REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA



Faculté de **T**echnologie

Département : génie électrique

Mémoire de fin d'études de Master

Filière : Electronique

Spécialité : Electronique des systèmes embarqués

Etude et implémentation sur FPGA d'un système pour la détection des défauts dans les roulements à billes

---

Présenté par :

ROUIBAH Abdelghani

RAHMI Mahfoudh

Proposé et Dirigé par :

LACHENANI SidAhmed

Promotion 2020

REMERCIEMENTS

Nous remercions ALLAH le tout-puissant de m'avoir donné le courage, la volonté et la patience de mener à terminer ce présent travail.

Nous tenons à remercier sincèrement Monsieur LACHNANI Sidahmed Pour son encadrement, sa compréhension, ses conseils, ses observations son aide et sa disponibilité durant la réalisation de ce mémoire.

Nous tenons à remercier également tous les membres de jury d'avoir accepté de juger ce modeste travail.

Nous tenons à remercier nos parents, nos frères et sœurs, nos proches et nos amis qui nous ont aidés à la réalisation de ce travail.

Dédicace :

Je dédie ce modeste travail à :

-Mon père.

-Ma mère.

-Mes frères et sœurs.

-Mes proches.

-Toute la famille (ROUIBAH).

-Mes collègues et à tous ceux qui m'ont aidés de près ou de loin durant toutes mes années universitaires.

Enfin, je dédie ce mémoire à tous ceux qui m'aiment et à tous mes amis que j'ai connus pendant ma période d'étude.

ROUIBAH Abdelghani

Dédicace :

Je dédie ce modeste travail à :

-Mon père.

-Ma mère.

-Mes frères et sœurs.

-Mes proches.

-Toute la famille (RAHMI).

-Mes collègues et à tous ceux qui m'ont aidés de près ou de loin durant toutes mes années universitaires.

Enfin, je dédie ce mémoire à tous ceux qui m'aiment et à tous mes amis que j'ai connus pendant ma période d'étude.

RAHMI Mahfoudh

الملخص :

في هذا العمل قمنا بوضع نظام لرصد والكشف عن بعض الاعطاب على مستوى محمل الكريات يعتمد على تحليل اشارات الاهتزاز الناتجة على عطب باتباع نضاريات معالجة ال اشارةFFT وكذلك Transformer de HILBETوقمنا بتنزيل هذا النظام على شريحة مصفوفة البوابات المبرمجة حلقياXILINIX FPGA

تم إنجاز العمل في بيئة MATLAB والمحاكاة في SIMULINK

الكلمات المفتاحية : الحاملة ، المحامل، الاهتزاز MATLAB، SIMULINK

Transformer de HILBET

Résumé :

Dans ce mémoire, nous avons étudié un système de détection des défauts de roulements, basé sur l'analyse des signaux de vibration résultant de chaque panne en suivant les théories du traitement du signal FFT ainsi que la transformé de Hilbert. Nous avons également implémenté ce système sur la carte FPGA.

Le travail a été fait dans un environnement MATLAB et la simulation par SIMULINK et **S**ystème **G**énérateur

Mots clés : Roulements, vibration, transformé de Hilbert, MATLAB, SIMULINK.

Abstract:

In this memory, we have installed a system to detect faults of bearings based on the analysis of vibration signals, resulting from every failure by following the antiquities of the signal processing FFF as well as Hilbert. We have also installed this system on the FPGA matrix of the circular programmed field of the American manufacturer Xilinx environment and the simulation done in SIMULINK and System Generator.

Key words: Bearings, vibration, Hilbert transform, MATLAB, SIMULINK.

Table des matières

Résumé……..... I
Table des matières…….…... II
Liste des figures……V
Liste des tableau……...……...VII
Liste des symboles…….. xI
Introduction générale……...…….…..1

Chapitre I : Les roulements et les signaux caractéristiques des défauts

I.1 Introduction :.…….....……2
I.2 Constitution d'un roulement :……2
I.2.1 La bague extérieure :……..3
I.2.2 La bague intérieure :…….....3
I.2.3 La cage :……..3
I.2.4 Eléments roulants :…….3
I.3 Les différents types de roulement :……3
I.3.2 Roulement à un rangée de billes à contact oblique :…….....4
I.3.3 Roulement à billes à contact radial :……..4
I.3.4 Les roulements à rouleaux :……...4
I.3.5 Roulements radiaux et axiaux :……..5
I.3.6 Les roulements à aiguilles :……6
I.3.7 Butées :……...6
I.3.8 Roulements standard et spéciaux :……7
I.4 Origine des défauts de roulement :……...7
I.5 Quelques défauts de roulement:……...8
I.5.1 Le grippage :.....8
I.5.2 La corrosion :.....8
I.5.3 Usure :……8
I.5.4 La fissuration des bagues :…….....8
I.5.5 L'écaillage :……9
I.6 Fréquences caractéristiques pour les défauts d'un roulement : 9

I.7.1 Classification des signaux :……..……....9
I.7.2 Types de signaux……..12
I.8 Analyse temporelle.....14
I.9 Analyse fréquentiel.....14
I.10 Transforme de Fourier, Transforme de Hilbert et le Signal analytique: 15
I.11 Conclusion : 17

Chapitre II : Les FPGAs et l'outil de développement

II.1 Introduction : 19
II.2 Les circuits programmable FPGA : 19
II.2. Définition..... …..19
II. 3 Structure interne :.....21
II. 3.1 Logique des FPGA : 22
II. 3.2 Interconnexion :.....23
II.3.3 Entrée et sortie :.....24
II.3.4 Horloge : 26
II.3.5 Configuration : 27
II. 4 MATLAB/Simulink: 28
II.5 System Générateur de Xilinix :.....28
II. 5.1 Configuration MATLAB et System Generator 28
II.5.2 Blocs Xilinx.....30
II.5.2.2 Blocs de générateur de système Xilinx.....30
II.5.2.2 Blocs Simulink 34
II.6 Conclusion 35

Chapitre III : simulation et implémentation

III.1 Introduction.....37
III.2 La base des données utilisées.....37
III.3 Schéma globale du programme.....38
III.3.1 Source 38
III.3.2 Affichage du résultat et interprétation 39
III.4 Programmation avec system générateur Xilinix 39
III.5 Les architecteurs proposées 39
III.5.1 Solution1 39

III.5.1.1 La description de Schéma de la solution simple 41

III.5.2 Solution 2 46

III.6 Discussion de résultat 49

III.7 Validation des résultats.....50

III.8 Implémentation sur la carte FPGA……..51

III.8.1 Compilation……51

III.8 .2 Conceptions de notre architecture sur vivado… ……51

III.9Utilisation des ressources ……..……53

III 10. Utilisation des ressources pour la solution optimisé…….55

III.11 Conclusion…….56

Conclusion générale…….57

Bibliographie……58

Liste des figures

Figure 1 :composantes d'un roulement 2
Figure I - 2 : Roulement à une rangée de billes à contact oblique 4
Figure I - 3 : Roulement à rouleaux cylindriques 5
Figure I -4: Roulement à rouleaux coniques [2] 5
Figure I -5: Roulement à rouleaux coniques [2] 6
Figure I - 6: Butées [2] 7
Figure I – 7 Grippage d'une bague [1] 8
Figure I – 8 Corrosion sur une piste de roulement |……...9
Figure I – 9 (a) écaillage de fatigue et (b) écaillage superficiel de la bague interne [1] 10
Figure I_10 Caractéristiques d'un roulement 11
Figure I.11 Classification des signaux.....12
Figure I_12 : signal Non périodique.....13
Figure I.13: signal aléatoire 13
Figure I_14 **:** (a) Spectre du signal continu x(t) et (b) Spectre du signal analytique z(t)…..…14
Figure II.1 : Structure interne du FPGA [5].....16
Figure II.2 : Structure interne simplifiée du FPGA [5]. 19
Figure II.3 : Une cellule logique est la pierre de fondation d'une FPGA [5]. 20
Figure II.4 **:** Bloc d'entrée / sortie générique [5]……..…...22
Figure II.6 **:**Configuration de FPGA basés sur la RAM statique[5]……..…….24
Figure II.5 : Réseaux de distribution d'horloge. À gauche :structure en H ; à droite : colonne vertébrale et structure des côtes [5].....25
Figure II.7 : Xilinx vivado 26
Figure II.8 : Xilinx System Generator.....27
Figure II.9 : Configuration MATLAB et System Generator Vivado. 28
Figure II.10 : blocs FFT v.9.1 31
Figure II.11 **:**XilinxSystemConstant……32
FigureII.12GatewayIn……..32
Figure II.13 : Gateway Out. 32
Figure II.14 : Simulink – Terminator 33
Figure II.15 **:** Xilinx System Generator……...33
Figure II.16 : Scope 33

Figure III.1 : Schéma synoptique du system global. 38
Figure III. Schéma globale de la Solution simple 40
Figure III. 3 : Fenêtre de bloc source.....41
Figure III.4: Fenêtre de bloc Gateway In Axi4. 41
Figure III.5 : signale brute. 42
Figure III.6 : Fenêtre de bloc FFT v.9.1 42
Figure III .7: sous-system(subsystem1).....43
Figure III.8 : FFT du signal(haut) et FFT sans les terme negatif (bas)……..….43
Figure III.9 : subsytem2.....44
Figure III .10:IFFT du signal(haut) et leur abs(bas).....44
Figure III.11: Sous-system(subsystem 3). 45
Figure III .12 : Spectre du signal de défaut sur la bague intérieur. 45
Figure III .13 : Schéma synoptique de solution optimisée et solution simple. 46
Figure III.14 : Solution optimisée. 47
Figure III .15: Subsystem1.....48
Figure III.16 : bloc FFT et subsystem.. 49
Figure III.17 : Spectre du signal de défaut sur la bague interne, base de donnée106 DE…….47
Figure III.18 : Spectre du signal de défaut sur la bague externe, base de donnée 130 DE…..50
Figure III.19 : Lancement de la génération……..…….51
Figure III .20 : architecture sur vivado hls de notre travaille hilbert8192……52
Figure III. 21. Utilisation des ressources de hilb_8192 par à port rest et périphérique axi.….52
Figure III.22: Resultat de l'implimentation sur la carte FPGA avant et apres.....……....52
Figure III. 23 bloc obtenu ave cod_simulation……54
Figure III. 24 : Utilisation des ressources de chaque bloc de hilb_8192……..54
Figure III. 25**.** Utilisation des ressources de hilb_optim ,par à rapport au reste et périphérique axi……..……56
Figure III. 26 **:** Utilisation des ressources de chaque bloc de hilb_optim.……..…56

Liste des tableaux

Tab II.1 : Organisation des bibliothèques de bloc Xilinx Symboles…..…….29

Tab III.1 : Teste de programme avec des fichiers porte un défaut connu……...…50

Tab III.2: : Utilisation des ressources du FPGA pour l'implémentation du hilb8192…….53

Tableau III.3 : Utilisation des ressources du FPGA pour l'implémentation du solution optim.....55

Symboles

FR: La fréquence de rotation.

dm : le diamètre du cercle primitif.

n : le nombre de billes.

db: le diamètre de la bille.

α : l'angle de contact.

BPFI : Fréquence de défaut interne, HZ.

BPFO:les fréquences de Défaut externe, HZ.

B : les fréquences de Défaut élément roulant, HZ.

T : période.

Abréviation

FFT :(Fast Fourier Tranform) Transforme de Fourier rapide.

IFFT : (Invers Fast Fourier Tranform) Transformé de Fourier rapide.

BPF : Fréquence corps roulants.

BPFI : (Ball Pass Frequency Inner) Fréquence bague intérieure.

BPFO : (Ball Pass Frequency Outter) Fréquence bague extérieure.

Lt: BPFI/BPFO.

ALU : arithmetic logic unit.

CLB : Configurable Logique Bloc.

E/S : entré/sortie.

FPGA : Field Programmable Gate Array.

Introduction générale

Les roulements sont des éléments les plus largement utilisés dans les machines tournantes, tels que les machines électriques, les véhicules électriques et dans le milieu industriel.

Le diagnostic des défauts des roulements a fait l'objet de plusieurs travaux de recherche, La plupart des recherches liées au diagnostic des défauts des roulements utilisent les signaux vibratoires au fait qu'ils contiennent des informations précieuses sur les défauts [1].

Notre travail est basé sur l'analyse spectrale des signaux vibratoires pour détecter les défauts des roulements. Cette technique nécessite généralement des théories de traitement de signal, tel que la transformé de Fourier et la transformé d'Hilbert ; pour l'implémentation de ces techniques, des circuits spécifiques sont utilisés, tels que les microcontrôleurs ou les circuits programmables comme les FPGA (Field Programmable Gate Array). L'FPGA est considéré comme le dispositif le plus efficace pour les applications en raison de sa performance, sa faible consommation d'énergie, et sa re-configurabilité.

Le but de notre travail est l'implémentation d'une technique de détection de défaut des roulements sur circuit FPGA, pour cela on a organisé notre mémoire comme suivant :

- Le premier chapitre est consacré à la présentation des généralités sur les roulements, et leurs différents types de défauts ; nous donnons aussi les fréquences caractéristiques pour chaque défaut d'un roulement.
- Le deuxième chapitre contient une description générale de la carte FPGA et l'outil de développement System Generateur Xilinix.
- Le troisième chapitre présente l'implémentation de notre solution, le schéma Simulink avec des blocs System Generateur Xilinix et la définition de chaque bloc .ensuite nous présentons les résultats de la simulation et nous présentons l'implémentation sur la carte FPGA.

Ce mémoire a pour objectif, d'appliquer le traitement du signal dans le domaine de maintenance industrielle, en effectuant un diagnostic pour la détection et la caractérisation des défauts des roulements sur circuit FPGA.

Chapitre I :
Les roulements et les signaux caractéristiques des défauts

1. Introduction :

En mécanique, un roulement est un dispositif destine à guider assemblage en rotation, c'est-à-dire a permettre à une pièce à tourner par a rapport à une autre selon un axe de rotation défini. Les roulements sont utilisés dans les machines rotatives et considérés comme un composant important, Toutes les machines en fonctionnement produisent des vibrations, images des efforts dynamiques engendrés par les pièces en mouvement. Ainsi, une machine neuve en excellent état de fonctionnement produit très peu de vibrations [2].

La majorité des défauts sont des creux ou des fissures apparaissant sur les composants du roulement et sont causés par l'usure ou une mal utilisation. Si ces défauts ne sont pas détectés au début, à travers le temps cela peut conduire à un échec du roulement et de la machines. Cela peut causer un grave problème de sécurité et pertes économiques. Dans les industries, la surveillance des vibrations est généralement faite pour vérifier la santé des roulements en termes de détection des défauts avant d'atteindre à l'étape critique. [2]

Ce chapitre est organisé comme suivant :

- Constitution d'un roulement.
- Généralité sur les types de roulements.
- Les defaults de roulements.
- Les différents types des signaux et sont traitements.
- A la fin les defaults de roulements (défaut sur la bague intérieur et extérieur).

2. Constitution d'un roulement

La majorité des roulements sont composés d'une bague intérieure et d'une bague extérieure, éléments roulants (billes ou rouleaux), et d'une cage. Les éléments roulants situés entre les deux bagues du roulement sont excentrique les uns des autres par la cage qui les guide et facilite leur rotation [1] (voir Figure I - 1).

C'est après le descriptif de chaque composant du roulement :

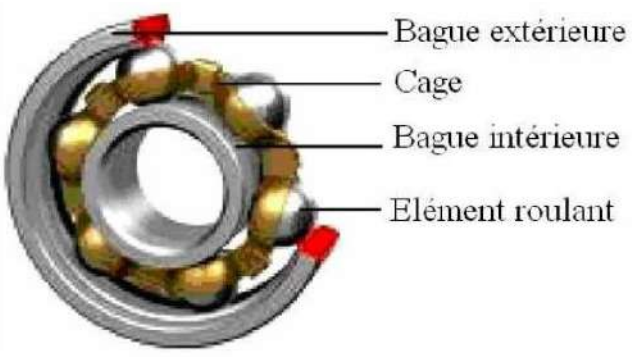


Figure I - 1 : composantes d'un roulement [2].

2.1 La bague extérieure

C'est l'élément qui limite les dimensions extérieure elle sera montée dans un logement fixe (bague extérieure fixe) ou dans un moyeu tournent (Bague extérieure tournante).

2.2 La bague intérieure

Elle est montée sur un arbre fixe ou mobile. C'est fixe si la bague extérieure est mobile, et vice versa.

2.3 La cage

La cage ne supporte pas directement les efforts appliqués au roulement. Elle permet de garder la distance entre les éléments roulants. Les types de cage varient en fonction de leur fabrication (en tôle emboutie, usinée, massive acier ou laiton, polyamide injecté).

2.4 Eléments roulants

Les éléments roulants se divisent en deux catégories : les billes et les rouleaux.
Les rouleaux sont classés en fonction de leur forme, rouleaux cylindriques, rouleaux coniques, rouleaux sphériques ainsi que les aiguilles. Le contact entre les éléments roulants et les bagues pour les roulements à billes est ponctuel alors que le contact pour les roulements à rouleaux est linéaire. Les roulements sont conçus de telle manière que les éléments roulants effectuent simultanément une rotation autour de leur propre axe et autour de l'axe des pistes [2].

3 Les différents types de roulement

Il existe plusieurs type de roulement et sont classées selon le type de l'élément roulant :

3.1 Les roulements à billes

Les roulements à billes sont classés en différentes catégories suivant la forme de leur bague (roulements à billes à gorge profonde, à contact oblique). Les roulements à billes sont les plus utilisés dans le monde industriel parce qu'ils présentent le meilleur rapport performance/prix [3].

Ces roulements peuvent être à une ou deux rangées de billes, à contact radial ou oblique. Les roulements à contact oblique permettent de supporter, en plus des efforts radiaux, des charges axiales plus importantes. Exemple de roulement à contact oblique [3].

3.2 Roulement à une rangée de billes à contact oblique

Les roulements à contact oblique peuvent supporter une charge axiale, mais ne peuvent être montés seuls à cause de l'angle de contact. Ils doivent toujours être montés par paires ou combinés. Les roulements à billes à contact oblique à deux rangées (pistes doubles) ont un angle de contact est de 25°. Les roulements à billes à quatre points de contact peuvent supporter des

charges axiales dans les deux directions. ls doivent être montés par paire et en opposition (voir Figure I - 2).

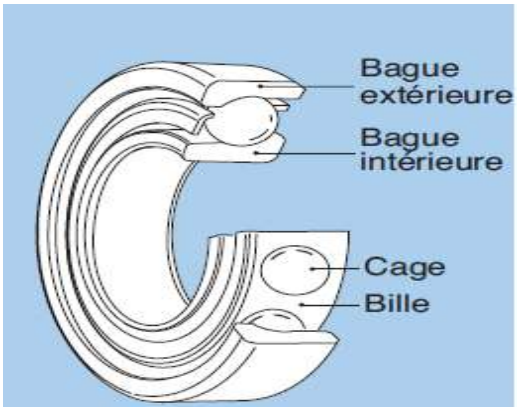


Figure I - 2 : Roulement à une rangée de billes à contact oblique [3].

3.3 Roulement à billes à contact radial

Très économiques ce sont les plus utilisés en petites et moyennes .sous charge ils se comportent comme un roulement à contact oblique dont l'angle d'inclinaison sera variable

3.4 Les roulements à rouleaux :

Les roulements à rouleaux sont classés suivant la forme des rouleaux (Roulements à rouleaux cylindriques, coniques)

- **Roulement à rouleaux cylindriques**

Les éléments roulants sont des rouleaux, ce qui donne une capacité de charge élevée. Les rouleaux sont guidés par les épaulements de la bague intérieure et/ou extérieure. Celles-ci peuvent être séparables pour faciliter le montage, et elles peuvent être montées en ajustement serre sur l'arbre et dans le logement. S'il n'y a pas d'épaulements, la bague concernée peut se déplacer librement en direction axiale. Les roulements à rouleaux cylindriques sont idéaux pour être utilisés comme des « paliers libres » qui encaissent la dilatation de l'arbre. Lorsqu'il y a des épaulements, le roulement peut supporter une légère charge axiale entre les faces des rouleaux et les épaulements. La forme des faces de rouleaux et des épaulements a été adaptée pour augmenter la capacité de charge axiale. La conception du roulement de type de roulement rouleaux coniques augmente la capacité de charge radiale [4] (voir figure I - 2).

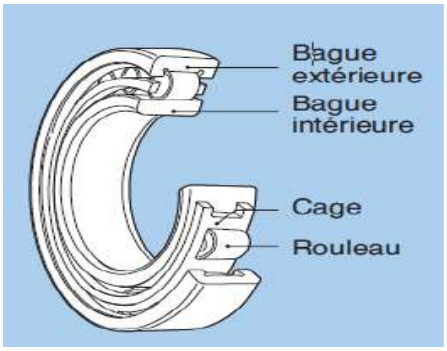


Figure I - 3 : Roulement à rouleaux cylindriques [3].

- **Roulement à rouleaux coniques**

Les roulements à rouleaux coniques sont conçus de manière à ce que le chemin de roulement des bagues intérieures et extérieures et les génératrices des rouleaux coniques se coupent en un point qui se trouve sur l'axe du roulement. Les enroulements reçoivent des charges combinées provenant de l'intérieure et roulent en se servant de cet appui comme guidage. Une force induite se produit axialement lorsqu'une charge radiale est appliquée. Les roulements doivent donc être montés par paire. La bague intérieure à rouleaux et la bague extérieure est séparables, ce qui facilite le montage avec un jeu ou pré-charge. Le réglage du jeu interne (ou pré-charge) après montage est délicat et demande une attention particulière. Les roulements à rouleaux coniques sont capables de supporter de fortes charges axiales et radiales [2] (voir figure I-4).

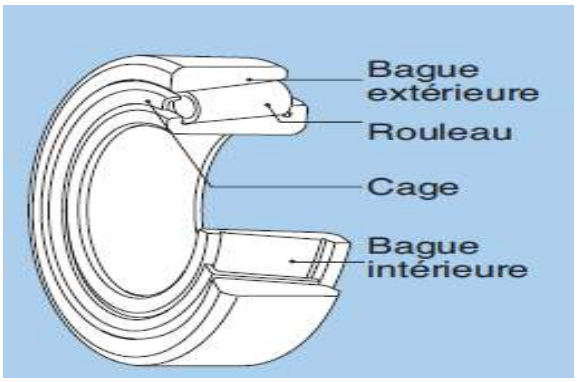


Figure I -4 : Roulement à rouleaux coniques [3].

3.5 Roulements radiaux et axiaux

La plupart des roulements peuvent supporter à la fois des charges radiales et des charges axiales. Les roulements ayant un angle de contact inférieur à 45° supportent mieux les charges radiales que les charges axiales. Ils sont appelés roulements radiaux. Les roulements ayant un angle de contact supérieur à 45° supportent mieux les charges axiales que les charges radiales.

Ils sont appelés roulements axiaux ou butées. Il est possible de faire des combinaisons de roulements radiaux et axiaux qui possèdent les caractéristiques propres des roulements radiaux et des roulements axiaux [3].

3.6 Les roulements à aiguilles

Les éléments roulants des roulements à aiguilles sont des aiguilles dont le diamètre maximal est de 5mm, et qui sont 3 à 10 fois plus longues que leur diamètre. La section des roulements est mince, mais la capacité de charge élevée malgré leur taille. Etant donné le grand nombre d'aiguilles, la rigidité de ces roulements est importante, et ils sont idéaux pour des mouvements d'oscillations et de pivotement .Il existe de nombreux types de roulements (voire figure I – 5**)**.

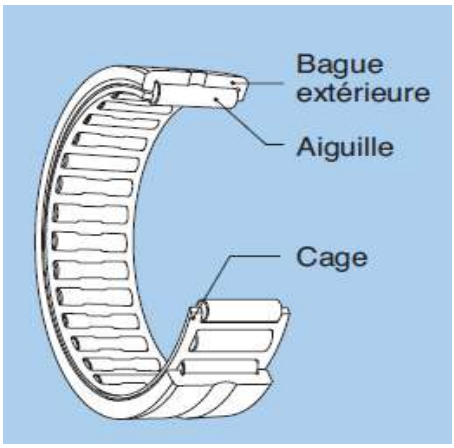


Figure I – 5 : roulements à aiguilles [3].

3.7 Butées

Il existe de nombreux types de butées qui diffèrent en fonction de la forme des éléments roulants et des applications. La vitesse de rotation admise est généralement faible et il faut être attentif à la lubrification. En plus des modèles donnés ci-dessous, il y a de nombreux types de butées pour des applications spéciales. Pour plus de détail, veuillez consulter le catalogue concerné (voire figure I – 6).

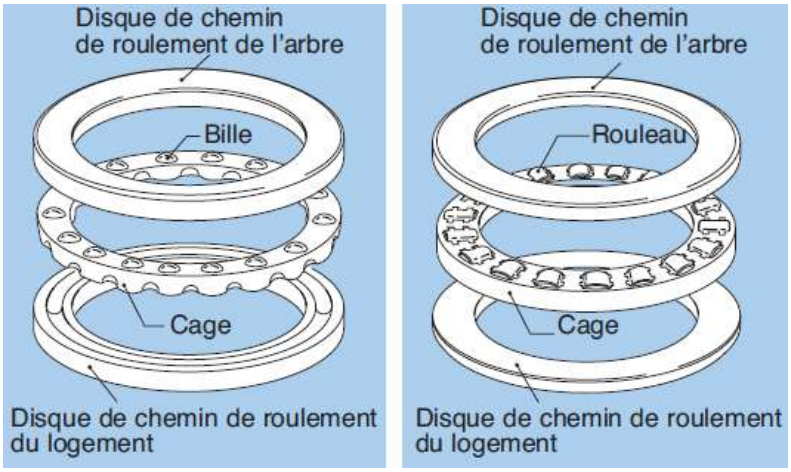


Figure I - 6 : Butées [3].

- **Butée à billes**

Peu utilisées, elles supportent des charges axiales uniquement dans un seul sens. Les vitesses de rotation sont limitées à couse de la force centrifuge sur les billes

3.8 Roulements standard et spéciaux

Grâce à leur normalisation et leur interchangeabilité internationale, les roulements peuvent être employés de façon très rationnelle. C'est pourquoi il est conseillé d'utiliser des roulements standards avec un équipement mécanique. Par ailleurs, il est possible ou parfois nécessaire d'utiliser des roulements non normalisés et de fabrication spéciale permettant de garantir des fonctions spécifiques au sein d'une machine [3].

Dans notre étude en travaille sur Les roulements à billes, qui est le plus utilise dans le domaine industriel.

4 .Origine des défauts de roulement

Il existe plusieurs défauts de roulement, les plus connus sont indiqué ces dessous :

- Mauvaise_lubrification : Le mauvais choix d'un lubrifiant où des quantités inadaptées de lubrifiant causent environ **36%** des avaries de roulements. Une mauvaise lubrification réduite d'une manière significative la durée de vie d'un roulement et comme, bien souvent, un roulement est un composant mécanique difficile d'accès, cela engendre des arrêts de maintenance longs et donc coûteux.
- Pollution_: Un roulement est un composant mécanique de précision. Son bon fonctionnement risque d'être gravement compromis par la contamination des chemins de roulement et/ou du lubrifiant par des corps étrangers. On estime ainsi que 14% des défaillances de roulements sont dues à ce phénomène.

- Mauvais_montage_: Environ 16% des avaries de roulements sont dues à un mauvais montage ou à l'utilisation d'outils inadaptés [2].
- Fatigue: Les machines ne fonctionnent pas toujours dans des conditions optimales, loin s'en faut. Surcharges, entretiens insuffisants ou inadaptés sont responsables d'environ **34%** des défaillances de roulements. Un roulement qui fonctionne dans des conditions de surcharge. [2]

5 .Quelques défauts de roulement

Les roulements sont parmi les composants les plus critiques des machines et représentent une source de panne fréquente. On peut noter deux types des défauts caractéristiques de la détérioration des roulements : les défauts dits naturels dus à la fatigue des roulements et des défauts dues à un mauvais montage ou une mauvaise utilisation du roulement [2].

En associant les différentes techniques de contrôle, l'expérience du personnel de la maintenance ainsi que l'historique des machines.

5.1 Le grippage

Le grippage est un mode d'endommagement instantané. Les matériaux en contact se soudent sous pression et température généralement dans des conditions de vitesses moyennes et élevées. Il résulte d'un manque de lubrifiant ; les contacts métal-métal entraînent des échauffements qui facilitent les microsoudures et le transfert de métal [2].

Donc, on peut dire que le grippage est un phénomène caractérisé par des micros arrachements des particules métalliques (voir figure I–7).

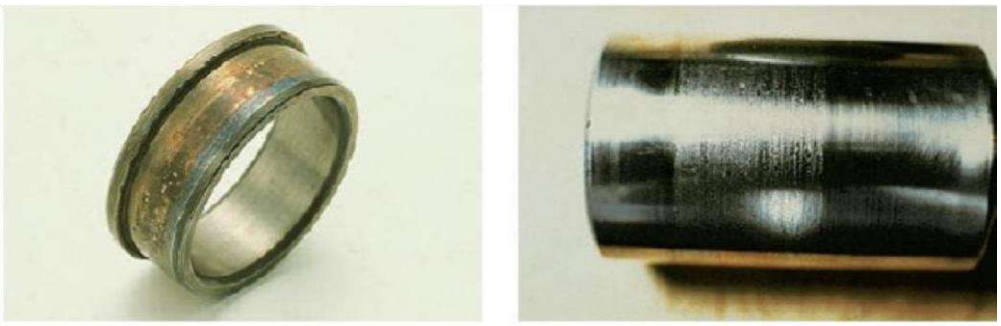


Figure I – 7 Grippage d'une bague [1].

5.2 La corrosion

- La corrosion chimique se produit par la présence de l'humidité aux points de contact des éléments roulants avec les chemins. Surtout quand les roulements viennent d'être nettoyés et sont contaminés par la transpiration des mains.

- La corrosion de contact (fretting) apporte une pâte brune formée par la rouille et le lubrifiant. Une corrosion profonde entraîne la rupture des bagues. due au mauvais choix d'ajustements entre les bagues et les arbres.



Figure I – 8 Corrosion sur une piste de roulement [2].

5.3 Usure

- Usure par déformation : C'est une déformation plastique des chemins des roulements, causée par des chocs importants ou une vibration à haute fréquence. C'est une usure sans perte de matière (présence des empreintes)
- Usure par abrasion : L'usure par abrasion donne aux roulements une couche brillante, due au glissement des éléments roulants sur les pistes, et ce par le manque ou une mauvaise lubrification.

5.4 La fissuration des bagues

C'est l'apparu des fissures et des petites cassures sur les éléments du roulement .Les coups lors du montage, le non-respect des portées de roulement (ajustement trop serré). Sont responsables sur ce type de détérioration, elle provoquées par une déformation plastique.

5.5 L'écaillage

Le cas le plus rencontré est un défaut d'écaillage dû à la fatigue des roulements. Ce défaut survient sous l'effet de la fatigue due aux contraintes de cisaillement alternées qui sévissent en sous-couche. Le fonctionnement élastohydrodynamique (type de lubrification dépendant l'élasticité du matériau) crée, par dépassement de la limite d'élasticité, des microdéformations permanentes et des fissures qui se propagent vers la surface et conduisent à l'écaillage. L'écaillage est un processus continu qui s'accélère plus ou moins après apparition des premières fissures. On peut distinguer deux types d'écaillage :

- Ecaillage de fatigue : C'est un écaillage unilatéral, il est caractérisé par des petites fissures et arrachement de fragments de matière. Il se produit lors de l'application d'une surcharge, ou lors d'un défaut désalignement.

- Ecaillage superficiel : Causé par une concentration de contraintes, due à une contrainte accentuée (résultent probablement d'un défaut d'alignement). Ou bien par la modification de la zone de pré charge. Aussi par la présence de particules étrangère à l'intérieur du roulement

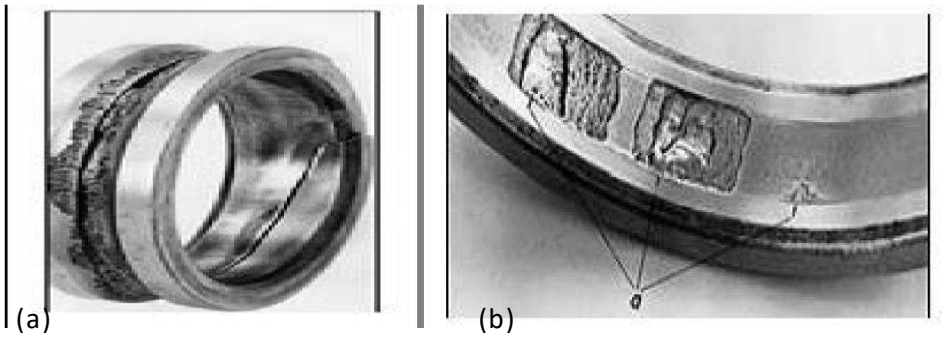


Figure I – 9 (a) écaillage de fatigue et (b) écaillage superficiel de la bague interne [2].

6 .Fréquences caractéristiques pour les défauts d'un roulement

Les défauts : écaillage, grippage, corrosion (qui entraine l'écaillage) ; les avaries les plus fréquentes sur les roulements sont les défauts d'écaillage. Lors de la mise en rotation, un train d'impulsion est généré par ces défauts, à une fréquence bien déniée que l'on appelle "fréquence caractéristique" de défaut du roulement. Ce signal périodique est l'origine de nombreuses méthodes de détection de défaut de roulement. Les fréquences caractéristiques sont déterminées à partir de la géométrie du roulement et de la cinématique de la machine étudiée

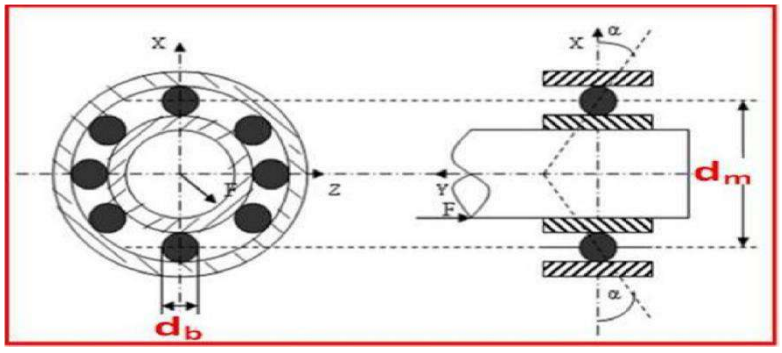


Figure I_10 Caractéristiques d'un roulement.

n : le nombre d'éléments roulants (billes, rouleaux ou aiguilles).

- dm : le diamètre primitif.
- db: le diamètre des éléments roulants.

- α : angle de contact.

1. Un défaut sur la bague extérieure générera un signal contenant des chocs espacés de la période de passage des billes sur la bague extérieure, la fréquence de ces chocs est appelée BPFO (Ball Passage Frequency Outer race).

$$\boldsymbol{BPFO} = f_{outer}(Hz) = \frac{n}{2} Fr\left(1 - \frac{d_b}{d_m}\cos\alpha\right) \qquad \textbf{(I.11)}$$

Fr: est la vitesse de rotation.

2. La fréquence de passage d'un élément roulant sur un défaut de bague intérieure, cette fréquence est BPFI (Ball Passage Frequency Inner race).

$$\boldsymbol{BPFI} = f_{inner}(Hz) = \frac{n}{2} f_r\left(1 + \frac{d_b}{d_m}\cos\alpha\right) \qquad \textbf{(I.12)}$$

3. Pour un défaut sur une bille, la vibration sera générée à deux fois la fréquence de rotation de la bille BSF (Ball Spin Frequency), puisque la bille sur un tour cognera sur la piste externe et interne :

$$\boldsymbol{BSF} = f_{ball}(Hz) = f_r \frac{d_m}{d_b}\left[1 - \left(\frac{d_b}{d_m}\cos\alpha\right)^2\right] \qquad \textbf{(I.13)}$$

Le calcul de ces fréquences caractéristiques nécessite une connaissance précise des caractéristiques dimensionnelles du roulement.

Notre travail consiste a détecté ces defaults en analysons les signaux bruts et faire correspondre le signal finale à la fréquence caractéristique du roulement**.**

7 Définition d'un signal

Un signal est la représentation physique de l'information qu'il transporte de sa source à son destinataire. Il sert de vecteur à une information. Il constitue la manifestation physique d'une grandeur mesurable (courant, tension, force, température, pression, etc.). Les signaux sont des grandeurs électriques variant en fonction du temps x(t) obtenues à l'aide de capteurs

7.1 Classification des signaux

Les signaux peuvent être classés selon leurs propriétés :

- signaux continus ou discrets.
- Signaux déterministes ou aléatoires.
- Signaux périodiques ou apériodique.

Comme illustré par la figure I.11 le signal peut être :

- signal à amplitude et temps continus (signal analogique)
- signal à amplitude discrète et temps continu (signal quantifié)

- signal à amplitude continue et temps discret (signal échantillonné)
- signal à amplitude discrète et temps discret (signal logique ou numérique)

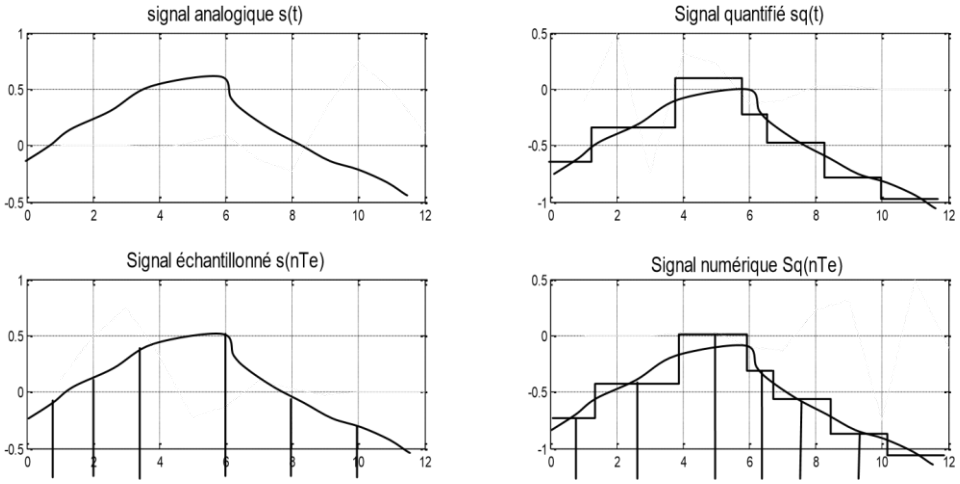
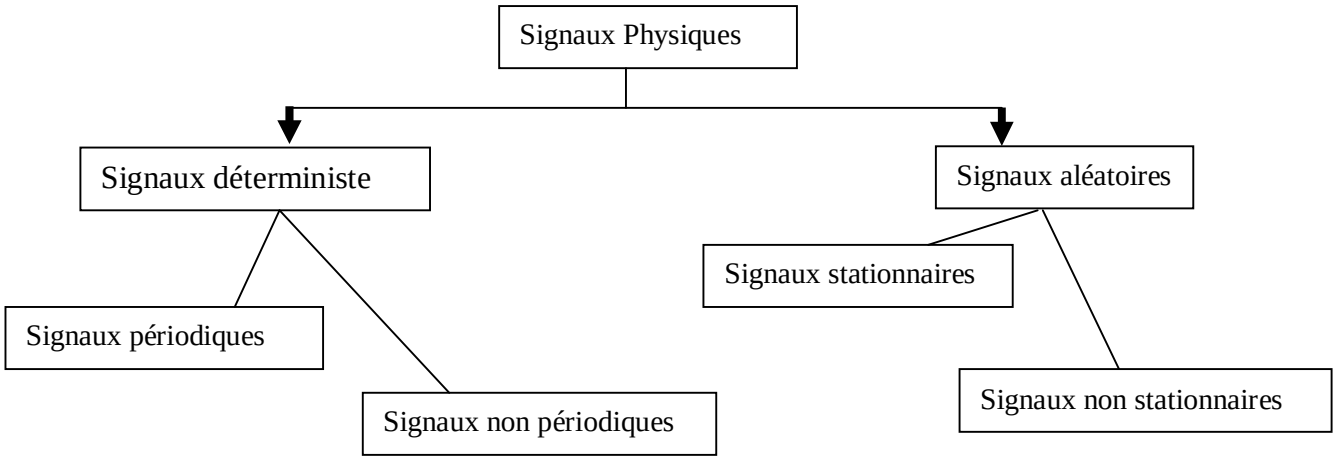


Figure I.11 Classification des signaux.

7.2 Types de signaux



a) Signaux déterministes

Un signal est appelé déterministe s'il est possible de déterminer son amplitude pour tout instant « t ». Un tel signal peut être défini par une expression mathématique ou par un graphique. Les signaux déterministes sont déterminés par un ensemble de phénomènes physiques connus ou non, Ils peuvent être périodiques et non périodiques

- **Périodiques** : un signal est dit périodique si les variations de son amplitude se reproduisent régulièrement au bout d'une période T constante.

On a donc pour tout

$$(t + T) = (t) \quad t \in R \qquad \textbf{(I.1)}$$

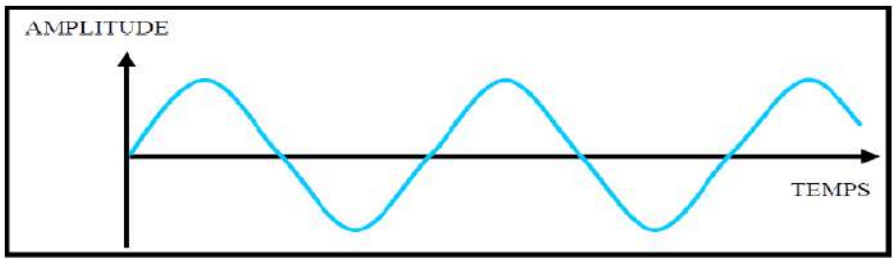


Figure I_12 : signal périodique.

La fréquence d'un signal périodique est le nombre de périodes par seconde. Elle s'exprime en hertz (Hz). La fréquence en hertz est donc égale à l'inverse de la période exprimée en secondes

$$f = {}^{1}/_{T} \qquad \textbf{(I.2)}$$

- **Non périodiques** : un signal est dit non périodique s'il n'est pas périodique, mais, existe à tout instant « t ».

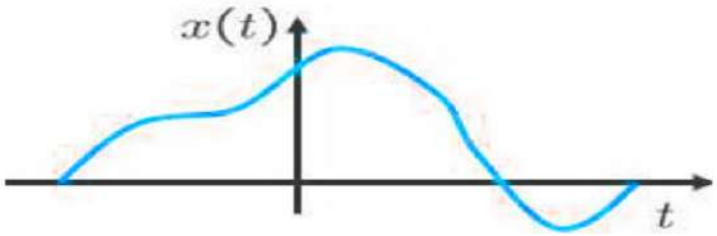


Figure I_13 : signal Non périodique

b) Signaux aléatoires :

Les signaux aléatoires sont caractérisés par le fait qu'il n'est pas possible de prédire leur amplitude à un instant donné.

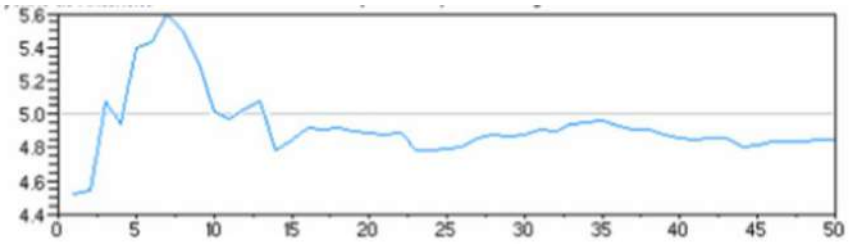


Figure I.14 : signal aléatoire.

Ils sont classés en deux types qui sont :

- **Stationnaires :** ce sont des signaux dont les caractéristiques statistiques sont

Indépendantes de l'origine du temps.

$$P\,X(t\,1), X(t\,2),..., X(t\,k) = p\,X(t\,1-t\,k), X(t\,2-t\,k),..., X(0) \qquad \textbf{(I.3)}$$

- **Non stationnaires :** ce sont des signaux qui ont les caractéristiques contraires à celles des Stationnaires

8 Analyse temporelle

L'analyse temporelle peut se faire soit à l'échelle du signal, c'est à dire en utilisant des indicateurs obtenus à partir d'une fenêtre glissante sur le signal, cela permet la détection et la localisation du défaut, soit à l'échelle d'un ensemble de signaux, les indicateurs ont une valeur scalaire calculée directement sur la totalité d'un signal. Cette analyse s'applique à des machines simples et consiste à effectuer des mesures de vitesse dans des gammes de fréquences faibles et des mesures d'accélération dans des gammes de fréquences élevée.

Cet outil est aisé à employer lorsque le signal est simple (vibration de type sinusoïdal du balourd d'un rotor par exemple), mais devient inexploitable dans le cas de sollicitations multiples et de faible influence sur les défauts [3].

9 Analyse fréquentiel

Les méthodes fréquentielles sont les méthodes les plus largement utilisées dans le monde industriel, de par leur facilité de mise en œuvre et d'interprétation**.** La connaissais des fréquences caractéristiques permet d'identifier et de localiser les défauts issus des composants mécaniques en analysant leur spectre. Elles sont basées sur la transformée de Fourier **[**2**]**.

- **Analyse d'enveloppe**

Elle est basée sur La transformée d'Hilbert qui est un outil très utilisé pour le diagnostic des machines tournantes.

10. Transforme de Fourier, Transforme de Hilbert et le Signal analytique :

Au 19ème siècle 1822, le mathématicien français J. Fourier a montré qu'une fonction périodique pouvait être décomposée en une somme infinie de fonctions exponentielles périodiques complexes [1].

La transformée de Fourier d'un signal réel est un signal complexe symétrique. Cela implique que le contenu aux fréquences négatives est redondant par rapport aux fréquences positives. Cette transformée visait à créer un signal analytique en supprimant le contenu fréquentiel négatif depuis le résultat de la transformation de Fourier. Le signal analytique a une valeur complexe, mais son spectre sera unilatéral (uniquement des fréquences positives), préservant le contenu spectral du signal à valeur réelle d'origine. Utilisation d'un signal analytique au lieu du signal à valeur réelle d'origine, s'est avérée utile dans de nombreuses applications de traitement du signal. Par exemple, en analyse spectrale, l'utilisation d'un signal analytique au lieu du signal original à valeurs réelles atténue les biais d'estimation et élimine les artefacts entre termes liés aux composantes de fréquence positive et négative.

Soit x (t) un signal d'énergie finie non limité dans la bande de valeurs réelles, pour lequel nous souhaitons construire un signal analytique correspondant z (t). La transformée de Fourier en temps continu de x (t) est donnée par équation (**I.4**).

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}\,dt \qquad \textbf{(I.4)}$$

Disons que le spectre de magnitude de X (f) est tel qu'illustré à (Figure I_14). Nous notons que le signal x (t) est une valeur réelle et son spectre de magnitude | X (f) | est symétrique et s'étend infiniment dans le domaine fréquentiel.

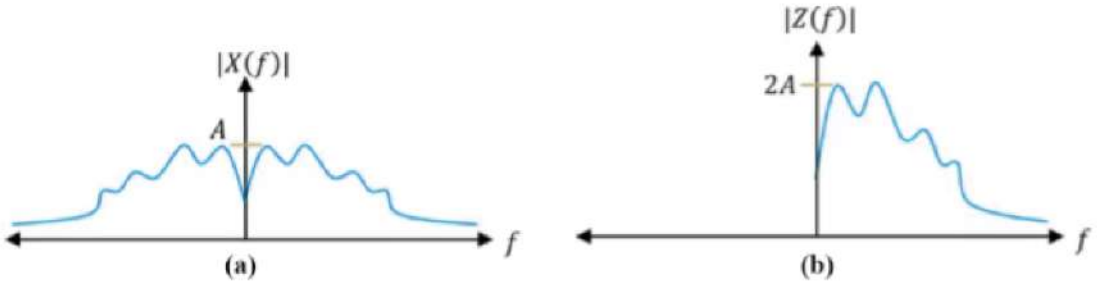


Figure I_15 : (a) Spectre du signal continu x(t) et (b) Spectre du signal analytique z(t). Un signal analytique peut être formé en supprimant le contenu de fréquence négative de la transformé de Fourier du signal à valeur réelle.

$$\mathbf{Z(f)} = \begin{cases} X(0) \;\; pour \;\; f = 0 \\ 2X(f) \; pour \; f > 0 \\ 0 \qquad pour \;\; f < 0 \end{cases} \qquad \textbf{(I.5)}$$

Autrement dit, dans le domaine fréquentiel, le contenu spectral Z(f) du signal analytique z(t) est donné par le spectre correspondant du signal analytique résultant Z(f) est représenté sur (voir Figure I_14 (b)) puisque le spectre du signal analytique est unilatéral, le signal analytique aura une valeur complexe dans le domaine temporel, ce qui permet de représenter le signal analytique en termes de composantes réelles et imaginaires, telles que

$$z(t) = z_r(t) + jz_i(t) \qquad \textbf{(I.6)}$$

Comme le contenu spectral est préservé dans un signal analytique, il s'avère que la partie réelle du signal analytique dans le domaine temporel est essentiellement le signal à valeur réelle initiale.

$$z_r(t) = x(t) \qquad \textbf{(I.7)}$$

Alors, que se passe-t-il de la partie imaginaire ? Qui est le compagnon de x (t) qui occupe la partie imaginaire dans le signal analytique résultant ? Résumant sous forme d'équation

$$z(t) = z_r(t) + jz_i(t) = \begin{cases} z_r(t) = x(t) \\ jz_i(t) = ?? \end{cases} \qquad \textbf{(I.8)}$$

Il est intéressant de noter que la transformation de Hilbert [2] peut être utilisée pour trouver une fonction complémentaire (partie imaginaire dans l'équation ci-dessus) à un signal à valeur réelle telle que le signal réel puisse être étendu analytiquement de l'axe réel à la moitié supérieure du plan complexe. En désignant transformée Hilbert comme HT {}, le signal analytique est donné par

$$z(t) = z_r(t) + jz_i(t) = x(t) + jHT\{x(t)\} \qquad \textbf{(I.9)}$$

À partir de ces discussions, nous pouvons voir qu'un signal analytique z (t) pour un signal à valeur réelle x (t) peut être construit en utilisant deux approches

- Approche par domaine de fréquence : le spectre unilatéral de z (t) est formé à partir du spectre bilatéral du signal à valeur réelle x (t) en appliquant l'équation(I.5).
- Approche du domaine temporel : utilisation de l'approche de transformation de Hilbert donnée dans l'équation(I.9).

Une des propriétés importantes d'un signal analytique est que ses composants réels et imaginaires sont orthogonaux.

$$\int_{-\infty}^{+\infty} z_i(t) z_r(t) dt = 0 \qquad \textbf{(I.10)}$$

11. Conclusion :

Dans ce chapitre, nous avons présenté une étude théorique des organes mécaniques des machines tournantes et leurs défauts. Ce dernier permet de donner une idée sur les impacts des défauts présentés par les signaux à analyser.

Dans notre cas, on a basé sur les défauts du roulement sur la bague intérieure et extérieure et leurs déférentes fréquences vibratoires.

Chapitre II :

Les FPGAs et l’outil de développement

1. Introduction :

Durant ces dernières années l'évolution de l'industrie a connu une évolution gigantesque grâce à la concurrence entre les fabricants.

Actuellement, la logique programmable est la plus utilisés par les microcontrôleurs (µc) et microprocesseurs(µp) mais l'inconvénient de ces circuit et c'est qu'on peut seulement les programmer selon le programme existant dans une mémoire, l'architecture interne est celle proposé par le fabriquant, tout comme les entrées/sorties, et aussi sans oublie le nombre de ces circuit nécessaire qui peut être important, ce qui avait pour conséquence un prix important, une mise en œuvre complexe et un circuit imprimé de taille. [9]

Pour diminuer le cout de fabrication, de développement et de maintenance Les PLD ont subi une évolution technologique au fil du temps depuis l'apparition de premier PAL (programmable array logique réseau logique programmable) jusque à l'aboutissement des premiers circuits intégrés reconfigurables de type FPGA (Field Programmable Gate Array ou réseau de cellules logiques programmables) par la société XILINX en 1985.

Ces circuits FPGAs sont capables de réaliser plusieurs fonctions logiques complexes avec la souplesse et la flexibilité apportée par la logique programmable est ça dans un seul circuit, et ils sont les circuits logiques programmables les plus performants qui existent en ce moment.

2. Les circuits programmables FPGA

2.1 Définition

Un FPGA est un appareil contenant une matrice de circuits logiques reconfigurables en réseau de portes. Lorsqu'un FPGA est configuré, le circuit interne est connecté de manière à créer une implémentation matérielle de l'application software. Contrairement aux processeurs, les FPGA utilisent un matériel dédié pour la logique de traitement et ne disposent pas de système d'exploitation. Les FPGA sont véritablement parallèles par nature, de sorte que différents traitements ne doivent pas concurrencer pour les mêmes ressources. En conséquence, les performances d'une partie de l'application ne sont pas affectées par l'ajout de traitements. En outre, plusieurs boucles de contrôle peuvent être exécutées sur un même périphérique FPGA à des vitesses différentes. Les systèmes de contrôle basés sur FPGA peuvent appliquer une logique de verrouillage critique et peuvent être conçus pour empêcher un opérateur de forcer les E / S. Cependant, contrairement aux conceptions de carte de circuit imprimé (PCB) câblées qui ont des ressources matérielles fixes, les systèmes basés sur FPGA peuvent littéralement recâbler leurs circuits internes pour permettre la reconfiguration après le déploiement du système de contrôle sur le terrain. Les dispositifs FPGA offrent les performances et la fiabilité

d'un circuit hardware dédié.

Un seul FPGA peut remplacer des milliers de composants discrets en incorporant des millions de portes logiques dans une seule puce de circuit intégré (IC). Les ressources internes d'une puce FPGA sont constituées d'une matrice de blocs logiques configurables (CLB) entourés d'une périphérie de blocs d'E / S illustrée à la Figure II.1 Les signaux sont acheminés au sein de la matrice FPGA par des commutateurs d'interconnexion programmables et des routes filaires [5][6].

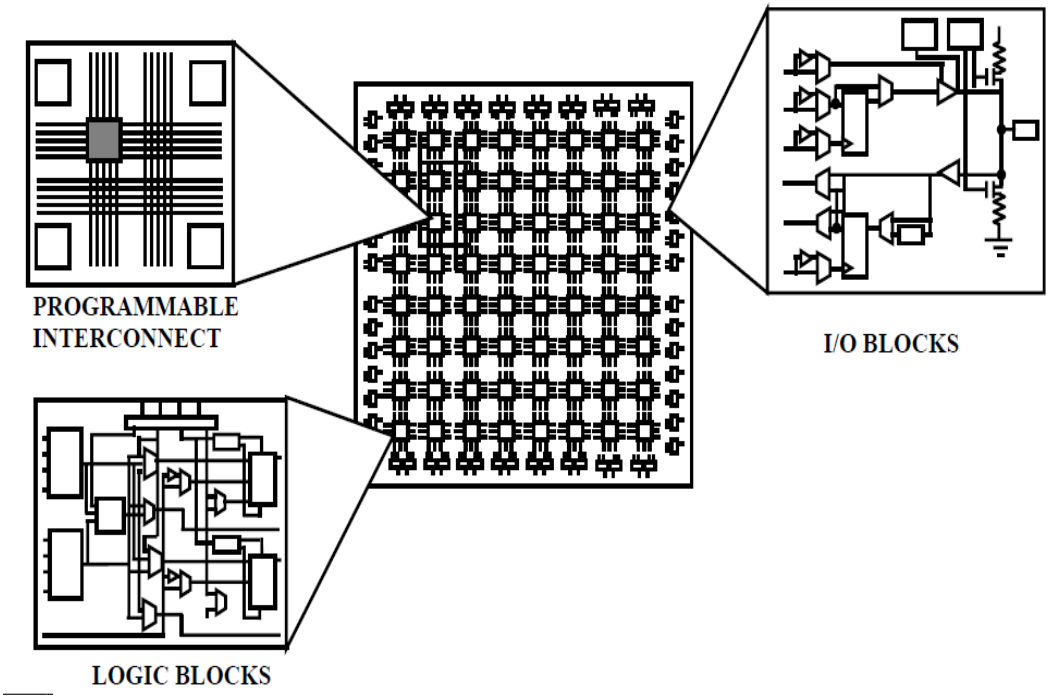


Figure II.1 : Structure interne du FPGA [5] [6].

Dans un FPGA, les blocs logiques sont implémentés à l'aide des portes d'entrée multiples à faible niveau de fan-in, ce qui lui confère une conception plus compacte par rapport à une.

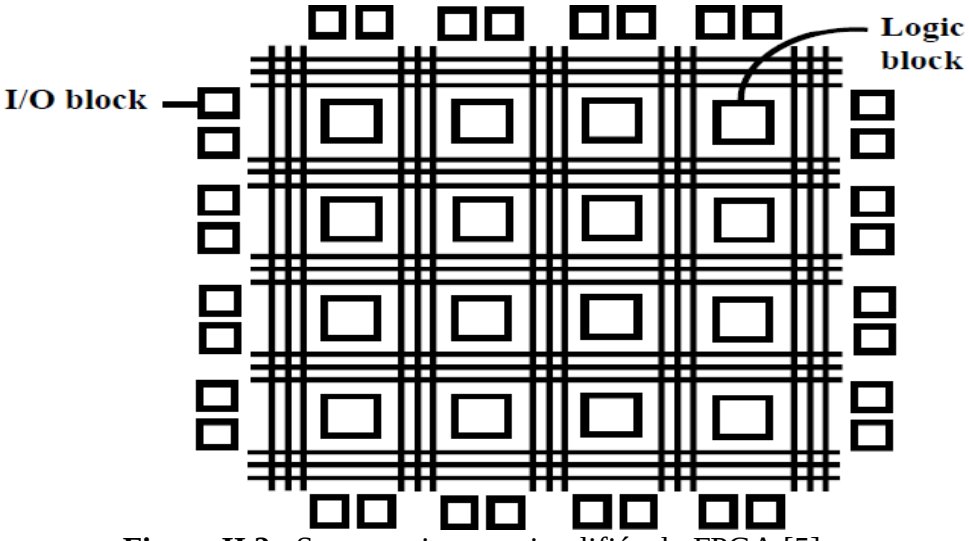


Figure II.2 : Structure interne simplifiée du FPGA [5].

Implémentation avec une logique AND-OR à deux niveaux. FPGA fournit à son utilisateur un moyen de configurer : L'intersection entre les blocs logiques AND et La fonction de chaque bloc logique. [8]

Le bloc logique d'un FPGA peut être configuré de manière à fournir une fonctionnalité aussi simple que celle d'un transistor ou aussi complexe que celle d'un microprocesseur. Il peut être utilisé pour mettre en œuvre différentes combinaisons de fonctions logiques combinatoire et séquentielle. [9]

3 Structure interne

La figure II.2 illustre la structure de base et les composants essentiels d'un FPGA générique. La logique programmable consiste en un ensemble de blocs à grain fin utilisés pour implémenter la logique de l'application. Ceci est parfois appelé la structure du FPGA. Les blocs logiques sont généralement basés sur une architecture de (LUT), leur permettant de mettre en œuvre toute fonction arbitraire des entrées. Les blocs logiques sont généralement disposés en mosaïque dans une structure en grille et interconnectés via une matrice de routage programmable qui permet aux blocs d'être connectés dans des configurations arbitraires. Les blocs d'entrée et de sortie (E / S) interface entre les composants internes ou le cœur du FPGA et les périphériques externes. Le routage signifie que pratiquement n'importe quel signal peut être acheminé vers n'importe quelle broche d'E / S du périphérique [5] [6].

Outre ces fonctionnalités de base, la plupart des FPGA fournissent une forme de synchronisation d'horloge permettant de contrôler la synchronisation d'un signal d'horloge par rapport à une source externe.

Un réseau de distribution d'horloge fournit des signaux d'horloge à toutes les parties du

FPGA tout en limitant le décalage d'horloge entre les différentes sections d'une conception. Il existe également une logique dédiée pour charger la configuration dans le FPGA. Cette logique ne forme pas directement une partie dans la conception de l'utilisateur, mais c'est la charge supplémentaire nécessaire pour que les FPGA soient programmables et compatibles fournit les mécanismes permettant de charger la conception de l'utilisateur sur un FPGA. Chacune de ces composantes est examinée un peu plus en détail dans les sections suivantes.

3.1 Logique des FPGA

La plus petite unité logique du FPGA est la cellule logique. C'est l'unité logique à grains la plus fine produisant un seul bit en sortie. Il se compose généralement d'une petite LUT (lookup table), table de conversion et d'un verrou sur la sortie, comme indiqué dans Figure II.3. Le loquet est utilisé pour enregistrer la sortie. Sur certains FPGA, les deux sorties peuvent être fournies, comme dans Figure II.3 ; sur d'autres, un multiplexeur peut être utilisé pour choisir si le signal enregistré ou non enregistré est sorti. Les registres peuvent être utilisés pour construire des machines à états finis, des compteurs et pour enregistrer des données dans les systèmes en pipeline.

La LUT peut implémenter toute fonction arbitraire de ses entrées. Des fonctions plus complexes peuvent être mis en œuvre en cascadant plusieurs niveaux de LUT en connectant la sortie, X, à l'entrée de la couche suivante. Les fabricants de FPGA doivent faire un compromis en choisissant la taille des LUT. Envisager de mettre en œuvre une simple porte AND à deux entrées. Avec une grande LUT, une fraction importante de la logique serait inutilisée de plus petites TAL permettraient une utilisation plus importante des ressources. Cependant, lors de l'implémentation complexe fonctions avec de petites LUT, plusieurs cellules logiques devraient être mises en cascade, ce qui peut considérablement augmenter le délai de propagation. Pour des fonctions complexes, il serait préférable d'utiliser une table de correspondance plus grande. Pour équilibrer cela, la surface de silicium de la table croît de manière exponentielle avec le nombre d'entrées est plus efficace pour scinder une fonction complexe en plusieurs LUT [5] [6].

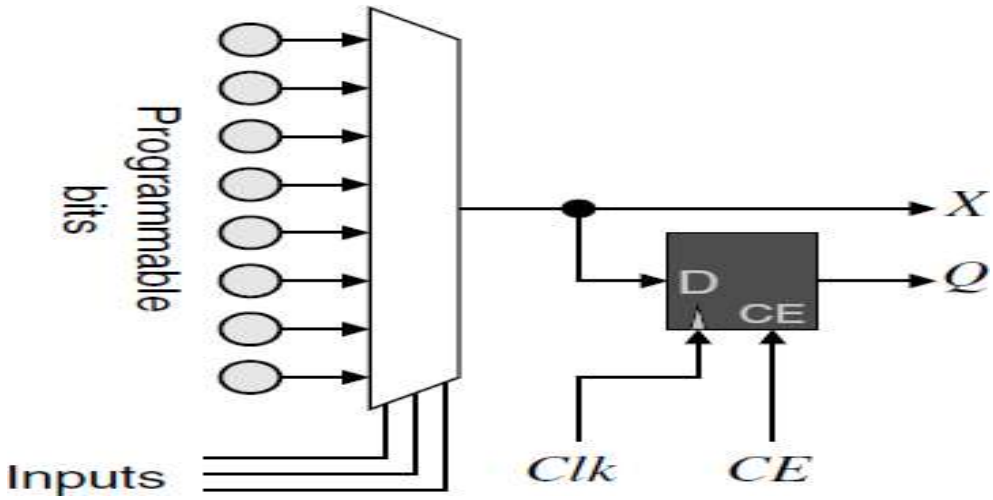


Figure II.3 : Une cellule logique est la pierre de fondation d'une FPGA [5] [6].

Les FPGA homogènes à grain fin consistent en une mer de blocs logiques identiques. Cependant, la plupart des FPGA actuels se sont éloignés de cette structure homogène, avec un mélange hétérogène de à la fois des blocs logiques à grain fin et des blocs plus gros à grain grossier destinés à l'accélération couramment utilisés opérations. Cela présente à la fois des avantages et des inconvénients en termes de conception. Les systèmes hétérogènes seront généralement plus rapides, car les blocs à grains grossiers n'auront pas le même niveau de frais généraux que ceux Cependant, les dispositifs hétérogènes sont également plus difficiles à programmer, et si le mélange des blocs spécialisés n'est pas correct, une grande partie des fonctionnalités peut rester sous-utilisée [5] [6].

3.2 Interconnexion

L'interconnexion programmable a pour but de permettre aux différents blocs logiques d'être utilisés de manière flexible connecté pour mettre en œuvre la fonctionnalité souhaitée. Il est impossible de fournir un dédié connexion directe de toutes les sorties possibles à toutes les entrées possibles. Dans une application donnée, seul une très petite fraction de ces connexions serait nécessaire ; cependant, aucune des connexions potentiellement peut être requise. La solution consiste à disposer d'un plus petit ensemble de ressources de connexion (lignes de routage) pouvant être utilisées. Partagés de manière flexible pour créer les connexions requises pour une application donnée.

Puisque l'interconnexion prend place sur le silicium, il y a un compromis. Si seulement quelques lignes de routage sont fournies, plus d'espace est disponible pour la logique, mais des circuits complexes ne peuvent pas être construits en raison des ressources d'interconnexion

insuffisantes. Alternativement, si trop de lignes de routage sont fournies, beaucoup ne seront pas utilisés, ce qui réduit la zone disponible pour la logique. Il y a un optimum quelque part entre ces extrêmes, où des lignes de routage suffisantes sont disponibles pour la plupart des applications, sans utiliser trop d'espace.

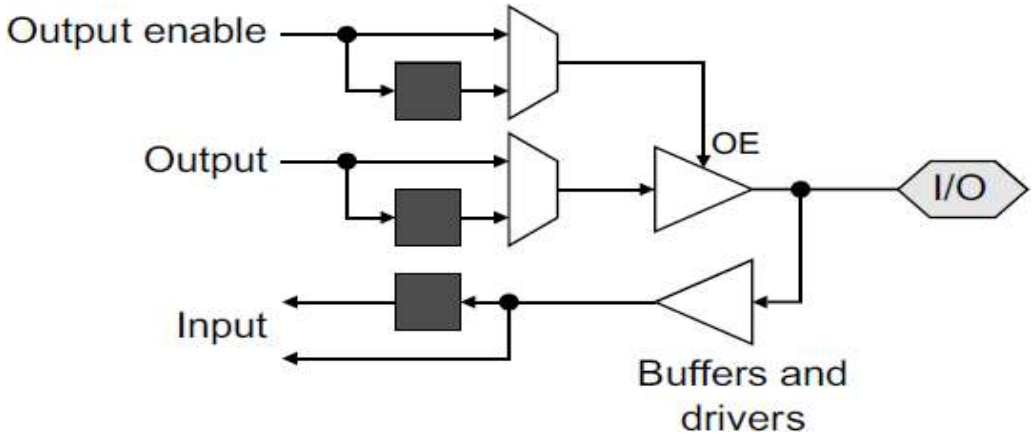
Le réseau de connexions pour la plupart des FPGAs sont basés sur une structure de grille, semblable à celle montré par la Figure II.2. À chaque intersection de grille, une forme de commutateur de traverse permet une connexion programmable entre les lignes de routage horizontale et verticale.

Un autre facteur qui doit être considéré est le retard de propagation par le réseau d'interconnexion. Chaque commutateur que le signal passe à travers ajoutera au retard. Cela a conduit à une structure segmentée, où pas toutes les lignes de routage est commuté à chaque jonction. En outre, des connexions directes spécifiques peuvent être fournies entre les blocs logiques adjacents pour réduire le nombre de signaux qui doivent passer par la matrice interconnecte. Tout en ayant une gamme de différentes ressources d'interconnexion améliore la flexibilité et la vitesse, il peut rendre le processus de trouver de bonnes routes nettement plus complexes, en particulier pour les dessins qui utilisent la plupart de la logique disponible sur la FPGA.

Il est important qu'une seule sortie pilote une ligne de routage. Si deux sorties sont connectées ensemble avec un haut et l'autre bas, l'alimentation sera effectivement court-circuitée, endommageant la puce. L'un des rôles des outils de programmation FPGA est de veiller à ce que cela ne se produise jamais. Périphériques utilisant le bus les structures de routage doivent limiter le courant des transistors de commande où utiliser un drain ouvert et un pull-up passif. Les deux approches affectent la vitesse et la dissipation de puissance [5] [6].

3.3. Entrée et sortie

Les blocs d'entrée et de sortie permettent la connexion de signaux entre le FPGA et des périphériques externes. La flexibilité des FPGA exige qu'ils soient capables de se connecter à un large éventail d'autres composants. La plupart des broches d'entrée / sortie peuvent être programmées comme entrées, sorties ou les deux et être connectées directement à la mémoire interne logique du FPGA, ou passer par les registres. Cette gamme de fonctionnalités peut être fournie par un périphérique d'E / S générique. Tel que celui illustré à la figure II.4, chaque multiplexeur étant contrôlé par un programmable bit de configuration. La principale complexité des E / S provient du grand nombre de signaux standards différents utilisées par différents appareils et en assurant une synchronisation appropriée entre le FPGA et les composants hors



puce.

Figure II.4 : Bloc d'entrée / sortie générique [5].

Lors de l'interfaçage des signaux hauts vitesse, il est important de prendre en compte les effets sur la ligne de transmission. En règle générale, toute connexion plus longue qu'un huitième environ du temps de montée doit avoir la terminaison appropriée pour minimiser les réflexions du signal. Les FPGA étant devenus plus grands et plus rapides, il a devenu pratique courante de fournir une terminaison programmable sur puce. Si non fourni, il est nécessaire d'assurez-vous que la terminaison appropriée est construite hors puce pour des signaux de commutation rapides. De nombreux protocoles de communication de haut niveau nécessiteraient d'importantes ressources FPGA simplement pour gérer le protocole. Bien que cela puisse être mise en œuvre dans la structure du FPGA, la conception est souvent assez complexe et est mieux implémenté en utilisant une puce d'interface dédiée. Ces puces gèrent généralement l'accès aux médias (MAC) et la signalisation physique (PHY) ne nécessitant que des signaux logiques standard dans leur interface avec le FPGA. Certains FPGA intègrent des circuits MAC et PHY dédiés aux communications couramment utilisées protocoles, tels que PCI Express et Gigabit Ethernet. La plupart des FPGA offrent une gamme de connexions d'E / S générales et dédiées [5] [6].

3.4. Horloge

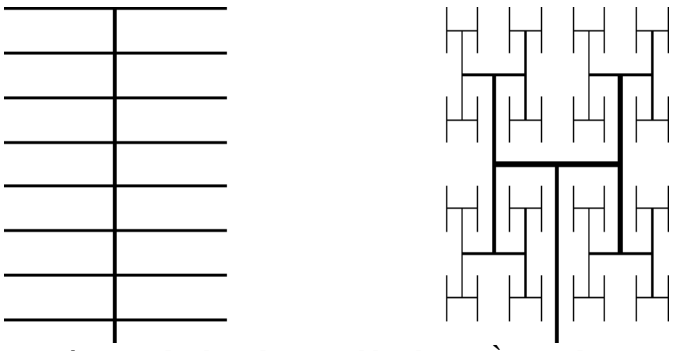


Figure II.5 : Réseaux de distribution d'horloge. À gauche : structure en H ; à droite : colonne vertébrale et structure des côtes [5] [6].

Les FPGA sont conçus principalement comme des dispositifs synchrones, ils nécessitent donc un signal d'horloge. Bien que multiples horloges puissent être utilisées dans une conception, les registres, les mémoires synchrones et les blocs d'E / S ne peuvent être contrôlés que par une horloge. Un domaine d'horloge comprend tous les circuits contrôlés par une horloge particulière. Étant donné qu'une horloge pilote potentiellement de nombreuses entrées sur l'ensemble du FPGA, il existe deux options spéciales. Caractéristiques des lignes d'horloge qui les distinguent des autres interconnexions d'un FPGA. Tout d'abord, c'est important que tous les registres d'un domaine d'horloge soient synchronisés simultanément. Le décalage d'horloge est la différence heure d'arrivée du signal d'horloge aux registres successifs d'un dessin (Friedman, 2001). Tout biais réduira la marge de synchronisation et impact sur la vitesse d'horloge maximale d'une conception. Bien que l'asymétrie puisse être minimisée en utilisant des réseaux de distribution d'horloge H-tree ou X-tree, ceux-ci ne correspondent pas bien à la structure en grille de FPGA (Lamoureux etWilton, 2006). Au lieu de cela, une structure de colonne vertébrale et de côtes est généralement utilisée (Figure II.5), où l'horloge est distribuée aux rangées à l'aide d'une épine puis aux éléments de la rangée à l'aide des nervures (Lamoureux etWilton, 2006). Une série de mémoires tampons est utilisée dans le réseau de distribution pour gérer les grande fan-out. En pratique, il est un peu plus complexe que cela, car le réseau doit prendre en charge plusieurs domaines d'horloges, et de nombreux FPGA ont également un mélange d'horloges locales et globales [5] [6].

3.5. Configuration

Tous les commutateurs, structures et options du FPGA sont contrôlés par des bits de. Plus généralement, les données de configuration sont contenues dans le FPGA dans des cellules de

mémoire statique (SRAM), bien que les appareils à programmation flash soient disponibles chez certains fabricants. Les FPGA programmés en flash sont non volatiles, conserver leur configuration après la mise hors tension, afin qu'ils puissent immédiatement commencer à travailler après mise sous tension. Les FPGA à mémoire statique perdent leur configuration à la mise hors tension, ce qui nécessite la configuration fichier à recharger chaque fois que le système est allumé. La mémoire statique avait l'avantage d'être infiniment reprogrammable ; La mémoire flash a généralement un nombre limité de cycles d'écriture.

Un FPGA est configuré en chargeant le fichier de configuration dans la mémoire interne. Ceci est généralement accompli en transmettant le fichier de configuration sur le FPGA soit en série, soit en parallèle 8, 16 ou en certains cas 32 bits à la fois. Dans un système autonome, le fichier de configuration est généralement contenu dans un fichier. Mémoire flash externe, telle que celle illustrée à la figure II.6, avec la carte configurée pour configurer automatiquement le FPGA au démarrage. Les FPGA peuvent également être programmés en série via un JTAG (Joint Test Action Group) interface, et cette interface est couramment utilisée lors de la programmation de FPGA à partir d'un système informatique hôte. La nécessité de transférer la configuration dans le FPGA chaque fois que le système est alimenté expose le fichier de configuration rendant la propriété intellectuelle contenue dans vulnérable au piratage ou au retournement ingénierie. Les FPGA haut de gamme surmontent cette limitation en chiffrant le fichier de configuration et déchiffrer la configuration une fois qu'elle est dans le FPGA. Certains FPGA prennent également en charge la relecture de la configuration à partir du FPGA. Cette fonctionnalité peut être utile pour déboguer une conception avec les outils appropriés, mais n'est généralement pas utilisée en fonctionnement normal.

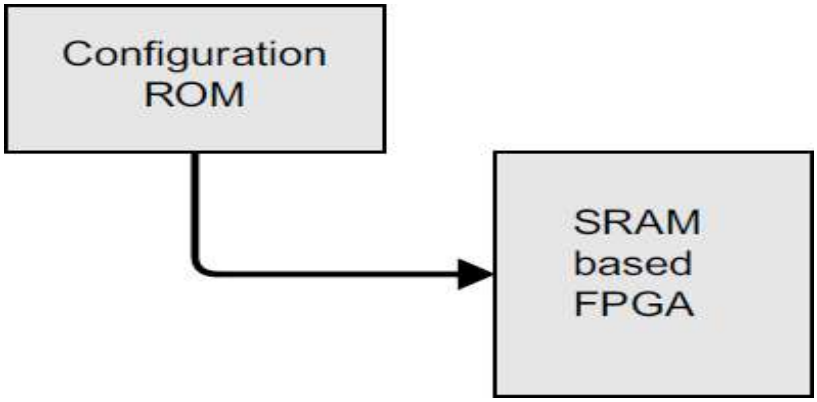


Figure II.6 : Configuration de FPGA basés sur la RAM statique [5] [6].

4. MATLAB/Simulink

MATLAB qui développé par MathWorks est le logiciel. Les utilisateurs peuvent

analyser données, peut développer des algorithmes et créer des modèles mathématiques. Il offre une large gamme de domaines d'utilisation. MATLAB est l'outil le plus pratique pour analyser les signaux numériques. Simulink a également été développé par MathWorks et intégré à MATLAB. Il est logiciel de programmation graphique qui offre une conception au niveau du système, une simulation, génération automatique de code et test et vérification continus des systèmes intégrés. Il prend également en charge la conception matérielle basée sur un modèle à l'aide du générateur de système (Xilinx). Dans ce mémoire a été utilisée pour la conception de FPGA basée sur un modèle. Dans ce projet nous utiliseront s la version Xilinx Vivado 2018.2 et MATLAB 2017a [7].

5. System Générateur de Xilinix

Le Système Générateur est un outil de conception DSP de Xilinx qui permet d'utiliser le Simulink basé sur un modèle Mathworks pour la conception FPGA. , qui utilise une description de haut niveau pour réaliser facilement un système complexe. Une expérience préalable avec les méthodologies de conception Xilinx FPGA ou RTL n'est pas requise lors de l'utilisation du système générateur.Les conceptions sont capturées dans un environnement de modélisation Simulink compatible DSP à l'aide d'un jeu de blocs spécifique à Xilinx. La conception du générateur de système peut ensuite être importée dans le projet de conception Vivado.

Quelques information's sur l'outil Xilinx System Generator, comment il peut metre utilisé pour développer divers modèles de conception DSP et comment ces modèles peuvent être implémentés dans le matériel [7].



Figure II.7 : Xilinx vivado.



Figure II.8: Xilinx System Generator.

5.1 Configuration MATLAB avec System Generator

Dans ce mémoire, les FPGA de Xilinx, ZedBoard, ont été utilisés comme indiqué précédemment. À FPGA basé sur un modèle de conception, l'environnement Simulink de MathWorks a été utilisé avec Blocs générateurs. Le logiciel utilisé pour concevoir le matériel doit être sélectionné avec versions. L'installation de versions exactes du logiciel est l'une des étapes les plus importantes d'une courte avance. Avec les mauvaises versions de Xilinx vivado et MATLAB, concevoir peut-être incompréhensible. Ils doivent également être compatibles avec le matériel à utiliser. Comme mentionné, la carte de développement ZedBoard a été utilisée comme matériel pour les logiciels Xilinx vivado 18.2 et MATLAB 2017b ont été utilisés. À configurer les logiciels Xilinx vivado et Matlab, MATLAB devrait être installé en premier, puis Xilinx doit être installé. Lors de l'installation Xilinx vivado, à partir de la fenêtre de configuration 'Select a MATLAB l'installation pour System Generator, la version réelle de MATLAB doit être sélectionnée, puis alors la configuration peut être complétée [7].

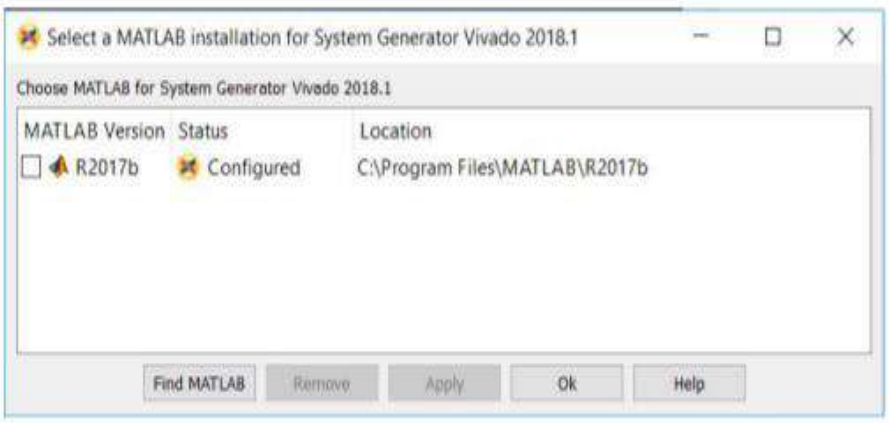


Figure II.9 : Configuration MATLAB et System Generator Vivado. [11].

Avant de commencer, vous devez vérifier que MATLAB est configuré pour Vivado

Design Suite. Faites ce qui suit : 1. Sélectionnez Démarrer> Tous les programmes> Outils de conception Xilinx> Vivado 2018.x> Générateur système> System Generator 2018.x MATLAB Configurateur. 2. Cliquez sur la case à cocher de la version de MATLAB que vous souhaitez configurer, puis cliquez sur OK. Sur les systèmes Windows, vous devrez peut-être lancer le configurateur MATLAB en tant qu'administrateur. Lorsque MATLAB configurateur est sélectionné dans le qu'administrateur

5.2. Blocs Xilinx

Dans cette partie en s'intéresse aux blocs Xilinx et plus spécialement les blocs utilise dans notre travail d'implémentation de la transforme d'Hilbert et décomposition de spectre du signal finale.

5.2.1 Organisation des bibliothèques de blocs de Xilinx

.

Bibliothèque	Description
AXI4 Bloks	Cette bibliothèque comprend tous les blocs supportés par l'interface AXI4.
Basic Elément Blocks	Cette bibliothèque comprend des blocs de construction standard pour la logique numérique.
Communication Blocks	Cette bibliothèque est couramment utilisée dans les systèmes de communications numériques. La bibliothèque de blocs de communication comprend des blocs de correction d'erreurs et des blocs modulateurs.
Control Logic Blocks	Cette bibliothèque comprend des blocs pour les circuits de commande et les machines d'état.
Data Type Blocks	Cette bibliothèque comprend des blocs qui convertissent les types de données (y compris les passerelles).
DSP Blocks	Cette bibliothèque comprend des blocs de traitement numérique du signal(DSP).
Floating-Point Blocks	Cette bibliothèque comprend des blocs qui prennent en charge le type de données à virgule flottante ainsi que d'autres types de données. Seul un seul type de données est pris en charge à la fois. Par exemple, une entrée à virgule flottante produit une sortie à virgule flottante ; une entrée à virgule fixe produit une

	sortie à virgule fixe.
Index Blocks	Cette bibliothèque comprend tous les blocs du générateur de système.
Math Blocks	Cette bibliothèque comprend des blocs qui implémentent des fonctions mathématiques.
Memory Blocks	Cette bibliothèque comprend des blocs qui implémentent et accèdent aux mémoires.
Tool Blocks	Cette bibliothèque comprend des blocs "Utilitaires".

Tab II.1 : Organisation des bibliothèques de blocs de Xilinx.

5.2.2. Blocs de générateur de système Xilinx

*** Blocs FFT v.9.1**

IL existe de nombreux cas où le traitement du signal implique la mesure des spectres de fréquence dans les signaux, soit en calculant la transformée de Fourier discrète (DFT) ou la transformation de Fourier rapide (FFT).

Le DFT est défini par la formule suivante :

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1.$$

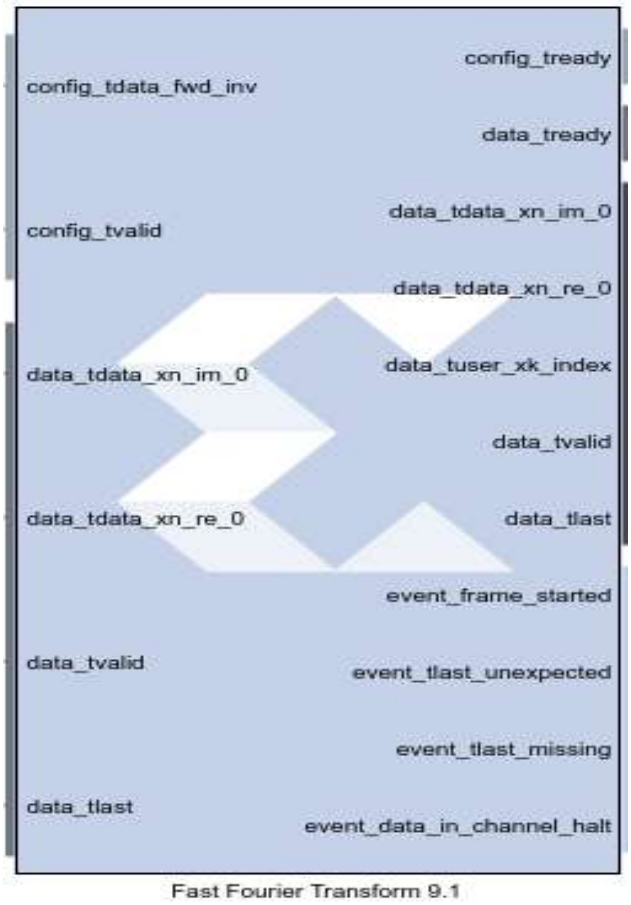


Figure II.10 : blocs FFT v.9.1 [11].

*Les caractéristiques de FFT v.9.1 sont :

-FFT complexe directe et inverse, temps d'exécution configurable.

-Tailles de transformation N = 2 ^ m, m = 3-16

-Précision de l'échantillon de données bx = 8 - 34

-Précision du facteur de phase bw = 8 - 34

-Type's arithmétiques :

° Point fixe non mis à l'échelle (pleine précision).

° virgule fixe.

° Bloc à virgule flottante.

-Interface à virgule fixe ou flottante.

-Arrondi ou troncature après le papillon.

-Bloquer la RAM ou la RAM distribuée pour le stockage des données et du facteur de phase.

-Taille de point de transformation configurable en temps d'exécution en option.

-Calendrier de mise à l'échelle configurable au moment de l'exécution pour les cœurs à virgule fixe mis à l'échelle.

-Ordre de sortie bit / chiffre inversé ou naturel

-Insertion de préfixe cyclique en option pour les systèmes de communication numériques

-Quatre architectures offrent un compromis entre la taille du cœur et le temps de transformation

***Constant :**

Le bloc Xilinx Constant génère une valeur constante. Ces valeurs peuvent être une valeur fixe, une valeur booléenne. Si l'on compare avec le bloc constant Simulink, il est similaire à ce bloc, mais peut être utilisé pour piloter directement les entrées.



Figure II.11: Xilinx System Constant. [11].

***Gateway In:**

Avec les blocs Xilinx Gateway In, les données produites par les blocs System Generator peuvent être saisies pour votre conception Simulink.



Figure II.12 Gateway In. [11].

*** Gateway Out:**

Les blocs Xilinx Gateway Out sont les sorties de la partie Xilinx de votre conception Simulink. Ce bloc convertit le type de données en virgule fixe ou en virgule flottante de System Generator en un type de données Simulink entier, simple, double ou en virgule fixe [2]



Figure II.13: Gateway out [11].

***System Generator:**

Le jeton du générateur de système sert de panneau de commande pour contrôler les paramètres du système et les paramètres de simulation. Il est également utilisé pour la génération de code. Chaque modèle Simulink contenant Xilinx Blockset doit contenir un bloc

System Generator.



Figure II.14: Xilinx System Generator [11].

5.2.3 Blocs Simulink

*** Terminator**

Ce bloc termine le port de sortie non coté.



Figure II.15: Simulink – Terminator [11].

***Real-Image to Complex:**

Ce bloc convertit des entrées réelles et/ou imaginaires en signaux complexes.



Figure II.16: Simulink - Real-Imag to Complex [11].

***Time Scope**

Le bloc Time Scope affiche les signaux dans le domaine temporel. Le bloc Time Scope accepte les signaux d'entrée ayant les caractéristiques suivantes :

-Temps d'échantillonnage continu ou discret.

-Valeur réelle ou complexe.

-Dimensions fixes ou variables.

-Type de données à virgule flottante ou fixe.

-N-dimensionnelle.

-Énumérations Simulink.



Figure II.17 Scope [11].

6 .Conclusion

Dans ce chapitre nous avons parlé l'évolution des circuits logiques programmables qui ont conduit à l'apparition des FPGA. Au départ, nous avons présenté le début de l'apparition des circuits programmables FPGA ainsi que l'outil de développement système générateur Xilinix et les blocs de cet outil qu'on a utilisé dans notre projet.

Chapitre III :

Simulation et implémentation

1. Introduction

Dans ce chapitre, nous présenterons la partie implémentation de notre travail qui permet la détection des defaults de roulement, en basant sur l'harmonique des fréquences. Pour la détection des défauts sur la bague intérieure ou extérieur, en utilisant la méthode dite Transforme de Hilbert, pour la validation de notre travail on a utilisé une base des données qui contient les signaux de défaut et en appliquant notre système de traitement (transformé de Hilbert et transformé de Fourier) sur cette base des données ; à la fin nous avons pu identifier le type de défaut.

Pour cela, Nous proposons deux solutions d'implémentions : la première solution (1) et la deuxième solution (2).

2. La base des données utilisées

Pour étudier, analyser et diagnostiquer les défauts de roulement, nous avons choisi de travailler sur les donneés des expériences faites dans le centre de données sur les roulements de l'Université Case Western Reserve [12].

Les roulements utilisés sont des roulements à billes normaux et défectueux installés sur un moteur électrique de 2 ch. Les roulements défectueux ont été réinstallés dans le moteur d'essai et les données de vibration ont été enregistrées pour des charges de moteur de 0 à 3 chevaux (vitesses de moteur de 1797 à 1720 tr / min).

Tous les fichiers de données sont au format Matlab (* .mat). Ces données numériques ont été collectées à une fréquence de 12 000 échantillons par seconde ; Chaque fichier contient les données de vibration suivantes :

- De : données de l'accéléromètre fixé sur le cote entraînement de moteur.
- Ba : données de l'accéléromètre fixé sur la plaque de base de moteur.
- Fe : données de l'accéléromètre fixé sur le côté ventilateur.
- rpm: tours par minute.

Dans ce cas-là on a travaillé sur des données échantillonnées à 12 000 hz, afin de capter le cote entraînement De.

- **Schéma globale du système**

Le schéma suivant donne les différentes étapes de notre système de manière globale comme suivant :

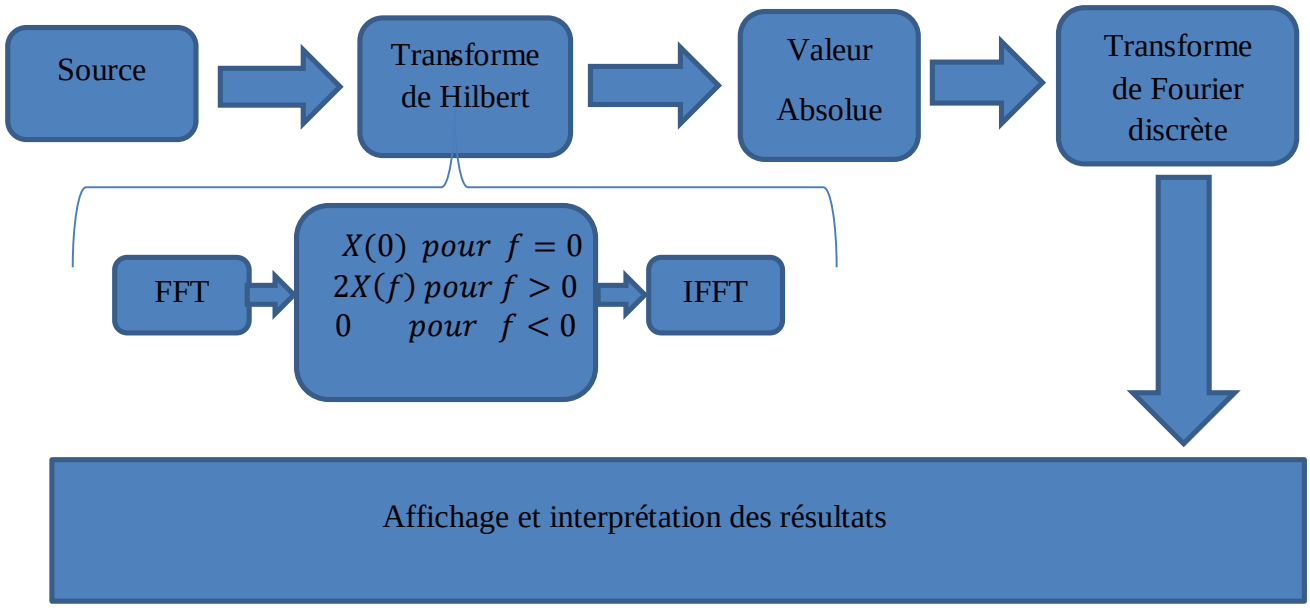


Figure III.1 : Schéma synoptique du system global.

Par la suite nous allons détailler chacune des étapes précédentes :

3. Source

Cette étape permet d'ouvrir la base des données qui contient le signale capture sous forme des données numériques en format Matlab (* .mat). On considère cette étape comme l'entrée du système.

3.1 Transforme de Hilbert

Transformé de Hilbert (HT), est un outil important dans le domaine du traitement numérique du signal (DSP), il est utilisé pour construire un signal analytique pour différents types de signaux allant de linéaire et non linéaire, stationnaire et non stationnaire. Le signal analytique obtenu est utilisé pour déterminer la nature des ondes via l'extraction de la phase et de la fréquence instantanée. [3]

HT peut être implémenté en quatre étapes comme suit :

- Obtenir la transformée de Fourier sur le signal d'entrée donné.

- Multiplier les termes de fréquence positifs par deux.
- Mettre les termes de fréquence négatifs à zéro.
- La transformée de Fourier inverse.

- **Valeur absolue**

Calculer la valeur absolue de la partie imaginaire et réelle du signal.

- **Transforme de Fourier discrète(DFT)**

L'objectif d'utiliser une transforme de Fourier dans cette étape est de décomposer l'enveloppe de spectre se forme des pics.

3.2 Affichage et interprétation des résultats

Après avoir fait la simulation de notre système nous transférons les données de sortie vers le Worckspace en appliquant un programme créé par le langage Matlab que nous avons développé pour extraire l'information et détecter le type de défaut.

4. Programmation avec système générateur Xilinix

Nous avons basé sur la simulation à interface graphique, à savoir MATLAB Simulink, cet environnement de développement est accessible, varié, rapide et compatible avec différentes applications.

MATLAB Simulink combiné avec Xilinx System Generator « XSG » représente un outil puissant de conception, test et implémentation de designs sur les systèmes embarqués. En effet, cet outil offre la possibilité d'utiliser le langage de haut niveau d'abstraction de MATLAB et les blocs graphiques de Simulink et Xilinx simultanément. [4]

5. Les architecteurs proposées

Pour faire le traitement de signal et d'extraire les informations utiles, nous avons proposé deux architectures déférentes, nous les avons appelées solution(1) et(2), les deux solutions sont des programmes créés par des blocs de Système générateur Xilinix.

5.1 Solution (1)

Dans cette dernière, nous avons utilisé trois blocs FFT, les deux premiers blocs pour faire la HT et le troisième pour décomposer l'enveloppe de spectre sous forme des pics ; Entre chaque bloc FFT il y a un sous-système (subsystem), ou il y a des blocs déférents pour faire (contage, calcule, conversion…etc.).

Le schéma bloc général de notre première solution dite solution(1), est donné comme suivant :

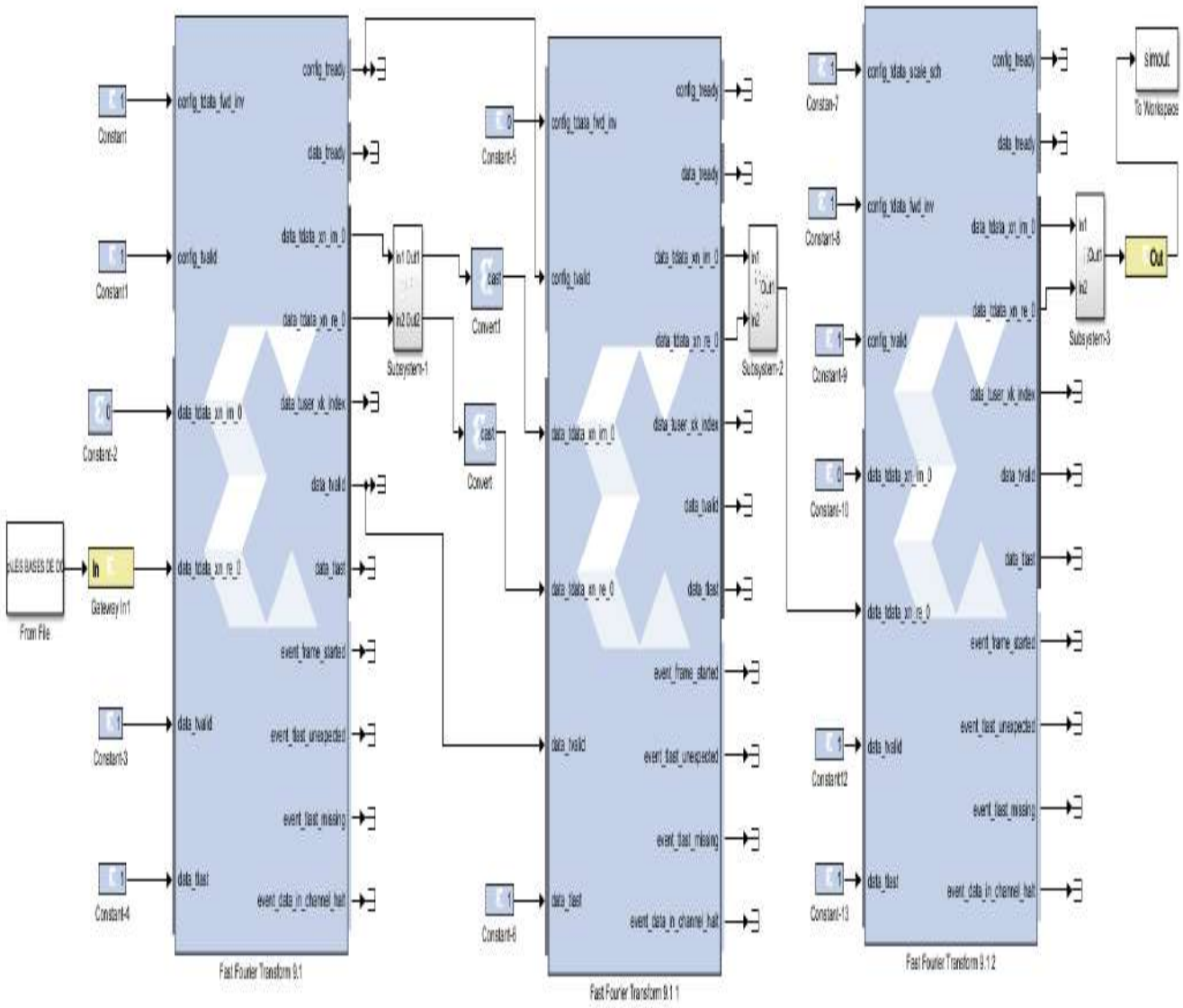


Figure III.2 : Schéma globale de la solution (1).

5.1.1 Description du Schéma de la solution 1

Notre schéma contient plusieurs blocs et sous-système, pour bien comprendre les blocs et le rôle de chacun, on a divisé les blocs comme suivant :

- **Source**

Ce bloc nous permettons d'importer les fichiers de format (.mat).

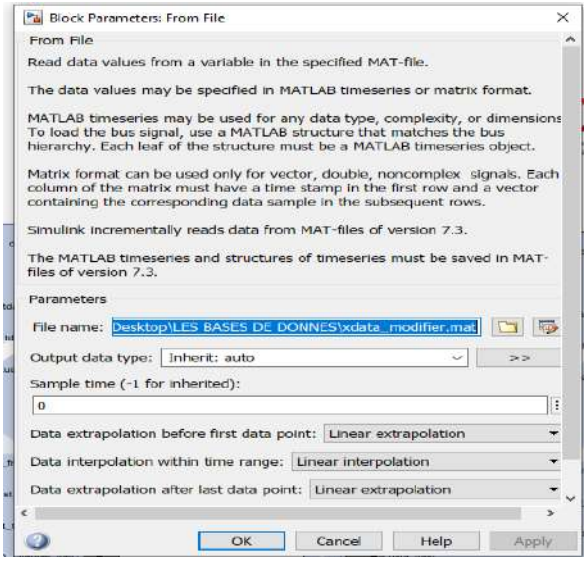


Figure III. 3 : Fenêtre de bloc source.

- **Gateway In :**

  Il est utilisé pour passer du domaine continu en virgule flottante au domaine Xilinx en temps discret à virgule fixe. Il permet de spécifier la fréquence à laquelle le signal sera échantillonné, ainsi que la précision et le type des données (data type) du signal. [4]

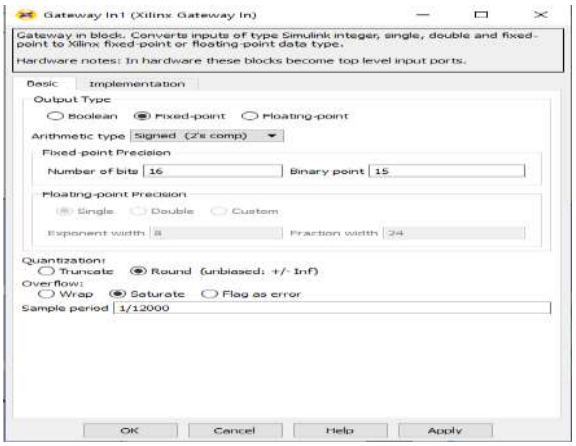


Figure III.4: Fenêtre de bloc Gateway In Axi4.

La figure III.5 présente les données du signal brut dans le bloc Gateway In. Axi4**.**

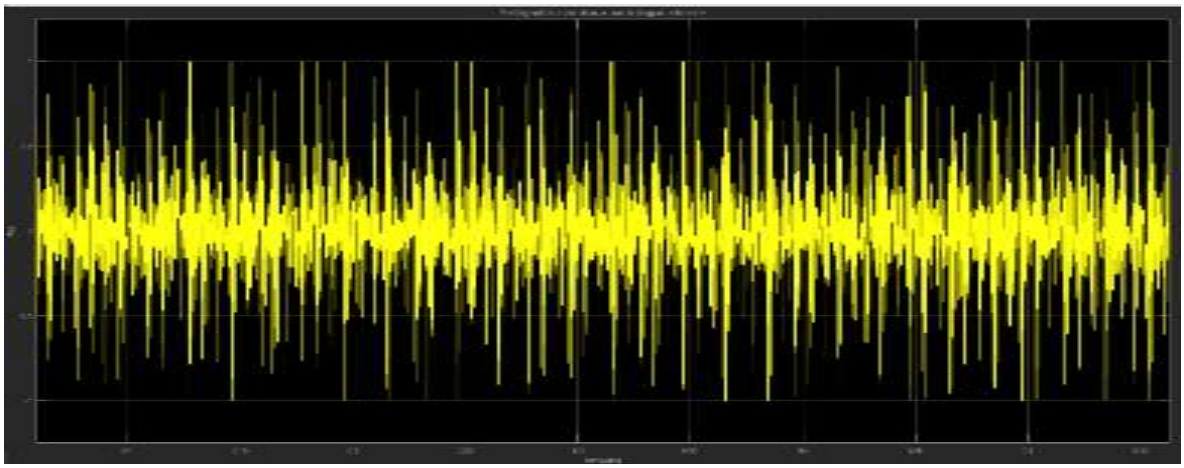
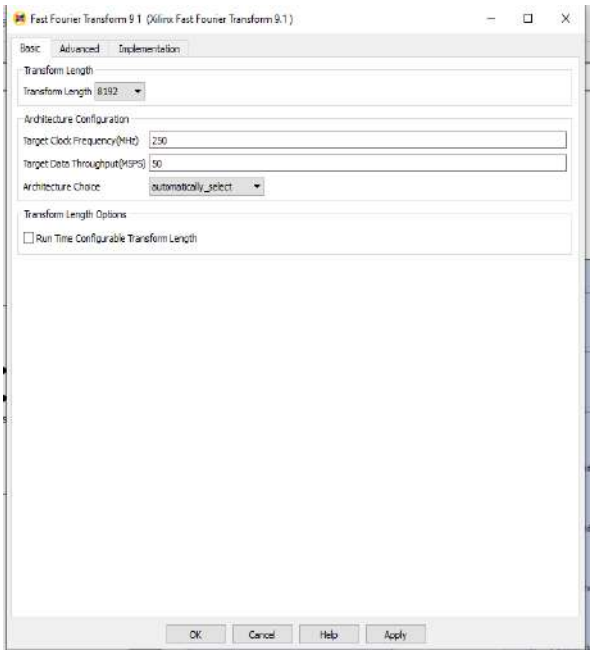


figure III.5 : signal brut.

- **Premier bloc FFT v.9.1**

Ce bloc contient les paramètres suivants, il doit être configuré avant l'utiliser par exemple (voi figure III.6**)**.

- Fonction FFT directe.
- Taille de transformation 8192.
- Type arithmétique Virgule-Fixe.



-

Figure III.6 : Fenêtre de bloc FFT v.9.1

- **Sous-système (subsystem1)**

Après le calcule de FFT du signal brut qui représente un signal de défaut et pour éliminer les termes de fréquence négative, nous avons développé avec le langage Matlab ce sous-système (voir figure III .7) qui contient un bloc (Mcod1) écrit comme suivant :

```
function [zIm,Zre ] = Mu(xIm,xRe, Cou, Lim)
 if Cou < Lim

    zIm =xIm*2;     % multiplie la partie imaginaire par deux
    Zre=xRe*2;      % multiplie la partie reel par deux
 else
    zIm=0;          %mettre le reste des valeures de la partie imaginaire a zero
    Zre=0;          %mettre le reste des valeures de la partie reel a zero
  end
```

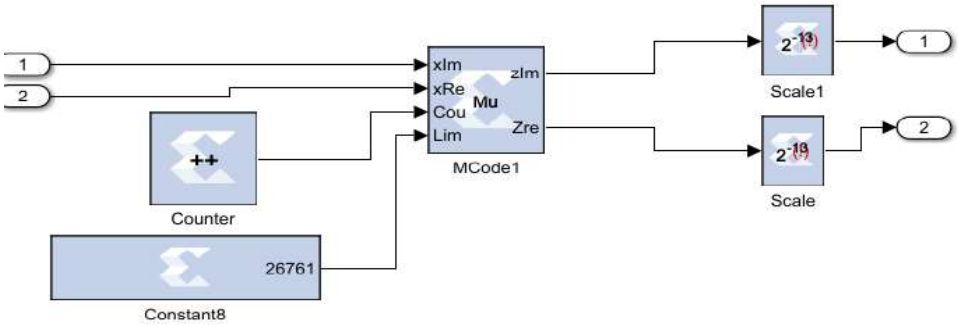


Figure III .7: sous- système (subsystem1)

La figure III.8 représente le signale FFT avant et après le traitement.

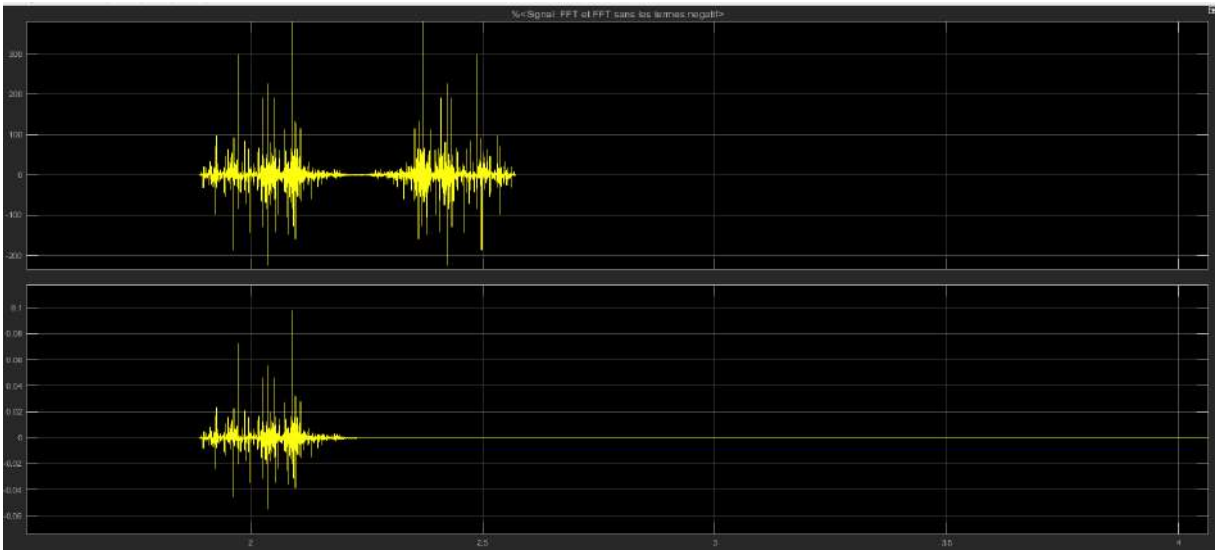


Figure III.8 : FFT du signal(haut) et FFT sans les termes negatifs (bas).

- **Deuxième bloc FFT**

Pour retourner au domaine temporel, nous avons appliqué une deuxième FFT inversée, pour cela on utilise un bloc FFT mais avec une fonction inverse. Dans cette étape nous pouvons dire que le processus de transformer de Hilbert est terminé (Voir figure III .10**)**

- **Sous-système (subsytem2)**

Ce subsysteme a pour objectif de calculer la valeur absolue du signal IFFT et le résultat affiché dans un scop.

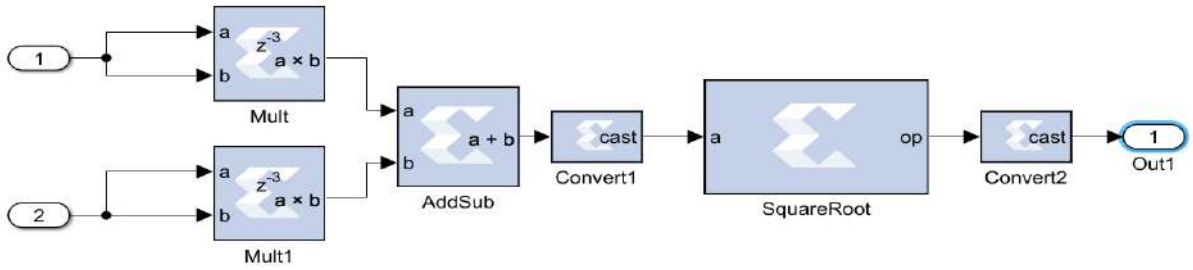


Figure III.9 : subsytem2 **.**

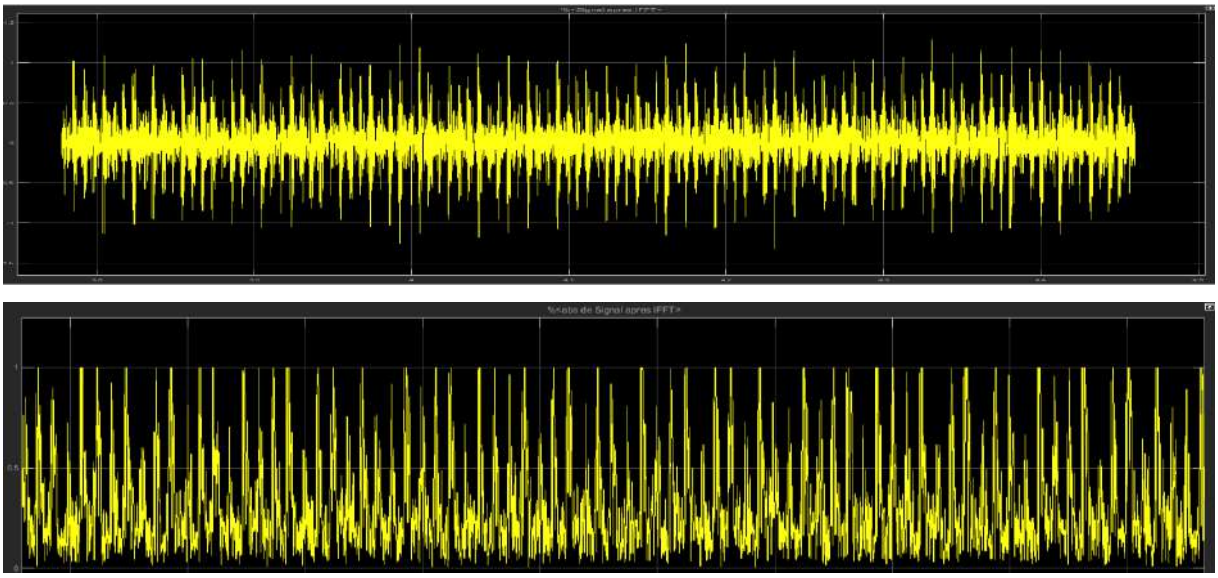


Figure III .10:IFFT du signal(haut) et leur abs(bas).

- **Le troisième bloc FFT**

Le troisième bloc FFT nous l'avons utilisé pour décomposer l'enveloppe de spectre sous forme des pics et calculer la valeur absolue des pics par un autre sous-système (subsystem3), cette étape est considérée comme la dernière étape dans notre Solution(1). Le signal obtenu vu par un scop (voir figure III.11**)** dans ce cas, il représente un spectre du signal de défaut sur la bague intérieure.

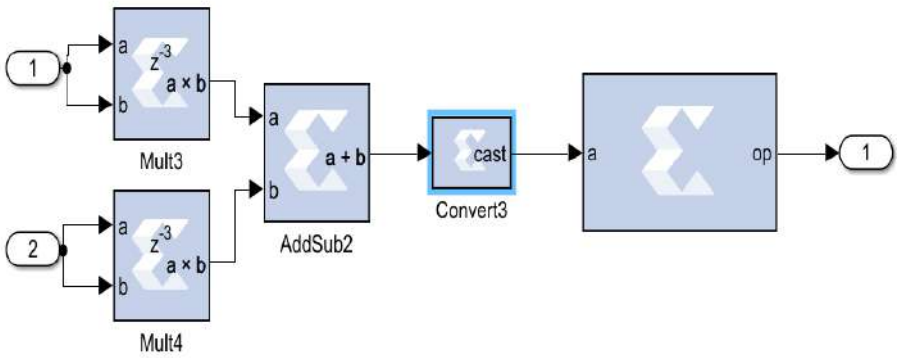


Figure III.11 : Sous-system(subsystem 3).

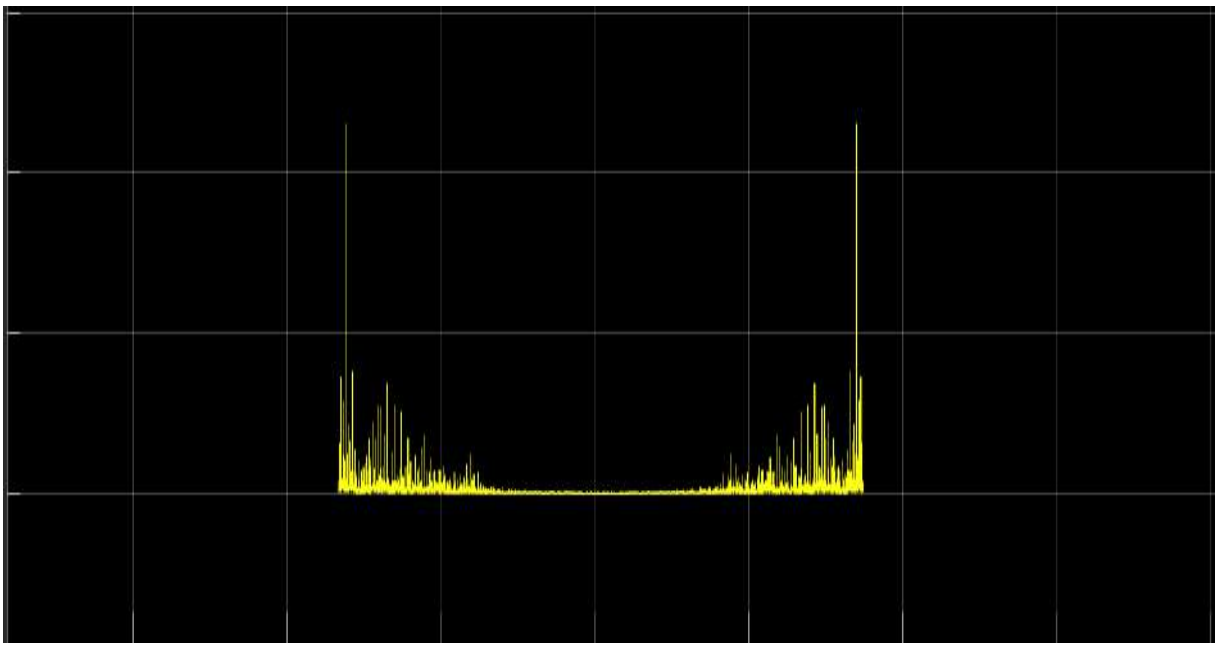


Figure III .12 : Spectre du signal de défaut sur la bague intérieure.

Pour optimiser les ressources, nous avons proposé l'architecture suivante : avec un schéma synoptique pour différencier entre la solution (1) et la solution(2). Comme il est montré dans l'étape.1, 2,3 et 4. (Voir figure III .13**)**

5.2 Solution 2

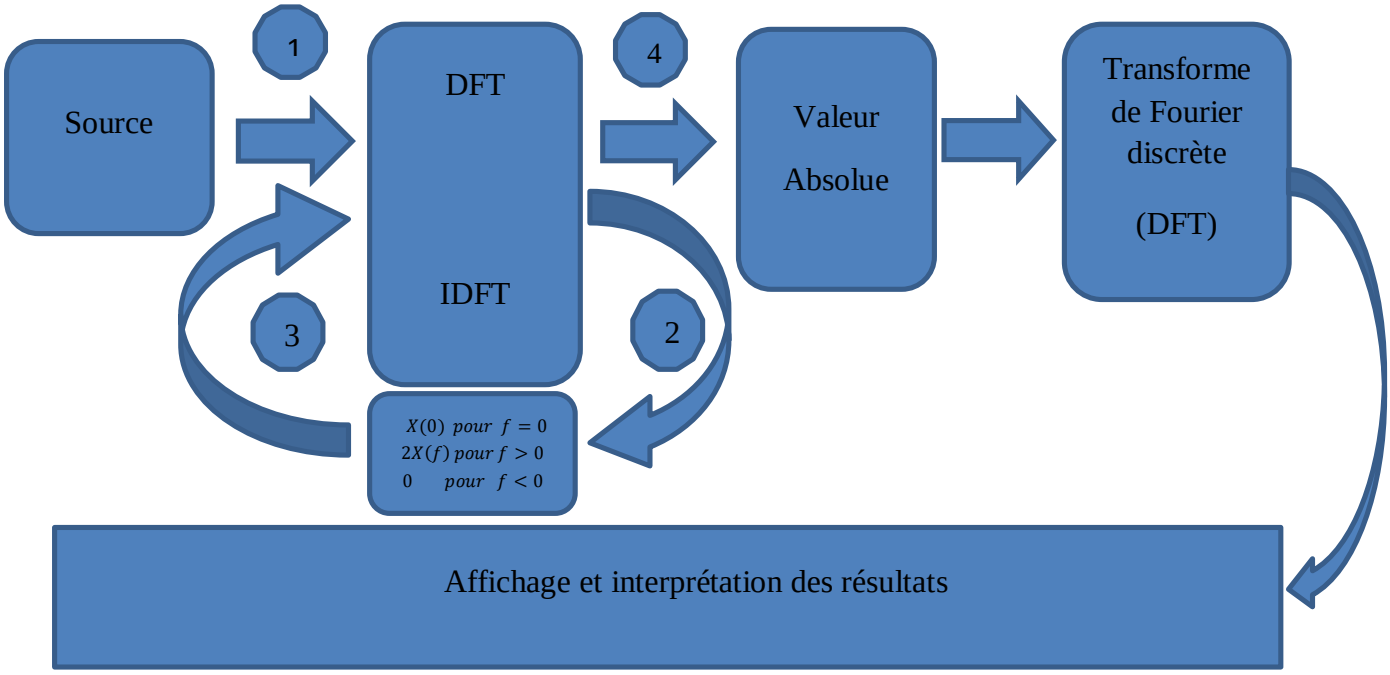


Figure III .13 : Schéma synoptique de solution 2.

Les étapes de la solution 2

- 1 : Obtenir la transformée de Fourier sur le signal d'entré.
- 2 : Mettre les termes de fréquence négative à zéro.
- 3 : Obtenir la transformée de Fourier inverse.
- 4 : Différencier les sorties FFT et IFFT.

Notre deuxième solution a comme objectif de diminuer l'utilisation des ressources de la carte FPGA. La nouvelle architecture est composée de 5 Multiplexeurs et un seul bloc FFT v 9.1 pour faire la transforme de Hilbert au lieu de deux blocs.

Un sous-système (subsystem1) pour éliminer les termes négatifs et un autre sous-système (subsystem2) pour calculer la valeur absolue qui est composé des blocs Xilinx suivant (voir figure III.14).

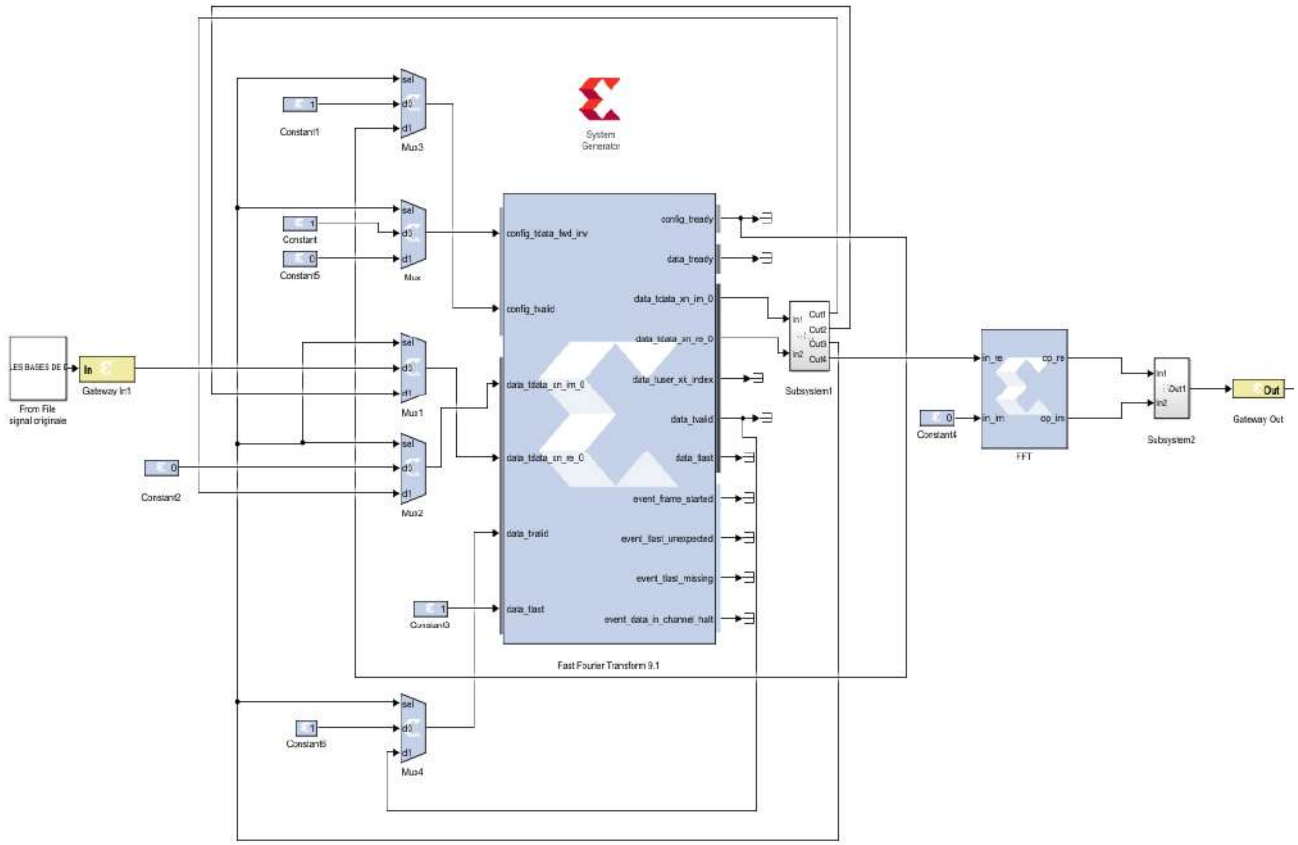


Figure III.14 : Solution (2).

- **Les Multiplexeurs**

Pour cette solution, Cinq Multiplexeurs à deux entrées sont utilisés. Les deux premiers sont utilisés pour la sélection de la partie réelle et imaginaire du signal d'entrée et de la partie réelle et imaginaire du signal de retour qui est nécessaire pour calculer de l'IFFT

Trios autres multiplexeurs sont utilisés pour sélectionner le signal de démarrage approprié de la mise en œuvre pour calculer FFT et IFFT.

- **La ligne de sélection**

La ligne de sélection de ces multiplexeurs joue un rôle très important pour avoir un bon résultat. Cette ligne de sélection est contrôlée à l'aide d'un programme Matlab que nous avons développé.

- **Sous-système** (subsystem1)

Le subsystem1 (voir figure III .15) est composé d'un bloc (Mcode) pour importer le fichier (.mat) et des blocs Delay ainsi que des blocs pour calculer (l'abs,Mul, Add, Squar…)

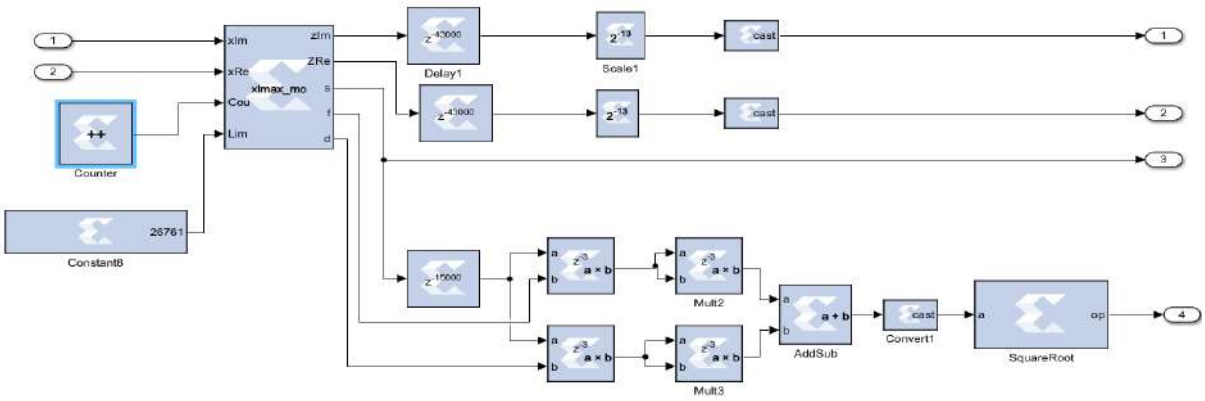


Figure III .15: Subsystem1.

- **Programme Matlab**

Pour différencier les sorties FFT et IFFT et contrôler la ligne de sélection nous avons développé le programme Matlab suivant.

```
function [zIm,ZRe,s,f,d] = xlmax_mo(xIm,xRe, Cou, Lim)

if Cou < Lim
    zIm = xIm*2;      % multiplie la partie imaginaire par deux
    ZRe=xRe*2;        % multiplie la partie reel par deux
    s=0;              % metre la ligne de selection a zero
    f=0;              % donne un signal nulle a la sortie f
    d=0;              % donne un signal nulle a la sortie d
else
    f=xIm;            %transfer les donnes d'entree xIm ver la sortie f
    d=xRe;            %transfer les donnes d'entree xRe ver la sortie d
    zIm =0;           %donne un signal nulle a la sortie zIm
    ZRe=0;            %donne un signal nulle a la sortie zRe
    s=1;              % metre la ligne de selection a un
end
end
```

- **Bloc FFT et valeur absolue (subsystem2)**

La drnière étape du traitement est représentée par un bloc simple pour calculer FFT et la valeur absolue (subsystem2), (voir la figure III.16**)**

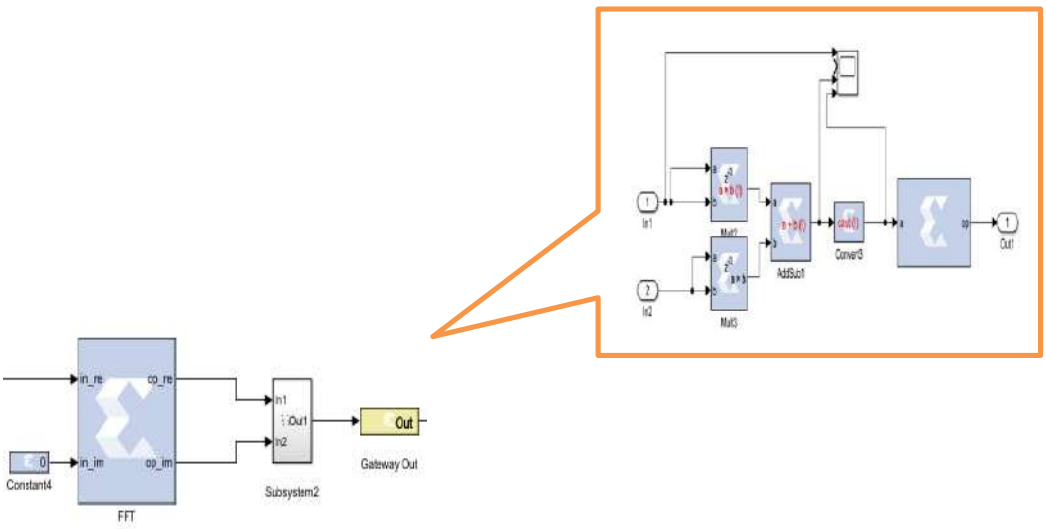


Figure III.16 : bloc FFT et subsystem2.

Dans la partie suivante nous discutons les résultats obtenus à partir de la mise en œuvre des architectures proposées (solution1 et 2), en notant que les résultats sont les mêmes pour la partie 1 et 2, pour cela nous affichons un seul résultat.

6. Discussion de résultat

Pour identifier le type de défaut nous avons exporté les données obtenues à la sortie de notre système vers le Workspace en traçant le spectre du signal obtenus (voir les figures III.17 et III.18).

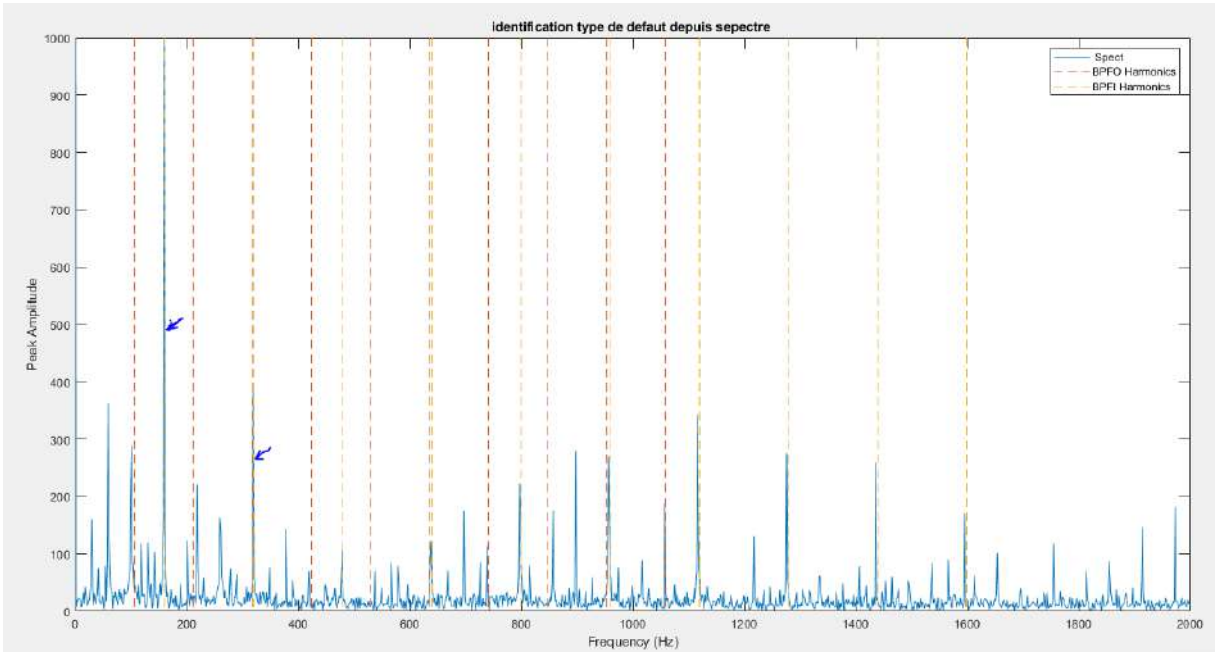


Figure III.17 : Spectre du signal de défaut sur la bague interne, base des données106 DE.

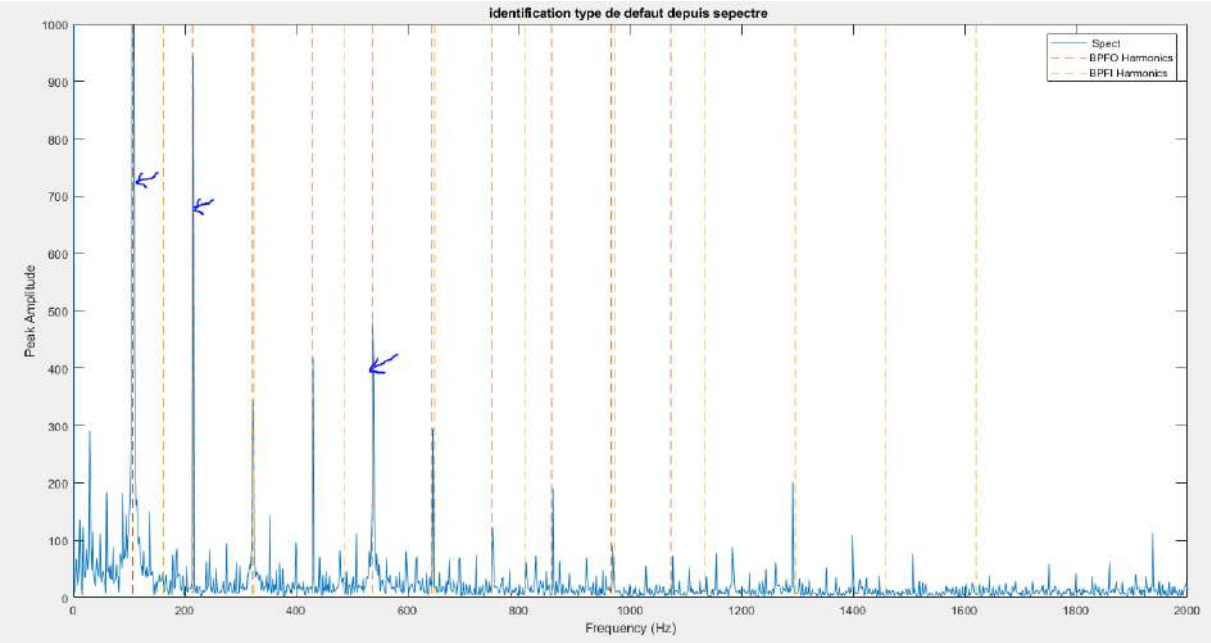


Figure III.18 : Spectre du signal de défaut sur la bague externe, base des données130 DE.

Sur les deux figures précédentes (voir les figures III.17et III.16).Nous avons remarqué que les pics des spectres sont identiques avec les lignes qui représentent les défauts (interne ou externe) et ses harmoniques. Le résultat observé dans les deux figures, il est obtenu avec deux bases de données déférentes à l'entrée.

7. Validation des résultats

Les résultats sont identiques à celle dans la base des données donc le fonctionnement de notre programme est correct .pour tester le fonctionnement nous avons appliqué 4 bases des données à l'entrée comme il est montré dans le tableau suivant :

Tableau III-1 : Teste de programme avec des fichiers porte un défaut connu

Diamètre de défaut (pouc)	Charge de moteur (chev)	Vitesse De moteur (tour /m)	Fichier de la base du donnée	Diagnostique par Système générateur Matlab
0.007	0	1797	IR007_0.mat	Défaut dans la bague interne
			OR007@6_0.mat	Défaut dans la bague externe
	2	1750	IR007_2.mat	Défaut dans la bague interne
			OR007@ 6_2.mat	Défaut dans la bague externe

1ch=745.69 wattes. 1poc=2.54 centimètre.

8. Implémentation sur la carte FPGA

Pour l'implémentation de notre solution du système générateur vers la carte FPGA on a opté pour deux solutions :

- ❖ Hardwar co-simulation.
- ❖ IP catalog (Pour la vérification des ressources utilisées).

Pour cela on suit les étapes suivantes :

8.1 Compilation

Il ne reste plus qu'à paramétrer le bloc System Generator et lancer la génération, comme il est montré (voir figure III.19)

Les configurations les plus importantes sont :

• Le type de compilation choisi.

• La carte FPGA ciblée.

• Le langage de description du matériel voulu.

• L'outil de synthèse



Figure III.19 : Lancement de la génération.

8 .2 Compilation de notre architecture avec Hardwar co-simulation.

La compilation avec cod_simulation permet de collectionner tous les blocs de notre solution (1) dans un seul bloc intitulé Jtag Cosim (Voir figure III. 20) . La simulation de notre resultat sur la carte FPGA. Et on a vu le resultat de traitement final representé dans la figure III. 21.

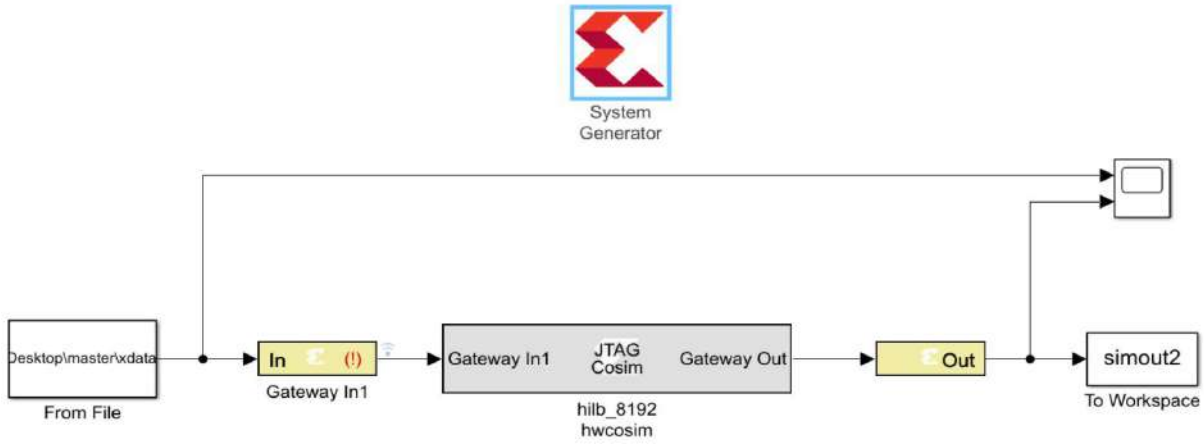


Figure III. 20 bloc obtenu ave cod_simulation.

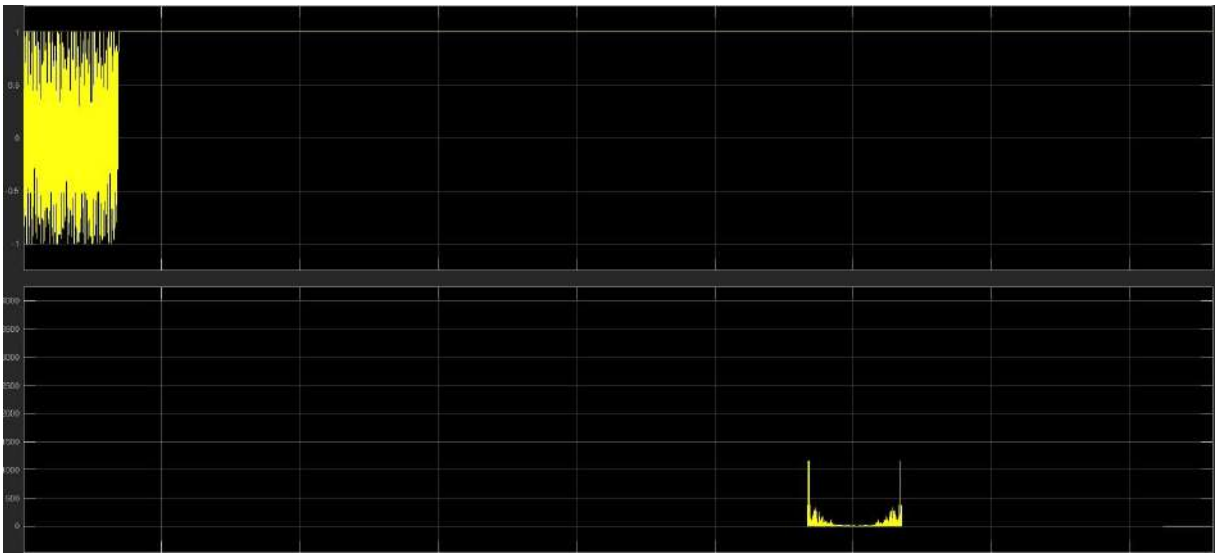


Figure III. 2 1: Resultat de l'implimentation sur la carte FPGA avant et après.

8 .3 Compilation de notre architecture avec IP catalogue.

Notre solution est implémentée sur FPGA Digilent ZYBO Zynq ZC7020, Pour la visualisation des ressources utilisées par notre solution(1) et (2), une implémentation par IP catalogue est effectuée.

Les étapes suivantes sont nécessaires pour faire la conception de notre architecture sur vivado.

1) Génération d'un IP bloc qui contient notre travail nommé hilb8192 ce dernier représente l'architecture de la solution (1).
2) Ajout du processeur Zynq.
3) Ajout du bloc IP nommé hilb8192.
4) Connexion des interfaces.

L'architecture sur Vivado est représentée comme suivant :

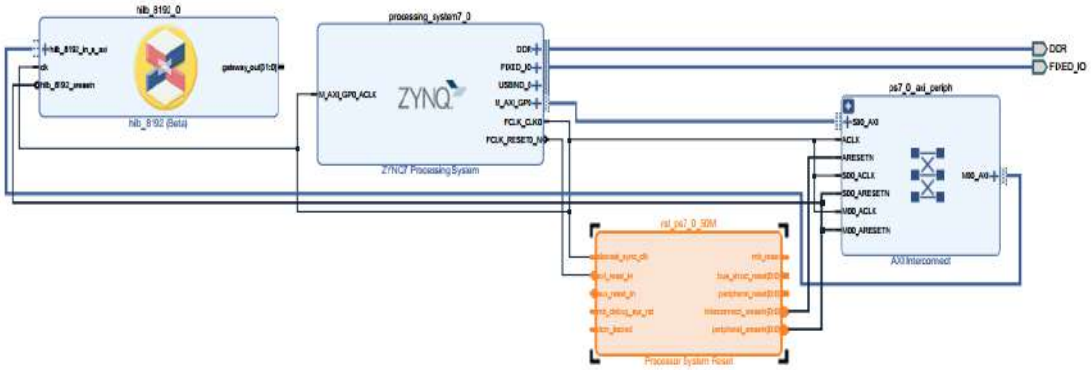


Figure III .22 : architecture sur vivado de notre travail hilbert8192.

9 .Utilisation des ressources pour la solution (1)

Après la génération du bitstream on a obtenu les résultats suivants pour l'utilisation des ressources :

Tableau III.2 : Utilisation des ressources du FPGA pour l'implémentation du hilb8192.

RESOURCE	UTILIZATION	AVAILABLE	UTILIZATION%
LUT	41224	53200	77.49
LUTRAM	576	17400	3.31
FF	45085	106400	42.37
BRAM	60	140	42.86
DSP	16	220	7.27

. Les résultats de l'implémentation montrent que les LUTs sont les plus utilisées dans l'architecture proposée, avec un pourcentage d'utilisation inférieur à 78%. Le reste des résultats sont donnés dans le Tableau III.1**.** Les ressources restantes sont suffisantes pour implémenter d'autres coprocesseurs.

Le graphe suivant présente l'utilisation des ressources pour notre designe, on a remarqué que la plus grande partie des ressources est utilisée par le bloc hilb_8192.

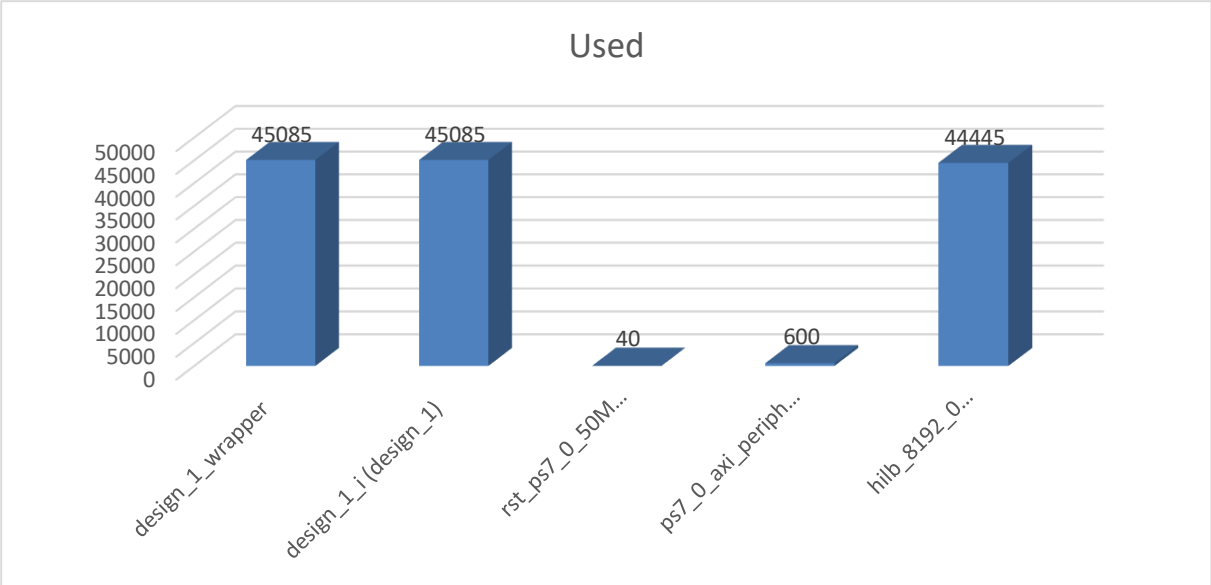


Figure III. 23. Utilisation des ressources de hilb_8192 solution(1) par rapport au rst et périphérique axi.

Dans le bloc hilb_8192, la figure III.19 montre l'utilisation des ressources dans ce bloc.

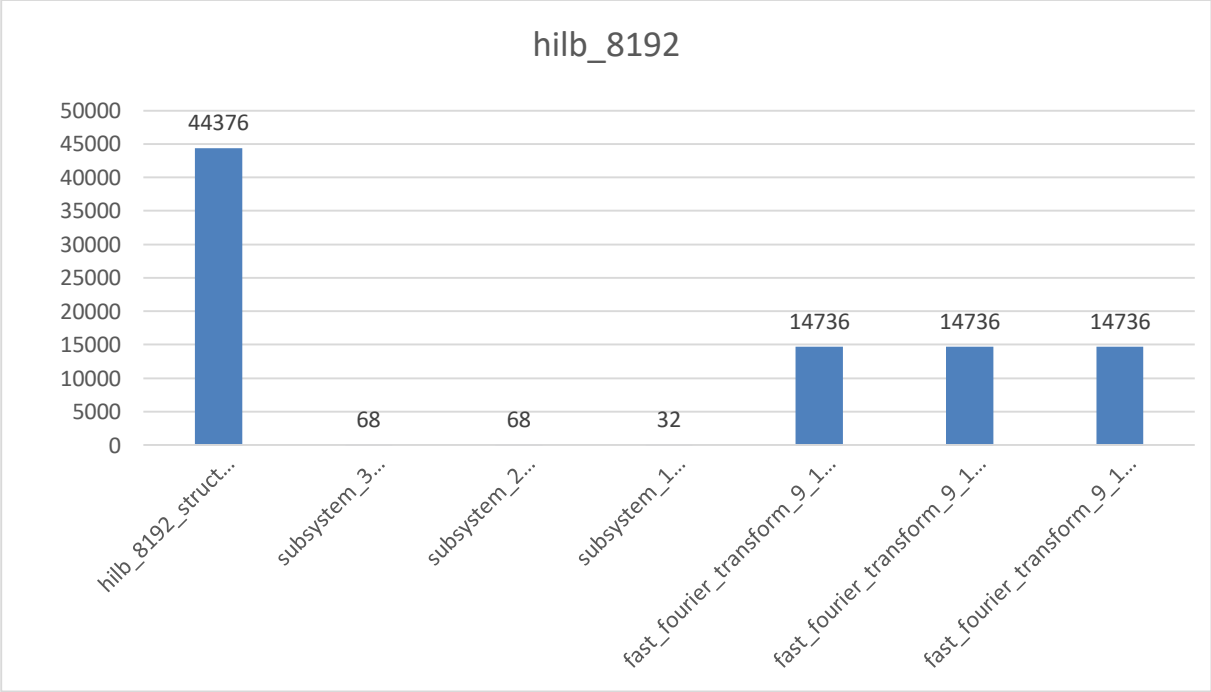


Figure III. 24 : Utilisation des ressources de chaque bloc de hilb_8192 solution1.

On voit très bien que les blocs FFT utilisent la plus grande partie des ressources réservées pour hilb_8192, pour cela on a proposé une architecture optimisée pour utiliser un seul bloc FFT au lieu de deux.

10. Utilisation des ressources pour la solution 2

Après l'exécution de l'implémentation on a obtenu les résultats suivants pour l'utilisation des ressources :

Resource	Utilization	Available	Utilization %
LUT	93145	53200	175,08
LUTRAM	78871	17400	453,28
FF	94279	106400	88,61
BRAM	20	140	14,29
DSP	12	220	5,45

Tableau III.3 : Utilisation des ressources du FPGA pour l'implémentation de la solution (2).

Les résultats de l'implémentation montrent que les ressources ne sont pas suffisantes et par conséquent le bitstream ne peut pas être généré.

Le graphe suivant présente l'utilisation des ressources pour notre designe, on remarque que la plus grande partie des ressources est utilisée par le bloc hilb_8192 optimisé.

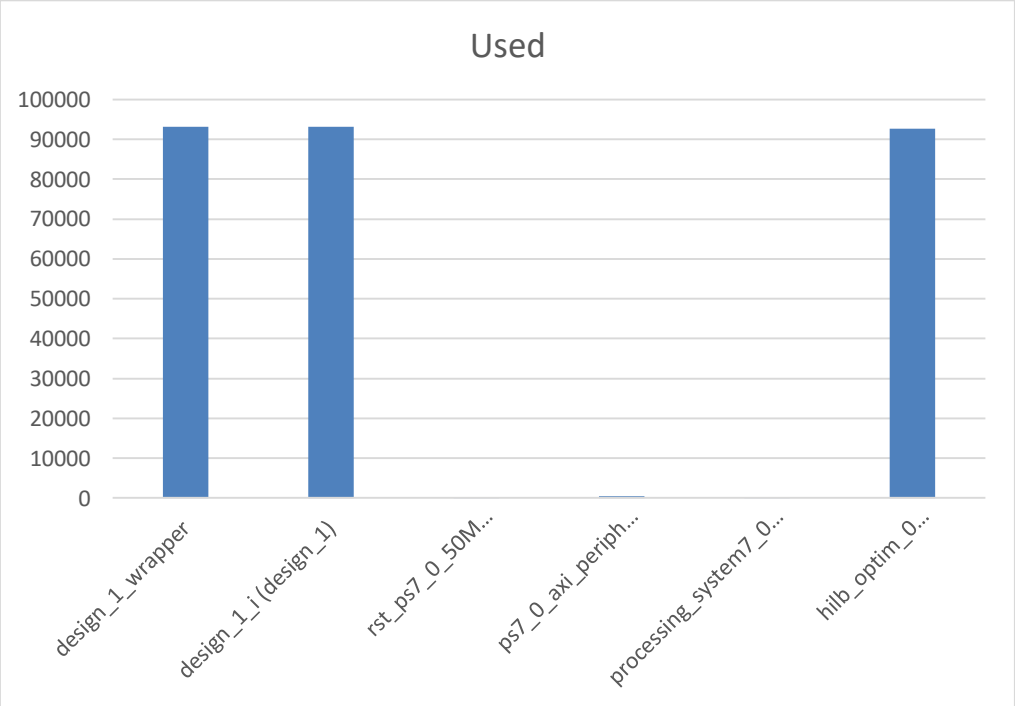


Figure III. 25. Utilisation des ressources de hilb_optim solution (2), par rapport au rst et périphérique axi.

Dans le bloc hilb_optim, la figure III.25 montre l'utilisation des ressources dans ce bloc.

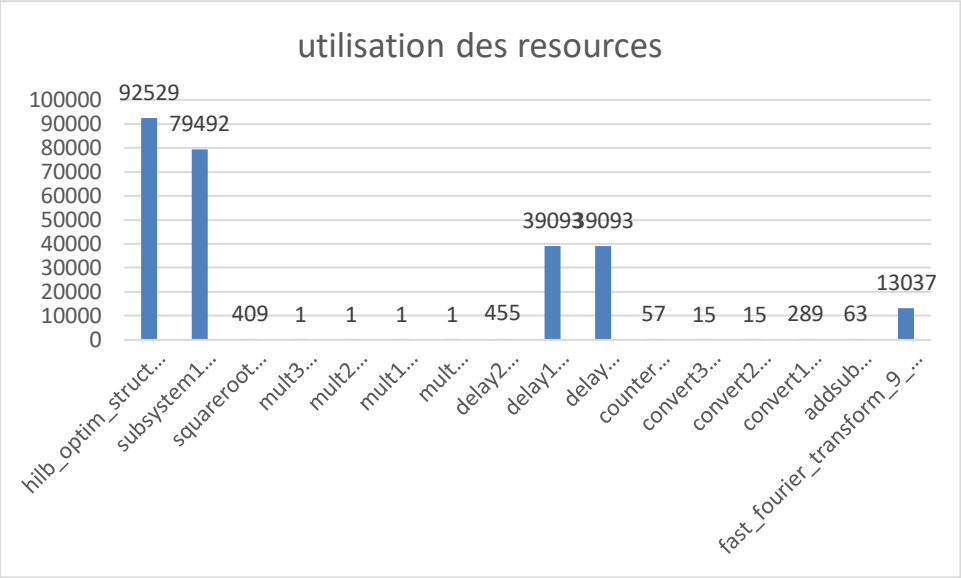


Figure III. 26 : Utilisation des ressources de chaque bloc de hilb_optim. Solution (2).

On voit très bien que l'utilisation des ressources du bloc FFT a diminué par rapport aux trois blocs FFT, mais on constate une grande utilisation pour le bloc delay qui a été utilisé pour créer un temps de retard pour terminer le calcul de la FFT et la re-injection le signal dans le même bloc. Comme solution dans le futur on propose l'utilisation d'une mémoire FIFO qui utilise moins de ressources.

11.Conclusion

Dans ce chapitre nous avons présenté notre implémentation et simulation à l'aide de l'outil de programmation Système Générateur Xilinix, deux solutions sont proposées, à savoir solution(1) et solution(2), ainsi que les résultats de simulation du Simulink. Les résultats d'implémentation sont présentés aussi avec la méthode Co-simulation, enfin une implémentation avec Ip catalogue est faite, pour générer le bit Stream et visualiser l'utilisation des ressources sur la carte FPGA.

Conclusion générale

L'objectif de ce mémoire est l'implémentation d'une technique d'analyse spectrale pour diagnostiquer et détecter les défauts des roulements. La transformé de Hilbert est une technique importante pour la maintenance moderne qui donne de bon résultat de diagnostic. Dans ce travail nous avons utilisé les circuits FPGA comme organe d'implémentation.

D'abord, nous avons décrit que ce qu'un FPGA, leur place parmi les composants numériques, cela nous a permis de déduire que la technologie FPGA est une évolution des composants logiques et le besoin croissant de composants plus performants.

Ensuite on a utilisé la plateforme de prototypage rapide Xilinx System Generator (XSG) qui est une partie de la plateforme de développement de Simulink. Cette approche nous a permis la simulation de notre système et faire un gain en temps de développement grâce aux outils proposés.

Une implémentation est faite sur une base des données connue pour la validation de nos solutions, les résultats de l'implémentation sont satisfaisants et donnent un bon diagnostic des defaults.

Bibliographie :

[1] : Djaballah Saïd, Mémoire de Magister, « Etude et optimisation de la transformée en ondelettes pour la détection des défauts dans les roulements ». Université Mohamed Chérif Messaâdia de Souk-Ahras.pdf

[2] Dechicha farid,ghrabli rafik, Mémoire de Master « Etude et implémentation d'un system pour la détection des défauts dans les roulement, université yahiya fares de medea.pdf

[3] : LAIFAOUI Wafa. « Implémentation des filtres adaptatifs sur une carte FPGA pour l'annulation d'écho acoustique » Université Mohamed Larbi Ben M'hidi - Oum El bouaghi.pdf

[4] TARANI GAUR« Étude et conception de la transformée de Fourier rapide (FFT) et du modulateur OFDM » recherche pratique ; INSTITUT DE TECHNOLOGIE ET DE SCIENCE DE BIRLA, PILANI, CAMPUS HYDERABAD.pdf

[5]: DESIGN FOR EMBEDDED IMAGE PROCESSING ON FPGAS Donald G. Bailey Massey University, New Zealand.pdf

[6] : FARADJI Mohamed, mémoire sur Etude et implémentation d'un réseau de neurones sur une plateforme de type Pynq, UNIVERSITE YAHIA FARES DE MEDEA, 2019.pdf

[8] : TOUZET, Claude. Les réseaux de neurones artificiels, Cours, exercices et travaux pratiques :" Introduction au connexionnisme ",129p :1992.pdf

[7] : HOCINI Ilyes et DERGAOUI Mohamed, mémoire sur Réalisation d'un modulateur et démodulateur OFDM sur FPGA ZYNQ, Université Aboubakr Belkaïd – Tlemcen, 2019.

[9]: design of embedded processors, version2, module 4 EE IIT,Kharagpur.

[10]: Help Matlab.

[11]: bibliothèque System Generator.

[12]: https://csegroups.case.edu