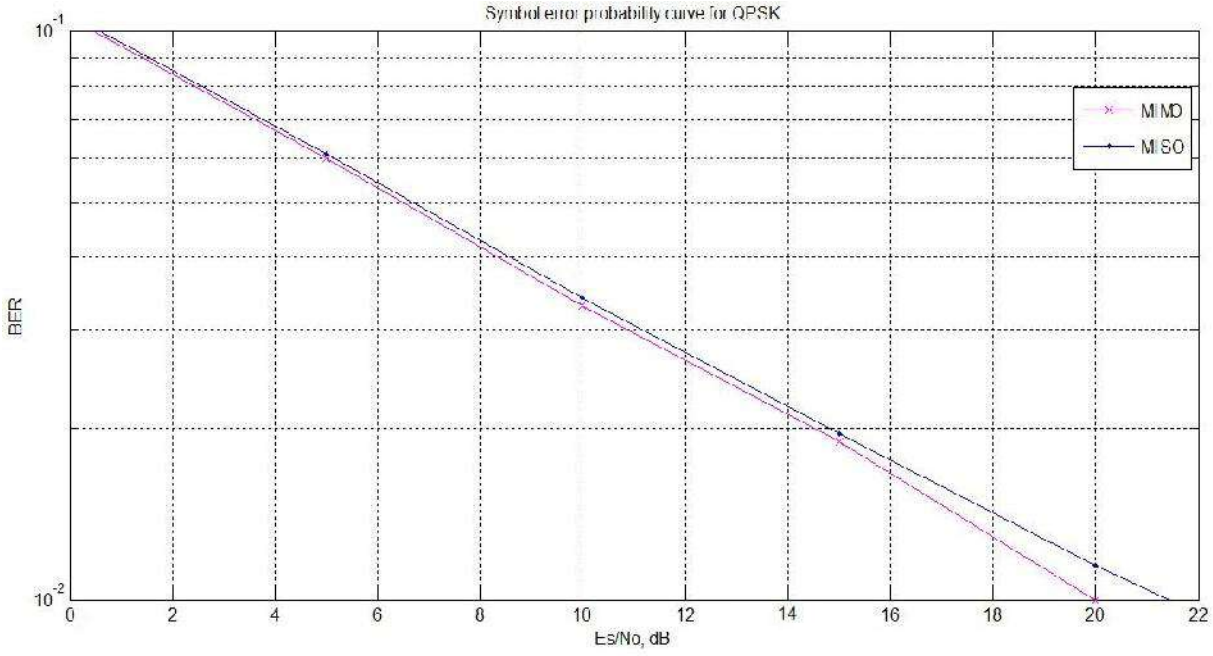


Figure III.21 : Le BER en fonction de $\frac{E_b}{N_0}$ du système conçu.

Nous avons analysé la modulation QPSK utilisant les systèmes MIMO. Le BER commence à diminuer avec l'augmentation de la valeur $\frac{E_b}{N_0}$. Il est très bien représenté dans le graphe précédant. Enfin, il est conclu que la performance du BER d'un système MIMO est meilleure qu'un système MISO.

III.6 Réduction des ressources utilisées (Etude de l'influence de nombres de bits) :

On a étudiés l'influence des nombre de bits utilisé dans le récepteur sur le BER et les ressources du circuit FPGA utilisées on utilisant le bloc **Resource estimator.** On a commencé par changer le nombre de bit des **Gateway in** on gardant les additionneurs et les soustracteurs et les multiplieurs en précision « full ».

Nombre de point (entier: fractionnel)	**8 :2**	**8 :4**	**8 :6**	**10 :4**	**10 :6**	**10 :8**	**12 :2**
S/B	20	20	20	20	20	20	20
Slices	321	321	321	391	391	391	461
FFs	71	71	71	83	83	83	95
LUTs	556	556	556	684	684	684	812
IOBs	55	55	55	67	67	67	79
Mults/DSP48s	16	16	16	16	16	16	16
BER	**0.0651**	**0.023**	**0.01642**	**0.023**	**0.03692**	**0.0153**	**0.0651**

Tableau III.1 : estimation des ressources

Pour un meilleur compromis entre ressources utilisées et BER obtenu nous prenons la précision 8 :6. Dans la suite nous avons examiné l'influence de la précision des multiplieurs sur le BER et les nombre ressources consommées.

Nombre de point (entier: fractionnel)	**16 :10**	**16 :12**	**14 :12**	**14 :10**	**12 :10**
S/B	20	20	20	20	20
Slices	299	321	289	289	257
FFs	31	71	71	39	39
LUTs	518	560	496	496	432
IOBs	55	55	55	55	55
Mults/DSP48s	16	16	16	16	16
BER	**0.0651**	**0.023**	**0.023**	**0.03692**	**0.0153**

Tableau III.2 : estimation des ressources

L'estimation des ressources nous a permis la réduction des ressources utilisé on gardant la même précision et rapidité .L'intérêt de cette étape avant la compilation est de voir si notre conception est réalisable sur FPGA.

III.7 Interface graphique de l'environnent ISE 12. 3 :

L'ISE contrôle tous les aspects d'un design flow. L'interface Project Navigator donne accès à toutes les ressources d'un projet et aux outils de l'implémentation. Elle procure aussi un accès aux fichiers et documents associés au projet (figure III.22).

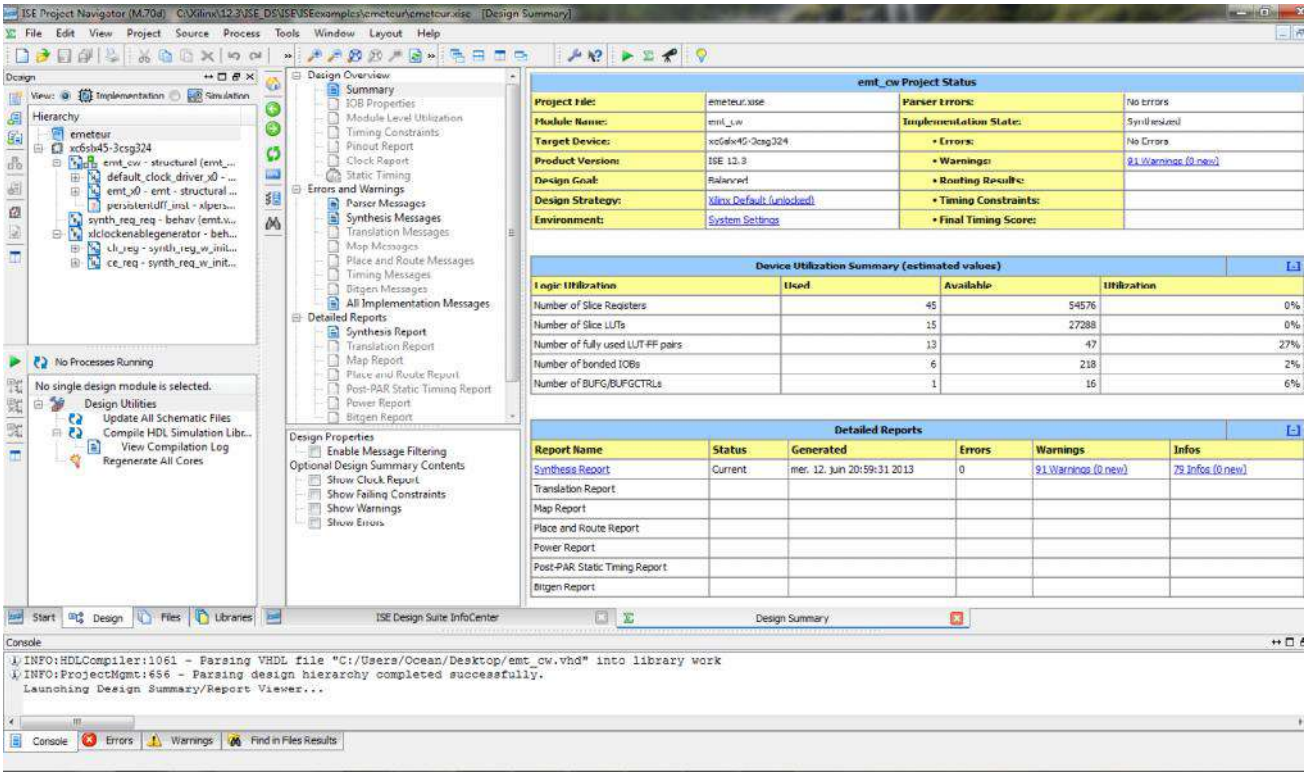


Figure III.22: Projet Navigator du logiciel ISE 12.3.

III.8 Synthèse et implémentation :

Le but de notre projet étant d'implanter un Codeur/Décodeur d'Alamouti sur un FPGA, nous avons travaillé avec le logiciel de développement XILINX ISE 12.3 et avec un circuit FPGA Spartan6 comme cible d'implémentation.

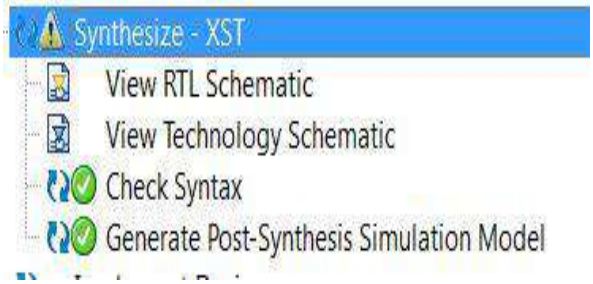
Pour accomplir notre projet on a suivi les étapes suivantes :

III.8.1Créer le projet :

Le logiciel system Generator permet de faire la conversion de la conception en un projet sous VHDL ou Verilog afin de faire la conception sous ISE.

III.8.2 Compilation (compile) :

C'est l'étape de vérification de la syntaxe et structure des codes VHDL.



FigureIII.23 : Compilation du projet

Les deux étapes précédente été accompli avec succès.

III.8.3 Synthèse et optimisation :

Il s'agit de réaliser le circuit RTL (Registre-Transfert level) avec des primitives technologiques (CLB, blocs RAM, I/O).

L'étape de la synthèse adonné le schéma logique suivant:

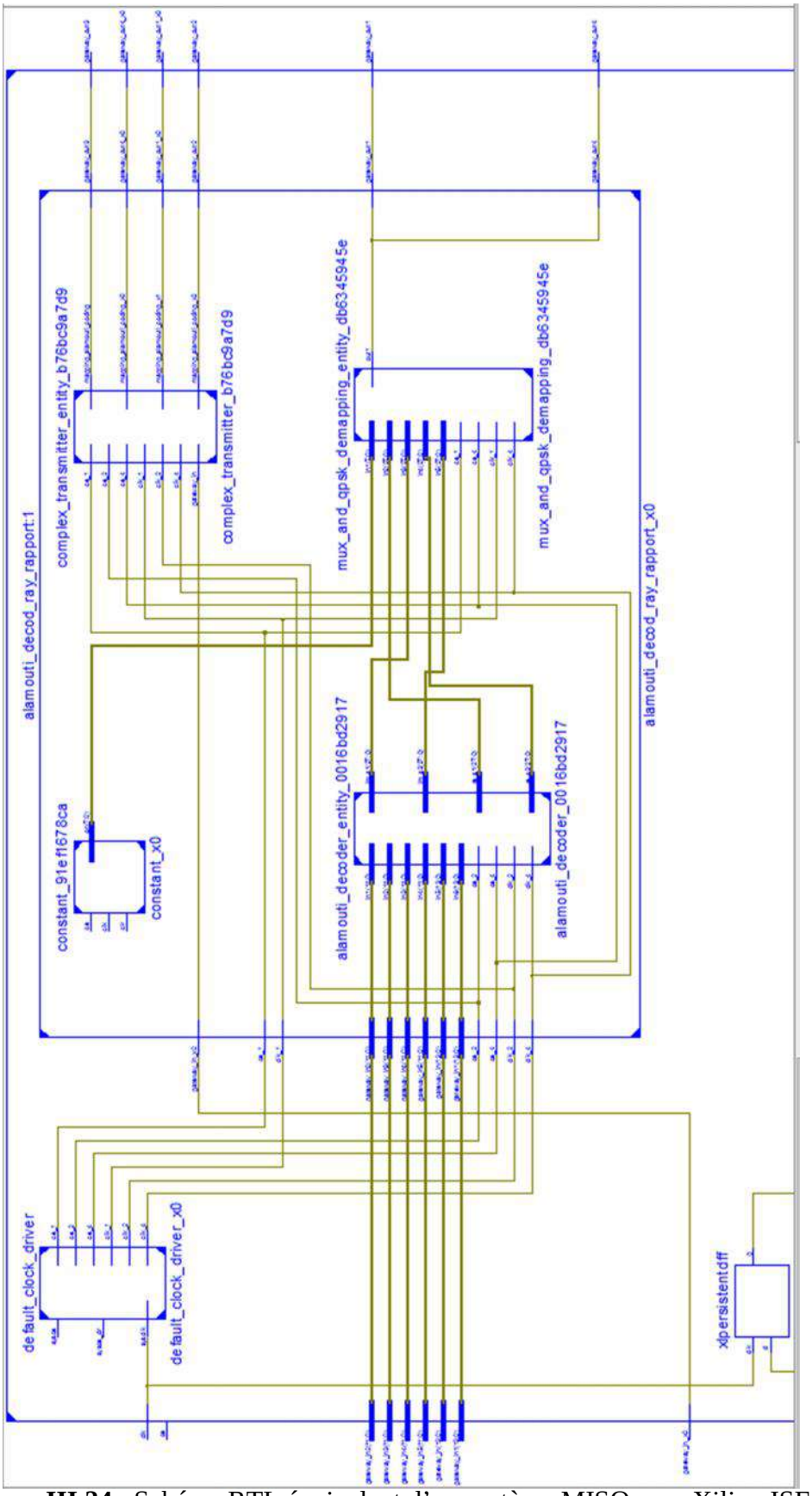


Figure III.24 : Schéma RTL équivalent d'un système MISO sous Xilinx ISE 12.3.

Utilisation élément logique	utilisé	Existant	pourcentage
Nombre de registres Slice	189	126800	0%
Nombre de LUT Slice	45	63400	0%
Nombre de paires LUT-FF pleinement utilisées	17	187	9%
Nombre de BUFG / BUFGCTRLs	1	32	3%
Nombres des entrées sorties	84	210	40%

Tableau III.3 : Rapport d'utilisation des ressources.

Pour mieux voire les circuits RTL, on a devisé le système on deux partie: la partie d'émission et la partie de réception.

Partie d'émission:

Utilisation élément logique	utilisé	Existant	pourcentage
Nombre de registres Slice	44	54576	0%
Nombre de LUT Slice	14	27288	0%
Nombre de paires LUT-FF pleinement utilisées	12	46	26%
Nombre de BUFG / BUFGCTRLs	1	16	6%
Nombres des entrées sorties	6	218	2%

Tableau III.4 : L'estimation des ressources de l'émetteur d'Alamouti.

La figure suivante illustre le fichier RTL Schématique de l'émetteur.

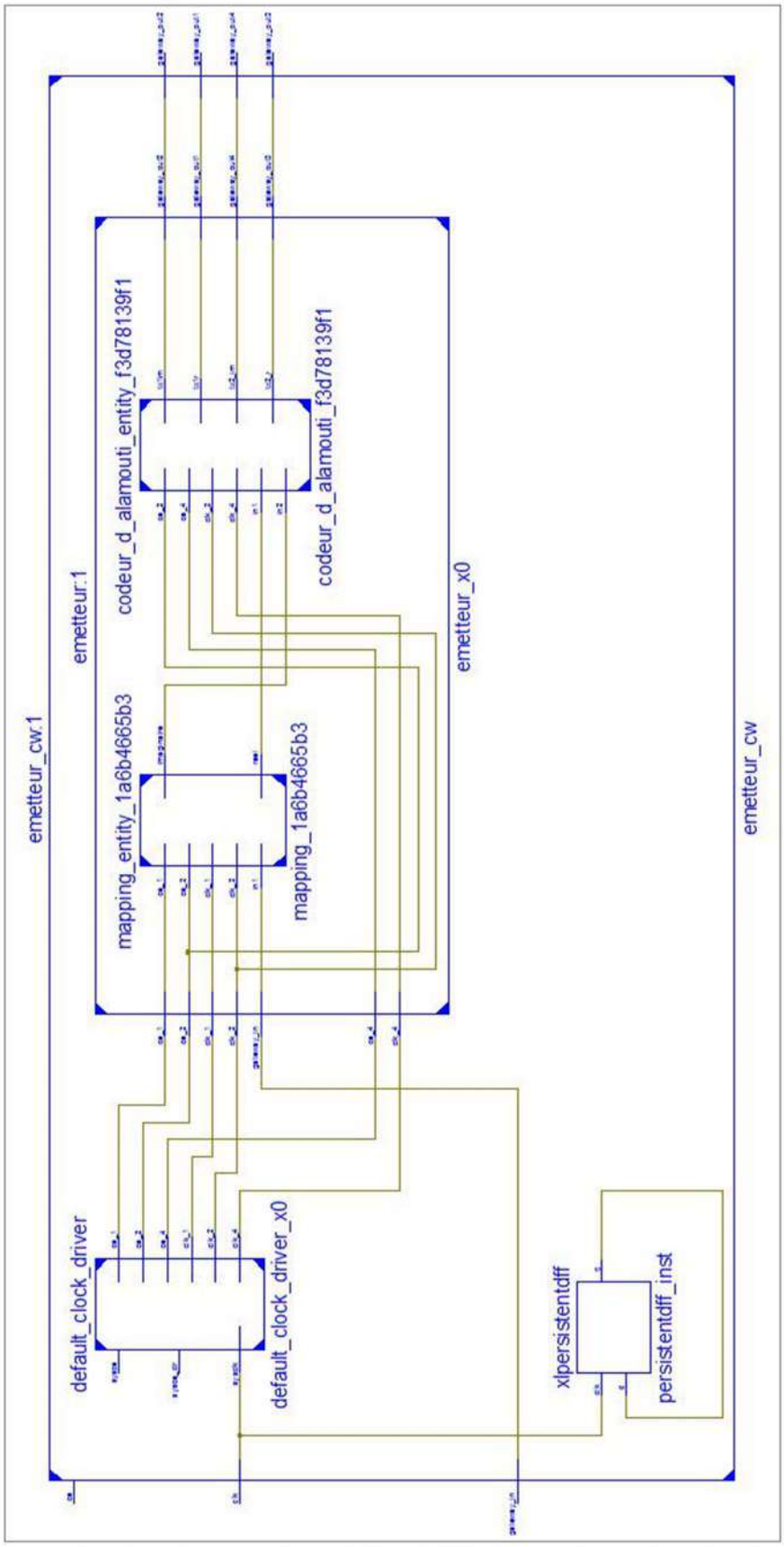


Figure III.25 : Vue d'ensemble du fichier RTL Schématique d'un émetteur d'Alamouti.

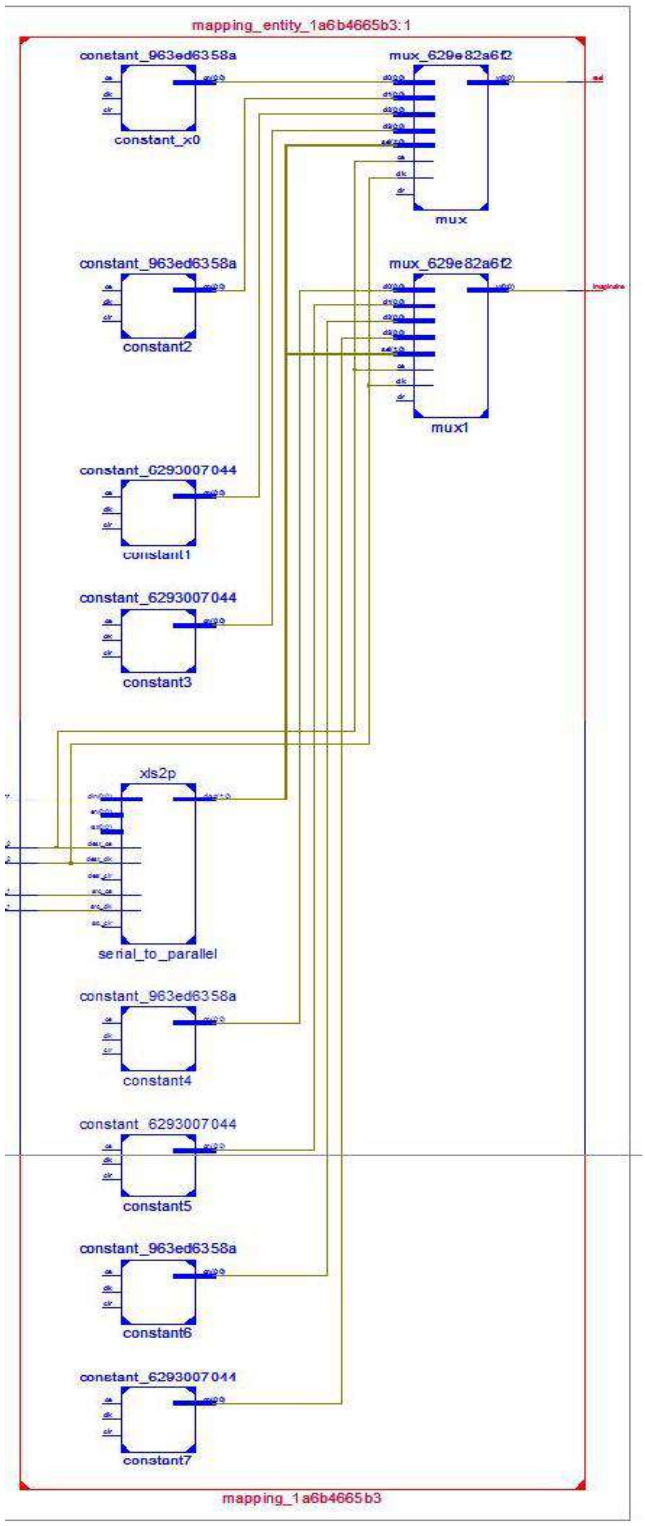


Figure III.26: Schéma RTL détaillé du mapping.

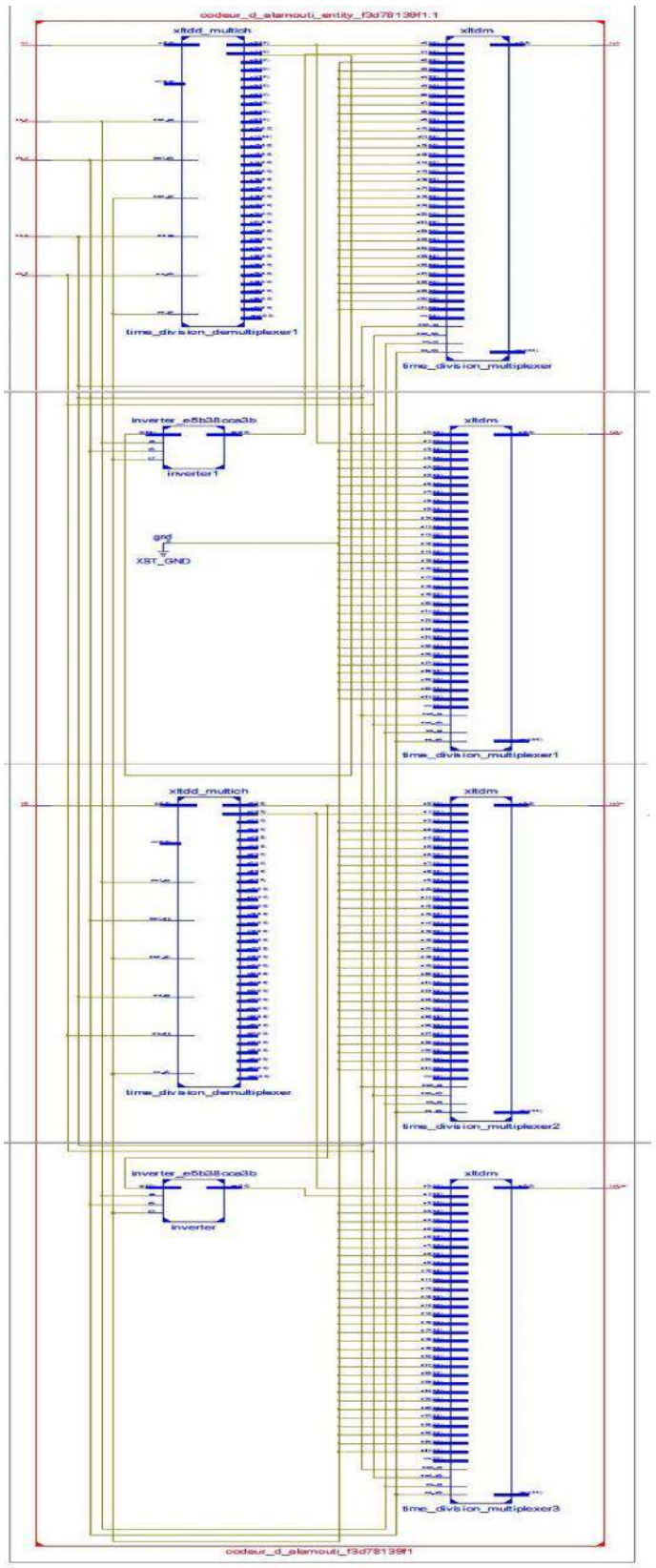


Figure III.27: Schéma RTL détaillé d'un codeur.

Partie de réception :

Utilisation élément logique	utilisé	Existant	pourcentage
Nombre de registres Slice	143	54576	0%
Nombre de LUT Slice	36	27288	0%
Nombre de paires LUT-FF pleinement utilisées	11	168	6%
Nombre de BUFG / BUFGCTRLs	1	16	6%
Nombres des entrées sorties	86	218	39%

Tableau III.5 : L'estimation des ressources d'un récepteur d'Alamouti.

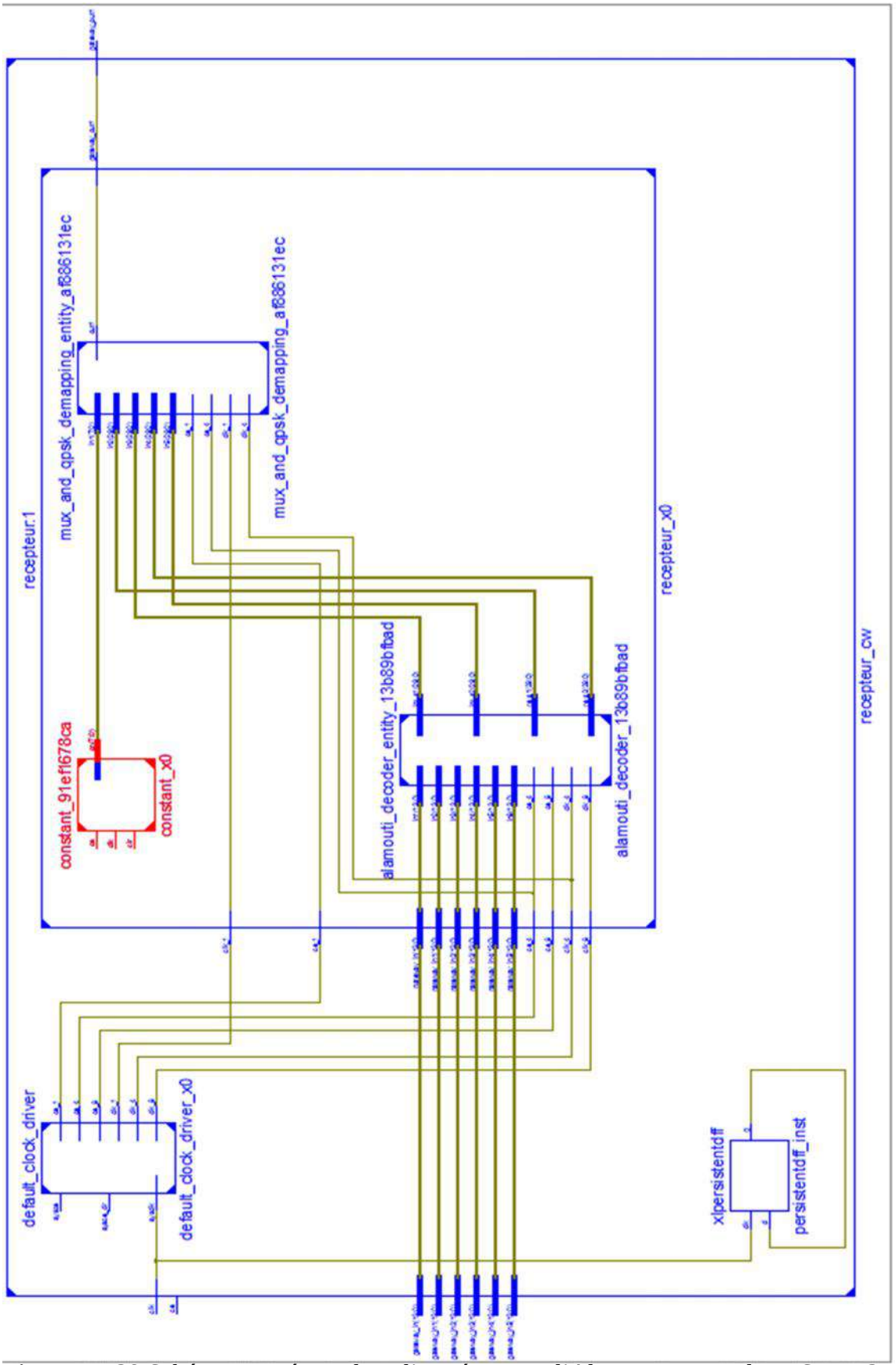


Figure III.28:Schéma RTL équivalent d'un récepteur d'Alamouti sous Xilinx ISE 12.3.

On voit bien que le récepteur est composé de trois blocs : bloc d'horloge, le décodeur d'Alamouti et le bloc de demapping.

Le décodeur d'Alamouti :

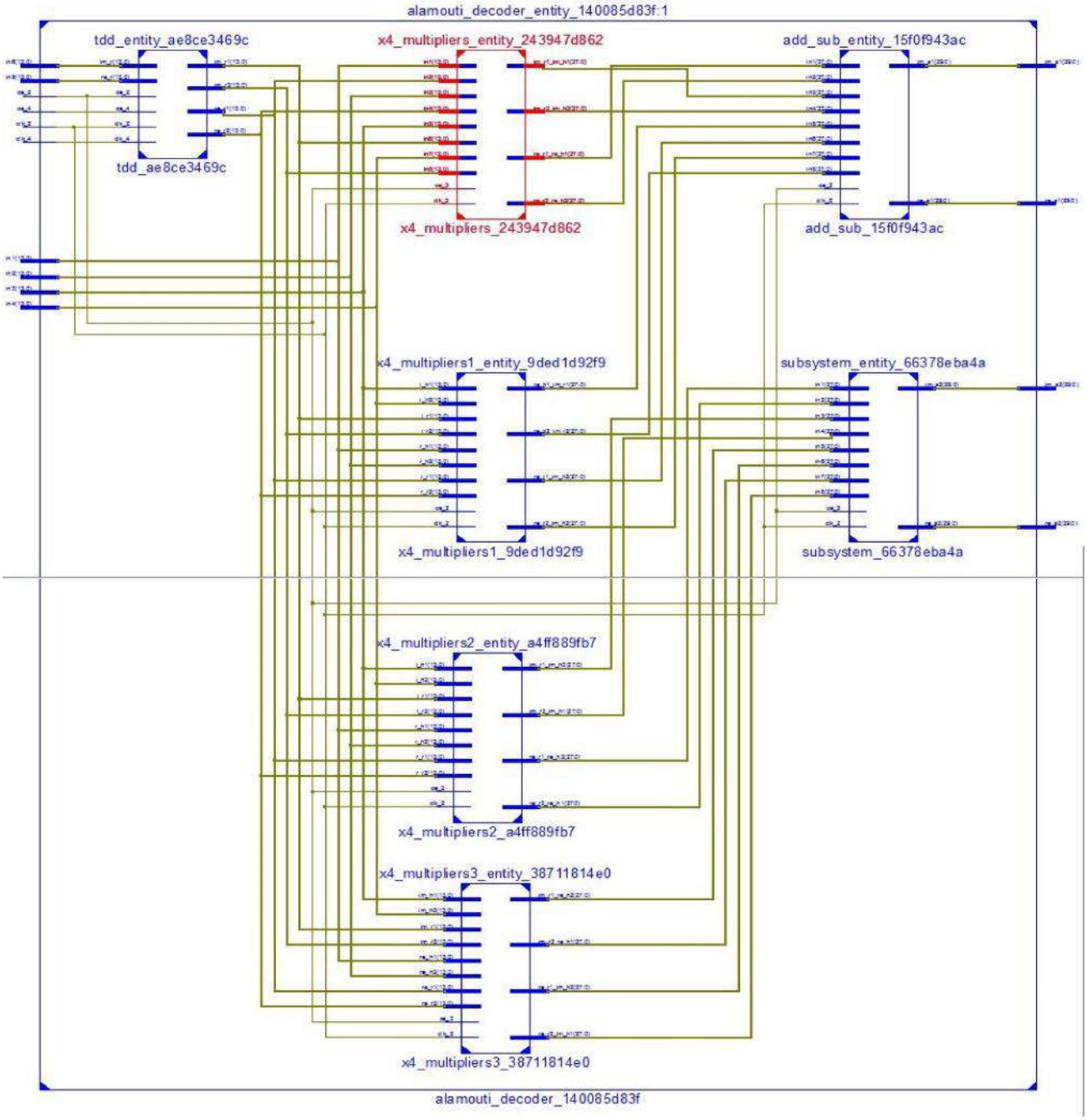


Figure III.29 : Schéma RTL équivalent d'un décodeur d'Alamouti.

Le demapping :

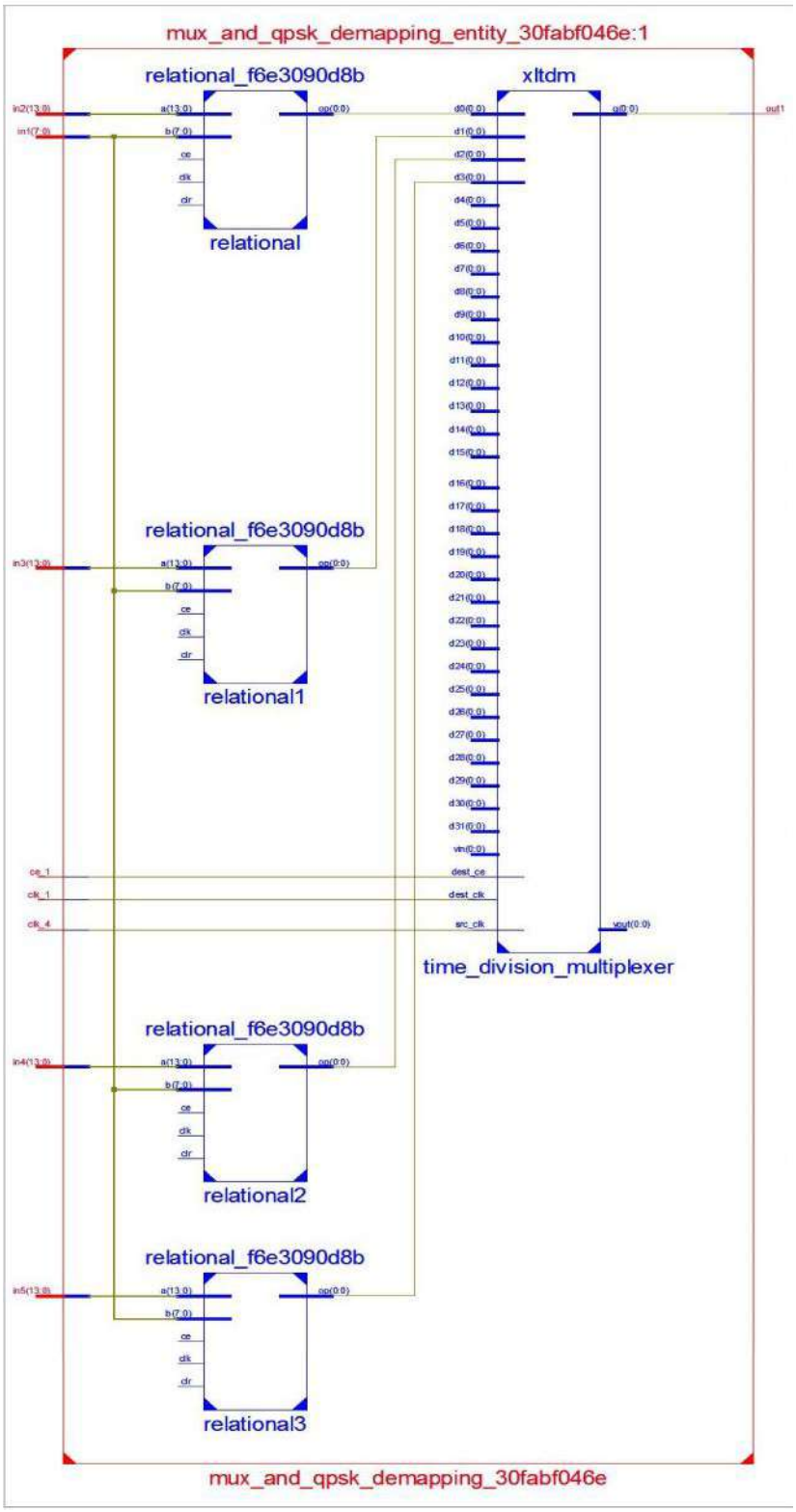


Figure III.30 : Schéma RTL équivalent du demapping.

III.8.4 Placement-routage :

Le rôle du placement-routage est double. Cibler les différentes primitives existantes dans le FPGA choisit. Et faire les différentes connexions entre les ces primitives.

III.8.5 Programmation :

Cette étape permet de générer le fichier de configuration appelé BITSTREAM et le charger sur le circuit FPGA.

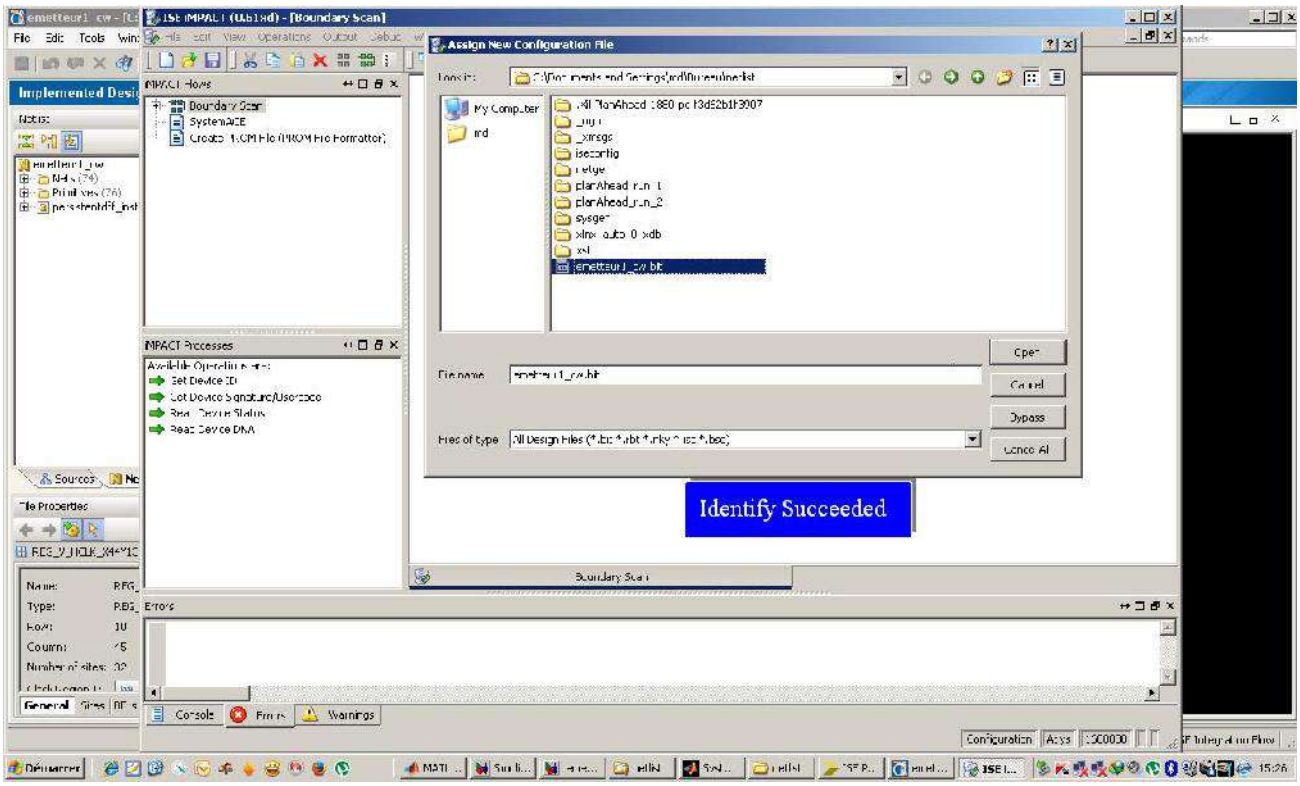


Figure III.31 : Génération du bit Stream et identification de la cible FPGA.

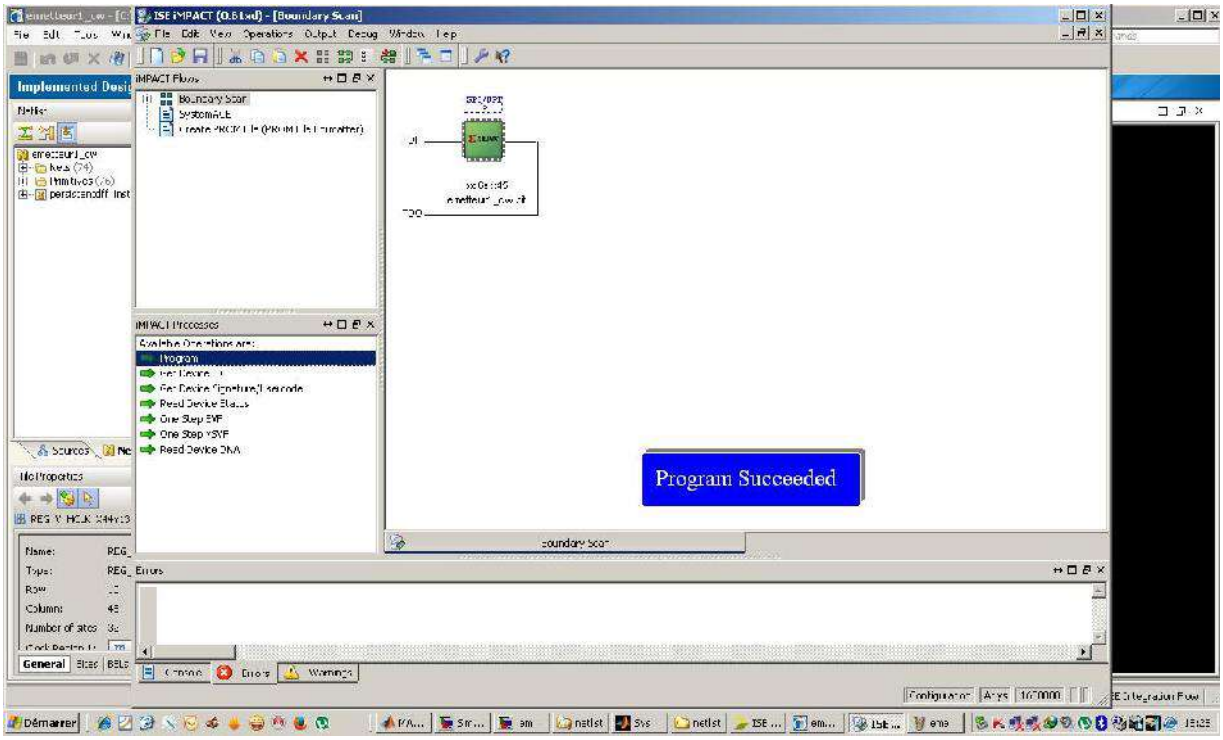


Figure III.32: Circuit implémenté sur Spartan 6

III.9 Conclusion :

Dans ce chapitre nous avons fait une conception d'un système de transmission et plus précisément d'un codeur/décodeur d'Alamouti pour deux modulations différentes en utilisant l'outil de conception « XILINX SYSTEM GENERATOR ».

Pour vérifier le bon fonctionnement de notre conception nous avons fait des simulations pour visualiser les formes des signaux dans les différents points de notre conception et nous avons comparé les signaux visualisés avec les calculs théoriques, la concordance des résultats nous a permis de valider notre conception.

Nous avons ensuite suivi cette étape par une étape d'optimisation permettant de réduire au minimum le nombre de blocs utilisés pour la conception et une étape de synthèse afin d'implémenter le circuit sur plateforme FPGA.

Conclusion générale

Conclusion général

Les travaux menés au cours de ce mémoire constituent l'ensemble des étapes indispensables pour l'implémentation d'un système de transmission sur un FPGA de façon générale et l'implémentation d'un codeur/décodeur d'Alamouti en particulier. Le bon fonctionnement du codeur/décodeur d'Alamouti MIMO et MISO est confirmé par la simulation numérique de la conception. L'implémentation physique des circuits est représentée par la vue interne établie par le logiciel ISE et qui donne un aperçu sur la représentation des portes logiques selon leurs implantations et routage dans le circuit FPGA.

Dans ce travail nous nous sommes intéressés à la modélisation, conception, simulation et implémentation d'un système de codage/décodage espace-temps en bloc proposé par Alamouti pour l'implémentation sur une plateforme de type FPGA. Ce codage permet une amélioration de la fiabilité et offre un gain de diversité permettant ainsi de combattre contre l'évanouissement de propagation en communication sans fil.

Nous avons commencé par la compréhension des notions fondamentales de la communication numérique : construction d'une chaine de transmission, les différents types de modulations numériques, canaux de transmissions, notions de diversité, codage/décodage espace-temps pour les MIMO, MISO. Nous avons vu que la communication numérique subie des évanouissements du signal dans un environnement de propagation multi-trajets, il est préférable donc d'utiliser des techniques de diversité (codage espace-temps). Nous avons vu aussi que le domaine des communications est caractérisé par une évolution très rapide en termes des exigences des utilisateurs en qualité de service, débit de transfert de données et nature de service offert. Cela conduit à la recherche des méthodes de conception et développement des systèmes de communication assurant à la foi la rapidité et la flexibilité.

Dans le deuxième chapitre nous avons donné un aperçu général sur les circuits FPGAs et leurs architecture et familles (Virtex, Spartan). Les systèmes de communication actuelle nécessitent des systèmes reconfigurable avec rapidité de mise en service pour cela nous nous somme intéressé particulièrement à la conception pour FPGA, ceci est dû à leur reconfigurabilité et facilité de mise en œuvre.

Dans le dernier chapitre nous avons fait une conception du décodeur d'Alamouti pour deux modulations différentes (BPSK et QPSK) en utilisant l'outil de conception XILINX SYSTEM GENERATOR et ISE pour l'implémentation de ce dernier. Pour vérifier le bon fonctionnement de notre conception nous avons fait des simulations pour visualiser les formes des signaux dans les différentes étapes : codage, canal et décodage.

La réalisation de ce travail nous a donné l'occasion d'acquérir des connaissances précieuses sur :

- Les systèmes de communication numérique en général et spécialement les systèmes MISO et MIMO
- Les canaux de transmission et leurs influences sur la chaine de transmission.

- Les circuits logiques FPGA.
- L'outil de développement Xilinx System Generator.
- L'outil d'implémentation ISE

Les étapes d'implémentation ont été décrites selon leurs ordres d'apparition dans le processus d'implémentation avec la présentation de l'outil logiciel (XILINX ISE 12.3) et les formats des fichiers de sorties. En final nous avons présenté le circuit implémenté sur FPGA.

Comme perspectives de notre travail nous pouvons envisager l'optimisation de la conception, spécialement la partie décodage où une architecture minimisant le nombre de multiplieurs est très souhaité pour une implémentation sur FPGA. Pour l'amélioration du BER du system ou l'augmentation du débit il est possible d'envisager d'autre schéma de codage/décodage espace-temps.

REFERENCES BIBLIOGRAPHIQUES

livre:

[1] Simon, Hakyin, « Communication *systems*», fourth *edition*, New York, Wiley, 1994,653 pages.

[3]Upamany, Madhow, *« Fundamentals of DIGITAL COMMUNICATION*», Cambridge, mars 2008,519 pages.

[4]B.P.Lathi, « *Modern digital and analog communication systems*», third edition, oxford university press, Inc, 1998,800 pages.

[5] Patrice, KADIONIK, *« Base de transmission numérique Les modulations numériques* », Bordeaux, ENSEIRB, 2000, 41 pages.

[6] Pascal, Djiknavorian, « *MIMO pour les nuls* », université Laval, Laboratoire de Radio télécommunications et de Traitement du Signal, Janvier 2006,137 pages.

[7] Maurice, Schiff « *Introduction to communication system simulation*», artech house Inc., January 31, 2006, 238 *pages.*

[14] S. Alamouti, « *A simple transmit diversity technique for wireless communications*», IEEE Journal on selected areas in communications, vol. 16, no. 8, pp. 1451–1458, 1998.

[19] NAVABI Zainalabedin, «*Digital Design and Implementation with Field programmable Devices*», Springer Science, 2005, 293 p.

Thèse ou mémoire :

[2] Zahra ZERDOUMI, 2006 « *Application des réseaux de neurones artificiels à la poursuite des non linéarités des systèmes fluctuantes des systèmes satellitaires*», Th. Dr : électronique : UNIVERSITE DE OUARGLA

[8] Basel RIHAWI, 2008 « *Analyse et réduction du Power Ratio des systèmes de radiocommunications multi-antennes* » Th. Dr : Traitement du signal et télécommunications : l'UNIVERSITE DE RENNES I

[9] Emmanuel AMADOR, 2008. « *Synthèse de canaux 'de Rice et de Rayleigh en chambre de réverbération* », M.Sc:génie électrique : FACULTE DES SCIENCES ET DE GENIE UNIVERSITE LAVAL.

[10] Nuttapol PRAYONGPUN, le 29 avril 2009. « *Modélisation et étude de la capacité du canal pour un système multi-antennes avancé exploitant la diversité de polarisation* », Th. Dr : Signal, image, parole et télécoms : UNIVERSITE JOSEPH FOURIER

[11] Olivier BERDER, décembre 2002 « *Optimisation et stratégies d'allocation de puissance des systèmes de transmission multi-antennes* »Th. Dr : Informatique et réseaux : l'Université de Bretagne Occidentale.

[13] Ghaya REKAYA-BEN OTHMAN, 17 décembre 2004. « *Nouvelles constructions algébriques de codes spatio-temporels atteignant le compromis multiplexage-diversité* », Th Dr : Electronique et communications télécom : l'Ecole Nationale Supérieure des Télécommunications paris.

[16] TEKOBON Jerry, 2007 « *FPGA and Traffic Network Analysis* », mémoire : Ingénieur : Génie Electrique et Télécommunication : Ecole Nationale Supérieure Polytechnique de Yaounde Cameroun.

[18] NAOUAR Mohamed Wissem, le 6 décembre 2007 « *Commande numérique à base de composant FPGA d'une machine synchrone Algorithme de contrôle de courant* ». Mémoire : Ingénieur : génie électrique. Tunis.

[20] WASSI Guy, 2005. « *Mise en œuvre de l'auto-reconfiguration partielle et dynamique sur FPGA Xilinx Virtex-II* ». Mémoire : Master : informatique industrielle et systèmes automatisés : Université Pierre et Marie Curie (Paris VI Jussieu).

Article :

[12] Didier Le Ruyet et Berna Ozbek, Juin 2007 « *Systèmes MIMO et codage spatio-temporel* ».

[15] GUELLAL Amar, 21 juin 2012 « *Bulletin des Energies Renouvelables* », CDER, N° 24.

[22] national instrument, 1avr 2012. « *Les cinq principaux atouts de la technologie FPGA* »

Page internet :

[17] « Introduction au microprocesseur et circuits assimilés les composants FPGA » consultée le 18 mars 2013
<URL: http://michel.hubin.pagesperso-orange.fr/physique/microp/chap_mp5.htm>

[21] www.xilinx.com.

[23] Réseau Logique Programmable (FPGA) Spartan 6 Digilent
<URL: http://www.robotshop.com/reseau-logique-programmable-fpga-spartan-6digilent.html.>

Annexe

-- system generator version 13.2 vhdl source file.
--
-- copyright(c) 2011 by xilinx, inc. all rights reserved. this
-- text/file contains proprietary, confidential information of xilinx,
-- inc., is distributed under license from xilinx, inc., and may be used,
-- copied and/or disclosed only pursuant to the terms of a valid license
-- agreement with xilinx, inc. xilinx hereby grants you a license to use
-- this text/file solely for design, simulation, implementation and
-- creation of design files limited to xilinx devices or technologies.
-- use with non-xilinx devices or technologies is expressly prohibited
-- and immediately terminates your license unless covered by a separate
-- agreement.
--
-- xilinx is providing this design, code, or information "as is" solely
-- for use in developing programs and solutions for xilinx devices. by
-- providing this design, code, or information as one possible
-- implementation of this feature, application or standard, xilinx is
-- making no representation that this implementation is free from any
-- claims of infringement. you are responsible for obtaining any rights
-- you may require for your implementation. xilinx expressly disclaims
-- any warranty whatsoever with respect to the adequacy of the
-- implementation, including but not limited to warranties of
-- merchantability or fitness for a particular purpose.
--
-- xilinx products are not intended for use in life support appliances,
-- devices, or systems. use in such applications is expressly prohibited.
--

-- any modifications that are made to the source code are done at the user's
-- sole risk and will be unsupported.
--
-- this copyright and support notice must be retained as part of this
-- text at all times. (c) copyright 1995-2011 xilinx, inc. all rights
-- reserved.

-- system generator version 13.2 vhdl source file.
--
-- copyright(c) 2011 by xilinx, inc. all rights reserved. this
-- text/file contains proprietary, confidential information of xilinx,
-- inc., is distributed under license from xilinx, inc., and may be used,
-- copied and/or disclosed only pursuant to the terms of a valid license
-- agreement with xilinx, inc. xilinx hereby grants you a license to use
-- this text/file solely for design, simulation, implementation and
-- creation of design files limited to xilinx devices or technologies.
-- use with non-xilinx devices or technologies is expressly prohibited
-- and immediately terminates your license unless covered by a separate
-- agreement.
--
-- xilinx is providing this design, code, or information "as is" solely
-- for use in developing programs and solutions for xilinx devices. by
-- providing this design, code, or information as one possible
-- implementation of this feature, application or standard, xilinx is
-- making no representation that this implementation is free from any
-- claims of infringement. you are responsible for obtaining any rights
-- you may require for your implementation. xilinx expressly disclaims

```
-- any warranty whatsoever with respect to the adequacy of the
-- implementation, including but not limited to warranties of
-- merchantability or fitness for a particular purpose.
--
-- xilinx products are not intended for use in life support appliances,
-- devices, or systems.  use in such applications is expressly prohibited.
--
-- any modifications that are made to the source code are done at the user's
-- sole risk and will be unsupported.
--
-- this copyright and support notice must be retained as part of this
-- text at all times.  (c) copyright 1995-2011 xilinx, inc.  all rights
-- reserved.
----------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.conv_pkg.all;
-- synopsys translate_off
library unisim;
use unisim.vcomponents.all;
-- synopsys translate_on
entity xlclockdriver is
  generic (
    period: integer := 2;
    log_2_period: integer := 0;
    pipeline_regs: integer := 5;
    use_bufg: integer := 0
  );
```

```
  port (
    sysclk: in std_logic;
    sysclr: in std_logic;
    sysce: in std_logic;
    clk: out std_logic;
    clr: out std_logic;
    ce: out std_logic;
    ce_logic: out std_logic
  );
end xlclockdriver;
architecture behavior of xlclockdriver is
  component bufg
    port (
      i: in std_logic;
      o: out std_logic
    );
  end component;
  component synth_reg_w_init
    generic (
      width: integer;
      init_index: integer;
      init_value: bit_vector;
      latency: integer
    );
    port (
      i: in std_logic_vector(width - 1 downto 0);
      ce: in std_logic;
      clr: in std_logic;
      clk: in std_logic;
```

```
    o: out std_logic_vector(width - 1 downto 0)
  );
end component;
function size_of_uint(inp: integer; power_of_2: boolean)
  return integer
is
  constant inp_vec: std_logic_vector(31 downto 0) :=
    integer_to_std_logic_vector(inp,32, xlunsigned);
  variable result: integer;
begin
  result := 32;
  for i in 0 to 31 loop
    if inp_vec(i) = '1' then
      result := i;
    end if;
  end loop;
  if power_of_2 then
    return result;
  else
    return result+1;
  end if;
end;
function is_power_of_2(inp: std_logic_vector)
  return boolean
is
  constant width: integer := inp'length;
  variable vec: std_logic_vector(width - 1 downto 0);
  variable single_bit_set: boolean;
  variable more_than_one_bit_set: boolean;
```

```
  variable result: boolean;
 begin
  vec := inp;
  single_bit_set := false;
  more_than_one_bit_set := false;
  -- synopsys translate_off
  if (is_xoru(vec)) then
    return false;
  end if;
   -- synopsys translate_on
  if width > 0 then
    for i in 0 to width - 1 loop
      if vec(i) = '1' then
        if single_bit_set then
          more_than_one_bit_set := true;
        end if;
        single_bit_set := true;
      end if;
    end loop;
  end if;
  if (single_bit_set and not(more_than_one_bit_set)) then
    result := true;
  else
    result := false;
  end if;
  return result;
 end;
 function ce_reg_init_val(index, period : integer)
  return integer
```

```
is
  variable result: integer;
 begin
   result := 0;
   if ((index mod period) = 0) then
      result := 1;
   end if;
   return result;
end;
function remaining_pipe_regs(num_pipeline_regs, period : integer)
 return integer
is
  variable factor, result: integer;
begin
   factor := (num_pipeline_regs / period);
   result := num_pipeline_regs - (period * factor) + 1;
   return result;
end;

function sg_min(l, r: integer) return integer is
begin
   if l < r then
        return l;
   else
        return r;
   end if;
end;
constant max_pipeline_regs : integer := 8;
constant pipe_regs : integer := 5;
```

```
  constant num_pipeline_regs : integer := sg_min(pipeline_regs, max_pipeline_regs);
  constant rem_pipeline_regs : integer := remaining_pipe_regs(num_pipeline_regs,period);
  constant period_floor: integer := max(2, period);
  constant power_of_2_counter: boolean :=
    is_power_of_2(integer_to_std_logic_vector(period_floor,32, xlunsigned));
  constant cnt_width: integer :=
    size_of_uint(period_floor, power_of_2_counter);
  constant clk_for_ce_pulse_minus1: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector((period_floor - 2),cnt_width, xlunsigned);
  constant clk_for_ce_pulse_minus2: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - 3),cnt_width, xlunsigned);
  constant clk_for_ce_pulse_minus_regs: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - rem_pipeline_regs),cnt_width, xlunsigned);
  signal clk_num: unsigned(cnt_width - 1 downto 0) := (others => '0');
  signal ce_vec : std_logic_vector(num_pipeline_regs downto 0);
  attribute max_fanout : string;
  attribute max_fanout of ce_vec:signal is "reduce";
  signal ce_vec_logic : std_logic_vector(num_pipeline_regs downto 0);
  attribute max_fanout of ce_vec_logic:signal is "reduce";
  signal internal_ce: std_logic_vector(0 downto 0);
  signal internal_ce_logic: std_logic_vector(0 downto 0);
  signal cnt_clr, cnt_clr_dly: std_logic_vector (0 downto 0);
begin
  clk <= sysclk;
  clr <= sysclr;
  cntr_gen: process(sysclk)
  begin
    if sysclk'event and sysclk = '1'  then
      if (sysce = '1') then
```

```
    if ((cnt_clr_dly(0) = '1') or (sysclr = '1')) then
      clk_num <= (others => '0');
    else
      clk_num <= clk_num + 1;
    end if;
  end if;
  end if;
end process;
clr_gen: process(clk_num, sysclr)
begin
  if power_of_2_counter then
    cnt_clr(0) <= sysclr;
  else
    if (unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus1
        or sysclr = '1') then
      cnt_clr(0) <= '1';
    else
      cnt_clr(0) <= '0';
    end if;
  end if;
end process;
clr_reg: synth_reg_w_init
  generic map (
    width => 1,
    init_index => 0,
    init_value => b"0000",
    latency => 1
  )
  port map (
```

```
    i => cnt_clr,
    ce => sysce,
    clr => sysclr,
    clk => sysclk,
    o => cnt_clr_dly
  );
pipelined_ce : if period > 1 generate
    ce_gen: process(clk_num)
    begin
      if unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus_regs then
        ce_vec(num_pipeline_regs) <= '1';
      else
        ce_vec(num_pipeline_regs) <= '0';
      end if;
    end process;
    ce_pipeline: for index in num_pipeline_regs downto 1 generate
      ce_reg : synth_reg_w_init
        generic map (
          width => 1,
          init_index => ce_reg_init_val(index, period),
          init_value => b"0000",
          latency => 1
          )
        port map (
          i => ce_vec(index downto index),
          ce => sysce,
          clr => sysclr,
          clk => sysclk,
          o => ce_vec(index-1 downto index-1)
```

```
            );
    end generate;
    internal_ce <= ce_vec(0 downto 0);
end generate;
pipelined_ce_logic: if period > 1 generate
    ce_gen_logic: process(clk_num)
    begin
      if unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus_regs then
        ce_vec_logic(num_pipeline_regs) <= '1';
      else
        ce_vec_logic(num_pipeline_regs) <= '0';
      end if;
    end process;
    ce_logic_pipeline: for index in num_pipeline_regs downto 1 generate
      ce_logic_reg : synth_reg_w_init
        generic map (
          width => 1,
          init_index => ce_reg_init_val(index, period),
          init_value => b"0000",
          latency => 1
          )
        port map (
          i => ce_vec_logic(index downto index),
          ce => sysce,
          clr => sysclr,
          clk => sysclk,
          o => ce_vec_logic(index-1 downto index-1)
          );
    end generate;
```

```
    internal_ce_logic <= ce_vec_logic(0 downto 0);
  end generate;
  use_bufg_true: if period > 1 and use_bufg = 1 generate
   ce_bufg_inst: bufg
    port map (
     i => internal_ce(0),
     o => ce
    );
   ce_bufg_inst_logic: bufg
    port map (
     i => internal_ce_logic(0),
     o => ce_logic
    );
  end generate;
  use_bufg_false: if period > 1 and (use_bufg = 0) generate
   ce <= internal_ce(0);
   ce_logic <= internal_ce_logic(0);
  end generate;
  generate_system_clk: if period = 1 generate
   ce <= sysce;
   ce_logic <= sysce;
  end generate;
end architecture behavior;
library ieee;
use ieee.std_logic_1164.all;
use work.conv_pkg.all;

entity default_clock_driver is
  port (
```

```
    sysce: in std_logic;
    sysce_clr: in std_logic;
    sysclk: in std_logic;
    ce_1: out std_logic;
    ce_2: out std_logic;
    ce_4: out std_logic;
    clk_1: out std_logic;
    clk_2: out std_logic;
    clk_4: out std_logic
  );
end default_clock_driver;

architecture structural of default_clock_driver is
  attribute syn_noprune: boolean;
  attribute syn_noprune of structural : architecture is true;
  attribute optimize_primitives: boolean;
  attribute optimize_primitives of structural : architecture is false;
  attribute dont_touch: boolean;
  attribute dont_touch of structural : architecture is true;

  signal sysce_clr_x0: std_logic;
  signal sysce_x0: std_logic;
  signal sysclk_x0: std_logic;
  signal xlclockdriver_1_ce: std_logic;
  signal xlclockdriver_1_clk: std_logic;
  signal xlclockdriver_2_ce: std_logic;
  signal xlclockdriver_2_clk: std_logic;
  signal xlclockdriver_4_ce: std_logic;
  signal xlclockdriver_4_clk: std_logic;
```

```
begin
 sysce_x0 <= sysce;
 sysce_clr_x0 <= sysce_clr;
 sysclk_x0 <= sysclk;
 ce_1 <= xlclockdriver_1_ce;
 ce_2 <= xlclockdriver_2_ce;
 ce_4 <= xlclockdriver_4_ce;
 clk_1 <= xlclockdriver_1_clk;
 clk_2 <= xlclockdriver_2_clk;
 clk_4 <= xlclockdriver_4_clk;

 xlclockdriver_1: entity work.xlclockdriver
  generic map (
    log_2_period => 1,
    period => 1,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_1_ce,
    clk => xlclockdriver_1_clk
  );

 xlclockdriver_2: entity work.xlclockdriver
  generic map (
    log_2_period => 2,
```

```
    period => 2,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_2_ce,
    clk => xlclockdriver_2_clk
  );

 xlclockdriver_4: entity work.xlclockdriver
  generic map (
    log_2_period => 3,
    period => 4,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_4_ce,
    clk => xlclockdriver_4_clk
  );

end structural;
library ieee;
use ieee.std_logic_1164.all;
use work.conv_pkg.all;
```

```
entity emetteur1_cw is
  port (
    ce: in std_logic := '1';
    clk: in std_logic; -- clock period = 10.0 ns (100.0 mhz)
    gateway_in: in std_logic;
    gateway_out1: out std_logic;
    gateway_out2: out std_logic;
    gateway_out3: out std_logic;
    gateway_out4: out std_logic
  );
end emetteur1_cw;

architecture structural of emetteur1_cw is
  component xlpersistentdff
    port (
      clk: in std_logic;
      d: in std_logic;
      q: out std_logic
    );
  end component;
  attribute syn_black_box: boolean;
  attribute syn_black_box of xlpersistentdff: component is true;
  attribute box_type: string;
  attribute box_type of xlpersistentdff: component is "black_box";
  attribute syn_noprune: boolean;
  attribute optimize_primitives: boolean;
  attribute dont_touch: boolean;
  attribute syn_noprune of xlpersistentdff: component is true;
```

```
  attribute optimize_primitives of xlpersistentdff: component is false;
  attribute dont_touch of xlpersistentdff: component is true;

  signal ce_1_sg_x1: std_logic;
  attribute max_fanout: string;
  attribute max_fanout of ce_1_sg_x1: signal is "reduce";
  signal ce_2_sg_x2: std_logic;
  attribute max_fanout of ce_2_sg_x2: signal is "reduce";
  signal ce_4_sg_x1: std_logic;
  attribute max_fanout of ce_4_sg_x1: signal is "reduce";
  signal clknet: std_logic;
  signal clk_1_sg_x1: std_logic;
  signal clk_2_sg_x2: std_logic;
  signal clk_4_sg_x1: std_logic;
  signal gateway_in_net: std_logic;
  signal gateway_out1_net: std_logic;
  signal gateway_out2_net: std_logic;
  signal gateway_out3_net: std_logic;
  signal gateway_out4_net: std_logic;
  signal persistentdff_inst_q: std_logic;
  attribute syn_keep: boolean;
  attribute syn_keep of persistentdff_inst_q: signal is true;
  attribute keep: boolean;
  attribute keep of persistentdff_inst_q: signal is true;
  attribute preserve_signal: boolean;
  attribute preserve_signal of persistentdff_inst_q: signal is true;

begin
  clknet <= clk;
```

```
gateway_in_net <= gateway_in;
gateway_out1 <= gateway_out1_net;
gateway_out2 <= gateway_out2_net;
gateway_out3 <= gateway_out3_net;
gateway_out4 <= gateway_out4_net;

default_clock_driver_x0: entity work.default_clock_driver
  port map (
    sysce => '1',
    sysce_clr => '0',
    sysclk => clknet,
    ce_1 => ce_1_sg_x1,
    ce_2 => ce_2_sg_x2,
    ce_4 => ce_4_sg_x1,
    clk_1 => clk_1_sg_x1,
    clk_2 => clk_2_sg_x2,
    clk_4 => clk_4_sg_x1
  );
emetteur1_x0: entity work.emetteur1
  port map (
    ce_1 => ce_1_sg_x1,
    ce_2 => ce_2_sg_x2,
    ce_4 => ce_4_sg_x1,
    clk_1 => clk_1_sg_x1,
    clk_2 => clk_2_sg_x2,
    clk_4 => clk_4_sg_x1,
    gateway_in => gateway_in_net,
    gateway_out1 => gateway_out1_net,
    gateway_out2 => gateway_out2_net,
```

```
    gateway_out3 => gateway_out3_net,
    gateway_out4 => gateway_out4_net
  );
 persistentdff_inst: xlpersistentdff
  port map (
    clk => clknet,
    d => persistentdff_inst_q,
    q => persistentdff_inst_q
  );
end structural;
```

```
---------------------------------------------------------------------
-- System Generator version 13.2 VHDL source file.
--
-- Copyright(C) 2011 by Xilinx, Inc.  All rights reserved.  This
-- text/file contains proprietary, confidential information of Xilinx,
-- Inc., is distributed under license from Xilinx, Inc., and may be used,
-- copied and/or disclosed only pursuant to the terms of a valid license
-- agreement with Xilinx, Inc.  Xilinx hereby grants you a license to use
-- this text/file solely for design, simulation, implementation and
-- creation of design files limited to Xilinx devices or technologies.
-- Use with non-Xilinx devices or technologies is expressly prohibited
-- and immediately terminates your license unless covered by a separate
-- agreement.
--
-- Xilinx is providing this design, code, or information "as is" solely
-- for use in developing programs and solutions for Xilinx devices.  By
-- providing this design, code, or information as one possible
-- implementation of this feature, application or standard, Xilinx is
-- making no representation that this implementation is free from any
-- claims of infringement.  You are responsible for obtaining any rights
-- you may require for your implementation.  Xilinx expressly disclaims
-- any warranty whatsoever with respect to the adequacy of the
-- implementation, including but not limited to warranties of
-- merchantability or fitness for a particular purpose.
--
-- Xilinx products are not intended for use in life support appliances,
-- devices, or systems.  Use in such applications is expressly prohibited.
--
```

-- Any modifications that are made to the source code are done at the user's
-- sole risk and will be unsupported.
--
-- This copyright and support notice must be retained as part of this
-- text at all times. (c) Copyright 1995-2011 Xilinx, Inc. All rights
-- reserved.

-- System Generator version 13.2 VHDL source file.

--
-- Copyright(C) 2011 by Xilinx, Inc. All rights reserved. This
-- text/file contains proprietary, confidential information of Xilinx,
-- Inc., is distributed under license from Xilinx, Inc., and may be used,
-- copied and/or disclosed only pursuant to the terms of a valid license
-- agreement with Xilinx, Inc. Xilinx hereby grants you a license to use
-- this text/file solely for design, simulation, implementation and
-- creation of design files limited to Xilinx devices or technologies.
-- Use with non-Xilinx devices or technologies is expressly prohibited
-- and immediately terminates your license unless covered by a separate
-- agreement.
--
-- Xilinx is providing this design, code, or information "as is" solely
-- for use in developing programs and solutions for Xilinx devices. By
-- providing this design, code, or information as one possible
-- implementation of this feature, application or standard, Xilinx is
-- making no representation that this implementation is free from any
-- claims of infringement. You are responsible for obtaining any rights
-- you may require for your implementation. Xilinx expressly disclaims

```
-- any warranty whatsoever with respect to the adequacy of the
-- implementation, including but not limited to warranties of
-- merchantability or fitness for a particular purpose.
--
-- Xilinx products are not intended for use in life support appliances,
-- devices, or systems.  Use in such applications is expressly prohibited.
--
-- Any modifications that are made to the source code are done at the user's
-- sole risk and will be unsupported.
--
-- This copyright and support notice must be retained as part of this
-- text at all times.  (c) Copyright 1995-2011 Xilinx, Inc.  All rights
-- reserved.
----------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use work.conv_pkg.all;
-- synopsys translate_off
library unisim;
use unisim.vcomponents.all;
-- synopsys translate_on
entity xlclockdriver is
  generic (
    period: integer := 2;
    log_2_period: integer := 0;
    pipeline_regs: integer := 5;
    use_bufg: integer := 0
  );
```

```
  port (
    sysclk: in std_logic;
    sysclr: in std_logic;
    sysce: in std_logic;
    clk: out std_logic;
    clr: out std_logic;
    ce: out std_logic;
    ce_logic: out std_logic
  );
end xlclockdriver;
architecture behavior of xlclockdriver is
  component bufg
    port (
      i: in std_logic;
      o: out std_logic
    );
  end component;
  component synth_reg_w_init
    generic (
      width: integer;
      init_index: integer;
      init_value: bit_vector;
      latency: integer
    );
    port (
      i: in std_logic_vector(width - 1 downto 0);
      ce: in std_logic;
      clr: in std_logic;
      clk: in std_logic;
```

```
    o: out std_logic_vector(width - 1 downto 0)
  );
end component;
function size_of_uint(inp: integer; power_of_2: boolean)
  return integer
is
  constant inp_vec: std_logic_vector(31 downto 0) :=
    integer_to_std_logic_vector(inp,32, xlUnsigned);
  variable result: integer;
begin
  result := 32;
  for i in 0 to 31 loop
    if inp_vec(i) = '1' then
      result := i;
    end if;
  end loop;
  if power_of_2 then
    return result;
  else
    return result+1;
  end if;
end;
function is_power_of_2(inp: std_logic_vector)
  return boolean
is
  constant width: integer := inp'length;
  variable vec: std_logic_vector(width - 1 downto 0);
  variable single_bit_set: boolean;
  variable more_than_one_bit_set: boolean;
```

```
  variable result: boolean;
 begin
  vec := inp;
  single_bit_set := false;
  more_than_one_bit_set := false;
  -- synopsys translate_off
  if (is_XorU(vec)) then
    return false;
  end if;
   -- synopsys translate_on
  if width > 0 then
    for i in 0 to width - 1 loop
      if vec(i) = '1' then
        if single_bit_set then
          more_than_one_bit_set := true;
        end if;
        single_bit_set := true;
      end if;
    end loop;
  end if;
  if (single_bit_set and not(more_than_one_bit_set)) then
    result := true;
  else
    result := false;
  end if;
  return result;
 end;
 function ce_reg_init_val(index, period : integer)
  return integer
```

```
is
  variable result: integer;
 begin
   result := 0;
   if ((index mod period) = 0) then
      result := 1;
   end if;
   return result;
end;
function remaining_pipe_regs(num_pipeline_regs, period : integer)
 return integer
is
  variable factor, result: integer;
begin
   factor := (num_pipeline_regs / period);
   result := num_pipeline_regs - (period * factor) + 1;
   return result;
end;

function sg_min(L, R: INTEGER) return INTEGER is
begin
   if L < R then
        return L;
   else
        return R;
   end if;
end;
constant max_pipeline_regs : integer := 8;
constant pipe_regs : integer := 5;
```

```
  constant num_pipeline_regs : integer := sg_min(pipeline_regs, max_pipeline_regs);
  constant rem_pipeline_regs : integer := remaining_pipe_regs(num_pipeline_regs,period);
  constant period_floor: integer := max(2, period);
  constant power_of_2_counter: boolean :=
    is_power_of_2(integer_to_std_logic_vector(period_floor,32, xlUnsigned));
  constant cnt_width: integer :=
    size_of_uint(period_floor, power_of_2_counter);
  constant clk_for_ce_pulse_minus1: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector((period_floor - 2),cnt_width, xlUnsigned);
  constant clk_for_ce_pulse_minus2: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - 3),cnt_width, xlUnsigned);
  constant clk_for_ce_pulse_minus_regs: std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - rem_pipeline_regs),cnt_width, xlUnsigned);
  signal clk_num: unsigned(cnt_width - 1 downto 0) := (others => '0');
  signal ce_vec : std_logic_vector(num_pipeline_regs downto 0);
  attribute MAX_FANOUT : string;
  attribute MAX_FANOUT of ce_vec:signal is "REDUCE";
  signal ce_vec_logic : std_logic_vector(num_pipeline_regs downto 0);
  attribute MAX_FANOUT of ce_vec_logic:signal is "REDUCE";
  signal internal_ce: std_logic_vector(0 downto 0);
  signal internal_ce_logic: std_logic_vector(0 downto 0);
  signal cnt_clr, cnt_clr_dly: std_logic_vector (0 downto 0);
begin
  clk <= sysclk;
  clr <= sysclr;
  cntr_gen: process(sysclk)
  begin
    if sysclk'event and sysclk = '1'  then
      if (sysce = '1') then
```

```
    if ((cnt_clr_dly(0) = '1') or (sysclr = '1')) then
      clk_num <= (others => '0');
    else
      clk_num <= clk_num + 1;
    end if;
  end if;
  end if;
end process;
clr_gen: process(clk_num, sysclr)
begin
  if power_of_2_counter then
    cnt_clr(0) <= sysclr;
  else
    if (unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus1
       or sysclr = '1') then
      cnt_clr(0) <= '1';
    else
      cnt_clr(0) <= '0';
    end if;
  end if;
end process;
clr_reg: synth_reg_w_init
  generic map (
    width => 1,
    init_index => 0,
    init_value => b"0000",
    latency => 1
  )
  port map (
```

```
    i => cnt_clr,
    ce => sysce,
    clr => sysclr,
    clk => sysclk,
    o => cnt_clr_dly
  );
pipelined_ce : if period > 1 generate
    ce_gen: process(clk_num)
    begin
      if unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus_regs then
        ce_vec(num_pipeline_regs) <= '1';
      else
        ce_vec(num_pipeline_regs) <= '0';
      end if;
    end process;
    ce_pipeline: for index in num_pipeline_regs downto 1 generate
      ce_reg : synth_reg_w_init
        generic map (
          width => 1,
          init_index => ce_reg_init_val(index, period),
          init_value => b"0000",
          latency => 1
          )
        port map (
          i => ce_vec(index downto index),
          ce => sysce,
          clr => sysclr,
          clk => sysclk,
          o => ce_vec(index-1 downto index-1)
```

```
            );
    end generate;
    internal_ce <= ce_vec(0 downto 0);
end generate;
pipelined_ce_logic: if period > 1 generate
    ce_gen_logic: process(clk_num)
    begin
      if unsigned_to_std_logic_vector(clk_num) = clk_for_ce_pulse_minus_regs then
        ce_vec_logic(num_pipeline_regs) <= '1';
      else
        ce_vec_logic(num_pipeline_regs) <= '0';
      end if;
    end process;
    ce_logic_pipeline: for index in num_pipeline_regs downto 1 generate
      ce_logic_reg : synth_reg_w_init
        generic map (
          width => 1,
          init_index => ce_reg_init_val(index, period),
          init_value => b"0000",
          latency => 1
          )
        port map (
          i => ce_vec_logic(index downto index),
          ce => sysce,
          clr => sysclr,
          clk => sysclk,
          o => ce_vec_logic(index-1 downto index-1)
          );
    end generate;
```

```
    internal_ce_logic <= ce_vec_logic(0 downto 0);
  end generate;
  use_bufg_true: if period > 1 and use_bufg = 1 generate
   ce_bufg_inst: bufg
    port map (
     i => internal_ce(0),
     o => ce
    );
   ce_bufg_inst_logic: bufg
    port map (
     i => internal_ce_logic(0),
     o => ce_logic
    );
  end generate;
  use_bufg_false: if period > 1 and (use_bufg = 0) generate
   ce <= internal_ce(0);
   ce_logic <= internal_ce_logic(0);
  end generate;
  generate_system_clk: if period = 1 generate
   ce <= sysce;
   ce_logic <= sysce;
  end generate;
end architecture behavior;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;

entity default_clock_driver is
  port (
```

```
    sysce: in std_logic;
    sysce_clr: in std_logic;
    sysclk: in std_logic;
    ce_1: out std_logic;
    ce_4: out std_logic;
    ce_8: out std_logic;
    clk_1: out std_logic;
    clk_4: out std_logic;
    clk_8: out std_logic
  );
end default_clock_driver;

architecture structural of default_clock_driver is
  attribute syn_noprune: boolean;
  attribute syn_noprune of structural : architecture is true;
  attribute optimize_primitives: boolean;
  attribute optimize_primitives of structural : architecture is false;
  attribute dont_touch: boolean;
  attribute dont_touch of structural : architecture is true;

  signal sysce_clr_x0: std_logic;
  signal sysce_x0: std_logic;
  signal sysclk_x0: std_logic;
  signal xlclockdriver_1_ce: std_logic;
  signal xlclockdriver_1_clk: std_logic;
  signal xlclockdriver_4_ce: std_logic;
  signal xlclockdriver_4_clk: std_logic;
  signal xlclockdriver_8_ce: std_logic;
  signal xlclockdriver_8_clk: std_logic;
```

```
begin
 sysce_x0 <= sysce;
 sysce_clr_x0 <= sysce_clr;
 sysclk_x0 <= sysclk;
 ce_1 <= xlclockdriver_1_ce;
 ce_4 <= xlclockdriver_4_ce;
 ce_8 <= xlclockdriver_8_ce;
 clk_1 <= xlclockdriver_1_clk;
 clk_4 <= xlclockdriver_4_clk;
 clk_8 <= xlclockdriver_8_clk;

 xlclockdriver_1: entity work.xlclockdriver
  generic map (
    log_2_period => 1,
    period => 1,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_1_ce,
    clk => xlclockdriver_1_clk
  );

 xlclockdriver_4: entity work.xlclockdriver
  generic map (
    log_2_period => 3,
```

```
    period => 4,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_4_ce,
    clk => xlclockdriver_4_clk
  );

 xlclockdriver_8: entity work.xlclockdriver
  generic map (
    log_2_period => 4,
    period => 8,
    use_bufg => 0
  )
  port map (
    sysce => sysce_x0,
    sysclk => sysclk_x0,
    sysclr => sysce_clr_x0,
    ce => xlclockdriver_8_ce,
    clk => xlclockdriver_8_clk
  );

end structural;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;
```

```
entity recepteur1_cw is
  port (
    ce: in std_logic := '1';
    clk: in std_logic; -- clock period = 40.0 ns (25.0 Mhz)
    gateway_in: in std_logic_vector(13 downto 0);
    gateway_in1: in std_logic_vector(13 downto 0);
    gateway_in2: in std_logic_vector(13 downto 0);
    gateway_in3: in std_logic_vector(13 downto 0);
    gateway_in4: in std_logic_vector(13 downto 0);
    gateway_in5: in std_logic_vector(13 downto 0);
    gateway_out1: out std_logic
  );
end recepteur1_cw;

architecture structural of recepteur1_cw is
  component xlpersistentdff
    port (
      clk: in std_logic;
      d: in std_logic;
      q: out std_logic
    );
  end component;
  attribute syn_black_box: boolean;
  attribute syn_black_box of xlpersistentdff: component is true;
  attribute box_type: string;
  attribute box_type of xlpersistentdff: component is "black_box";
  attribute syn_noprune: boolean;
  attribute optimize_primitives: boolean;
```

```
attribute dont_touch: boolean;
attribute syn_noprune of xlpersistentdff: component is true;
attribute optimize_primitives of xlpersistentdff: component is false;
attribute dont_touch of xlpersistentdff: component is true;

signal ce_1_sg_x1: std_logic;
attribute MAX_FANOUT: string;
attribute MAX_FANOUT of ce_1_sg_x1: signal is "REDUCE";
signal ce_4_sg_x9: std_logic;
attribute MAX_FANOUT of ce_4_sg_x9: signal is "REDUCE";
signal ce_8_sg_x2: std_logic;
attribute MAX_FANOUT of ce_8_sg_x2: signal is "REDUCE";
signal clkNet: std_logic;
signal clk_1_sg_x1: std_logic;
signal clk_4_sg_x9: std_logic;
signal clk_8_sg_x2: std_logic;
signal gateway_in1_net: std_logic_vector(13 downto 0);
signal gateway_in2_net: std_logic_vector(13 downto 0);
signal gateway_in3_net: std_logic_vector(13 downto 0);
signal gateway_in4_net: std_logic_vector(13 downto 0);
signal gateway_in5_net: std_logic_vector(13 downto 0);
signal gateway_in_net: std_logic_vector(13 downto 0);
signal gateway_out1_net: std_logic;
signal persistentdff_inst_q: std_logic;
attribute syn_keep: boolean;
attribute syn_keep of persistentdff_inst_q: signal is true;
attribute keep: boolean;
attribute keep of persistentdff_inst_q: signal is true;
attribute preserve_signal: boolean;
```

```
 attribute preserve_signal of persistentdff_inst_q: signal is true;

begin
 clkNet <= clk;
 gateway_in_net <= gateway_in;
 gateway_in1_net <= gateway_in1;
 gateway_in2_net <= gateway_in2;
 gateway_in3_net <= gateway_in3;
 gateway_in4_net <= gateway_in4;
 gateway_in5_net <= gateway_in5;
 gateway_out1 <= gateway_out1_net;

 default_clock_driver_x0: entity work.default_clock_driver
  port map (
   sysce => '1',
   sysce_clr => '0',
   sysclk => clkNet,
   ce_1 => ce_1_sg_x1,
   ce_4 => ce_4_sg_x9,
   ce_8 => ce_8_sg_x2,
   clk_1 => clk_1_sg_x1,
   clk_4 => clk_4_sg_x9,
   clk_8 => clk_8_sg_x2
  );

 persistentdff_inst: xlpersistentdff
  port map (
   clk => clkNet,
   d => persistentdff_inst_q,
```

```
    q => persistentdff_inst_q
  );

 recepteur1_x0: entity work.recepteur1
  port map (
    ce_1 => ce_1_sg_x1,
    ce_4 => ce_4_sg_x9,
    ce_8 => ce_8_sg_x2,
    clk_1 => clk_1_sg_x1,
    clk_4 => clk_4_sg_x9,
    clk_8 => clk_8_sg_x2,
    gateway_in => gateway_in_net,
    gateway_in1 => gateway_in1_net,
    gateway_in2 => gateway_in2_net,
    gateway_in3 => gateway_in3_net,
    gateway_in4 => gateway_in4_net,
    gateway_in5 => gateway_in5_net,
    gateway_out1 => gateway_out1_net
  );

end structural;
```