الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DE TECHNOLOGIE

Département du Génie Électrique



Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme
De Master en

Option : Automatique des systèmes

THEME

COMMANDE SCALAIRE D'UNE MAS TRIPHASEE EN UTILISANT UN CIRCUIT FPGA

Proposé et dirigé par,

MR : KABACHE NADIR

Préparé et présenté par,

Melle: BOUAISSA Souad

Remerciements

Je remercie dieu le tout puissant de m'avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Je tiens à exprimer vivement mon profonde gratitude à mon promoteur : N.KABACHE pour leur confiance, leur encouragement, leur suivi et pour les conseils qu'il a apporté pour l'achèvement de ce projet.

Je remercie également tous les enseignants de l'université de médéa YAHIA FARES

Je tiens à remercier très chaleureusement les nombres de jury, qui ont accepté la charge de juger ce mémoire malgré un très court délai a consacre à la lecture.

Dédicace

Avant tout, Je tiens à remercie le bon dieu, et l'unique qui m'offre le courage et la volonté pour terminer ce travail.

Je dédie ce modeste travail

A mon Père.

A ma mère.

A mes frères et mes sœurs.

A KAMEL.

A mes Copines MESSEAOUDA ET HADJIRA.

الملخص

يهدف هذا العمل إلى تنفيذ السيطرة على حلقة مغلقة بواسطة بطاقة FPGA SPARTAN 3 E ونريد من هذا العمل أن نحسن أداء الجهاز الغير متزامن .في البداية نقوم بعرض بطاقة FPGA ثم شرح مختلف طرق المحاكاة على هذا الجهاز.ثم نبرمج هذا الجهاز بواسطة برنامجVHDL ثم نطبق هذا البرنامج على بطاقة FPGA.

الكلمات المفتاحية :

الجهاز الغير متزامن ،السيطرة العددية,FPGA,VHDL,SPARTAN 3 E

Résumé

Ce travail a pour but d'améliorer les performances de la machines asynchrone grâce au circuit FPGA, on présente dans notre projet l'implémentation de la commande scalaire en boucle fermé sur la carte FPGA SPARTAN 3 E, en premier lieu on expose le circuit FPGA et leur langage de programmation après on fait des simulations de la commande scalaire pour la machine asynchrone ensuit on décrit le programme VHDL de la commande on passant par les différents étapes de l'implémentation tel que la simulation, la synthèse, le placement et le routage jusqu'au la création de fichier point bit(.bit) à la fin on a implémenter la commande et on a analysé le comportement de moteur.

Mots clés :

Machine Asynchrone, commande scalaire, SPARTAN 3 E, VHDL, FPGA.

Abstract

This work aims to improve the performance of the induction motor through the FPGA, we present our project in the implementation of the scalar open loop control on the FPGA board SPARTAN 3 E, first we presents the FPGA and programming language then made simulation of scalar control for different induction motor follows describes the VHDL program to control it through the vas stages of implementation such as simulation, synthesis, placement and routing to create the bit file (.bit) at the end we implemented the control and motor behavior were analyzed.

Key words:

Asynchronous machine, scalar control, SPARTAN 3 E, VHDL, FPGA.

SOMMAIRE

TABLE DES MATIERES

RESUME

TABLE DES METIERES

TABLE DES FIGURES

ISTE DES ACRONYMES

INTRODUCTIONN GENERALE.....1

CHAPITRE I

LES CIRCUITS FPGA ET LEURS APPORTS A LA COMMANDE DES MACHINES

I.1)-Introduction Page 3

I.2)-Les différents types des circuits numériques Page 4

I.3)-Les circuits logiques programmables Page 5

I.4)-Les circuits FPGA Page 6

I.4.1)-Domaines d'application des FPGA Page 7

I.4.2)-Principaux vendeurs D'FPGA Page 8

I.4.3)-Configuration et reconfiguration des FPGA Page 8

I.4.4)-Technologie de programmation des FPGA Page 9

I.4.4.1)-Technologie a base de ram (XILINX et ALTERA) Page 9

I.4.4.2)-Technologie a base d'EEPROM ou FLASH (LATTICE ET ACTEL) Page10

I.4.4.3)-Technologie a base d'anti-fusibles(ACTEL) Page 11

I.4.5)-Architecture interne des FPGA Page11

I.4.6)-Les cinq caractéristiques de la technologie FPGA Page13

I.5)-langage de description VHDL Page 15

I.6)-Prototypage des FPGA Page16

I.6.1)-La programmation des FPGA par les outils HDL Page16

I.6.1.1)-spécification du design Page17

I.6.1.2)-Conception du design Page17

I.6.1.3)-Placement et routage Page18

I.6.1.5)-Intégration et implémentation Page19

I.6.2)-La programmation des FPGA par XSG/MATLAB-SIMULINK Page20

I.7)-Conclusion Page21

CHAPITRE II

LA COMMANDE SCALAIRE DE LA MAS

II.1)-Introduction Page22

II.2)-Constitution et principe de fonctionnement Page23

II.3)-Modélisation de la machine asynchrone Page24

II.4)-Equations générales Page25

II.5)-Transformation de « PARK » Page28

II.6)-Simulation du modèle moteur asynchrone triphasé Page31

II.7)-Commande scalaire du moteur asynchrone Page32

II.8)-Résultat de simulation de la commande scalaire Page34

II.9)-Conclusion Page37

CHAPITRE III

IMPEMENTATION DE COMMANDE SCALAIRE D'UN MAS

III.1)-Introduction Page 38

III.2)-Le codeur incrémental Page 38

III.2.1)- Principe de fonctionnement général d'un codeur Page 38

III.2.2)-Principe du codeur incrémental Page 39

III.3)- travail de l'encodeur Page 40

III.3.1)-Les étapes de VHDL Page 40

III.3.2)-Les résultats de simulations de l'encodeur Page 41

III.4)-La commande dans VHDL Page 42

III.5)-Description de la stratégie(MLI) Page 44

III.5.1)- Le signal triangulaire (porteuse) Page 44

III.5.2)- Génération de signal sinusoïdal Page 45

III.5.3)-simulation software des signaux Page 46

III.6)-description VHDL Page 47

III.7)-Partie de Réalisation Page 47

III.8)- Teste hardware Page 50

III.9)- Conclusion Page 51

LISTE DES FIGURES ET TABLEAUX

Figure (I.1) : *Reprogrammable sur site d'un FPGA* Page 9

Figure(I.2) : *caractéristiques des technologies SRAM* Page10

Figure(I.3) : *Caractéristiques des technologies FLASH.* Page10

Figure(I.4) : *Caractéristiques des technologies ANTI-FUSIBLES* Page 11

Figure(I.5) : *architecture interne d'un FPGA Fabriqué XILINX* Page 12

Figure(I.6) : *Différents block de langage VHDL* Page 16

Figure(I.7) : *Mode d'exécution matériel de la CAO* Page17

Figure(I.8) : *les étapes de développement une application sur FPGA.* Page 19

Figure(II.1) : *constitution de moteur asynchrone à cage.* Page23

Figure(II.2) : *Représentation schématique d'une machine asynchrone triphasée* Page25

Figure(II.3) : *Représentation du passage d'un système triphasé à celui biphasé* Page28

Figure(II.4) : *Courant statorique Isa* Page31

Figure(II.5) : *Couple électromagnétique* Page31

Figure(II.6) : La *vitesse de rotation.* Page31

Figure(II.7) : *schéma générale de la commande scalaire en boucle fermée.* Page33

Figure(II.8) : *La vitesse de la commande scalaire.* Page34

Figure(II.9) : *le couple électromagnétique de la commande scalaire* Page34

Figure(II.10) : *le flux FDN* Page35

Figure(II.11) : le *flux FQN* Page35

Figure(II.12) :*le courant Ids de la commande scalaire* Page36

Figure (II.13**)** : *le courant Iqs de la commande scalaire* Page36

Figure(III.1) : *Codeur Incrémentale* Page39

Figure(III.2) : *signal A et B* Page 40

Figure(III.3) : *le compteur réalisé* Page 41

Figure(III.4) : *résultats de la vitesse* Page 41

Figure(III.5) : *Le programme d'un PI* Page 42

Figure(III.6) : *Principe de la MLI sinusoïdaux-triangulaire* Page 44

Figure(III.7) : *La lecture des points* Page 45

Figure(III.8) : *le signal de 0 à 2π* Page 45

Figure(III.9) : *Le signal triangulaire* Page 46

Figure(III.10) : *les3 signaux pour une vitesse de15 rad/s* Page 46

Figure(III.11) : *l'entité d'une description VHDl.* Page 47

Figure(III.12) : *Le moteur asynchrone* Page 47

Figure(III.13) : *L'encodeur.* Page 47

Figure(III.14) : *le signal A* Page 48

Figure(III.15) : *la structure externe de la carte FPGA* Page 49

Figure(III.16) : *la structure de pupitre électrique* Page 49

Figure(III.17) : *Oscilloscope numérique* Page 49

Figure(III.18) : *la vitesse Wr* Page 50

Figure(III.19) : *les signaux sinusoïdaux sina et sinb* Page 50

Tableau(I.1) : *comparaison des différentes solutions numériques.* Page 4

Tableau(I.1) : *Statistiques du marché occupé par les vendeurs d'FPGA* Page 8

LISTE DES ACRONYMES

ASIC :	**A**pplication **S**pecific **I**ntegrated **C**ircuit.
CAO:	**C**onception **A**ssistée par **O**rdinateur.
CLB :	**C**onfiguration **L**ogic **B**loc.
CPLD :	**C**omplex **P**rogrammable **L**ogique **D**evise.
CORDI :	**C**oordinate **R**otation **D**igital **C**omputer.
DSP :	**D**igital **S**ignal **P**rocessor.
EPROM :	**E**rasable **P**rogrammable **R**ead **O**nly **M**emory.
EEPROM:	**E**lectrically **E**rasable **P**rogrammable Read **O**nly **M**emory.
EEBLD:	**E**lectrically **E**rasable **P**rogrammable **L**ogic **D**evice.
FPGA:	**F**ield **P**rogrammable **G**ate **A**rray.
IOB:	**I**nput **O**utput **B**loc.
LUT:	**L**ook **U**p **T**able.
MLI:	**M**odulation de **L**argeurs d'**I**mpulsions.
PED :	**P**rogrammable **L**ogique **D**evice.
PROM :	**P**rogrammable **R**ead **O**nly **M**emory.
PI :	**P**roportional **I**ntegral.
PAL :	**P**rogrammable **A**rray **L**ogic.
PLD :	**P**rogrammable **L**ogic **D**evice.
ROM :	**R**ead **O**nly **M**emory.
RTL:	**R**egister **T**ransfer **L**evel.
SRAM:	**S**tatic **R**andom **A**ccess **M**emory.
LCA:	**L**ogic **C**ell **A**rray.
MAS:	**M**achine **A**synchrone.
GAL:	**G**eneric **A**rray **L**ogic.
VHDL:	**V**hsic(**V**erry **H**ight **S**peed **I**ntgrated **C**ircuit) **H**ardware **D**escription **L**angauge.

TABLE DES NOTATIONS ET DES SYMBOLES

A ,B,C	Axes liés aux enroulements triphasés.
d,q	Axes de référentiel de park.
[P]	Matrice de Park.
A (rad)	Angle entre le stator et le rotor.

Grandeurs électriques et magnétiques au stator

Vs(V)	Tension statorique.
Vsabc(V)	Tension statorique phase a,b ou c.
Vsdq(V)	Tension statorique sur l'axe d ou q.
Is(A)	Courant statorique.
Isabc(A)	Courant statorique phase a,b ou c.
Isdq(A)	Courant statorique sur l'axe d ou q.
Rs(Ω)	Résistance statorique par phase.
ls(H)	Inductance propre statorique par phase.
Ms(H)	Inductance mutuelle entre deux phases statorique.
Ls(H)=ls-Ms	Inductance statorique par phase.
ϕsabc(Wb)	Flux statorique phase a, b ou c.
ϕsdq(Wb)	Flux statorique sur l'axe d ou q.

Grandeurs électriques et magnétiques au rotor

Vr(V)	Tension rotorique.
Vrabc(V)	Tension rotorique phase a, b ou c.
Vrdq(V)	Tension rotorique sur l'axe d ou q.
Ir(A)	Courant rotorique.
Irabc(A)	Courant rotorique phase a, b ou c.
Irdq(A)	Courant rotorique sur l'xe d ou q.
Rr(Ω)	Résistance rotorique par phase.
ϕrdq(Wb)	Flux rotorique sur l'axe d ou q.
Mr(H)	Inductance mutuelle entre deux phases rotorique.

INTRODUCTION GENERALE

INTRODUCTION GENERAL

Cette thèse à pour objectif principal d'étudier la commande scalaire du MAS. La robustesse, le faible cout, les performances et la facilité d'entretien font l'intérêt du moteur asynchrone dans de nombreuses applications industrielles ou grand public.

Vu que la machine asynchrone représente l'un des systèmes physiques les plus complexes, l'apparition de ces outils a donné un grand souffle pour améliorer de celle-ci. En effet, l'amélioration de la commande du MAS conduit à utiliser des algorithmes de commandes plus compliqués dont leurs réalisations dépasse parfois les capacités des outils à base de microprocesseurs. De à vient l'utilité d'exploiter la puissance de traitement des circuits FPGA pour aller loin dans la commande des MAS.

L'implémentation de la commande scalaire en boucle fermé, en utilisant un circuit FPGA. Cette commande repose sur la technique de codeur, par conséquent en utilisant le langage de description VHDL, sous l'environnement de développement de Xilinx, cette technique a été décrite, synthétisée, et implémenter.

Cette thèse s'articule sur trois chapitres :

Le premier chapitre est consacré à l'état d'art des circuits reconfigurables FPGA, on traitera l'architecture interne de ces derniers et une comparaison des FPGA et les circuits séquentiels classiques puis les différents modes de prototypage de ces circuits.

Le Deuxième chapitre présente brièvement des rappels sur le modèle de la machine asynchrone tout en utilisant une commande scalaire pour ajuster ses performances.

Le dernier chapitre est consacré à l'implémentation de la commande scalaire en boucle fermé. L'implémentation envisage principalement la réalisation d'un codeur en utilisant un circuit FPGA de type SPAARTAN-3 E. cette commande permettra de varier la vitesse d'un moteur asynchrone suivant le principe de la commande scalaire (U/F) constant.

CHAPITRE I

LES CIRCUITS FPGA ET LEURS APPORTS A LA COMMANDE DES MACHINES

I.1)-Introduction

Le circuit intégré (CI), aussi appelé puce électronique, est un composant électronique reproduisant une ou plusieurs fonctions électroniques plus ou moins complexes. Il intégré souvent plusieurs types de composants électroniques de base dans un volume réduit, rendant le circuit facile à mettre en œuvre. Le début des circuits intégrés a commencé après l'invention du transistor aux années quarante du siècle passé. Depuis, le monde de l'électronique numérique a entré dans le concept d'intégrer sur un même substrat plusieurs éléments électronique (transistors, résistances, capacités), c'est l'ère de circuit intégrés. Avec le temps, la capacité d'intégration a vu une énorme évolution, d'après Gordon Moore, le taux d'intégration de transistors dans ces circuits double tous les deux ans. Actuellement, ils existent des circuits numériques très réduite. Avec une telle capacité d'intégration, il est devenu possible de réaliser et d'implémenter des fonctions très compliquées avec des vitesses et des précisions satisfaisantes. Arrivant à cette étape, une question a été posée, concernant la possibilité de réaliser l'analogique avec le numériques. La réponse à cette question était avec l'arrivé des circuits logiques reconfigurables, en particuliers les circuits FPGA. Ces circuits sont susceptibles d'être reconfigurés (Architectures programmé modifiable) partiellement ou entièrement suivant l'application. Via ces circuits, la logique programmableséquentielle a été substituée par une logique câblée tout en restant dans le monde numérique mais avec l'avantage d'exécuter, à la fois, des processus séquentiels et combinatoires. Ces propriétés permettent aux circuits FPGA de résoudre plusieurs problèmes complexes dont la puissance de traitement dépasse la capacité des outils de traitement à base de microprocesseurs.

Ce premier chapitre s'inscrit dans un contexte qui traite les FPGA et leurs positions au sein des autres systèmes digitaux. [1] [2]

I.2)-Les différents types des circuits numériques

La grande partie des circuits numériques programmables sont issues de la célèbre architecture ‹‹Von-Neumann›› proposé par John Von Neumann en 1945 et porte son nom. Ensuite une autre architecture qui vienne pour compenser les lacunes de la précédente dans certains domaines et afin d'améliorer la cadence de calcule où le facteur temps d'exécution est le plus important. Cette architecture nommée ‹‹Architecture Harvard ›› qui porte le nom de l'université américaine qui la proposée sachons que parallèlement l'architecture ‹‹Von-Neumann ›› n'est pas figée mais on évolution. Mais le besoin croissant de composant très rapide a orienté les chercheurs à développer une autre solution qui sera complètement

différents des deux précédentes architectures. Cette solution réside dans le mode de programmation qui est devenu architectural à logique câblée inversement aux deux premières architectures précédentes où la programmation est séquentielle

D'une manière générale, il existe deux familles de circuits numériques :

- **Une famille logicielle :** Elle est nommée aussi solutions programmables du type ‹‹processeur›› où un traitement séquentiel relativement lent et programmation dépendante du composants. Cette famille regroupe les Microprocesseurs et les DSP.

- **Une famille matérielle :**Elle est nommée aussi solutions programmable du ‹‹type logique›› où un traitement parallèle en temps réel et une programmation architecturale avec un langage de description matériel HDL (Méthodologie de Conception CAO) indépendante du composant. Cette famille regroupe les ASIC, es CPLD, les FPGA, etc. [1]

Tableau(I.1) : *comparaison des différentes solutions numériques.*

	ASICS	**FPGA**	**DSP**
Performance	Très élevées	Elevées	Faible
Taille	Faible	Moyen	Faible
Consommation	Faible	Moyen	Très élevées
Intégration	Système sur puce	Système sur puce	Composants supplémentaires
Souplesse	Fonction figées	Reconfigurable	Programmable
Mise en œuvre	Complexe	Complexité moyenne	Complexité moyenne
Cout comportement	Très élevées	Moyen	Faible

- **Les circuits de types DSP/Microprocesseurs :** un rapport performance/cout faible, un temps de conception très court et une grande souplesse d'utilisation.

- **Les circuits du type spécialisé ASIC :** très performants mais avec un cycle de conception long et une architecture figée.

- **Les circuits de type FPGA :** Des performances proches des ASICS, un cout unitaire intermédiaire et un cycle de conception moyen et une architecture modifiable.

Le fonctionnement du mode séquentiel dépassent aujourd'hui 2GHz, la réduction du temps de cycle ne suffira pas à compenser les insuffisances de ce mode de fonctionnement. La fréquence d'horloge ou de fonctionnement des FPGA est relativement faible devant les microprocesseurs et les DSP et ne dépasse pas quelques centaines de mégahertz, mais cette faiblesse est largement compensée et même surpassée grâce au parallélisme de traitement. L'exploitation du parallélisme est une technique en pleine expansion dans les circuits numériques FPGA qui sont une alternative des DSP.

I.3)- Les circuits logiques programmable

Auparavant, la distinction était nette entre le logiciel et le matériel, le circuit logique programmable FPGA est venu s'introduire comme un hybride entre les deux approches. Les circuits logiques programmables et reprogrammables architecturalement sont classifiés en trois grandes familles les PLD, CPLD et FPGA.[3]

- **Les PLD (programmablelogicDevice) :** Famille des circuits programmables qui comprend les PAL, GAL.

  -PAL signifie (**Programmable arraylogic) :** c'est-à-dire réseau logique programmable. La programmation de ces circuits s'effectue par la destruction de fusibles. Une fois programmée on ne peut plus les effacer.

  - **GAL** ce qui signifie (**Genericarraylogic) :** le nom GAL a été déposé par LATTICE SEMICONDUCTOR. Leur fonctionnement est identique P.A.L.CMOS, ils sont programmables et effaçables éclectiquement.

- **LES CPLD ou EPLD** (**Complex Programmable LogicDevice**ou**Erasable Programmable LogicDevice**) : ces circuits sont composés de plusieurs PALs reliés entre eux par une zone d'interconnexion. Grace à cette architecture, ils permettent d'atteindre des vitesses de fonctionnement élevées.
- **Les FPGA** (**Field Programmable GateArray**) **:** ces circuits sont une évolution de CPLD. Récemment, ils intègrent également des mémoires entières, des multiplieurs et même des noyaux de processeur, ils ne sont pas effaçables électriquement.

I.4)- LES circuits FPGA

Les FPGA, sigle anglais qui signifie « Field Programmable Gates Arrays » traduit en français par réseau de portes programmables, sont des circuits intégrés programmable. Ils offrent la possibilité de réaliser des fonctions numériques plus au moins complexes, tout comme leurs homologues figés : les ASICS[4].Les FPGA sont développés récemment grâce aux progrès de la technologie VLSI, l'apparition de ce type de circuits est une révolution des systèmes digitaux et ouvrants des perspectives de traitement numérique inaccessibles auparavant. Les années 80 ont vu l'apparition des premiers circuits FPGA qui sont des circuits intégrés que l'on peut configurer en un temps relativement court pour réaliser n'importe quelle fonction logique « câblée » à bas cout pour une programmation de ses cellules logiques et ses interconnexions avec une restriction de ne pas épuiser les ressources du FPGA. Typiquement, un circuit FPGA haute densité peut contenir jusqu'à plusieurs millions d'éléments programmable. Pour réussir une application à base d' FPGA et afin d'obtenir un système plus performant, consommant un minimum de puissance, il est nécessaire de respecter un certain nombre de règles comme : [5]

- Bien connaitre les caractéristiques de FPGA ciblé pour assurer son adéquation avec les besoins du projet.
- Elaborer une méthodologie de conception
- Maitriser les outils d'implémentation et de choisir des outils de synthèse de qualité.

I.4.1)-Domaines d'application des FPGA

Les FPGA ont fait révolutionner certain domaines de contrôle numérique et de plus en plus utilisés pour intégrer des architectures numériques complexes. Ils sont devenus les plus populaires en matière d'implantation et prototypage des circuits numériques après leur apparition sur le marché en 1984. La clé de leurs réussites est l'aspect de programmation de ces derniers. Leurs utilisations actuelles couvrent es deux domaines : civil et militaire parmi ces application nous citons :

- **Informatique :** périphérique spécialisés.
- **Machinerie industrielle :** contrôleur pour machines.
- **Télécommunication :** traitement d'images, filtrage.
- **Instrumentation :** équipement médicales, prototypage.
- **Transport :** contrôle d'avions et métros.
- **Aérospatiale :** satellites.
- **Militaire :** radar, communication protégée, la détection ou la surveillance.

I.4.2)- Principaux vendeurs d'FPGA

Les fabricants des FPGA ne cessent pas d'améliorer leurs produits par l'efficacité et la puissance. L'ensemble des firmes (principaux vendeurs) qui conçoivent ce type de circuit sont : **Actel,Altera,Atmel,Cypress,lattice,Minc,QuickLogic, Xilinxet d'autres** (tableauI.2).

Tableau(I.2) : *Statistiques du marché occupé par les vendeurs d'FPGA.[17]*

Rank 2007	**Rank 2008**	**Company**	**Revenue ($M) 2007**	**Revenue ($M) 2008**	**Market Share 2008**
1	1	Xilinx	1.809	1.906	51,2%
2	2	Altera	1,216	1,323	35,5%
3	3	LatticeSemiconductor	229	222	6.0%
4	4	Actel	196	218	5.9%
6	5	QuickLogic	28	23	0.6%
5	6	CypressSemiconductor	32	21	0.6%
7	7	Atmel	14	9	0.2%
8	8	Chengdu Sino Microelctronics System	4	3	0.1%
		Others	0	0	0.0%
		Total market	**3,528**	**3,725**	**100.0%**

I.4.3)-Configuration et reconfiguration des FPGA

Un système reconfigurable est un système qui est constitué de composants ou entités à architecture modifiable afin de répondre à un objectif bien déterminé. Ce système reconfigurable dispose d'un mécanisme permettant de choisir une nouvelle configuration et de la mettre en place dans le cadre du processus de reconfiguration. Les circuits FPGA sont un type de ces circuits reconfigurables. Ils sont programmables ou configurables sur les cartes sur quelles ils sont implantés par l'utilisateur. Le FPGA est une abréviation anglaise de qui signifie réseau des portes programmables sur site ce qui est décrit sur la figure suivante (Figure I.1) [6].

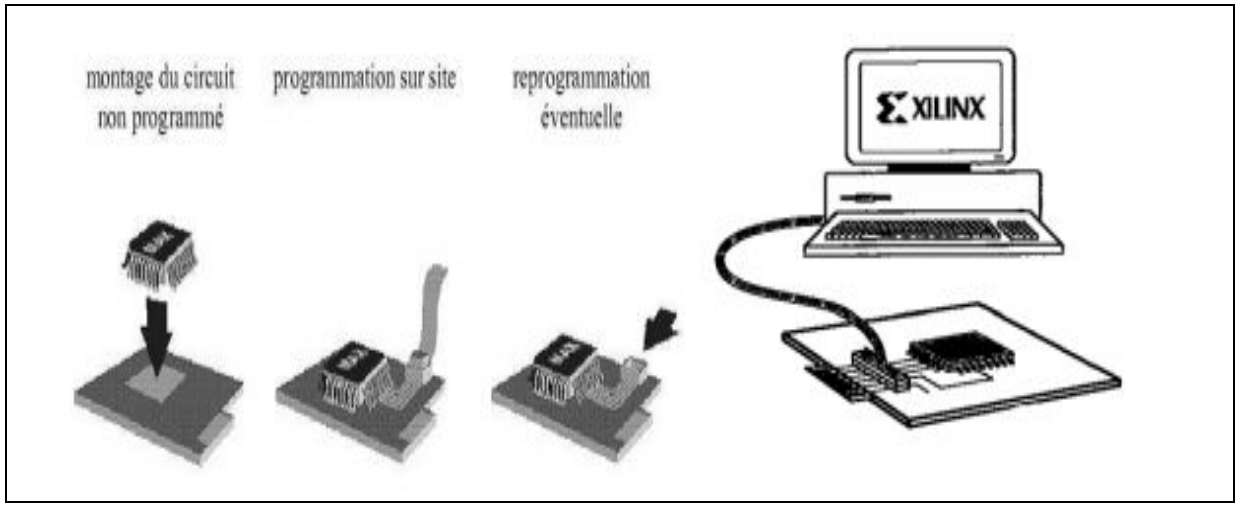


Figure (I.1) : *Reprogrammable sur site d'un FPGA*

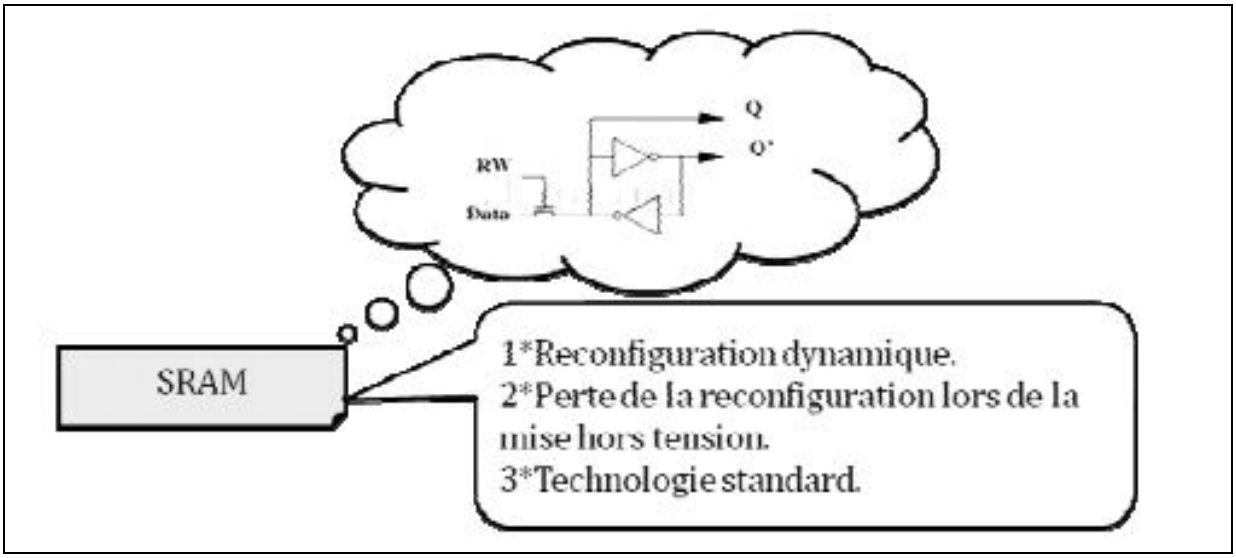
I.4.4)-Technologies de programmation des FPGA

Pour franchir les inconvénients susmentionnés des mémoires, et dans le but de faire un ensemble de technologies complémentaire adaptable suivant l'environnement des cahiers de charge, il existe trois types d'FPGA reprogrammable suivant la technologie de mémorisation pour répondre aux différentes applications. Ces trois principales technologies d'FPGA sont :

- Technologie de programmation par RAM. ;
- Technologie de programmation par EEPROM ou FLASH.
- Technologies de programmation par ANTI-FISIBLE.

I.4.4.1)-Technologie à base de RAM (XILINX ET ALTERA)

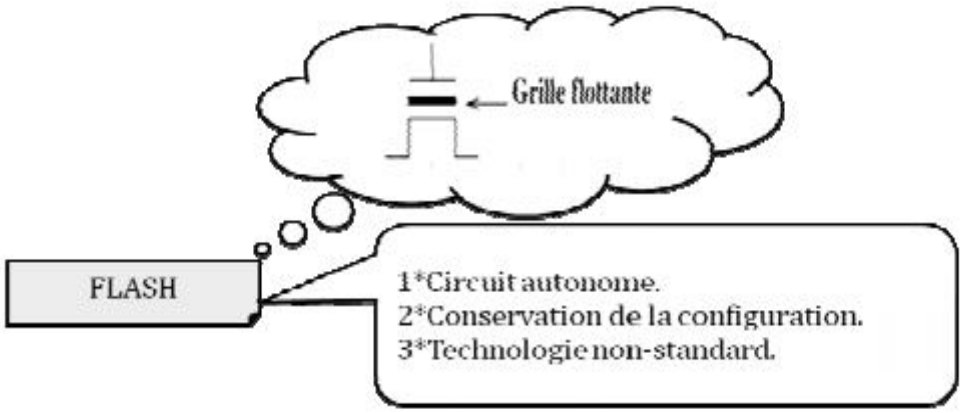
Cette technologie permet d'avoir une reconfiguration rapide des FPGA(Fig.I.2).Les points de connexions sont des ensembles de transistors commandés. L'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du FPGA dans une autre mémoire Flash.



Figure(I.2) : *caractéristiques des technologies SRAM .*

I.4.4.2)-Technologie à base d'EEPROM ou FLASH (LATTICE et ACTEL)

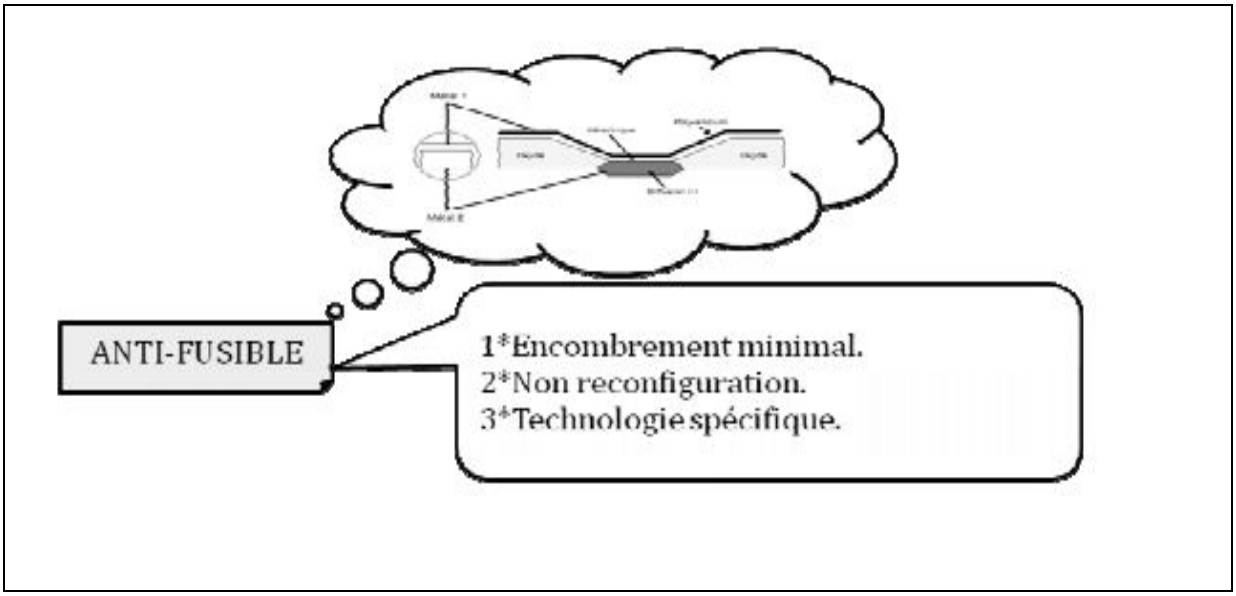
Cette technologie garde sa configuration mais un nombre limité de configuration avec une configuration plus lente par rapport à SRAM (Fig. I.3).



Figure(I.3) : *Caractéristiques des technologies FLASH.*

I.4.4.3)-Technologie à base d'anti-fusibles(ACTEL)

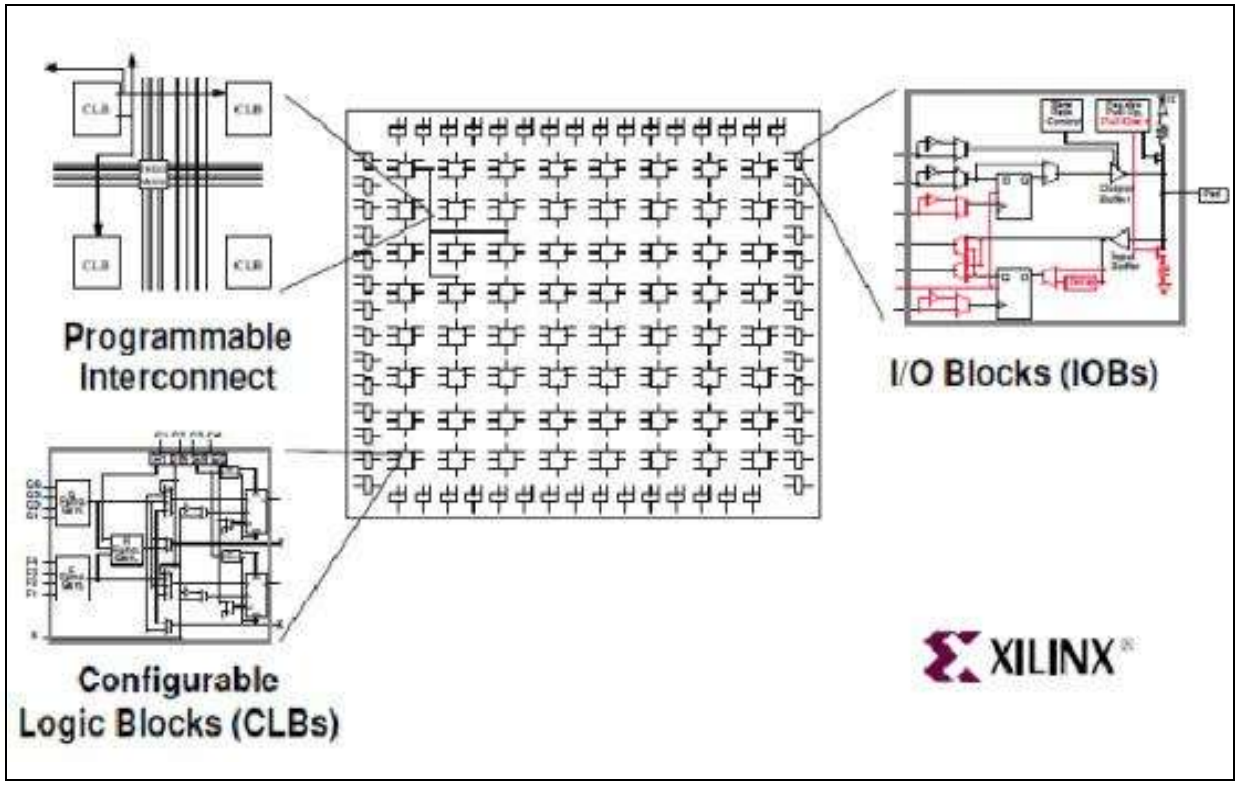
Les points de connexions sont du type ROM, c'est-à-dire que la modification du point est irréversible. Le nom anti-fusible vient du fait que l'état initial du fusible ou la couche isolante est présent et il n'y a pas de contact pour l'établir il faut détruire le fusible ce qui est contradictoire au fonctionnement habituel d'un fusible. Des composants moins génériques mais plus petits et plus rapides ont été développés (Figure. I.4) [7].



Figure(I.4) : *Caractéristiques des technologies ANTI-FUSIBLES.*

I.4.5)-Architecture interne des FPGA

Un FPGA (Field-Programmable GateArray) est un circuit intégré avec une structure adaptablepar l'utilisateur et composée d'un réseau de cellules élémentaires ou d'éléments logiques programmables (CLB) et un ensemble d'entrées/sorties (IOB). Les CLB répartis régulièrement et reliés entre eux grâce à des connections qui forment une matrice de routage programmable pour obtenir un comportement spécialisé du circuit. Les composants logiques programmables sont des circuits composés de nombreuses cellules logiques élémentaires librement assemblables. Celles-ci sont connectées de manière définitive ou réversible par programmation afin de réaliser les ou les fonctions numériques désirées. Puisque tous les éléments sont programmables, le FPGA peut émuler et réaliser n'importe quel circuit a l'unique condition que celui-ci n'épuise pas les ressources logiques et de routage du FPGA. L'architecture interne des FPGA est différente d'un concepteur à un autre et même entre les différentes gammes du même constructeur mais rien n'empêche que leurs ressemblances peuvent être rassemblées dans le schéma représentatif de la figure (1.5).



Figure(I.5) : *architecture interne d'un FPGA Fabriqué XILINX.*

Les macro-cellules sur la périphérie sont appelées : IOB abréviation anglaise de « Input Output block », signifiant bloc logique d'entrées sorties.

L'ensemble des points de connexions est appelé PIP, abréviation anglaise de « Programme InterConnect Points ».

La granularité des FPGA par les macro-cellules CLB nous permet d'implémenter des fonctions logiques « combinatoires ou séquentielles » complexes car chaque CLB est constitué d'une partie combinatoire et d'une partie séquentielle. Chaque fonction est décomposée en petites fonctions booliennes qui peuvent être contenues par de petites cellules élémentaires SLICES. Ces dernières comportent des LUT pour la partie combinatoires et une ou des bascules (généralement de type D) pour la partie séquentielle.

I.4.6)-Caractéristiques de la technologie des FPGA

On peut distinguer cinq caractéristiques principale pour les circuits FPGA :

- Performances
- Temps de mise sur le marché
- Coût
- Fiabilité
- Maintenance à long terme

I.4.6.1)-Performances

Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application [4].

I.4.6.2)-Temps de mise sur le marché

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la Technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, vous pouvez tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé. Par la suite, vous Pourrez apporter les éventuelles modifications nécessaires à votre FPGA, en quelques heures au lieu de quelques semaines [4].

I.4.6.3)-Coût

Les coûts d'ingénierie des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC peut être justifié en cas des demandes dépassant des milliers de circuits par année. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage [4].

I.4.6.4)-Fiabilité

Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche[4].

I.4.6.5)-Maintenance à long terme

Les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la préconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications Éventuellement nécessaires [4].

I.5)- Langage de description VHDL

Le VHDL (Vhsic-VerryHight Speed Integrated CircuitHardware DescriptionLanguage) est l'un des langages de description matérielle le plus utilisé pour la description, La simulation, et la synthèse des circuits électroniques) de grand densité d'intégration et travaillant à haut vitesse. Le VHDL est un langage moderne, puissant et général à cause de sa lisibilité, à sa modularité, à sa sécurité d'emploi et la fiabilité de ses descriptions[8]. Il autorise plusieurs méthodologie de conception (comportemental, flot de données, structurel) tout en étant d'un très haut niveau d'abstraction en électronique (il est indépendant de la technologie utilisée : FPGA, CPLD, ASIC, etc.). Cette abstraction permet d'ailleurs de le simuler sur ordinateur avant de programmer la moindre puce. Historiquement, le développement du langage VHDL a passé par plusieurs phases, parmi lesquelles [1]:

- 1980 : Début du projet, financé par le DoD
- 1982 : Contrat pour Intermetrics, IBM et Texas
- 1985 : Version 7.2 dans le domaine public
- 1987 : Standard IEEE 1076 (VHDL-87)
- 1993 : Nouvelle version du standard (VHDL-93)
- 2001 : Nouvelle version du standard (VHDL-2001)
- 2008 : Nouvelle version du standard (VHDL-2008)

I.5.1)-Structure générale d'une description VHDL

Une description VHDL est composée de 3 parties indissociables à savoir (Fig.I.6)

_ **Bibliothèque IEEE** (Library de déclaration)

_ **L'entité (ENTITY),** elle définit les entrées et sorties.

_ **L'architecture (ARCHITECTURE),** elle contient les instructions VHDL permettent de réaliser le fonctionnement attendu

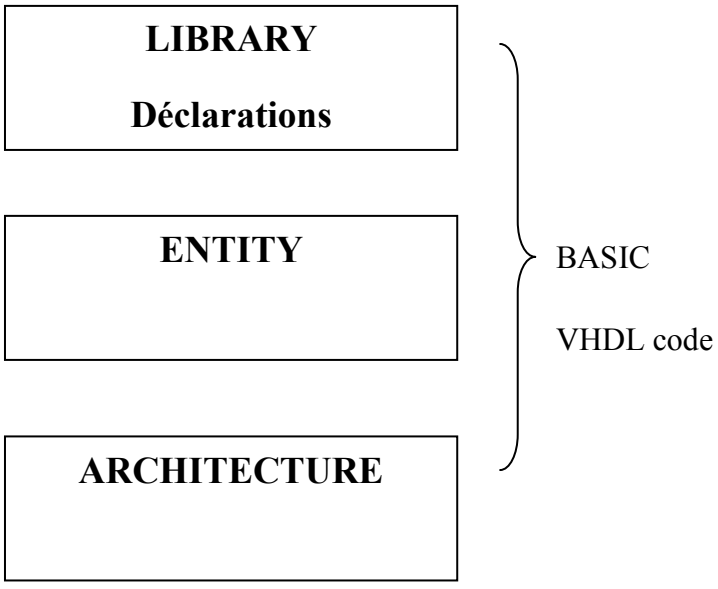


Figure (I.6) : *Différents block de langage VHDL.*

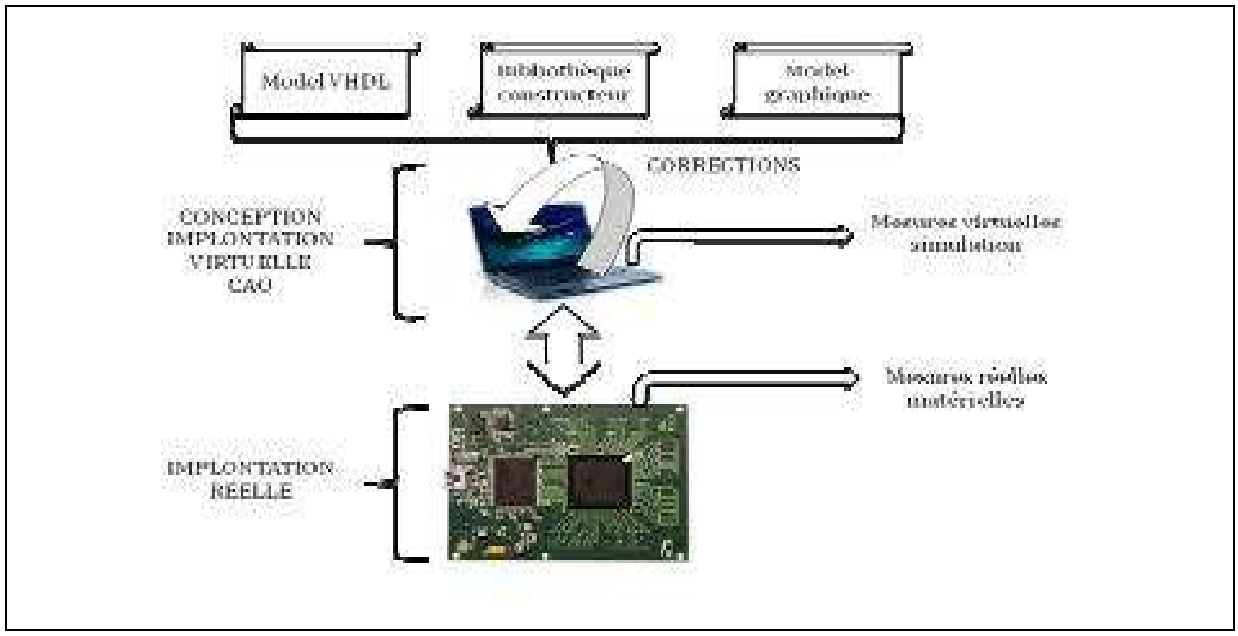
I.6)-Prototypage des FPGA

Il existe différents moyens pour programmer ou configurer un circuit FPGA, parmi lesquels on peut citer :

- Programmation en utilisant la conception HDL.
- Programmation en utilisant l'interface Matlab-Simulink-XSG (Xilinx System Generator).

I.6.1)-La programmation des FPGA par les outils HDL

Avant d'arriver à l'étape de configuration d'un circuit FPGA, le flot de conception d'un design regroupe plusieurs niveaux d'abstraction (FIg.I.7). Dans chaque niveau, le concepteur s'intéresse à la résolution d'un problème. Pour les circuits FPGA, ils existent des outils CAO spécifiques qui sont utilisés pour réaliser les différents niveaux d'abstraction. On trouve l'environnement ISE-XILINX pour les circuits FPGA de la firme XILINX, l'environnement QUARTUS pour les circuits FPGA de la firme ALTERA, etc.[9].



Figure(I.7) : *Mode d'exécution matériel de la CAO.*

Communément, le développement d'une application sur FPGA suit l'enchainement des étapes suivantes :

I.6.1.1)-Spécification du design

- Le nombre de branche d'entrée-sortie et leur localisation dans la puce **FPGA**.
- La spécification de la fréquence d'horloge du système.
- La spécification de la mémoire requise pour l'application.

I.6.1.2)-Conception du design

C'est une étape essentielle à travers laquelle le design ou l'application envisagée est décrite en utilisant un outil de description matériel approprié, nommé langage HDL. En effet, deux langages HDL sont largement utilisés, il s'agit du VHDL et du Verilog. Ces deux langages sont standardisés, portable et compatible avec toutes les types des FPGA. En plus de la description, certain signaux de test sont créés par le concepteur, dans cette étape, pour simuler le comportement de la description.

I.6.1.3)-Synthèse

La synthèse est le processus qui convertit la représentation du design à partir du code HDL fourni pour produire une représentation au niveau porte logique. Elle s'occupe de déterminer quelles sont les structures susceptibles de répondre au cahier des charges étudié et de produire un code booléen unique sous forme d'un fichier.

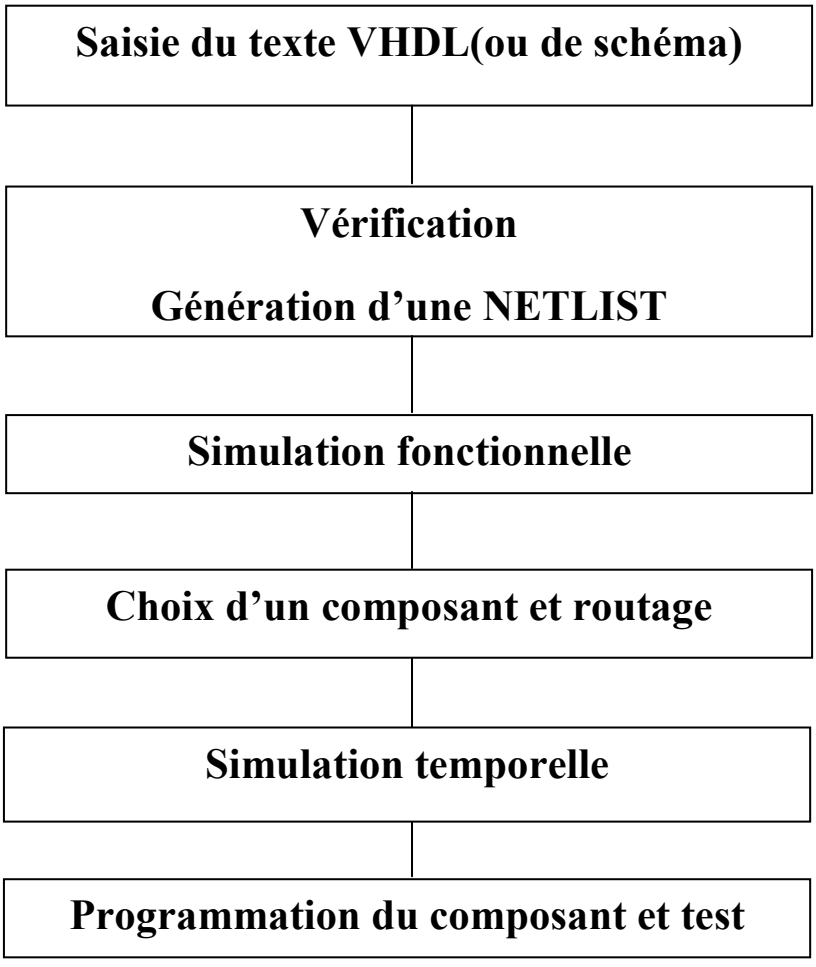
I.6.1.4)-Placement et routage

A partir des fichiers de synthèse.L'outil de conception procède au placement et routage. Un algorithme de routage est sensé de faire l'aiguillage des donnéesqu'il reçoit vers leurs destination par action sur les nœuds de routage ce qui est équivalent à définir les chemins qui relient l'ensemble des **CLB**contenus dans la fonction désirée. Ces algorithmes de routage sont différents d'un concepteur à un autre. Plusieurs traitements sont nécessaires pour obtenir un fichier de configuration. [9]

- Partitionnement
- Placement
- Routage
- Génération des données numériques de configuration

I.6.1.5)-Intégration et implémentation

L'implémentation est la réalisation proprement dite qui consiste à mettre en œuvre l'algorithme sur l'architecture du circuit configurable cible, c'est-à-dire à compiler, charger, puis lancer l'exécution sur un circuit FPGA. C'est une étape de programmation physique et de testes électriques qui clôture la réalisation du circuit, La figure(I.8) résume l'ensemble des étapes précédentes.



Figure(I.8) : *les étapes de développement une application sur FPGA.*

I.6.2)-La programmation des FPGA par XSG/MATLAB-SIMULINK

Malgré la puissance des outils traditionnels de conception matérielle (HDL), pour certains applications, ils sont compliqués à mettre en œuvre et nécessitant des professionnels. D'autre part, les langages matériels ne sont pas très populaires dans le domaine de la commande automatique des systèmes où l'environnement MATLAB/Simulink est le plus utilisé. Heureusement, avec l'incorporation du Toolbox 'Xilinx-System-Generator (XSG)', proposé par Xilinx, dans l'environnement Matlab/Simulink, il est devenu possible de programmer, de simuler et d'implémenter les descriptions matérielles en utilisant l'environnement graphique de Simulink. En plus de cet avantage, le XSG permet d'exécuter ce qu'on appelle 'the Hardware In the Loop (HIL)', le Co-design et la Co-simulation. En effet, ces opportunités sont très appréciées du fait qu'elles permettent de tester virtuellement un algorithme de commande ou une partie d'un système compliqué sont aller directement sur le matériel et évitant, par conséquent, des dégâts pouvant être engendrés[7].

CHAPITRE II

LA COMMANDE SCALAIRE DE LA MAS

II.1)-Introduction

Les machines électriques ou machines tournants sont des dispositifs électriques Qui permettent de générer un mouvement ou une énergie mécanique à partir d'une énergie électrique et inversement. Elles occupent une place prépondérante dans tous les secteurs industriels. Les machines électrique tournantes sont classées en trois catégories qui sont les machines à courant continu, les machines synchrones et les machines asynchrones.

Depuis son invention par NICOLA TESLA, la machines asynchrone est devenue l'actionneur le plus important parmi les machines tournantes de nos jours. Elle est caractérisée par sa simplicité de conception, de fabrication, d'entretien, de robustesse et peu couteuse avec un excellent rendement. Le moteur asynchrone a des avantages assez connus et assez nombreux. Par contre, il est difficile et onéreux d'en faire une variation de sa vitesse lors du fonctionnement. C'est la raison pour laquelle, le moteur à courant continu a resté longtemps le plus utilisé dans les entrainements à vitesse variable. Avec le développement de l'industrie et les techniques de commande numérique en Temps réel, le moteur asynchrone suscite de plus en plus d'intérêts.

Le présent chapitre est consacré au moteur asynchrone triphasé qui est la cible de notre application. Le principe de fonctionnement est d'abord étudié puis les éléments de contrôle et les boucles de régulation. Finalement, nous présenterons le schéma de commande qui sera implémenté sur les FPGA.

II.2)-Constitution et principe de fonctionnement

Les machines asynchrones, sont des machines électriques de vitesse variable, à courant alternatif, sont pratiquement des moteurs triphasés. En termes de constitution, le moteur asynchrone comprend différent éléments (Fig.II.1)[10].

Ce type de machine comportant deux armateurs coaxiales, l'une fixée appelée le stator ou (l'inducteur) et l'autre est mobile appelée rotor, entre les deux armateurs il y a l'entrefer.

Le stator, portant un enroulement triphasé, est alimenté en triphasé par l'intermédiaire de la plaque au bornes de la machine, ce qui permet de l'alimenter en couplage Y ou Δ. Les courants alternatifs dans le stator créent un champ magnétique tournant à la pulsation de synchronisme résultant des fréquences des tensions d'alimentation. Pour le rotor, il porte des barres en cuivre ou en aluminium, ce dernier est appelé (cage d'écureuil ou rotor en court-circuit). Le courant dans ses barres est induit uniquement par le champ statorique, il est plus

utilisé en fonctionnement moteur, tandis que, le rotor bobiné est plus utilisé en fonctionnement générateur[11].

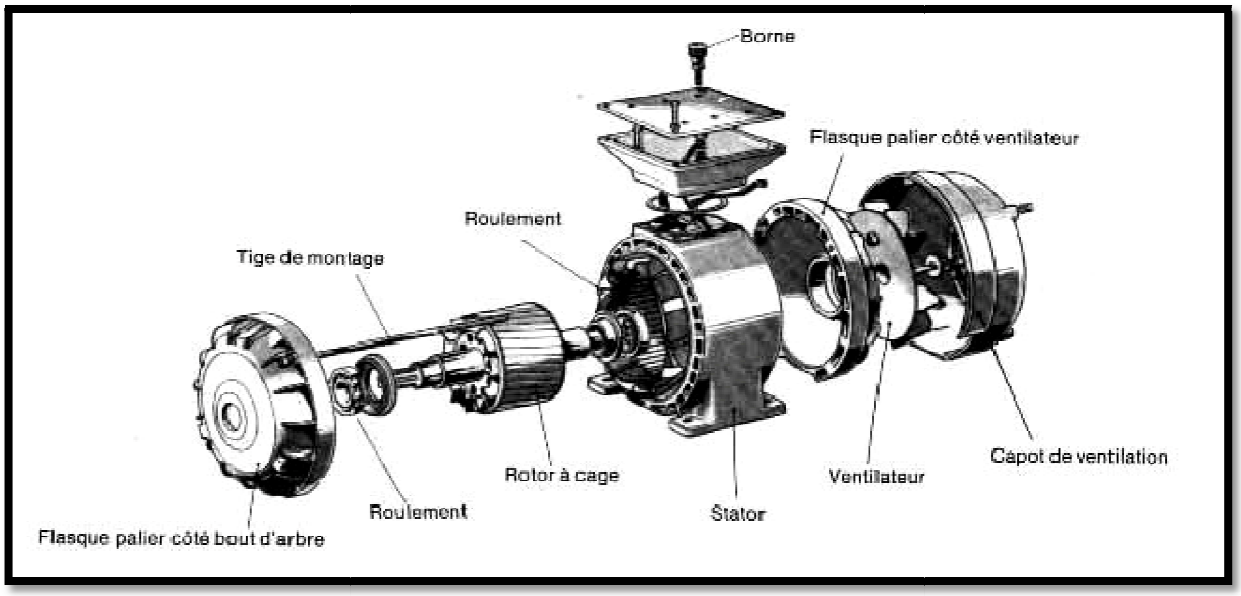


Figure (II .1) : *constitution de moteur asynchrone à cage*

Concernant le principe de fonctionnement, L'enroulement statorique reçoit de l'énergie électrique du réseau électrique, ce qui crée un champ tournant à la vitesse angulaire synchronisme, ce champ en balayant les barres rotoriques induit des F.E.M et donc des courants, ces courants induits qui, en interaction avec champ magnétique, produisent un couple moteur qui entrainera la mise en mouvement du rotor dans les sens de champ tournant statorique [12].

Le moteur possède plusieurs avantages, entre autres, on trouve :

- La robustesse.
- La simplicité de construction.
- Le bas cout.
- Le rendement important.

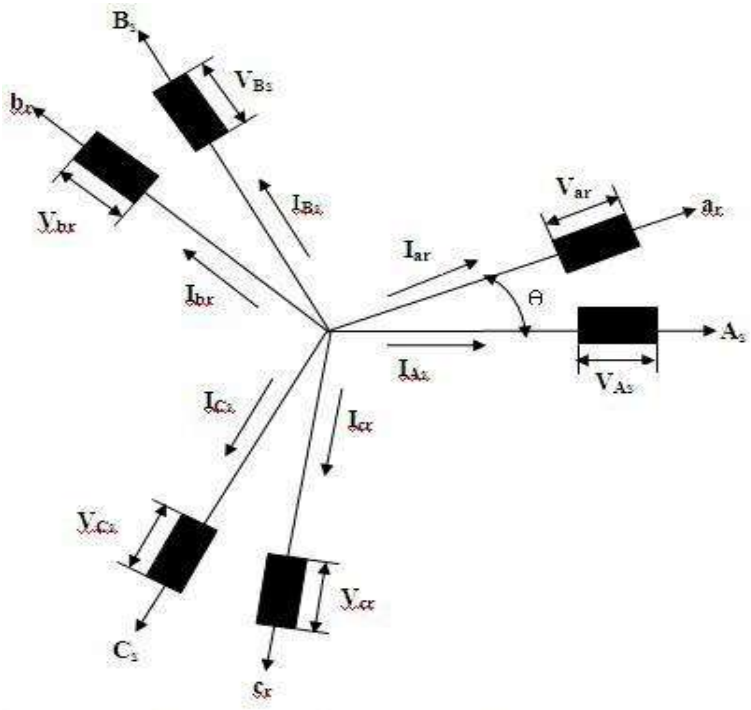
Toutefois et, en dépit de ces avantages, le moteur asynchrone pose un problème avec sa dynamique qui fortement non-linéaire et couplé. A cet effet, la commande de ce moteur a été un dur défit pour ses adaptes. [13]

II.3)-Modélisation de la machine asynchrone

Avant d'entamer le volet modélisation du moteur asynchrone, il est nécessaire d'adopter certaines hypothèses simplificatrices, telles que :

- La saturation du circuit magnétique n'est pas considérée
- L'effet d'hystérésis est négligé
- La construction mécanique est parfaitement équilibrée
- L'entrefer est lisse
- La dispersion du champ magnétique aux deux bouts de la machine est négligeable.

Au niveau de la méthode de modélisation mathématique, nous considérons que le stator est composé de trois bobines identiques (A,B et C) régulièrement déphasées de $2\pi/3$ radians électriques (Figure.II.2) [13].



Figure(II.2) : *Représentation schématique d'une machine asynchrone triphasée.*

La machine asynchrone est définie par trois types d'équations, à savoir :

- Les équations électriques.
- Les équations magnétiques.
- Les équations mécaniques.

II.3.1)-Equations électriques

La loi de faraday permet d'écrire :

$$v = Ri + \frac{d\varphi}{dt} \quad \textbf{(II.01)}$$

Pour les 3 phases statoriques on résume cette écriture par l'écriture matricielle condensée :

$$\begin{bmatrix} V_{sa} \\ V_{sb} \\ V_{sc} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} I_{sa} \\ I_{sb} \\ I_{sc} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \varphi_{sa} \\ \varphi_{sb} \\ \varphi_{sc} \end{bmatrix} \quad \textbf{(II.02)}$$

$$[V_{sabc}] = [R_s][I_{sabc}] + (d/dt)[\varphi_{sabc}] \quad \textbf{(II.03)}$$

Pour les phases rotoriques :

$$\begin{bmatrix} V_{ra} \\ V_{rb} \\ V_{rc} \end{bmatrix} = \begin{bmatrix} R_r & 0 & 0 \\ 0 & R_r & 0 \\ 0 & 0 & R_r \end{bmatrix} \begin{bmatrix} I_{ra} \\ I_{rb} \\ I_{rc} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \varphi_{ra} \\ \varphi_{rb} \\ \varphi_{rc} \end{bmatrix} \quad \textbf{(II.04)}$$

$$[V_{rabc}] = [Rr][I_{rabc}] + (d/dt)[\varphi_{rabc}] \quad \textbf{(II.05)}$$

II.3.2)-Equations magnétiques

Chaque flux comporte une interaction avec les courants de toutes les phases.

Exemple de la phase (a) statorique :

$$\varphi_{sa} = l_s i_{sa} + M_s i_{bs} + M_s i_{cs} + M_1 i_{ra} + M_3 i_{rb} + M_2 i_{rc} \quad \textbf{(II.06)}$$

Exemple de la phase (a) rotorique :

$$\varphi_{ra} = M_1 i_{sa} + M_2 i_{bs} + M_3 i_{cs} + l_r i_{ra} + M_r i_{rb} + M_r i_{rc} \quad \textbf{(II.07)}$$

En matriciel :

$$\begin{bmatrix}\varphi_{sa}\\ \varphi_{sb}\\ \varphi_{sc}\\ \varphi_{ra}\\ \varphi_{rb}\\ \varphi_{rc}\end{bmatrix}=\begin{bmatrix} l_sM_s & M_s & M_1 & M_3 & M_2 \\ M_s & l_s & M_s & M_2M_1 & M_3 \\ M_s & M_s & l_sM_3 & M_2 & M_1 \\ M_1 & M_2 & M_3 & l_rM_r & M_r \\ M_3 & M_1 & M_2 & M_r & l_r & M_r \\ M_2 & M_3 & M_1 & M_r & M_r & l_r \end{bmatrix}\begin{bmatrix}i_{sa}\\ i_{sb}\\ i_{sc}\\ i_{ra}\\ i_{rb}\\ i_{rc}\end{bmatrix} \quad \textbf{(II.08)}$$

Pour l'écriture condensée, on met :

$$[L_{ss}]=\begin{bmatrix} l_s & M_s & M_s \\ M_s & l_s & M_s \\ M_s & M_s & l_s \end{bmatrix} \textbf{(II.09)} \qquad [L_{rr}]=\begin{bmatrix} l_rM_rM_r \\ M_rl_rM_r \\ M_rM_rl_r \end{bmatrix} \textbf{(II.10)}$$

On aura :

$$[M_{sr}]=[M_{rs}]^t=M_{sr}\begin{bmatrix} \cos(\alpha) & \cos\left(\alpha+\frac{2\pi}{3}\right) & \cos(\alpha-\frac{2\pi}{3}) \\ \cos\left(\alpha-\frac{2\pi}{3}\right) & \cos(\alpha) & \cos(\alpha+\frac{2\pi}{3}) \\ \cos\left(\alpha+\frac{2\pi}{3}\right) & \cos\left(\alpha-\frac{2\pi}{3}\right) & \cos(\alpha) \end{bmatrix} \quad \textbf{(II.13)}$$

On aura finalement :

$$[V_{sabc}]=[R_s].[i_{sabc}]+\frac{d}{dt}([L_{ss}.i_{sabc}+M_{sr}.i_{rabc}]) \quad \textbf{(II.14)}$$

$$[V_{rabc}]=[R_r].[i_{rabc}]+\frac{d}{dt}([L_{rr}.i_{rabc}+M_{sr}.i_{sabc}]) \quad \textbf{(II.15)}$$

II.3.3)-Equations mécaniques

Il faut ajouter l'équation de mouvement au système d'équations différentielles pour étudier les phénomènes transitoires électromécaniques avec une vitesse rotorique variable (par exemple le démarrage, le freinage, la variation de la charge, etc…).

$$J.\frac{d\Omega r}{dx}=c_e-c_r-f\Omega_r \quad \textbf{(II.16)}$$

La vitesse électrique du rotor est donnée par l'expression suivante :

$$\omega_r=p\Omega_r \quad \textbf{(II.17)}$$

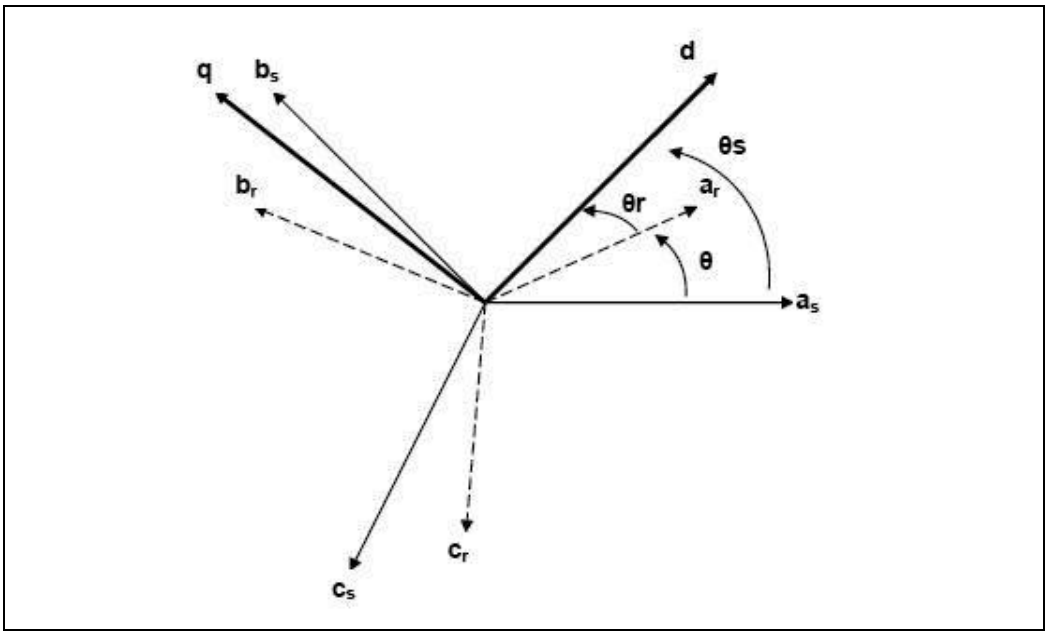
Cette modélisation précédente présente l'inconvénient de la présence des paramètres variables dépendant de l'angle de rotation mécanique (les matrices [Msr] et [Mrs]).

II.4)-Transformation de Park

La transformation de Park consiste à transformer un système triphasé en un système diphasé, elle permet de passer du repère (a,b,c) vers le repère (d,q). Cette transformation est utilisée pour rendre le système (modèle de la machine) plus simple, cette dernière consiste à faire des projections sur deux axes orthogonaux de toutes les grandeurs du système (Fig. II.3). La condition qui permet de remplacer la machine triphasée par son modèle biphasé est sa symétrie. La matrice de Park et son inverse sont les suivantes [11].

$$[P]=\sqrt{\frac{2}{3}}\begin{bmatrix} \cos\theta & \cos(\theta-\frac{2\pi}{3}) & \cos(\theta-\frac{2\pi}{3}) \\ -\sin\theta & \sin\left(\theta-\frac{2\pi}{3}\right) & -\sin(\theta-\frac{4\pi}{3}) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \quad \textbf{(II.18)}$$

$$[P^{-1}]=\sqrt{\frac{2}{3}}\begin{bmatrix} \cos\theta & -\sin\theta & \frac{1}{\sqrt{2}} \\ \cos\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta-\frac{2\pi}{3}\right) & \frac{1}{\sqrt{2}} \\ \cos\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta+\frac{2\pi}{3}\right) & \frac{1}{\sqrt{2}} \end{bmatrix} \quad \textbf{(II.19)}$$



Figure(II.3) : *Représentation du passage d'un système triphasé à celui biphasé.*

En utilisant la transformation de Park, les équations, décrivant la dynamique de la machine, présentés précédemment peuvent êtres réédités comme suivant :

Equations électriques

$$V_{ds}=R_s.i_{ds}+\frac{d\phi_{ds}}{dt} - w_s.\phi_{qs} \quad \textbf{(II.20)}$$

$$V_{qs}=R_s.i_{qs}+\frac{d\phi_{qs}}{dt}+ w_s.\phi_{ds} \quad \textbf{(II.21)}$$

$$V_{dr}=R_r.i_{dr}+\frac{d\phi_{dr}}{dt}-(w_s - w_r) . \phi_{qr}= 0 \quad \textbf{(II.22)}$$

$$V_{qr}=R_r.i_{qr}+\frac{d\phi_{qr}}{dt}+(w_s - w_r) . \phi_{dr}= 0 \quad \textbf{(II.23)}$$

Equations magnétiques

$$\phi_{ds}=L_s.i_{ds} + \mathrm{M} . i_{dr} \quad \textbf{(II.24)}$$

$$\phi_{dr}=L_r.i_{dr} + \mathrm{M} . i_{ds} \quad \textbf{(II.25)}$$

$$\phi_{qs}=L_s.i_{qs} + \mathrm{M} . i_{qr} \quad \textbf{(II.26)}$$

$$\phi_{qr}=L_r.i_{qr} + \mathrm{M} . i_{qs} \quad \textbf{(II.27)}$$

Equations mécaniques

$$\mathrm{J}.\frac{d\Omega_r}{dt} = C_e - C_r - f\Omega_r \quad \textbf{(II.28)}$$

$$C_e = (\mathrm{p.M})\left(i_{rd}.i_{sq} - i_{sd}.i_{rq}\right) \quad \textbf{(II.29)}$$

$$w_r = p\Omega_r \quad \textbf{(II.30)}$$

Avec :

J : Moment d'inertie du rotor.

f : Coefficient de frottement visqueux.

Ce : couple électromagnétique.

Cr : couple résistant. P : Nombre de paires de pôles.

Modèle globale du moteur asynchrone

Suivant la vitesse attribuée au référentiel biphasé, on distingue trois variantes pour le modèle du moteur asynchrones triphasé :

- Modèle lié stator ou référentiel fixe

we = 0

- Modèle lié champ tournant

we = ws

- Modèle lié rotor

we = wr

En considérant le référentiel fixe, le modèle du moteur peut être donné par le système d'équations suivant :

$$\frac{dI_{s\alpha}}{dt} = -\left(\frac{R_s}{\sigma L_s} + \frac{M^2}{\sigma L_s L_r}\right) I_{s\alpha} + \frac{M}{\sigma L_s L_r T_r}\phi_{r\alpha} + \frac{M}{\sigma L_s L_r}\omega_r \phi_{r\beta} + \frac{1}{\sigma L_s} V_{s\alpha}$$

$$\frac{dI_{s\beta}}{dt} = -\left(\frac{R_s}{\sigma L_s} + \frac{M^2}{\sigma L_s L_r}\right) I_{s\beta} + \frac{M}{\sigma L_s L_r T_r}\phi_{r\alpha} + \frac{M}{\sigma L_s L_r}\phi_{r\beta} + \frac{1}{\sigma L_s} V_{s\beta}$$

$$\frac{d\phi_{r\alpha}}{dt} = \frac{M}{T_r} I_{s\alpha} - \frac{1}{T_r}\phi_{r\alpha} - \omega_r \phi_{rq}$$

$$\frac{d\phi_{r\beta}}{dt} = \frac{M}{T_r} I_{s\beta} - \frac{1}{T_r}\phi_{r\beta} - \omega_r \phi_{rq}$$

$$\frac{d\Omega}{dt} = \frac{1}{J}\left(C_{em} - k_f \Omega - C_r\right)$$

Avec :

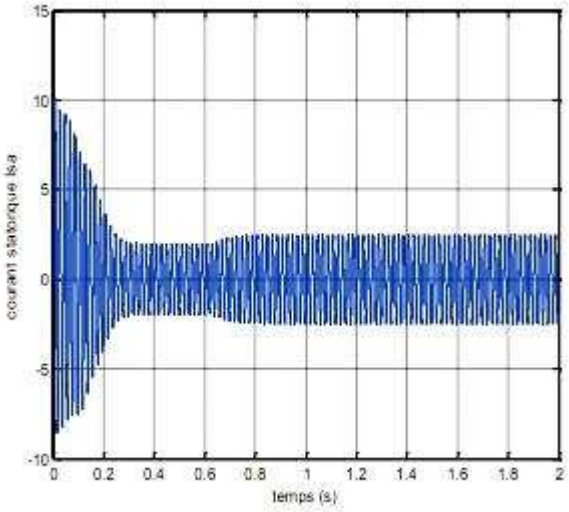
$C_{em} = p\frac{M}{L_r}(\phi_{r\alpha} I_{s\beta} - \phi_{r\beta} I_{s\alpha})$ Et $\sigma = 1 - \frac{M^2}{L_s L_r}$ c'est le coefficient de dispersion total.

$T_r = \frac{L_r}{R_r}$ C'est la constante du temps rotorique.

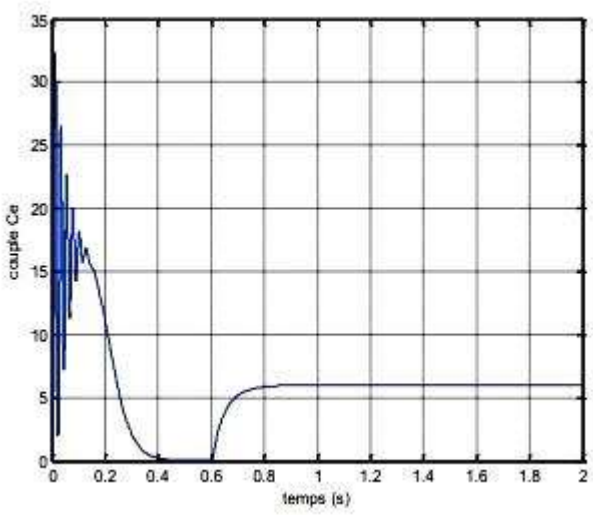
II.5-simulation du modèle moteur asynchrone triphasé

La machine étant alimentée par un système de tensions sinusoïdales, Va, Vb et Vc. Après démarrage, un couple de charge de 6N.m a été appliqué à l'instant 0.6s. Les figures (II.4, II.5, II.6) illustrent les réponses dynamiques du moteur avec les conditions de fonctionnement appliquées.

Pour la vitesse (fig. II.6), en régime transitoire, on remarque un croissement linéaire de vitesse jusqu'à la vitesse nominale. Avec l'application de la charge, la vitesse chute et maintient sa nouvelle valeur vue l'absence de la commande. Pour le couple électromagnétique (fig. II.5), dans l'intervalle de temps 0 à 0.6 on observe une allure caractéristique type de tous les moteurs asynchrones à cage. En charge, le moteur développe un couple équivalent au couple de charge appliqué. Pour le courant (II.4), après un régime transitoire caractérisé par des fortes oscillations, le courant prends des formes sinusoïdales [10].



Figure(II.04) : *Courant statorique Isa.*



Figure(II.05) : *Couple électromagnétique*

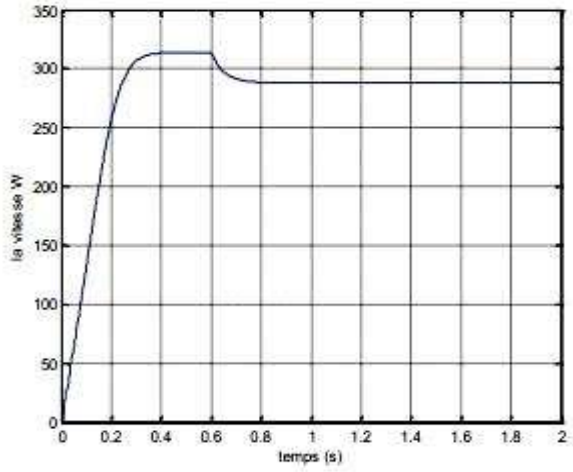


Figure (II.06) : *La vitesse de rotation.*

II.6)-Commande scalaire du moteur asynchrone

Plusieurs commandes scalaires existent selon que l'on agit sur le courant ou sur la tension. Elles dépendent surtout de la topologie de l'actionneur utilisé (onduleur de tension ou de courant). L'onduleur de tension étant maintenant le plus utilisé en petit moyenne puissance, c'est la commande en V/f (V sur f) constant qui est la plus utilisée. On distingue deux types de commande scalaire : [14,10]

- La commande scalaire indirecte où le flux magnétique est contrôlé en imposant le rapport amplitude /fréquence.

- La commande scalaire directe ou le flux magnétique est contrôlé à partir de son estimation.

En boucle ouverte, la vitesse est variée directement en agissant directement sur l'onduleur tout en maintenant un rapport fixe entre la tension et la fréquence pour maintenir constante l'amplitude du flux du moteur.

En boucle fermée, l'erreur de poursuite de la vitesse permet, par l'intermédiaire d'un régulateur PI, de générer la pulsation de glissement qui est de son tour ajoutée à la pulsation rotorique permet d'avoir la pulsation (fréquence) des tensions statoriques (Fig. II.7). La pulsation générée est utilisé à la fois pour calculer l'amplitude des tensions statoriques en utilisant la loi (u/f) constante ainsi que pour le commande de la fréquence de l'onduleur [15].

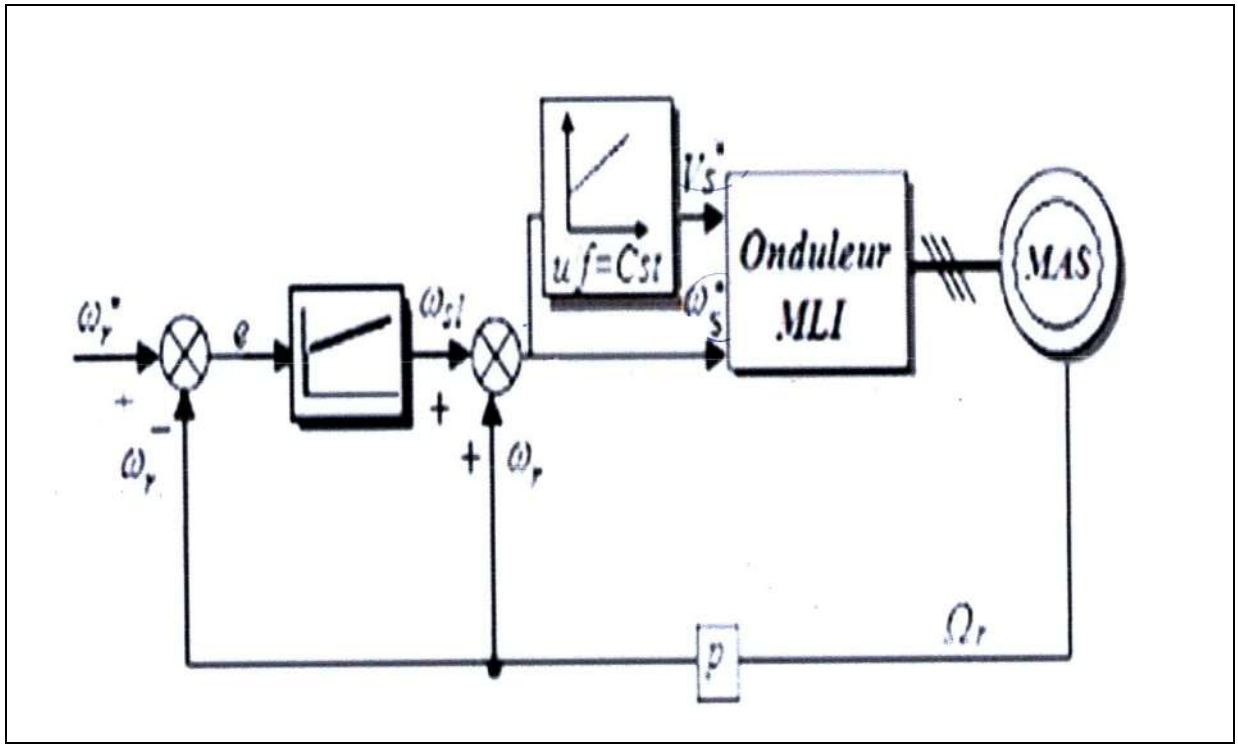


Figure (II.7) : *schéma générale de la commande scalaire en boucle fermée.*

La commande scalaire permet d'obtenir des résultats satisfaisants pour les régimes permanents de la vitesse. Toutefois, pour les régimes transitoires, les performances se dégradent où le flux oscille fortement avec des grandes amplitudes et son module est variables. Ceci explique pourquoi les relations utilisées pour calculer cette commande ne sont valables que dans le régime permanant. Ces oscillations vont se répercuter sur a qualité du couple et la vitesse. A cet effet, ce type de commande n'est donc employé que pour des applications où la variation de vitesse n'est pas grande telle en pompage ou en ventilation. Pour de meilleures performances, il faut faire recours à d'autres commandes plus performantes telles que la commande vectorielle [3].

II.7) Résultat de simulation de la commande scalaire

Les résultats de simulation de la commande sont présentés sur les figures (II.8, II.9, II.10, II.11, II.12 et II.13).La vitesse a un temps de réponse très rapide de 0.22s, le couple électromagnétique suit bien le comportement de la vitesse.

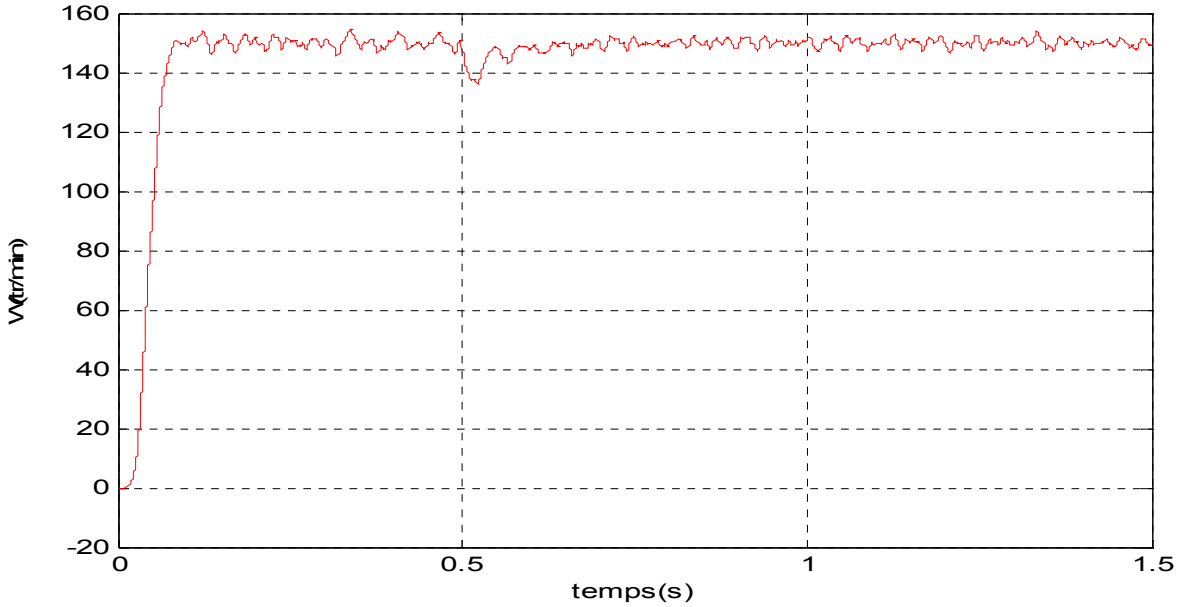


Figure (II.8) : *La vitesse de la commande scalaire.*

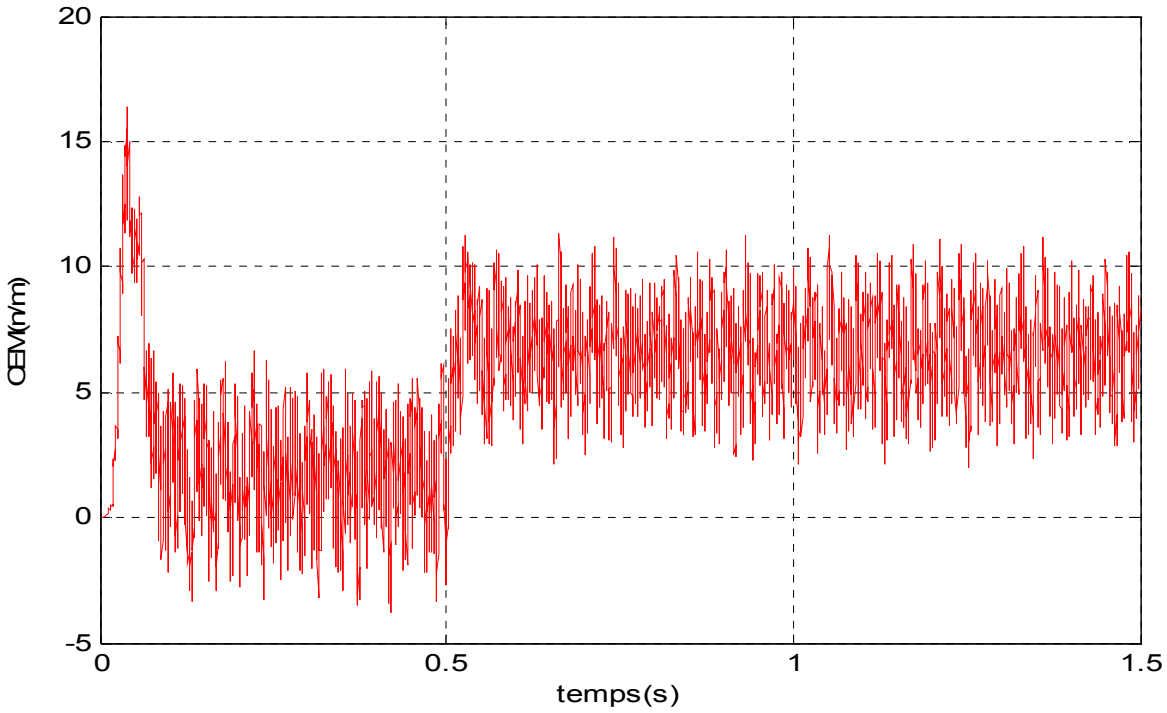


Figure (II.9) : *le couple électromagnétique de la commande scalaire.*

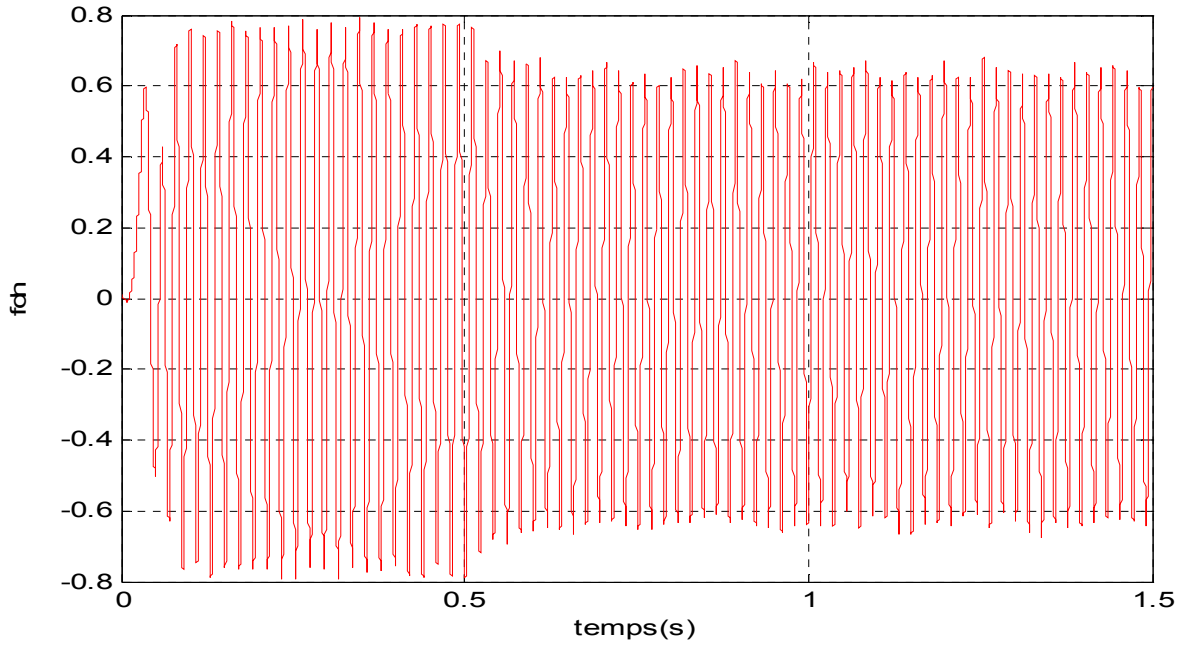


Figure (II.10) : *le flux FDN.*

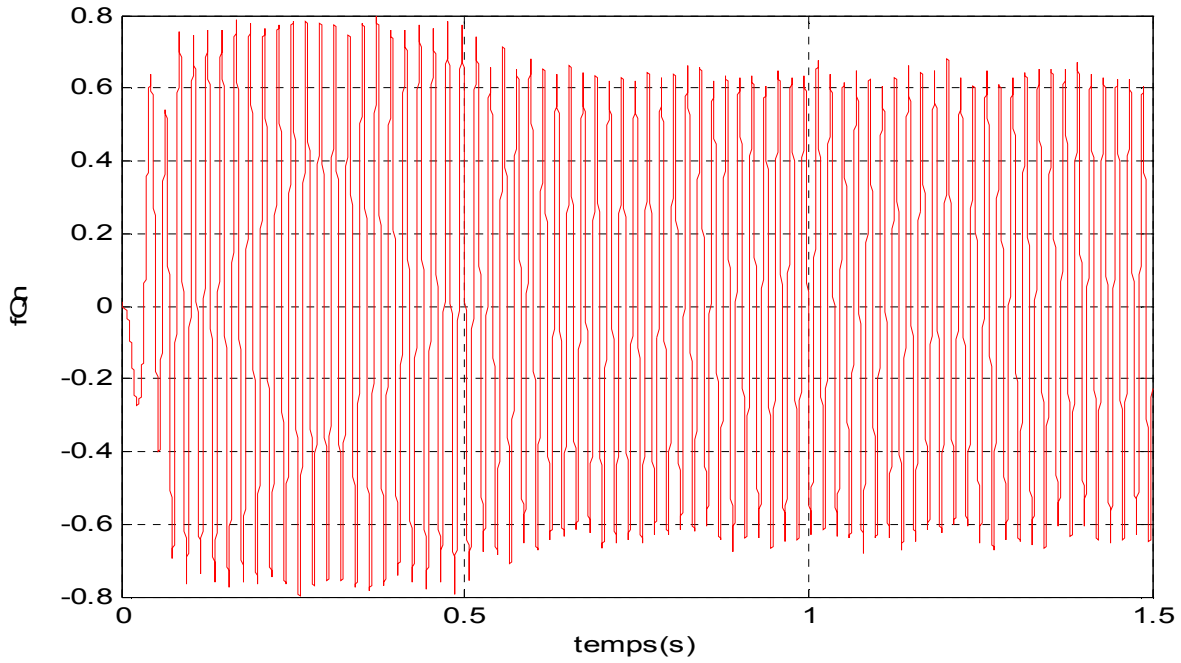


Figure (II.11) : *le flux FQN.*

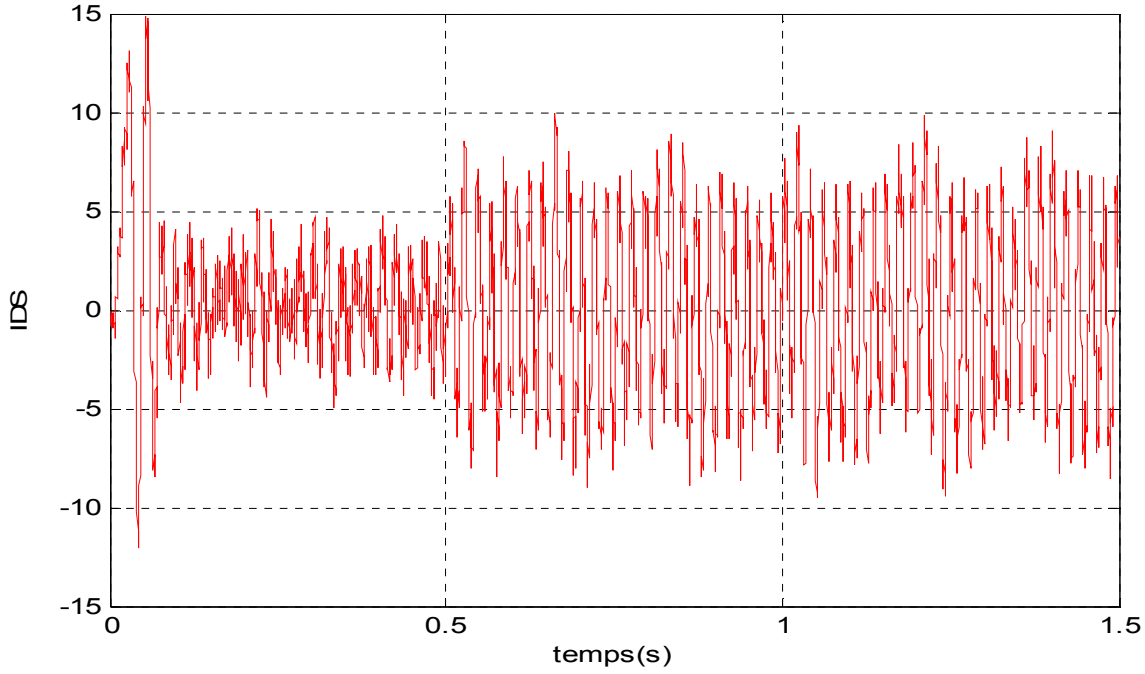


Figure (II.12) : *le courant Ids de la commande scalaire.*

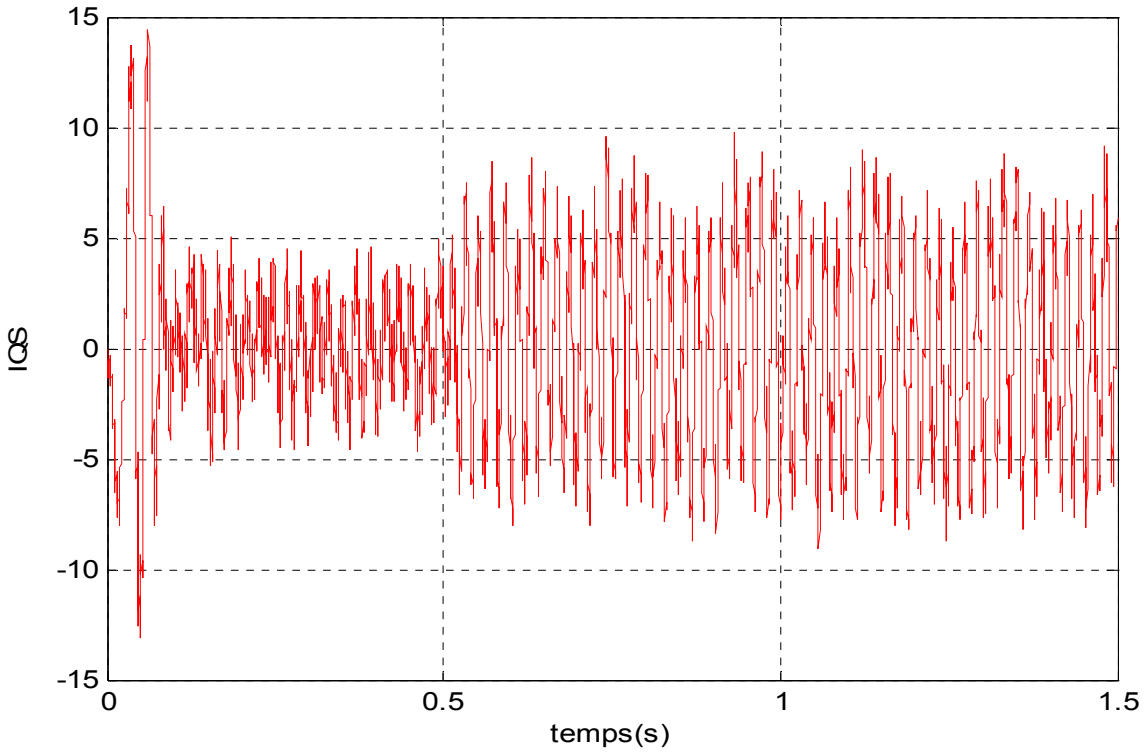


Figure (II.13) : *le courant Iqs de la commande scalaire.*

II.8) Conclusion

Dans ce chapitre on a présenté la modélisation du moteur asynchrone alimenté par une source de tension sinusoïdal triphasée et équilibrée à fréquence constante établi sous des hypothèses simplificatrices. À la suite on a utilisé la transformation du Park pour rendre les équations différentielles moins complexes et à la fin pour commander le moteur asynchrone, la commande scalaire a été appliquée. En effet cette commande fera l'objet du chapitre suivant où un circuit FPGA sera utilisé pour l'implémenter.

CHAPITRE III

IMPLEMENTATION DE LA COMMNADE SCALAIRE D'UN MAS

III.1)-Introduction

Dans de nombreuses applications, le mode de fonctionnement séquentiel limite l'utilisation des microprocesseurs ou les microcontrôleurs pour mettre en œuvre les contrôleurs en raison du nombre de cycles nécessaires pour l'exécution des instructions du programme. Par conséquent, au cours des dernières décennies, le traitement parallèle et le nombre élevé de portes ont conduit à une augmentation rapide de la popularité des circuits reconfigurables, en particulier les circuits FPGA, dans les systèmes embarqué en vue de substituer les moyens à base de microprocesseurs (microcontrôleurs, DSP)[6].

Pour profiter des avantages des circuits FPGA, le présent chapitre est consacré à l'implémentation de la commande scalaire en boucle fermé. L'implémentation envisage principalement la réalisation d'une interface d'acquisition pour calculer la vitesse du moteur à partir des signaux prévenant de l'encodeur incrémental. En plus et, en se basant sur la vitesse obtenue, l'algorithme de la commande scalaire est élaboré en utilisant le langage VHDL afin de l'implémenter sur un circuit FPGA de type SPARTAN-3E.

III.2)- Elaboration de la description hardware du schéma de la commande scalaire

Comme mentionné sur la figure (3.1), la réalisation de la commande scalaire passe par les étapes suivantes :

- Réalisation de l'interface d'acquisition de la vitesse du moteur
- Réalisation du régulateur PI pour la régulation de la vitesse.
- Réalisation de la commande MLI pour l'onduleur :
  - Déterminer les tensions de référence
  - Déterminer le signal triangulaire pour générer les impulsions MLI
  - Génération des impulsions MLI

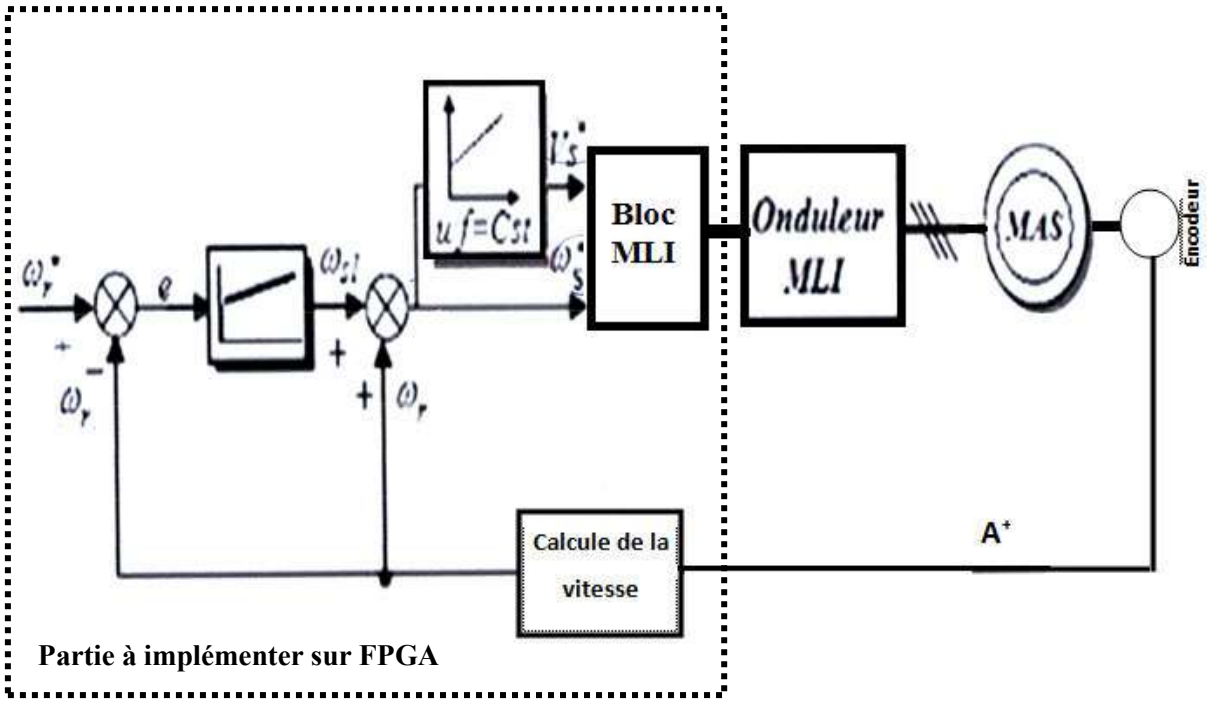


Figure (III.1) : Schéma de commande à implémenter sur carte FPGA

III.2.1)- Interface d'acquisition de la vitesse du moteur

L'acquisition de la valeur de la vitesse représente une étape primordiale pour accomplir le schéma de commande du moteur. Dans notre projet, on a élaboré une interface pour calculer la valeur de la vitesse à partir des signaux prévenant de l'encodeur incrémental

III.2.1.1)-Principe de fonctionnement général d'un codeur

Les codeurs sont liés mécaniquement à un arbre qui l'entraine. Le codeur comprend un disque qui est solidaire de cet arbre moteur (Fig. 3.2). Il comporte une succession de zones opaques et transparentes. Une lumière émise par des DEL traverse les fentes du disque, créant sur des diodes photosensibles (ou des phototransistors) un signal analogique.

Ce signal est converti en un signal numérique ou logique (selon le type de codeur) qui est transmis à l'unité de traitement de l'information [7].

Le codeur incrémental, de conception plus simple, et donc moins onéreux, génère un code Gray sur deux bits.

Les modèles un peu plus sophistiqués, optiques par exemple, peuvent présenter une info supplémentaire : le « top zéro ».Il s'agit d'une impulsion qui est générée une seule fois par tour et qui détermine une position de référence permettant une réinitialisation à chaque tour [8].

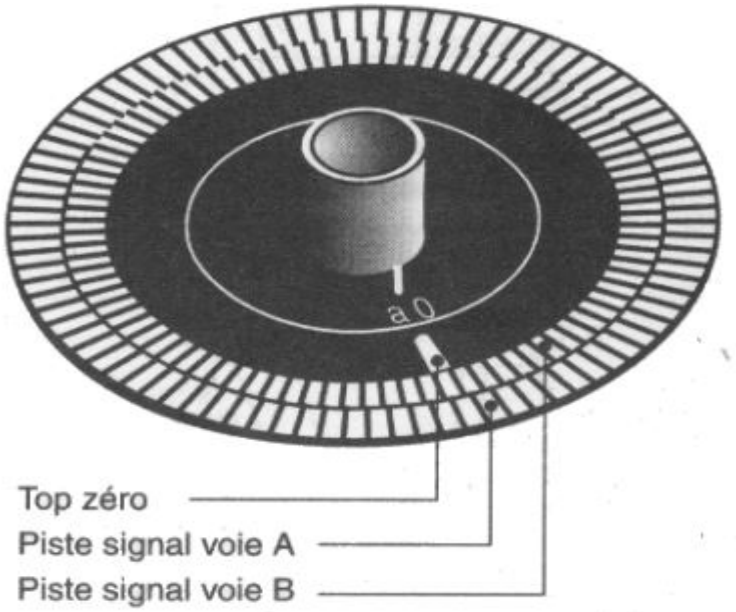


Figure (III.2) : Schéma de base d'un *codeur Incrémentale*

Le disque rotatif comporte au maximum 3 pistes. Une ou deux pistes extérieures divisées en (n) intervalles d'angles égaux alternativement opaques et transparents. Pour un tour complet du codeur, le faisceau lumineux est interrompu (n) fois et délivre (n) signaux carrés (A et B) en quadrature (Fig. 3.3). Le nombre n détermine ce qu'on l'appelle la résolution du codeur.

Le déphasage de 90° électrique des signaux A et B permet de déterminer le sens de rotation:

- dans un sens pendant le front montant du signal A, le signal B est à zéro.
- dans l'autre sens pendant le front montant du signal A, le signal B est à un.

La piste intérieure (Z: top zéro) comporte une fenêtre transparente et délivre un seul signal par tour. Ce signal Z d'une durée de 90° électrique, détermine une position de référence et permet la réinitialisation à chaque tour [7].

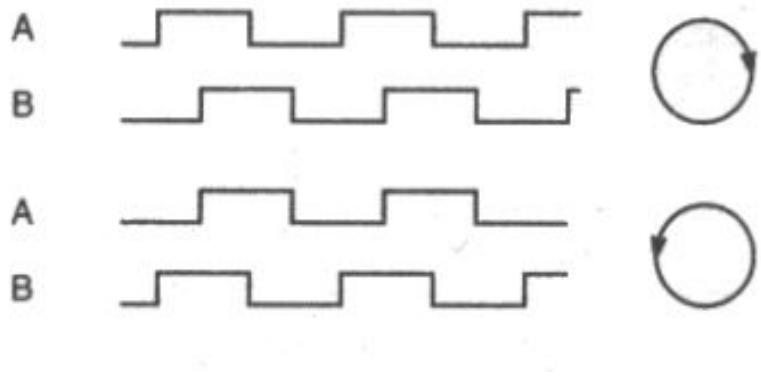


Figure (III.3) : Position dessignaux(A, B) et le sens de rotation

III.2.1.2)- Description de l'interface élaborée

Par principe l'interface proposée exploite les signaux délivrés par le codeur incrémental pour reconstituer la valeur de la vitesse. Pour notre cas, on a utilisé un codeur d'une résolution de n=4096 points. Pour l'interface, on a exploité le signal A^+ du codeur.

En considérant le nombre de points du codeur, on peut calculer l'angle parcourue par le moteur pour chaque période du signal A^+. En notant cette angle par $\Delta\theta$, elle peut être donnée par :

$$\Delta\theta = \frac{2 * \pi}{4096}$$

Pour calculer la vitesse, reste à détermine la période du signal A^+. En notant cette période Δt, la vitesse de rotation du moteur peut être donnée par :

$$\Omega_r = \frac{\Delta\theta}{\Delta t}$$

Etant donné que l'angle $\Delta\theta$ est fixe et bien connue, il reste à calculer le temps Δt afin de déterminer la vitesse.

Dans la description VHDL, le temps est calculé en utilisant un compteur qui compte le nombre de cycles d'horloge d'une demi-période du signal A^+. Alors, le compteur compte le temps dès que le signal $(A^+ = 1)$ jusqu'à la valeur zéro. Dans ce cas, durant la première

demi-période on calcule le temps, tandis que le reste du travail sera effectuer durant la deuxième demi période.

Le temps Δt peut être évalué en termes de cycles d'horloge comme suivant :

$$\Delta t = N.Per_{ck}$$

Où (N) c'est le nombre donné par le compteur et Per_{ck} c'est la période de du signal d'horloge de la carte FPGA utilisé. La carte utilisée est du type Spartan-3^E ayant une horloge de 50 MHz donc $Per_{ck} = 20.10^{-9}\ s$.

La vitesse du moteur peut être évaluée maintenant comme suivant :

$$\Omega_r = \frac{\Delta\theta}{2.\Delta t} = \frac{\pi}{2 * 4096} * \frac{1}{N * 20 * 10^{-9}}$$

$$= \frac{38349}{N}$$

Finalement, la vitesse est calculée en divisant le nombre (38349) par la valeur (N) donnée par le compteur utilisé pour le comptage des cycles d'horloge. A ce stage il faut noter qu'en VHDL on e trouve pas une routine prédéfinie pour achever l'opération de la division qui est l'une des opérations les plus compliquées et les plus couteuses en temps et en ressources dans les descriptions hardwares. Pour ce faire, on a opté pour l'un des algorithmes disponibles dans la littérature qui est le non-restoring algorithme.

L'interface réalisée pour l'acquisition de la vitesse comporte les éléments suivant :

- Un compteur pour calculer le temps d'une demi-période du signal du codeur
- Un diviseur pour achever l'opération de la division
- Un bloc pour déterminer la valeur finale de la vitesse

La figure (III.4) illustre un exemple de simulation utilisé pour tester l'interface d'acquisition élaborée.

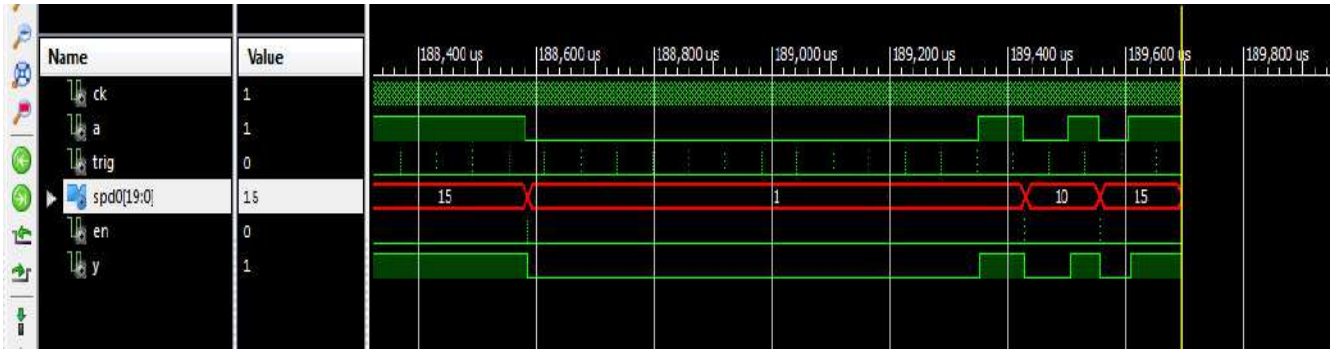


Figure (III.4) : *la simulation de la vitesse.*

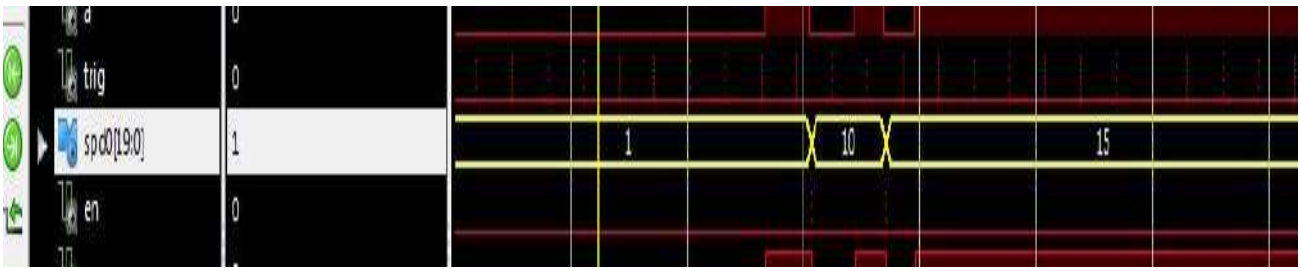


Figure (III.5) : *la simulation de vitesse en zoom.*

III.2.2)-Réalisation VHDL du régulateur PI.

En se basant sur la valeur de la vitesse du moteur calculée précédemment, la description hardware de la régulation est effectuée selon les étapes suivantes :

- Calcule de l'erreur de poursuite entre la vitesse calculée et la valeur de référence choisie
- Etablir les actions proportionnelle et intégrale du régulateur. A noter que pour l'intégrale on a utilisé la méthode d'Euler qui peut se traduire en un simple accumulateur en VHDL.
- A partir des deux actions du régulateur, la vitesse de glissement est calculée selon le schéma de la commande scalaire établis en deuxième chapitre.

La figure (III.6) donne une partie code VHDL utilisé pour réaliser la boucle du réglage.

```
148  begin
149
150     if rising_edge(ck)then
151
152       er <= wrf - (spd0(9 downto 0) & "0000000000");
153
154       ui1 <= ui1 + er  ;
155       ui2 <= ki*ui1 ;
156       ui<= "0000" & ui2(29 downto 14);
157
158       up <= ui - (spd0(9 downto 0) & "0000000000");
159       wgg <= up*kp;
160
161
162        end if;
163  end process;
```

Figure (III.6) : *Le programme d'un PI.*

III.2.3)-Réalisation de la commande MLI pour l'onduleur

Le principe de la stratégie MLI est de commander les bras de l'onduleur par une décision livrée par un algorithme au début de chaque période d'échantillonnage. Le principe se repose sur une simple comparaison, pour chaque bras, entre un signal de référence sinusoïdal (la modulante) et un signal triangulaire de fréquence plus élevée (la porteuse), on appel : MLI sinusoïdal-triangulaire (Fig.III.7). La fréquence de porteuse définie la fréquence de découpage et les points d'intersection entre la modulante et la porteuse correspondant aux instants de commutations de l'onduleur.

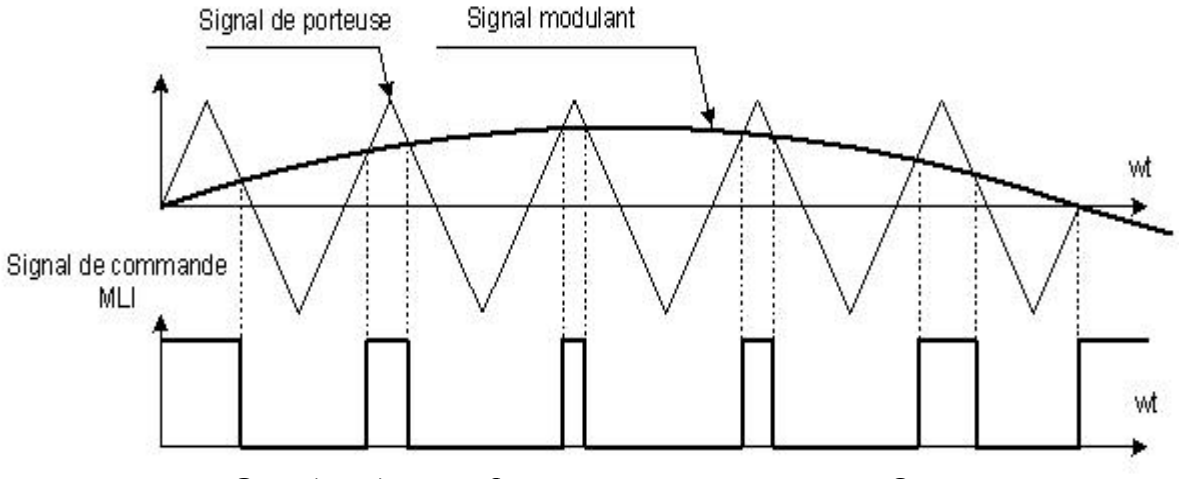


Figure (III.7) : *Principe de la MLI sinusoïdaux-triangulaire.*

Pour détermines les tensions de référence sinusoïdales il faut :

- Déterminer la fréquence et les amplitudes des tensions
- Calculer la fonction sinusoïdale pour chaque tension.

Pour l'amplitude et la fréquence et, suivant le principe de la commande scalaire, ces deux quantités sont calculées directement en utilisant la vitesse de glissement obtenue via le régulateur PI et la vitesse du moteur donnée par l'interface d'acquisition. La somme de ces deux quantités nous donne la pulsation des tensions, et par la suite, l'amplitude des tensions de référence.

Pour générer le signal sinusoïdal, dans notre projet, on a utilisé des RAM contenant des valeurs discrétisée de la sinusoïde. La réalisation d'un signal sinusoïdal est obtenu on calculant l'angle et par la suite lire la valeur du sinus correspondant à partir de la RAM (Fig. III.8 et III.9). En effet, l'angle est obtenu en intégrant la pulsation des tensions. Pour économiser les ressources de la carte FPGA on a utilisé une seule ram pour générer les trois signaux (Va, Vb, Vc). Pour différencier les trois signaux sinusoïdaux, on a joué sur l'ordre de lecture à partir de la ram en faisant des décalages correspondant aux décalages naturels entre les trois phases.

```
83  if      tt_int<11 then
84          sina<= to_signed (snx(0), 13);
85          sinb<= to_signed (snx(100), 13);
86          sinc<= to_signed (snx(200), 13);
87  elsif tt_int>=11 and tt_int<21 then
88          sina<= to_signed (snx(1), 13);
89          sinb<= to_signed (snx(101), 13);
90          sinc<= to_signed (snx(201), 13);
91  elsif tt_int>=21 and tt_int<32 then
92          sina<= to_signed (snx(2), 13);
93          sinb<= to_signed (snx(102), 13);
94          sinc<= to_signed (snx(202), 13);
95  elsif tt_int>=32 and tt_int<43 then
```

Figure (III.8) : *La lecture des points 0-π.*

```
210  if teta1<= 3217 then
211        teta<= unsigned(teta1) ;
212        sinaa <= sina;
213  else
214        teta <= unsigned(teta1 - 3217);
215        sinaa <= 0 - sina;
216
217  end if;
218
219  end if;
220  end process;
```

Figure (III.9) : *La lecture des points π-2π.*

Les dents du porteuse (signal triangulaire) s'appellent dents de scie, on veut avoir un nombre de dents prédéfinie N dans chaque période de modulante. La période de N dents et la période de modulante doit être égaux, mais avec une amplitude supérieure à l'amplitude de signal de référence (pour notre cas on a pris un rapport de 0.8). La réalisation du signal triangulaire est effectuée en utilisant un compteur Up-Down (Fig. III.10).

```
262   if rising_edge(ck) then
263   --compteur d'un signal triangulaire
264
265   if sg <= 2500 or (sg > 7500) then
266         sg<=sg + 1;
267         sgna <= '1';
268
269   elsIF (sg > 2500) and (sg <= 7500) then
270         sg<=sg + 1;
271         sgna <= '0';
272   else
273         sg <= "000000000000000";
274
275   end if;
276
277   If sgna = '1' then
278      tr <= tr + 1;
279   else
280      tr <= tr - 1;
281   end if;
```

Figure (III.10) : *Le signal triangulaire.*

III.3)- Résultats de simulation VHDL sur PC

Pour valider la description VHDL élaborée pour la commande scalaire, une simulation a été effectuée en appliquant différentes forme pour le signal du codeur. Les résultats de simulation obtenus (Fig. III.11) prouvent le fonctionnement correct pour chaque partie de l'algorithme que soit pour l'interface d'acquisition de la vitesse, soit pour la génération des signaux sinusoïdaux ou que pour la commande globale. Cet effet nous permet donc de passer à la phase de la validation expérimentale.

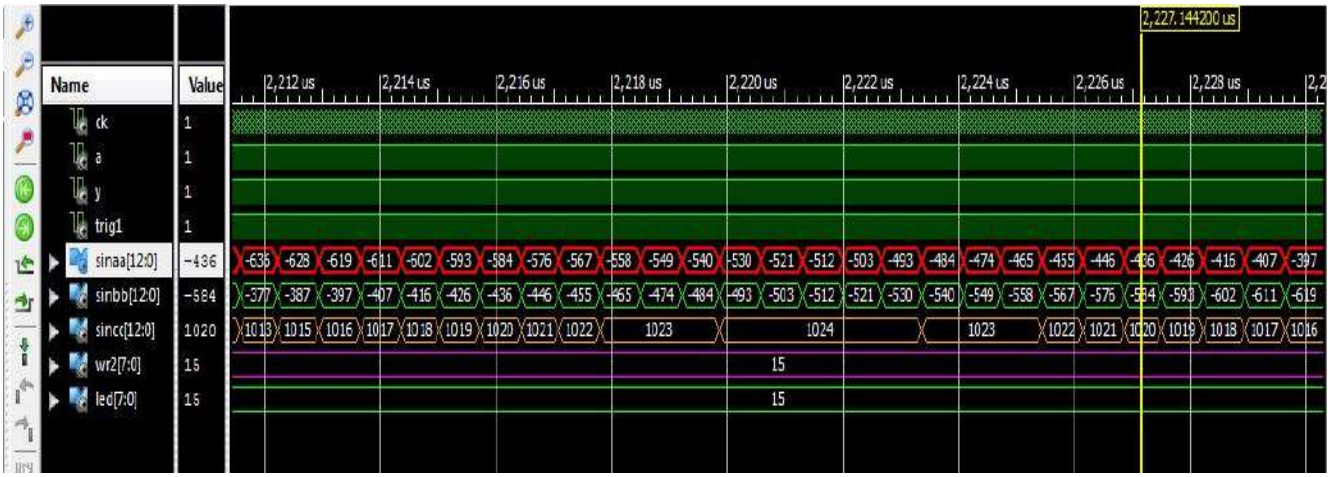


Figure (III.11) : *les 3 signaux pour une vitesse de 15 rad/s*

III.3.1)- Résultats expérimentaux

Pour valider expérimentalement la technique de commande présentée dans le cadre de ce travaille, nous construirons un banc d'essais complet, ce dernier est constitué de (Fig. III.12) :

- Une carte FPGA de type Spartan-3E
- Un moteur asynchrone triphasé mené d'un codeur incrémental de 4096 points
- Une alimentation pour le moteur
- Une alimentation stabilisée pour l'encodeur
- Et de outils de mesure et de visualisation

Le premier test concerne l'acquisition du signal prévenant du codeur par le circuit FPGA, la figure (Fig. III.13) illustre la forme des signaux à la sortie du codeur et son image acquis pour le circuit FPGA. Les signaux sont identiques avec un certain décalage dû à un filtre utilisé à l'entrée de l'interface d'acquisition pour filtrer le signal acquis.

Pour le deuxième test, il concerne le calcul de la vitesse du moteur et les tensions de référence (la reconstitution des trois sinus). Les figures (III.14 et III.15) montrent les résultats obtenus.

Pour la vitesse (Fig. III.15), on peut constater certaines fluctuations qui sont due aux déformations dans le signal acquis du codeur. Mais en moyenne, la vitesse obtenue correspond à celle du moteur.

Pour les formes sinusoïdales (Fig. III.16), ils se basent l'angle obtenu par l'algorithme de la commande scalaire. On peut voire clairement la forme sinusoïdale des tensions de référence obtenu par la description élaborée.



Figure.III.12: Banc d'expérimentation.

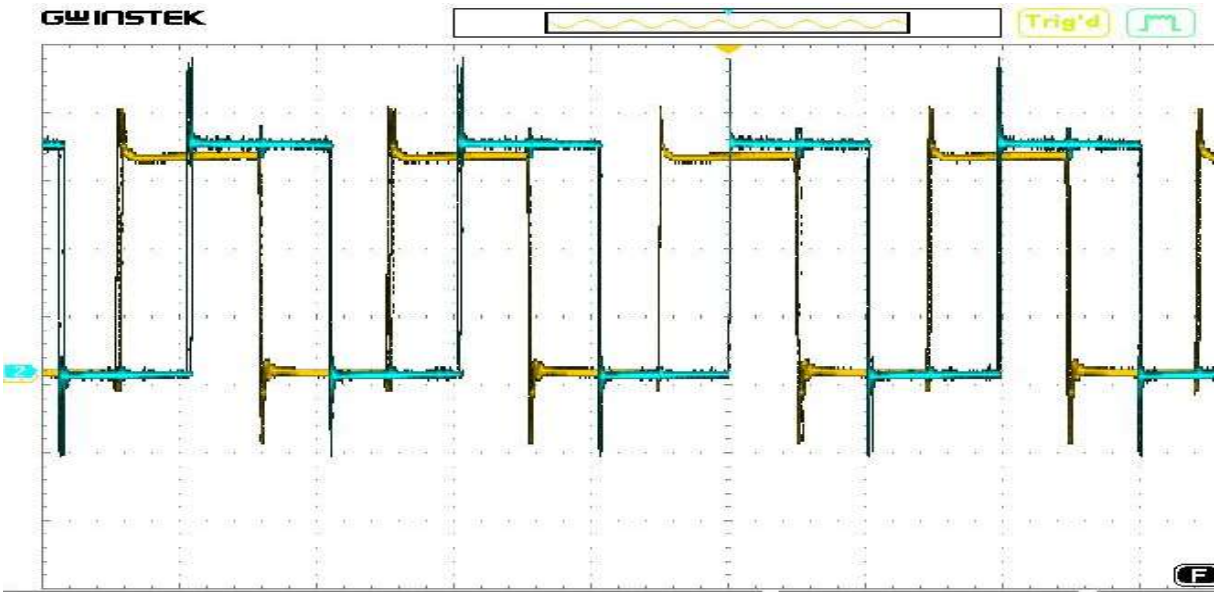


Figure (III.13) : *le signal A.*

- En jaune c'est le signal obtenu directement du codeur
- En bleu c'est le signal acquis par la carte FPGA.

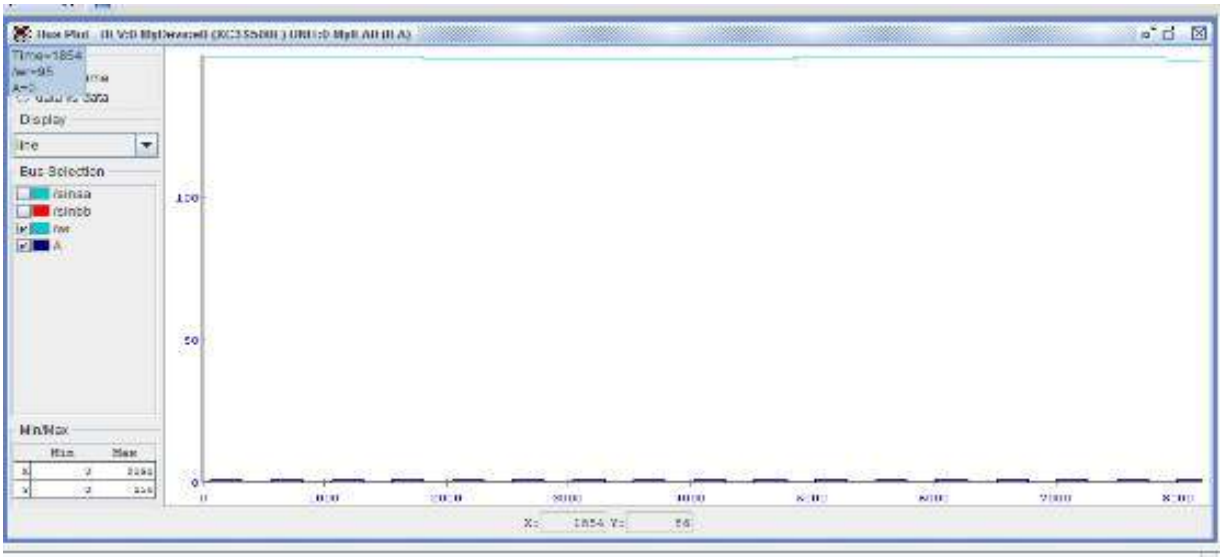


Figure (III.14) :*la vitesse Wr.*

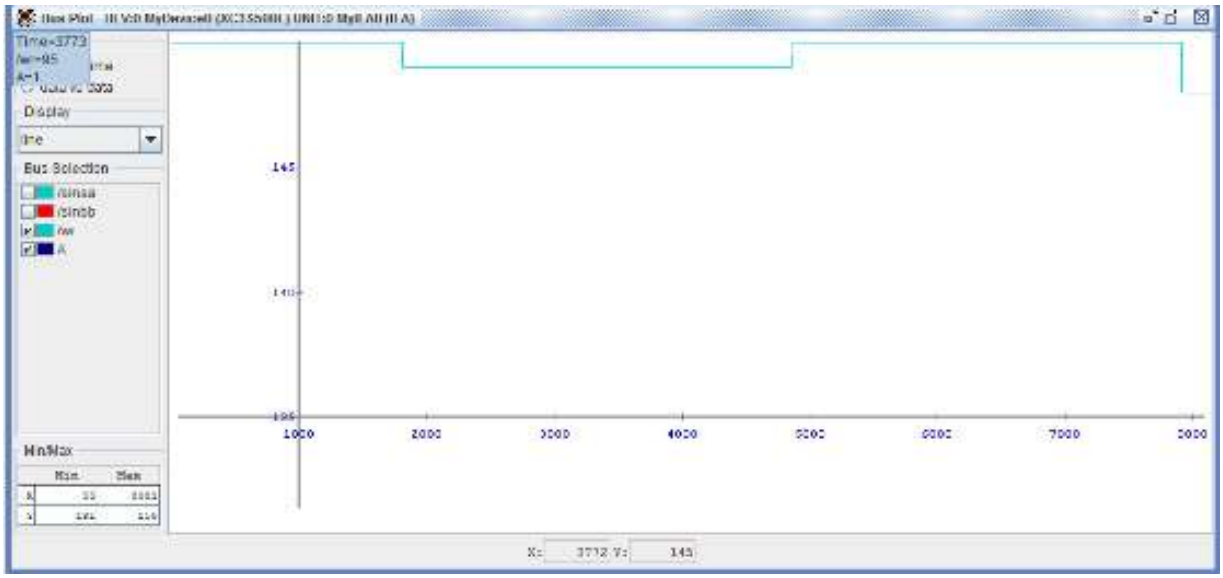


Figure (III.15) :*la vitesse Wr avec zoom.*

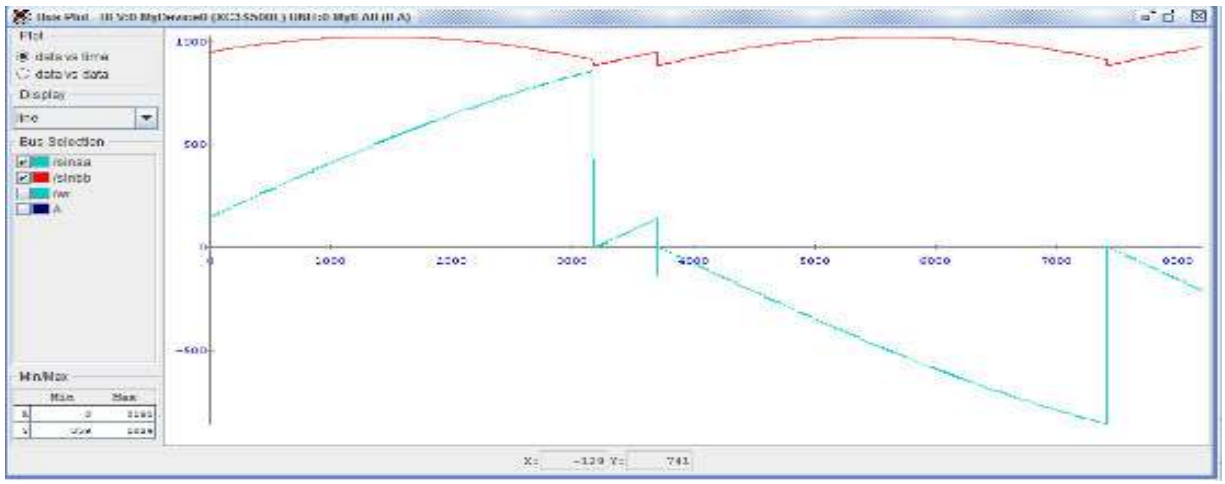


Figure (III.16) :*les signaux sinusoïdaux sina et sinb.*

III.4)- Conclusion

Ce dernier chapitre a été consacré la description hardware de l'algorithme de la commande salaire. Les différentes parties de cet algorithme ont été décrites, synthétisée et implémentée en utilisant un circuit FPGA. En commençant par l'interface du décodeur, passant par la procédure et terminant par la génération des différents signaux nécessaire au bloc MLI. Toutes ces taches ont été élaborées et testées sur PC et expérimentalement. Les résultats obtenu sont satisfaisante et nous encourage à accomplir le travail par liée ces tache en boucle fermée pour réaliser la commande scalaire du moteur asynchrone.

CONCLUSION GENERALE

CONCLUSION GENERALE

Le travail présenté dans ce mémoire concerne l'implémentation de la commande scalaire d'un moteur asynchrone triphasé en utilisant un circuit FPGA.

Au premier lieu un aperçu a été donné sur les circuits reconfigurables tout en montrant leurs avantages par rapport aux autres circuits numériques à base de microprocesseur. En effet, l'architecture des FPGA, leur caractéristique ainsi que les moyens de prototypage et de configuration ont été présentés.

En suite, le moteur asynchrone triphasé, la cible de notre application, a été modélisée et simulé en appliquant la commande scalaire.

Dans le dernier volet, l'implémentation de la commande scalaire en boucle fermé a été abordée. La stratégie de l'encodeur avec a été décrit et synthétiser en implémentée sur une carte FPGA en utilisant le VHDL sous l'environnement de développement de XILINX. Pour valider l'application implémentée.

Comme perspectifs, les résultats obtenus nous permettant d'étendre objectif de la réalisation effectué vers :

_ Essayer d'implémenter d'autres commandes plus performantes pour le moteur asynchrone telles que la commande directe du couple (DTC) et a commande vectorielle.

BIBLIOGRAPHIE

[1] Zahir ait ouali « application des FPGA à la commande d'un moteur asynchrone », Thèse de master,Université Mouloud Mammeri,Tizi-Ouzou,Algérie.

[2] Nicolas Collins, « les circuits intégrés,PDF, www.reso-nance.org.

[3] B.K.Bose, « modern power electrics and ac drive »,prentice hall,upper saddle river,2002

[4] Moncef.KHITAS «Implantation d'application de traitement de signal sur système mono-puce reconfigurable 'SoPc' »,mémoire,univ FERHAT ABBAS,Setif-1-,UFAS Algerie.

[5] Corzine.K.A,wielebski.M.K,peng.F.Z,wang.j « control of multi-level inverter » : IEE power electrics Society, version 2003

[6] T.Sutikno,N.Idris,A.Jidin,and N.Cirstea, "An Improved FPGA Impementation of Direct Torque Control for Induction Machines",IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS,Vol.9,NO.3,pp.1280-1290,2013.

[7] Amaury.Bourbon,et Marie joubard « le codeur innrémental ».

[8] Les encodeurs numériques,PDF

[9] T. Sutikno, A. Jidini,N.R.Idrisi, « New approach FPGA-based implémentation of discontinous SVPM »,Turk.J.Electr.Eng.Compt.SCI.,vol.18,p.6,2010.

[10] BABIO BAYALA, machines électriques,La machine asynchrone,PDF,Edition revue 2010.

[11] Tir Zoheir « modélisation des machines asynchrones », thèse.

[12] K.Bouhoune « application de nouvelle technique de commande aux machine asynchrones : monophasé et triphasé »,UNIV Houari Boumediene,alger,algerie,2005.

[13] N.Khenfer « machine asynchrone : ses modèles, son identification et sa commande »,thèse de doctorat en génie électrique,institut national polytechnique de lorraine,INLP,Nancy,,France,1995

[14] Cours-Commande_Scalaire,PDF.

[15] Patrick BRUNET, Introduction à la commande vectorielle des machines asynchrones, 25 Avenue Henri BRISSON ,PDF.

[16] Y.DIAF « modélisation et commande d'un onduleur triphasé pilote par MLI a structure multi niveaux », mémoire de master 2012.

[17] Pong P.chu, « FPGAprototyping by VHDL exaples »,Edition Jhon Wiley & Sons, ltd.,publication,2008.

[18] DS1104pc controler board,feature,realize 5.1,may 2006,DSPACE

[19] T.Abdelkrim,E.M.Berkouk K.Aliouan,K.Benamrane et T.