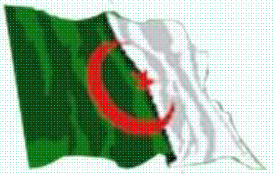
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université SAAD DEHLEB BLIDA
Faculté des Sciences de l'Ingénieur
Département : Aéronautique



Mémoire de fin d'études
En vue d'obtention
Du diplôme d'ingénieur d'état en Aéronautique
Spécialité : Navigation Aérienne
Option : Installation

THEME

IMPLEMENTATION FPGA DE DETECTEURS CFAR POUR L'ACQUISITION D'UN SIGNAL GALILEO

Présenté par :

Mlle: DEHOUCHE SIHAM

Proposé et suivi par :

M^{r}: BENACHENHOU. K

ANNEE UNIVERSITAIRE 2008 /2009

Résumé

Ce travail consiste à étudier et simuler l'opération d'acquisition pour le code CA du signal GPS et le futur signal GALILEO en utilisant le logiciel Matlab. Le bloc d'acquisition se compose d'un étage de recherche et d'un étage de détection, à ce niveau, nous proposons plusieurs structures CFAR (Constant False Alarm Rate) (Cell Averaging (CA), Greatest Of (GO), and Smallest Of (SO)), nous les étudions à la présence de bruit et de multitrajets dans un milieu urbain. Nous proposons aussi les performances des détecteurs CFAR en termes de probabilité de détection. Les résultats obtenus pourront être utilisés pour le système GPS tout comme pour GALILEO. Les structures CFAR simulés sur le Matlab seront aussi réalisées sur FPGA(Field Programable Gates Array) en utilisant le langage VHDL.

Abstract

In this work, the acquisition algorithms are studied and analyzed with help of Matlab simulations for GPS C/A code and Binary offset carrier BOC modulated GALILEO signals. The acquisition blocs consists of search stage and detection stage, at this level different Constant False Alarm Rate (CFAR) detectors (Cell Averaging (CA), Greatest Of (GO), and Smallest Of (SO)), are proposed and studied in the presence of noise and strong urban multipath interference. The detection performance of the CFAR is studied and analyzed. Also is studied the detection probability and performance of these different types of CFAR detectors. The obtained results for these CFAR detectors can be used in both, GPS or GALILEO receivers in case of indoor navigation. The CFAR detectors will be realized on FPGA circuits using the VHDL language.

ملخص

هذا العمل يتضمن دراسة و محاكاة عملية اكتساب الشفرة "CA" لإشارة "GPS" و لإشارة المستقبل "GALILEO" باستعمال برنامج « Matlab ». الاكتساب يتكون من طبقة البحث و طبقة الكشف, في هذا المستوى, نقترح عدَّة هياكل « CFAR » (CA, GO et SO) , ندرسهم في حضور التشويش و متعدد المسارات في وسط مدني. نقترح كذلك إمكانيات كواشف « CFAR »من ناحية احتمالات الكشف. النتائج المحصل عليها يمكن استعمالها في "GPS" كما يمكن ذلك في"GALILEO".

الهياكل « CFAR » التي تمت محاكاتها على« Matlab » سيتم إنجازها على« FPGA » باستعمال برنامج « VHDL ».

Remerciements

Sans sa volonté, son aide et son soutien ce travail ne serait sûrement pas réalisé, dieu merci ;

Je remercie profondément mes chers parents pour leur soutien sans cesse, leur amour et leur compréhension durant toute ma vie et spécialement pour réaliser ce travail ;

Merci à Amine ; mon fiancé pour sa présence continue, ses encouragements, son aide et son soutien ;

Je n'oublierais pas ma chère belle mère pour son soutien et son aide ;

Je tiens à exprimer ma sincère gratitude et mes remerciements pour mon promoteur Mr. BENACHNOU pour ses efforts, ses conseils et son aide considérable ;

Je n'oublierais pas mes enseignants durant mon cursus d'études, spécialement Mr.HELLAL, et Mr. MENGUELLATI ;

Je remercie tous mes amis et collègues qui m'ont aidé à leur manière à réaliser ce travail, remerciement spécial pour mes collègues : OUALI Nordine, RAHMOUNI Lyes sans oublier mon collègue du département de l'électronique Samir KHDEDJI pour son aide.

Sommaire

RESUMEE

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

INTRODUCTION GENERALE……. 1

Chapitre I -Généralités sur le système Galileo-

I.1.Introduction …….... 2

I .2. Description du système Galiléo ……. … 2

I .2.1.Le segment spatial …… 3

I .2. 2. Le segment de contrôle …….. 4

I .2. 3. Le segment utilisateur ……. 5

I.3. Les services offerts par le système Galiléo……...….. 5

I.4. Détermination de la position et précision attendue……...……. 7

I.5. Les signaux Galiléo…….. 8

I.5.1. Particularité des signaux Galiléo ……..... 8

I.5.2. Description des signaux …….. 10

I.5.3. Modulation des signaux …… 11

I.6. Le récepteur Galiléo……..... 11

Chapitre II -Généralités sur la technologie FPGA-

II.1. Introduction…… 15

II. 1.1. Limites de la FPGA ……... 15

II.1.2. Technologie FPGA du Xilinx ……. 18

II.1.3. La famille FPGA Xilinx Virtex-II…… 21

II.1.4. FPGA Virtex-II Pro…….. 22

II.2. Flot de conception FPGA…… 25

II.2.1.Vue Générale …… 25

II.2.2.Les différents étages du flot de la conception …….. 27

II.2.2.1. La synthèse…… 27

II.2.2.2. Simulation fonctionnelle …….. 28
II.2.2.3.Translation en NetList…… 28
II.2.2.4. Implémentation…….. 29
II.2.2.5. Mapping …… 30
II.2.2.4. Placement et Routage…… 30
II.2.2.5. Génération de flot de bits…….. 30
II.2.2.6. Vérification…… 30
II.2.2.7. Le processus « *Back annotation* »……..... 32
II.2.2.8. Simulation fonctionnelle……... 33
II.2.2.9. Simulation temporelle……... 34
II.2.2.10. Analyse temporelle statique…… 34
II.2.2.11. Vérification sur le circuit……. 34
II.3. Architecture d'un Circuit FPGA……. 34
II.3.1. Le circuit configurable …… 35
II.3.2. La couche réseau mémoire …….. 35
II.4. Le module ADM-XRC-II…… 36
II.4.1. Présentation de la plateforme ADM-XRC-II……... 36
II.4.2. Spécifications techniques du module ADM-XRC-II……... 37
II.5. Le langage VHDL ……..... 38
II.5.1. Structure d'une description VHDL……. 39
II.5.1. 1. Déclaration des bibliothèques ……. 39
II.5.1. 2. Déclaration de l'entité et des entrées/sorties ……... 40
II.5.1. 3. Déclaration de l'architecture correspondante à l'entité ……. 40
II.5.2. Les Opérateurs VHDL ……. 41
II.5.3. Exemple d'une description VHDL simple …….. 42

CHAPITRE III -Etude des structures de la détection adaptative CFAR-

III. 1. Introduction……... 43
III.2. L'étalement de spectre …….. 44
III.3. Codes d'étalement pour GPS/GALILEO …….. 46
III.3.1. Les codes de longueur maximale ……... 46
III.3.2. Les codes de Gold ……. 47
III.3.3. Les codes C/A …… 48

III.3.4. Les codes BOC…… 49
III.4. L'acquisition Du Signal …… 52
III.4.1. Etage d'estimation ……. 53
III.4.1.1.L'acquisition en recherche série……... 53
III.4.1.2.L'acquisition parallèle sur la fréquence…… 54
III.4.1.3. L'acquisition parallèle sur le code phase (acquisition circulaire) ……... 57
III.4.2 Etage de détection ……... 58
III.5 Détection adaptative CFAR…… 60
III.5.1. Détection adaptative CA-CFAR……. 62
III.5.2. Détection adaptative GO-CFAR……. 63
III.5.3. Détection adaptative SO-CFAR…….. 63
III.5.4. Détection adaptative OS-CFAR…….. 63
III.6. Evaluation des performances des différents détecteurs……. 64

CHAPITRE IV -Simulation et réalisation des structures CFAR utilisées pour l'acquisition du signal Galiléo,

IV.1. Introduction ……... 67
IV.2. Simulation des structures CFAR pour l'acquisition sous Matlab …… 67
IV. 3. simulation des structures CFAR en utilisant le VHDL……..... 73
IV.3.1. Les étapes nécessaires au développement d'un projet sur FPGA……..……. 73
IV.3.1. 1.Rédaction du texte VHDL …….. 74
IV.3.1. 2.Vérification des erreurs ….…….. 74
IV.3.1. 3.La synthèse ……. ……... 74
IV.3.1. 4.La simulation : …… 75
IV.3.1. 5.Optimisation, placement et routage ……..... 75
IV.3.1. 6.Programmation du composant et test ...……... 76
IV.3. 2.Réalisation des structures CFAR ...…….……... 76
IV.3.2.1. Le CA-CFAR.…….……..... 76
IV.3. 2.2. Le GO-CFAR.…….……..... 81
IV.3. 2.3. Le SO-CFAR.…….……..... 82
IV.3.3. Combinaison des trois structures CFAR par un multiplexeur.…….. 84

CONCLUSION GENERALE ……. 86
ANNEXES
BIBLIOGRAPHIE

Liste des figures

Figure. I.1. La constellation des satellites Galiléo……. 3
Figure. I.2. Le segment de contrôle du système Galiléo…… 4
Figure. I.3. Le principe de triangulation pour la localisation……. 7
Figure. I.4.Occupation spectrale du signal Galiléo…… 9
Figure. I.5. Schéma simplifié d'un récepteur Galélio…… 12
Figure. I.6. Chaîne de réception RF…….. 13
Figure. I.7. Le synoptique d'un canal récepteur……. 14
Figure. II.1.Comparaison des temps de routage entre circuits configurables…… 16
Figure. II.2. Réseau d'aiguillage typique d'un FPGA……... 17
Figure. II.3. Zones de congestion dans le routage des FPGA…… 17
Figure. II.4.Organisation des FPGA de Xilinx. …… 18
Figure. II.5. Architecture simplifiée d'un « Slice »……... 19
Figure. II.6. Les modes de configuration des LUT…… 20
Figure. II.7. Configuration des blocs d'E/S……... 20
Figure. II.8.Détail de l'architecture d'un composant Virtex-II…….. 22
Figure. II.9. Vue Globale du VIRTEX-II PRO…….. 23
Figure. II.10.Architecture standard autour du PowerPc405 sur VIRTEX-II PRO……. 24
Figure. II.11.Flot de conception générique……..... 26
Figure. II.12.Flot de synthèse sous Xilinx……. 28
Figure II.13.Implémentation sous Xilinx……... 29
Figure II.14.Le processus de vérification sous Xilinx…….. 31
Figure II.15.Le processus « Back-Annotation ». …….. 32
Figure II.16.Architecture interne du FPGA……... 35
Figure II.17. Plateforme de prototypage ADM-XRC-II (Alpha Data)…….. 37
Figure II.18. Structure de base d'une description VHDL…….. 39
Figure. III.1. Etalement du spectre……. 44
Figure. III.2. Rejection des interférences à bande étroite dans les systèmes à étalement….. 45
Figure III.3. Génération d'un code pseudo-aléatoire de longueur N = 31……. 46
Figure III.4. Fonction d'autocorrélation d'un code pseudo-aléatoire……. 47
Figure II.5.Génération du code C/A……... 48

Figure III.6. Fonction de corrélation du code C/A……. 49
Figure. III.7 . Générateur du signal BOC……... 51
Figure. III.8. Fonction de corrélation du code BOC(1,1)……... 51
Figure. III.9. Processus d'estimation du code phase et du Doppler……..... 52
Figure. III.10. L'acquisition en recherche série……. 53
Figure. III.11. L'acquisition parallèle sur la fréquence…….. 54
Figure. III.12. L'acquisition circulaire……... 57
Figure. III.13.Densité de probabilité en absence et présence du signal …… 59
Figure. III.14. L'acquisition adaptative dans les systèmes CDMA……... 60
Figure. III.15. L'étage CFAR dans les systèmes CDMA…….. 61
Figure. III.16. Structure générale d'un circuit CFAR…… 62
Figure III.17. Performance de détection des détecteurs CA, GO et SO CFAR pour le code CA…….. 65
Figure III.18. Performance de détection des détecteurs CA, GO et SO CFAR pour le code BOC……... 65
Figure III.19. Performance de détection pour un nombre de cellules n=8,12 et 16 dans le cas du code BOC……..... 66
Figure III.20. Comparaison entre les deux codes C/A et BOC dans le cas du détecteur CA_CFAR……. 66
Figure. IV.1. Acquisition d'un code C/A sans multi trajet…… 68
Figure. IV.2. Acquisition du BOC(1,1) sans multi trajet …….. 68
Figure. IV.3. Détection CFAR du code C/A sans multi trajet……... 69
Figure. IV.4. Détection CFAR du BOC(1,1) sans multi trajet ……. 69
Figure. IV.5. Acquisition d'un code C/A avec multi trajet…… 71
Figure. IV.6. Acquisition du BOC(1,1) avec multi trajet …….. ……..... 71
Figure. IV.7. Détection CFAR d'un signal CA en présence de multi trajet …….. 72
Figure IV.8. Détection CFAR d'un signal CA en présence de multi trajet ……... 72
Figure. IV.9. Etapes nécessaires au développement d'un projet FPGA…….. 73
Figure. IV.10. Vue générale du CA_CFAR généré par le synthétiseur…… 77
Figure. IV.11. Les différents blocs du CA_CFAR générés par le synthétiseur…… 77
Figure. IV.12. Structure interne du registre à décalage du CA_CFAR……. 78
Figure. IV.13. Structure interne du multiplieur du CA_CFAR …… 79
Figure. IV.14. Structure interne du comparateur du CA_CFAR …… 79

Figure. IV.15. Simulation des résultats sur le CA_CFAR…… 80
Figure. IV.16. Structure du GO_CFAR généré par le synthétiseur…… 81
Figure. IV.17. Simulation des résultats sur le GO_CFAR…… 82
Figure. IV.18. Structure du SO_CFAR généré par le synthétiseur …….. 83
Figure. IV.19. Structure du CFAR généré par le synthétiseur…… 84
Figure. IV.20. Structure interne du CFAR généré par le synthétiseur ……... 84

Liste des abréviations

Abréviation	Signification
ASIC	Application-Specific Integrated Circuit
IOB	Input Output Blocks
BOC	Binary Offset Carrier
C/A	Code - Acquisition
CA-CFAR	Cell Averaging -CFAR
CDMA	Code Division Multiple Access
CFAR	Constant False Alarm Rate
CLB	Configurable Logic Bloc
CS	Commercial Service
CS	Commercial Service
CUT	Cell Under Test
DCM	Digital Clock Manager
DLL	Data Locked Loop
EDIF	Electronic Data Interchange Format
EDK	Embedded Development Kit
EDK	Embedded Development Kit
ESA	European Space Agency
FPGA	Field Programmable Gate Array
GNSS	Global Navigation Satellite System
GO-CFAR	Greatest Of-CFAR
GPS	Global Positioning System
IEEE	Institue of Electrical and Electronics Engineers
ISE	Integrated Software Environment
ISE	Integrated Software Environment
ITRF	International Terrestrial Reference Frame
LUT	Look-Up Table
MEO	Mid Earth Orbit
MSB	Max Signifiant Bit
NCD	Native Circuit Description

NGA	Native Generic Annotation
NGD	Native Generic Database
NGO	Native Generic Object
NGO	Native Generic Object
NMC	hard placed and routed macros
OS	Open Service
OS-CFAR	Order Statistic CFAR.
PLB	Processor Local Bus
PLL	Phase Locked Loop:
PRN	Pseudo Random Number :
PRS	Public Regulated Service
PRS	Public Regulated Service
RF	Radio Frequency
SAR	Search And Rescue
SDF	Standard Delay Format
SDRAM	Synchronous Dynamic Read Access Memory
SO-CFAR	Smallest Of -CFAR
SoL	Safety of Life
SPS	Standard Positionning Service
SRAM	Static
TRACE	Timing Reporter And Circuit Evaluator
TTL	Transistor-Transistor Logic
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VHSIC	Very High Speed Integrated Circuits
WGS84	Word Geodesic System 1984

Liste des tableaux

Tableau I.1. Fréquences porteuses pour le système GALILEO……... 9
Tableau I.2. Bande de réception et polarisation des signaux GALILEO……. 10
Tableau I.3. Caractéristiques des signaux data du système GALILEO……... 11
Tableau II.1. Famille FPGA VIRTEX-II……. 21
Tableau II.2. Ressources du circuit VIRTEX-II PRO/ VIRTEX-II PRO X…… 25
Tableau IV.1. Ressources du circuit FPGA utilisées pour le circuit CA_CFAR……. 79
Tableau IV.2. Ressources du circuit FPGA utilisées pour le circuit GO_CFAR……. 81
Tableau IV.3. Ressources du circuit FPGA utilisées pour le circuit SO_CFAR……. 83
Tableau IV.4. Ressources du circuit FPGA utilisées pour le circuit CFAR…… 85

INTRODUCTION GENERALE

CONCLUSION GENERALE

ANNEXES

BIBLIOGRAPHIE

REMERCIEMENTS

Introduction

Dans les applications de positionnement par systèmes de navigation GPS ou le futur GALILEO, la navigation en milieu urbain semble d'être l'une des plus importantes. L'environnement urbain est caractérisé par la présence d'un grand nombre d'obstacles qui produisent des trajets multiples. Il est donc primordial de développer des méthodes ou des techniques de traitement de signal qui permettent de les réduire ou les éliminer.

L'acquisition d'un signal satellitaire est l'une des opérations qui influence la précision de la mesure puisqu'elle permet essentiellement d'identifier les satellites visibles à un moment donné, de plus, l'acquisition est soumise assez souvent à plusieurs incertitudes sachant que le signal est affecté par le bruit et les multi trajets. Cette opération d'acquisition est composée principalement, de deux étages, le premier réalise la recherche des paramètres, retard et décalage Doppler, le deuxième est dédié à la détection. Généralement, l'approche de détection implémentée dans les récepteurs actuels, impose de choisir un seuil de détection fixe et faible ce qui engendre un nombre élevé de fausses alarmes. Comme solution à ce problème, nous proposons d'utiliser la détection adaptative CFAR (Constant False Alarm Rate) qui était auparavant utilisée dans les radars.

Dans ce contexte, notre travail consiste à étudier différentes structures CFAR en analysant leur probabilité de détection, les simuler sous MATLAB pour différentes situations de multitrajets et finalement les implémenter sur FPGA (Field Programmable Gates Array) en utilisant la programmation VHDL.

Pour ce faire, notre travail sera organisé en quatre chapitres. Le premier chapitre présente des généralités sur le système de positionnement par satellites GALILEO. Le deuxième chapitre sera dédié à des notions sur la technologie FPGA ainsi que quelques généralités sur le langage VHDL. Nous élaborons des notions concernant la détection adaptative ainsi que des généralités sur quelques codes utilisés dans les systèmes CDMA dans le troisième chapitre. Finalement, un dernier chapitre sera consacré aux simulations des structures CFAR sous Matlab, et la présentation des différentes étapes suivies pour la réalisation de ces structures CFAR ainsi que leurs implémentations. Finalement, des discussions des résultats seront proposées.

CHAPITRE I

GENERALITES SUR LE SYSTEME GALILEO

TABLE DES MATIERES

1

I.1.Introduction …….... 2

I .2. Description du système GALILEO …… 2

I.3. Les services offerts par le système GALILEO……. 5

I.4. Détermination de la position et précision attendue…… 7

I.5. Les signaux Galiléo……...…... 8

I.6. Le récepteur Galiléo……...……... 11

CHAPITRE II

GENERALITES SUR LA TECHNOLOGIE FPGA

TABLE DES MATIERES

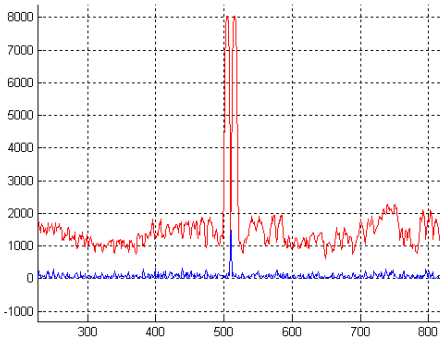
II.1.Introduction……. …… 15

II.2. Flot de conception FPGA ……... 25

II.3. Architecture d'un Circuit FPGA……. 33

II.4. Le module ADM-XRC-II……. 36

II.5. Le langage VHDL …….. 38

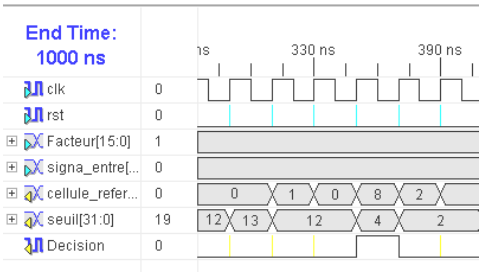


CHAPITRE III

LA DETECTION CFAR

TABLE DES MATIERES

III. 1. Introduction ……..... 43
III.2. L'étalement de spectre …… 44
III.3. Codes d'étalement pour GPS/GALILEO …… 46
III.4. L'acquisition du signal …… 52
III.5. Détection adaptative CFAR …… 60
III.6. Evaluation des performances des différents détecteurs ……... 64



CHAPITRE IV

RESULTATS & SIMULATIONS

TABLE DES MATIERES

4

IV.1. Introduction 67

IV.2. Simulation des structures CFAR pour l'acquisition sous Matlab 67

IV.3. Simulation des structures CFAR en utilisant le VHDL..... 73

I.1. INTRODUCTION :

La navigation est la science et les techniques qui permettent de connaitre la position d'un mobile par rapport à un système de référence ou un point fixe déterminé, de fournir des informations concernant la route à suivre pour rejoindre un autre point de coordonnées connues et d'estimer avec un certain degrés de précision toute information relative au déplacement de ce mobile : distance, vitesse, heure estimée d'arrivée…

La navigation par satellite est une technologie qui a occupé une vaste place dans le domaine de la navigation aérienne, et constituera l'unique clé pour un bel avenir doté de performances meilleures à savoir l'exactitude, l'intégrité et la continuité du service, dans le but de préserver l'objectif principal qui est la sécurité.

Le GPS (*Global Positioning System*) fut le premier système conçu pour répondre à ces besoins et est actuellement le seul système mondiale de positionnement, la Russie dispose de son système GLONASS, l'Inde se propose de développer dans les sept années à venir son système régional IRNSS et la Chine se prépare à lancer les premiers satellites de son système régional BEIDOU.

La « vedette » des futurs systèmes de navigation sera sans doute la merveille des Européens GALILEO promettant d'offrir plusieurs services d'autant plus performants que précis imbattables par l'actuel GPS.

I .2. DESCRIPTION DU SYSTEME GALELIO :

Le programme de navigation par satellite GALILEO est le plus grand projet du genre lancé par la collaboration entre l'Union Européenne et l'agence spatiale européenne, l'ESA (*European Space Agency*). Déjà en 1994, la Commission Européenne a soulevé la nécessité de contribuer au développement d'un nouveau système de positionnement par satellite nommé GNSS (*Global Navigation Satellite System*).

Le 26 mars 2002, le Conseil Européen est arrivé à un accord unanime quant au lancement du nouveau programme civil de navigation par satellite GALILEO. Ce nouveau système contribuera à accélérer la popularisation des systèmes de positionnement par satellites dans tous les domaines liés à la navigation et facilitera son adoption comme moyen principal de navigation. La grande précision et l'intégrité des messages vont contribuer également aux opérations de sauvetages en assurant une couverture globale. Avec la

disponibilité beaucoup plus grande des signaux GALILEO dans les conditions urbaines, le nouveau système de navigation contribuera au développement des autres applications comme le repérage des usagers par les réseaux locaux ou le guidage des mobiles.

Comme pour chaque système de localisation par satellite, GALILEO est constitué de trois principaux segments :

- Le segment spatial ;
- Le segment de contrôle ;
- Les utilisateurs.

I .2.1.Le segment spatial :

La constellation GALILEO est formée de 27 satellites répartis en trois plans orbitaux avec 9 satellites chacun, les orbites sont circulaires moyennes (*MEO*). Le rayon des orbites est de 29.994 km et l'inclinaison est de 56 degrés, à cette altitude, la période de révolution des satellites sur leur orbite est de 14 heures et 21 minutes, cela permettra d'assurer une excellente couverture de la terre, en effet, les signaux GALILEO pouvant être fournis jusqu'à une latitude de 75° degrés Nord, ce qui correspond à la position du Cap Nord et même au-delà.

Pour assurer la redondance nécessaire et le remplacement rapide en cas de pannes des satellites, il est prévu l'utilisation d'un satellite supplémentaire dans chacun des trois plans orbitaux pour arriver à une constellation de 30 satellites au total, d'autres lancements permettront de renouveler les équipements défaillants et donc de régénérer le système au terme de la vie des satellites.

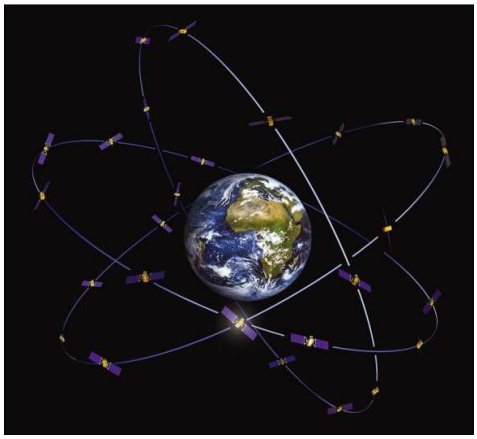


Figure I.1 : *La constellation des satellites GALILEO*

Chaque satellite aura une masse d'environ 700 kg et sera stabilisé suivant les trois axes grâce à un ensemble classique de gyromètres, roues à inertie et magnéto-coupleurs. Leur dimension est de 2,7 par 1,2 par 1,1 m. La puissance embarquée prévue est de 1600 watts.

La charge utile de chaque satellite comprendra un ensemble de quatre horloges atomiques, l'électronique de génération des signaux transmis par les émetteurs (codes pseudo aléatoires orthogonaux de modulation des porteuses, signaux de navigation proprement dits), ainsi que les émetteurs radio fréquence en bande L et les antennes correspondantes.

I .2. 2. Le segment de contrôle :

De manière externe au système de contrôle de la constellation, un ensemble de stations et un centre opérationnel spécialisé, formant ce qu'on appelle « le segment d'intégrité », surveillera en permanence les performances du système et aura la capacité d'informer en temps réel les utilisateurs du système de son état et de ses performances. Cette information sera fournie via les satellites de la constellation GALILEO, qui remplissent là une fonction de relais de l'information d'intégrité. Cette fonction d'intégrité est essentielle pour beaucoup d'applications où la connaissance instantanée du bon état de fonctionnement du système est indispensable (navigation aérienne par exemple). Il est important de savoir qu'elle n'est pas fournie dans le système GPS actuellement en service.

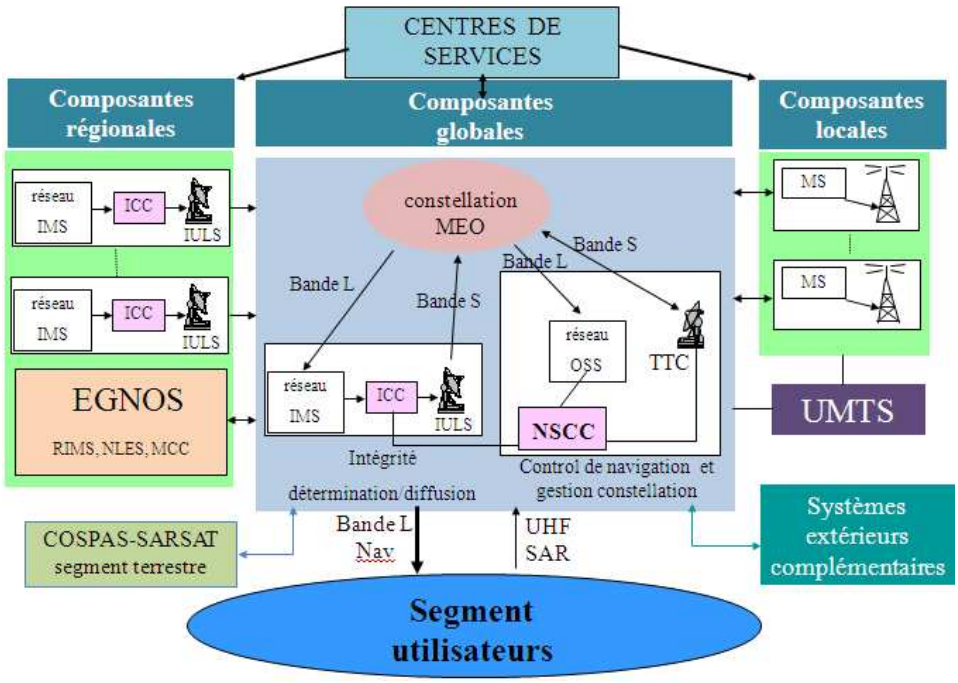


Figure I.2 : *Le segment de contrôle*

Le segment de contrôle sera composé essentiellement de deux centres de contrôle qui se situeront en Europe, ainsi que des stations de contrôle qui seront d'un nombre de vingt chargées de la réception des télémétries, de l'envoi des télécommandes aux satellites et du suivi de ceux-ci seront réparties autour du globe. Les données seront centralisées par les deux centres de contrôles. Les autres stations sont chargées de la mesure de la qualité du signal de navigation: c'est l'intégrité du système citée au part avant.

I .2. 3. Le segment utilisateur :

Il est constitué de l'ensemble des utilisateurs bénéficiant des services du GALILEO militaires qu'ils soient ou civils, disposant de récepteurs capables de décoder et d'exploiter le signal reçu des satellites afin de fournir au minimum les informations de temps, de position et de vitesse.

L'utilisation militaire pourra être le guidage des bombes et des missiles. Pour les civils le domaine d'utilisation est assez vaste, les services qu'offrira GALILEO lui permettront d'équiper les véhicules, les bateaux ainsi que les avions. Ces derniers ont été depuis longtemps équipés de récepteurs GPS et leurs navigation se fait à base de ses informations, le pilote automatique par exemple utilise les informations du GPS en temps réel, la précision pour ces utilisateurs est de 20 mètres sans la dégradation volontaire et pourra atteindre 3 mètres avec les augmentations faites pour le GPS, mais le GPS est exclu et non certifié comme moyen d'atterrissage vu les exigences des catégories d'atterrissage élaborées par l'OACI, en mettant en œuvre la notion d'intégrité. GALILEO par contre sera capable d'assurer cette fonction.

Les services de GALILEO lui permettront de toucher un grand nombre de domaines, la topographie, la géodésie, les applications industrielles et agricoles ne pourront sans doute pas s'en passer.

I.3. LES SERVICES OFFERTS PAR LE SYSTEME GALILEO :

Les services Galileo sont adoptés OS (Open Service), SoL(Safety of Life), PRS(Public Regulated Service), CS(Commercial Service), SAR(Search And Rescue) et ERIS(External Regional Integrity Service). L'utilisateur pourra obtenir un positionnement avec les services OS et SoL (en accès libres) et PRS(en accès restreint).les autres servies, CS, SAR et ERIS ne

fournissent pas de positionnement, mais consistent à diffusé des informations complémentaire. Ainsi si l'intégrité(ou la garantie) du positionnement sera fournie globalement par l'operateur Galileo à travers les services SoL et PRS, les operateurs régionaux pourront assurer eux-mêmes une intégrité régionale avec les services ERIS.

Le service CS consiste essentiellement à fournir des données à valeur ajoutée, dont la définition est totalement flexible .quand au service SAR, il est dédié à la diffusion des données pour les opérations de recherche et d'assistance.

- ***Le service ouvert*** « Open Access », est le service de base permettant la localisation et la datation comparable au service de base fourni par le GPS américain (Service SPS). Il est gratuit et ne comprend aucune restriction d'accès. C'est le service « grand public » qui concernera la majorité des utilisateurs.
- ***Le service de sûreté de la vie*** est plus connu sous son sigle anglais « Safety of Life » (SoL). Il s'agit du service ouvert complété par un signal d'intégrité indispensable pour toutes les applications où l'absence de cette information d'intégrité pourrait mettre en danger des vies humaines (transport aérien, ferroviaire et maritime).
- ***Le service commercial*** ou « Commercial Service » (CS), est destiné aux applications Commerciales exigeant une précision supérieure à celle que fournit le service ouvert. Il utilise deux signaux supplémentaires, protégés par un chiffrement commercial décryptable par les terminaux équipés et disposant de la clef d'accès. Ce service sera géré par les fournisseurs d'accès au service commercial Galileo. Il se prête particulièrement bien aux services à valeur ajoutée qui seront proposés en complément de la navigation.

- ***Le service public réglementé*** ou « Public Regulated Service » est réservé aux applications gouvernementales (sécurité civile, transports, militaires) pour lesquelles la continuité du service doit être garantie quelles que soient les circonstances, donc particulièrement robuste vis-à-vis de brouillages éventuels ou d'interférences électromagnétiques accidentelles. Ce service PRS utilise deux signaux dédiés et chiffrés, dont un sur la même fréquence que le futur code militaire (code M) du GPS. Son accès sera contrôlé par les autorités en charge des questions de sécurité.
- ***Le service de recherche et sauvetage*** « Search and Rescue » en anglais (SAR), poursuivra à l'ère de Galileo le service rendu par le système actuel.

SARSAT/COSPAS mis en place par le Canada, la France, les États-Unis et la Russie depuis 1982. Il permet de recueillir et de localiser les émissions des balises de détresse opérant à 406 MHz, en service sur les avions, les bateaux et pour les expéditions terrestres.

I.4. DETERMINATION DE LA POSITION ET PRECISION ATTENDUE:

Comme pour le GPS, la position du terminal utilisateur est calculée à partir des mesures de distance (pseudo-distances) entre le terminal et au moins quatre satellites de la constellation dont les signaux sont reçus simultanément. Plus précisément, la différence des distances entre le terminal et deux satellites permet de placer celui-ci sur un lieu géométrique qui est une surface dans l'espace appelée hyperboloïde, et la simultanéité de mesures de ce type avec quatre satellites permet de calculer la position géographique par intersection géométrique entre ces surfaces, ainsi que l'écart entre l'échelle de temps du terminal et celle du système GALILEO.

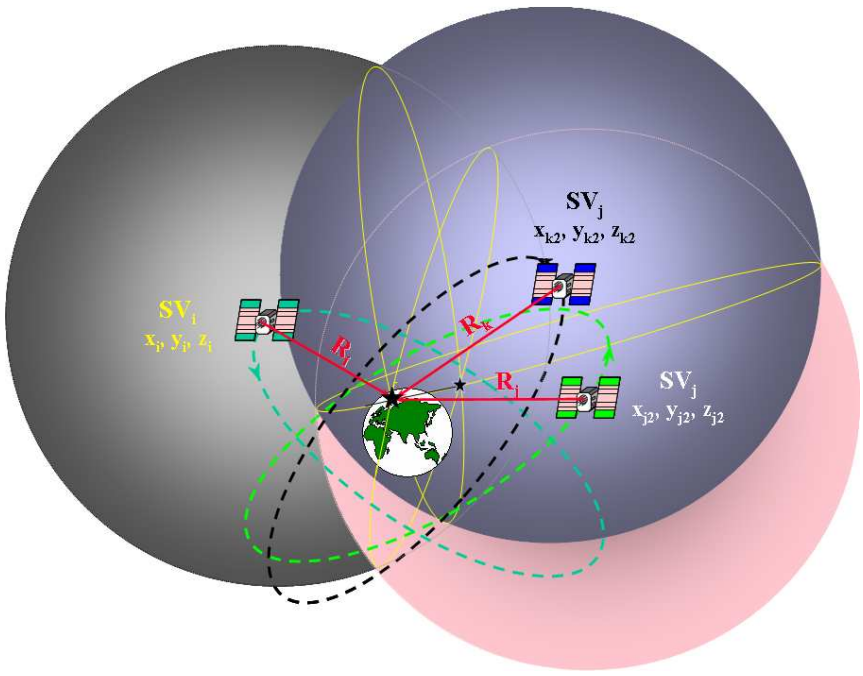


Figure I.3 : *Le principe de triangulation pour la localisation*

Cette méthode suppose, bien sûr, que soit connue avec une précision suffisante la position des satellites dans l'espace rapportée à un référentiel géodésique mondial commun à tous les utilisateurs. Celle-ci est calculée à partir des éphémérides contenues dans les signaux de navigation transmis par les satellites, calculées au sol et injectées périodiquement en mémoire par le réseau de stations de contrôle de la constellation.

Les éphémérides des satellites sont élaborées à partir d'un ensemble de stations d'orbitographie dont les positions sont précisément connues dans le repère international de référence terrestre, l'ITRF ou ce qui est équivalent au mètre près, dans le repère WGS84 déjà établi pour le GPS et très utilisé aujourd'hui.

I.5. LES SIGNAUX GALILEO :

I.5.1. Particularité des signaux GALILEO :

Le système de navigation par satellite GALILEO offre dix signaux répartis dans les bandes de fréquences E5a, E5b, E6 et E2-L1-E1. L'occupation spectrale des signaux est représentée dans la figure I.4. Quatre fréquences porteuses sont prévues dont certaines sont les mêmes que pour le système GPS.

Six signaux incluant les trois canaux non modulés avec les données de navigation sont accessibles par tous les utilisateurs de GALILEO dans les bandes E5a, E5b et L1 pour le service OS (*Open Service*) et pour les services SoL (*Safety-of-life Service*). Les deux signaux dans la bande E6 incluant un canal non modulé sont destinés aux utilisateurs concernés pour le service CS (*Commercial Service*). Finalement, les deux derniers signaux partagés entre la bande E6 et la bande E2-L1-E1, sont prévus pour le service autorisé PRS (*Public Regulated Service*). Les schémas de modulations des signaux résultent du plan de fréquences présenté ci-dessous et ils seront décrits dans ce qui suit.

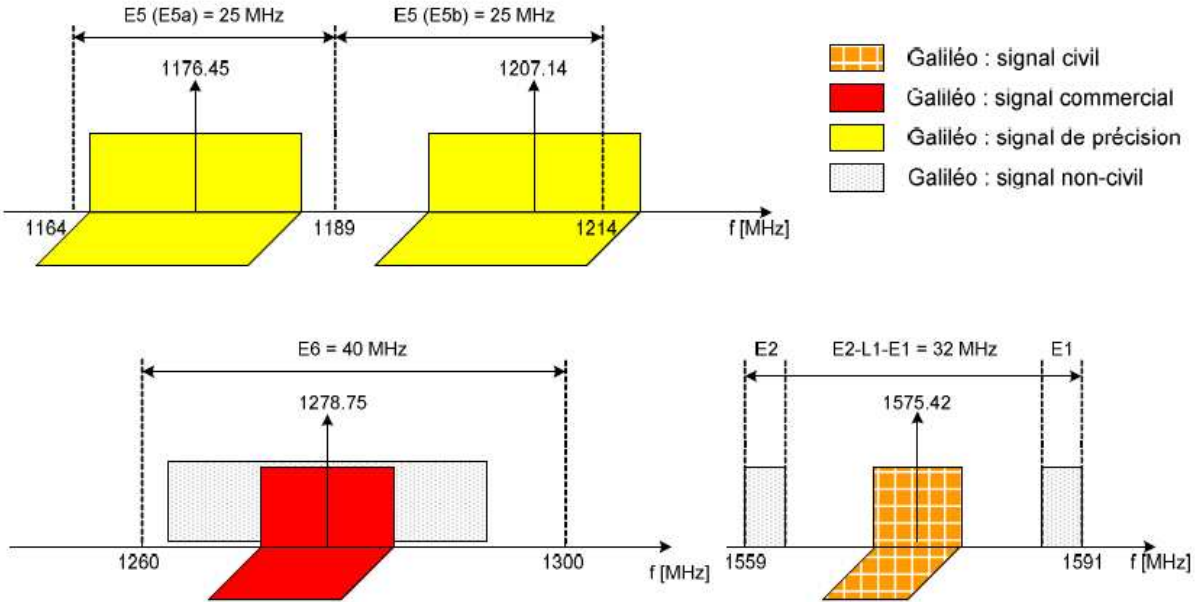


Figure I.4 : *Occupation spectrale du signal GALILEO*

Il faut mentionner que certains signaux n'ont pas encore un statut définitif, ils font toujours l'objet de recherche et d'analyse plus poussée. L'enjeu principal est la question de compatibilité et de meilleure interopérabilité entre le nouveau système de positionnement GALILEO et les systèmes existants (GPS et GLONASS). Le bon choix des signaux est très important car il permettra non seulement de ne pas dégrader les services actuellement disponibles, mais aussi de profiter des avantages offerts par l'utilisation de récepteurs compatibles avec les deux systèmes (une plus grande constellation de satellites, plusieurs fréquences porteuses, une variété de signaux, etc.). Les fréquences porteuses, les bandes passantes et la polarisation des signaux Galileo sont classées dans le tableau I.1 et le tableau I.2.

Signal	***Fréquence porteuse***
E5a	1176.45 MHz
E5b	1207.14 MHz
E5	(E5a+E5b) 1191.795 MHz
E6	1278.75 MHz
E1	1575.42 MHz

Tableau I.1 : *Fréquences porteuses pour le système Galileo*

Signal	*Bande de réception*	*Polarisation*
E5	51.150 MHz	Circulaire droite
E6	40.92MHz	Circulaire droite
E1	24.552MHz	Circulaire droite

Tableau I.2 : *Bande de réception et polarisation des signaux Galileo*

La composition des deux signaux E5a et E5b est considéré en un seul signal E5 de fréquence centrale ($f_{Ea} + f_{Eb}$)/2. Les deux fréquences étant proches l'une de l'autre, ils peuvent être traitées comme une seule bande large avec une implémentation spécifique au niveau des récepteurs. E5 n'est pas un signal à part entière, il est juste la composition des deux signaux E5a et E5b.

I.5.2. Description des signaux :

Chaque satellite Galileo transmet des messages dans les signaux E1, E6, E5a et E5b.

- **Signal E1 :**

Ce signal est transmis sur la bande L1 comprenant deux canaux (E1-B et E1-C). Le canal E1-B transmet des données et le canal E1-C transmet des signaux pilotes. Les signaux pilotes permettent, comme indique leurs noms, de guider le récepteur pour l'acquisition du signal de données.

- **Signal E6** :

Ce signal transporte des informations pour un but commercial. Il comprend deux canaux (E6-B et E6-C). Le premier étant un canal de données et le deuxième transmet des signaux pilotes.

- **Signal E5a** :

Ce signal est accessible pour le grand public. Il est transmis sur la bande E5 et comprend deux canaux (données, pilote). Il transporte des données de navigation accessible pour tous les utilisateurs.

- **Signal E5b** :

Ce signal est aussi transmis sur la bande E5 et comprend aussi deux canaux, un pour les données de navigation et un pour les signaux pilotes. Ce signal comporte des données non cryptées accessible par tous les utilisateurs et des données crypté réservées pour un usage commercial.

I.5.3. Modulation des signaux :

Les signaux E5a et E5b sont modulés sur une seule porteuse E5 en utilisant une technique de modulation appelé AltBOC (***A**lternate **B**inary **O**ffset **C**arrier*). La composition des deux signaux E5a et E5b est considéré en un seul signal E5 et peut être traité comme une seule bande large avec une implémentation spécifique au niveau des récepteurs. La puissance est divisée en deux entre le canal de données et le canal pilote, le tableau I.3 récapitule les caractéristiques des signaux Galileo.

Signal Minimale	Canal	Modulation	Puissance de réception (dBW)
E5	E5a data	AltBOC(15,10)	-155
E5	E5b data	AltBOC(15,10)	-155
E6	E6-B data	BPSK(5)	-155
E1	E1-B data	BOC(1,1)	-157

Tableau I.3 : *Caractéristiques des signaux data du système Galileo*

I.6. LE RECEPTEUR GALILEO :

Un récepteur GALILEO contient, de manière générale, les modules suivants :

- L'antenne de réception ;
- La chaîne de réception radiofréquence ;
- Un module de traitement du signal en bande de base ;
- Un module numérique qui est dédié à l'acquisition et la synchronisation des boucles de code, et au calcul de la corrélation ;
- Un module de navigation et de calcul des coordonnées ;
- Interface homme machine.

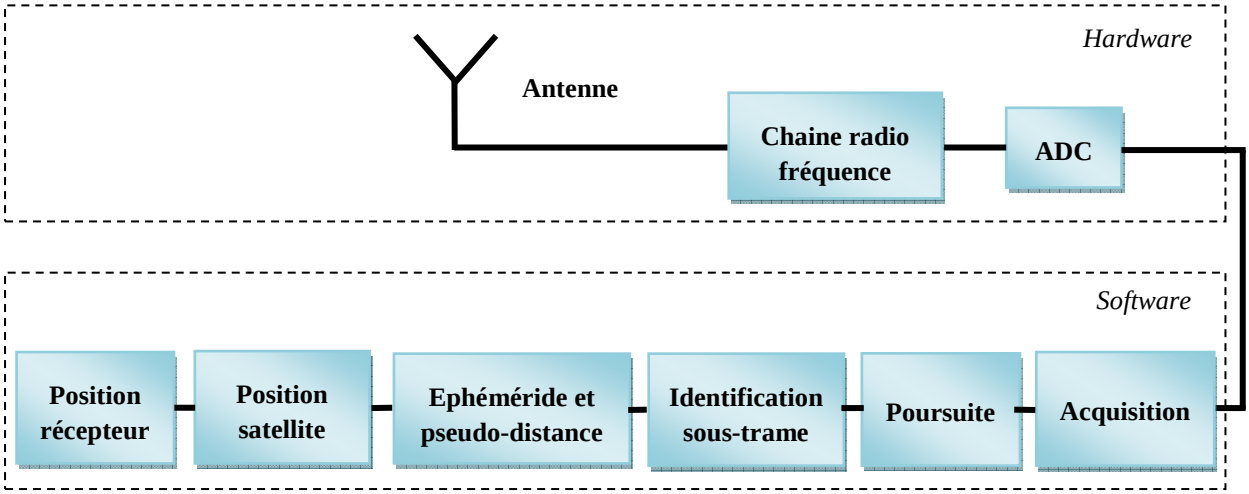


Figure I.5 : *Schéma simplifié d'un récepteur Galiléo*

➢ *L'Antenne de Réception :*

L'antenne, élément de réception du signal, agit sur le niveau de signal (absorption) et sur les interférences (réflexion) produisant des atténuations et des multiples trajets qui faussent la mesure jusqu'à la rendre impossible. Tout d'abord cette antenne doit être dégagée au maximum de tout obstacle.

L'antenne sera soigneusement choisie en fonction de l'utilisation principale du récepteur. Le gain est un facteur très important qui caractérise une antenne. Il est généralement compris entre 20 et 30 dB. Au-delà de 30 dB de gain, certains récepteurs GPS par exemple peuvent présenter un disfonctionnements dus à une saturation de l'étage d'entrée du récepteur. Il faut aussi tenir compte de l'atténuation du câble qui peut être compris entre 0,2 et 1,5 dB (suivant le câble utilisé). Les autres facteurs importants caractérisant une antenne sont la réjection des fréquences parasites, le bruit généré par l'amplificateur de l'antenne (≈4.0 dB).

Si une antenne est utilisée pour recevoir les fréquences f_1 et f_2, elle peut avoir une grande largeur de bande de fréquence pour couvrir la plage de fréquence entière ou avoir deux bandes étroites pour couvrir les intervalles de fréquences désirées. Une antenne avec deux bandes étroites peut éviter les interférences des signaux entre les deux bandes.

Le filtre d'antenne élimine les signaux indésirables dus aux fréquences images. Il est placé avant l'ampli RF, de façon à éviter sa saturation par des signaux hors de la bande utile.

- ***La chaîne de réception radiofréquence :***

Cette partie est chargée d'amplifier, de filtrer et de faire la conversion du signal reçu en fréquence intermédiaire FI.

Le changement de fréquence permet d'amplifier et de filtrer à une fréquence fixe. Le filtre utilisé dépend de la bande relative. Il doit supprimer les signaux indésirables à des fréquences proches de f_p, ainsi que les composantes indésirables générées par le mélangeur.

Les grandes opérations que réalise la chaîne de réception radiofréquence sont présentées dans le schéma de la figure suivante :

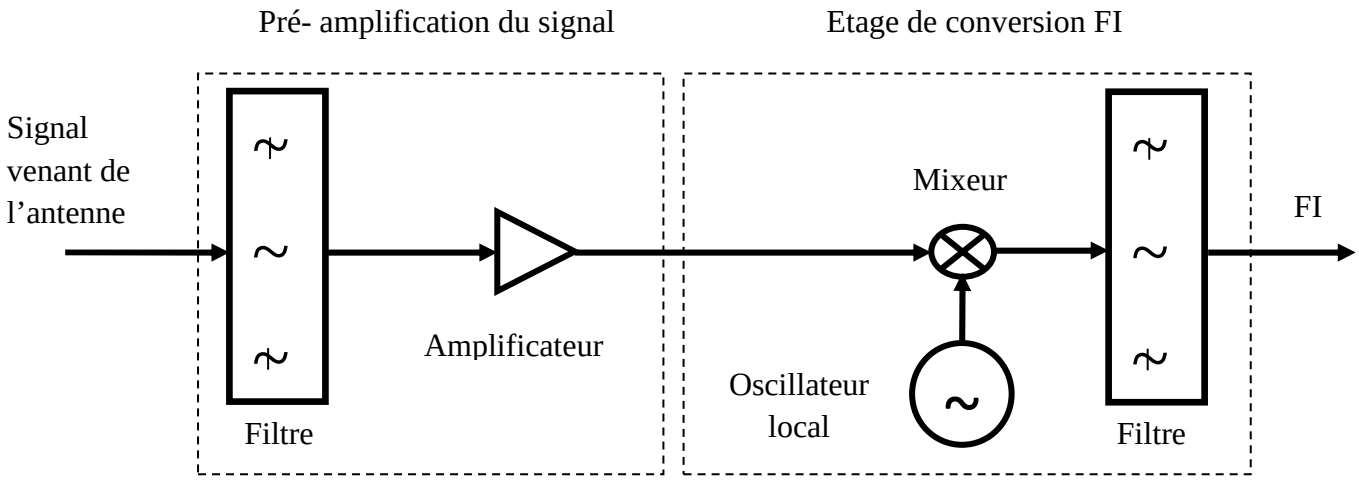


Figure I.6 : *Chaîne de réception RF*

➢ ***Processeur de Navigation :***

Ce dernier est chargé de traiter le message de navigation, d'effectuer la correction des mesures faites par l'étage de traitement de signal. Il permet de déterminer les coordonnées du récepteur.

➢ ***Interface homme machine:***

Cette interface est conçue pour recevoir des instructions provenant de l'utilisateur et les données qui se présentent à son entrée.

Le récepteur se compose de plusieurs canaux où les signaux reçus à partir des différents satellites seront traités. Chaque canal se chargera des deux opérations d'acquisition et de poursuite, il contient les étages nécessaires pour le traitement de son signal, et en extraire les informations souhaitées. Il possède la structure suivante :

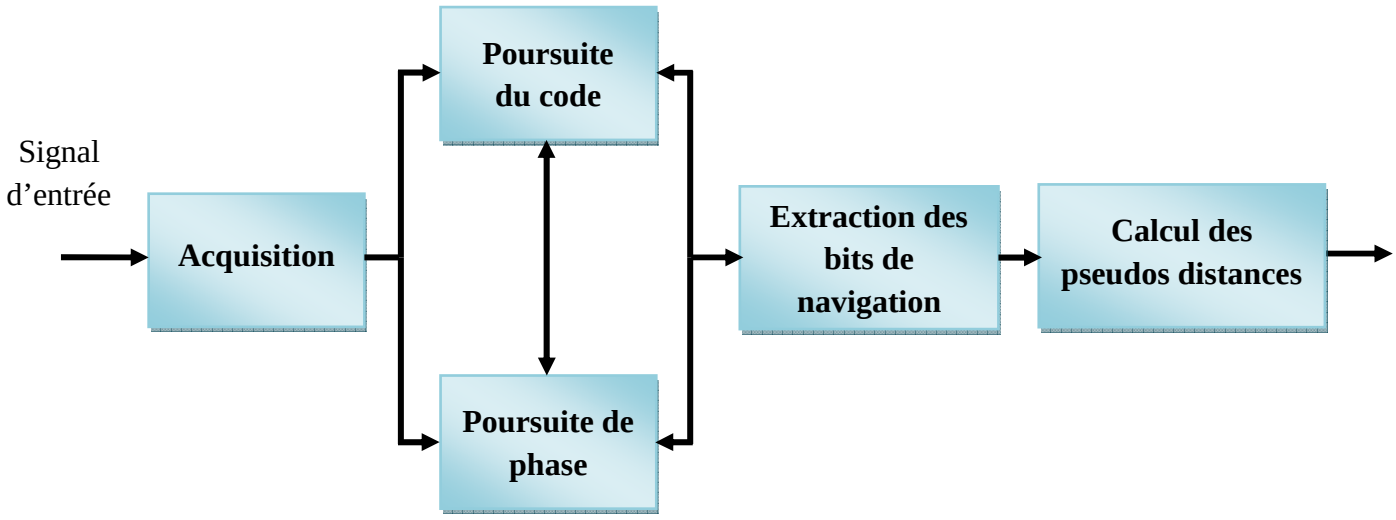


Figure I.7 : *Le synoptique d'un canal récepteur*

La première opération à réaliser est l'acquisition du signal, elle sert à identifier es satellites visibles à un temps donné, de récupérer son code et sa fréquence. Les résultats obtenus seront en suite exploités par l'étage de poursuite, qui permettra de se synchroniser en temps et en fréquence avec le signal reçu en permanence, cette opération se réalise à travers deux boucles de verrouillage de phase PLL et de code DLL. Le fonctionnement simultané des deux boucles permettra de récupérer les bits de navigation qui seront utilisés pour le calcul des pseudos distances.

II.1. INTRODUCTION :

Durant ces deux dernières decenies, les méthodes de conception et d'implémentation des fonctions numériques ont connu un développement important. Dans les années soixante-dix, les applications de la logique câblée était des circuits intégrés standars, pris assez souvent de la famille *TTL*. C'était dans les années quatre-vintg que les premiers circuits programables fûrent apparut, ils peuvent être simples pour les applications ordinaires, ou des circuits intégrés spécifiques (*ASIC*) pour les fonctions complexes. La complexité de ces circuits a necessité la création des logiciels de hauts niveuax, pour leur description structurelle. Actuellement, le langage de programmation pour l'implémentation des circuits le plus utilisé est le *VHDL*.

Les circuits *FPGA* (*Field Programmable Gate Array*) demeurent toujours les circuits reprogrammables les plus développés, les plus fiables et les plus fertiles. Ils sont en effet, entièrement configurables par programmation, et ils permettent d'implanter physiquement, par simple programmation, n'importe quelle fonction logique. De plus, ils ne sont pas limités à un mode de traitement séquentiel de l'information comme avec les microprocesseurs ; et en cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement.

Ainsi, cette technologie, qui ciblait principalement les domaines : public, médical, télécommunication et automobile, devient intéressante pour des applications plus simples et portant sur des volumes moins importants. Et ceci particulièrement depuis que les *FPGA* sont proposés à un prix très faible (0.0001dollar par porte).

II.1.1. Limites de la FPGA :

L'emploi de *FPGA* est devenu, à juste titre, une solution privilégiée pour produire rapidement des circuits. Pourtant, la complexité croissante des plus grandes matrices engendre de graves problèmes à l'étape de routage. Entre les *ASIC* et les *FPGA* de haute complexité, la durée du procédé de placement-routage sous contrainte temporelle varie beaucoup avant de converger vers la solution optimale (Figure II.1).

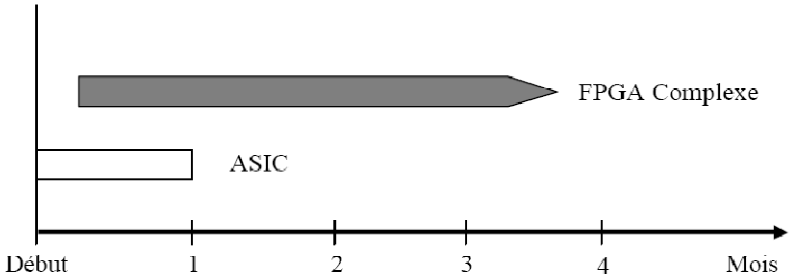


Figure II.1 : *Comparaison des temps de routage entre circuits configurables*

Les *FPGA* ont certes de nombreux atouts, mais les concepteurs de systèmes ont tendance à se tourner, par habitude, vers cette solution sans regarder si elle répond correctement aux objectifs de performances et de coût, voire si elle s'inscrit bien dans le calendrier de développement de leurs projets. Pourtant, les *FPGA* de forte complexité induisent des limitations sévères qui peuvent pénaliser le bon déroulement d'une conception.

Certaines applications, dans le domaine des réseaux et communications notamment, exigent des fonctions complexes et un nombre élevé de broches, ceci assorti de performances toujours croissantes. De ce fait, les plus grands réseaux logiques programmables choisis pour réaliser ces projets sont fréquemment employés au-delà de leurs possibilités physiques, des limitations inhérentes à leurs architectures. Avec de tels *FPGA*, il faut souvent plusieurs mois avant de converger vers un routage satisfaisant au niveau des performances temporelles. Dans ce cas, la convergence vers un timing optimal est presque toujours un procédé long, imprévisible, onéreux, et extrêmement laborieux.

Au niveau des *FPGA* de forte complexité, la difficulté d'utilisation de ces matrices dans des applications de hautes performances réside dans le principe d'interconnexions programmables pour le routage des signaux. Et ceci n'est pas entièrement dû au fait que la quantité de ressources de routage disponibles est plus faible dans les *FPGA* basés sur des points SRAM que dans les technologies concurrentes. Entre en jeu également le retard substantiel, de type R-C, inhérent à ces circuits programmables par *SRAM* (Figure II.2).

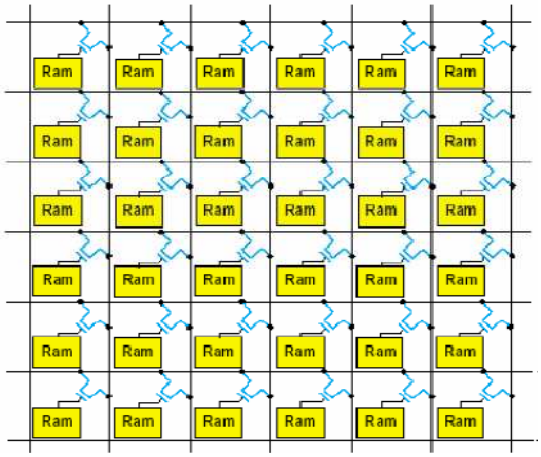


Figure II.2 : *Réseau d'aiguillage typique d'un FPGA*

Chaque point d'aiguillage consomme au moins cinq transistors pour la Ram et un transistor interrupteur de grande taille. Ce délai de propagation rend les performances du composant hautement sensibles au moindre changement dans un chemin de routage. Comme l'illustre la *Figure II.3*, le point de congestion des équipotentielles, situé au centre de la puce, est d'une taille beaucoup plus importante pour les circuits les plus complexes d'une famille de *FPGA*. En outre, quand une interconnexion doit prendre un chemin détourné pour éviter cette congestion, l'opération se traduit par une augmentation conséquente des délais de propagation.

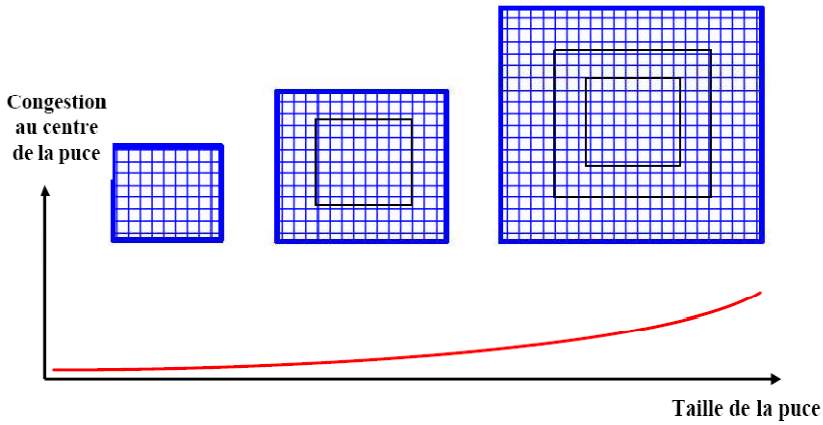


Figure II.3 : *Zones de congestion dans le routage des FPGA*

De ce fait, alors les matrices de petite ou moyenne taille permettront de converger rapidement vers une solution de routage temporellement satisfaisante, cette bonne conclusion demande souvent trois ou quatre mois pour les grands *FPGA*.

II.1.2. Technologie FPGA du Xilinx

Dès l'origine, les *FPGA*, tels que *Xilinx* les a inventés, avaient la réputation de mettre à disposition de l'utilisateur une conception rapide, fiable et simple. Les progrès technologiques ont permis d'accéder à des matrices logiques programmables de plusieurs millions de portes. Cette complexité actuelle reste absolument gérable et permet la réalisation d'applications très performantes moyennant une bonne connaissance des ressources offertes et le respect d'une méthodologie de conception.

La Figure II.4, montre l'architecture simplifiée d'un *FPGA Xilinx* (pour les familles *Spartan-IIE*, *Virtex-E* et *Virtex-II*). Les principales caractéristiques de ces trois familles sont :

- Complexités allant de 1500 à plus de 8 millions de portes ;
- Faible consommation ;
- Grande souplesse d'utilisation des entrées/sorties avec adaptation d'impédance (*Virtex-II*) et configuration en mode différentiel (*Spartan-IIE*, *Virtex-E* et *Virtex-II*) ;
- Fonctions mémoire (distribuée et blocs de Ram) ;
- Dispositifs de gestion des horloges (*DLL* et *DCM*) ;
- Multiplieurs câblés (*Virtex-II*) ;

Et bien d'autres possibilités permettant d'optimiser à la fois la performance et la densité des fonctions logiques et arithmétiques.

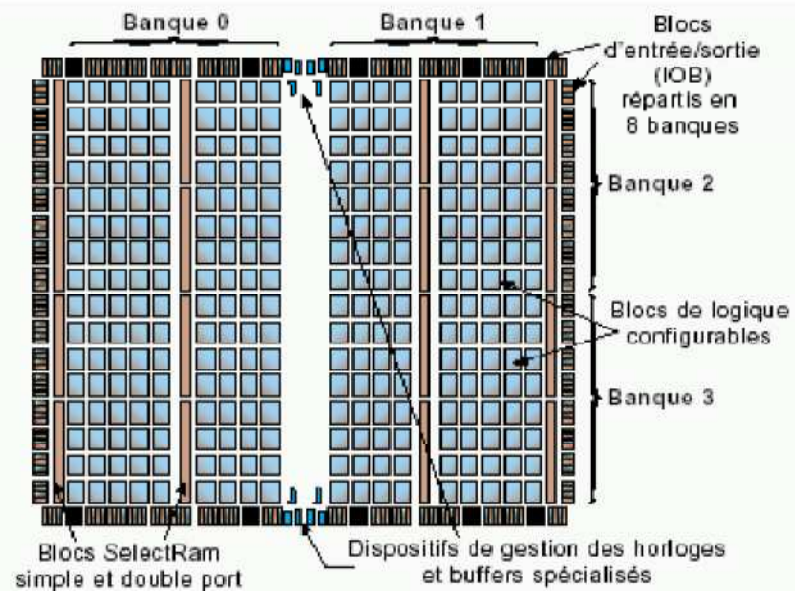


Figure II.4 : *Organisation des FPGA de Xilinx*

Nous y voyons qu'un des modules de base est le bloc logique configurable (ou *CLB*). Lui-même constitué de s*lice*. La *Figure II.5*, illustre l'architecture simplifiée d'un slice. La logique combinatoire est implantée grâce aux *LUT* (*Look-Up Table*) contenues dans chaque slice. Ces *LUT* peuvent également être configurées comme éléments de mémoire synchrone, simple ou double-port de 16 bits, ou encore comme registres à décalage de 16 bits. Il existe donc trois modes de configuration de ces *LUT*. Plus précisément, le fonctionnement en mode combinatoire est obtenu en lisant le contenu pointé par les signaux d'entrée (Figure II.5a). Autrement dit, les *LUT* sont des mémoires dont le contenu est initialisé lors de la configuration du FPGA. De ce fait, elles permettent à l'utilisateur d'en disposer en mode « élément mémoire » dans chacun des slices si nécessaire (Figure II.5b). La Figure II.8c décrit le mode de configuration particulier en registre à décalage de longueur programmable jusqu'à 16 bits. Par ailleurs, une logique supplémentaire utile pour la réalisation des fonctions arithmétiques est disponible dans chaque slice. Grâce à ces éléments, et au style d'écriture adapté, des modules de type accumulateur chargeable en addition/soustraction pourront être implantés à raison de 2 bits par slice.

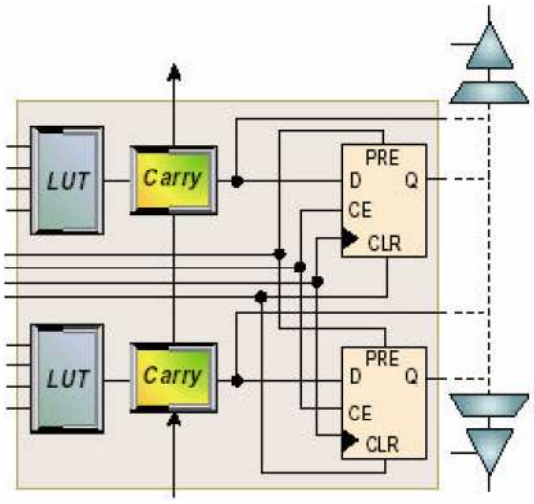


Figure II.5 : *Architecture simplifiée d'un « Slice »*

Les bascules dans chaque slice ont aussi des caractéristiques importantes pour le concepteur. En particulier, elles sont initialisées systématiquement à la mise sous tension (par défaut à valeur '0'), et sont utilisables indépendamment de la logique combinatoire disponible dans le même slice. En outre, chaque bascule bénéficie de broches de contrôle telles que : entrée dédiée de validation de l'horloge (*Clock Enable*) permettant d'activer ou de suspendre le fonctionnement de chacune des bascules individuellement, et ceci sans avoir à insérer de la

logique combinatoire sur le chemin de l'horloge (entrée de « *set* » et de « *reset* » synchrones ou asynchrones). La polarité des signaux d'horloge, de « *Clock Enable* », de *set* et *reset* est programmable pour chacune des bascules. Autrement dit, ces signaux peuvent être individuellement actifs au niveau haut ou au niveau bas.

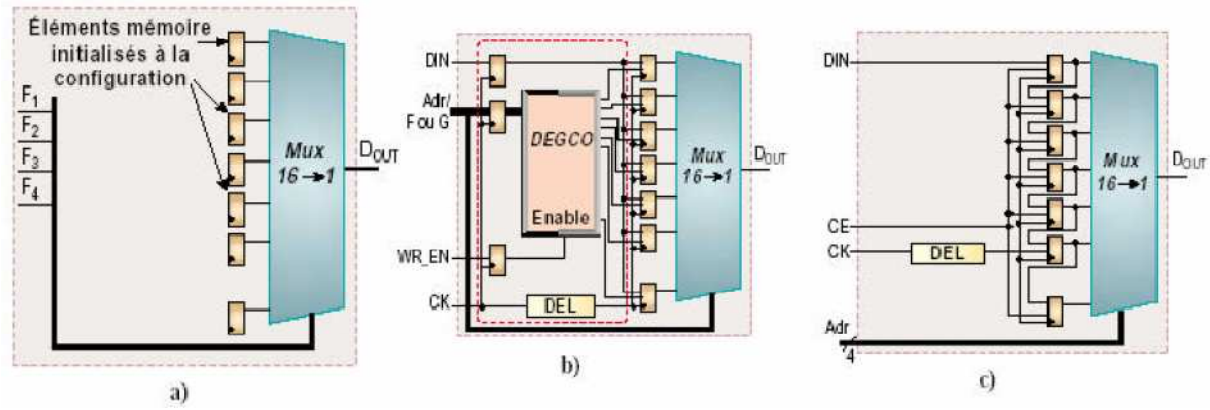


Figure II.6 : *Les modes de configuration des LUT*

L'échange des données avec les circuiteries externes (microprocesseur, *DSP*, mémoires rapides, convertisseurs A/N et N/A) est souvent un critère important dans la perspective des performances à atteindre. Les blocs d'entrées/sorties disposent de bascules sur les chemins d'entrées/sorties et de contrôle à trois-états. Les outils de synthèse le s plus sophistiqués permettent l'utilisation systématique de ces bascules.

Enfin, les *FPGA* de la famille *Virtex-II* intègrent des registres doubles sur chacun de leurs trois chemins, autorisant ainsi la réalisation d'interfaces *DDR* (*Double Data Rate*) pour la communication avec les mémoires *SDRAM* du même nom ou autres dispositifs de même type (Figure II.6).

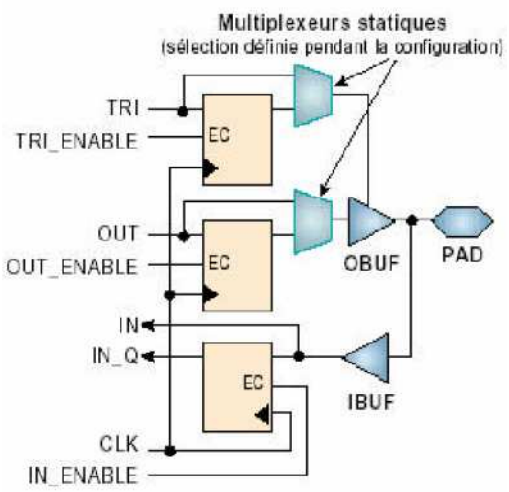


Figure. II.7 : *Configuration des blocs d'E/S*

La configuration électronique des blocs d'E/S donne la possibilité d'ajuster la raideur des fronts sur les étages de sortie, la sortance, les seuils de communication et les standards de communication (*LVTTL, LVCmos, SSTL, PCI, LVDS*…). Par ailleurs, la famille *Virtex-II* offre d'adapter en impédance aussi bien les entrées que les sorties, sans avoir besoin d'insérer les traditionnelles résistances d'adaptation.

II.1.3. La famille FPGA Xilinx Virtex-II

La famille *FPGA Xilinx Virtex-II* est constituée de onze composants dont les caractéristiques sont résumées par le *Tableau II.1*.

Device	System Gates	CLB (1CLB=4slices=Max128 bits)			Multipliers Blocks	Select RAM Blocks		DCMs	Max I/O pads
		Array Row X col	Slices	Maximum Distributed RAM Kbits		18Kbits Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1.536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3.072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5.120	160	40	40	720	8	432
XC2V1500	1.5M	48 x 40	7.860	240	48	48	864	8	528
XC2V2000	2M	56 x 48	10.752	336	56	56	1.008	8	624
XC2V3000	3M	64 x 56	14.336	448	96	96	1.728	12	720
XC2V4000	4M	80 x 72	23.040	720	120	120	2.160	12	912
XC2V6000	6M	96 x 88	33.792	1.056	144	144	2.592	12	1.104
XC2V8000	8M	112 x 104	46.592	1.456	168	168	3.024	12	1.108

Tableau II.1 : *Famille FPGA Virtex-II*

Les blocs « *SelectRam* » dans l'architecture *Virtex-II* sont des mémoires Ram double port de 18Kbits, configurables de différentes manières entre 18x1Kbit et 512x36 bits. Chaque port est totalement synchrone et indépendant. Les blocs « *Selectram* » peuvent être mis en cascade pour réaliser des larges zones des stockages enfouies dans le *FPGA*. Un module multiplieur 18x18 bits est disposé à proximité de chaque bloc « *SelectRam* » et optimisé pour opérer sur le contenu d'un port de la mémoire.

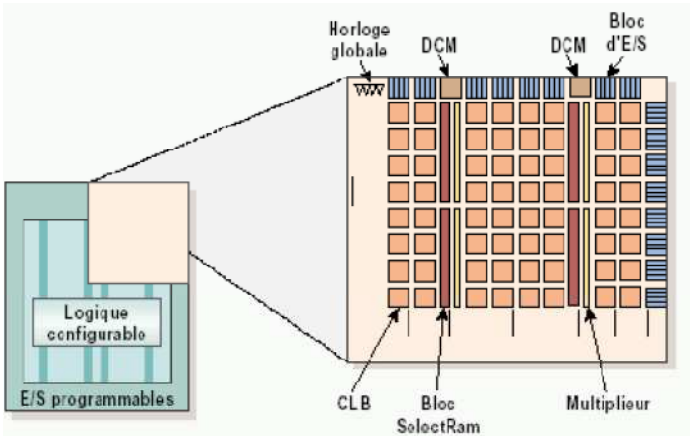


Figure II.8 : *Détail de l'architecture d'un composant Virtex-II*

Les fonctions mémoire « *SelectRam* » et les multiplieurs sont connectés aux matrices d'interconnexions, cette configuration leur ouvrant l'accès aux ressources de routage globales de la matrice *FPGA*. La *Figure II.8* présente en détail l'architecture d'un composant *Virtex-II*. Dans cette famille de *Virtex-II*, on s'intéresse particulièrement au composant *XC2V6000*, puisque c'est le composant supporté par la plateforme *ADM-XRC-II*.

II.1.4. FPGA Virtex-II Pro

Xilinx a mis depuis peu sur le marché, une nouvelle technologie de circuits *FPGA* intégrants de nouvelles fonctionnalités, ces derniers permettent la réalisation d'architectures qui auparavant s'avéraient très compliquées à entreprendre. Parmi ces nouveaux circuits, le *VIRTEX-II PRO (Figure II.9)* qui est un composant *FPGA* intégrant un cœur de processeur PowerPC 405 de chez IBM pouvant fonctionner à plus de 300MHz. Afin de rendre le co-design plus simple à réaliser, la famille de composant *Virtex-II-Pro* possède plusieurs atouts tel que :

- L'intégration d'un ou deux cœurs de processeur *powerPC405 ;*
- Un grand nombre de slices pouvant atteindre 44.096 unités permet d'implémenter des algorithmes dont la complexité les rend gourmand en consommation des ressources logiques ;
- Des blocs de mémoire de 18Kb sont aussi disponible ce qui diminuera l'accès vers des ressources externes et ce qui peut être fatal pour le circuit surtout du point de vue consommation d'énergie ;
- Des multiplieurs 18x18 bits, des *DCM* (*Digital Clock Manager*).

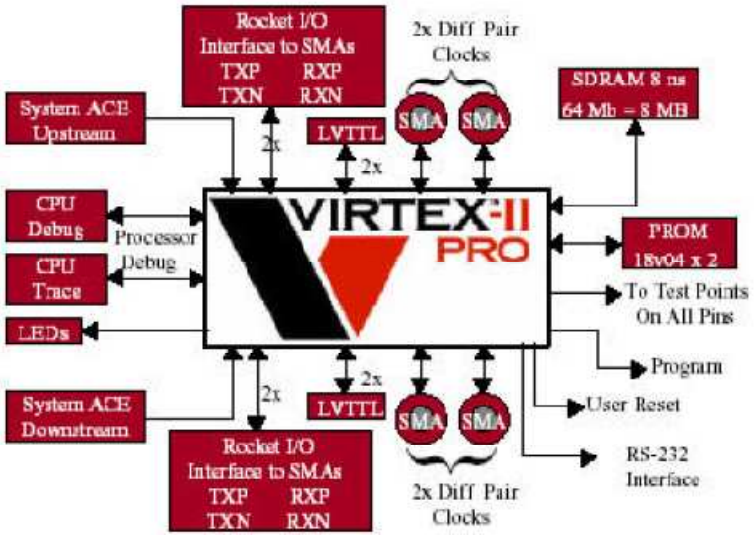


Figure II.9 : *Vue Globale du VIRTEX-II PRO*

Pour cela *Xilinx* fourni des IPs soft (Figure II.9) écrites en langage *VHDL* qui permettent d'exploiter le PowerPC405 et de lui fournir tous les périphériques nécessaires à son fonctionnement. Les premières IPs à implémenter autour du processeur sont le *PLB* « *Processor Local Bus* » et le « *On Chip Memory Controller* », la première constitue le bus de communication sur lequel viennent se greffer d'autres IPs, quant au «*On Chip Memory Controller* » il permet de connecter le PowerPC405 avec les Bram's du *FPGA* ou bien avec des RAMs externes où sera logé le code à exécuter.

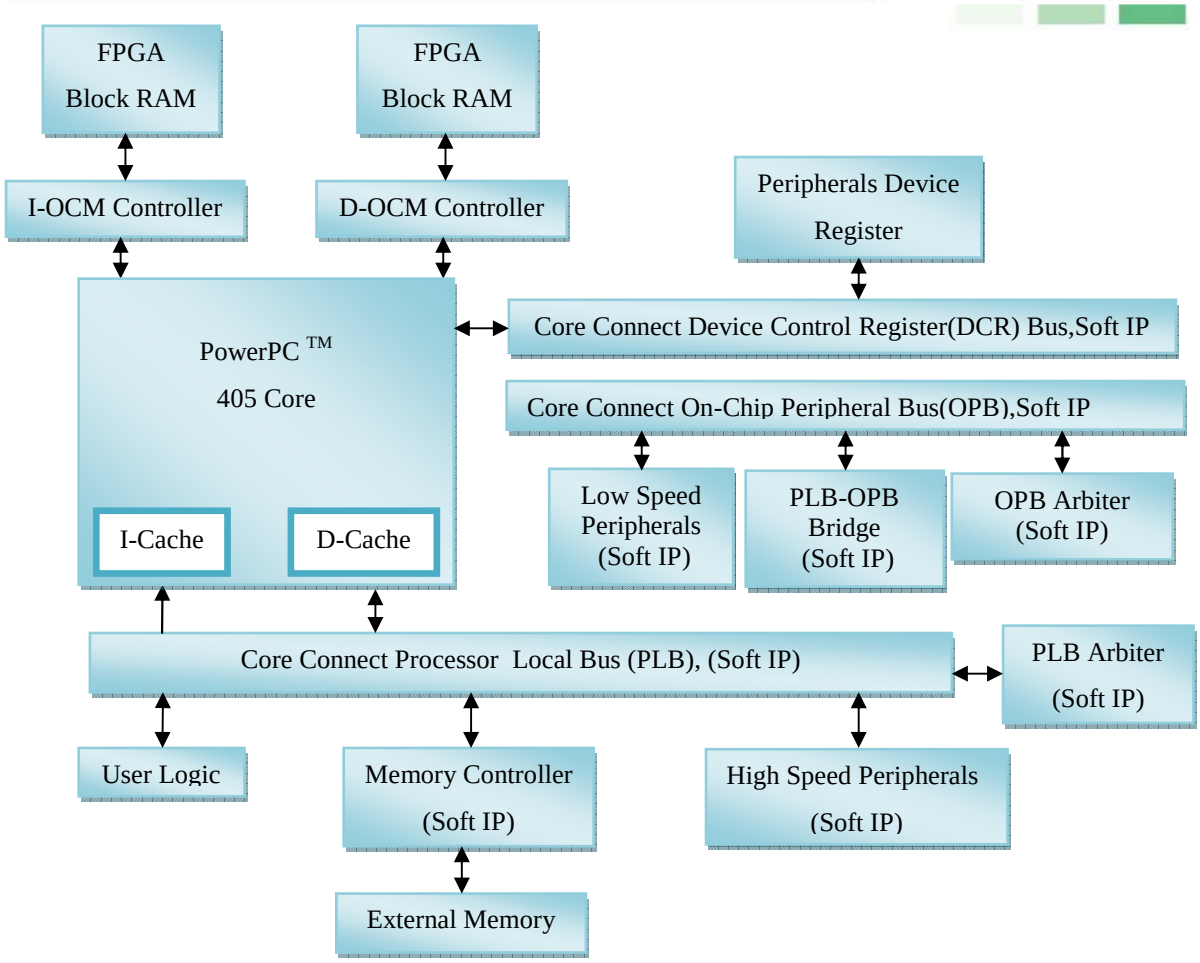


Figure II .10 : *Architecture standard autour du PowerPc405 sur VIRTEX-II PRO*

Une application peut se limiter à une configuration avec un PowerPC405, une IP « *Processor Local Bus* » et un « *On Chip Memory Controller* », l'utilisateur peut ajouter des bus ou bien des contrôleurs de mémoire externe ou autres IPs, selon les exigences de son architecture.

Device	RoketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells	CLB		18X18 Multi Blocks	Select RAM Blocks		DCMs	Max I/O pads
				Slices	Max Dist RAM		18KB Blocks	Max RAM (KB)		
XC2VP2	4	0	3.168	1.408	44	12	12	216	4	204
XC2VP4	4	1	6.768	3.008	94	28	28	504	4	348
XC2VP7	8	1	11.088	4.928	154	44	44	792	4	396
XC2VP20	8	2	20.880	9.280	290	88	88	1.584	8	564
XC2VPX20	8	1	22.032	9.792	306	88	88	1.584	8	552
XC2VP30	8	2	30.816	13.696	428	136	136	2.448	8	644
XC2VP40	0, 8, or 12	2	43.632	19.392	606	192	192	3.456	8	804
XC2VP50	0 or 16	2	53.136	23.616	738	232	232	4.176	8	852
XC2VP70	16 or 20	2	74.448	33.088	1.034	328	328	5.904	8	996
XC2VPX70	20	2	74.448	33.088	1.034	308	308	5.544	8	992
XC2VP100	0 or 20	2	99.216	44.096	1.378	444	444	7.992	12	1.164

Tableau II.2 : *Ressources du circuit VIRTEX-II PRO/ VIRTEX-II PRO X*

La famille *VIRTEX-II PRO* dispose d'un nombre suffisant de ressources permettant la réalisation d'applications « *System On Chip* » et ce avec une électronique associée très réduite. Le *Tableau II.2* donne un récapitulatif des ressources dont dispose le composant de la famille *VIRTEX-II PRO*.

II.2. Flot de conception FPGA :

Xilinx fournit dans son pack d'outils différents soft permettant la création de systèmes embarqués sur puce, parmi ces softs on dénombre *ISE* (*Integrated Software Environment*) et *EDK* (*Embedded Development Kit*), tous deux nous offrent la possibilité d'avoir un bitstream pour la programmation des *FPGA* suivant l'application ciblée.

II.2.1.Vue Générale :

Le flot de conception standard est composé de plusieurs étapes qui peut se résumer en :

- Conception et synthèse du design ;
- L'implémentation et la vérification du design .

Une vue générale du flot de conception est présentée par la figure suivante :

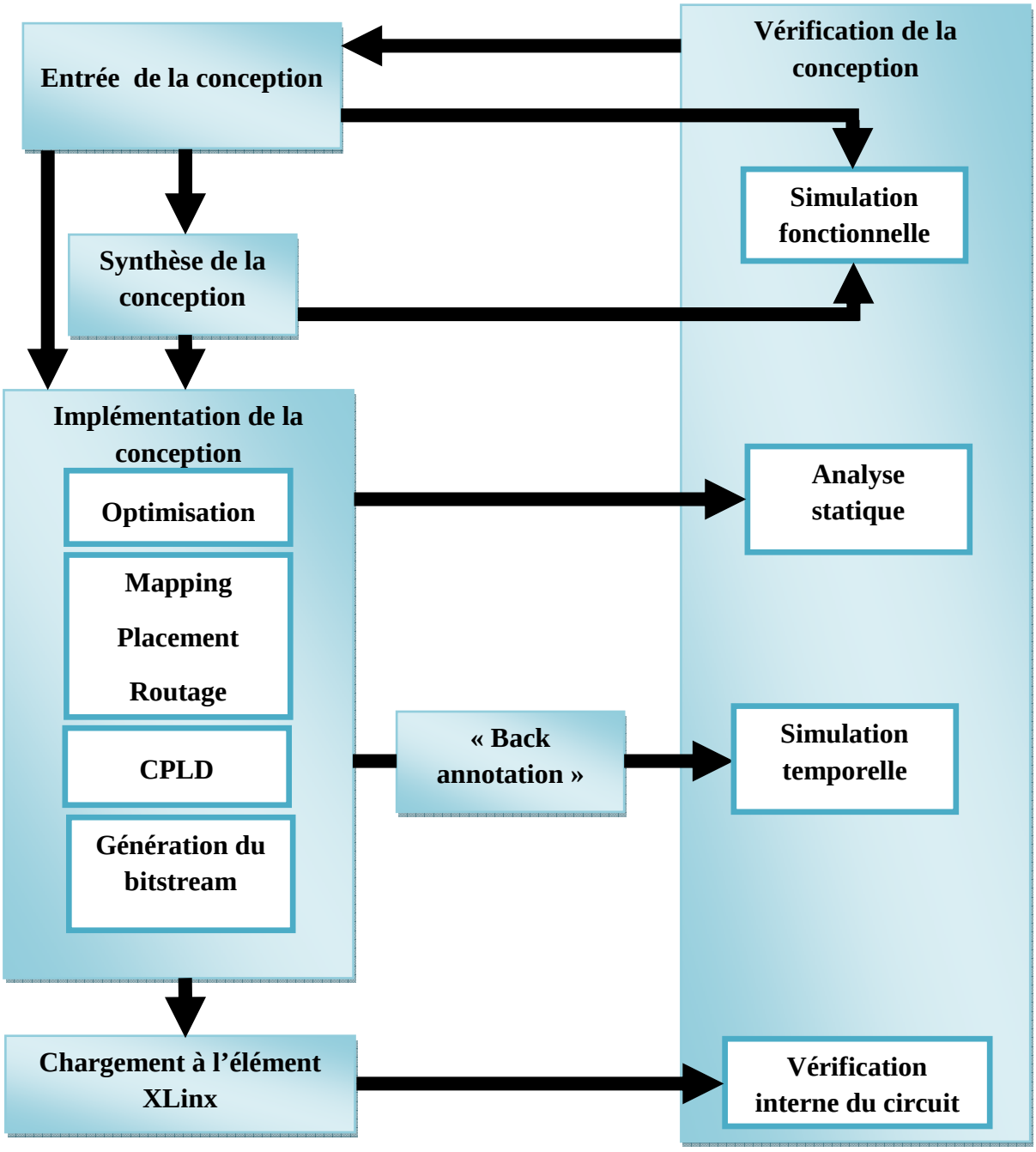


Fig.II.11 : *Flot de conception générique*

II.2.2.Les différents étages du flot de la conception :

II.2.2.1. La synthèse :

Un design peut être conçu à l'aide d'un éditeur schématique ou d'un outil de traitement de textes. La conception commence par un concept exprimé par un schéma ou une description fonctionnelle. A partir du design original, une *netlist* est créée, synthétisée puis translatée en fichier *NGO* (*Native Generic Object*). Ce fichier est utilisé ensuite par un logiciel appelé *NGDBuild* qui produira alors un fichier *NGD* (*Native Generic Database*).

- **Design Hiérarchique**

Une hiérarchie du design est importante dans les deux modes de conception : Schématique et HDL. Et ceci pour plusieurs raisons :

- Pour avoir un design bien structuré et facile à déboguer

-Combinaison de plusieurs types de conceptions (Schématiques, *VHDL*, *Verilog*, etc.)

-Design incrémental qui consiste à concevoir, implémenter et vérifier des parties de système individuellement.

-Réduit le temps d'optimisation et facilite les designs concurrents par une répartition du système en parties qui seront développées simultanément par plusieurs personnes (comme de design modulaire).

- **Schématiques**

Les outils schématiques produisent des interfaces graphiques pour décrire le design. Ces outils peuvent être utilisés pour connecter des symboles représentants des instances des composants utilisés dans le design. Le design peut être construit à l'aide des portes individuelles, ou à l'aide de blocks fonctionnels en utilisant des éléments de librairies et des outils comme *Core Generator* et *LogiBLOX*.

- **Les éléments des Librairies**

Les primitives et les macros sont les constituants fondamentaux des librairies de composants. Les librairies de *Xilinx* offrent des primitives et des macros de haut niveau. Une primitive est une partie élémentaire du circuit (portes logiques, Flip-Flops, etc.). Chaque primitive a des propriétés caractéristiques (nom de librairie, symbole et description). Un macro contient plusieurs éléments de librairies qui peuvent être primitives ou des macros. Deux types de macros sont disponibles à l'utilisation avec les FPGA de Xilinx :

-Les *soft-macros* dont les fonctionnalités sont prédéfinies, mais qui sont flexibles du point de vue *mapping*, placement et routage.

-Les RPMs (*Relationnaly Placed Macros*) qui possèdent un *mapping* et un placement relatif fixés.

Les macros ne sont pas disponibles pour la synthèse parce que les outils de synthèses possèdent leurs propres générateurs de modules et ne requièrent pas de RPMs.

- **Outil de Génération de Cores**

L'outil de conception *CORE Generator* de Xilinx offre des cores paramétrables qui sont optimisés pour les *FPGAs* de *Xilinx*. La librairie contient des éléments qui varient entre de simples registres à des applications très complexes (filtres et multiplexeurs *DSP*).

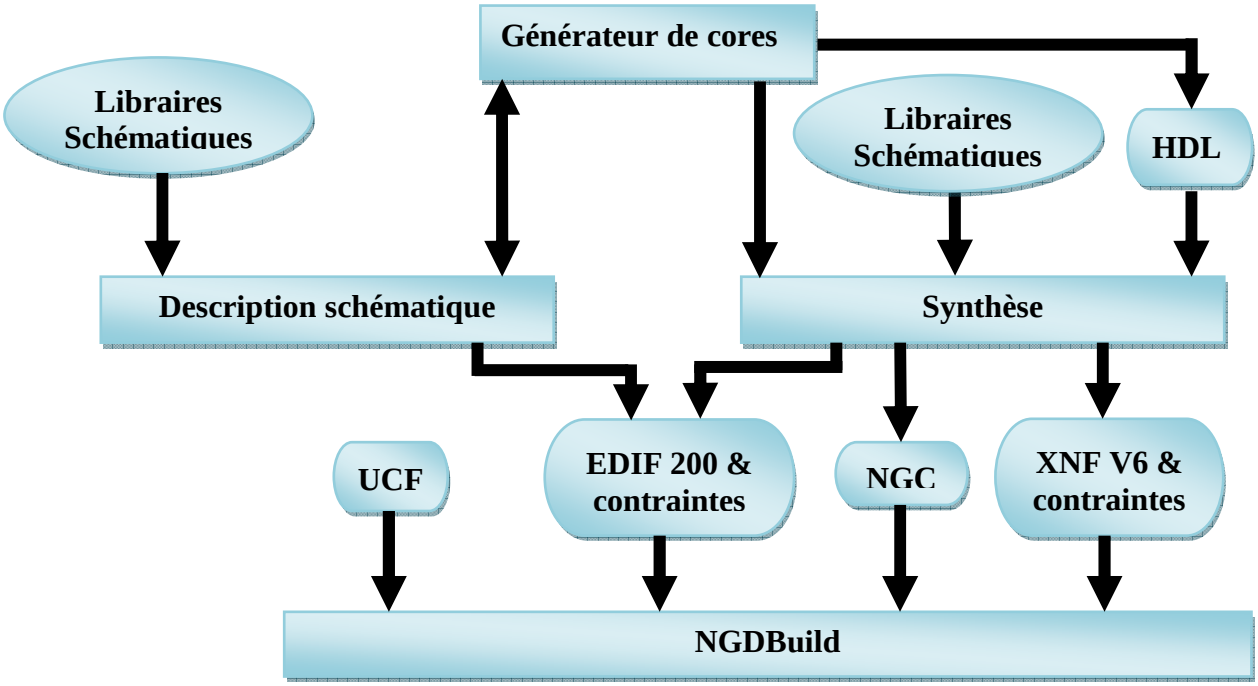


Figure II.12 : *Flot de synthèse sous Xilinx*

II.2.2.2. Simulation fonctionnelle :

Après sa conception et sa description, la simulation fonctionnelle teste la logique du design pour détecter et corriger tout dysfonctionnement dû à une mauvaise conception ou à une description erronée. Le logiciel utilisé pour la simulation fonctionnelle est le *ModelSim*.

II.2.2.3.Translation en NetList

C'est le format requis par les outils de développement de Xilinx pour compléter le flot de conception est *NGD* (*Native Generic Database*). Deux outils sont disponibles pour effectuer la translation d'autres formats en format NGD. Par exemple *EDIF2NGD* traduit un fichier *EDIF* (*Electronic Data Interchange Format*) en *NGD* et le programme *NGDBuild*

convertie plusieurs formats (*XNF*, ou *EDIF*) en *NGD*. A noter que ce dernier outil, utilise le premier pour la traduction des fichiers *EDIF*.

II.2.2.4. Implémentation

L'implémentation du design commence avec le *mapping* qui se traduit par l'association des éléments logiques avec les éléments physiques d'un composant donné, et se termine lorsque le design physique est routé totalement et traduit en série de bits chargeable dans le composant.

Des contraintes peuvent toujours être ajoutées durant le processus d'implémentation pour que le design satisfasse le cahier de charge. Le processus d'implémentation est illustré par la *Figure II.13*.

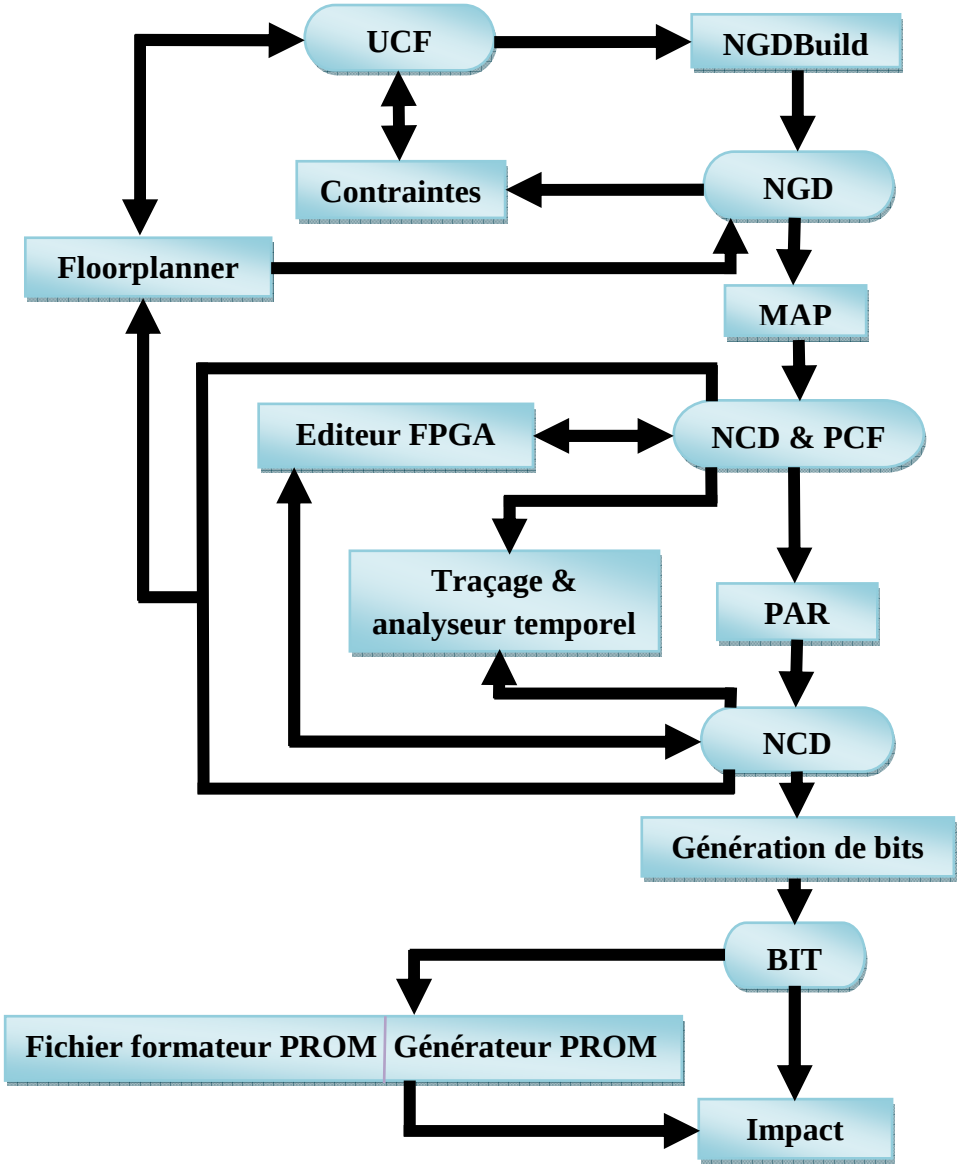


Figure II.13 : *Implémentation sous Xilinx*

II.2.2.5. Mapping :

L'outil *MAP* arrange les éléments logiques d'un design dans un *FPGA* de *Xilinx* et les associe à des éléments physiques. L'outil *MAP* prend en entrée un fichier *NGD* qui contient une description logique du design en terme de composants hiérarchiques utilisés dans le design et de primitives de Xilinx de bas niveau, et d'un nombre quelconque de fichiers *NMC* (*hard placed and routed macros*), chacun contenant la définition d'un macro physique. *MAP* associe alors des composants (cellules logiques, cellules d'entrée/sortie, etc) aux différentes parties logiques du design. La sortie du *MAP* est un fichier de description de circuit d'extension *NCD* (*Native Circuit Description*). Ce fichier est une représentation physique du design à l'aide des éléments constitutifs du *FPGA* cible. Le fichier *NCD* peut être placé et routé.

II.2.2.4. Placement et Routage

L'outil *PAR* (*Place And Route*) prend à l'entrée une description physique du design (un fichier *NCD*), le place et le route pour donner un autre fichier *NCD*, utilisable par *BitGen*, afin qu'il soit transformé en données de configuration. Le fichier *NCD* de sortie peut être aussi utilisé comme fichier guide pour refaire le placement routage après avoir effectuer de petits changements au design. L'outil *FPGA editor* permet de placer et router des composants critiques avant le lancement du placeur/routeur automatique, et permet aussi de modifier manuellement le placement et le routage.

II.2.2.5. Génération de flot de bits

L'outil *BitGen* produit une série de bits pour la configuration d'un *FPGA* de *Xilinx*. *BitGen* prend un fichier *NCD* totalement placé et routé pour générer la série de bits de configuration sous la forme d'un fichier d'extension *.bit*. Le fichier *BIT* contient toutes les informations contenues dans le fichier *NCD* (la logique interne et les interconnections entre les éléments du *FPGA*), et des informations caractéristiques du composant cible à partir d'autres fichiers associés avec le composant cible. Après la génération du fichier *BIT*, il peut être chargé dans le *FPGA* à l'aide de l'outil *iMPACT*. Un fichier *PROM* peut aussi être créé à partir du fichier *BIT*, dans le but de le charger dans une *PROM*, affin d'être utilisé ultérieurement par le *FPGA*.

II.2.2.6. Vérification

La vérification du design est un processus de test des fonctionnalités et des performances du design. *Xilinx* offre plusieurs méthodes pour ce but :

- La simulation fonctionnelle et temporelle ;

- Une analyse temporelle statique ;
- la vérification sur le circuit.

La procédure de vérification du design se fait normalement comme l'illustre la figure :

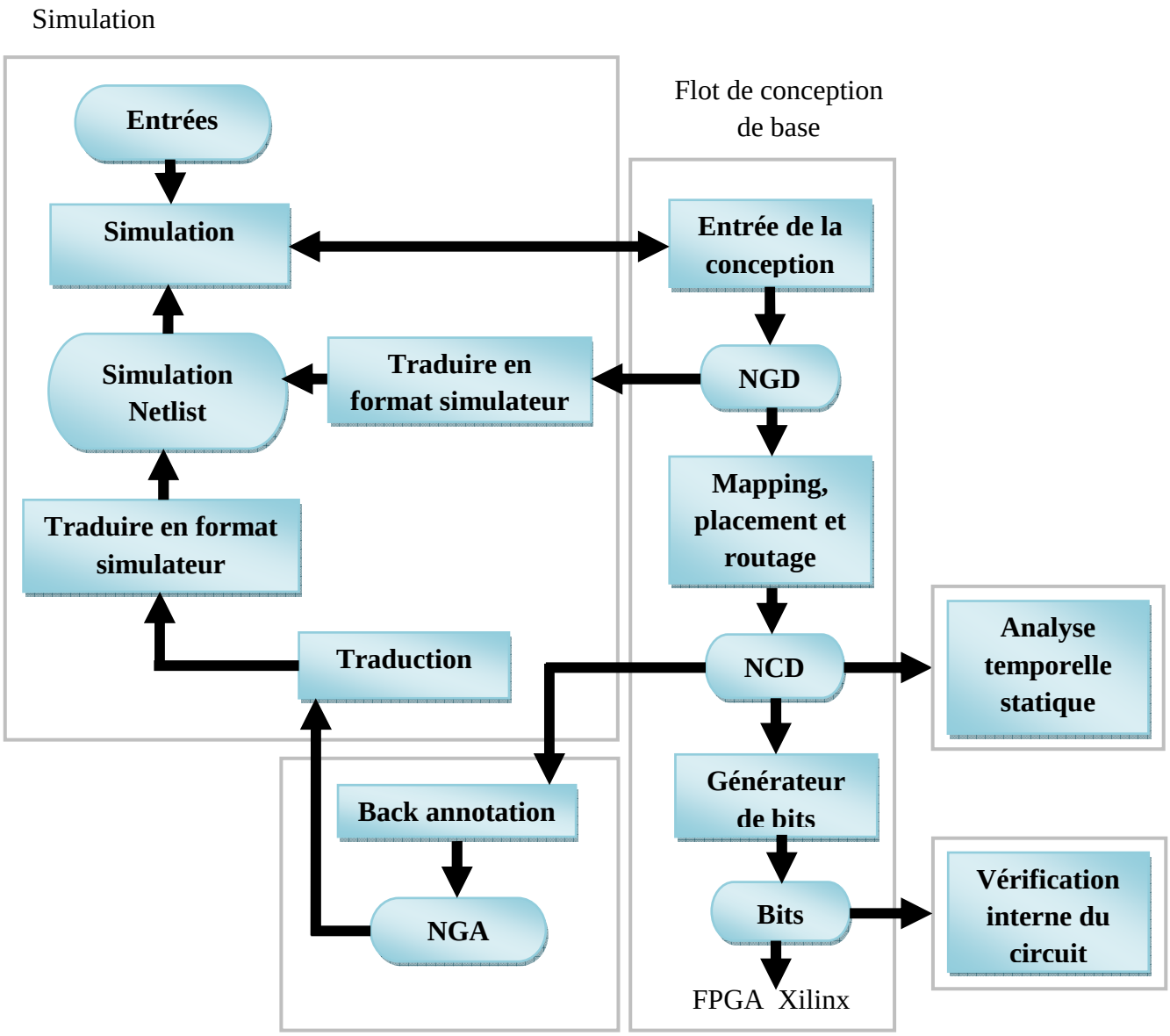


Figure II.14 : *Le processus de vérification sous Xilinx*

II.2.2.7. Le processus « *Back annotation* »

Pour vérifier le design, une simulation fonctionnelle ou temporelle peut être exécutée. Un processus de *Back-Annotation* doit avoir lieu avant la simulation temporelle. Avant la phase de la simulation temporelle, la description physique du design doit être traduite en design logique compréhensible par le simulateur. Cette tâche est appelée *Back-Annotation process* en anglais. Un outil appelé *NGDAnno* est chargé de créer une base de données afin qu'elle soit utilisée par l'outil créateur de *netlist*, qui traduit ces informations en format *netlist* compréhensible par le simulateur. La figure suivante illustre le *Back-Annotation process*.

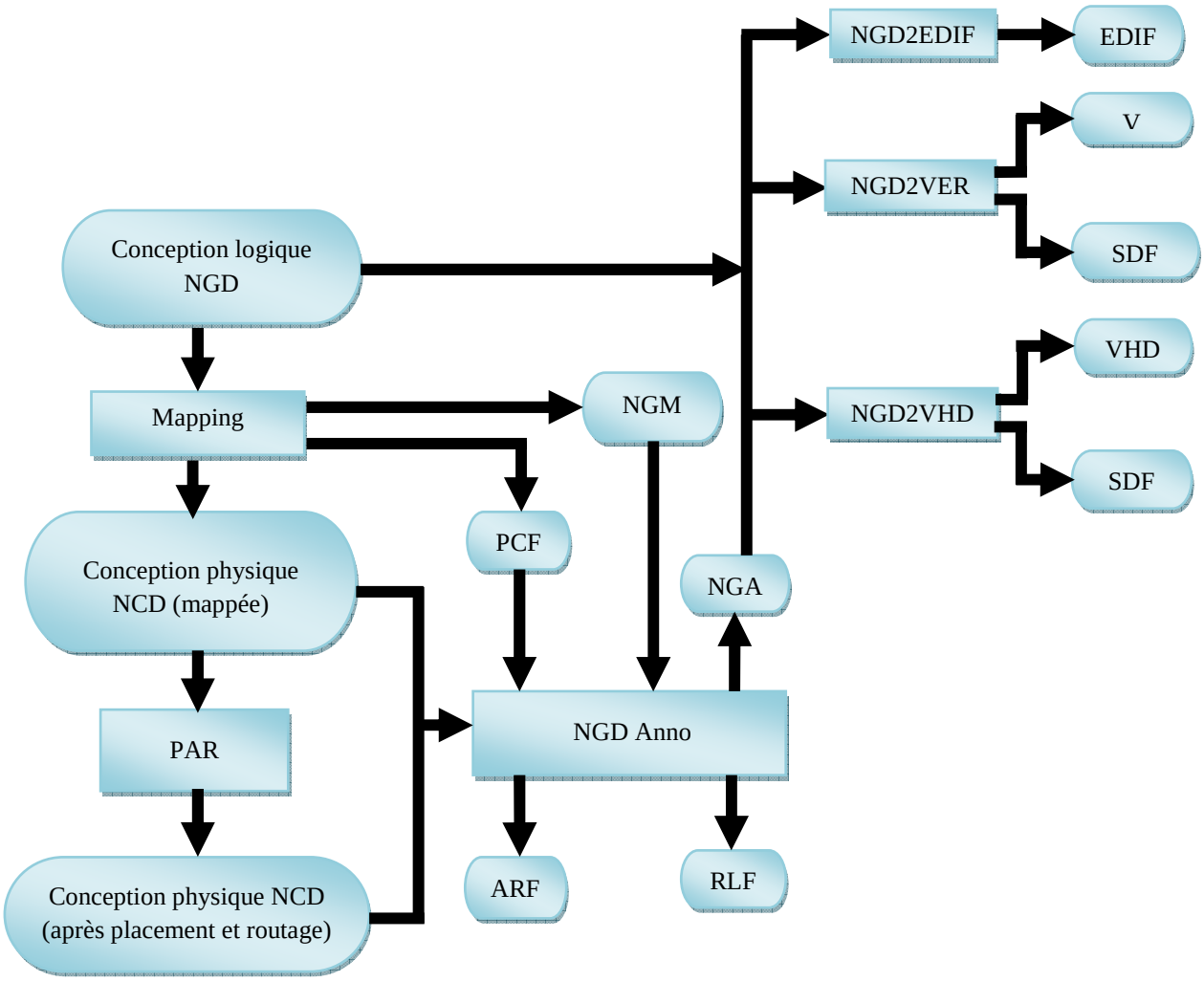


Figure II.15 : *Le processus « Back-Annotation »*

NGDAnno est un outil qui traduit les caractéristiques physiques (délais, temps de propagation des signaux, temps de réponses, etc) trouvées dans un fichier de description de circuit *NCD*, en fichier de description logique du design *NGD* (en tenant compte de toutes les caractéristiques physiques du circuit). Le fichier *NCD* à l'entrée peut être un design totalement ou partiellement placé et routé, ou qui n'est ni placé ni routé (*NCD* à la sortie de l'outil *MAP*). Un fichier *NGM* créé aussi par *MAP* est une entrée optionnelle pour *NGDAnno* qui fusionne les informations de *mapping* contenues dans ce dernier fichier avec les informations de placement, routage et les informations temporelles provenant du fichier *NCD* employé. A la sortie, cet outil génère un fichier *NGA (Native Generic Annotation*) qui est un fichier *NGD* reconstruit à partir d'un *NCD*. Ce fichier constituera une entrée pour l'outil de génération de netlist approprié, qui convertira le design du format NGA binaire, en format netlist *ASCII*. Les *netlist writers* (*NGD2EDIF*, *NGD2VER* ou *NGD2VHDL*) prennent la sortie de l'outil *NGDAnno* et créent une *netlist* de simulation au format spécifié. Un fichier *NGD* ou NGA peut être une entrée d'un *netlist writer*. *NGD2EDIF* traduit un fichier *NGD* ou *NGA* en *netlist* au format *EDIF* (fichier *EDN*). *NGD2VER* traduit un fichier *NGD* ou *NGA* en netlist au format *Verilog* (fichier *V*). Cet outil génère aussi un autre fichier SDF (*Standard Delay Format*) qui contient des informations temporelles qui est utilisable uniquement avec le fichier Verilog créé par le même outil, à partir du même fichier *NGD* ou *NGA*. *NGD2VHDL* traduit un fichier *NGD* ou *NGA* en *netlist VHDL* (fichier *VHD*). Si le fichier d'entrée est du format *NGA*, cet outil génère aussi un fichier SDF spécifique pour ce fichier *VHD*.

II.2.2.8. Simulation fonctionnelle

La simulation fonctionnelle ou comportementale détermine si la logique du design est correcte avant la phase d'implémentation. Ce type de simulation peut avoir lieu tôt dans le processus de conception du système. Et puisque les informations temporelles ne sont pas disponibles à ce moment, le simulateur teste la logique en utilisant des délais élémentaires comme unités. Le simulateur utilisé (*ModelSim*) est un simulateur intégré dans l'environnement de développement de *Xilinx* : le passage entre les outils de conceptions (éditeurs *HDL* ou schématiques) et le simulateur se fait automatiquement sans besoin d'utilisation intermédiaire d'outils de translation.

II.2.2.9. Simulation temporelle

La simulation temporelle examine le temps d'exécution du design dans les pires conditions. Ce processus peut avoir lieu après le *mapping*, le placement et le routage du design. A ce moment là, tous les délais du design sont bien connus. La simulation temporelle est très importante parce qu'elle peut vérifier les relations temporelles et détermine les chemins critiques du design dans les pires conditions. Avant la simulation temporelle, il faut passer par le *Back-Annotation process* déjà mentionné.

II.2.2.10. Analyse temporelle statique

L'analyse temporelle statique est favorisée dans le cas de vérification rapide d'un design après placement et routage. Elle permet de déterminer les délais des chemins du design. Les deux buts majeurs de l'analyse temporelle sont : la vérification temporelle (vérifier que le design obéit aux contraintes temporelles) et le rapportage (décrire le comportement temporel d'une façon technique qui peut être utilisée comme documentation pour évaluer le design). L'outil *TRACE* (*Timing Reporter And Circuit Evaluator*) est responsable de faire cette analyse.

II.2.2.11. Vérification sur le circuit

La vérification du comportement de design dans l'application cible est considérée comme un test final. La vérification du design sur le circuit test, circuit dans les conditions typiques de fonctionnement. Ce type de test est simple du fait que les composants de *Xilinx* sont reprogrammables (plusieurs itérations du design peuvent être essayées).

Avant la création du fichier *BIT* final, il est utile d'utiliser l'option *DRC* dans l'outil *BitGen* pour évaluer le fichier *NCD*, et chercher les problèmes qui peuvent empêcher le design de fonctionner correctement sur le composant cible.

II.3. Architecture d'un Circuit FPGA

Une structure *FPGA* est composée généralement de deux blocs. Des sous blocs d'entrées/sorties et des sous blocs logiques programmables. Quelques circuits *FPGA* possèdent aussi des mémoires *RAM* et des noyaux de processeurs.

De nombreux constructeurs élaborent des structures *FPGA* dont la technologie diffère d'un constructeur à un autre. Nous nous intéresserons dans ce qui suivra à des structures

FPGA développées par la société *XILINX*. Une structure *FPGA* construite par *XILINX* se présente sous forme de deux couches :

II.3.1. Le circuit configurable :

Elle est constituée de blocs logiques configurables (*CLB*), elle contient des fonctions combinatoires et des fonctions séquentielles. Autour de ces circuits configurables existent des blocs d'entrées/sorties (*IOB*) qui ont pour rôle la gestion des entrées et des sorties réalisant ainsi une interface entre le circuit et les modules extérieurs. La figure suivante illustre ces différents blocs.

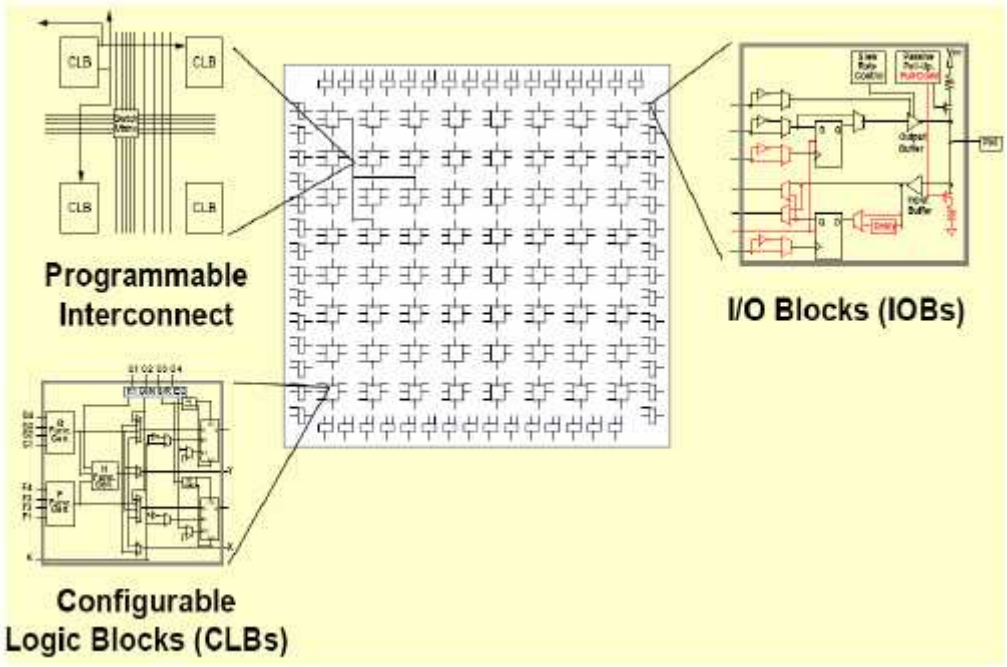


Figure II.16 : *Architecture interne du FPGA.*

II.3.2. La couche réseau mémoire :

C'est dans cette partie que la configuration du circuit est mémorisée, on déduit facilement que la programmation d'un circuit *FPGA* est volatile puisqu'elle s'enregistre sur une *SRAM*, mais le contenu est stocké tout de même sur une *ROM* externe. Le processus est simple ; à chaque mise sous tension, la *SRAM* interne est chargée à partir de la *ROM* externe.

Cette caractéristique fait que le contenu interne n'est jamais définitif, ainsi un même circuit peut être exploité avec des *ROM* différentes, de ce fait une erreur quelconque en phase de programmation est facilement corrigeable.

II.4. Le module ADM-XRC-II

II.4.1. Présentation de la plateforme ADM-XRC-II

La plateforme *ADM-XRC-II [43]* utilise un *FPGA* de type *Xilinx Virtex-II XC2V6000* (Tableau II.2). Ce type de composant est très utile pour les applications de télécommunications car en plus de sa grande densité d'intégration, sa faible consommation et sa haute fréquence de travail, le *Virtex-II* possède des entrées/sorties compatibles pour les interfaces *LVDS* et le *PCI*. Le *FPGA Virtex-II XC2V6000* employé par la plateforme est un boîtier de type *FF1152* (boîtier BGA de 35mm x 35mm avec 1152 broches).

Les caractéristiques techniques de ce composant (Tableau II.2) sont les suivantes :

- Nombre de portes : 6 Millions
- Horloge interne maximale : 420 MHz
- Alimentation du cœur VCCint = 1,5V
- 824 entrées/sorties
- Il possède 16 multiplexeurs d'horloge interne.

La particularité de ce type de technologie (famille *FPGA Virtex-II*) est d'accepter la reconfiguration dynamique partielle. C'est-à-dire qu'en cours d'exécution d'une application sur le *FPGA*, nous pouvons reconfigurer une ou plusieurs zone sans devoir reconfigurer entièrement le *FPGA*. Dans ce cas, nous configurons le *FPGA* avec une IP de processeur possédant des paramètres déterminés, ensuite au cours de l'exécution nous modifions une partie de ce processeur avec des paramètres différents. Ceci permet un gain de temps au niveau de la phase de la synthèse de design (re-synthétiser et configurer uniquement la partie concernée sur l'*FPGA*).

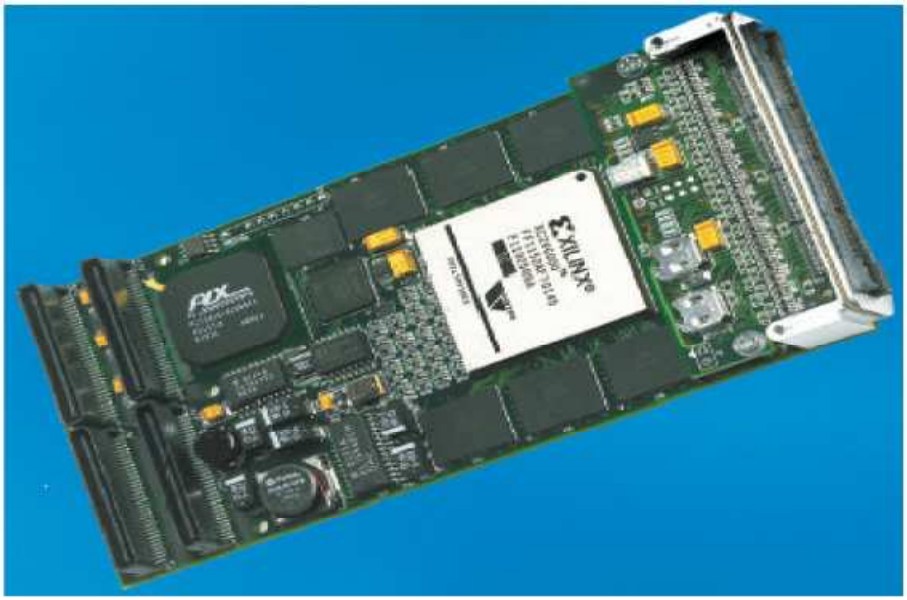


Figure II.17 : *Plateforme de prototypage ADM-XRC-II (Alpha Data)*

D'autre part, le module *ADM-XRC-II* (Figure II.17) intègre une interface *PCI* (à travers un connecteur *PMC*), permettant à un *PC*, de reconfigurer et/ou communiquer avec le *FPGA*. Avec ce module, sont fournis tous les drivers et bibliothèques, permettant au *PC* de communiquer avec celui-ci sous les environnements *Linux*, *Windows* ...

II.4.2. Spécifications techniques du module ADM-XRC-II

Le module *ADM-XRC-II* supporte les hautes performances du bus *PCI* sans avoir à utiliser une *IP PCI* dans le *FPGA*. Une interface *PCI Hard PLX 9656 [46]* gère les communications entre le *FPGA* et le *PC*. A travers cette interface *PCI*, gère aussi les commandes du module tel que l'horloge programmable.

Les caractéristiques techniques sont les suivantes :

- Bus *PCI* 64bits – 66 MHz
- 6 banques indépendantes de *ZBT SSRAM* 256K x 32bits
- 64 entrées/sorties via un connecteur *PMC*.
- 146 entrées/sorties via un connecteur *XRM*.
- Sur la carte, une horloge programmable fournit au bus local (entre *FPGA* et interface *PCI*) une fréquence variable : 400kHz à 66 MHz.
- Une autre horloge programmable fournit une fréquence au FPGA : 0 à 100 MHz.
- Possibilité de connecter 256 MO de *DDR SDRAM* via le connecteur *XRM*.

II.5. Le langage VHDL :

Le développement colossal qu'avait connu l'électronique a fait que la description d'un circuit programmable a connu un saut dans le temps, il fut un temps, en effet, les techniques utilisées pour traduire une structure en un circuit électrique étaient des outils de saisie de shémas, ou des langages de bas niveaux (*ABEL, PALASM,ORCAD/PLD,…*). Actuellemnt, on ne peut plus parler de ces outils simples pour une simple raison ; les circuits d'aujourdh'ui ne sont plus ceux d'autre fois ; ils sont plus compliqués et beaucoup plus denses. C'est à partir de la que le besoin de langages de hauts niveaux a eu lieu à savoir le *VHDL* et le *VERILOG*.

Le programme *VHSIC* (*Very High Speed Integrated Circuits*), développé par le département de la défense des Etats-Unis dans les années 1970-1980 a donné naissance au langage *VHSIC-HDL* (*VHSIC Hardware Description Language*) ou *VHDL*. En 1987, la première version du langage a été standarisée par l'*IEEE* (*Institue of Electrical and Electronics Engineers*) portant ainsi le nom *IEEE Std.* 1076-1987 (*VHDL 87*), en 1993 quelques ambaiguités ont été supprimées et de nouvelles commandes ont été mises en place dans la version (*IEEE Std. 1076-1993 ou VHDL-93*), en 1997, de nouvelles fonctions pour la conception des circuitsont été ajoutées. La dernière version du *VHDL* est celle de 2002 nommée (*IEEE Std. 1076-2002 ou VHDL-2002).*

Un langage de description Hardware (*HDL*) offre donc la possibilité de description à plusieurs niveaux dans laquelle les portes élémentaires et les *netlists* peuvent être utilisés avec une description fonctionnelle. Cette variation de niveau permet de décrire l'architecture du système à un niveau d'abstraction plus élevée, puis d'une façon incrémentale raffiner l'implémentation du design jusqu'au niveau des portes.

La description HDL offre les avantages suivants : la fonctionnalité du design peut être vérifiée immédiatement ce qui permet d'évaluer les décisions architecturales, une description HDL est lue et comprise plus facilement qu'une *netlist* ou une description schématique et elle constitue une documentation du design et de sa fonctionnalité indépendante de la technologie, et sans oublier qu'une description *HDL* facilite la manipulation de larges designs par rapport à une description schématique.

II.5.1. Structure d'une description VHDL :

Une structure *VHDL* comporte généralement deux parties indissociables à savoir l'entité (*ENTITY*) et l'archietécture (*ARCHITECTURE*). La première est définie comme un bloc contenant des entrées et des sorties ; c'est une vue externe de la description. L'archietecture contient les instructions écrites en *VHDL* ayant pour but la réalisation du fonctionnement attendu ; c'est la description interne de la structure.

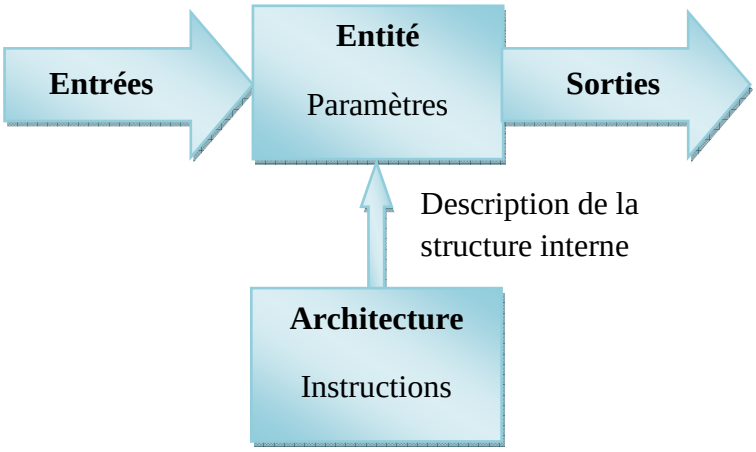


Figure II.18 : *Structure de base d'une description VHDL*

II.5.1. 1. Déclaration des bibliothèques :

Les bibliothèques sont un élément indispensable pour toute decription VHDL. Elle contiennnent les définitions des signaux électroniques, des fonctions et sous programmes à utiliser pour réaliser les opérations arithmétiques et logiques. La commande ***use*** permet de désigner les bibliothèques à utiliser.

Exemple :

```
library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;
Use ieee.std_logic_unsigned.all;
```

II.5.1. 2. Déclaration de l'entité et des entrées/sorties :

C'est une opération qui permet de définir le nom de la description et les entrées et sorties utilisées, l'instruction utilisée est ***prot***.

Exemple:

```
entity essai is

   Port ( a : in  std_logic;
          b : in  std_logic;
          y : out std_logic
                      );
end essai;
```

Un signal peut être défini en entrée (***in***), en sortie (***out***), en entrée/sortie (***inout***) ou en sortie pouvant être utilisé en entrée à l'intérieur de la description (***buffer***). Le langage VHDL ne fait pas la distinction entre la majuscule et la minuscule.

Un signal peut être :

- ***std_logic*** pour un signal à un bit ;
- ***std_logic_vector*** pour un registre composé de plusieurs bits.

II.5.1. 3. Déclaration de l'archietecture correspondante à l'entité :

L'architecture définit le fonctionnement tracé pour un circuit donné. Ce sont tout simplement des instructions écrites en langage *VHDL* décrivant les différentes relations entre les entrées et les sorties.

Exemple:

```
architecture Behavioral of essai is

begin

y<=a and b;

end Behavioral;
```

II.5.2. Les Opérateurs VHDL

L'opérateur le plus utilisé dans les structures *VHDL* est l'opérateur d'affectation (<=). Il permet de modifier l'état d'un signal donné. L'opérateur (**&**) permet d'associer deux signaux pour former un nouveau signal dont le nombre de bits est égal à la somme des signaux à associer.

Le *VHDL* dispose également des opérateurs logiques (***and***, ***or***, ***not***, ***nor***, ***xor***, …), arithmétiques (**+,-,*,/**) et relationnels **(=, >, <, >=, <=, /=)** qui sont prédéfinis dans les bibliothèques ieee.std_logic_1164.all et ieee.numeric_std.all.

Exemple :

```
If (A>=F or B=C) then
S1<=A & B;
S2<=(A*C)+D;
End if;
```

II.5.3. Exemple d'une description VHDL simple

L'exemple suivant est une description *VHDL* d'un décodeur, chaque partie est illustrée par des commentaires.

```
library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
```

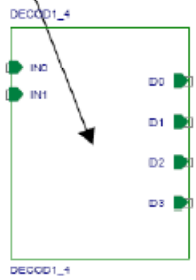
Déclaration des bibliothèques

Commentaires, en VHDL ils commencent par - -

```
-- décodeur
-- Un parmi quatre
```

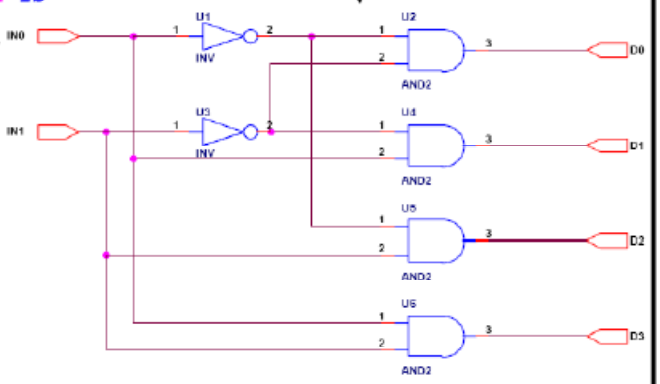
Déclaration de l'entité du décodeur
Correspondance schématique

```
entity DECOD1_4 is
      port(IN0, IN1: in std_logic;
           D0, D1, D2, D3: out std_logic);
end DECOD1_4;
```



Déclaration de l'architecture du décodeur
Correspondance schématique

```
architecture DESCRIPTION of DECOD1_4 is
begin
      D0 <= (not(IN1) and not(IN0));
      D1 <= (not(IN1) and IN0);
      D2 <= (IN1 and not(IN0));
      D3 <= (IN1 and IN0);
end DESCRIPTION;
```



III. 1. INTRODUCTION :

Le système de navigation par satellite GALILEO est basé sur la communication par étalement de spectre, plus précisément la technique CDMA (*Code Division Multiple Access*). L'étalement de spectre consiste à étendre la bande de fréquence du signal à transmettre, l'énergie associée sera donc répartie sur une bande plus large; la densité spectrale de puissance du signal utile est diminuée, par conséquent le signal sera perçu comme un bruit pour un utilisateur non concerné par la transmission. Cela est réalisé grâce à un codage de l'information à transmettre par une séquence pseudo-aléatoire (Pseudo Noise-code, PN code), que nous expliquerons un peu plus loin, dans le cas du GPS c'est une séquence de Gold, pour GALILEO il s'agit d'un code d'étalement bien spécifique appelé BOC (*Binary Offset Carrier*); ces codes d'étalement sont caractérisés principalement par une forme particulière de leurs fonctions de corrélation, car elles présentent un pic étroit pouvant être détecté par l'étage d'acquisition au niveau du récepteur. On décide la présence d'un code donné si ce pic dépasse un seuil fixe préétabli.

Le choix du seuil est donc très important, si le seuil est trop haut, des codes présents peuvent ne pas être détectés, nous parlons alors de mauvaise détection, si par contre, le seuil était choisi très bas, un bruit fort pourrait être considéré comme étant un code ; c'est une fausse alarme. A l'addition du bruit un autre phénomène se manifeste, c'est la présence des multitrajets, ces derniers présentent une dégradation importante des performances d'acquisition.

Afin de résoudre ce problème, nous proposons l'utilisation d'une technique de détection adaptative basée sur une probabilité de fausse alarme constante, c'est la détection adaptative CFAR (*Constant False Alarm Rate*) où le seuil sera variable en fonction des variations du signal. Cette technique était utilisée dans les systèmes radars, mais dernièrement son application est apparue dans les systèmes de communication à accès multiples par code *CDMA*.

Dans ce chapitre, nous allons exposer le principe de communication par étalement de spectre, nous présentons les codes de Gold et BOC, leur principe de génération et leur fonction de corrélation seront présentés. Nous donnons ensuite les différentes méthodes d'acquisitions pouvant être utilisées dans un récepteur GALILEO, suivies de la proposition de

plusieurs structures CFAR. Finalement, les courbes de performance exprimant la probabilité de détection seront présentées.

III.2. L'ETALEMENT DE SPECTRE :

Initialisées dans le domaine militaire pour assurer une certaine confidentialité aux communications, les techniques d'étalement de spectre (*Spread spectrum*) autorisent le partage d'une bande de fréquence et augmentent la résistance aux interférences et signaux parasites.

L'étalement de spectre consiste à étendre la bande de fréquence du signal à transmettre, l'énergie radioélectrique est ainsi repartie comme le montre le schéma de la figure III.1; la densité spectrale de puissance du signal utile est diminuée. Ce signal est perçu comme un bruit pour un utilisateur non concerné par la transmission. Cela est réalisé grâce à un codage de l'information à transmettre par une séquence pseudo-aléatoire (Pseudo Noise-code, PN code) connue seulement par l'utilisateur.

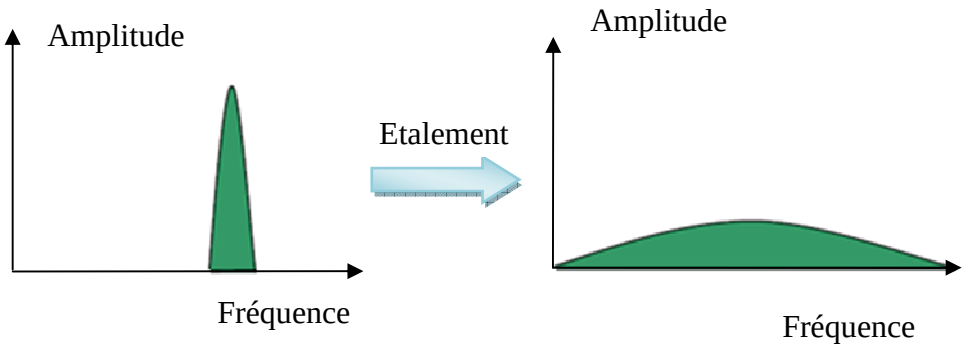


Figure III.1 : *Etalement du spectre*

Dans le cas du système GALILEO, les satellites transmettent sur les mêmes bandes de fréquences utilisant la technique CDMA. Chaque satellite possède son propre code pour être différencié lors de la réception de ses signaux. Cette technique possède des avantages tels que :

- ***Confidentialité* :** pour une puissance du signal d'information donné, l'étalement de spectre permet de répartir cette puissance sur les différentes répliques ce qui abaisse le niveau global du spectre. Ainsi, celui-ci peut passer en dessous du niveau du bruit.

- ***Cryptage :*** l'étalement de spectre constitue un moyen de cryptage : en effet, le signal étant déjà codé et en dessous du bruit, le seul moyen de le retrouver est de trouver le bon code utilisé à l'émission.
- ***Capacité de mesure du retard de propagation :*** les fonctions d'autocorrélation permettent la synchronisation entre le code local et le code entrant et donc de déterminer le retard et la pseudo-distance. Pour une transmission n'utilisant pas l'étalement de spectre, seule la boucle à verrouillage de phase fonctionne ce qui ne permet pas une telle mesure.
- ***Partage du canal d'émission :*** les signaux des différents satellites sont émis simultanément dans une même bande de fréquence. Chaque signal ayant son propre code, il n'y a pas d'interférences avec les autres.
- ***Résistance au brouillage :*** le spectre du signal portant l'information étant dupliqué sur une large bande, l'interférence d'un signal brouilleur n'affectera qu'une partie des répliques, les autres restants exploitables pour récupérer les données, la figure III-2 illustre ce principe.

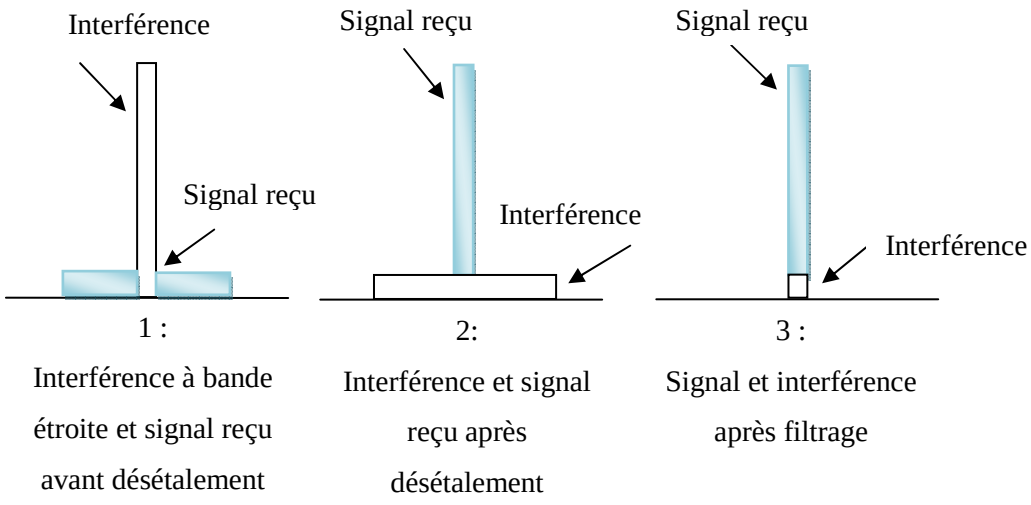


Figure III.2 : *Rejection des interférences à bande étroite dans les systèmes à étalement*

L'inconvénient majeur de l'étalement de spectre est son efficacité spectrale ; en effet, à débit donné la largeur de bande doit être plus importante. Cependant, on peut considérer que la bande n'est occupée que partiellement car des transmissions à bande étroite dans cette même bande peuvent se faire sans créer de perturbations.

III.3. CODES D'ETALEMENT POUR GPS/GALILEO :

Les codes d'étalement utilisés en GPS et GALILEO sont des codes dérivés d'une combinaison de séquences maximales.

III.3.1. Les codes de longueur maximale :

Un code linéaire est construit à partir de combinaisons linéaires faisant intervenir des opérations d'additions et de multiplications. Les séquences linéaires sont générées à partir de registres à décalages bouclés. Un code linéaire est de longueur maximale si sa période N est égale à 2^n-1, où 'n' représente le nombre d'étages du registre à décalage considéré. La figure II.3 représente un exemple de génération de code linéaire de longueur maximale avec n = 5. Chaque case du registre est dans un premier temps initialisée à 1. Ensuite, la somme modulo-2 est effectuée entre la première case et la cinquième pour cet exemple. Chaque valeur est ensuite décalée sachant que la cinquième constitue la sortie. Le résultat de la somme modulo-2 est alors introduit dans le première case. Ce processus est alors itéré et permet de générer un code pseudo-aléatoire périodique de longueur N = 31.

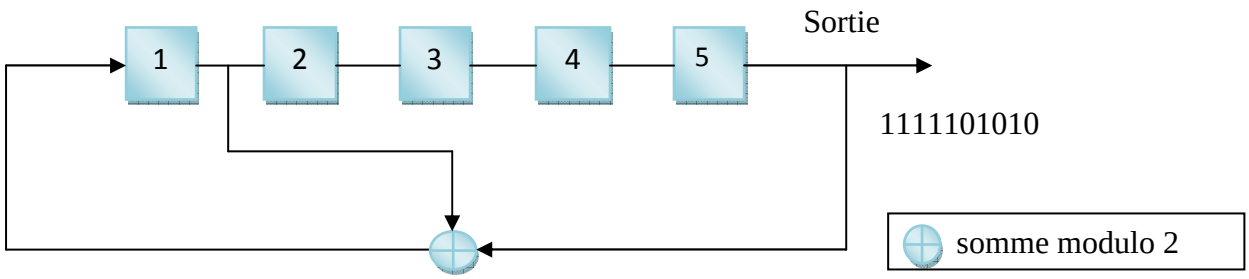


Figure III.3 : *Génération d'un code pseudo-aléatoire de longueur N = 31*

La fonction de corrélation $R_c(\tau)$ d'un code c(t) de longueur maximale de période 'N.T_c,' est définit sur une période par :

$$Rc(\tau) = \frac{1}{NTc}\int_0^{NTc} c(t+\tau)c(t)dt$$

Soit :

$$Rc(\tau) = \begin{cases} 1 - \frac{|\tau|}{Tc}\left(1+\frac{1}{N}\right) & \text{Pour} \quad 0 \le |\tau| \le Tc \\ -\frac{1}{N} & \text{Pour} \quad Tc \le |\tau| \le (N-1)Tc \end{cases}$$

Cette fonction d'autocorrélation est représentée sur la figure III.4 pour N = 31 bits. Il s'agit d'une fonction triangulaire se rapprochant d'une forme impulsionnelle sur une période. On note que la fonction d'autocorrélation est périodique car le code lui même est périodique.

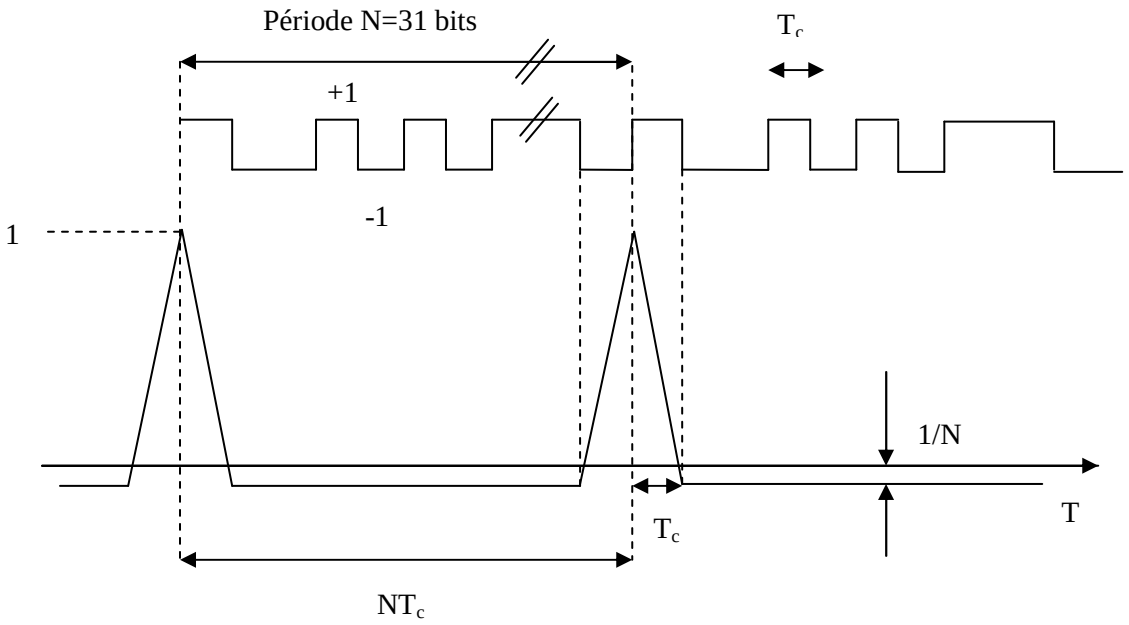


Figure III.4 : *Fonction d'autocorrélation d'un code pseudo-aléatoire*

III.3.2. Les codes de Gold :

Afin de générer un code de Gold, on génère en premier lieu une séquence maximale de longueur $N = 2^n$-1 (1023 bits avec n =10 dans le cas du GPS). Une des particularités de ces séquences est que si on calcule la corrélation entre un mot et ce même mot décalé d'un nombre arbitraire de bits, on n'obtient que deux valeurs :

$$1 \qquad \text{si le décalage est de 0}$$

$$-\frac{1}{N} \qquad \text{Autrement}$$

Mais il est possible de générer des paires particulières de séquence maximale dont la corrélation donne trois valeurs :

$$-\frac{1}{N}t(n), -\frac{1}{N} \text{ et } -\frac{1}{N}[t(n)-2]$$

Avec : $t(n) = \begin{cases} 1+2^{\frac{n+1}{2}} & \text{Si n est pair} \\ 1+2^{\frac{n+2}{2}} & \text{Si n est impair} \end{cases}$

Soit une corrélation maximale de l'ordre de $2^{-\frac{n}{2}}$ en valeur absolue.

Si on considère une paire privilégiée de séquences maximales (m_1, m_2), on construit les mots d'un code de Gold.

Les codes de Gold ainsi construits sont faiblement corrélés, avec les mêmes trois valeurs de corrélation possibles que pour la corrélation entre m_1 et m_2. De plus, leur autocorrélation présente un pic étroit.

III.3.3. Les codes C/A :

Le code C/A fait partie de la famille des codes de GOLD. Le choix de deux séquences de même longueur N est tel que leur intercorrélation ne peut prendre que trois valeurs. Ces deux codes sont obtenus à partir de deux registres linéaires à décalages bouclés de n = 10 étages. Il s'agit alors d'un code de longueur 1023 bits cadencé à une fréquence de 1.023 Mbits/s. La figure II.5 illustre le principe de génération de ce code. Les cases des registres G1 et G2 sont initialisé à 1. On obtient la séquence G2i en effectuant une addition modulo-2 entre les sorties de deux étages du registre à décalage G2. Cela équivaut à retarder la séquence G2 d'un certain nombre entier de bits. On note alors

$$C = G1\,(t)\,G2\,(t + n_i\,T)$$

avec ' $n_i\,T$' l'offset de phase.

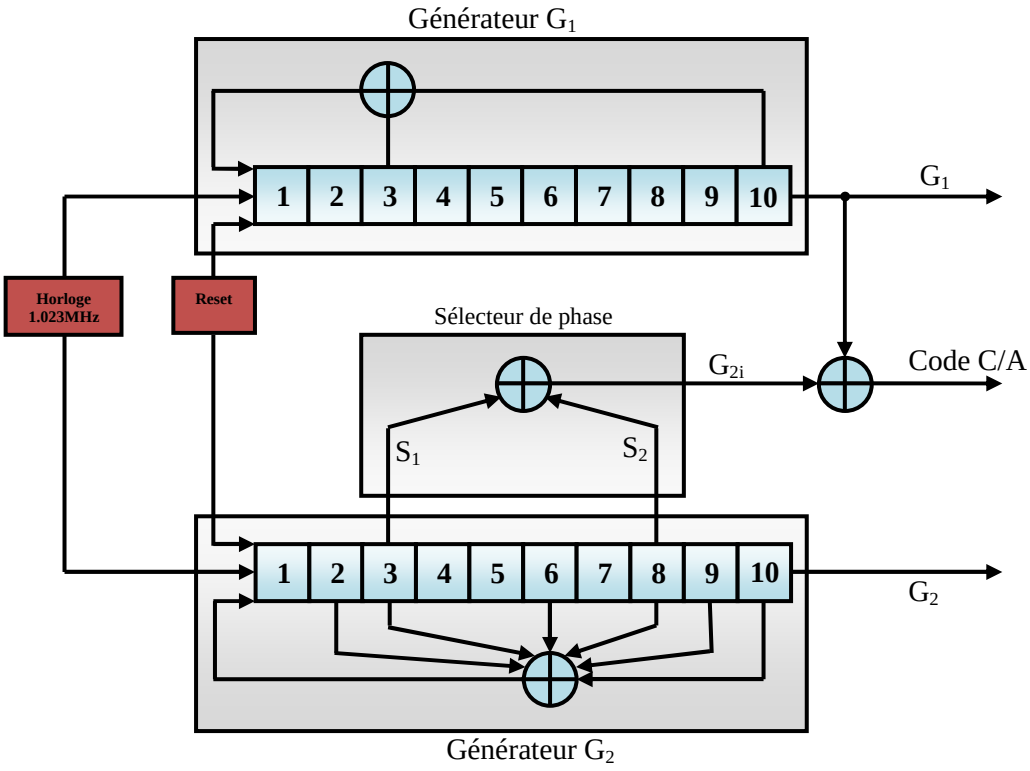


Figure III.5 : *Génération du code C/A*

Il existe alors 1023 retards possibles, mais ce sont les 36 retards présentant les meilleurs produits d'intercorrélations (c'est-à-dire les plus faibles possibles) qui sont attribués aux satellites. Si N est grand, on peut alors approximer la fonction d'autocorrélation par :

$$Rc(\tau) = \begin{cases} 1 - \frac{|\tau|}{Tc} & \text{Pour} \quad 0 \leq |\tau| \leq Tc \\ 0 & \text{Pour} \quad Tc \leq |\tau| \leq (N-1)Tc \end{cases}$$

Cette fonction est représentée par la figure II-6 suivante :

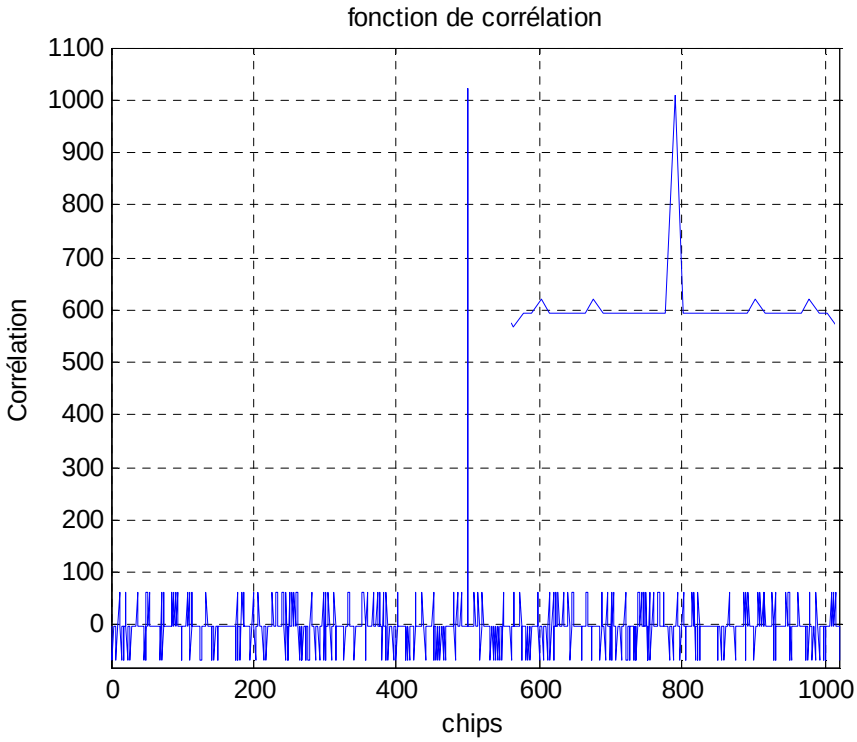


Figure III.6: *Fonction de corrélation du code C/A*

III.3.4. Les codes BOC:

La bande de fréquence allouée pour les systèmes de navigation par satellite est la bande L (1164-1610 MHZ), elle est actuellement partagée par les systèmes GPS, GLONASS et GALILEO. Le GPS et GALILEO cohabite la sous bande E2-L1-E2 où les seules bandes disponibles pour GALILEO sont de 4 MHZ de part et d'autre de L1 déjà occupée par le GPS. La naissance de la modulation BOC (*Binary Offset Carrier*) a pu résoudre ce problème de la saturation spectrale.

La modulation BOC se caractérise par un spectre composé de deux lobes principaux décalés symétriquement par rapport à la fréquence centrale de la porteuse (1575MHZ), cette fréquence sera toujours utilisée par le GPS et GALILEO mais sans problèmes d'interférences, au contraire, cette caractéristique permettra de faciliter la réalisation de récepteur bi-modes. Les codes BOC correspondent alors à l'apport d'une modulation supplémentaire par une sous porteuse de type rectangulaire et de fréquence $F_{s,}$ en vue d'une meilleure utilisation de la bande allouée. L'expression générale du signal *BOC* étudié est : $s(t) = A\ d(t)c(t)S_{boc}(t)\cos(2\pi\ \mathrm{L}_1\mathrm{t})$

Avec :

$$S_{boc}(t) = sign[sin(2\pi\ F_s\ t)]\mathrm{S}_{\mathrm{boc}}(\mathrm{t})$$

$\mathrm{S}_{\mathrm{boc}}(\mathrm{t})$ représente la sous porteuse supplémentaire, qui sera prise de type rectangulaire, mais on notera que l'on aurait pu travailler sur d'autres sous porteuses de type sinusoïdal, appelées M-LOC.

Deux paramètres sont utilisés pour définir les signaux de modulation BOC. Ces paramètres sont :

- La valeur de la fréquence du signal$s(t)$, déterminée à travers un paramètre q : $F_s = q * F_0$
- La valeur de la fréquence des bits du code d'étalementF_c, déterminée à travers un paramètre p : $F_c = p * F_0$

Ainsi, on définit $n = 2 * (\frac{q}{p})$ des paramètres caractéristiques des signaux *BOC*. Le rapport de ces deux paramètres à travers n a une importance sur la forme du spectre du signal *BOC*. En effet, suivant que n est pair ou impair, notre signal *BOC* n'aura pas les mêmes propriétés.

La fréquence F_0 est la fréquence caractéristique du récepteur, elle correspond à sa puissance de traitement numérique et sera fixée pour notre étude à : $F_0 = 1.023\ MHZ$

Les autres caractéristiques des signaux ne varient pas par rapport à l'étude des signaux GPS classiques, soit :

- $d(t)$ la matérialisation du message de navigation (50 Hz) par un P/NRZ/L ;
- $c(t)$ la matérialisation du code C/A (cadence de 1,023 MHz) par un P/NRZ/L;
- La fréquence L1 est la fréquence classique de la porteuse GPS à 1575,42 MHz.

On peut générer des signaux *BOC* selon le synoptique suivant :

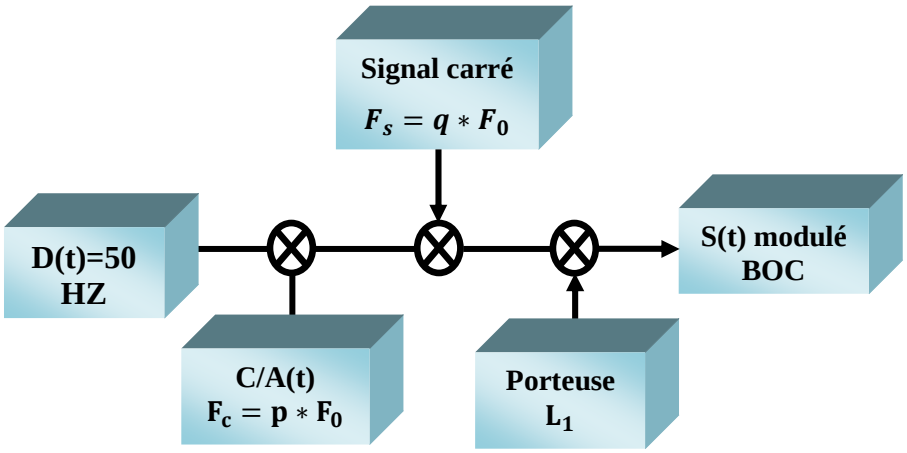


Figure III.7 : Générateur du signal BOC

On note que le générateur PRN est le même utilisé en GPS, sauf qu'ici sa fréquence peut être multiple de 1.023 MHz, aussi nous avons la possibilité de changer la fréquence du signal.

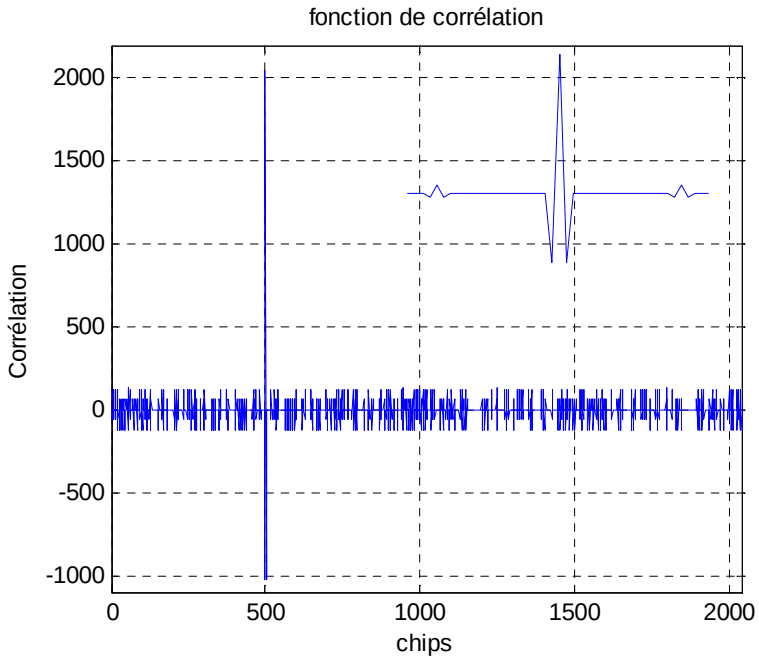


Figure III.8: *Fonction de corrélation du code BOC(1,1)*

III.4. L'ACQUISITION DU SIGNAL :

Dans le cas de la communication CDMA, il est nécessaire d'estimer les décalages temporel et fréquentiel du signal reçu afin de réaliser le désétament et la récupération des données. Ce processus est appelé synchronisation, il est réalisé en deux étapes : acquisition et poursuite. L'acquisition composée de deux étages, estimation puis détection, est une recherche bidirectionnelle en temps et en fréquence qui a pour buts d'identifier les satellites visibles à partir des codes PRN (*Pseudo Random Number*) reçus et de récupérer les retards associés à ses codes.

III.4.1. Etage d'estimation :

Dans cet étage, l'ensemble des décalages temporels et fréquentiels sont explorés en calculant à chaque décalage la fonction d'intercorrélation entre un signal généré localement et celui reçu. On note que cette recherche est bidimensionnelle selon des résolutions prédéterminées et l'espace de recherche en temps est typiquement égal à la longueur du code utilisé, tandis que pour la fréquence qui diffère à cause du décalage Doppler on choisit un pas de 500 Hz. On note que pour un récepteur stationnaire la variation de la fréquence est supposée entre ±5 kHz. La figure II-9, illustre cette recherche.

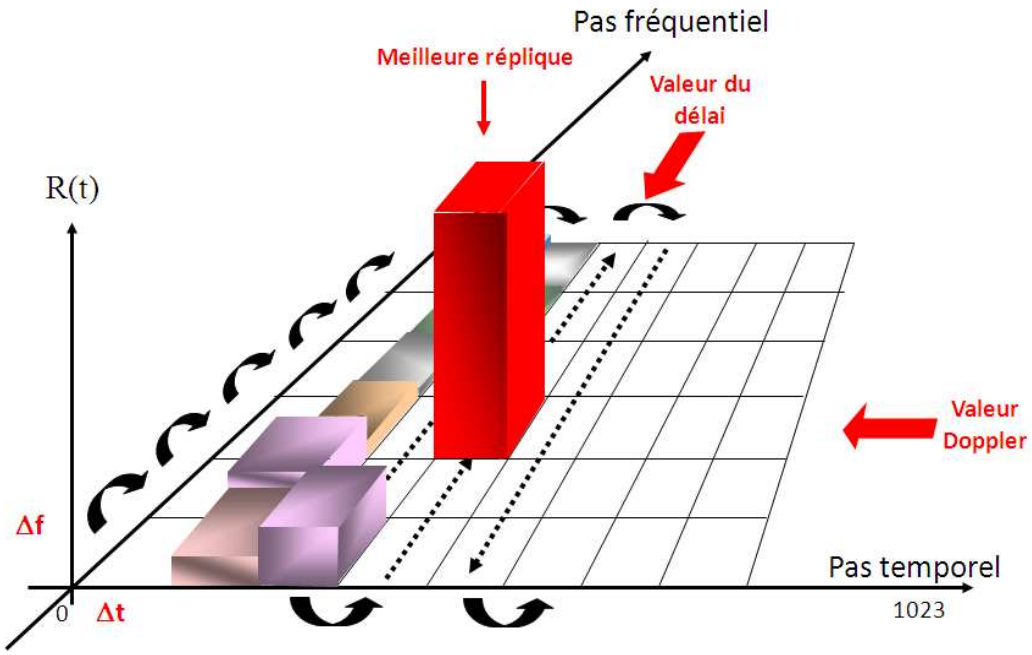


Figure III.9 : *Processus d'estimation du code phase et du Doppler*

Dans cette figure, on représente les incertitudes sur le décalage temporel et le Doppler par des cellules d'une largeur ΔT_c et d'une hauteur Δf représentant le Doppler maximum toléré. Si le code généré localement est bien aligné avec le code du signal reçu et la fréquence de la porteuse générée localement correspond à la fréquence du signal reçu, un dépassement de seuil sera alors détecté au niveau de l étage de détection, ainsi l'acquisition est assurée. La fréquence de la porteuse et la phase du code C/A sont correctes, les paramètres peuvent être transmis aux algorithmes de poursuite.

Dans un récepteur GPS ou GALILEO, les codes utilisés sont des codes longs par conséquent le temps de recherche est important, pour cette raison, plusieurs méthodes ont été proposées, principalement : la recherche série, la recherche parallèle et la recherche hybride. Ces différentes méthodes seront détaillées en ce qui suit.

III.4.1.1.L'acquisition en recherche série:

La recherche série est la méthode la plus simple la plus fréquemment utilisée. La technique consiste à effectuer la corrélation des répliques des codes et des porteuses générées localement avec le signal reçu en essayant plusieurs hypothèses sur la phase du code et sur la valeur du Doppler, avec un pas d'échantillonnage suffisamment fin pour ne pas manquer le pic de corrélation. Le principe d'acquisition série dans ce cas est illustré sur la figure III.10 suivante :

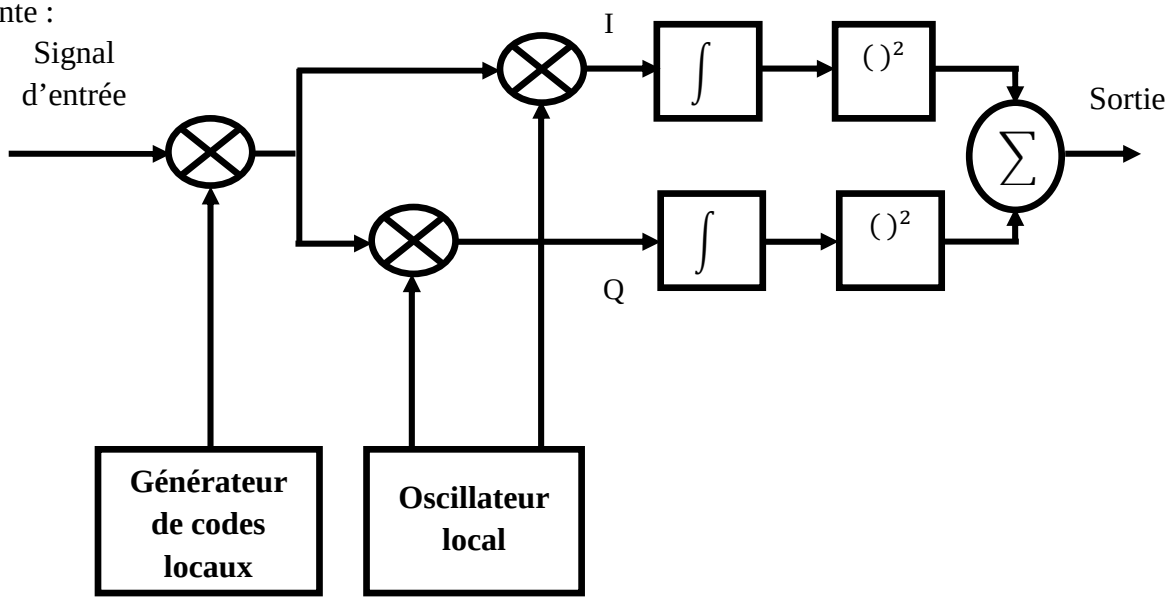


Figure III.10 : *L'acquisition en recherche série*

Les satellites GPS sont différenciés par les 32 séquences PRN différentes. Au lieu de générer des séquences PRN chaque fois que l'algorithme d'acquisition est exécuté, toutes les séquences possibles sont générées. Avec les 32 PRN générés, toutes les séquences provenant de satellites GPS sont réalisées. Il est nécessaire de connaître aussi la phase du code (0 à 1022 chips) afin de générer un code PRN qui soit parfaitement aligné avec le code reçu du satellite, cela donne un total de 32736 codes PRN différents. Dans cette méthode la sortie sera donnée par la figure III.10.a.

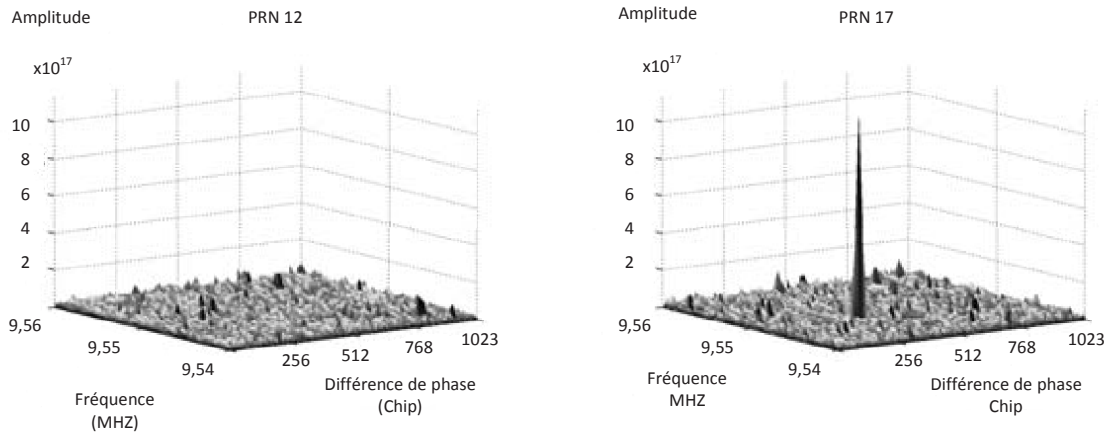


Figure III.10.a : Sortie *de l'acquisition par recherche série. (à droite) Satellite #12 n'est pas visible. (à gauche) Satellite #17 est visible (présence du pic à une fréquence égale à 9.548MHz et une phase de code égale à 359 chip).*

III.4.1.2.L'acquisition parallèle sur la fréquence:

Le schéma de principe de cette technique est représenté par la figure III.11 ci-dessous.

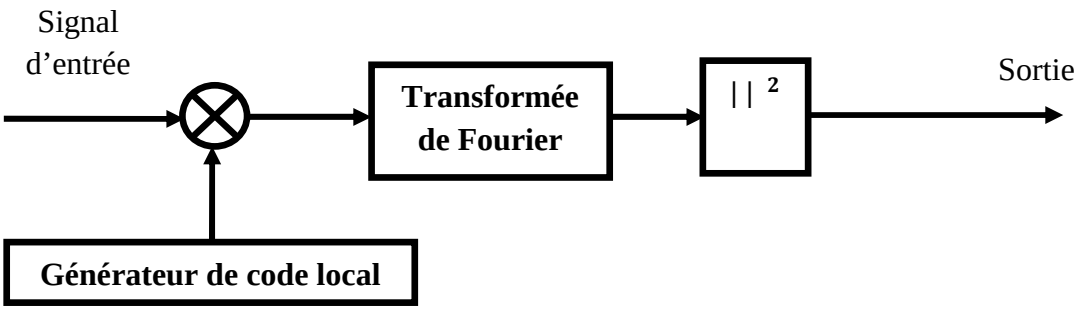


Figure III.11 : *L'acquisition parallèle sur la fréquence*

La technique est effectuée en deux étapes :

- Dans un premier lieu, le signal reçu multiplié avec une réplique de code générée identique à celle qui étale le spectre de la porteuse avec un code phase variant entre 0 et 1022 chips ;
- De plus, on peut détecter la fréquence de la porteuse en utilisant la transformée de Fourier TF. La TF présente un pic localisé sur la fréquence porteuse du signal reçu.

La figure III.11.a donne le résultat de multiplication entre le signal reçu de satellite et le code local généré parfaitement aligné avec le code reçu. Après la multiplication des deux codes, le signal est transformé en domaine des fréquences par transformée de Fourier. Si le code généré localement est bien aligné avec le code reçu, la sortie de la TF aura un pic à la fréquence de la porteuse plus le Doppler. Pour trouver la fréquence possible, la valeur absolue de toutes les composantes est calculée. La figure III.11.b donne la représentation spectrale en utilisant l'algorithme FFT pour l'acquisition. La figure III.11.b (gauche) donne la sortie de la transformée de Fourier pour une séquence générée parfaitement alignée avec le code du signal reçu (présence du pic). La figure III.11b (droite) donne le résultat pour un code non aligné (absence du pic).

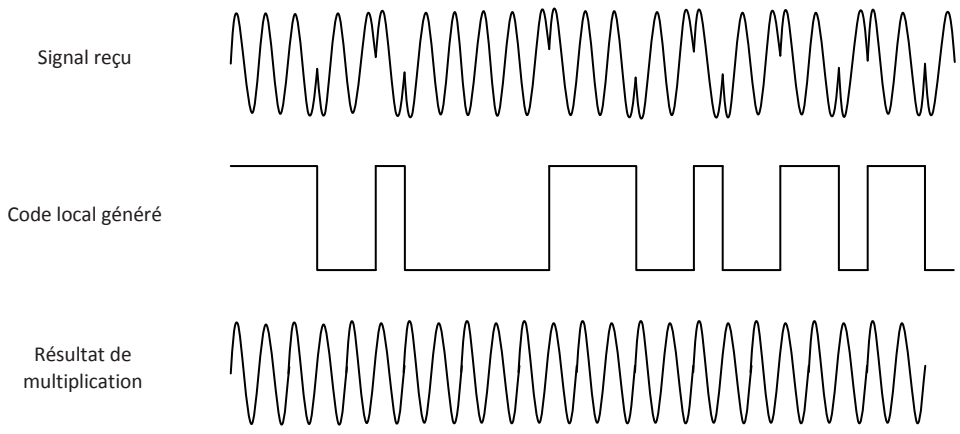


Figure III.11.a : *Démodulation du code PRN : le code local est aligné avec le code reçu.*

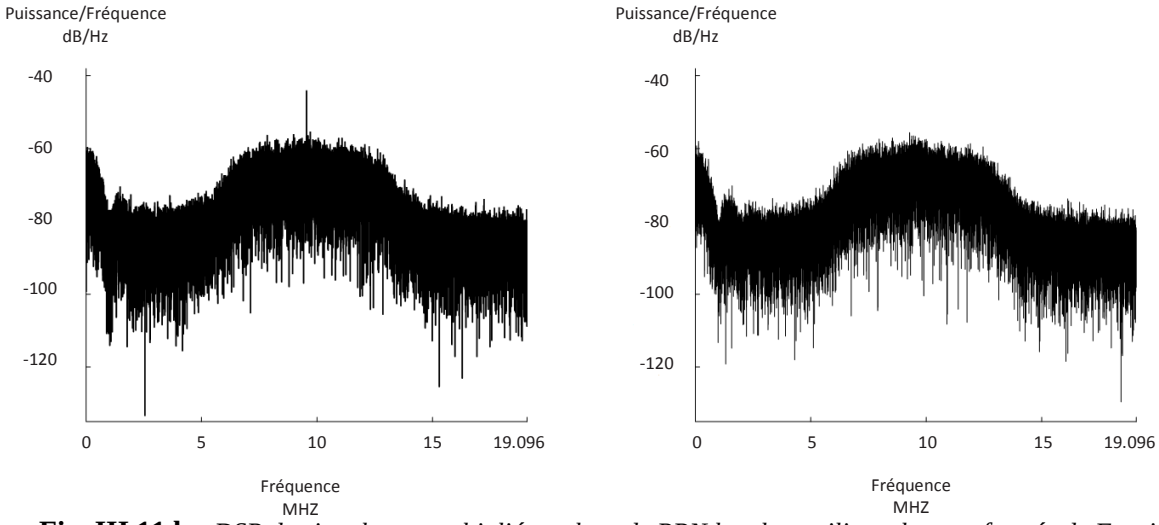


Fig. III.11.b : *DSP du signal reçu multiplié par le code PRN local en utilisant la transformée de Fourier. (gauche) La sortie de la transformée de Fourier pour une séquence générée localement identique et parfaitement en phase avec celle qui étale le spectre de la porteuse. (droite) Les codes ne sont pas alignés (absence du pic).*

La figure III.12 donne la sortie de l'acquisition par recherche par recherche Parallèle sur la fréquence.

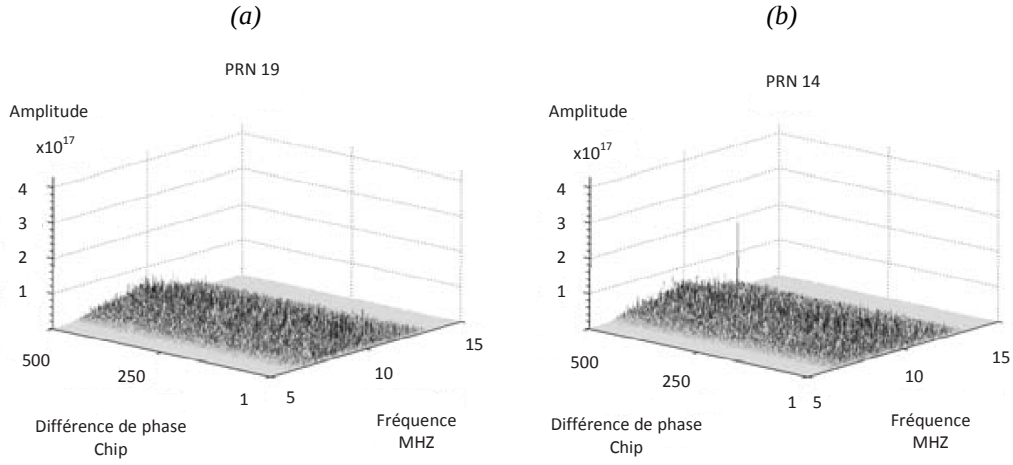


Figure III.12 : *Sortie de l'acquisition par recherche Parallèle sur la fréquence. (a) Satellite #19 n'est pas visible (absence du pic). (b) Satellite #14 est visible (présence du pic à une fréquence égale à 9.548MHz et une phase de code égale à 359 chip).*

III.4.1.3. L'acquisition parallèle sur le code phase (acquisition circulaire):

Le schéma de principe de cette technique est représenté sur la figure III.12 ci-dessous.

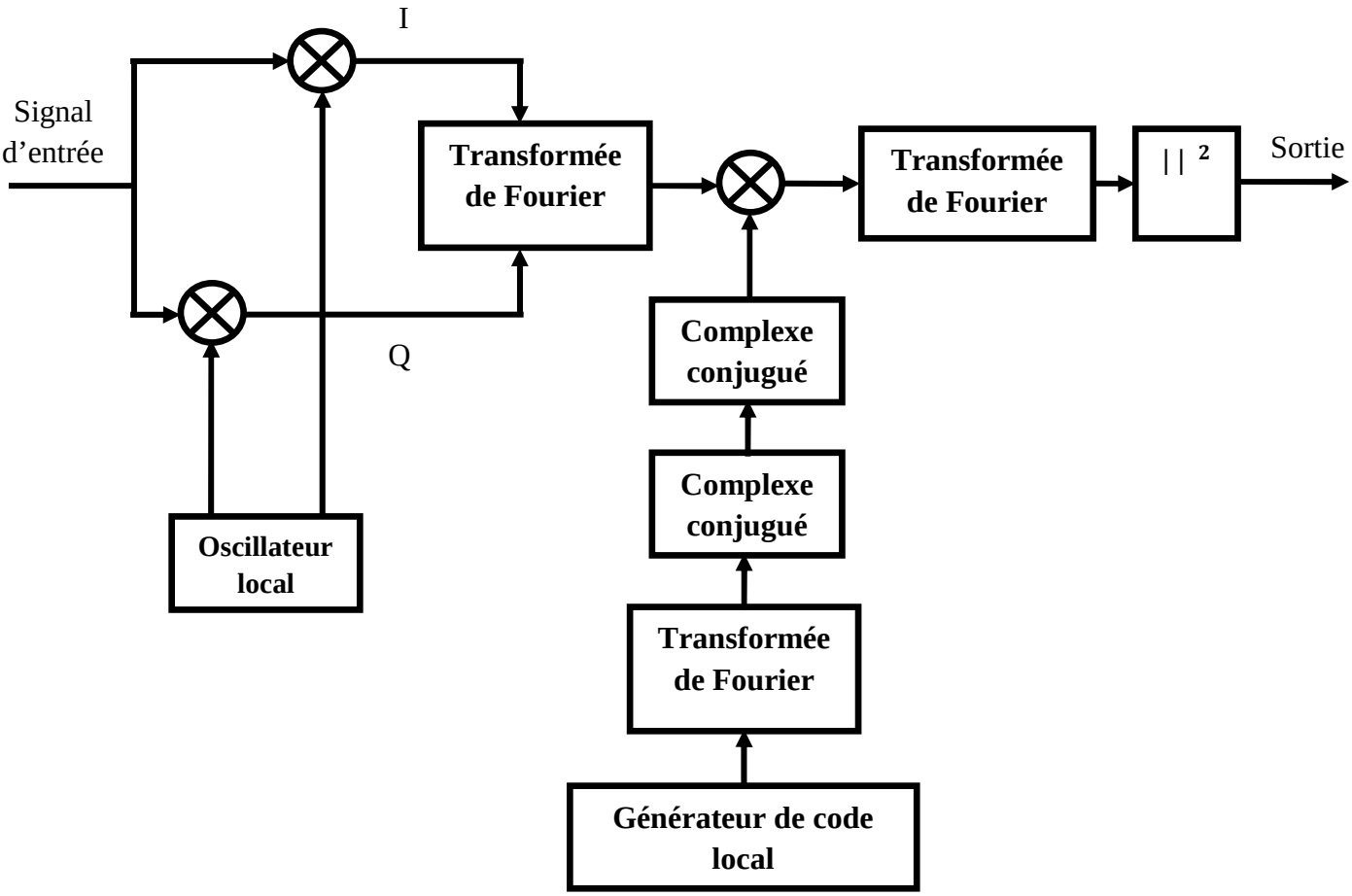


Figure III.12 : *L'acquisition circulaire*

Ce synoptique représente le principe de l'acquisition par recherche parallèle sur la phase du code. La porteuse reçue est multipliée par la composante en phase et par la composante en quadrature de la porteuse générées localement par l'oscillateur local. Nous déterminons ainsi les composantes en phase I et en quadrature Q. Puis les deux voies I et Q sont combinées à fin de fournir une entrée complexe $x(n) = I(n) + jQ(n)$ de la TFD.

Le résultat est multiplié par le conjugué de la transformée de Fourier de la réplique de code PRN générée localement. La valeur absolue de la sortie de la transformée de Fourier inverse représente la corrélation entre le entrée et le code PRN. Si un pic est présent dans la

corrélation, l'indice de ce pic représentera le code phase du code PRN du signal reçu (comme le montre la figure III.12.a).

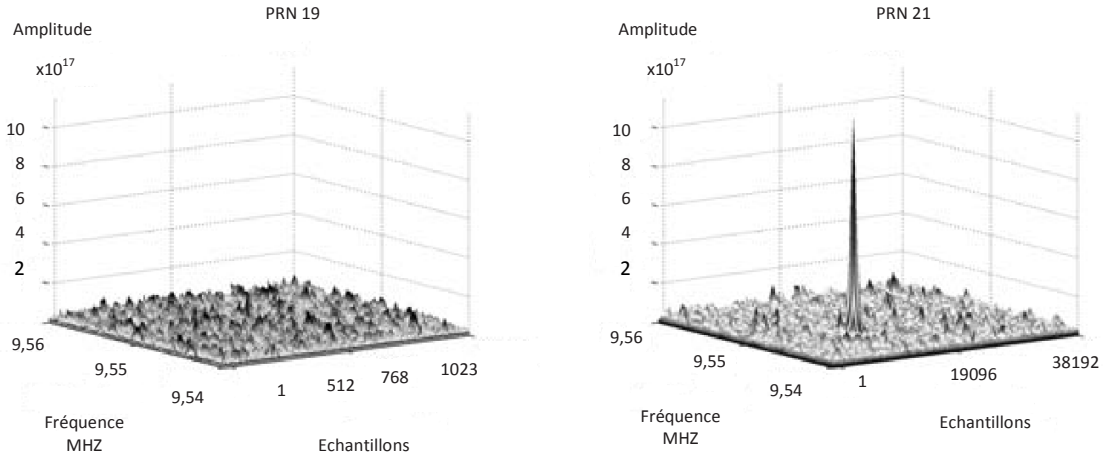


Figure III.12.a : Sortie *de l'acquisition par recherche Parallèle sur la phase du code. (gauche) Satellite #19 n'est pas visible. (droite) Satellite #21 est visible (fréquence égale à 9.5475MHz et un retard égal à 13404 échantillon).*

III.4.2 Etage de détection :

Une fois, l'opération de recherche est réalisée, la sortie présente un pic déclarant la présence du code considéré, mais seulement si ce pic dépasse un certain seuil fixe préétabli. Ce seuil est choisi selon une modélisation statistique basée sur les lois de probabilité. A chaque fois le pic de corrélation dépasse le seuil, on décide que le code est présent et la probabilité d'avoir détecté cette présence correctement est notée probabilité de détection P_d ; Dans le cas d'un dépassement de seuil or le code n'est pas présent, on parle de fausse alarme décrite par une probabilité de fausse alarme P_{fa} ; on peut rencontrer un autre cas, c'est quand le seuil ne dépasse pas le seuil or que le code est présent, il s'agit d'une non détection, décrite par une probabilité de non détection P_{nd}, cette dernière situation apparaît si on choisit un seuil très élevé ou dans le cas de présence d'un bruit important.

Le choix du seuil possède une influence significative sur la performance de l'étage d'acquisition. Comme le bruit accompagnant le signal reçu est aléatoire, le seuil doit être fixé dans un contexte probabiliste selon des règles statistiques basées sur la densité de probabilité. Dans le cas d'un bruit blanc de moyenne nulle affectant un signal constant qui apparaît à un

moment donné, la densité de probabilité en présence et absence du signal est illustrée par la figure III.13 suivante :

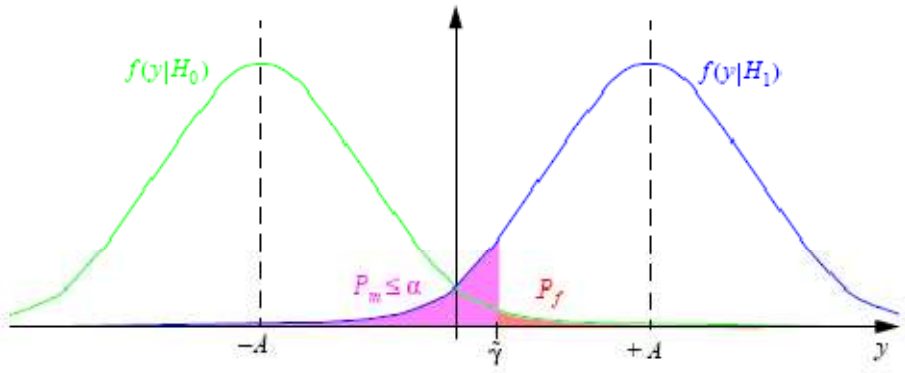


Figure III.13. *Densité de probabilité en absence et présence du signal*

Dans ce cas l'espace de décision est partagé en deux parties, H_0 qui correspond à une décision d'absence du pic de corrélation et H_1 qui correspond à une décision d'absence de ce pic. La probabilité de détection est donnée par :

$$P_d = P[D = H_1/H_0]$$

Et la probabilité de fausse alarme :

$$P_f = P[D = H_0/H_1]$$

On démontre aussi que le test statistique associé est donné par :

$$L(y) = \frac{f(y/H_1)}{f(y/H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma$$

Où : $f(y/H_0)$ est la densité de probabilité en absence du code désiré,

$f(y/H_1)$ est la densité de probabilité en présence du code désiré,

γ est le seuil fixe à déterminer en fonction de la probabilité de fausse alarme Pfa et la probabilité de détection Pd.

III.5 DETECTION ADAPTATIVE CFAR :

Le concept de la détection adaptative *CFAR* dans les systèmes de communication à accès multiple *CDMA* a commencé à apparaître dans les sept dernières années. Le principe est le même que celui du radar, mais la philosophie est totalement différente. La technique d'accès multiple par code avait été utilisée dans le domaine militaire pour ses propriétés d'éviter les interférences et la saturation spectrale. Comme il a été déjà cité, un système utilisant la technique *CDMA* doit assurer la transmission d'un code supplémentaire indépendant du message à transmettre connu pour le récepteur lui permettant d'identifier l'émetteur.

Le réglage du seuil de détection possède un rôle important dans un système de détection, puisqu'il est la base de la décision de la présence d'une cible ou de la détection du pic d'acquisition. L'application de la technique *CFAR* dans les systèmes *CDMA* est très récente, des études ont montré l'efficacité et la performance de cette technique par rapport à la détection à seuil fixe qui n'est plus utilisée actuellement.

Tout système *CDMA* comprend un bloc d'acquisition sous la forme illustrée dans la figure *(Figure II.14)*, il est composé de deux principaux sous-systèmes à savoir le filtre non-cohérent et le détecteur CFAR. La figure *(Figure II.15)* montre le détail du détecteur *CFAR*, le signal d'entrée $r(t)$ est composé du message, du code et bien évidemment d'un bruit additionnel. Si le détecteur adaptatif détecte que la séquence testée est la bonne, l'étage de poursuite de phase sera activé pour terminer l'opération de poursuite en phase, le temps du signal généré localement sera retardé de ΔT_c pour détecter la prochaine cellule. Si par contre le détecteur déclare H_0 le processus sera répété pour traiter toutes les valeurs du signal.

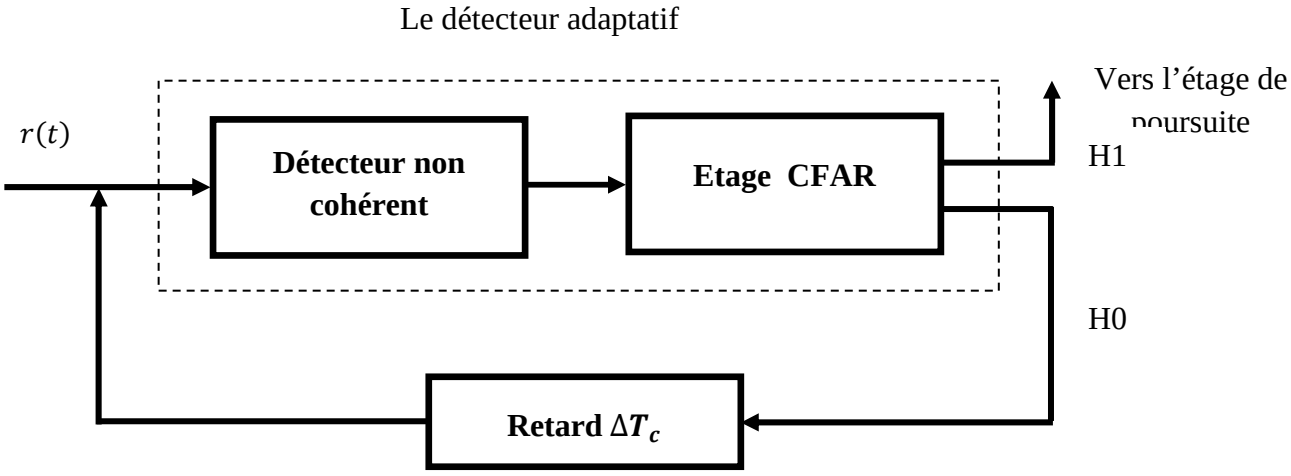


Figure III.14 : *L'acquisition adaptative dans les systèmes CDMA*

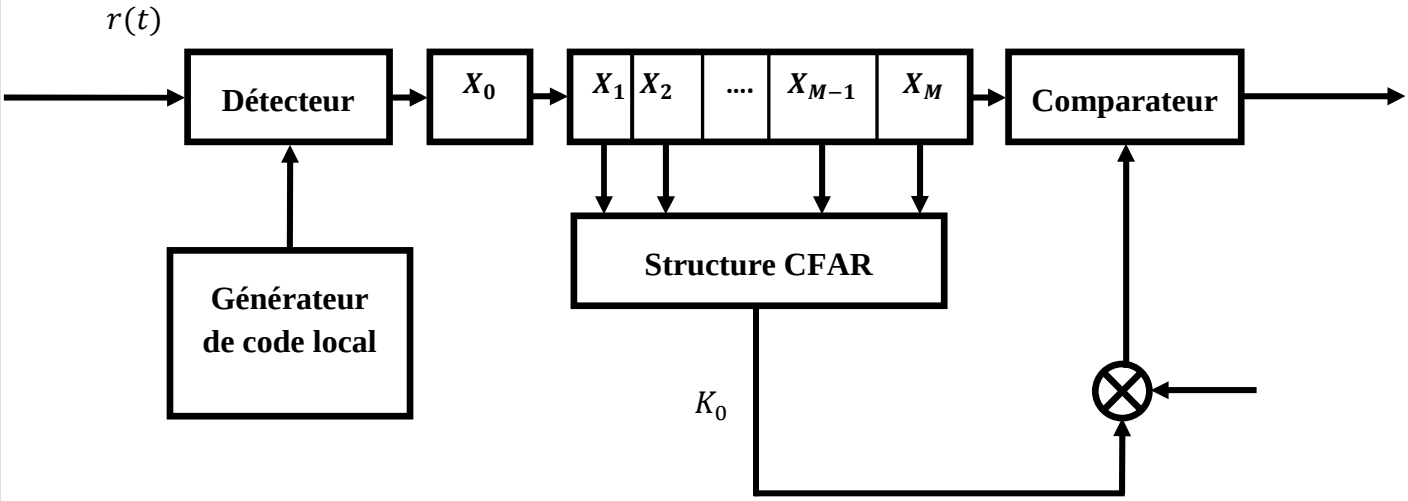


Figure III.15 : *L'étage CFAR dans les systèmes CDMA*

La particularité de la détection adaptative est que la valeur du seuil est variable en fonction des valeurs du signal reçu, celles-ci sont introduites dans un registre à décalage permettant de tester toutes les valeurs du signal, notons que les données enregistrées dans les cellules de test sont les même que celles utilisées dans la détection adaptative dans le radar, les opérations de calcul du seuil sont les mêmes et même le facteur K_0est similaire.

Il existe plusieurs types de détecteurs *CFAR*, on note principalement :

- Le détecteur CA-CFAR : Cell Averaging CFAR ;
- Le détecteur GO-CFAR : Greatest Of CFAR;
- Le détecteur SO-CFAR : Smallest Of CFAR;
- Le détecteur OS-CFAR: Order Statistic CFAR.

On note que ces processus diffèrent selon la manière par laquelle le test statistique est obtenu, ce qui est clarifié par le synoptique suivant:

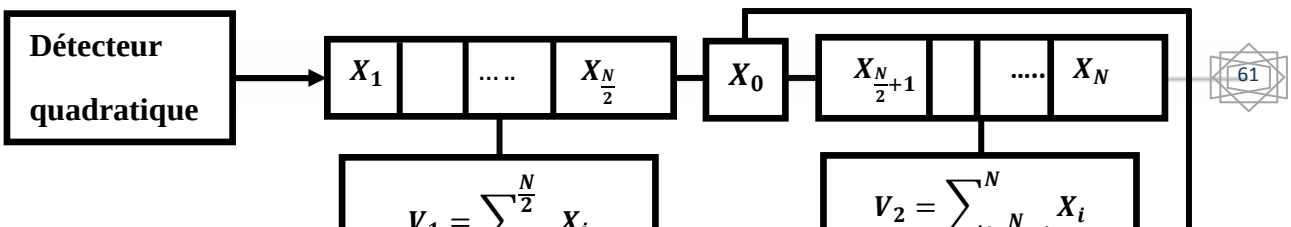


Figure III.16: Les structures CFAR, CA, GO, SO et OS CFAR

III.5.1. Détection adaptative CA-CFAR:

C'est la technique de détection adaptative la plus ancienne, le principe de ce type de détection est basé sur le calcul des moyennes de plusieurs valeurs du signal contenues dans des cellules d'un registre à décalage.

Les échantillons Z à la sortie du détecteur sont envoyés en série vers un registre à décalage de taille M +1 = 2m +1, comme le montre la figure III.16. Le test statistique X est obtenu à partir de l'estimation de la puissance moyenne du signal contenu dans les cellules de référence (cellules entourant la cellule à tester). La valeur fournie par le test statistique est ensuite multipliée par un facteur de seuil T et la détection est déclarée lorsque le signal contenu dans la cellule à tester dépasse le seuil résultant '*T*X*'. '*T*' représente un facteur constant utilisé pour obtenir la probabilité de fausse alarme désirée pour une fenêtre de taille '*M*'.

Cette technique suppose que le signal utile soit contenu dans la cellule centrale *CUT* et les cellules adjacentes de part et d'autre contiennent des échantillons de bruit Gaussien et indépendants et de moyennes nulles.

En utilisant la formule de la probabilité de fausse alarme évaluée en annexe I, nous aurons :

$$P_{fa} = \frac{1}{(1 + K_0)^N}$$

Nous remarquons que la probabilité de fausse alarme est constante et indépendante de la puissance du bruit, et que le seuil est fonction du bruit, c'est-à-dire, varie en fonction de la variation de la puissance du bruit; c'est le principe de la détection adaptative.

III.5.2. Détection adaptative GO-CFAR:

Dans ce cas, pour choisir le seuil on ne choisit pas la somme, mais le maximum des deux moyennes sera considéré. Ce maximum sera multiplié par le coefficient T puis comparé avec le signal contenu dans la cellule *CUT*. L'inconvénient de cette technique est qu'elle n'est pas très performante dans le cas de plusieurs cibles.

III.5.3. Détection adaptative SO-CFAR:

Dans ce type de détecteur, on considère le minimum des deux moyennes, cette valeur sera multipliée par un coefficient T pour être enfin comparé au signal contenu dans la cellule centrale comme pour le *CA-CFAR*. Cette méthode est très efficace dans le cas de plusieurs cibles très proches.

III.5.4. Détection adaptative OS-CFAR:

Les différents détecteurs *CFAR* élaborés précédemment présentent en général deux problèmes majeurs, à savoir le changement brusque de la puissance d'un signal inutile (*clutter*) et la présence de plusieurs pics à cause des multitrajets. Le premier problème se produit quand la puissance totale du bruit reçu dans la cellule de référence change brusquement, cela peut mener à une estimation incorrecte du seuil et par la suite des fausses alarmes ou des mauvaises détections. Le deuxième souci se manifeste dans le cas de la présence d'un multitrajet très proche en distance.

Le *OS-CFAR* peut remédier à ces problèmes à un certain degré, son principe de fonctionnement consiste à trier les valeurs contenues dans les cellules contenant les valeurs des signaux reçus en fonction de leurs amplitudes et de choisir en suite une valeur de k^{eme} ordre pour le calcul du seuil, le paramètre T sera donc un paramètre supplémentaire qui sera choisi auparavant. Les performances du *OS-CFAR* ont été démontrées par plusieurs études dans le cas de plusieurs pics.

III.6. ÉVALUATION DES PERFORMANCES DES DIFFERENTS DETECTEURS :

L'évaluation des performances des différents détecteurs est nécessaire afin de formuler une idée sur les plages de variation de chaque paramètre. Elle peut être réalisée par l'étude des courbes ROC (*Receiver operating characteristics*) correspondantes à la probabilité de détection fonction du rapport signal à bruit pour une probabilité de fausse alarme constante. Ces courbes caractéristiques peuvent être évaluées directement par des formules mathématiques exprimant la probabilité de détection pour chaque détecteur, ou bien en utilisant la simulation Monte-carlo. Cette dernière, est utilisée lorsque l'estimation de la densité de probabilité ne peut pas être obtenue analytiquement, ce qui correspond à notre cas étudié. Les résultats de simulation obtenus sont résumés dans les figures suivantes.

La figure III-17, représente la probabilité de détection en fonction du rapport signal à bruit pour une probabilité de fausse alarme $P_{fa}=10^{-8}$, ceci dans le cas du code CA pour les structures CA, GO et SO CFAR, on observe que ce dernier présente une meilleure performance en présence du bruit. De même, la figure III-18, représente la probabilité de détection en fonction du rapport signal à bruit pour une probabilité de fausse alarme $Pfa=10^{-8}$, aussi pour les structures CA, GO et SO CFAR, mais dans le cas du code BOC. On constate que le détecteur SO-CFAR est bien meilleur.

La figure III-19, illustre la probabilité de détection du détecteur CA-CFAR en utilisant le code BOC pour différent nombre de cellules N. On observe qu'à partir d'un rapport signal à bruit supérieur à 10 la détection est améliorée pour N élevé.

La figure III-20, montre que l'utilisation du code BOC présente une légère amélioration en probabilité de détection par rapport au code C/A.

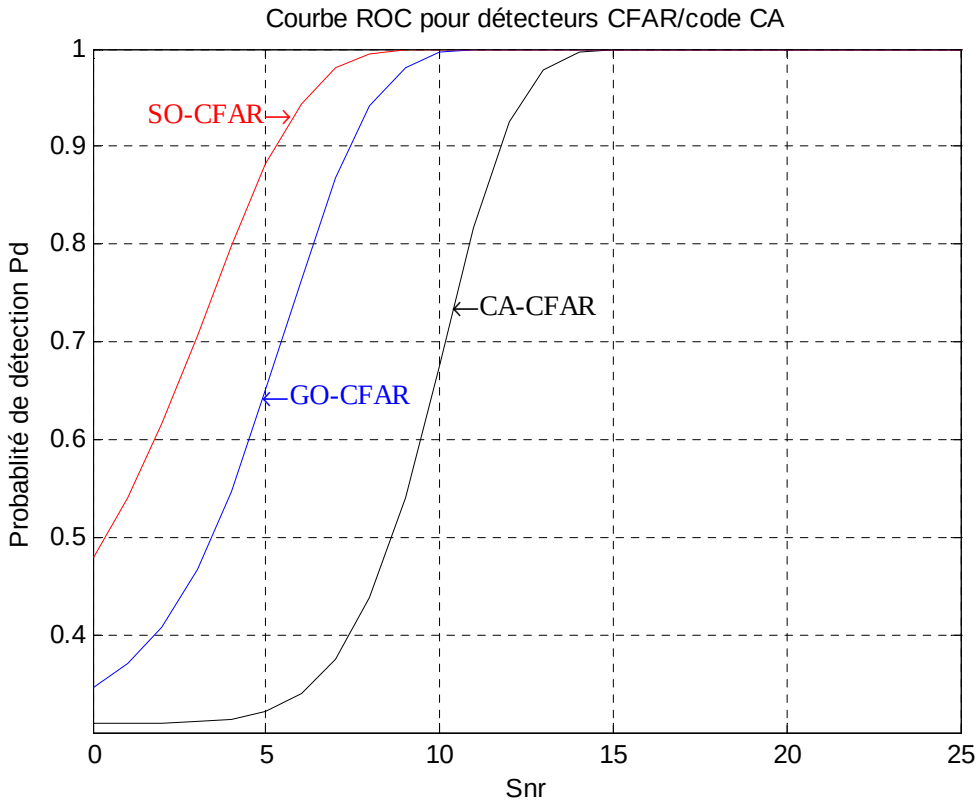


Figure III.17: Performance de détection *des detecteurs CA, GO et SO CFAR pour le code CA*

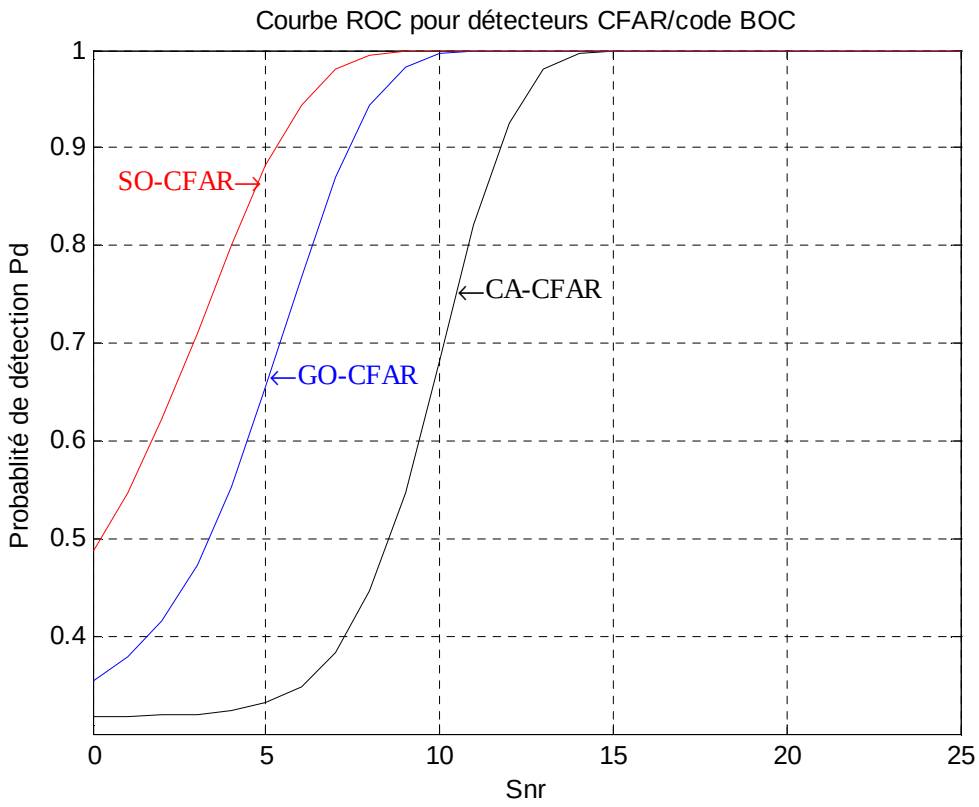


Figure III.18: Performance de détection *des detecteurs CA, GO et SO CFAR pour le code BOC*

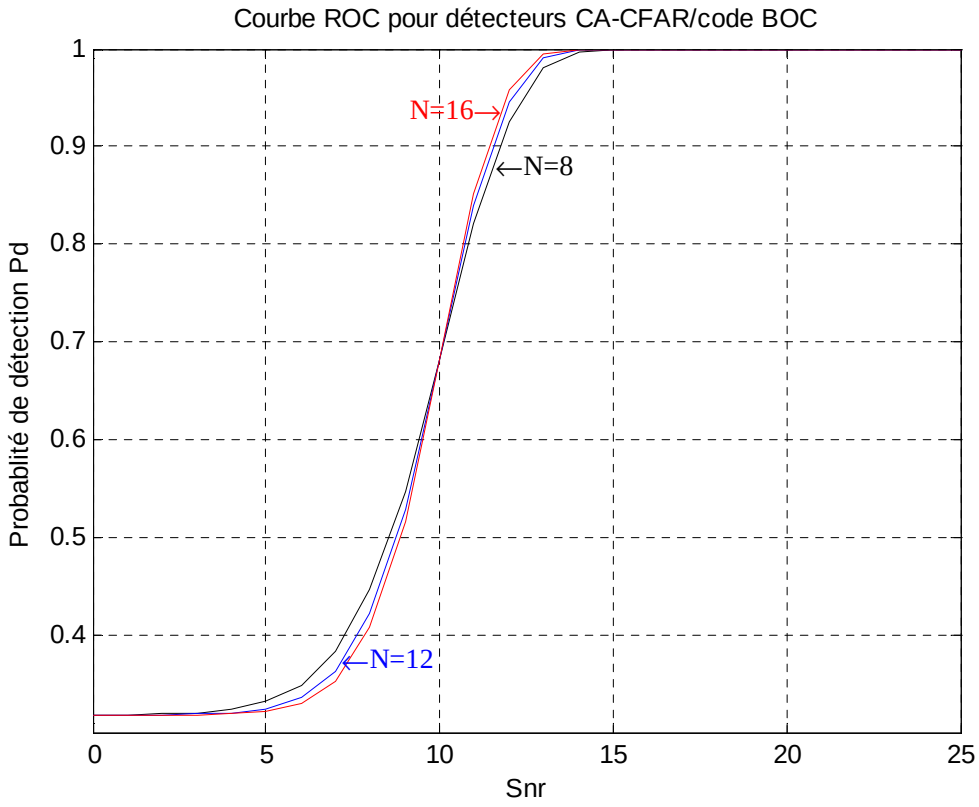


Figure III.19: Performance de détection pour un nombre de cellules n=8,12 et 16 dans le cas du code BOC

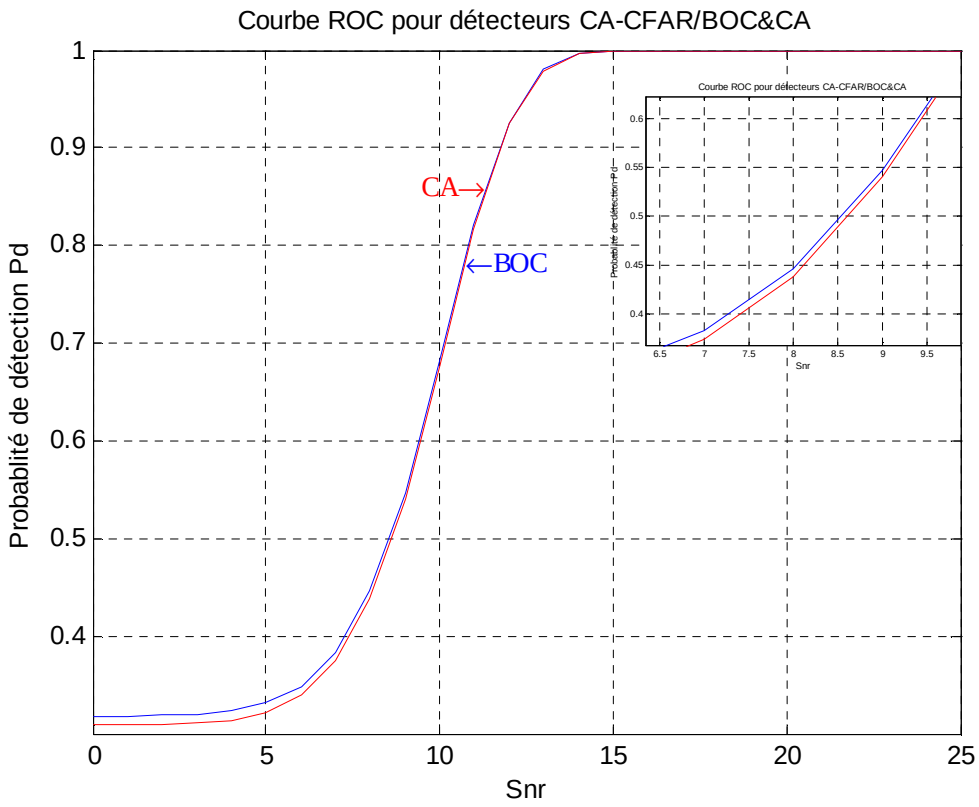


Figure III.20: Comparaison entre les deux codes C/A et BOC dans le cas du détecteur CA_CFAR

IV.1. INTRODUCTION :

Dans ce chapitre nous allons présenter la partie la plus importante de ce travail, à savoir la simulation sous Matlab des structures CFAR ainsi que leur réalisation sur circuit FPGA en utilisant le langage VHDL, des simulations et des interprétations des résultats seront proposées.

IV.2. STRUCTURES CFAR POUR L'ACQUISITION SOUS MATLAB :

Nous avons réalisé dans cette section la simulation de l'étage d'acquisition en utilisant les structures CFAR étudiées dans le chapitre précédent, afin de détecter la présence en premier lieu d'un code C/A puis on s'intéresse au code BOC. Ces simulations visent aussi de montrer l'efficacité des détecteurs adaptatifs afin d'éliminer d'éventuels multitrajets.

La figure IV-1 illustre la sortie d'acquisition obtenue en utilisant un signal composé d'un code C/A décalé de 400 chips (400µs) et une porteuse calée sur la fréquence intermédiaire affectée d'un Doppler de 500Hz ; On note aussi la présence du bruit. On observe bien que l'acquisition circulaire permet d'obtenir les deux valeurs de décalage et de fréquence à condition de détecter exactement la position du pic.

La figure IV-2, montre d'une façon similaire que la précédente, le résultat d'acquisition, mais cette fois pour un signal composé d'un code BOC décalé de 1000 chips (500µs) et affecté d'un Doppler de 500Hz. De même, on observe l'efficacité de l'acquisition circulaire.

Les deux figures IV-3 et IV-4, sont obtenues en utilisant des détecteurs CFAR placés à la sortie de l'acquisition circulaire. On observe dans les deux cas le seuil adaptatif superposé au signal résultant. Ce seuil est matérialisé dans le cas des détecteurs CA, GO et SO-CFAR, on observe que le pic est détecté dans les trois cas et que le détecteur CA-CFAR présente un pic plus haut en comparant avec le GO et le SO-CFAR.

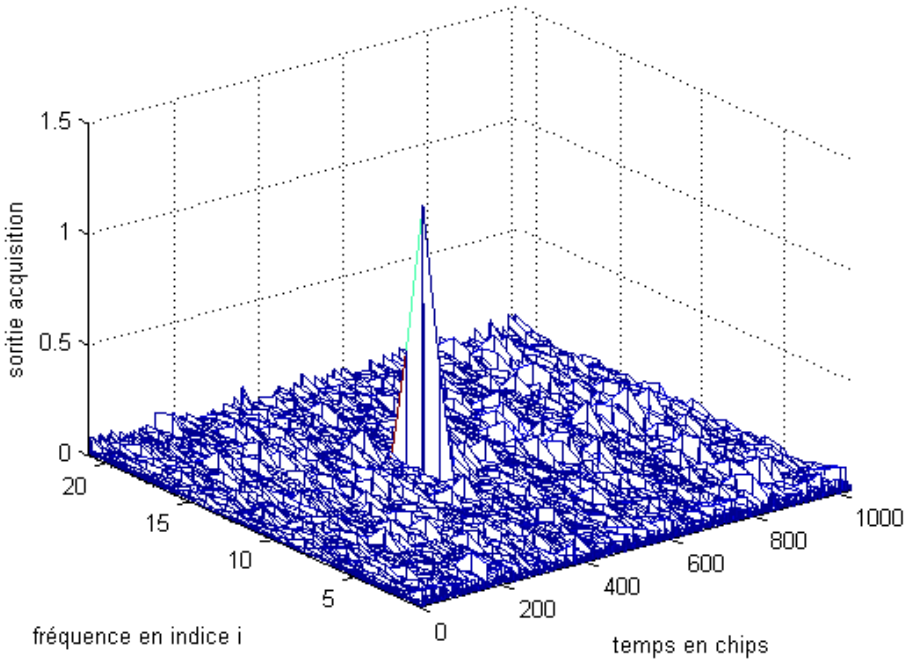


Figure IV.1 : *Acquisition d'un code C/A sans multi trajet*

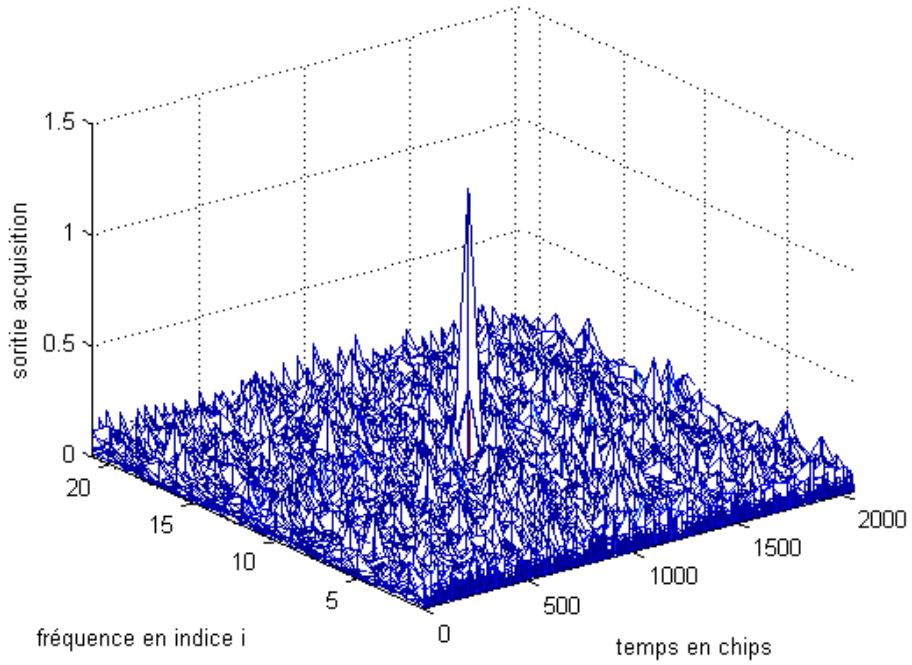


Figure IV.2 : *Acquisition du BOC(1,1) sans multi trajet*

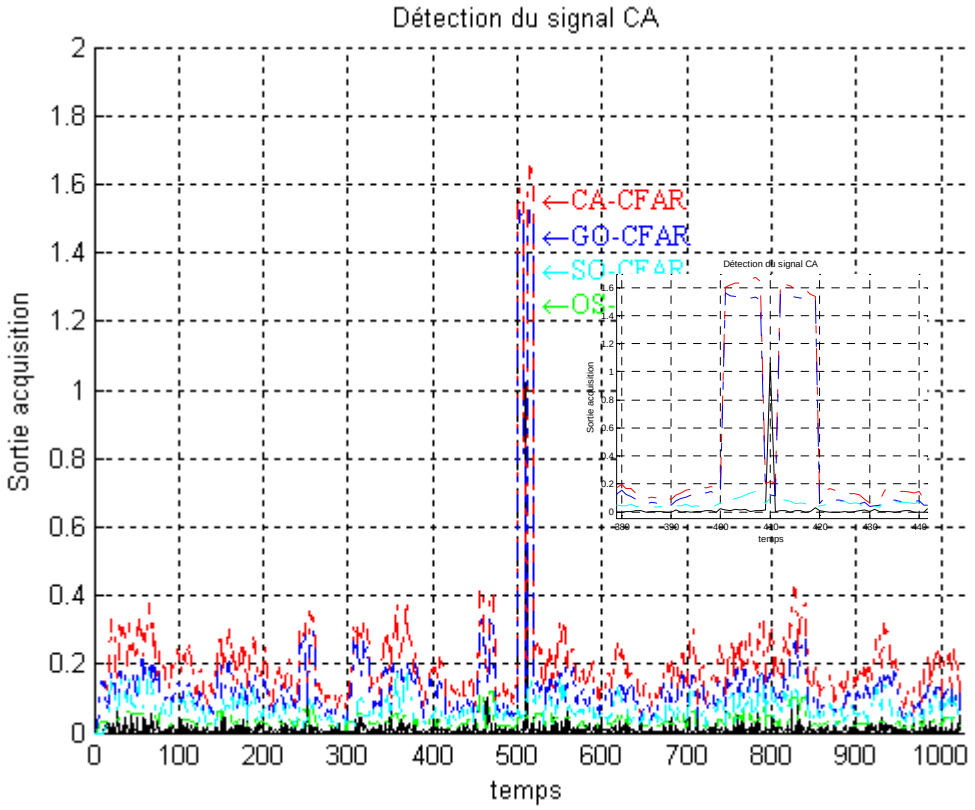


Figure IV.3: *Détection du code CA sans multi trajet*

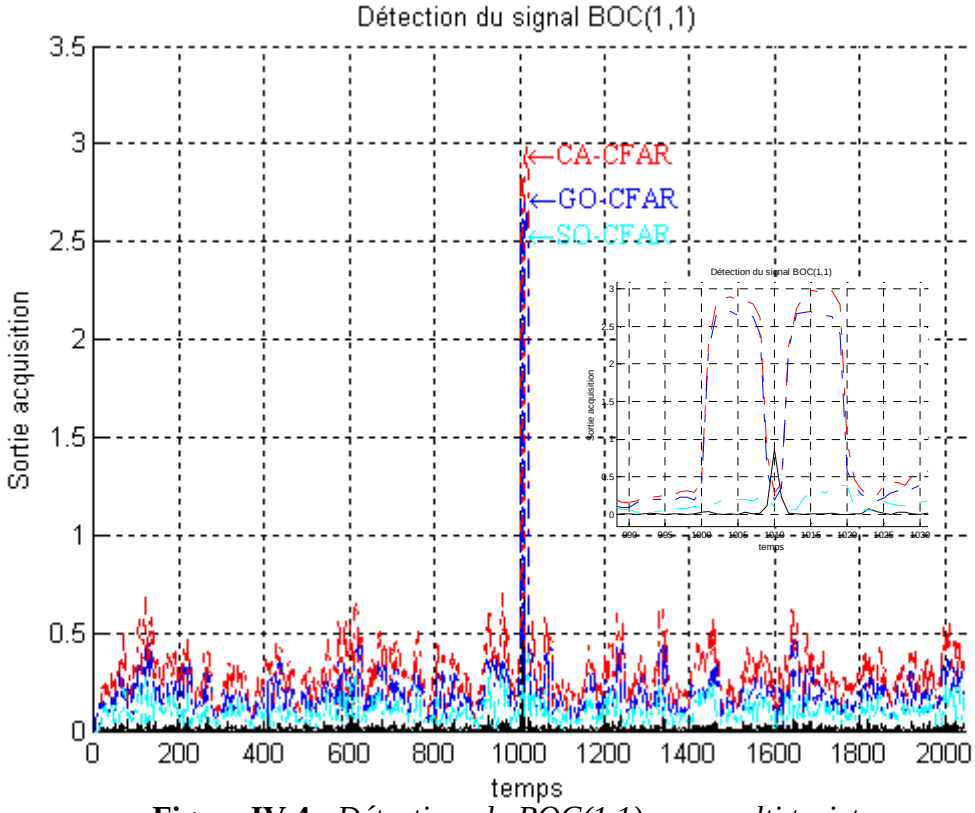


Figure IV.4 : *Détection du BOC(1,1) sans multi trajet*

Dans les applications de navigation par satellites, la navigation en milieu urbain semble d'être l'une des plus importantes. Cet environnement est caractérisé par la présence d'un grand Nombre d'obstacles qui produisent des trajets multiples. Nous proposons maintenant d'utiliser la détection CFAR afin de résoudre ce problème sachant qu'on s'intéresse aux multitrajets qui résultent des réflexions à partir d'obstacles situés à une distance supérieure à 300m.

La figure IV-5, montre l'influence d'un multitrajet décalé de trois chips par rapport au signal direct ce qui correspond à une distance de 900m, ceci en utilisant un code C/A. on constate l'apparition de deux pics, dont le premier de puissance plus importante correspond au signal direct et le deuxième faible correspond au multitrajet. On observe que l'acquisition circulaire donne un résultat exact concernant la position en temps et en fréquence des pics.

La figure IV-6, montre la même situation, mais dans le cas d'un code BOC. De plus, nous avons ajouté deux autres multitrajets, ce qui correspond à un total de trois.

Les figures IV-7&8 permettent également d'analyser les performances de détection CFAR, pour le cas du code C/A ainsi que pour le code BOC. On observe sur la figure 7, qui correspond au code C/A avec un seul multitrajet que les deux détecteurs CA et GO-CFAR peuvent détecter le signal direct tout en éliminant les multitrajets, contrairement au détecteur SO-CFAR. Tandis que la figure 8, illustre le cas du code BOC contaminé par trois multitrajets retardés de 3, 10 et 11 chips consécutivement. Nous observons, de même, que seulement le détecteur CA-CFAR a détecté le signal direct et il a éliminé les trois multitrajets, contrairement au cas du détecteur GO-CFAR, qui a détecté le signal direct mais aussi un seul multitrajet. Pour le détecteur SO-CFAR, il a détecté le signal direct mais il n'a pas réussi a éliminé les multitrajets.

Ces résultats, confirment l'efficacité de l'utilisation de la détection CFAR au niveau de l'étage d'acquisition du récepteur GALILEO. Nous pouvons conclure que le détecteur CA-CFAR et le mieux adapté à la navigation dans un milieu urbain.

On note qu'afin de simplifier la manipulation de nos programmes, nous avons réalisé une interface conviviale, regroupant les fonctions principales. Pour ce faire nous avons choisi le logiciel MATLAB. Notre interface regroupe l'ensemble des résultats obtenus et permet de choisir les différents paramètres du signal direct, multitrajet et des détecteurs CFAR.

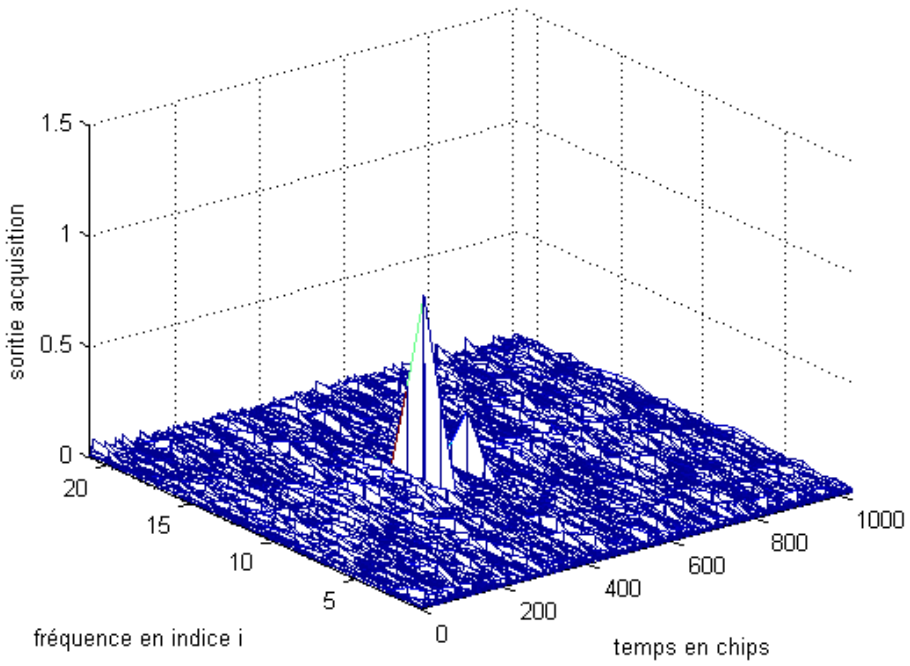


Figure IV.5 : *Acquisition d'un code C/A avec multi trajet*

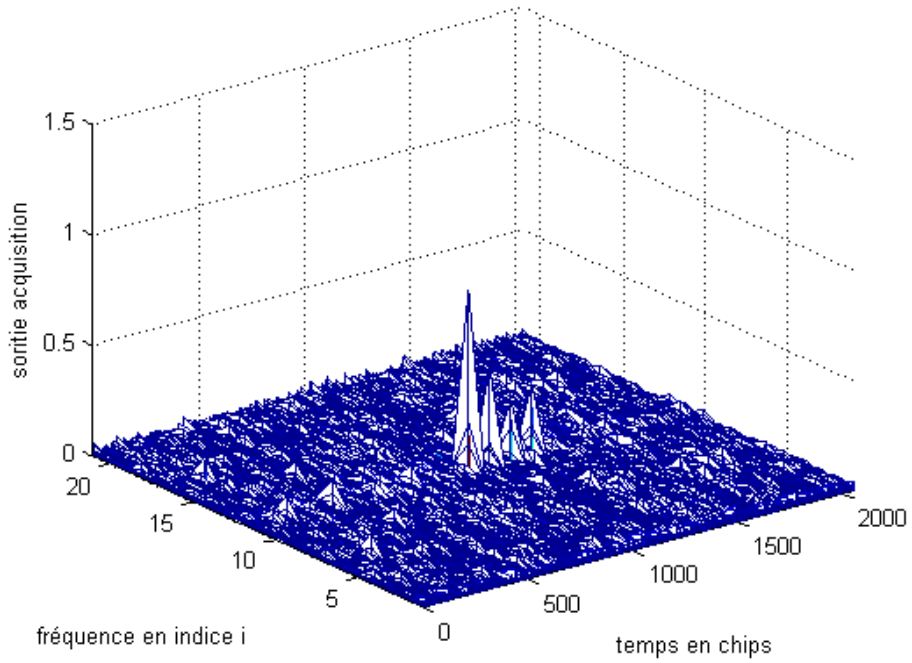


Figure IV.6 : *Acquisition du BOC(1,1) avec multi trajet*

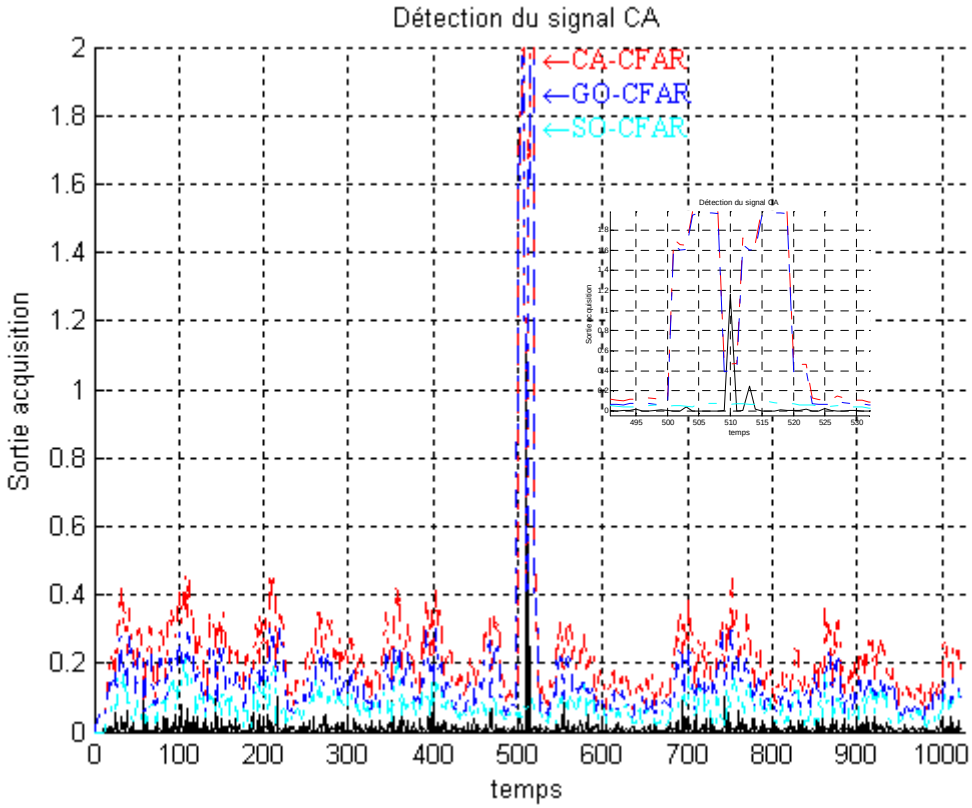


Figure IV.7 : *Détection CFAR d'un signal CA en présence de multi trajet*

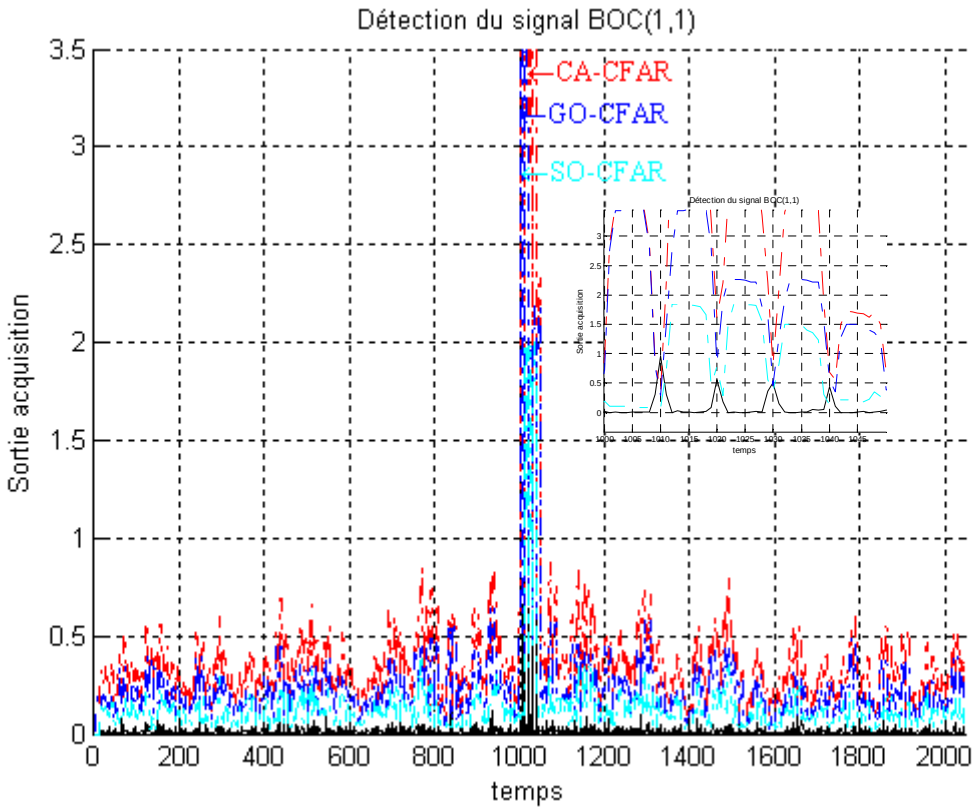


Figure IV.8: *Détection CFAR du BOC(1,1) en présence de multi trajet*

IV. 3. STRUCTURES CFAR EN UTILISANT LE VHDL :

IV.3.1. Les étapes nécessaires au développement d'un projet sur FPGA :

Deux outils sont indispensables pour la réalisation d'un projet sur *FPGA*; le simulateur et le synthétiseur. Comme son nom l'indique, le simulateur permet de simuler la description *VHDL* avec un fichier de simulation appelé « *test-bench* », ce dernier comprend les instructions *VHDL*. Le synthétiseur par contre traduit les instructions *VHDL* en fonctions logiques. L'outil nécessaire pour finaliser le travail est celui de placement et de routage. Les différentes étapes de la réalisation d'un projet *FPGA* sont résumées dans la figure suivante :

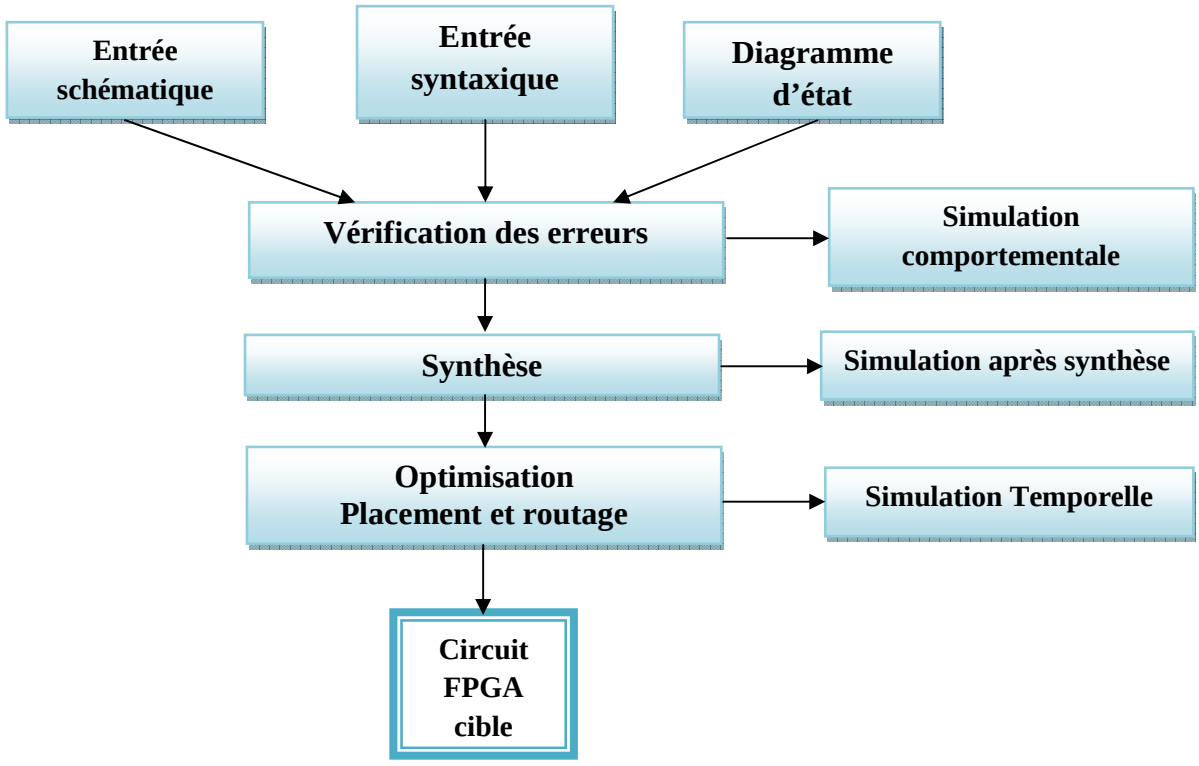


Figure IV.9 : *Etapes nécessaires au développement d'un projet FPGA*

IV.3.1. 1.Rédaction du texte VHDL :

Un langage de description Hardware (*HDL*) offre la possibilité de description à plusieurs niveaux dans laquelle les portes élémentaires et les *netlists* peuvent être utilisés avec une description fonctionnelle. Cette variation de niveau permet de décrire l'architecture du système à un niveau d'abstraction plus élevée, puis d'une façon incrémentale raffiner l'implémentation du design jusqu'au niveau des portes.

La description *HDL* offre les avantages suivants : la fonctionnalité du design peut être vérifiée immédiatement ce qui permet d'évaluer les décisions architecturales, une description *HDL* est lue et comprise plus facilement qu'une *netlist* ou une description schématique et elle constitue une documentation du design et de sa fonctionnalité indépendante de la technologie, et sans oublier qu'une description *HDL* facilite la manipulation de larges designs par rapport à une description schématique.

Les instructions *VHDL* seront rédigées sur le logiciel « *Xilinx ISE* ». La première étape est la création du projet, puis l'inclusion des fichiers sources dans lesquels seront écrites les instructions *VHDL*. Le logiciel comporte tous les outils nécessaires pour la réalisation d'un projet pouvant être implémenté sur circuit *FPGA*.

IV.3.1. 2.Vérification des erreurs :

Cette étape permet de vérifier la syntaxe du texte *VHDL* saisi, elle est réalisé en appuyant sur le bouton « *check syntax* ». Elle fournira les erreurs commises dans le texte ainsi que les alarmes liées au programme. Un texte *VHDL* contenant des erreurs ne peut être synthétisé mais s'il est doté d'alarmes, il sera synthétisé. C'est dans cette étape que le fichier « *netlist* » est généré.

IV.3.1. 3.La synthèse :

Après la création du design en *HDL*, il faut le synthétiser. Durant la synthèse, les informations comportementales dans le fichier *HDL* sont traduites en *netlist* structurale, et le design est optimisé pour les composants de Xilinx. L'outil de synthèse utilisé par *ISE* est *XST* ou *Xilinx Synthesis Tool*.

Comme il a déjà été cité, le rôle du synthétiseur est de convertir le projet décrit en VHDL en une structure du type *FPGA* cible constituée de portes logiques et bascules. Pour visualiser le circuit après synthèse, nous utilisons l'outil « *View RTL Schematic* ».

Pour forcer le système à travailler selon certaines limitations temporelles ou de surfaces, des contraintes sont utilisées. Les contraintes peuvent être introduites manuellement, ou à l'aide de l'éditeur de contraintes, *floorplanner* ou *FPGA editor* (outils accompagnant *ISE*). Pour évaluer le circuit après implémentation sous ces contraintes *Timing Analyser* ou *TRACE* peuvent être utilisés.

- ***Contraintes de Mapping***

Pour spécifier le *mapping* d'un bloc logique dans les *CLBs*, les contraintes *FMAP* ou *HMAP* (pour certaines familles) peuvent être utilisées. Une utilisation excessive de telles contraintes peut rendre le routage difficile ou même impossible.

- ***Contraintes de Placement***

Le placement d'un bloc peut être forcé à une certaine position donnée. La location peut être spécifiée dans la description schématique, dans l'outil de synthèse, ou dans un fichier UCF (*User Constraints File*). Le mauvais placement d'un bloc peut empêcher le design d'être placé et routé complètement. Typiquement, seuls les blocs d'entrée/sortie requièrent des contraintes de placement pour qu'ils soient dans les bonnes positions par rapport aux pins.

- ***Contraintes temporelles***

C'est pour spécifier les contraintes temporelles pour limiter le temps de propagation des signaux le long des chemins de traitement. L'outil de placement et de routage PAR utilise ces contraintes pour achever une performance optimale durant l'opération de placement et routage.

IV.3.1. 4.La simulation :

La simulation est une étape très importante, elle permet de vérifier le comportement du circuit réalisé avant ou après implémentation sur la carte *FPGA*. Le logiciel pouvant être utilisé comme simulateur est le « *ModelSim simulator* ». Cette étape permet également de connaitre le temps de propagation des signaux sur le futur circuit *FPGA*, elle valide aussi l'application du circuit sur le circuit cible.

IV.3.1. 5.Optimisation, placement et routage :

Le but de l'optimisation est d'optimiser le circuit, c'est-à-dire, minimiser les temps de propagation des signaux et l'espace occupé par la structure sur la carte *FPGA*. Le placement

et le routage permet de tracer les routes à suivre pour réaliser le fonctionnement tracé pour le circuit. L'outil de placement et de routage est le « *FPGA Editor* » de visualiser et d'éditer le circuit routé.

IV.3.1. 6.Programmation du composant et test :

C'est la dernière étape à réaliser, elle est appelée (*Generate Programming Files*), grâce à cette étape, on génère le fichier à charger sur la carte *FPGA*.

IV.3. 2.Réalisation des structures CFAR :

Nous décrirons la structure de nos programmes *VHDL* de chaque circuit, et nous présenterons par la suite les résultats de la simulation.

D'après les notions citées concernant la détection adaptative dans le deuxième chapitre, une structure *CFAR* se compose généralement d'un registre à décalage, de deux accumulateurs, des comparateurs donnant soit le maximum ou le minimum, d'un multiplieur et d'un comparateur offrant une impulsion à la présence d'une cible.

Dans nos programmes nous avons utilisé un registre à décalage 16 bits, ayant 19 cases dont seize sont dédiées à représenter deux sous registres de gauche et de droite (chacun huit cases), la cellule centrale sera la cellule de test et les deux cellules restantes seront des cellules de garde. Ce registre est programmé de tel sorte à offrir à sa sortie les sommes de gauche et de droite qui seront utilisées ensuite pour le calcul du seuil. La carte FPGA utilisée est la « *Virtex II Pro* ».

IV.3.2.1. Le CA-CFAR :

- ***Description du programme écrit en VHDL***

Nous utilisons le registre à décalage décrit ci-dessus qui offrira la somme des cases de gauche et de droite, un multiplieur pour multiplier la somme par le facteurK_0, le résultat du produit sera comparé avec la séquence contenue dans la cellule de test par un comparateur qui donnera 1 à chaque présence d'un pic.

➢ *Synthèse*

Pendant l'étape de synthèse, le synthétiseur convertit le programme *VHDL* en portes logiques et bascules de base, c'est-à-dire en structure électronique. L'outil « *View RTL Schematic* » permet de visualiser les schémas équivalents générés. Voici le schéma généré pour la structure CA_CFAR.

La première vue globale du circuit CA_CFAR générée par le synthétiseur est la suivante :

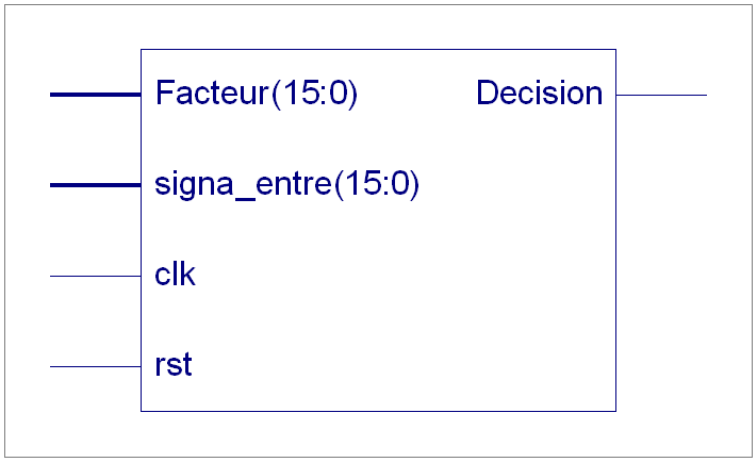


Figure IV.10 : *Vue générale du CA_CFAR généré par le synthétiseur*

Le circuit comprend quatre entrées : Le facteur de multiplication du seuil, le signal d'entrée, le « *clock* » et le « *reset* ». La figure suivante montre la structure interne du circuit CA_CFAR, nous en observons le registre à décalage, le comparateur et le multiplieur. L'élément FDE est une bascule de passage des données d'entrée.

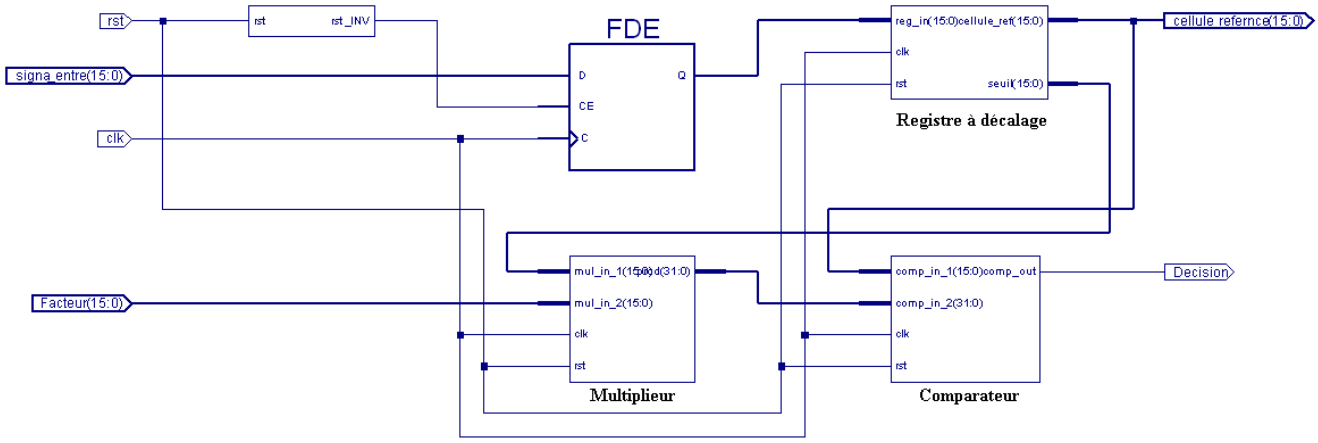


Figure IV.11 : *Les différents blocs du CA_CFAR générés par le synthétiseur*

Les structures de chaque bloc du CA_CFAR générée sont illustrées sur les figures suivantes :

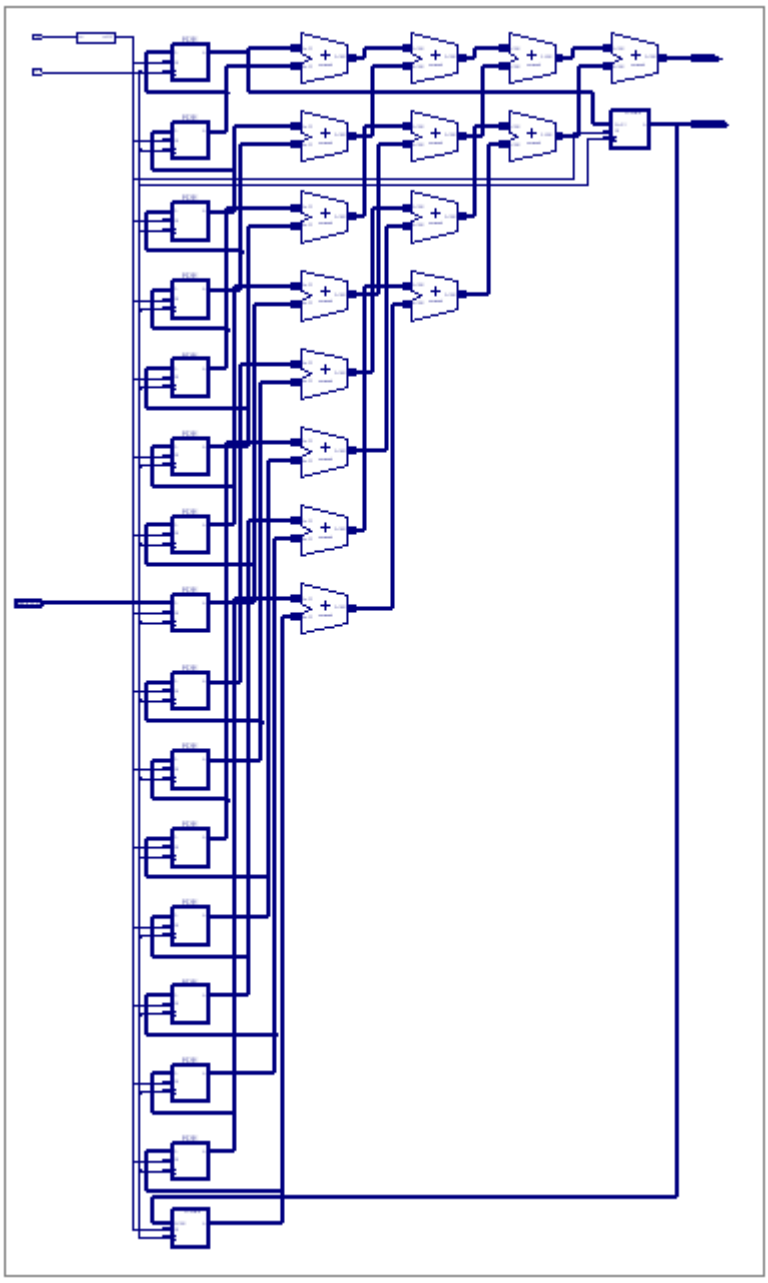


Figure IV.12 : *Structure interne du registre à décalage du CA_CFAR*

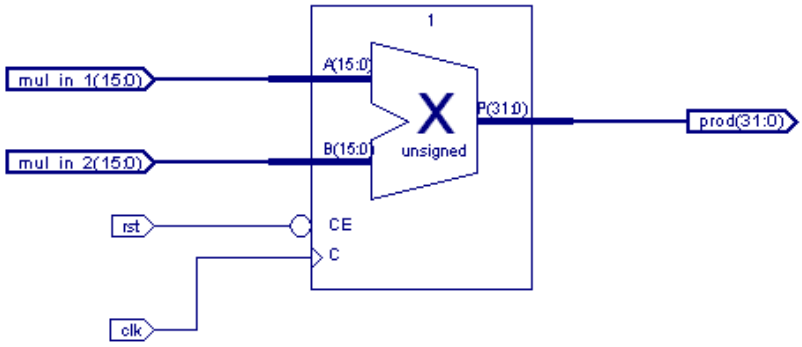


Figure IV.13 : *Structure interne du multiplieur du CA_CFAR*

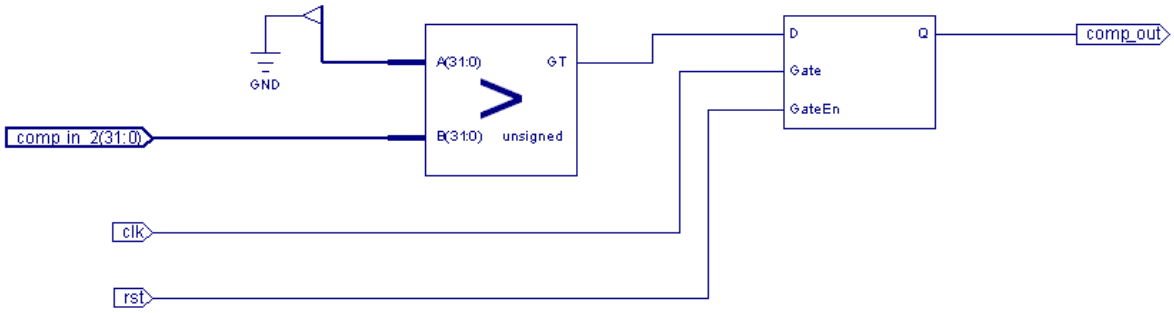


Figure IV.14 : *Structure interne du comparateur du CA_CFAR*

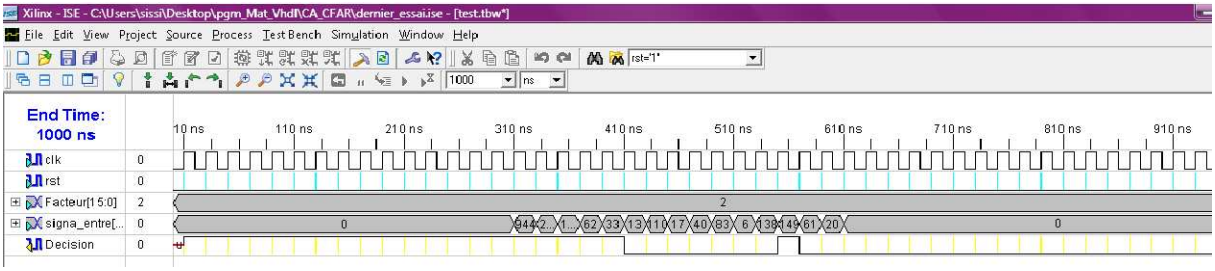
Les caractéristiques du circuit réalisé ainsi que la liste des ressources utilisées pour sa réalisation sont données dans le tableau suivant, ces informations sont fournies à la fin de la synthèse.

Ressources utilisées	Le nombre utilisé	Le pourcentage
Slices	232	16%
Bascules «Flip-flops »	225	7%
LUT à quatre entrés	224	7%
IOB	35	25%
Multiplieurs 18 x 18	1	8%
Horloges	1	6%

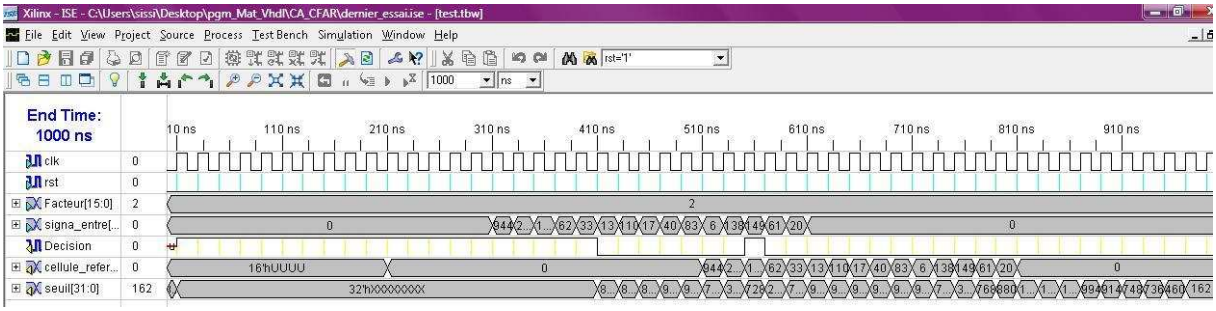
Tableau IV.1 : *Ressources du circuit FPGA utilisées pour le circuit CA_CFAR*

➢ *Simulation*

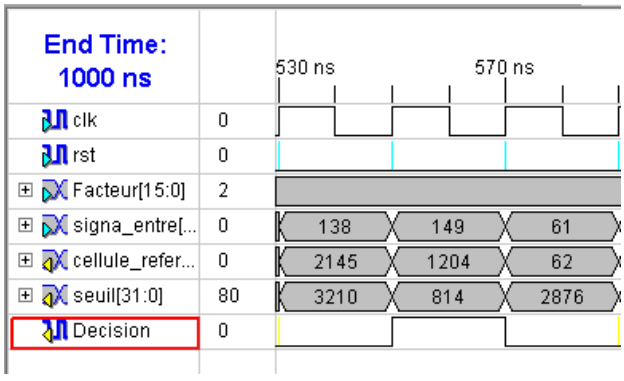
Comme il a été déjà cité, cette étape permet de visualiser les résultats. Nous intégrons un échantillon des résultats de la corrélation d'un signal GALILEO obtenus dans le Matlab simulés auparavant dans les chapitres précédents. Les résultats de la simulation sont sur la figure IV.15.a. Nous affichons le contenu de la cellule de référence et le seuil pour valider les résultats, la figure IV.15.b l'illustre. La figure IV.15.c montre le zoom de la partie où apparait le « 1 ».



a)



b)



c)

Figure IV.15 : *Simulation des résultats sur le CA_CFAR*

Nous constatons l'apparition d'un « 1 » à la sortie du comparateur, cela correspond au dépassement de la valeur du signal à celle du seuil calculé.

IV.3. 2.2. Le GO-CFAR

➢ *Description du programme écrit en VHDL*

Le registre à décalage utilisé cette fois ci donnera seulement les sommes de gauche et de droite. Un comparateur offrira par la suite le maximum de ces sommes, le multiplieur sert à multiplier le signal issu du comparateur par le facteurK_0, le résultat du produit sera comparé avec la séquence contenue dans la cellule de test par un comparateur qui donnera 1 ou 0.

➢ *Synthèse*

La syntaxe sera vérifiée pendant cette étape, le circuit sera synthétisé et le synthétiseur générera le circuit de la figure suivante :

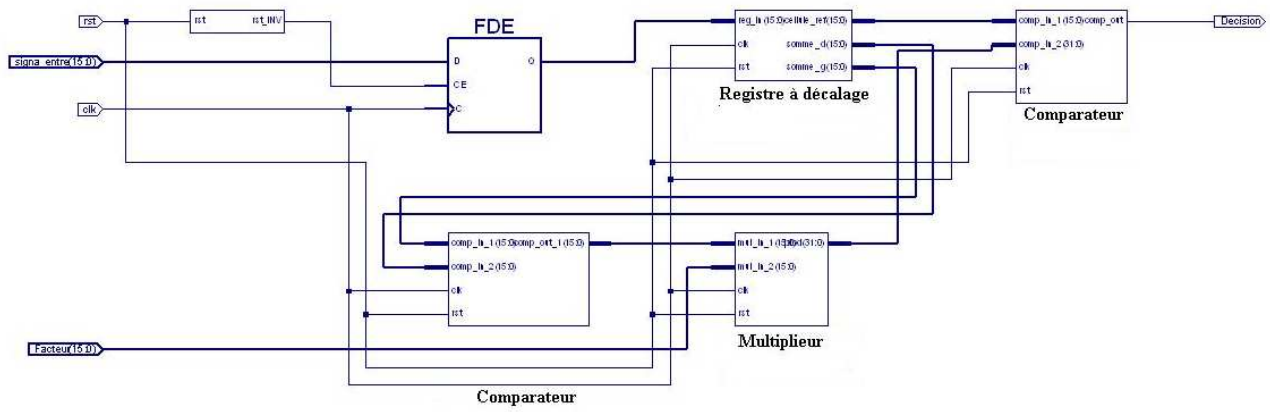


Figure IV.16 : *Structure du GO_CFAR généré par le synthétiseur*

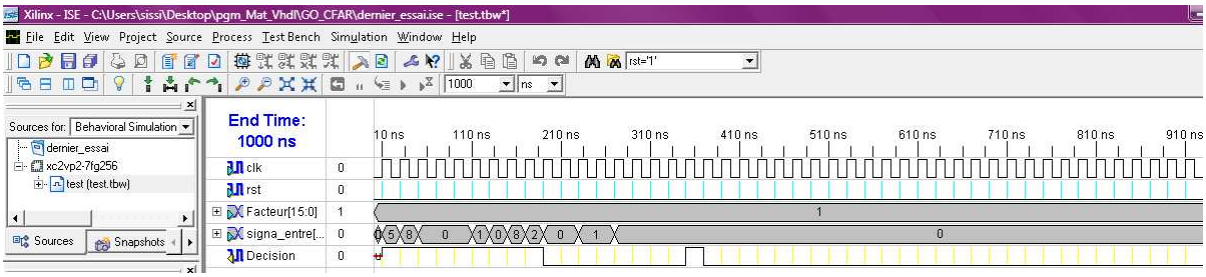
La liste des composants utilisés de la carte sont résumés dans le tableau :

Ressources utilisées	Le nombre utilisé	Le pourcentage
Slices	310	22%
Bascules «Flip-flops »	305	10%
LUT à quatre entrés	331	11%
IOB	35	25%
Multiplieurs 18 x 18	1	8%
Horloges	2	12%

Tableau IV.2 : *Ressources du circuit FPGA utilisées pour le circuit GO_CFAR*

➢ ***Simulation***

Nous intégrons une séquence des valeurs de l'acquisition du signal BOC(1,1) dans le signal d'entrée, nous aurons les résultats suivants :



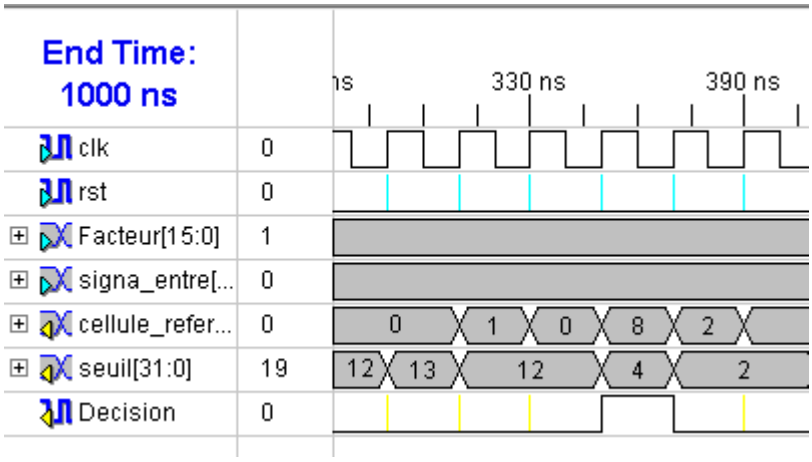


Figure IV.17 : *Simulation des résultats sur le GO_CFAR*

Quand la séquence dans la cellule de référence dépasse le seuil adaptatif, le comparateur donnera « 1 ».

IV.3. 2.3. Le SO-CFAR

➢ ***Description du programme écrit en VHDL***

Dans cette structure un comparateur donnera le minimum des deux sommes latérales. Le résultat de la comparaison sera multiplié par le facteur et le comparateur final donnera le 1 à la présence d'une cible.

➢ *Synthèse*

Le circuit généré pour le SO_CFAR est le suivant :

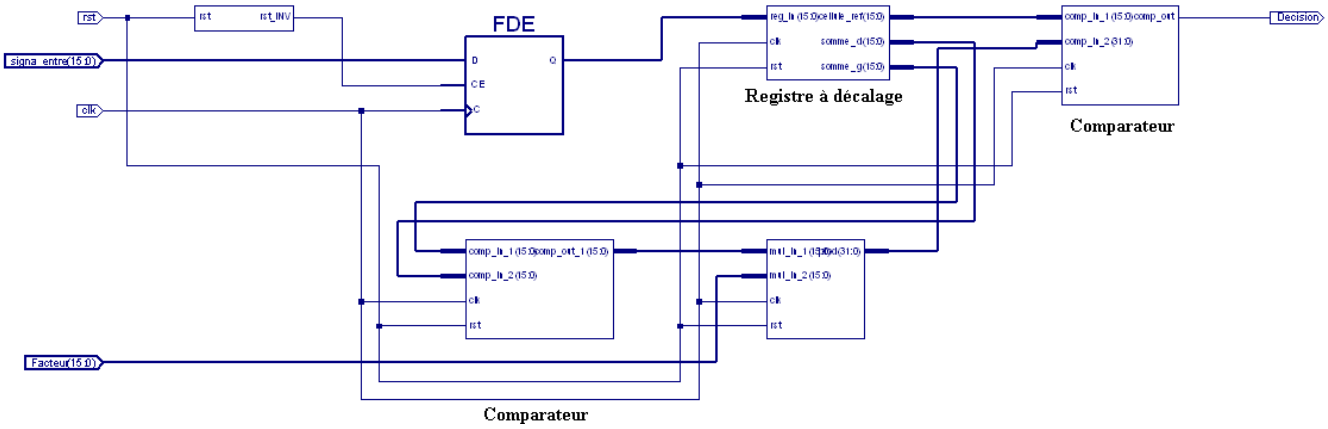


Figure IV.18 : *Structure du SO_CFAR généré par le synthétiseur*

La liste des ressources utilisées de la carte ainsi que le pourcentage associé sont résumés dans le tableau suivant :

Ressources utilisées	Le nombre utilisé	Le pourcentage
Slices	310	22%
Bascules «Flip-flops »	305	10%
LUT à quatre entrés	331	11%
IOB	35	25%
Multiplieurs 18 x 18	1	8%
Horloges	2	12%

Tableau IV.3 : *Ressources du circuit FPGA utilisées pour le circuit SO_CFAR*

IV.3.3. Combinaison des trois structures CFAR par un multiplexeur :

Nous allons proposer dans cette partie un circuit global qui contient les trois structures CFAR commandé par un multiplexeur, une sélection sera donc faite à l'entrée du circuit pour choisir la structure désirée, soit le CA_CFAR, le SO_CFAR ou le GO_CFAR.

- ***Synthèse***

Le circuit généré par le synthétiseur est le suivant :

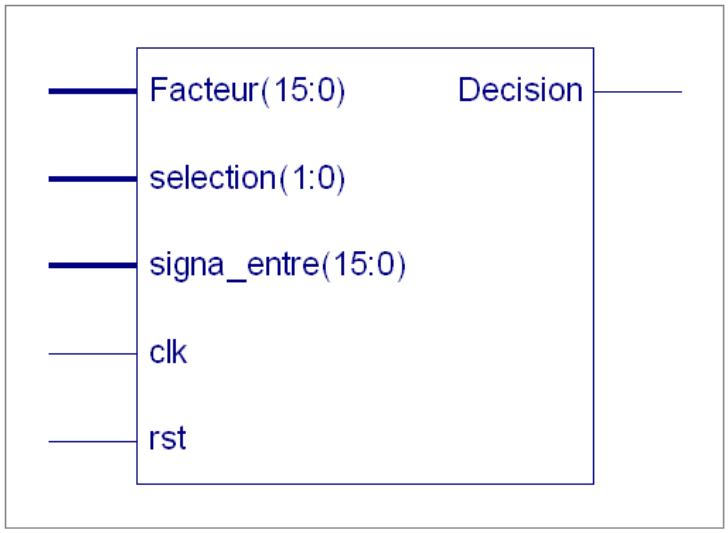


Figure IV.19 : *Structure* *du CFAR généré par le synthétiseur*

La figure suivante montre la structure interne du circuit :

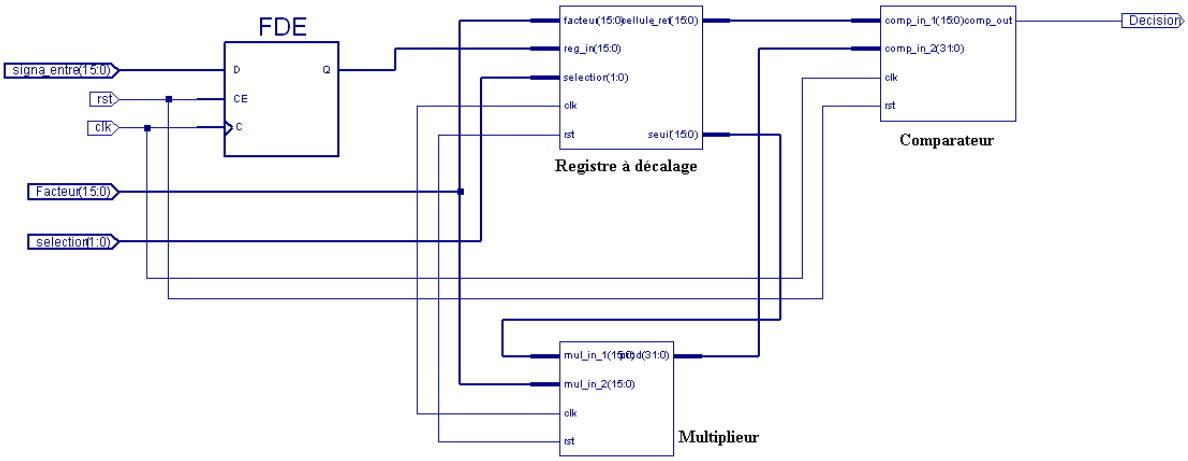


Figure IV.20 : *Structure* *interne du CFAR généré par le synthétiseur*

Le tableau suivant contient les caractéristiques du circuit réalisé ainsi que la liste des composants utilisés de la carte.

Ressources utilisées	Le nombre utilisé	Le pourcentage
Slices	276	19%
Bascules «Flip-flops »	273	9%
LUT à quatre entrés	364	12%
IOB	37	26%
Multiplieurs 18 x 18	1	8%
Horloges	2	12%

Tableau IV.4 : *Ressources du circuit FPGA utilisées pour le circuit CFAR*

Conclusion

L'étude du système de positionnement par satellites européen GALILEO montre qu'il sera une solution pour les problèmes que rencontrent les systèmes actuels. Par la diversité et la disponibilité dans les milieux urbains de ses signaux ainsi que leur précision, il sera adopté pour plusieurs applications, il promet d'offrir une multitude de services performants inexistants et imbattables actuellement.

Nous avons présenté dans ce projet, une technique récemment utilisée dans les récepteurs GPS afin de l'utiliser dans un récepteur Galileo, elle consiste à utiliser un détecteur CFAR au niveau de l'étage d'acquisition du signal satellitaire. L'avantage de cette technique, une fois réalisée, est qu'elle arrive à détecter le signal direct quelque soit le nombre, les retards, et les amplitudes des signaux multitrajets situés à des distances supérieures à trois cents mètres.

Aussi, nous avons étudié et analysé dans ce contexte, différentes structures CFAR, nous avons retenu les trois détecteurs CA, GO et SO-CFAR pour leur performance de détection. Les résultats ont démontré que les deux structures CA et GO sont meilleures pour la détection du signal dans un milieu bruité. La structure CA-CFAR était la meilleure pour la détection dans un milieu urbain puisqu'elle est la plus adaptée pour l'élimination des multi trajets.

Puis en deuxième partie, nous avons intégré dans un circuit programmable de type FPGA les différentes structures CFAR étudiées, plus précisément les détecteurs CA, GO et SO-CFAR. Pour cela une première étude sur les outils existants a été menée. Le composant choisi est un FPGA «Virtex2P »pour des raisons de disponibilité et de flexibilité. À l'aide du logiciel XILINX, nous avons implanté les fichiers synthétisables du modèle VHDL. L'étape de synthèse nous a montré rapidement que la capacité d'intégration du composant était suffisante pour traiter toutes les branches du détecteur considéré, de plus, nous utilisons 86% de la capacité d'intégration du FPGA.

Finalement, nous proposons, pour un travail futur d'étudier, analyser et implémenter des détecteurs CFAR, cette fois au niveau de l'étage de poursuite afin de réaliser la synchronisation des signaux reçus.

Bibliographie

[1] Mourad Barkat, Signal Detection and Estimation, *ARTECH HOUSE*, 2005.

[2] Volnei A. Perdroni, Circuit Design with VHDL, 2004.

[3] Ivan Garvanov, Lyubka Doukovska, Vladimir Kyovtorov, Christo Kabakchiev, Comparative analysis of CFAR structures for GPS signals in conditions of intensive urban pulse interference.

[4] Uwe Meyer-Baese, Digital signal processing with FPGA,edition Springer,

[5] BENMOSBAH Amine, MECHERAOUI Choukri Adel, Implémentation sur FPGA des méthodes MPPT : "P&O" et "floue optimisée par les Algorithmes Génétiques", Juin 2006.

[6] Hong Gil Kim, Iickho Song, Sun Yong Kim, Jung Hoon Lee, and Suk Chan Kim, adaptive code acquisition using a GO-CFAR processor in DS/CDMA systems

[7] B.Wei, MY. Sharif, TD. Binnie, AEA. Almaini,Adaptive PN code acquisition in multi-path spread spectrum communications using FPGA

[8] B.Wei, MY. Sharif, AEA.Almaini, TD.Binnie, PN code acquisition with a CA-CFAR adaptive digital matched filter and its realisation using FPGA

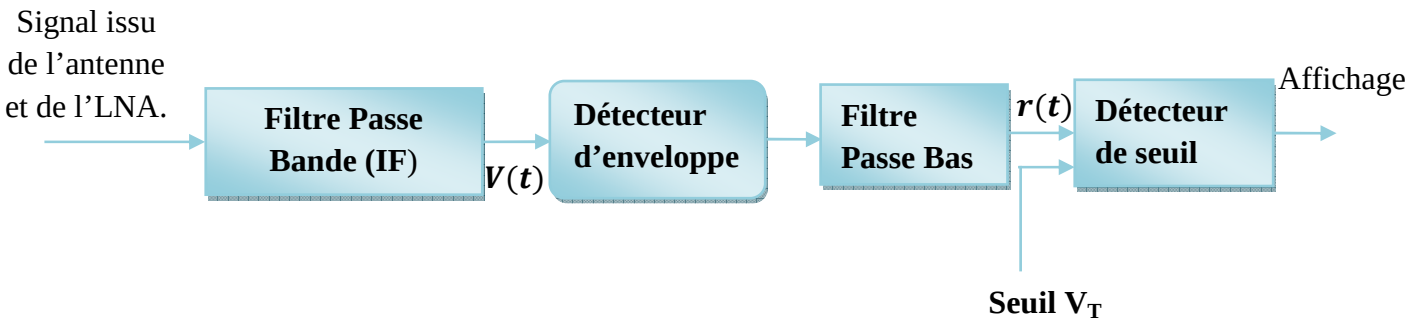
[9] Jacques Weber, Maurice Meaudre, Le langage VHDL cours et exercices, *édition DUNOD,*

[9]GALILEO un enjeu stratégique scientifique technique, Juin 2003.

[11]A constant false alarm rate (CFAR) detector for RADARSAT-2 along-track interferometry, 2005.

Détection à seuil fixe

Un récepteur se basant sur le principe de détection est généralement structuré comme suit :



Le signal provenant de l'antenne et de l'amplificateur à faible bruit est composé de l'écho $s(t)$ (signal retournant de la cible vers le radar) et d'un bruit additif $n(t)$ supposé dans la plus part des cas blanc, gaussien de moyenne nulle et de variance Ψ^2.Le signal provenant du filtre IF $v(t)$ peut s'écrire comme suit :

$$v(t) = v_I(t)\cos\omega_0 t + v_Q(t)\sin\omega_0 t = r(t)\cos(\omega_0 t - \varphi(t)) \qquad (1)$$

Avec :

$$v_I(t) = r(t)\cos\omega_0 t$$
$$v_Q(t) = r(t)\sin\omega_0 t \qquad (2)$$

Sachant que $\omega_0 = 2\pi f_0$ est la pulsation du signal dont f_0 est la fréquence (fréquence de travail du radar, $r(t)$est l'enveloppe du signal $v(t)$ et les indices I et Q désignent respectivement les composantes en phase et en quadrature de phase.

Une cible est dite détectée quand l'enveloppe r(t) dépasse le seuil V_T, d'où les hypothèses possibles :

$s(t) = n(t) > V_T$: Détection.
$n(t) > V_T$: Fausse alarme. (3)

La sortie du filtre IF est un processus aléatoire composé soit du bruit seul ou du bruit additionné au signal retournant de la cible.

Dans le premier cas, les composantes en phase et en quadrature de phase correspondantes sont :

$$v_I(t) = n_I(t)$$
$$v_Q(t) = n_Q(t) \quad (4)$$

Dans le cas où ces deux composantes sont considérées Gaussiennes, de moyennes nulles et de variances ψ^2 égales, la densité de probabilité des deux variables est donnée par :

$$f(n_I, n_Q) = \frac{1}{2\pi\Psi^2}\exp\left(-\frac{n_I^2+n_Q^2}{2\Psi^2}\right) = \frac{1}{2\pi\Psi^2}exp\left(-\frac{(r\cos\varphi-A)^2+(r\sin\varphi)^2}{2\Psi^2}\right) \quad (5)$$

La probabilité conjointe des deux variables r et φ est donnée par :

$$f(r,\varphi) = f(n_I, n_Q)[J] \quad (6)$$

Avec I$[J]$ est une matrice des dérivées définie par:

$$[J] = \begin{bmatrix} \frac{\partial n_I}{\partial r} & \frac{\partial n_I}{\partial \varphi} \\ \frac{\partial n_Q}{\partial r} & \frac{\partial n_Q}{\partial \varphi} \end{bmatrix} \begin{bmatrix} \cos\varphi & -r\sin\varphi \\ \sin\varphi & r\cos\varphi \end{bmatrix} \quad (7)$$

Le déterminant de cette matrice est appelé le Jacobéen, dans ce cas il vaut $r(t)$.

En remplaçant alors dans l'équation précédente nous obtiendrons la densité de probabilité des deux variables r et φ:

$$f(r,\varphi) = \frac{r}{2\pi\Psi^2}\exp\left(-\frac{r^2+A^2}{2\Psi^2}\right)exp\left(\frac{rA\cos\varphi}{\Psi^2}\right) \quad (8)$$

La probabilité liée à la variable r seule est donnée par :

$$f(r) = \int_0^{2\pi} f(r,\varphi)d\varphi = \frac{r}{\Psi^2}exp\left(-\frac{r^2+A^2}{2\Psi^2}\right)\frac{1}{2\pi}\int_0^{2\pi} exp\left(\frac{rA\cos\varphi}{\Psi^2}\right)d\varphi \quad (9)$$

L'intégrale dans cette équation est la fonction de Bessel d'ordre 0, d'où le résultat suivant :

$$f(r) = \frac{r}{\Psi^2}I_0\left(\frac{rA}{\Psi^2}\right)exp\left(-\frac{r^2+A^2}{2\Psi^2}\right) \quad (10)$$

Cette dernière correspond à la densité de probabilité de Rice.

Dans le cas où nous considérons le bruit seul, $\frac{A}{\Psi^2} = 0$ elle devient une distribution de Rayleigh, soit :

$$f(r) = \frac{r}{\Psi^2} exp\left(-\frac{r^2}{2\Psi^2}\right) \quad (11)$$

Quand par contre le rapport $\frac{A}{\Psi^2}$ est très large elle sera une distribution Gaussienne de moyenne A et de variance Ψ^2, soit :

$$f(r) \approx \frac{1}{\sqrt{2\pi\Psi^2}} \exp\left(-\frac{(r-A)^2}{2\Psi^2}\right) \quad (12)$$

La densité de probabilité de la variable φ est obtenue comme suit :

$$f(\varphi) = \int_0^r f(r, \varphi) dr \quad (13)$$

D'où :

$$f(\varphi) = \frac{1}{2\pi} exp\left(\frac{-A^2}{2\Psi^2}\right) + \frac{A\cos\varphi}{\sqrt{2\pi\Psi^2}} exp\left(-\frac{(A\sin\varphi)^2}{2\Psi^2}\right) F\left(\frac{A\cos\varphi}{\Psi}\right) \quad (14)$$

Avec :

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{\varepsilon^2}{2}} d\varepsilon \quad (15)$$

Si nous considérons le bruit seul, la distribution correspondra à une distribution uniforme sur l'intervalle $[0\,, 2\pi]$.

- **Probabilité de fausse alarme :**

La probabilité de fausse alarme P_{fa} est définie comme la probabilité d'avoir $r(t)$ supperieur au seuil V_T quand le bruit seul existe. Elle est donnée par :

$$P_{fa} = \int_{V_T}^{\infty} \frac{r}{\Psi^2} exp\left(-\frac{r^2}{2\Psi^2}\right) dr = exp\left(\frac{-V_T^2}{2\Psi^2}\right) \quad (16)$$

D'où :

$$V_T = \sqrt{2\Psi^2 \ln\left(\frac{1}{P_{fa}}\right)} \quad (17)$$

➢ **Probabilité de détection :**

La probabilité de détection P_D est la probabilité que le signal $r(t)$ dépasse le seuilV_T, dans le cas de la présence du signal additionné de bruit. Elle est donnée par :

$$P_D = \int_{V_T}^{\infty} \frac{r}{\Psi^2} I_0\left(\frac{rA}{\Psi^2}\right) exp\left(-\frac{r^2+A^2}{2\Psi^2}\right) dr \quad (18)$$

Le développement de la probabilité de détection est compliqué, des formules approximatives liant la probabilité de détection à la probabilité de fausse alarme ont été mises en place dont :

$$P_D \approx F\left(\frac{A}{\Psi} - \sqrt{2\ln\left(\frac{1}{P_{fa}}\right)}\right) \quad (19)$$

Simulation de Monte-Carlo

La simulation de Monte-carlo est un outil mathématique de plus en plus utilisé dans les problèmes de détection, Elle est utilisé, principalement, quand il est trop complexe et surtout trop onéreux d'avoir une distribution exactes des observations ou lorsque les formules mathématiques des densités de probabilité des variables considérées n'existes pas. La simulation de Monte-Carlo permet dans ces cas de décrire la distribution d'une variable aléatoire et d'en calculer aussi son espérance. Afin de réaliser une approximation, on peut tester un nombre important d'observation issus de la structure de détection proposée afin de valider un modèle proche selon les principes de la loi des grands nombres. Bien évidemment ces démarches sont dorénavant possibles grâce à la puissance de nos ordinateurs.

La simulation de Monte-carlo, consiste à reproduire de nombreux échantillon de lois connues, pour ce faire, nous constituons des échantillons, nous effectuons nos calculs d'estimateurs (moyenne et écart-types) sur ces échantillons puis nous consolidons les résultats afin d'avoir une moyenne et un écart-type globaux. On peut résumer les étapes de cette méthode comme suite :

1- Tout d'abord, il faut simuler la distribution d'une loi uniforme.
2- Ensuite, il faut simuler la distribution de mon phénomène : A partir de la distribution uniforme obtenue précédemment, nous simulons la distribution du phénomène qu'on veut obtenir. En détection, nous simulerons le plus souvent des lois normales centrées réduites de paramètres 0 et 1. Ces lois subiront des traitements selon l'architecture de détection analysée.
3- Ensuite, nous simulons les paramètres importants de notre phénomène : en détection, pour simuler la probabilité de détection, on compte le nombre de dépassement du seuil et on le divise sur le nombre total d'échantillons.