République Algerienne Démocratique et Populaire

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE Dr YAHIA FERES DE MEDEA

Faculté des sciences et de la technologie

Département de Génie Electrique et d'Informatique

laboratoire de Recherche en Electrotechnique et en Automatique(LREA)



Mémoire pour l'obtention du diplôme

Master en génie électrique

Option: Automatique des systemes

Théme:

Etude et implémentation de la stratégie de commande MLI sur un circuit FPGA

Réalisé par :

Mr: RAHMANI Mustapha.

Mr: MOULOUED Amine.

Proposé et dirigé par :

Dr: KABACHE Nadir.

Dr: MOULAHOUM Samir.

Promotion: 2012-2013

Remerciement

Au terme de ce travail, nous remercions Dieu de nous avoir donné la volonté et le courage pour accomplir ce Travail.

Nous tenons à exprimer vivement notre profonde gratitude à nos promoteurs Monsieur : Kabache Nadir et Moulahoum Samir pour leur confiance, leurs encouragements, leur suivi et pour les conseils qu'ils ont apporté pour l'achèvement de ce projet.

Nous tenons également à remercier l'ensemble de membres de jury qui nous ont fait l'honneur de juger notre travail.

Nous tenons à remercier toutes les personnes qui m'ont aidé de prés ou de loin avec leurs conseils.

Dédicace

Le seul remerciement est a dieu le tous puissant qui nous a données le courage et la patience de faire ce travail

En premier lieu je dédie ce travail à mes chers parents :

Ma très chère mère : la source d'affection et de joie

Mon adorable père : le symbole de patience et de sacrifice

A ma bien aime grand mère source de délicatesse et d'affection

A mes très chers sœurs qui ma soutenu, mes très cher cousins et cousines "TOUFIK, MAHFOUD, ISMAIL, BOUDISSA, " ainsi qu'aux autres

A tous mes amies "Aissa, Abdel Kader, Samir banbo, Rabah, Nadir, Fouad, Fouzi… " Ainsi que mes collègues " Youcef, Rabah, Malek " et a tous mes collègues de l'université et en particulier les étudiants de

2éme année Master Génie Electrique AUTO.

A tous ceux qui me sont chers, qui pensent à moi et qui occupe une place dans mon cœur.

A tous ceux qui m'ont soutenu afin d'achever ce travail.

MUSTAPHA

Dédicace

Le seul remerciement est a dieu le tous puissant qui nous a données le courage et la patience de faire ce travail

En premier lieu je dédie ce travail à mes chers parents :

Ma très chère mère : la source d'affection et de joie

Mon adorable père : le symbole de patience et de sacrifice

A mon bien aime frères ''SOFIANE, BACHIR '' source de délicatesse et d'affection

A mes très chers sœurs qui ma soutenu, et bien sure leurs maries FAROUK, MOHAMED et le petit WASSIME et la petite SIRINE.

A mes très cher cousins et cousines "OKBA, MOHAMED, YOUCEF, BILAL, ACHOUR, MOURAD, MOHA BRAHIM " ainsi qu'aux autres

A tous mes amies "AISSA, MOHAMED, HICHEM, NADIR, FOUAD, FOUZI, SAMIR, EL KAIED…. " Ainsi que mes collègues " YOUCEF, RABAH, MALEK, MOHAMED HAMDADOU, TOUBA" et a tous mes collègues de l'université et en particulier les étudiants de

2éme année Master Génie Electrique AUTO.

A tous ceux qui me sont chers, qui pensent à moi et qui occupe une place dans mon cœur.

A tous ceux qui m'ont soutenu afin d'achever ce travail.

AMINE

Les concepteurs des systèmes de commande modernes s'investissent dans le contrôle de projets associant plusieurs disciplines et technologies. Pour répondre aux différents besoins notamment la conception des systèmes de commande, le recours aux outils de Conceptions Assistée par Ordinateur est plus qu'indispensable. L'avantage de ces méthodes est de faire une conception matérielle et logicielle simultanément afin de réduire le temps de développement et d'augmenter la fiabilité par le test des prototypes virtuels avant la réalisation sur un circuit intégré. Ce dernier sera le lieu d'implantation de la solution finale.

L'introduction et l'accumulation d'innovations technologiques sur les circuits intégrés ainsi que leurs expansion a permis l'automatisation des taches considérés complexes auparavant. L'approche classique de programmation séquentielle qui est une solution logicielle est considérée insuffisante pour la commande en temps réel car les exigences de temps d'exécution ne cessent d'augmenter. L'approche par la programmation architecturale qui est une solution matérielle permet de surpasser relativement cet inconvénient. Afin de développer des systèmes de commande de très hautes performances et d'améliorer la dynamique des machines. Mais la complexité croissante des algorithmes de commande des systèmes électriques ne cessent d'augmenter a cause des contraintes liées aux modèles et d'autres contraintes liées aux performances.

Les FPGA représentent une alternative pour compenser les DSP qui sont devenus classique avec une grande souplesse et de bonnes performances La logique programmable FPGA permet l'intégration de la circuiterie numérique de commande dans les systèmes automatisés et particulièrement la commande en temps réel des moteurs. Les FPGA présentent beaucoup de perspectives pour l'implantation d'algorithmes de contrôle des machines.

Pour ces FPGA, les outils de CAO servent à passer directement d'une description fonctionnelle en VHDL à un schéma en porte logique. Ces outils ont révolutionné la conception des circuits numériques .

Par ailleurs, il est bien entendu que la maitrise ou la commande d'une machine électrique passe indispensablement par la maitrise de son alimentation. Dans ce cas, lorsqu'il s'agit des entrainements à vitesse variable, on fait appel aux convertisseurs électriques. Vue l'importance de ces convertisseurs, en particule l'onduleur triphasée, une grande importance est donnée à sa commande.

Par conséquence, le sujet de présent travail est orienté vers l'utilisation des circuits intégrés programmables FPGA pour la commande des stratégies de commande plus précisément la stratégie MLI comme sujet d'application de notre travail. Ce travail a pour objectif l'élaboration d'une étude complète avec simulation et implémentation d'une stratégie MLI sur un circuit FPGA en vue d'améliorer les performances et la fiabilité du circuit de commande en tenant compte des spécificités structurelles des circuits reconfigurables FPGA.

La présent travail est organisé comme suite :

- Le premier chapitre consiste a présente les circuits reconfigurables FPGA. (Leur architecture et leur outil de programmation).
- Le deuxième chapitre présente un rappel sur la modélisation de la machine asynchrone, l'onduleur et ainsi que la modélisation de la stratégie MLI.
- Le troisième chapitre conclure les étapes de travail effectué et la validation et l'implémentation de notre solution avec un langage VHDL.

Enfin, nous clôturons ce document avec une conclusion générale, faisant rappel aux résultats obtenus et les perspectifs envisagés.

I.1 Introduction

L'histoire des circuits intégrés est commencée après l'invention des transistors en 1947 par les laboratoires Bell. Relativement, l'apparition du premier circuit intégré est au cours des années 1958 et 1959 grâce à un jeune ingénieur du nom de Jack Kilby de la firme de Texas Instruments qui a intégré sur un même substrat de silicium plusieurs éléments électroniques (transistors, résistances, capacités) ce qui est qualifié à cette époque le premier circuit intégré.

Juste après, dans les années 60, et plus précisément dans 1965 un des fondateurs de la compagnie Intel nommé Gordon Moore a fait une étude concernant l'évolution du secteur des circuits intégrés, ce qui a lui permit de prédire que le taux d'intégrations des transistors dans ces circuits doubles tous les deux ans. A nous jour, ce constat reste vérifier sur les circuits FPGA qui n'ont pas échappé à cette loi, jusqu'au début des années 80 et même à une époque plus récente, la conception d'un système sur puce (SOC) n'était accessible qu'aux firmes et sociétés spécialisées à cause de la complexité des circuits et de fonctions à intégrer qui demandent divers efforts et compétences. Par conséquence des couts élevée et cette technologie est inaccessible au grand public.

Aujourd'hui, l'avènement de dernières générations des FPGA a permis de mettre la technologie SOC à la portée d'un public nettement plus large. Ceci est particulièrement depuis que les FPGA sont proposé à un prix très faible et raisonnable. Ce prodigieux essor a été rendu possible grâce aux progrès concernant les technologies de fabrication des transistors et les méthodes de conception assistée par ordinateur (CAO).

Le rôle des FPGA est d'intégrer des circuits logiques complexes. Ces circuits sont susceptibles d'être de reconfigurés (Architecture programmée modifiable) partiellement ou entièrement suivant l'application.

A cet effet et, avant d'aborder le vif du présent sujet, ce premier chapitre s'inscrit dans un contexte qui traite les FPGA et leurs positions au sein des autres systèmes digitaux. Nous allons ensuite décrire l'apport de cette technologie programmable sur la stratégie MLI Intersectivité.

I.2.Les circuits programmable

La plupart des circuits numériques programmables sont issue de la célèbre architecture ''***Architecture de VON-NEUMANN*** ''proposée 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 ou 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 composants très rapides a orienté les chercheurs à développer une autre solution qui sera complètement différente de deux précédentes architectures. Cette solution réside dans le mode de programmation qui est devenu architectural à logique câblée inversement aux deux premières architectures précédentes ou la programmation est séquentielle (fig I.1).

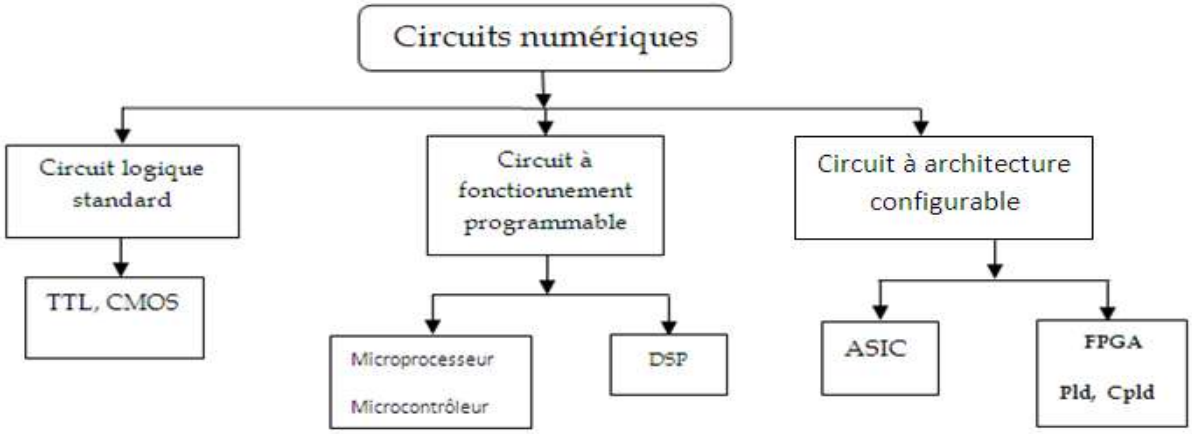


Figure I.1 : Classification des circuits numériques.

D'une manière générale, il existe deux alternatives ou solutions qui sont fig(I.2):

- ✓ **Une solution logicielle :** elle est nommée aussi solution programmables du type *Processeur* ou un traitement séquentiel relativement lent et programmation dépendante des composants (DSP, microprocesseur, microcontrôleur).
- ✓ **Une solution matérielle :** elle est nommée aussi solution programmable de type *Logique* ou un traitement parallèle en temps réel et une programmation architecturale avec un langage de description matériel HDL (Méthodologie de conception CAO) qui est indépendante du composant (ASIC et FPGA).

Les caractéristiques de ces circuits sont :

1. Les circuits du type **DSP/microprocesseur :** C'est un séquenceur qui exécute des ordres mémorisés constituant un programme. Avec un rapport performance/cout faible, un temps de conception très court et une grande souplesse d'utilisation.
2. Les circuits du type spécialisé **ASIC :** sont regroupe un grand nombre de fonctionnalités uniques, le concept des ASIC est d'assurer une optimisation maximale du circuit à réaliser, elle est très performants mais avec un cycle de conception long et une architecture figée.

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

4. Les circuits séquentiels ont une architecture matérielle figée et inversement pour les circuits logiques reconfigurables FPGA.

D'autre part, les fréquences de fonctionnement du mode séquentielle sont aujourd'hui 2 **GHz**. En effet, les circuits FPGA montre que la fréquence d'horloge de fonctionnement de ces derniers est relativement faible devant les microprocesseurs et les DSP et ne dépasse pas quelques centaines de **MHz,** 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. La figure (I.2), résume la différance entre les DSP et les FPGA **[01].**

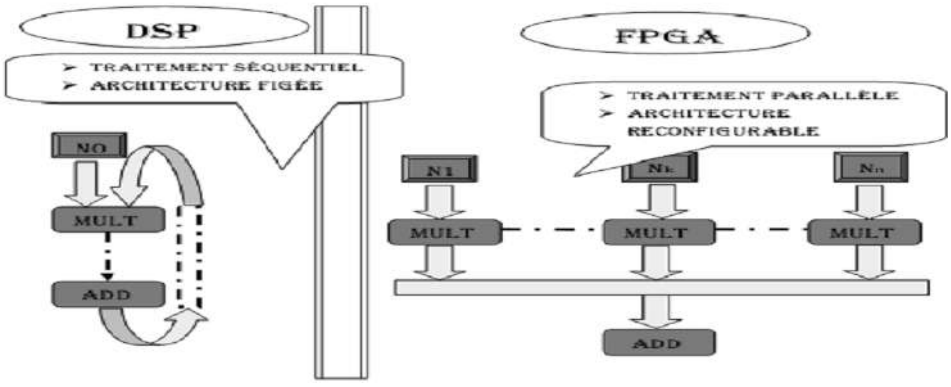


Figure I.2 : Schémas comparatif d'un **DSP** et d'un **FPGA.**

I.3. Les circuits logiques programmables

Alors qu'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***. La figure (I.3) illustre l'arborescence des différents types des circuits programmable **[02].**

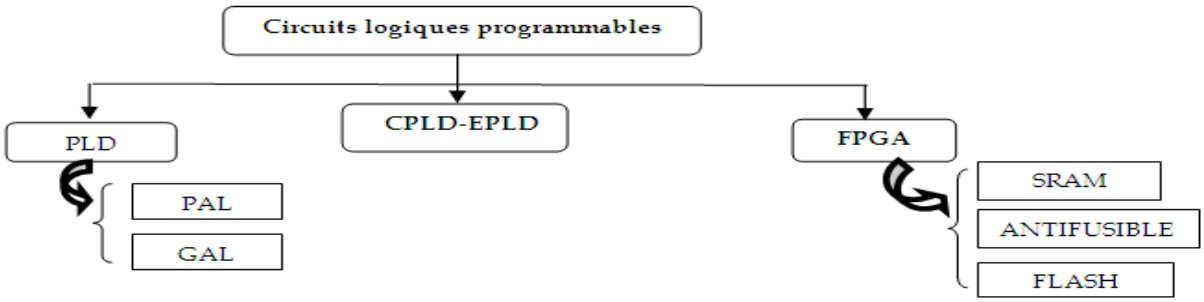


Figure I.3 : Diagramme des différents types de circuits logiques programmable.

1. Le **PLD (*Programmable Logic Device) :*** est un circuit intégré contient des éléments distribues de logique séquentielle et combinatoire, et un réseau d'interconnexion.
2. Le **CPLD *:*** Circuits logiques reprogrammables.
3. **Les FPGA (*Field Programmable Gâte Array) :*** ces circuits sont une évolution des **CPLD.** Récemment, ils intègrent également des mémoires entières, des multiplieurs et même des noyaux de processeur.

I.4. Les circuits logiques programmable de type 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, sont des composants VLSI, l'apparition de ce type de circuits est une révolution des systèmes digitaux et ouvrants des perspectives de traitement numérique inaccessibles auparavant. La fin des années 80 a 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 par 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 programmables. Pour réussir une application à base d'FPGA et afin d'obtenir un système plus performant, consommant un minimum de puissance, les circuits FPGA sont programmables ou configurables sur les cartes sur quelles ils sont implantés par l'utilisateur. Cette reconfigurabilité est une propriété nécessaire face aux systèmes à charges et contraintes variables. Le FPGA est une abréviation anglaise de qui signifie réseau des portes programmables sur site ce qui est décrit dans la figure suivante (**FIG :I.4**) [03]:



Figure I.4 : Reprogrammabilité sur site d'un FPGA.

Il est nécessaire de respecter un certain nombre de critères de choix tels que :

1. Couts.
2. Temps de conception.
3. Performances.
4. Fiabilité.
5. Maintenance à long terme.
6. Ressources.

D'une manière générale, le choix d'un circuit pour une application quelconque est justifié pour les exigences applications.

I.4.1. Technologie de programmation des FPGA

Il existe trois types de programmation d'FPGA, ces trois principales technologies sont :

- Technologie de programmation « **SRAM** ».
- Technologie de programmation «**EEPROM ou FLASH** ».
- Technologie de programmation « **ANTI -FUSIBLE**» [04].

I.4.1.1.Technologie à base de SRAM (**Xilinx** et **Altera**)

Cette technologie permet d'avoir une reconfiguration rapide des FPGA. 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,** la figure (I.6) représente la structure d'une cellule **SRAM (fig : I.5)**.

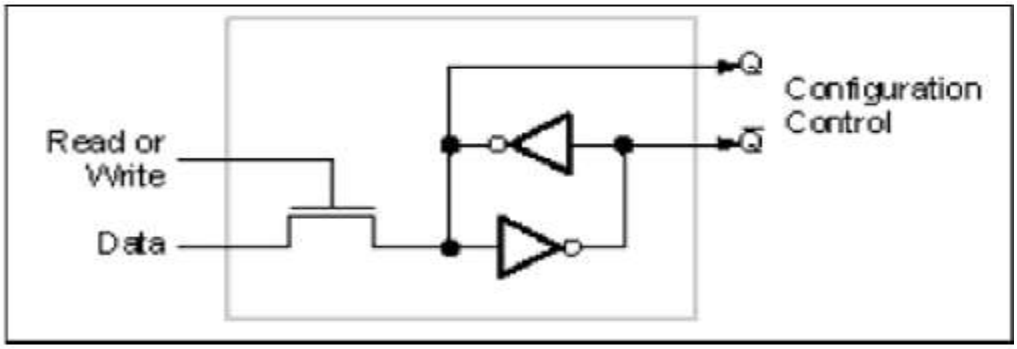


Figure I.5 : Structure d'une cellule **SRAM.**

I.4.1.2. Technologie à base d'EEPROM ou FLASH (**Lattice** et **Actel)**

Cette technologie garde sa configuration mais un nombre limité de configurations avec une configuration plus lente par rapport à **SRAM** [01].

I.4.1.3.Technologie à base d'ANTI-FUSIBLE

Les points de connexions sont du type ROM, c'est –à-dire que la modification du point est irréversible pour comprendre le mécanisme de connexions sans rentrer dans les détails des semi-conducteurs, on considère que le point de connexion est le point de rencontre de deux segments conducteurs ou lignes conductrices. Le non 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 [01].

I.5.Architecture interne d'FPGA (retenue par Xilinx)

Les circuits FPGA sont des composants VLSI (Very Large Scale Intégration) programmables par l'utilisateur. Ils possèdent une structure matricielle constituée de trois parties essentielles :

- Une matrice de blocs logiques configurables CLB (Configurable Logic Block).

- Des blocs d'entrées/sorties configurables IOB (In/Out Blocks).
- Un réseau d'interconnections programmables.

L'architecture interne des ***FPGA*** est différente d'un fabricant 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 (I.6) [05].

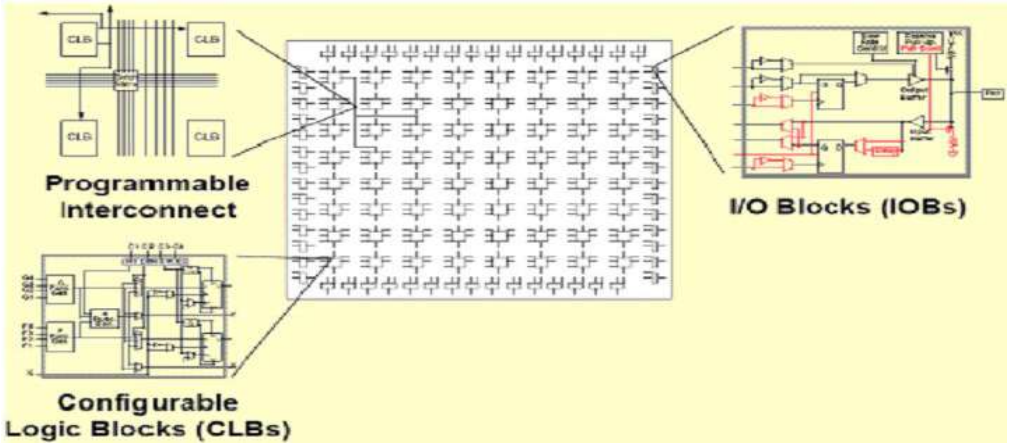


Figure I.6 : Architecture interne du ***FPGA.***

En plus, des ces élément de base, certains circuits **FPGA** intègrent également des mémoires **RAM**, des multiplieurs et même des noyaux de processeurs.

I.5.1. Les CLB (Configurable Logic Block)

La programmation d'un circuit FPGA est volatile, la configuration du circuit est donc mémorisée sur la couche réseau **SRAM** et stockée dans une **ROM** externe. Un dispositif interne permet à chaque mise sous tension de charger la **SRAM** interne à partir de la **ROM.**

Ainsi on constate aisément qu'un même circuit puisse être exploité successivement avec des **ROM** différentes puisque sa programmation interne n'est jamais définitive. On voit tout le parti que l'on peut tirer de cette souplesse en particulier lors d'une phase de mise au point. Donc, une erreur n'est pas rédhibitoire, mais peut aisément être réparée.

Les blocs logiques configurables sont les éléments déterminants des performances du circuit **FPGA**. Chaque CLB est un bloc de logique combinatoire composé de générateurs de fonctions à quatre entrées (LUT) et d'un bloc de mémorisation-synchronisation composé de bascules D. Quatre autres entrées

permettent d'effectuer les connexions internes entre les différents éléments du CLB. La LUT (Look Up Table) est un élément qui dispose de quatre entrées, il existe donc 2^4=16 combinaisons différentes de ces entrées. L'idée consiste à mémoriser la sortie correspondant à chaque combinaison d'entrée dans une petite table de 16 bits, la LUT devient ainsi un petit bloc générateur de fonctions. La figure ci-dessous montre le schéma simplifié d'un CLB de la famille XC4000 de XILINX (fig : I.7) [05].

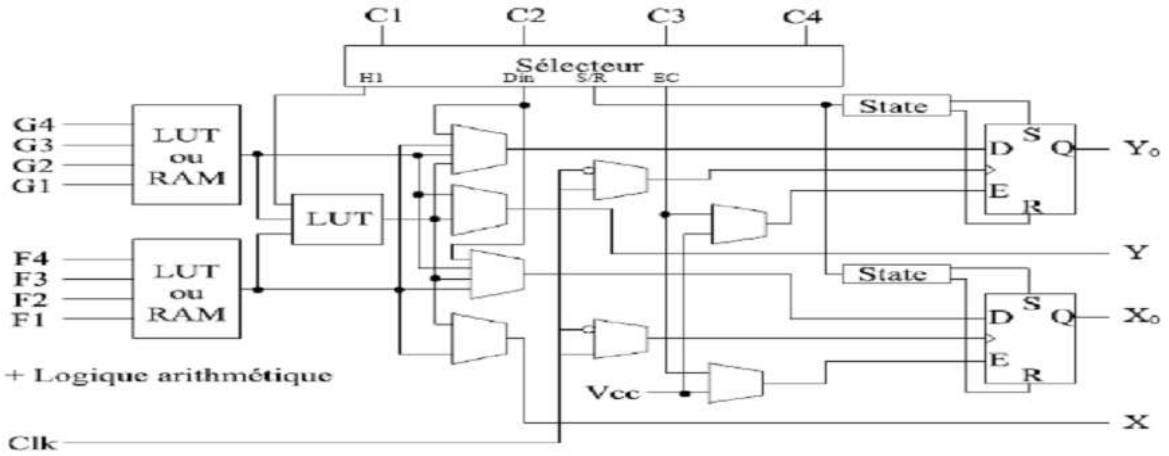


Figure I.7 : Schéma d'une cellule logique (XC 4000 de XILINX).

I.5.2. Les IOB (Input Output Bloc)

Ces blocs d'entrée/sortie permettent l'interfaçage entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA, chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance), la (figure I.8) présente la structure de ces blocs [06].

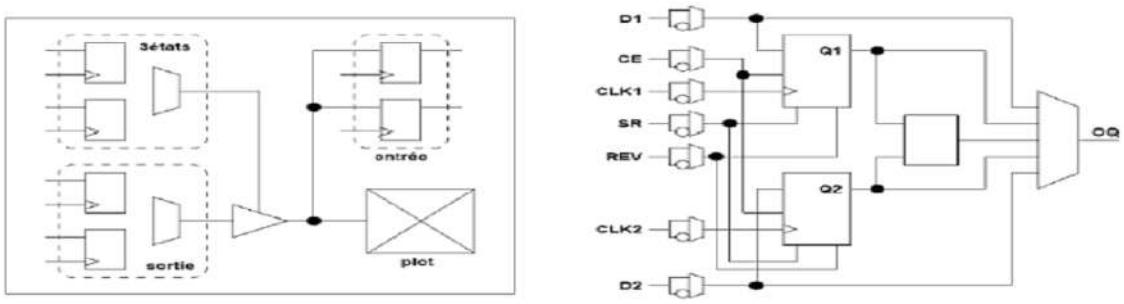


Figure I.8 : Schéma d'un bloc d'entrée/sortie (IOB).

I.5.3. Réseau d'interconnections programmables

Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM (Random Access Memory).

Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les blocs d'entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible [06].

Pour parvenir à cet objectif, Xilinx propose trois sortes d'interconnexion selon la longueur et la destination des liaisons :

- L'interconnexion à usage général.
- L'interconnexion directe.
- Les longues lignes.

I.5.3.1.Les interconnexions à usage général

Des aiguilleurs appelés aussi (Switch matrix) sont situés à chaque intersection. Leur rôle est de raccorder les segments entre eux selon diverses configurations, ils assurent ainsi la communication des signaux d'une voie vers l'autre. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre CLB.

Pour éviter que les signaux traversant les grandes lignes ne soient affaiblis, nous trouvons généralement des buffers implantés en haut et à droite de chaque matrice de commutation (figure **I.9**) [07].

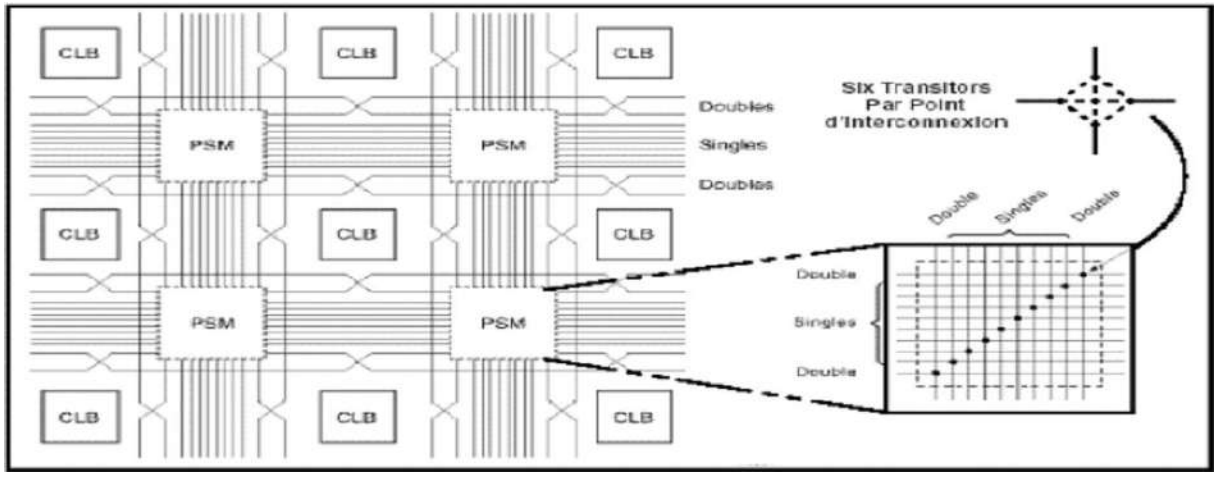


Figure I.9 : Connexions à usage général et détail d'une matrice de commutation.

I.5.3.2. Les interconnexions directes

Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en termes de vitesse et d'occupation du circuit. De plus, il est possible de connecter directement certaines entrées d'un CLB aux sorties d'un autre (fig : I.10) [07].

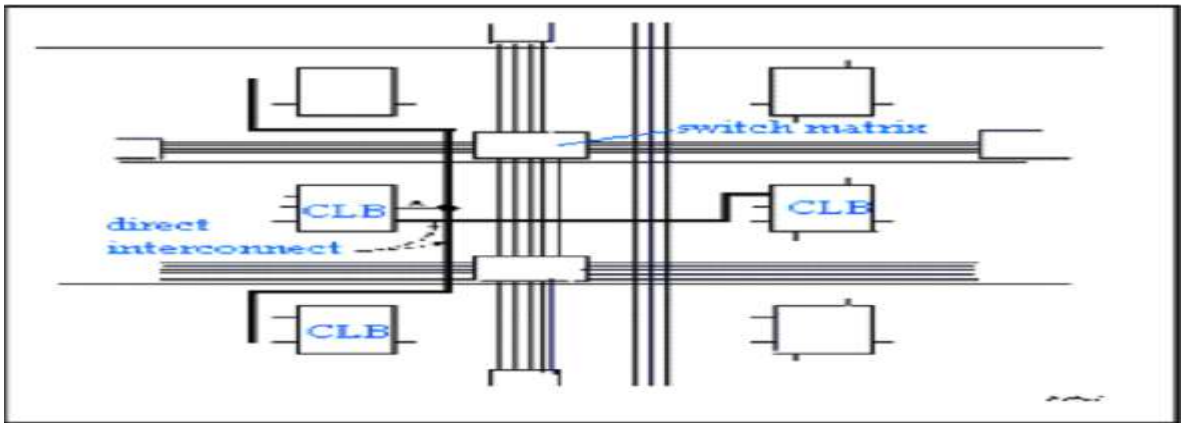


Figure I.10 : Les interconnexions directes.

I.5.3.3. Les longues lignes

Les longues lignes sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion[07].

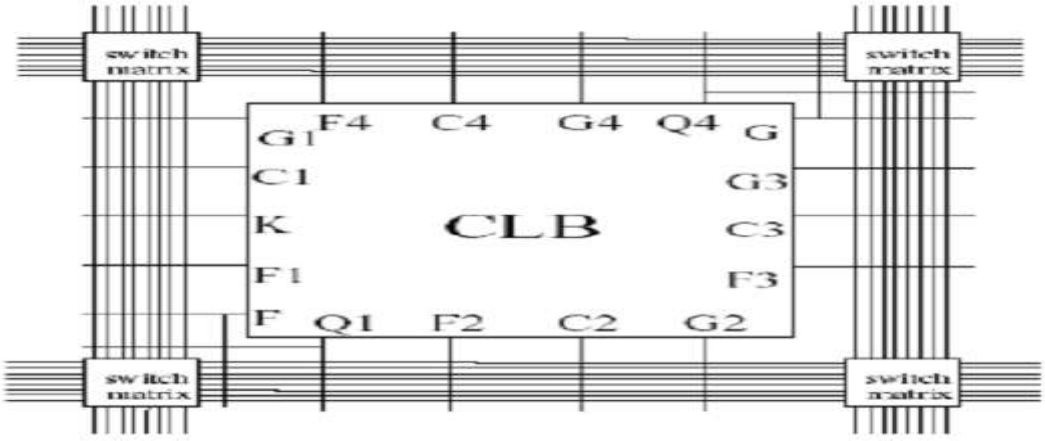


Figure I.11 : Les longues lignes.

I.6. Langage de description matérielle

La conception sur les circuits FPGA est un challenge dans lequel l'objectif est de trouver le bon compromis entre densité, flexibilité et performances temporelles. Vue la particularité de ces circuits, ils nécessitent des outils spécifique pour leur configuration. Etant donnée que la configuration de ces circuits consiste à modifier (programmer) leur architecture interne, il est nécessaire de recourir à un langage de description matériel.

En effet, un langage de description matérielle, est un langage informatique permettant la description d'un circuit électronique. Celui-ci peut décrire les fonctions réalisées par le circuit (description comportementale) ou les portes logiques utilisées par le circuit (description structurelle).Il est possible d'observer le fonctionnement d'un circuit électronique modélisé dans un langage de description grâce à la simulation.

En analysant l'évolution de la production industrielle d'**ASIC (A**pplication Specific Integrated Circuit) ou de FPGA (Field Programmable Gâte Array), on constate que ceux-ci, bénéficiant des progrès technologiques, sont de plus en plus complexes. On sait intégrer à l'heure actuelle sur silicium des millions de portes pouvant fonctionner à des fréquences supérieures à 600 MHz.

Qu'auparavant, si un ingénieur veut réalise un circuit de **n** portes (manuellement), il prend un temps de développement très élevé et un couts important , mais avec le développement de langage de description matérielle , pour faire une réalisation on va gagner beaucoup de temps et le couts de développement sera raisonnable [06] .

Le langage VHDL (Vhsic « Very High Speed Integrated Circuit » **H**ardware **D**escription **L**angage) est un sous-produit né du besoin d'un outil de spécification a pour but de décrire un fonctionnement global de haut niveau sans souci des détails d'implantation, pour servir de cahier des charges pour la réalisation des projets complexes. Ils sont devenus un véritable outil de conception des circuits numériques. Il permet d'autre part de concevoir et de vérifier un système électronique complexe.

Depuis son invention et son découvert par les militaires des Etats-Unis en 1980, la version initiale de VHDL incluait un large éventail de type de données (numériques, logiques…..).

Ceci a été adopté dans le VHDL-93(seconde version de la norme IEEE 1076). Afin de répondre aux différents problèmes de l'électronique, la norme VHDL a du évoluer.

En 1987, VHDL devient une norme IEEE. Elle permet la description et la simulation de circuits analogiques, numériques, et mixtes (analogique et numérique).

En général, une description VHDL est composée de 3 parties indissociables, à savoir (Fig I.12):

- ✓ ***Les bibliothèques,*** elle définit le types des entrées et sorties.
- ✓ ***L'entité (ENTITY),*** elle définit les entrées et sorties.
- ✓ ***L'architecture (ARCHITECTURE),*** elle contient les instructions VHDL permettent de réaliser le fonctionnement attendu [05].

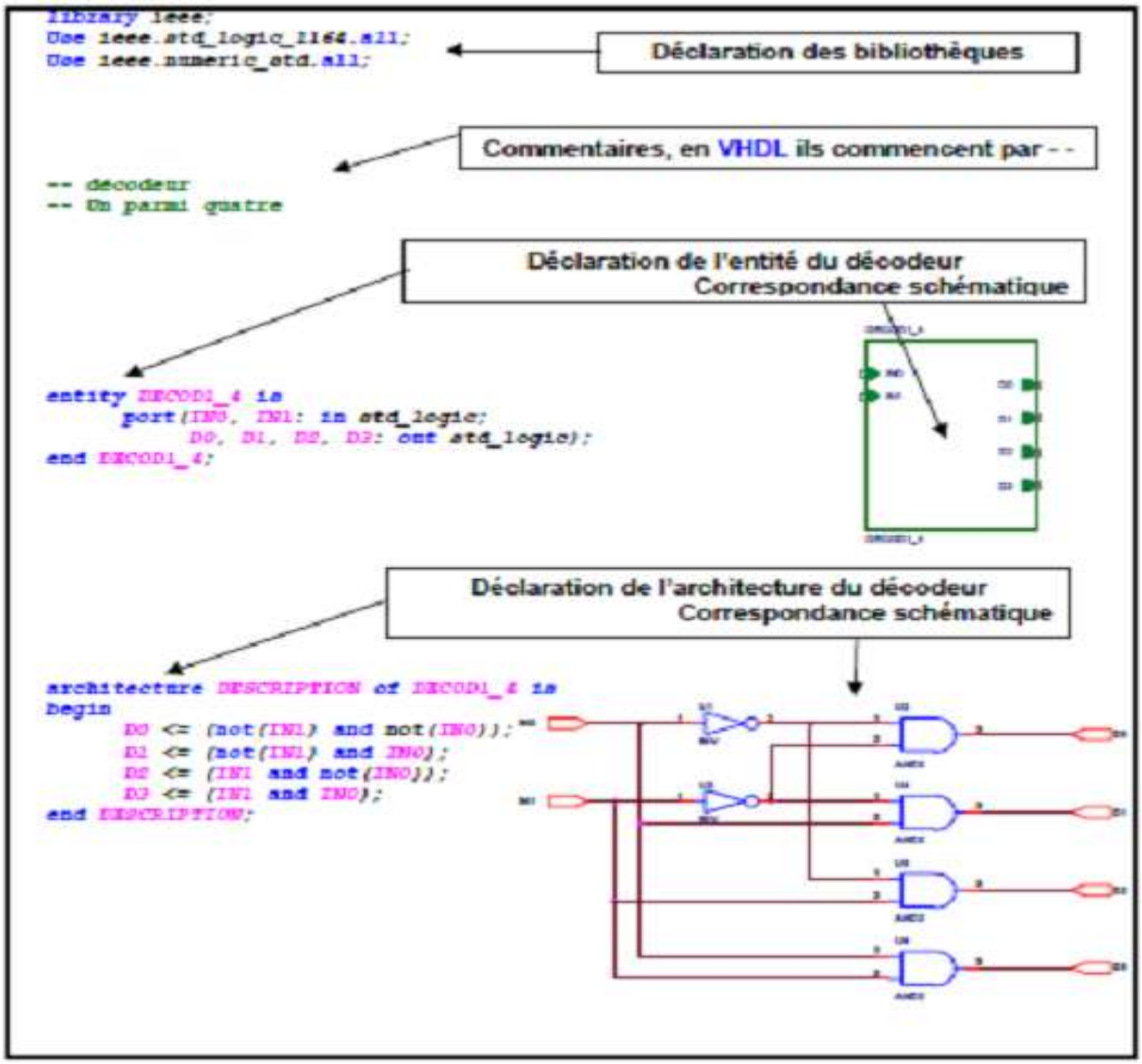


Figure I.12 : Différent parties d'une description VHDL

I.7. Configuration 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 Genrator) [08].

I.7.1. La configuration des FPGA par les outils HDL

Les techniques de conception CAO (Conception Assistée par Ordinateur) sont aujourd'hui très éprouvées et largement employées afin de concevoir des circuits électroniques nécessaires à mettre en pratique les connaissances algorithmiques. L'approche moderne pour la conception des circuits (logiques) électriques et la manière d'introduire une fonctionnalité sur un support physique sont confiées aux outils CAO. Les outils CAO sont utilisés pour générer le fichier de configuration des FPGA qui s'appelle (bit-Stream) à partir d'une description de haut niveau. Les principaux rôles confiés aux outils CAO sont :

- la description
- la simulation
- la synthèse
- le placement et le routage

Un design peut être conçu à l'aide d'un éditeur schématique ou d'un outil de traitement de textes. La conception de circuit simple peut se faire par l'approche schématique mais dans les circuits complexes elle cède le champ de conception à l'approche textuelle. Il est à noter que, dans le cadre de ce projet, c'est ce type de développement qui est retenu pour développer l'application envisagée.

Le flot de conception d'un système sur puce regroupe plusieurs niveaux d'abstraction. Dans chaque niveau, le concepteur s'intéresse à la résolution d'un problème. Les outils de **CAO** sont utilisés intensivement et assurent la transition entre les différents niveaux d'abstraction [08].

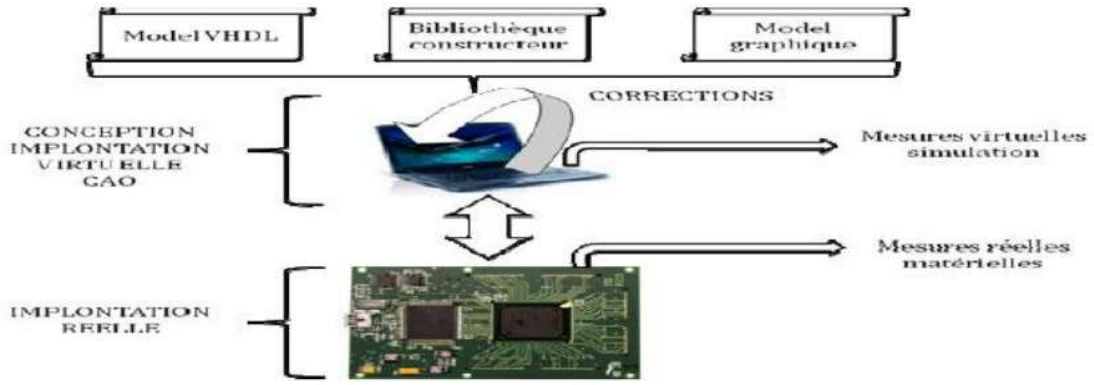


Figure I.13 : Mode d'exécution matériel de la CAO.

Le développement d'une application sur **FPGA** par des outils **CAO** suit l'enchainement des étapes suivantes:

I.7.1.1.Spécification du design

- Le nombre de broches 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.7.1.2.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 (Hardware Description Language). En effet, deux langages HDL sont largement utilisés, il s'agit du VHDL (Very High Speed Integrated HDL) et le Verilog. Ces deux langages sont standardisés, portables et compatibles avec toutes les types des **FPGA**. En plus de la description, certains signaux de test sont créés par le concepteur, dans cette étape, pour simuler le comportement de la description [01].

I.7.1.3.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.7.1.4.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 données qu'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:

- **Partitionnement**: Les équations logiques sont partitionnées en un autre ensemble équivalent d'équations. Chaque équation de ce nouvel ensemble peut être implantée dans un seul bloc logique du composant cible FPGA.
- **Placement** : Des blocs logiques sont sélectionnés dans la matrice et affectés au calcul des nœuds du réseau booléen.
- **Routage** : Les ressources d'interconnexion sont affectées à la communication de l'état des nœuds du réseau vers les différents blocs logiques qui en ont besoin.
- **Génération des données numériques de configuration** : Les informations abstraites de routage, de placement et les équations implantées dans les blocs sont transformées en un ensemble de valeurs numériques, qui seront chargées sur le composant FPGA [08].

I.7.1.5.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 ordinateur ou calculateur. C'est une étape de programmation physique et de tests électriques qui clôture la réalisation du circuit, La figure (I.14) résume un peut l'ensemble de ces étapes [09].

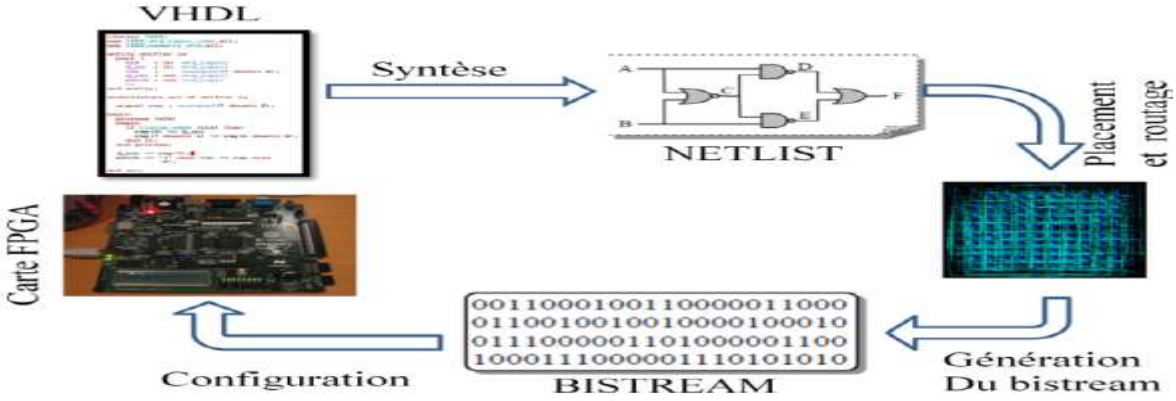


Figure I.14 : les étapes de développement une application sur FPGA.

I.7.2. La configuration des FPGA par les outils XSG

Les outils de description HDL représentent la solution typique et optimale pour réaliser les descriptions matérielles, toutefois, la mise en œuvre devient plus en plus difficile avec l'accroit de la complexité des applications à réaliser. Pour simplifier la description, Xilinx a développé un toolbox pour être intégré dans l'environnement Matlab-Simulink, permettant ainsi une interface graphique laissant l'usager créer des systèmes hautement parallèles pour FPGA d'une manière plus simple. Les modèles créés sont affichés sous forme de blocs, et peuvent être raccordés aux autres blocs et autres toolbox de Matlab-Simulink. Une fois le système complété, un genre de traduction vers le code VHDL s'effectue par l'outil XSG (Xilinx System Generator) reproduit exactement le comportement observé dans Matlab. Pour le prototypage rapide, le choix de cet outil est facilement explicable (**fig : I.15**) [08].

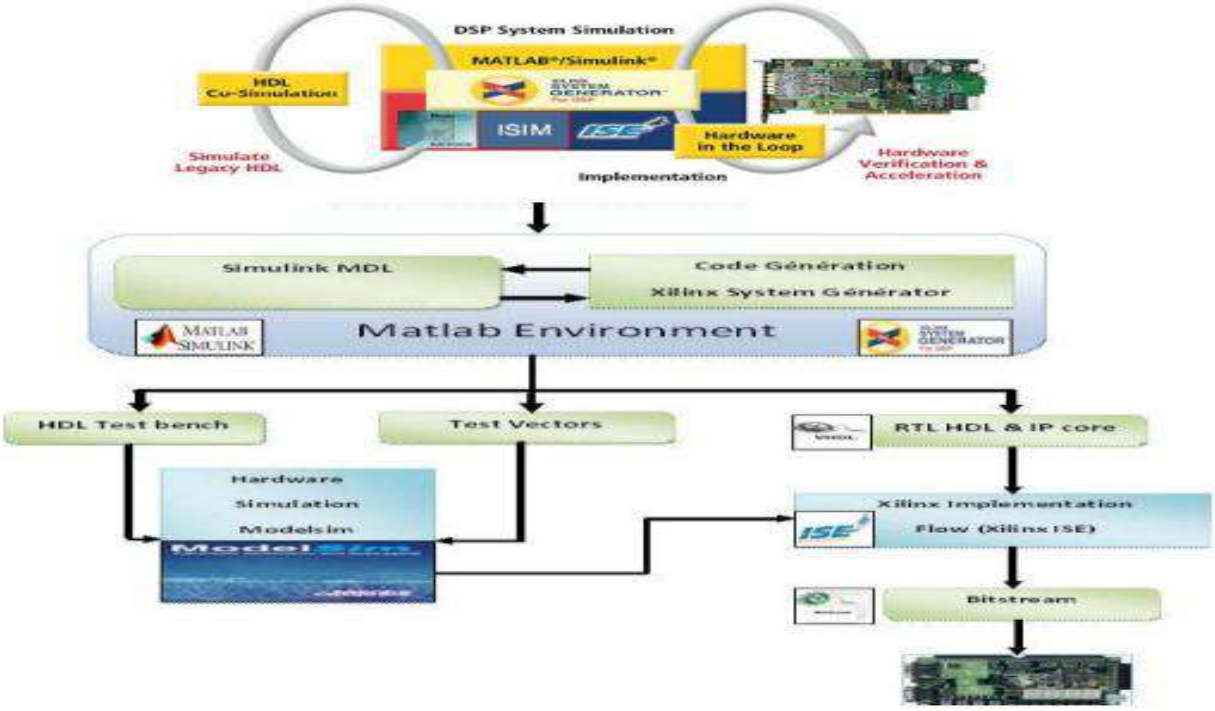


Figure I.15: Hardware Co-simulation XSG.

Les étapes de conception, de simulation et de vérification représentent des phases primordiales pour chaque système avant sa réalisation finale. Les coûts en termes de temps, d'effort et d'argent imposent aux concepteurs de recourir à des moyens plus sophistiqués pour optimiser ces coûts, en particulier lorsqu'ils deviennent considérables pour des systèmes complexes. La rapidité des circuits FPGA offre des solutions simples et prometteuses pour accélérer les étapes de conception et de

simulation. Actuellement, des concepts tels que le Co-design et la Co-simulation à l'aide des FPGA sont devenus très populaires [01].

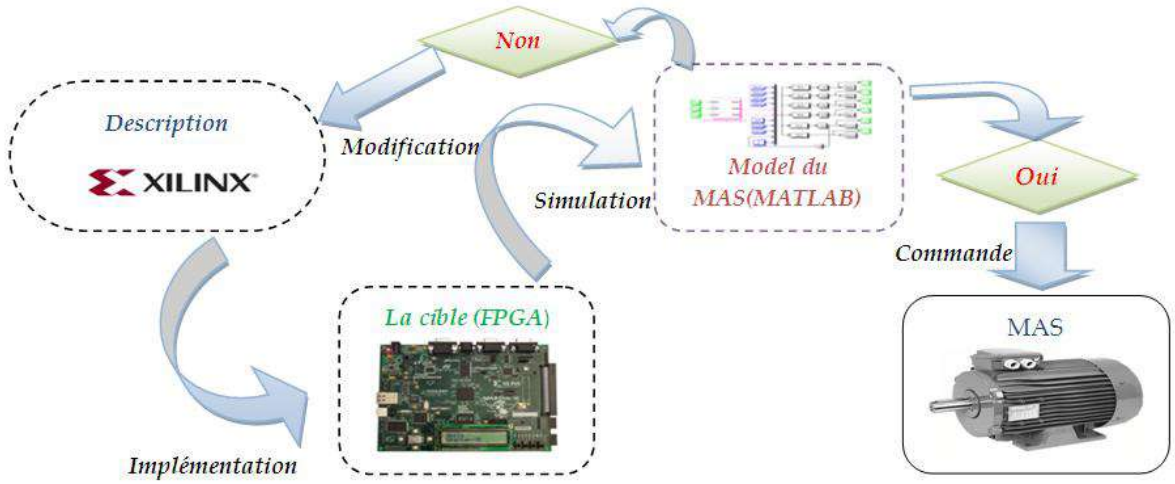


Figure I.16: Co-simulation ET Co-design par FPGA.

Pour les systèmes électriques, ces concepts facilitent pas mal de choses. Prenant comme exemple, la commande d'un moteur électrique. Quelque soit l'idéalité, prouvée lors de la simulation, des lois de commande développées théoriquement, leurs validité reste incomplète s'ils ne sont pas testés pratiquement. Dans ce cas, ce n'est pas une simple expérimentation qui est appliquée directement sur le système réel, mais il faut la répéter pour chaque réajustement ou modification du circuit de commande. On peut imaginer quels sont les résultats sur le système réel. L'utilisation des FPGA permet d'éviter tout ça en offrant un environnement pour teste ce circuit de commande sans user ou endommager notre système réel, en particulier, l'interface Matlab-XSG. Dans ce cas, la carte FPGA peut être utilisée en parallèle avec Matlab pour simuler le comportement du système : la carte comporte les lois de commande tandis que Matlab comporte le modèle du système. Le passage à l'application pratique ne s'effectue que lorsque on est sur du bon fonctionnent de tout le système. [08]

I.8. Différents domaines d'applications d'FPGA

Les FPGA ont fait révolutionner certain domaines de contrôle numérique et, de plus en plus, ils sont utilisés pour intégrer des architectures numériques complexes. Ils sont devenus très populaires en matière d'implantation et de prototypage des circuits numériques après leur apparition sur le marché en 1984. Leurs utilisations actuelles couvrent les deux domaines : civil et militaire. Parmi ses applications nous citons [10] :

1. **Informatique :** Périphériques spécialisés.
2. **Machinerie industrielle :** Contrôleur pour machines.
3. **Télécommunications :** Traitement d'images, Filtrage.
4. **Instrumentation** : Équipement médical, Prototypage.
5. **Transport :** Contrôle d'avions et métros.
6. **Aérospatiale :** Satellites.
7. **Militaire :** Radar, Communication protégée, la détection ou la surveillance.

I.9. Principaux fabricants d'FPGA

Les fabricants des FPGA ne cessent pas d'améliorer leurs produits par l'efficacité et la puissance. L'ensemble des principaux firmes qui conçoivent ce type de circuits sont : **Xilinx, Altera, Lattice, Actel, Lucent Technologie, Atmel, Cypress, Minc, QuicLogic** et d'autres, le tableau suivant représente la classification des principaux fabricants des circuits FPGA (TAB **I.1**) [11].

Fabricants d'FPGA	Pourcentages (%)
Xilinx	35.5
Altera	32.7
Lattice	16.1
Actel	6.7
Lucent Technologie	4.7
Autres	4.3

Tableau I.1 : Statistiques du marché occupé par le vendeur d '**FPGA.**

I.10. Avantages et inconvénients de FPGA

Les avantages et les inconvénients des FPGA sont multiples on trouve [12] :

Avantages	**Inconvénients**
• **T**echnologie ''facile'' à maitriser. • Temps de développement réduit. • Reprogrammable. • Idéal pour le prototypage. • Cout peu élevé. • Parallélisme de traitement. • Flexibilité et la possibilité de réduire fortement les délais de développement et commercialisation. • La reconfiguration, parfois en temps réel.	• Performances non optimisées. • Temps de réponse long par rapport aux ASIC. • Rapidité due au parallélisme par rapport aux circuits numériques standards (Mp, Mc, DSP)

Tableau I.2 : Avantages et inconvénients des **FPGA**.

I.11.CONCLUSION

Ce premier chapitre est dédié à la présentation des principaux concepts de l'outil qui sera utilisé pour réaliser le présent travail. Au départ, nous avons présenté les circuits programmables puis nous avons étudié le type FPGA ce qui nous a permis de conclure que cette technologie (FPGA) s'inscrit au sommet de l'évolution des composants logiques et le besoin croissant de composants plus performants, plus économiques.

La programmation des circuits FPGA, représente une étape essentielle avant tout mise en œuvre d'une application sur ce support. A cet effet, une partie de ce chapitre a été consacré à présenter les différents outils utilisé pour cette programmation. Parmi ces outils, la conception HDL, Ce langage de description matérielle est standardisé, portable et compatible avec toutes les types des **FPGA**.

II.1 Introduction

Depuis son invention et son découvert par ***NICOLA TESLA***, la machines asynchrone triphasée est largement utilisée dans l'industrie. Elle est de simple construction, très robuste et peu couteuse avec un excellent rondement. Cette simplicité s'accompagne d'une complexité de contrôle à cause des non-linéarités de son modèle mathématique et du caractère fortement couplé de ses variables d'état liées aux interactions électromagnétiques entre le stator et le rotor. La machine asynchrone a des avantages assez connus et assez nombreux. Par contre, elle est difficile d'en faire une variation de sa vitesse lors de fonctionnement. C'est la raison pour laquelle, le moteur à courant continu est 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 d'intérêts. Nous consacrerons ce chapitre au moteur asynchrone triphasé qui est notre application.

II.2. Description et principe de fonctionnement du MAS

Le moteur asynchrone triphasé est le plus utilisé pour la variation de vitesse et du couple dans les mécanismes industriels. Il est constitué d'une partie fixe, le stator qui comporte le bobinage (parcourus par des courants alternatifs triphasé), et d'une partie rotative, le rotor qui est bobiné par ensemble de trois enroulements qui sont court-circuités, et dans une machine asynchrone a cage d'écureuil, les spires au rotor sont constituées par des barres de fer entourant le rotor et formant une cage cylindrique appelée ''cage d'écureuil'' (**Fig : II.1)** [13].

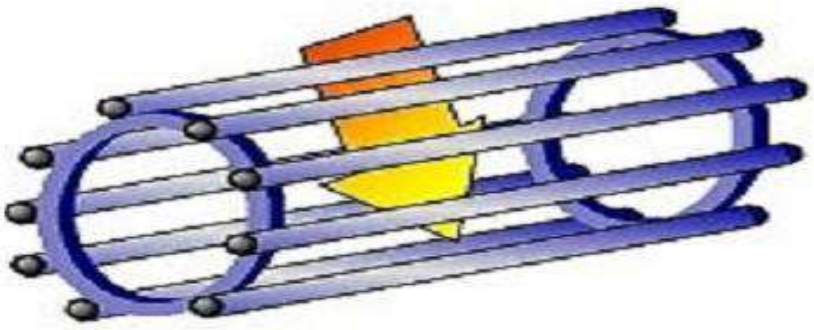


Figure II.1: Rotor d'une machine asynchrone a cage d'écureuil.

Les circuits magnétiques du rotor et du stator sont constitués d'un empilage de fines tôles métalliques pour éviter la circulation de courants de Foucault. Les trois enroulements statoriques créent donc un champ magnétique tournant (**Fig : II.2)**, sa fréquence de rotation est nommée fréquence de synchronisme. Le rotor «en cage d'écureuil» est balayé par le champ magnétique tournant. Les conducteurs sont alors traversés par des courants de Foucault induits. Des courants circulent dans les anneaux formés par la cage, les forces de Laplace qui en résultent exercent un couple sur le rotor. D'après la loi de Lenz les courants induits s'opposent par leurs effets à la cause qui leur a donné naissance. Le rotor tourne alors dans le même sens que le champ mais avec une vitesse légèrement inférieure à la vitesse de synchronisme de ce dernier. [13]

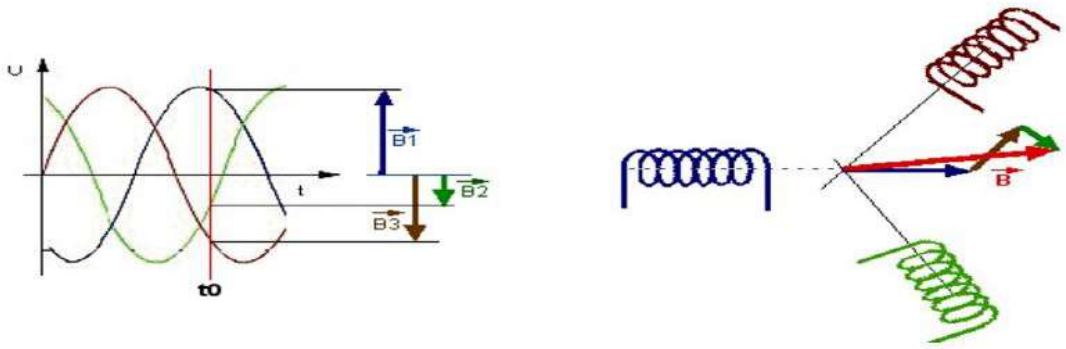


Figure II.2 : principe de fonctionnement d'un MAS.

II.3.Modélisation du moteur asynchrone

L'établissement des diverses relation qui constituent le système est en vue de command de ce dernière car la conception d'un système de commande en temps réel passe nécessairement par une étape de modalisation. La théorie généralisée des machines électrique triphasées classique, est basée sur la transformation de ***PARK*** qui rapporte les équations électrique statorique et rotoriques a deux axes perpendiculaires électriquement (direct et en quadrature). L'étude et l'analyse du moteur asynchrone consiste à obtenir un modèle mathématique représentatif du fonctionnement de ce moteur et qui permet de prévoir le comportement de ce système et l'évolution de ces variable d'état sous l'action d'un événement particulier avec une pris en compte de toutes les simplifications possibles et leurs influences sur les résultats de synthèse. L'étape de modélisation s'avère donc indispensable pour l'analyse et la synthèse de la commande du moteur. [14]

II.3.1 Modèle d'une MAS dans le référentiel triphasé (naturel)

- **Hypothèses simplificatrices**

Avant d'établir le modèle de la machine asynchrone, quelques hypothèses simplificatrices sont a adoptées, parmi lesquelles : [15]

- Une parfaite symétrie.
- Les trois phases sont considérés équilibrées.
- Non saturation des circuits magnétiques.
- La répartition de la force magnétomotrice est sinusoïdale.
- Les pertes de fer, l'effet des encoches et l'effet de peau sont négligés.

Par conséquence a ces simplifications, les flux sont additifs, les inductances propre sont constantes et une variation sinusoïdale pour les inductances mutuelles en fonction de l'ongle électrique de leurs axes de rotation.

II.3.2.Equations électriques sur les axes « a, b, c »

Les enroulements des trois phases, des côtés rotor et stator, dans l'espace peuvent être représentés comme indiqué sur la figure (**II.3**). A rappeler, les phases rotoriques sont court-circuitées sur elles-mêmes.

θ est l'angle électrique entre l'axe de la phase a statorique et la phase a rotoriques.

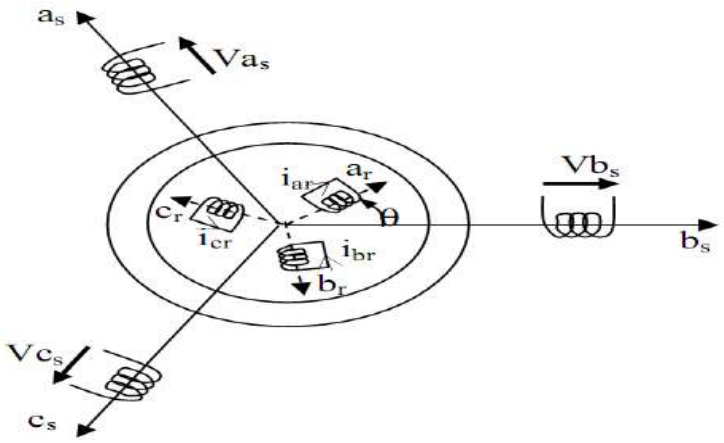


Figure II.3 : Représentation des enroulements statorique et rotoriques.

Les tensions statorique et tentions rotoriques dans une machine asynchrone sont données d'après la loi de ***FARADY*** comme suit [16] :

$$\boldsymbol{v = R_i + \frac{d\varphi}{dt}} \qquad (\mathbf{II.1})$$

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

$$[\boldsymbol{v_{abcs}}] = \boldsymbol{R_s}[\boldsymbol{i_{abcs}}] + \frac{\boldsymbol{d}}{\boldsymbol{dt}}[\boldsymbol{\varphi_{abcs}}] \qquad (\mathbf{II.2})$$

Cette notation est l'écriture condensée de :

$$\begin{bmatrix} \boldsymbol{v_{as}} \\ \boldsymbol{v_{bs}} \\ \boldsymbol{v_{cs}} \end{bmatrix} = \boldsymbol{R_s} \begin{bmatrix} \boldsymbol{i_{as}} \\ \boldsymbol{i_{bs}} \\ \boldsymbol{i_{cs}} \end{bmatrix} + \frac{\boldsymbol{d}}{\boldsymbol{dt}} \begin{bmatrix} \boldsymbol{\varphi_{as}} \\ \boldsymbol{\varphi_{bs}} \\ \boldsymbol{\varphi_{cs}} \end{bmatrix} \qquad (\mathbf{II.3})$$

La résistance statorique étant la même pour les 3 phases, il n'y pas lieu d'écrire une matrice de résistances.

De même pour le rotor :

$$\begin{bmatrix} \boldsymbol{v_{ar}} \\ \boldsymbol{v_{br}} \\ \boldsymbol{v_{cr}} \end{bmatrix} = \boldsymbol{R_r} \begin{bmatrix} \boldsymbol{i_{ar}} \\ \boldsymbol{i_{br}} \\ \boldsymbol{i_{cr}} \end{bmatrix} + \frac{\boldsymbol{d}}{\boldsymbol{dt}} \begin{bmatrix} \boldsymbol{\varphi_{ar}} \\ \boldsymbol{\varphi_{br}} \\ \boldsymbol{\varphi_{cr}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \qquad (\mathbf{II.4})$$

Le rotor étant en court-circuit, ses tensions sont nulles.

II.3.3.Equations magnétiques

Chaque flux comporte une interaction avec les courants de toutes les phases y compris la sienne (notion de flux / inductance propre).

Exemple de la phase $\boldsymbol{a}$ statorique :

$$\boldsymbol{\varphi_s} = \boldsymbol{l_s i_{as}} + \boldsymbol{m_s i_{bs}} + \boldsymbol{m_s i_{cs}} + \boldsymbol{m_1 i_{ar}} + \boldsymbol{m_3 i_{br}} + \boldsymbol{m_2 i_{cr}} \qquad (\mathbf{II.5})$$

D'où la représentation matricielle résultante est comme suite :

$$\begin{bmatrix} \boldsymbol{\varphi_{as}} \\ \boldsymbol{\varphi_{bs}} \\ \boldsymbol{\varphi_{cs}} \\ \hline \boldsymbol{\varphi_{ar}} \\ \boldsymbol{\varphi_{ar}} \\ \boldsymbol{\varphi_{ar}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{l_s} & \boldsymbol{m_s} & \boldsymbol{m_s} & \boldsymbol{m_1} & \boldsymbol{m_3} & \boldsymbol{m_2} \\ \boldsymbol{m_s} & \boldsymbol{l_s} & \boldsymbol{m_s} & \boldsymbol{m_2} & \boldsymbol{m_1} & \boldsymbol{m_3} \\ \boldsymbol{m_s} & \boldsymbol{m_s} & \boldsymbol{l_s} & \boldsymbol{m_3} & \boldsymbol{m_2} & \boldsymbol{m_1} \\ \hline \boldsymbol{m_1} & \boldsymbol{m_2} & \boldsymbol{m_3} & \boldsymbol{l_r} & \boldsymbol{m_r} & \boldsymbol{m_r} \\ \boldsymbol{m_3} & \boldsymbol{m_1} & \boldsymbol{m_2} & \boldsymbol{m_r} & \boldsymbol{l_r} & \boldsymbol{\varphi_{ar}} \\ \boldsymbol{m_2} & \boldsymbol{m_3} & \boldsymbol{m_1} & \boldsymbol{m_r} & \boldsymbol{m_r} & \boldsymbol{l_r} \end{bmatrix} \begin{bmatrix} \boldsymbol{i_{as}} \\ \boldsymbol{i_{bs}} \\ \boldsymbol{i_{cs}} \\ \hline \boldsymbol{i_{ar}} \\ \boldsymbol{i_{br}} \\ \boldsymbol{i_{cr}} \end{bmatrix} \qquad (\mathbf{II.6})$$

Avec

- $\boldsymbol{m_1} = \boldsymbol{m_{sr}}\cos(\theta)$
- $\boldsymbol{m_2} = \boldsymbol{m_{sr}}\cos(\theta - 2\pi/3)$
- $\boldsymbol{m_3} = \boldsymbol{m_{sr}}\cos(\theta + 2\pi/3)$
- $\boldsymbol{m_{sr}}$: est le maximum de l'inductance mutuelle entre une phase statorique et une phase rotorique.

II.3.4.Equation mécanique

Le comportement mécanique de la machine asynchrone dépend de l'inertie J, du couple électromagnétique $\boldsymbol{Ce}$, du couple mécanique résistant Cr et du couple de

frottement $C\ f(t) = K_f\ \Omega(t)$ ou K_f est la constante de frottement [16]. L'équation mécanique est donnée par :

$$J\frac{d\Omega(t)}{dt} + Kf\Omega(t) = Ce(t) - Cr(t) \qquad (\mathbf{II.7})$$

II.3.5. Modèle d'une MAS dans le référentiel biphasé

Le modèle de la machine asynchrone que nous venons de présenter, dans le paragraphe précédent, présente l'inconvénient d'être relativement complexe dans la mesure où les matrices d'inductances mutuelles qui vont être découlé de la dérivation des flux magnétique contiennent des éléments variable en fonction de l'angle θ. Une solution pour obtenir des coefficients constants consiste à appliquer une transformation mathématique au système et cette transformation est plus connue sous le nom de transformation de ***Park*** [16]. Le but de l'utilisation de cette transformation c'est de passer d'un système triphasé ***abc*** vers un système diphasé (dq) tournant avec une vitesse approprié. Suivant la vitesse attribuée au repère biphasé, on distingue deux formes de transformations [15] :

- Transformation de Clarke (repère tournant ou lié au champ tournant), Elle conserve l'amplitude des grandeurs mais pas la puissance ni le couple (on doit multiplier par un coefficient $\frac{3}{2}$)
- Transformation de Concordia (repère fixe ou lié au stator (**Fig : II.4**)), Elle conserve la puissance mais pas les amplitudes [17] (on doit multiplier par un coefficient $\sqrt{\frac{3}{2}}$), elle s'effectue de la manier suivant :

  Ou T_1 c'est la matrice de passage du repère $(\alpha, \beta, 0)$ on a :

$$\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \xrightarrow{T_1} \begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix} \text{ C.-à-d. } [x_{\alpha\beta}] = T_1[x_{abc}]$$

Avec

$$T_1 = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & \cos(\frac{2\pi}{3}) & \cos(\frac{4\pi}{3}) \\ 0 & \sin(\frac{2\pi}{3}) & \sin(\frac{4\pi}{3}) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \qquad (\mathbf{II.8})$$

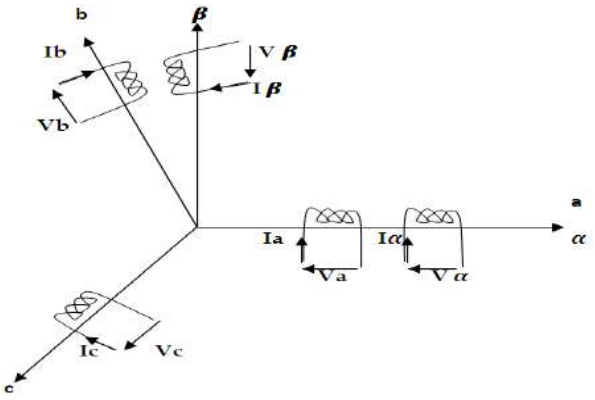


Figure II.4 : repère de Concordia.

A partir du repère de la figure(**II.4**), on ajoute un autre repère, ce référentiel est caractériser par un système d'axe (d, q) tournant a la vitesse $\boldsymbol{\omega c} = \frac{\boldsymbol{d\theta_c}}{\boldsymbol{dt}}$ par rapport au stator. Ce repère est appelé repère de **Park** (**Fig II.5**) .on peut définir une matrice de passage du repère ($\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{0}$) au repère ($\boldsymbol{d, q, 0}$). Cette matrice est notée $\boldsymbol{T_1}$, tel que [18] :

$$\begin{bmatrix} \boldsymbol{x_d} \\ \boldsymbol{x_q} \\ \boldsymbol{x_0} \end{bmatrix} = \begin{bmatrix} \mathbf{cos(\theta)} & \mathbf{sin(\theta)} & \mathbf{0} \\ \mathbf{-sin(\theta)} & \mathbf{cos(\theta)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_a} \\ \boldsymbol{x_b} \\ \boldsymbol{x_c} \end{bmatrix} \qquad (\mathbf{II.9})$$

Ou θ représente : θ_c : pour les grandeurs statoriques
$\theta_c - \theta_r$: pour les grandeurs rotoriques

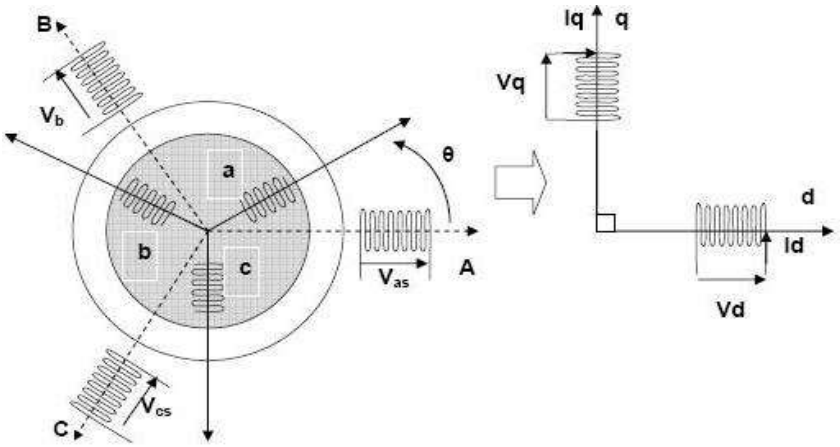


Figure II.5 : Transformation de Park.

Le passage direct du système triphasé au système diphasé tournant permet par une matrice notée $\boldsymbol{P}$ et définie par le produit matricielle
$[P]= [T_1]\,[T_2]$:

$$[P(\theta)]= \sqrt{\frac{2}{3}}\begin{bmatrix} \cos(\theta) & \cos(\theta-\frac{2\pi}{3}) & \cos(\theta-\frac{4\pi}{3}) \\ -\sin(\theta) & -\sin(\theta-\frac{2\pi}{3}) & -\sin(\theta-\frac{4\pi}{3}) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \qquad (\mathbf{II.10})$$

On définie de la même façon la matrice inverse permettant le passage (diphasé_triphasé) notée$[\boldsymbol{P}]^{-1}$.

L'application de la transformation de **Park** sur les enroulements statorique et rotorique conduit aux équations suivantes [16] :

$[X_{dqs}]= [P(\theta_c)]\,[X_{abc}]_s$ pour les grandeurs statoriques.

$[X_{dqr}]= [P(\theta_c-\theta_r)]\,[X_{abc}]_r$ pour les grandeurs rotoriques.

Donc les équations de la machine asynchrone deviennent :

- **Equations électriques**

$$\begin{cases} v_{ds} = R_s i_{ds} + \frac{d\varphi_{ds}}{dt} - \omega_c \varphi_{ds} \\ v_{qs} = R_s i_{qs} + \frac{d\varphi_{qs}}{dt} + \omega_c \varphi_{ds} \\ v_{qr} = 0 = R_r i_{dr} + \frac{d\varphi_{dr}}{dt} + (\omega_c - \omega_r)\varphi_{qr} \\ v_{qr} = 0 = R_r i_{qr} + \frac{d\varphi_{qr}}{dt} + (\omega_c - \omega_r)\varphi_{dr} \end{cases} \qquad (\mathbf{II.11})$$

- ω_c **:** vitesse angulaire du repère biphasé.
- ω_r **:** vitesse angulaire du rotor.
- $(\omega_c - \omega_r) = \omega_{sl}$

- **Equations magnétiques**

$$\begin{cases} \varphi_{ds} = L_s i_{ds} + L_m i_{dr} \\ \varphi_{qs} = L_s i_{qs} + L_m i_{qr} \\ \varphi_{dr} = L_r i_{ds} + L_m i_{dr} \\ \varphi_{qr} = L_r i_{qs} + L_m i_{qr} \end{cases} \qquad (\mathbf{II.12})$$

❖ **Equation mécanique**

Le couple électromagnétique peut être dérivé de l'expression de la Co-énergie ou obtenu à l'aide d'un bilan de puissance. Il en résulte plusieurs expressions toutes égales (où p est le nombre de paires de pôles) :

$$C_e = p(\varphi_{ds} i_{qs} - \varphi_{qs} i_{ds})$$

$$C_e = p(\varphi_{qr} i_{dr} - \varphi_{dr} i_{qr})$$

$$C_e = pM(i_{qs} i_{dr} - i_{ds} i_{qr})$$

$$C_e = p\frac{M}{l_r}(\varphi_{dr} i_{qs} - \varphi_{qr} i_{ds}) \qquad \textbf{(II. 13)}$$

Trois types de référentiels peuvent être envisagés à savoir :

► Référentiel lié au stator. $(\omega_c = 0)$et$(\omega_{sl} = -\omega_r)$

► Référentiel lié au rotor. $(\omega_c = \omega_r)$

► Référentiel lié au champ tournant. $(\omega_c = \omega_s)$et$(\omega_{sl} = \omega_s - \omega_r)$

Le premier référentiel est immobilier par rapport au stator : $(\omega_c = 0)$et$(\omega_{sl} = -\omega_r)$

$$\begin{cases} v_{ds} = R_s i_{ds} + \frac{d\varphi_{ds}}{dt} \\ v_{qs} = R_s i_{qs} + \frac{d\varphi_{qs}}{dt} \\ v_{qr} = 0 = R_r i_{dr} + \frac{d\varphi_{dr}}{dt} + \omega_r \varphi_{qr} \\ v_{qr} = 0 = R_r i_{qr} + \frac{d\varphi_{qr}}{dt} - \omega_r \varphi_{dr} \end{cases} \qquad \textbf{(II. 14)}$$

Ce référentielle est très intéressant lorsqu'on veut étudier la variation de la fréquence associée ou non avec la variation de vitesse de rotation. Il est aussi très utilisé pour la commande directe du couple et pour l'estimation des grandeurs électromagnétique. [19]

On peut, à partir de ces équations, les réécrire pour avoir un vecteur d'état (variables d'état du système) différent. C'est-à-dire qu'au lieu d'avoir les flux, on peut l'écrire en courants, il suffit juste d'opérer des substitutions dans les équations :

$$[x] = [\varphi ds, \varphi qs, \varphi dr, \varphi qr] \ ou \ [x] = [ids, iqs, idr, iqr] ;$$

Ou même avoir les flux rotoriques et courant statoriques :

$$[x]= [ids ,iqs, \varphi dr, \varphi qr].$$

Dans notre travail on utilise pour la modélisation le référentielle liée au stator et prend on considération les courant statorique et les flux rotoriques, ou :

$$[x]= [ids ,iqs, \varphi dr, \varphi qr].$$

Après les calcules et les simplifications les équations de la machine asynchrone deviens :

$$\begin{cases} \dot{\boldsymbol{I}_{\alpha s}} = \frac{1}{a}\boldsymbol{V}_{\alpha s} - \left(\frac{R_s}{a} + \frac{{l_m}^2 R_r}{{l_r}^2 a}\right)\boldsymbol{I}_{\alpha s} + \frac{l_m R_r}{{l_r}^2 a}\boldsymbol{\Phi}_{\alpha r} + \frac{\omega_r l_m}{l_r a}\boldsymbol{\Phi}_{\beta r} \\ \dot{\boldsymbol{I}_{\beta s}} = \frac{1}{a}\boldsymbol{V}_{\beta s} - \left(\frac{R_s}{a} + \frac{{l_m}^2 R_r}{{l_r}^2 a}\right)\boldsymbol{I}_{\beta s} + \frac{l_m \omega_r}{l_r a}\boldsymbol{\Phi}_{\alpha r} + \frac{R_r l_m}{{l_r}^2 a}\boldsymbol{\Phi}_{\beta r} \\ \dot{\boldsymbol{\Phi}_{\alpha r}} = \frac{l_m R_r}{l_r}\boldsymbol{I}_{\alpha s} - \frac{R_r}{l_r}\boldsymbol{\Phi}_{\alpha r} - \boldsymbol{\omega_r}\boldsymbol{\Phi}_{\beta r} \\ \dot{\boldsymbol{\Phi}_{\beta r}} = \frac{l_m R_r}{l_r}\boldsymbol{I}_{\beta s} + \boldsymbol{\omega_r}\boldsymbol{\Phi}_{\alpha r} - \frac{R_r}{l_r}\boldsymbol{\Phi}_{\beta r} \end{cases} \qquad (\mathbf{II.15})$$

Dans la suite de ce chapitre on fait la simulation de ce modèle (**II. 14**)en utilisant l'environnement MATLAB/SIMULINK.

II.4. La fonction de modulation MLI (PWM)

II.4.1.Généralités

L'étude de la régulation et la commande d'une machine alimentée via un dispositif d'électronique de puissance qui représente le variateur de vitesse passe par deux blocs ou étages (**Fig : II.6)** qui sont [20] :

- **Etage de commande :** Pour contenir un algorithme complexe de commande.
- **Etage de modulation :** Pour contenir la technique de modulation.

Le rôle de la fonction de la modulation est déterminer les instants de commutation et les ordres de commande logiques des interrupteurs afin d'obtenir une séquence de commutation de ces derniers. Le choix d'une stratégie de modulation peut s'effectuer en fonction des performances souhaitées par l'utilisateur et toutes les stratégies ont des avantages et des inconvénients et peuvent être réalisé par programmation logicielle ou matérielle. Il faut cependant remarquer que l'étage de modulation ne doit pas être confondu avec l'algorithme proprement dit de commande de la machine. [17]

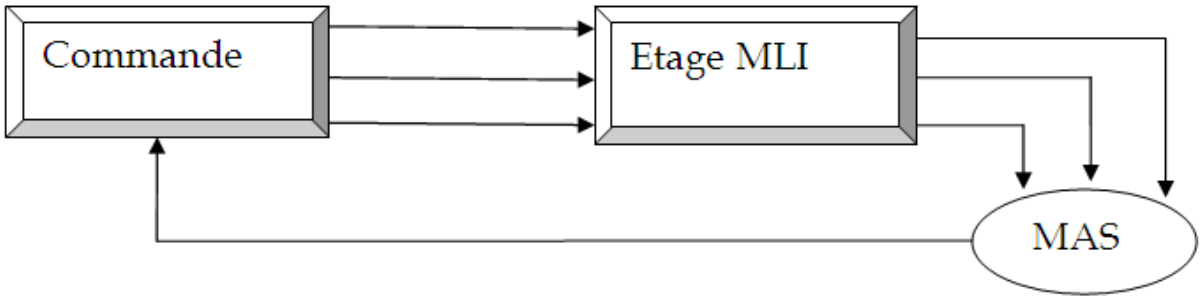


Figure II.6 : Schémas de position de MLI sur la chaine de régulation du moteur.

L'ensemble non-exhaustif de ces stratégies sont [17] :

- **Stratégie de modulation Tout ou Rien :** Le principe de cette stratégie est de commander les bras de l'onduleur par une comparaison de deux grandeurs (tensions ou courants) par des comparateurs.
- **Stratégie de modulation à Pleine Onde :** Le principe de cette stratégie est de commander les bras de l'onduleur tous les tiers de période.
- **Stratégies Modulation à largeur d'impulsions MLI :** Le principe de cette stratégie est de commander les bras de l'onduleur par une décision livrée par un algorithme au début de chaque période d'échantillonnage.
- **Stratégie de modulation Sigma-Delta :** Le principe de cette stratégie est de commander les bras de l'onduleur par une décision livrée par un algorithme durant chaque période d'échantillonnage.

II.4.2. La technique de modulation MLI

La technique de modulation en largeur d'impulsions MLI (PWM Pulse Width Modulation, en anglais) est l'essor et le fruit du développement d'électronique de puissance à la fin du siècle dernier. Elle est le cœur du contrôle des convertisseurs statiques. Le choix de la technique MLI pour contrôler l'onduleur de tension est en vue d'avoir une réponse rapide et des performances élevées et dépend du type de la machine a commander, du type des semi-conducteurs d'onduleurs, de la puissance mise en jeux et la simplicité ou complexité d'algorithmes. La technique MLI est composés d'impulsions dont la largeur dépend des choix effectués pour la stratégie de modulation [17].

Il existe plusieurs types de méthodes ou fonctions MLI sont résumées comme suit :

- **MLI Intersective (modulation par porteuses).**
- **MLI Pré calculée (Modulation pré-calculée).**
- **MLI Vectorielle (modulation poste calculer).**

II.4.3.MLI Intersective (modulation par porteuses)

Par principe, avec une simple comparaison entre un signal de référence (***la modulante***) et un signal triangulaire « ***dent de scie*** » de fréquence plus élevée (***la porteuse***), un signal de commande est généré pour chaque bras. La fréquence de porteuse définie la fréquence de découpage, et les points d'intersection entre la modulante et la porteuse correspondent aux instants de commutations au moment desquels l'onduleur change d'état. C'est une stratégie simple à réaliser en analogique [17].

Toutefois, une implantation numérique est assez compliquée où un nombre important d'échantillons de la modulante et de la porteuse doit être sauvegardé dans une mémoire ROM pour avoir obtenir une bonne précision du signal modulé. Parmi les variantes de cette technique, la plus populaire la modulation triangulo-sinusoïdale. Le principe d'élaboration de la MLI est résumé dans la figure (**II.7)**.

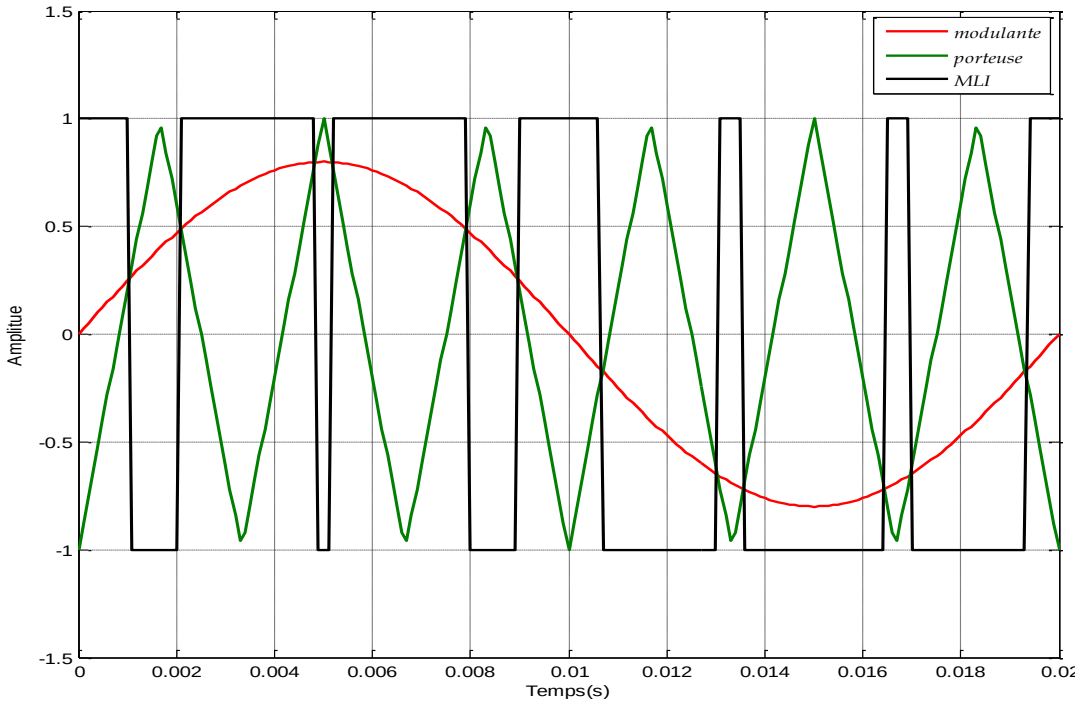


Figure II.7 : Principe de la SPWM (MLI sinusoïdale).

De part sa simplicité de mise en œuvre et de et de sa fiabilité, cette MLI a été l'une des plus utilisées avant le développement de systèmes numériques dédiés.

Dans les applications où l'onduleur de tension est connecté au réseau, la fréquence du fondamentale de la modulante est aussi celle des fondamentaux des tensions en sortie de l'onduleur : ainsi sa valeur est fixe (fréquence du réseau). L'indice de modulation **m**, permet de fixer la fréquence de commutation (ou fréquence de découpage) des interrupteurs de puissance. **[18]**

$$\mathbf{m} = \frac{\textbf{Fréquence de la porteuse}}{\textbf{Fréquence de la modulante}} \qquad (\mathbf{II.16})$$

II.5. Modélisation de l'onduleur :

La machine asynchrone est alimentée par un onduleur de tension MLI. Cet onduleur est alimenté par une tension triphasée alternatif (220/380 v) à travers d'un redresseurs à diode et d'un filtre LC délivrant une tension continue.

La figure (II.8) Présente un schéma simple d'un onduleur triphasé à structure de tension. Il se compose de trois bras à interrupteurs réversibles en courant, commandés à la fermeture et à l'ouverture, réalisés à partir d'un transistor (GTO ou IGBT) et d'une diode en antiparallèle [20].

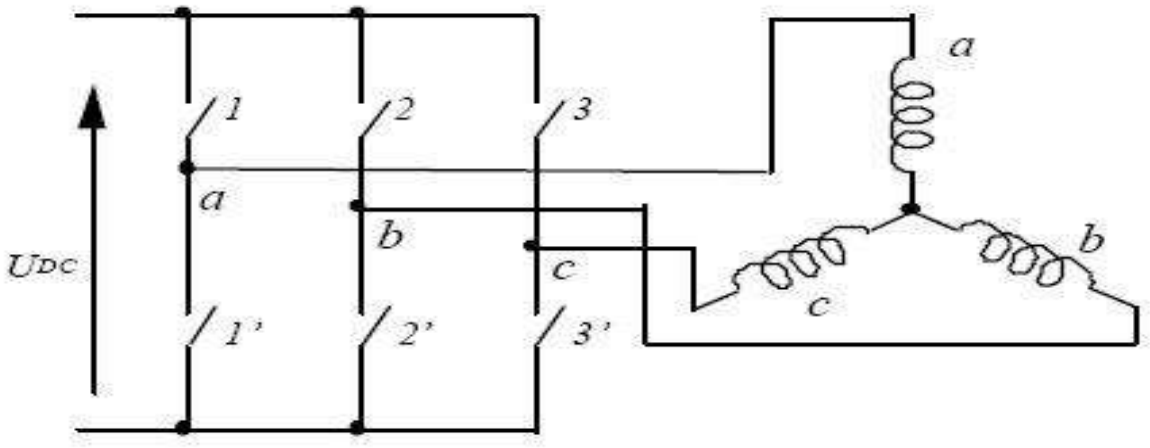


Figure II.8 : Onduleur de tension triphasé.

L'ouverture et la fermeture des interrupteurs de l'onduleur dépendent de l'état des signaux de commande (**a, b, c**), comme défini ci-dessous :

$$a = \begin{bmatrix} 1 \ T1 \ fermé \ et \ T'1 \ ouvert \\ 0 \ T1 \ ouvert \ et \ T'1 \ fermé \end{bmatrix}.$$

$$b = \begin{bmatrix} 1 \ T2 \ fermé \ et \ T'2 \ ouvert \\ 0 \ T2 \ ouvert \ et \ T'2 \ fermé \end{bmatrix}. \qquad (\mathbf{II.17})$$

$$c = \begin{bmatrix} 1 \ T3 \ fermé \ et \ T'3 ouvert \\ 0 \ T3 \ ouvert \ et \ T'3 \ fermé \end{bmatrix}.$$

Ainsi on pourra exprimer les tensions de lignes par:

$$\begin{cases} Uab = Udc(a-b) \\ Ubc = Udc(b-c) \\ Uca = Udc(c-a) \end{cases} \qquad (\mathbf{II.18})$$

Les tensions simples s'expriment par :

$$\begin{cases} Va = \frac{Udc}{3}(2a-b-c) \\ Vb = \frac{Udc}{3}(2b-a-c) \\ Vc = \frac{Udc}{3}(2c-a-b) \end{cases} \qquad (\mathbf{II.19})$$

II.6. Le modèle de MAS sur Matlab/Simulink

La figure schématique (**Fig : II.9**) représente le modèle de la machine asynchrone sur l'environnement MATLAB/Simulink :

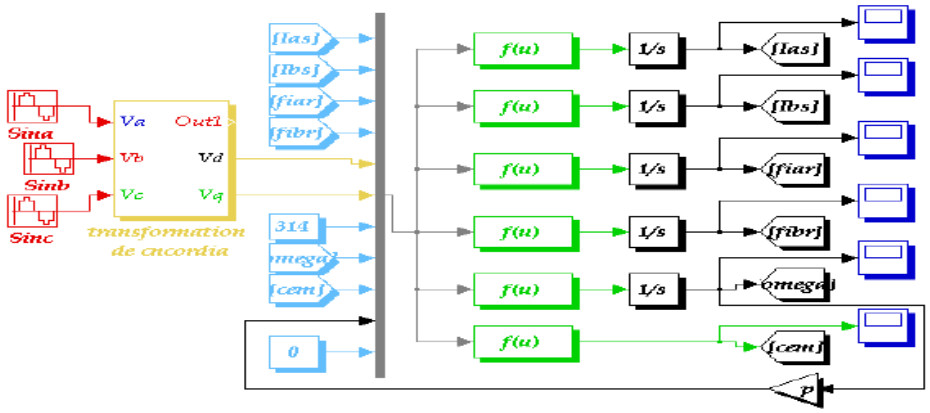


Figure II.9 : Schémas équivalent de modèle de la MAS

Apres la simulation on obtien les résultats suivants(**Fig : II.10)** :

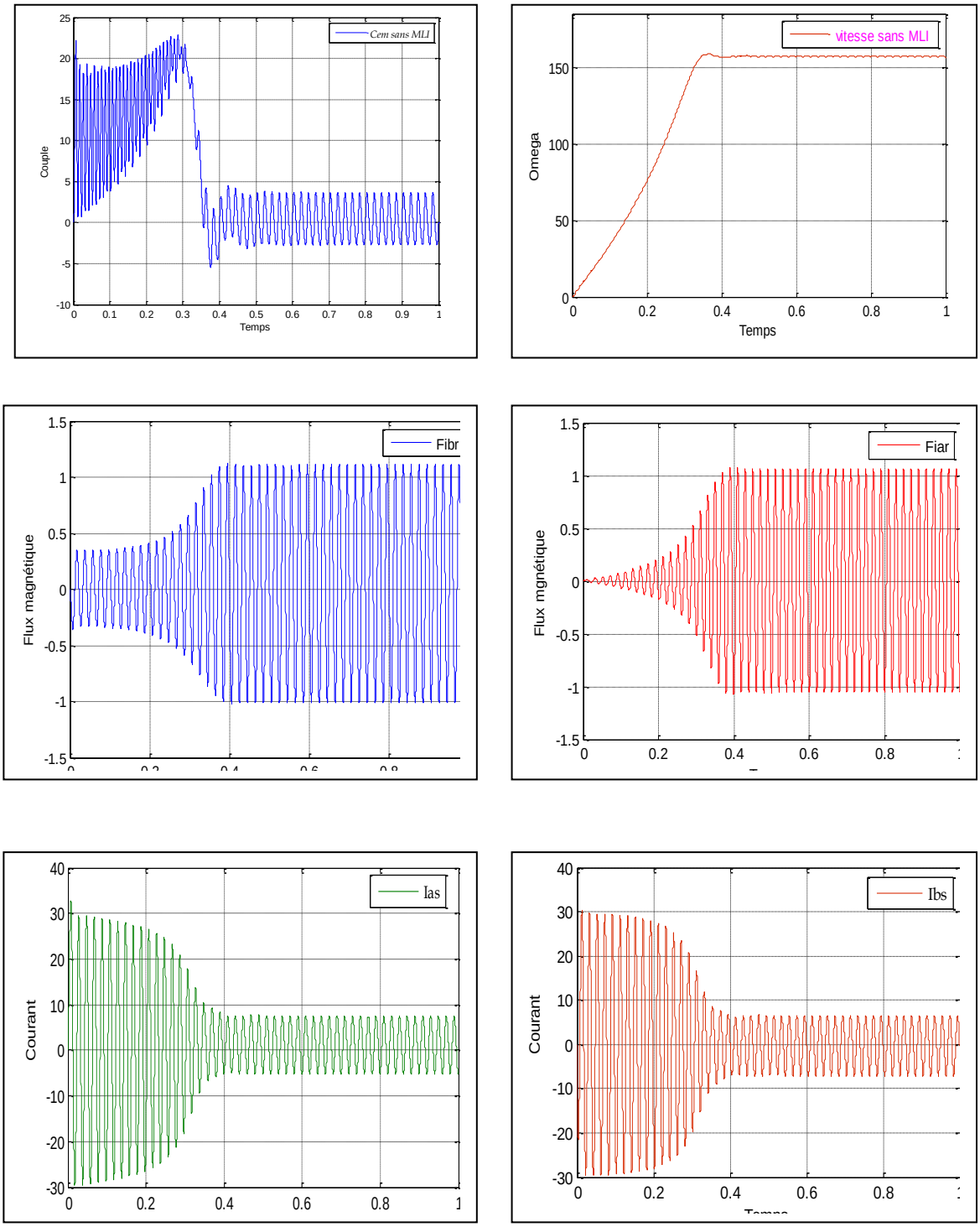


Figure II.10 *Résultat de simulation de modèle de la MAS*

II.10. La technique MLI sur Matlab/Simulink

La figure schématique (**Fig : II.11**) représente la stratégie MLI (sinusoïdaux triangulaire) sur l'environnement MATLAB/Simulink :

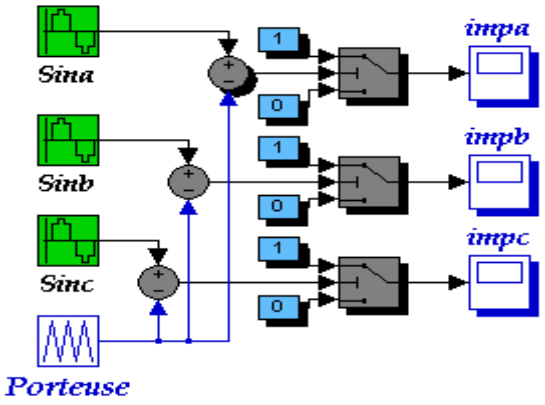


Figure II.11 : Schémas équivalent de la technique MLI sur Matlab/Simulink.

On obtient par simulation les résultats suivants (**Fig :II.12)**:

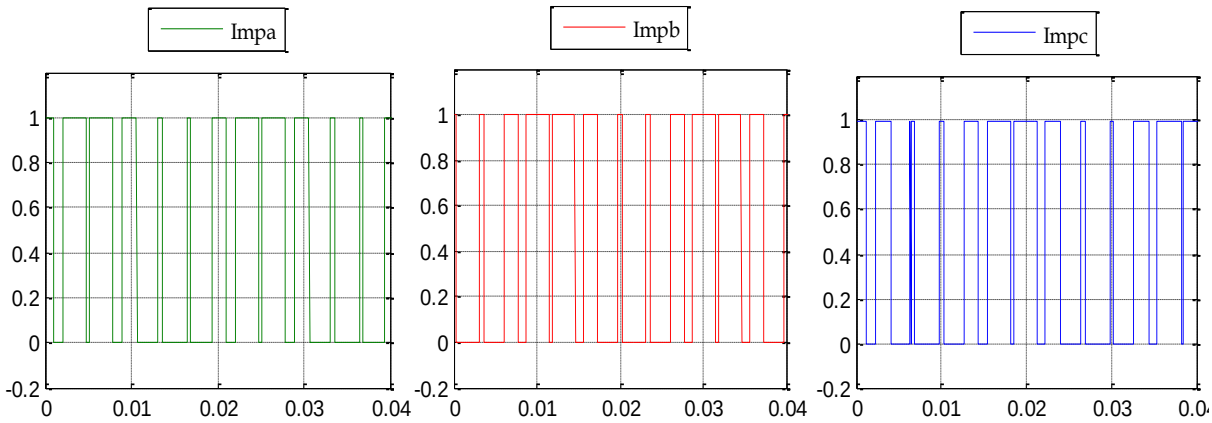


Figure II.12 : résultat de simulation de modèle de MLI

En suit on va injecter le schéma de MLI précédent (**Figure II.13**) dans un « subsystem » et le prend comme une source pour alimenter le model de la MAS

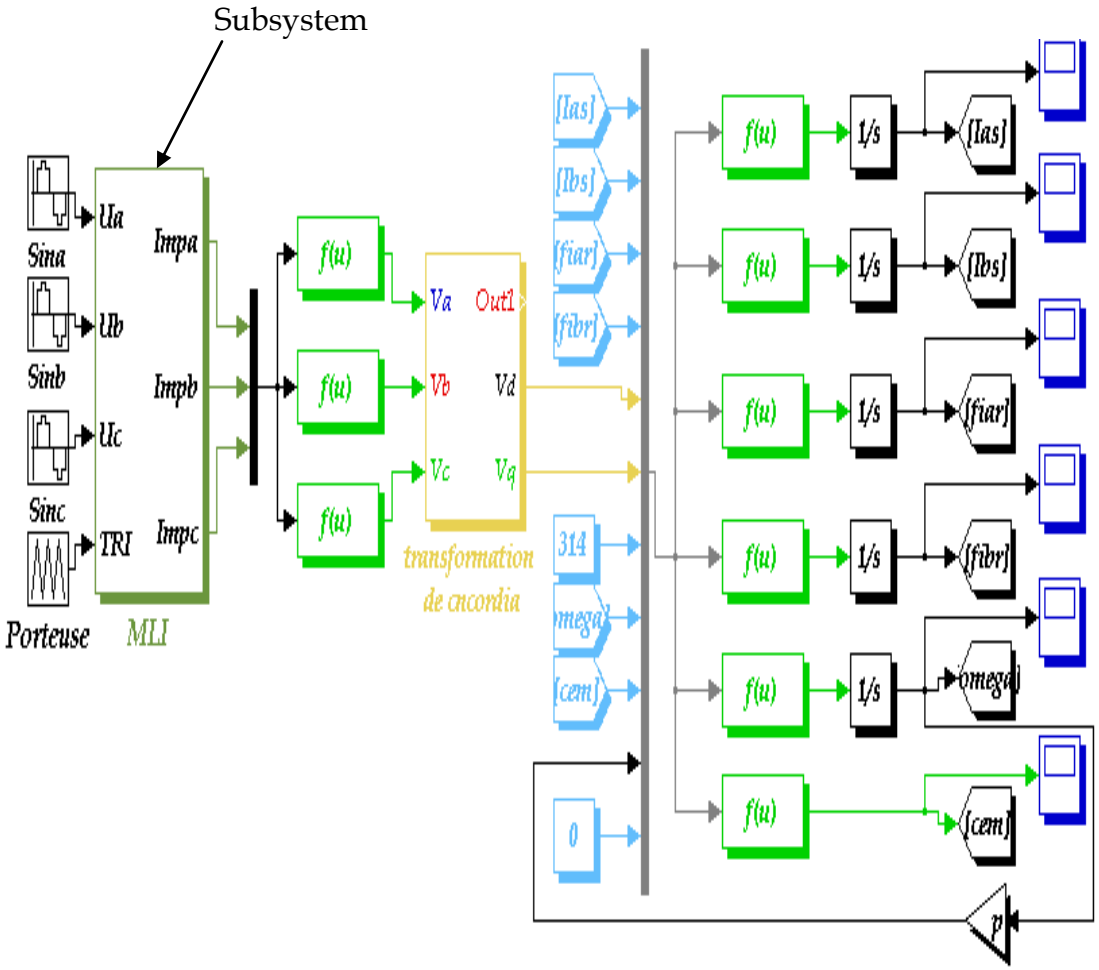


Figure II.13 : Schémas équivalent de la model de la MAS avec MLI.

La simulation de ce nouveau modèle causer un changement dans les résultats, tel que :

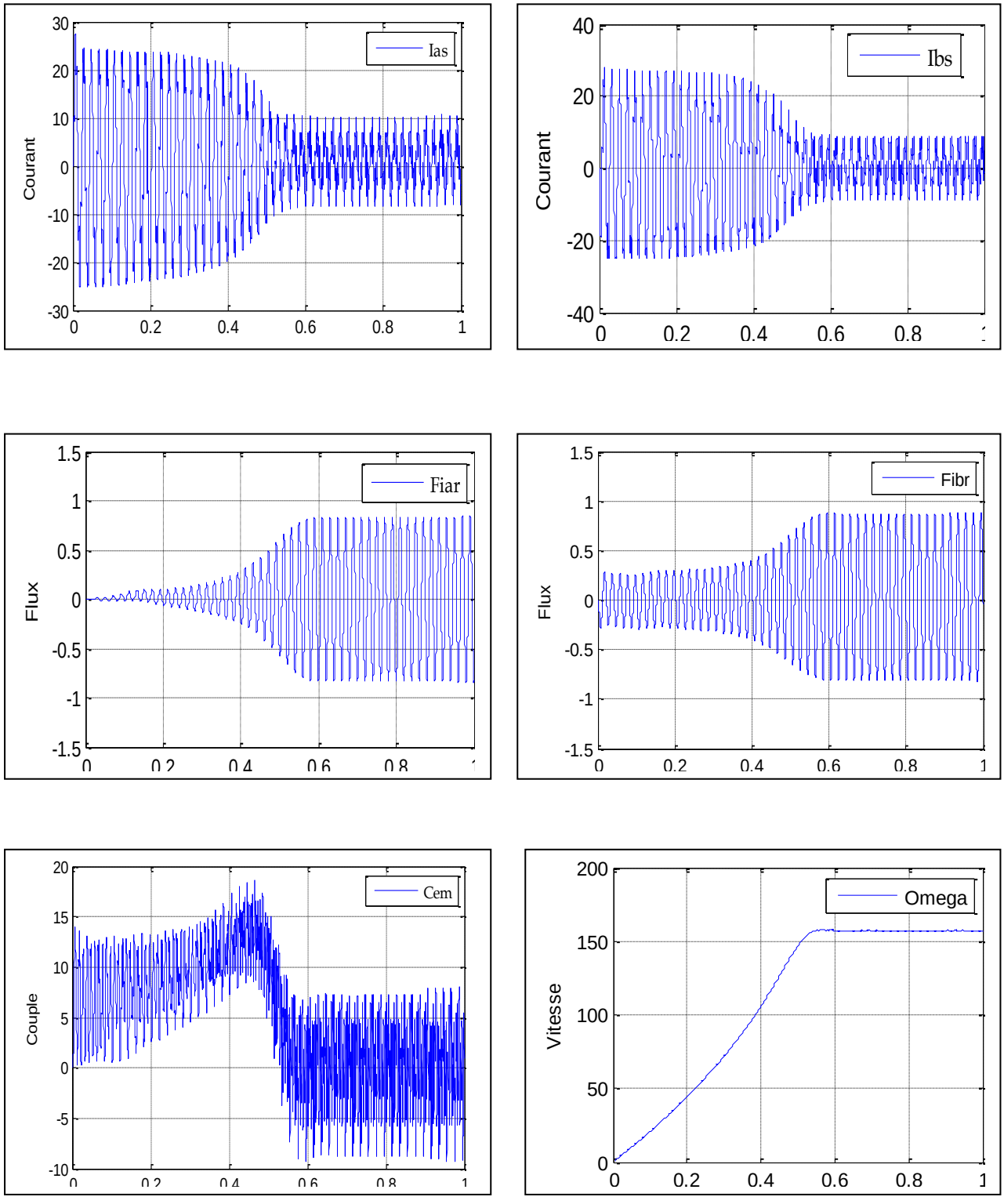


Figure II.14 *Résultat de simulation de modèle de la MAS avec MLI*

II.11.Conclusion

Ce deuxième chapitre a été consacré à la modélisation de la machine asynchrone triphasée ainsi que la détermination de leurs paramètres électriques et mécaniques et ainsi que la modélisation d'onduleur de tension triphasée.

En un premier temps on pose les équations différentielles qui régissent le comportement dynamique de la machine, en suite nous avons procédé à la transformation des axes qui permet de simplifier la modélisation du la machine asynchrone, a la fin de cette partie, on détermine le modèle mathématique de la MAS on utilisant la référentielle liée au stator.

En deuxième partie, nous avons étudié la fonction de modulation MLI qu'il permet de déterminer les instants de commutation et les ordres de commande logiques des interrupteurs afin d'obtenir une séquence de commutation de la stratégie MLI puis faire la modélisation de cette dernière.

En fin, nous avons montré la validité de nos modèles par l'utilisation de logiciel Matlab/Simulink, puis les comparais avec celles obtenue par l'environnement ISE XILINX qui est bien détaillé dans le chapitre suivant.

III. Introduction

Un ordinateur est incapable de concevoir un circuit car une machine est dépourvue d'intelligence et de réflexions mais rien n'empêche que les taches fastidieuses et complexes (*Mathématiques et logiques*) sont accomplies par des moyens automatiques confiés à ces calculateurs (*Ordinateurs*) qui sont dotés d'une importante puissance de calcule avec un minimum de temps d'exécution. L'électronique des circuits intégrés et la programmation informatique ont été les secteurs pionniers de la sureté de fonctionnement. Un besoin et une nécessité croissante pour la réduction de l'effort et du temps de conception des circuits, a rendu l'utilisation des outils CAO (Conception Assistée par Ordinateur) microélectronique indispensable. Les outils de CAO prennent en compte les contraintes de vitesse, de consommation de puissance, et de surface qui interviennent dans le processus d'optimisation de la description et de son circuit synthétisé. Cette partie du travail qui s'agit de concevoir, à partir du langage VHDL, un modèle équivalent l'algorithme MLI sachons qu'un modèle d'un circuit n'est qu'une abstraction de son comportement. Ce modèle va nous permettre de tester cet algorithme et d'assurer de la validité grâce au moyen de simulation software et hardware offerts par l'environnement ISE XILINX. Au cours de ce test la description fait et migrer sur un circuit FPGA.

Donc, ce chapitre s'occupe de la conversion vers un format exécutable (langage de description matériel) de l'algorithme MLI puis la vérification du circuit intégré correspondant à cette commande en utilisant les outils **de l'environnement ISE XILINX,** en particule son interface de visualisation hardware ChipScope qui est équivalons a un oscilloscope numérique.

III.1. Présentation de l'outil de développement ISE XILINX

Le logiciel **ISE XILINX *(Integrated Software Environment)*** est un environnement de développement qui possède différent outils de **CAO.** Tous les fabricants de **FPGA** proposent des outils de **CAO** pour configurer leurs circuits. L'offre logicielle dans le domaine de conception des circuits numériques est très varié et l'un parmi ces environnement que nous allons exploiter au cours de ce travail est **ISE XILINX**. C'est un logiciel multitâche qui possède dans son soft différents outils permettent la création des systèmes ou circuits numériques. L'introduction de projets se fait de deux manières qui sont textuelle ou graphique en vue d'une intégration dans un circuit logique programmable (CPLD ou FPGA). XILINX ISE permet la simulation de la description et la synthèse du circuit logique équivalent puis placer et router ce circuit sur un prototype correspondant à une technologie FPGA bien précise et enfin lorsque toute les vérifications sont faites vient l'implantation sur une carte réelle ce qui correspond à générer un fichier (bit Stream) correspondant au circuit logique

conçu avec optimisation de ressources disponibles au niveau circuit programmable FPGA. D'une manière générale, le XILINX ISE permet de réaliser toutes les étapes de conception et de programmation des FPGA de XILINX et même pour d'autres circuits programmables tel que les CPLD.

La conception de circuits sur XIILINX ISE met en œuvre quatre outils : Un éditeur de texte ou entrée graphique : c'est pour dessiner ou écrire le circuit avec une interface graphique ou textuelle

- Un simulateur : la simulation du système est faite pour vérifier la validité du code avant-synthèse
- Un synthétiseur : la synthèse consiste à faire la transcription de la description d'une forme texte vers une autre graphique (RTL) à base de portes logiques
- Un placeur-routeur : c'est pour l'adaptation du circuit logique synthétisé sur les ressources disponibles dans le circuit FPGA ciblé.

III.2.Présentation de L'analyseur ChipScope

L'analyseur ChipScope de Xilinx est un ensemble d'outils fabriqués par Xilinx qui vous permet de sonder facilement les signaux internes de la conception dans un circuit FPGA, mais moins que comme on peut faire avec un analyseur logique. Par exemple, si notre conception est exécutée sur le FPGA, nous pouvons le déclencher lorsque certains événements se produisent et afficher l'un des signaux internes de notre conception parce que la logique de l'analyseur de ChipScope est implémentée dans le FPGA. En autre terme, le ChipScope représente une version d'un Oscilloscope numérique (fig III.1).



Figure III.1 : L'analyseur ChipScope.

Pour cet outil de visualisation, on peut constater quelque inconvénient, tel que :

- La mémoire d'échantillons de l'analyseur est limitée par les ressources de mémoire du FPGA.

- Dans une conception qui utilise une grande partie de la mémoire du FPGA, il ne peut pas être beaucoup mémoire reste pour les noyaux de ChipScope.

En dehors de ces inconvénients, le ChipScope permet de donner une image des signaux circulant dans le circuit réaliser et implémenter dans une carte FPGA, ou certains avantage de cet outil par rapport aux analyseurs logique externes, sont à citer, permis lesquels :

- La réduction des délais de sondes pour l'analyse des signaux.
- La réduisissions de la dégradation des performances du circuit due à sonder.
- La portabilité et facilité à analyser des circuits sur FPGA.
- La minimisation de cout, l'analyseur logique cout très chers

Pour utiliser l'analyseur ChipScope dans un projet de conception existante, il faut générer les modules de base de ChipScope, qui configurer la gâchette (trigger) et la fonctionnalité de capture de forme d'onde sur le FPGA :

- Déterminer le mode de connections des modules de ChipScope avec les signaux que nous souhaitons surveiller.
- Recompiler la conception pour prendre en charge les modifications fautes.
- Charger le fichier (Bit Stream) résultant sur le FPGA en utilisant impact l'analyseur de ChipScope.

L'analyseur ChipScope fournit également l'interface pour la définition des critères de déclenchement pour les noyaux de ChipScope, et l'affichage des formes d'ondes enregistrées par ces noyaux

III.3.Réalisation et implémentation de la stratégie MLI dans un circuit FPGA

III.3.1.Descriptions VHDL de la stratégie(MLI)

La description *VHDL* permet de décrire le fonctionnement du circuit à réaliser en utilisant un code écrit portable, c'est-à-dire qu'une description écrite pour un circuit peut être facilement utilisée pour un autre circuit. Dans ce chapitre, on va essaye de faire la description de la stratégie MLI (triangulo_sinusoïdal), en utilisant un langage HDL sur l'environnement ISE XILINX (Fig : III.2).

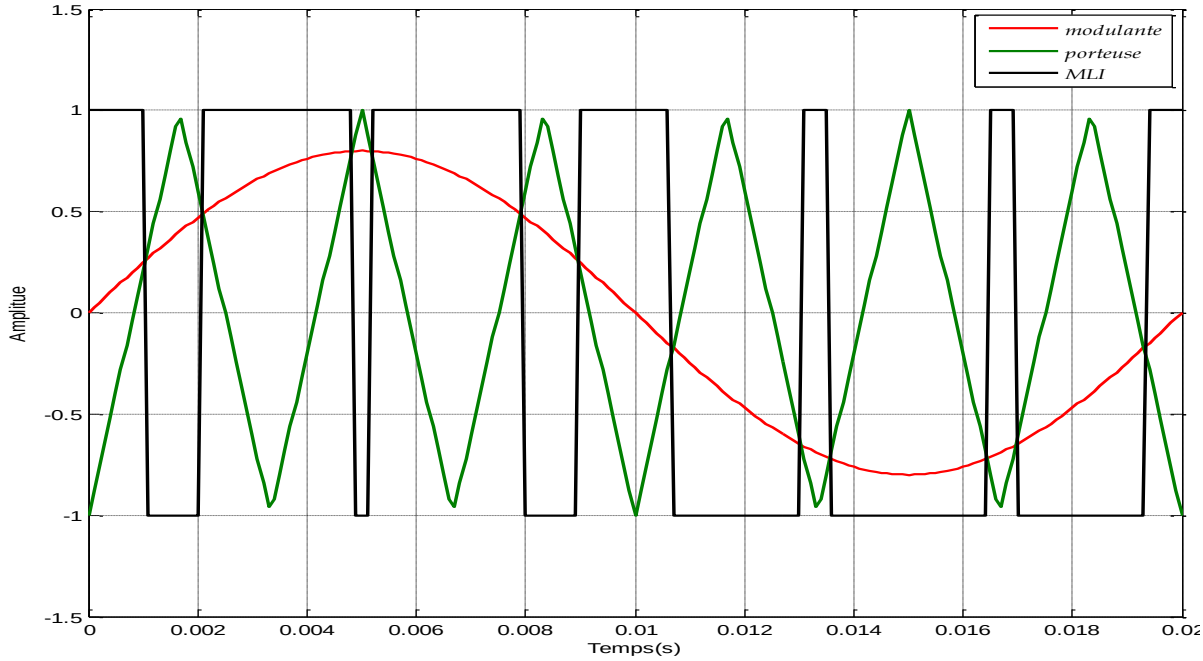


Figure III.2 : Principe d'élaboration de la MLI Intersective.

En effet, il a été invoqué dans le chapitre précédent, la stratégie (triangulo _ sinusoïdal), nécessite la génération du modulante (signale sinusoïdal), et la porteuse (signal triangulaire).

En suite, sur la base d'une comparaison entre ces deux signaux, les instants de commutation de l'onduleur sont générer (**Figure III.2**).

Pour générer les signaux nécessaire a la réalisation de l'MLI, $on a deux possibilités :

- L'utilisation de la formule mathématique de ces signaux, cette solution présente certain inconvénient tel que : la complexité des formules utiliser pour approximer les signaux sinusoïdaux, le nombre important de ressource appelée, le temps de fonctionnement…etc.
- L'utilisation des mémoires, dans ce cas les signaux nécessaires sont échantillonnées et stockés dans des mémoires en utilisant les ressources du circuit FPGA cible. C'est cette solution qui est adoptée dans ce travail.

La figure (**III.3)** montre un program MATLAB simple utilisé pour l'échantillonnage des signaux sinusoïdaux (Ua, Ub, Uc).

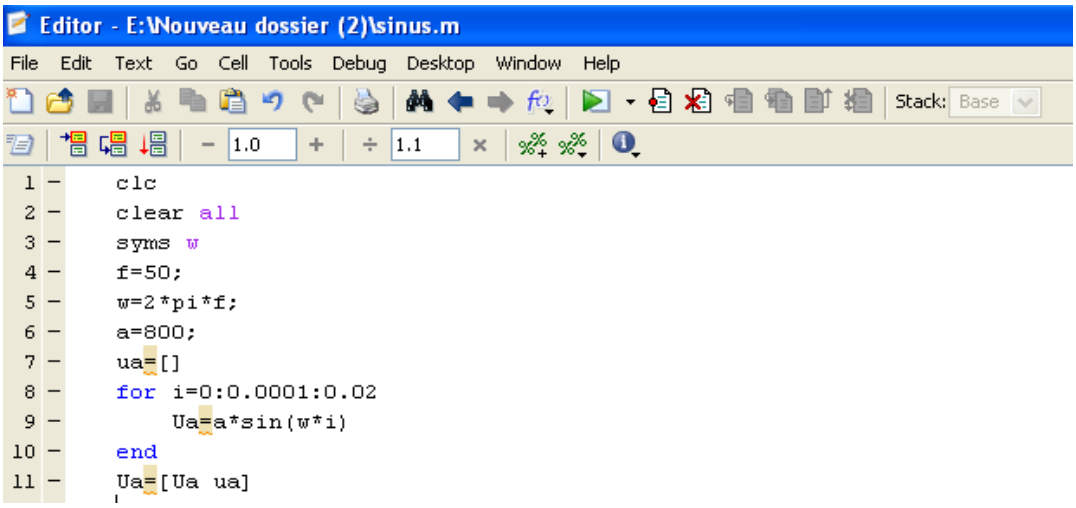


Figure III.3 : Génération de signal sinusoïdal échantillonné en utilisant Matlab.

En effet on utilisant une période d'échantillonnage ($\boldsymbol{te = 10^{-4}\ s}$), on obtient 200 points pour un signal de fréquence f=50Hz (avec une période de 0.02s)) et 400 points pour un signal de f=25Hz (avec une période de 0.04s).

En ce qui concerne le signal triangulaire, pour les dents de scie, on veut avoir un nombre de dents prédéfinie **N** dans chaque période de la modulante. La période de **N** dents et la période de modulante doit être égaux, mais avec une amplitude strictement supérieure a l'amplitude de signal de référence. Dans ce cas, et vue la forme des dents de scie, l'utilisation d'un compteur (haut/bas, up/down) approprié, présente une manier simple pour les générer.

Pour une efficace comparaison, il faut faire un compromis entre **N** et le nombre de points résultant de l'échantillonnage des signaux sinusoïdaux, donc si :

X : le nombre de points dans une période d'un signal sinusoïdal.

N : le nombre de dents de scie qu'on cherche a cadencé dans cette période.

On a :

$\boldsymbol{M = {}^{X}/_{N}}$ **:** le nombre de points résultants pour chaque dent de scie.

- Pour le cas d'un signal a une fréquence $\boldsymbol{f}$**=50Hz**, on a **X=200pts**, donc :

Cas de **N=06** :

$$M=X/N={}^{200}/_{6} \Rightarrow \mathbf{M \approx 33,}$$

Cas de **N=08** :

$$M=\frac{X}{N}=\frac{200}{8}\Rightarrow M\approx 25,$$

Cas de **N=09** :

$$M=\frac{X}{N}=\frac{200}{9}\Rightarrow M\approx 21,$$

Cas de **N=12** :

$$M=\frac{X}{N}=\frac{200}{12}\Rightarrow M\approx 17,$$

- Pour le cas d'un signal a une fréquence f**=25Hz**, on a **X=400pts**, donc :

Cas de **N=06** :

$$M=\frac{X}{N}=\frac{400}{6}\Rightarrow M\approx 62,$$

Cas de **N=08** :

$$M=\frac{X}{N}=\frac{400}{8}\Rightarrow M\approx 48,$$

Cas de **N=09** :

$$M=\frac{X}{N}=\frac{400}{9}\Rightarrow M\approx 44,$$

Cas de **N=12** :

$$M=\frac{X}{N}=\frac{400}{12}\Rightarrow M\approx 32,$$

III.3.2. La Description VHDL

Apres la création d'un nouveau projet et la sélection des propriétés de la carte FPGA, il faut désigner les entrées/sorties de la description et ça c'est le paire indissociable entité/architecture.

- L'entité définit le nom de la description VHDL ainsi que les I/O.

```
30  entity ondl is
31      Port ( ck : in  STD_LOGIC;
32             sina : out  STD_LOGIC_VECTOR (10 downto 0);
33             dac_mosia : out  STD_LOGIC;
34             dac_mosib : out  STD_LOGIC;
35             dac_mosic : out  STD_LOGIC);
36  end ondl;
```

Figure III.4 : l'entité d'une description VHDl.

- l'architecture décrit le fonctionnement du système, les relations entre les entrées et les sorties.

```
architecture <arch_name> of <entity_name> is
   -- declarative_items (signal declarations, component declarations, etc.)
begin
   -- architecture body
end <arch_name>;
```

Figure III.5 : architecture d'une description VHDl.

Pour la description des signaux (sinusoïdal, triangulaire) il faut générer un tableau en utilisant le type ''array'', après la synthèse, ce tableau sera traduit en une ram qui sera conçu en utilisant les ressources de la carte FPGA.

```
38  architecture mli25_6 of ondl is
39
40  type ram_sina is array (0 to 399) of integer range -800 to 800;
41  type ram_sinb is array (0 to 399) of integer range -800 to 800;
42  type ram_sinc is array (0 to 399) of integer range -800 to 800;
43  type ram_tri is array (0 to 62) of integer range -1000 to 1000;
44
45  signal ram:ram_sina;-- ram pour les valeurs de ua.
46  signal ramb:ram_sinb;-- ram pour les valeurs de ub.
47  signal ramc:ram_sinc;-- ram pour les valeurs de uc.
48  signal ram_t:ram_tri;-- ram pour les valeurs de porteuse.
```

Figure III.6 Principe de création des ram pou stocké les signaux nécessaire

Apres la déclaration, les Ram sont remplies en utilisant les valeurs des signaux obtenus par l'échantillonnage.

Pour la phase de comparaison, on fait la lecture des Ram, chaque point de signal sinusoïdale (p_s) est comparer a une point du signal triangulaire (p_t) .

- si $p_s > p_t$ $\Longrightarrow$ le signal de comparaison (impulsions : impa, impb, impc) prend la valeur ''**0**''.
- si $p_t \geq p_s$ $\Longrightarrow$ signal de comparaison (impulsions : impa, impb, impc) prend la valeur ''**1**''

Étant données que le protocole de communication en utilisant le convertisseur numérique analogique (CAN) de l'FPGA utilisé nécessite 32 bit, le signal d'impulsion résultant et organisé sur 32bit (suivant la forme exigée par le protocole précédent).

III.3.1.Convertisseur numériques analogique CNA

La carte **FPGA Spartan®-3E** comprend quatre séries de canaux convertisseur numériques analogique(**CNA**). Le dispositif de **CNA** est une technologie linéaire **LTC2624** avec 12 bit de résolution non signé .Les quatre sorties de **CNA** apparaître sur la tète **(J5)** qui utilise le Digilent 6_pin Format de module périphérique

(**Fig :III.7).**

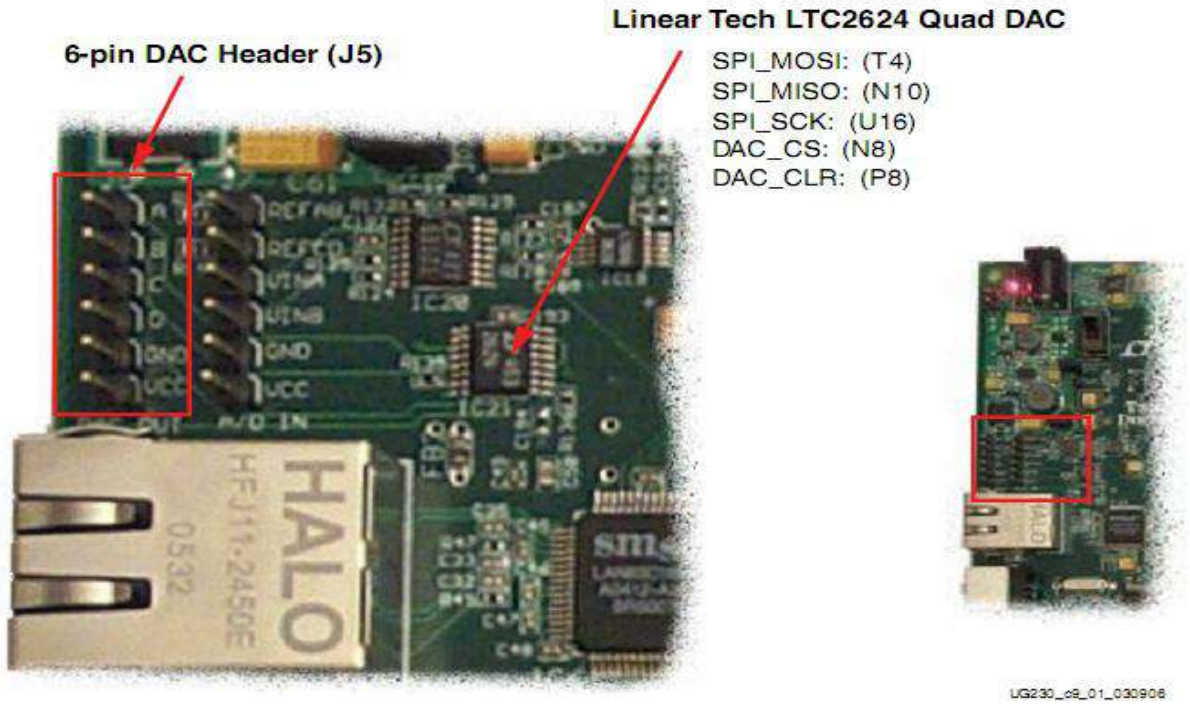


Figure III.7 : Convertisseur numérique-analogique et la tête associé.

Pour le protocole de communication nécessaire pour l'interface avec le **CNA**, on a le 24-bit et le 32-bit, pour un **SPARTAN 3E**, on trouve celui de 32 Bits.

Dans ce cas, a l'intérieur du convertisseur numérique / analogique, l'interface **SPI** est formé par un registre à décalage de 32 bits. Chaque 32-bit mot de commande est composé d'une commande « 0011 », une adresse (canal choisi), suivie par des valeurs de données. Pour le reste des bits, sont des bites nulles (Pour séparer les commandes envoyées) (fig : III.8).

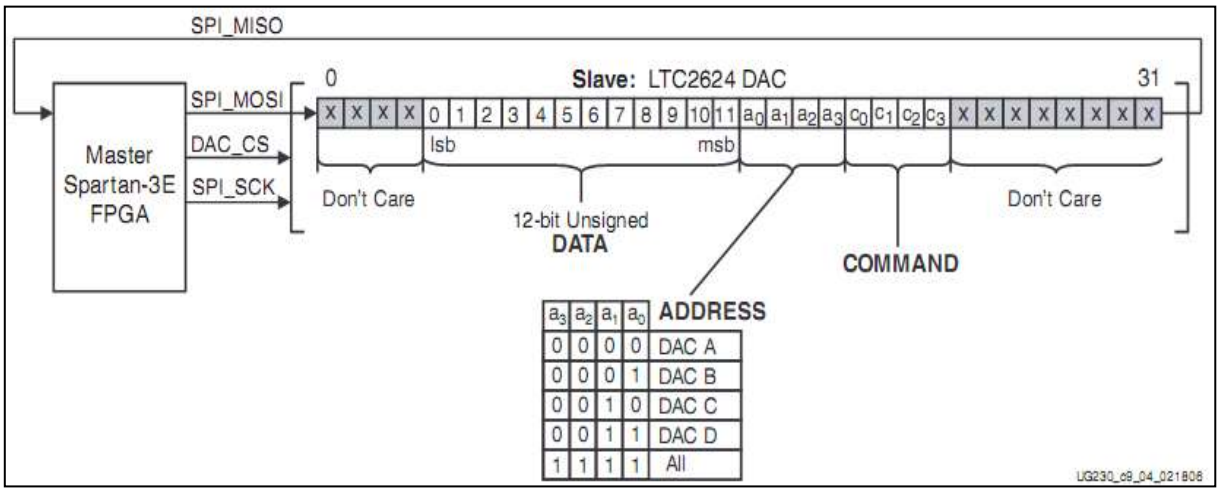


Figure III.8 : Protocole de communication dans un **CNA.**

Notre travail (implémentation de la technique MLI sur FPGA) passe par les étapes suivantes :

- Génération des signaux sinusoïdaux et triangulaires (stockage et lecture des valeurs échantillonnées).
- Faire une comparaison entre les signaux générer.
- Validation des résultats de comparaison (comparer les résultats de simulateur ISE avec les résultats des Matlab/Simulink).
- L'adaptation des signaux avec le protocole de communication du CNA.
- L'implémentation de la conception faite sur un circuit FPGA en utilisant le fichier de configuration ***BitStream*** généré lors de l'étape de synthèse.
- L'utilisation de l'analyseur ChipScope pour visualiser et analyser les signaux réels qui seront générées par le circuit FPGA ultérieurement pour une application expérimentale (connexion avec un onduleur réel).

III.4. Synthèse et simulation de la conception réalisée pour la MLI

La conception du circuit met en œuvre quatre outils : un éditeur de texte, un simulateur, un synthétiseur et un placeur-routeur. La description en VHDL ne concerne que le circuit de commande et rien de ce qui constitue l'environnement du circuit. Le modèle VHDL est inspiré de l'architecture proposée ci-dessus. Par la suite, nous allons réaliser une simulation manuelle en appliquant des valeurs sur les entrées et en vérifiant visuellement l'état des sorties.

III.4.1. Génération des signaux de comparaison (sinusoïdale et triangulaire)

Après le chronogramme de la simulation, on remarque que la période d'horloge est de **100 µs (**pour toute la durée de simulation), alors durant chaque période d'horloge on va lire une valeur (sur 11 bits) à partir d'une **RAM** bien déterminé précédemment (au début d'architecture), afin de crée un signal sinusoïdale par VHDl (Fig.). De même façon, le signal triangulaire est crée (Fig.II.9).

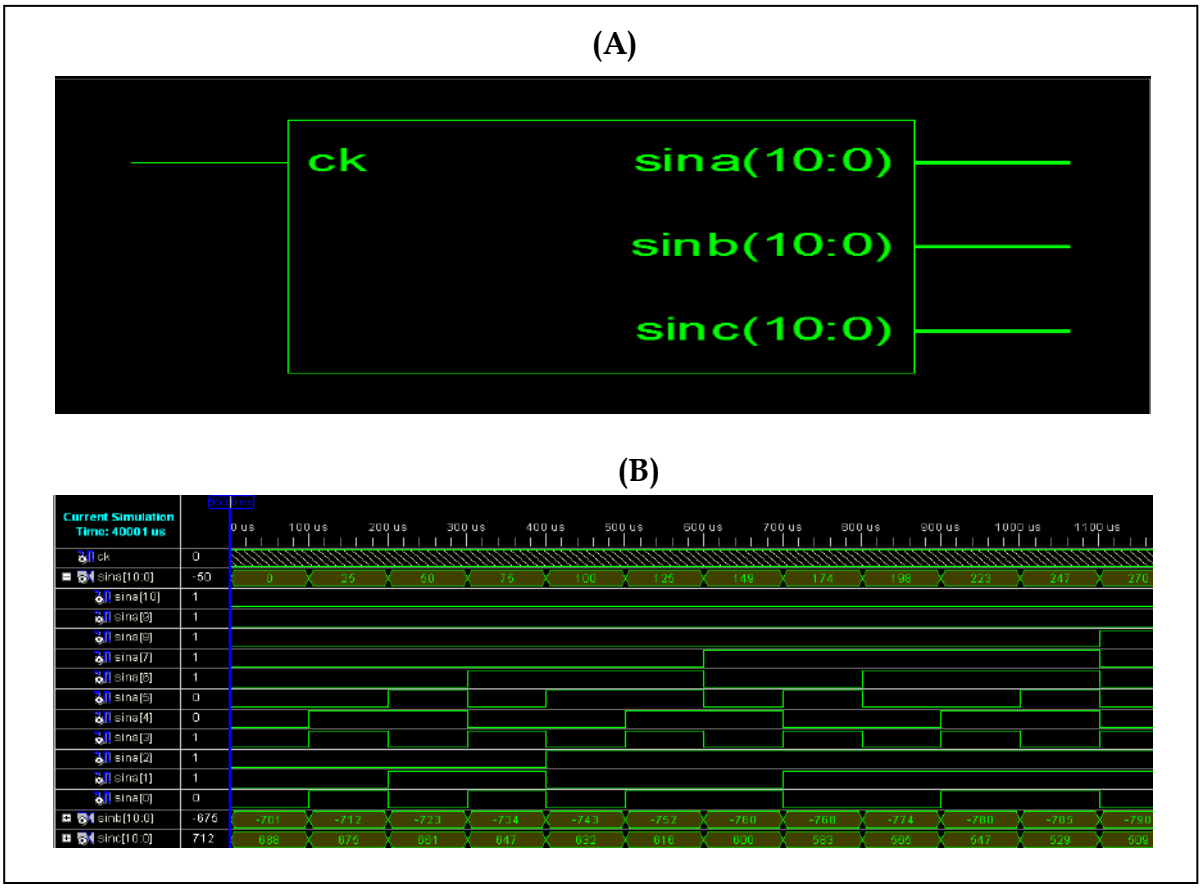


Figure III.9 : génération des signaux sinusoïdaux, A : résultat de synthèse (RTL), B : résultats de simulation.

(A)



(B)

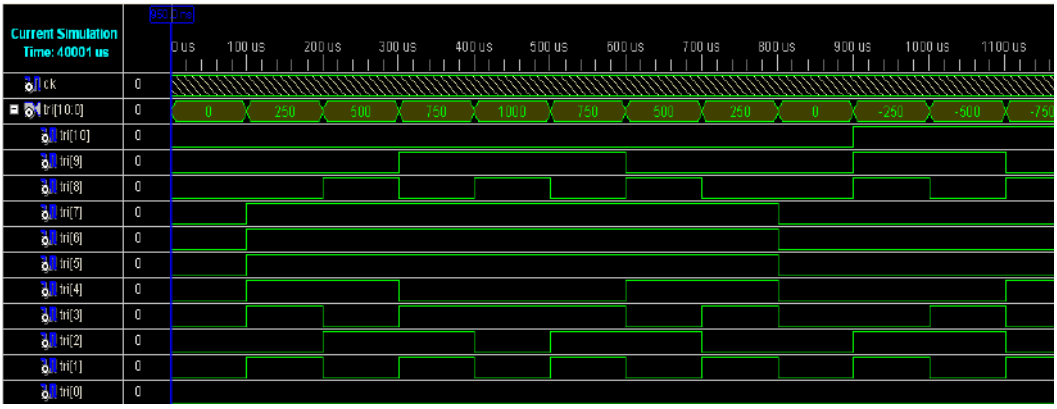


Figure III.10 : génération du signal triangulaire, A : résultat de synthèse (RTL), B : résultats de simulation

❖ **Comparaison de la modulante avec la porteuse**

Dans cette étape, on va faire une comparaison entre la modulante et le signal triangulaire, pour générer les impulsions MLI. Le résultat de la synthèse est représenté sur figure (III.11) :

Les chronogrammes de la figure (III.11) résument les résultats obtenus par simulation HDL, d'après ces chronogrammes on remarque que les résultats prennent soit le **1** soit le **0** :

- ✓ Si *la porteuse* ≥ *signal sin* ⟶ le signal de comparaison prend'0'.
- ✓ Si *la porteuse* < *signal sin* ⟶ le signal de comparaison prend'1'.

Pour validé les résultats du simulateur ISE, on les a comparé avec ceux obtenu par MATLAB/SIMULINK. La comparaison permet de confirmer que les résultats des deux simulations sont identiques à part légère différence due aux problèmes de résolution des signaux (pour les signaux MATLAB c'est du double précision, tandis que pour l'ISE XILINX c'est du point fixe) (fig III.12).

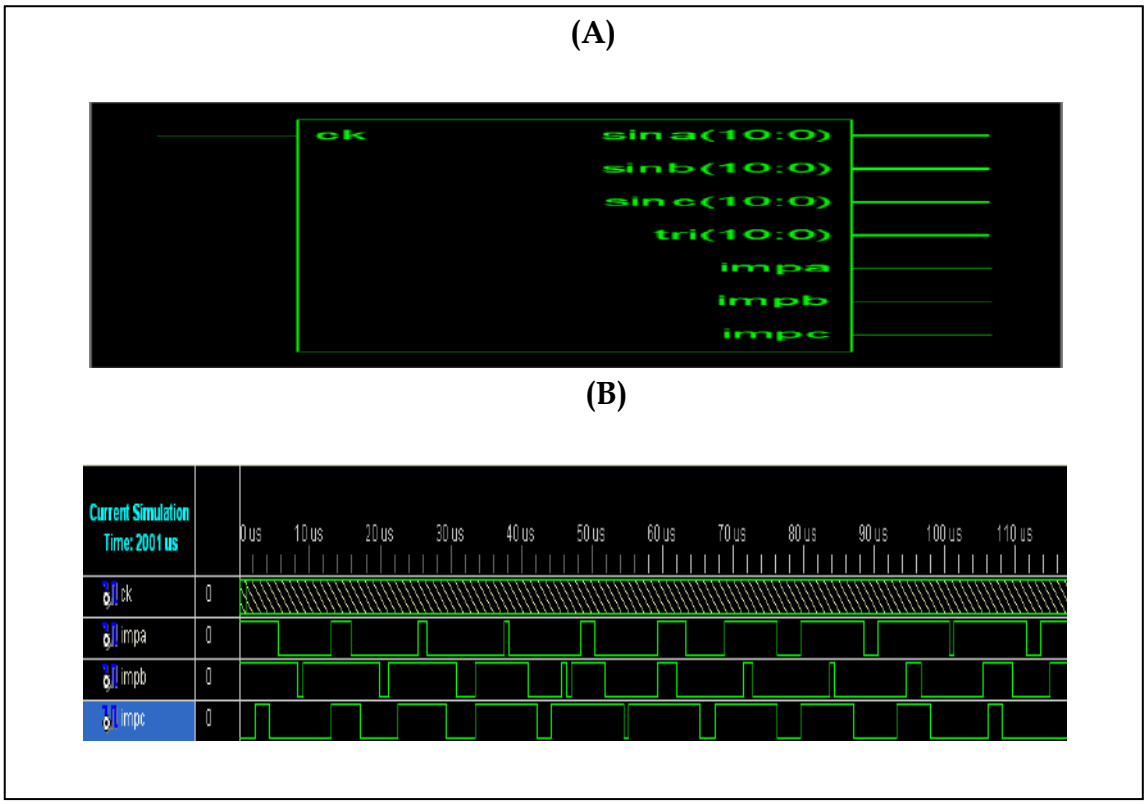


Figure III.11 : Comparaison des signaux, A : résultat de synthèse (RTL), B : résultats de simulation

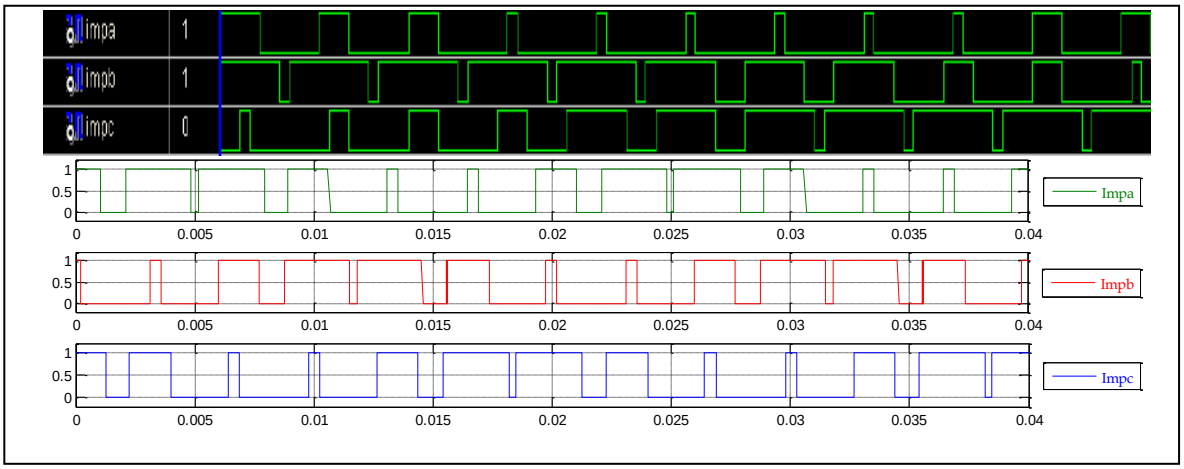


Figure III.12 : Résultats de la comparaison.

III.4.2. Synthèse du bloc générale de la stratégie MLI

Après la validation de la conception par simulation vient l'étape la réalisation du bloc général de la MLI tout en considérant le protocole de la conversion numérique analogique(CAN). La résume l'image du circuit résultant de la conception faite, ce circuit sera utilisée pour le reste du travail.

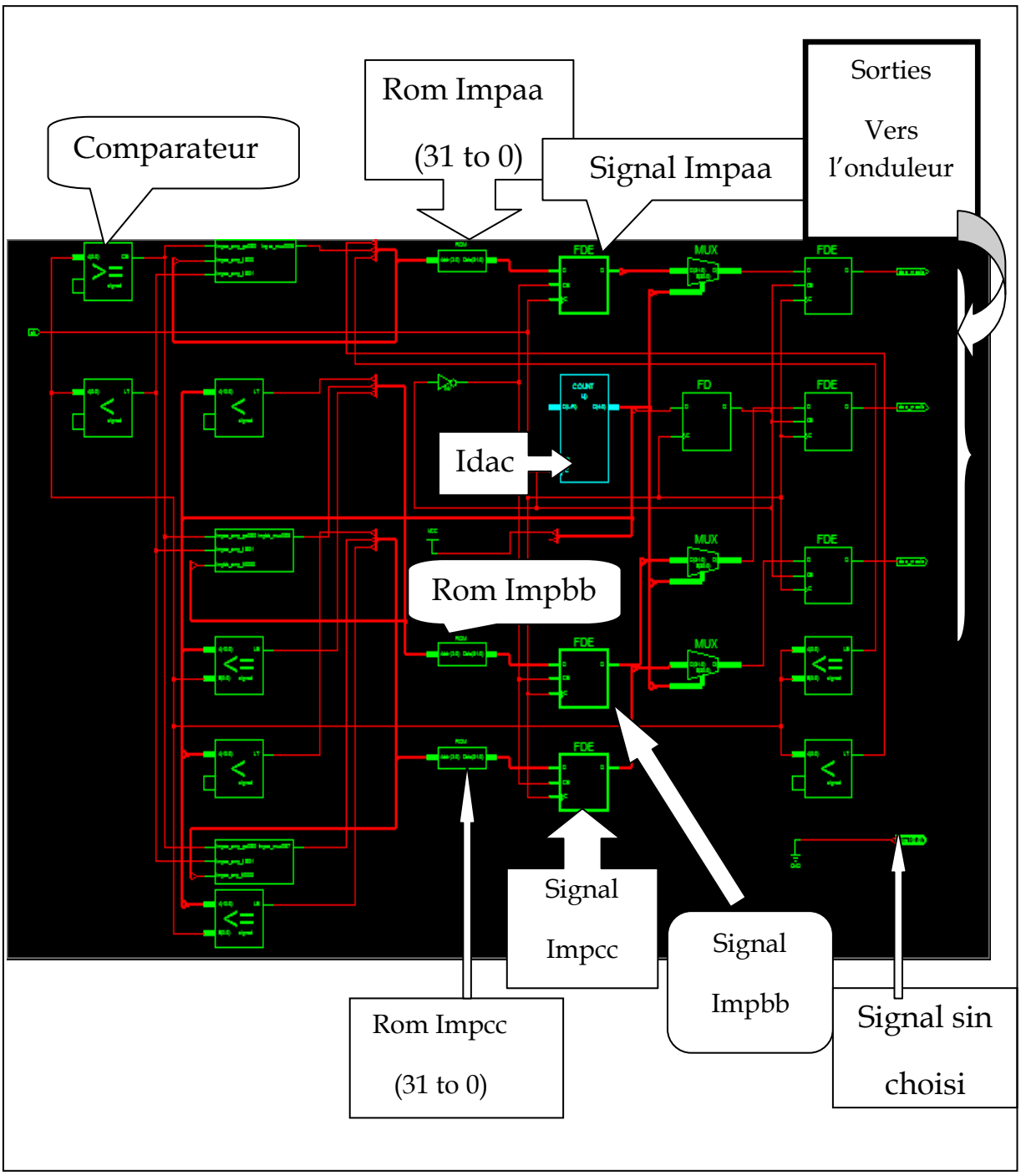


Figure III.13: Vue interne (RTL) du MLI.

❖ La Simulation

Comme montre les figure (III.15, 16), le résultat présente les trois sorties de l'onduleur (chaque point comparée, elle prend les 12 Bits de données), on remarque que les 32 bits sont répéter à cause de la période d'horloge utilisé (**100 µs**), on va assurer la répétition d'envois de ces 32 bits au cours de (100 **µs**), pour que l'information sera reçu. On remarque que le nombre de dents scie est variée (6, 8, 9,12), L'avantage de cette variation est de rendre le système plus précis.

- **Résultats de la simulation pour f=50 Hz :**

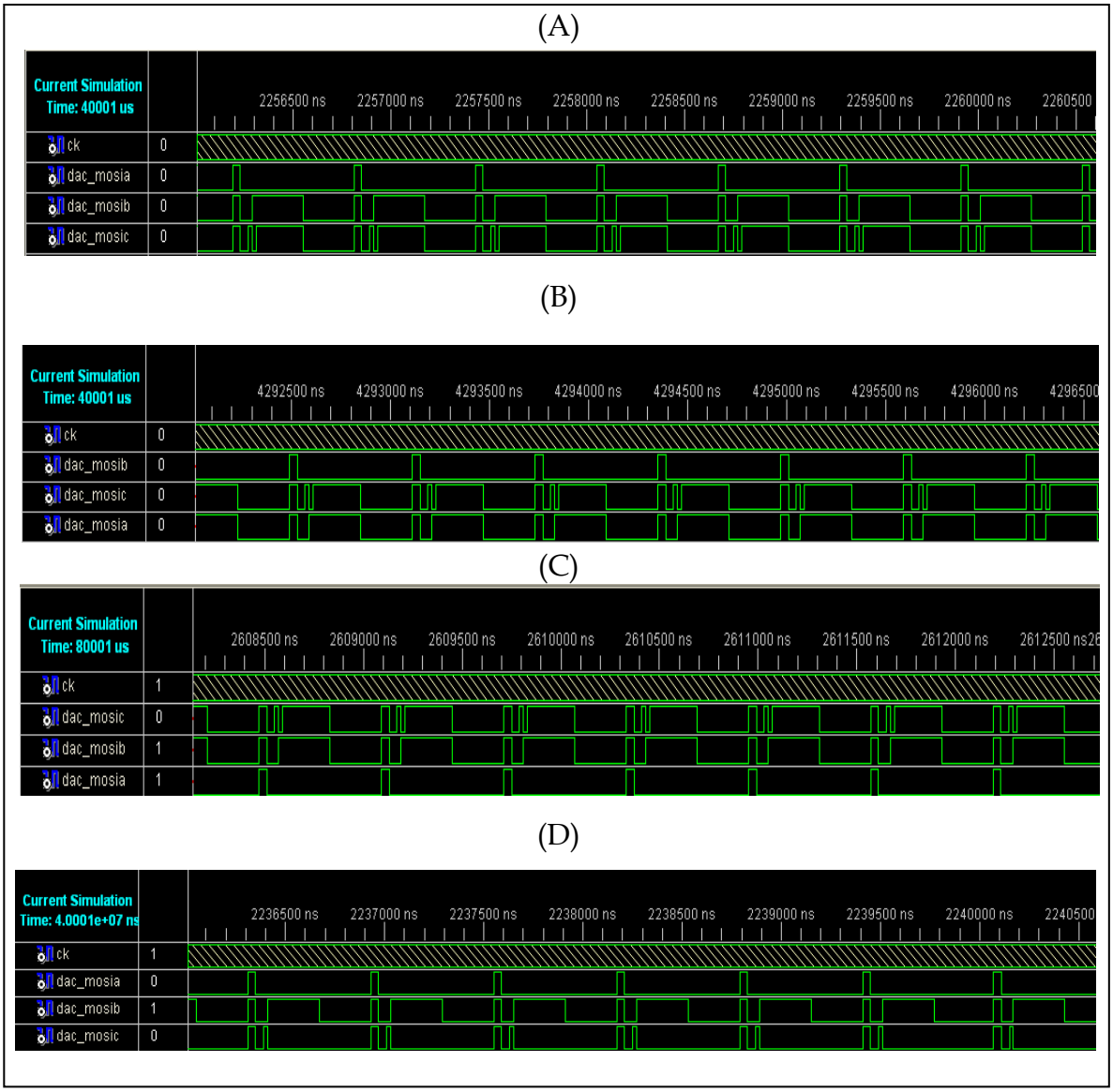
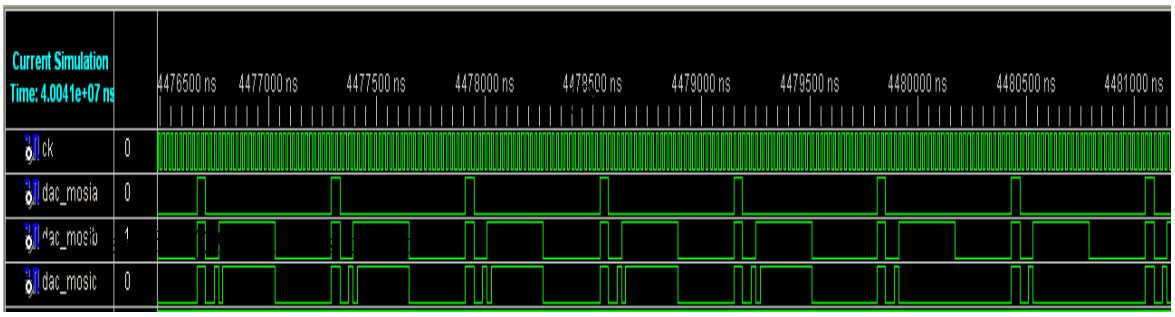


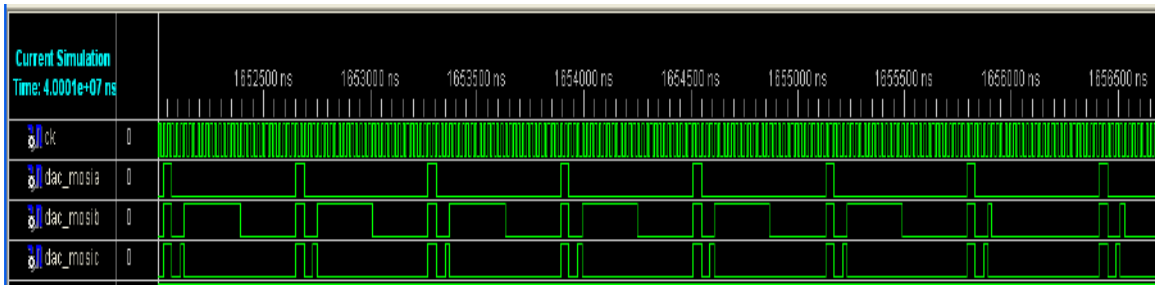
Figure III. 14: Résultats de simulation du bloc générale de la stratégie MLI (50 Hz).A : 6 dents, B : 8 dents, C : 9 dents, D : 12 dents.

- **Résultats de la simulation pour f=25 Hz :**

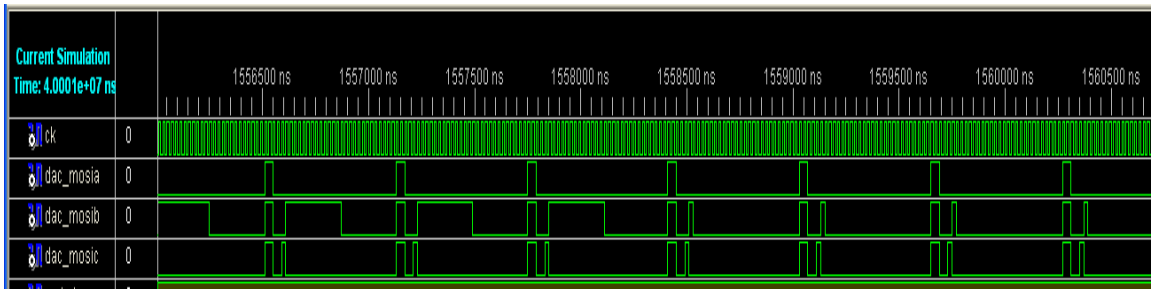
(A)



(B)



(C)



(D)

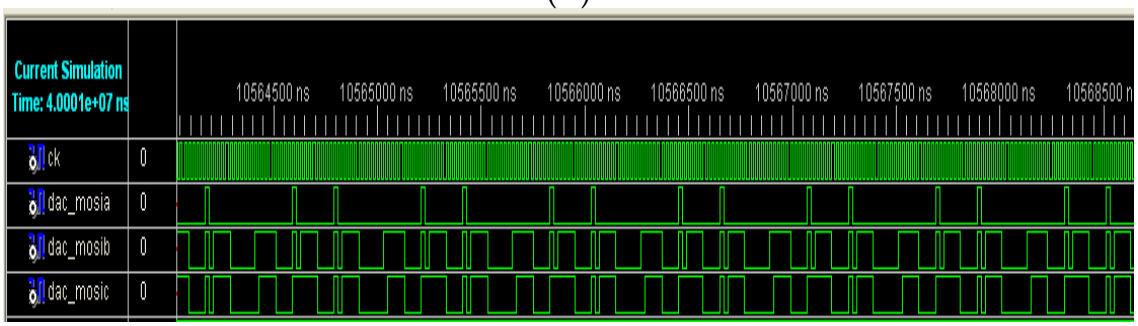


Figure III.15 : Résultats de simulation du bloc générale de la stratégie MLI (25 Hz).

.A : 6 dents, B : 8 dents, C : 9 dents, D :12 dents.

III.5. L'implémentation de la description de la stratégie MLI

Apres la partie de simulation, on passe a l'étape d'implémentation, cette dernière est une étape de programmation physique et de teste électrique consiste à mettre en œuvre l'algorithme sur l'architecture de circuit configurable ciblé, c'est-à-dire à compiler, charger, puis lancer l'exécution sur l'ordinateur, mais avons ca, il faut crée un fichier «. Cdc » sur l'analyseur ChipScope qu'on a déjà parlé au début de ce chapitre. Dans notre travail, on va implémenter les deux descriptions VHDL qui concerne deux fréquences (**50 Hz** et **25Hz)** pour chaque cas de porteuse **(6, 8, 9, 12** dents de scies**)** .

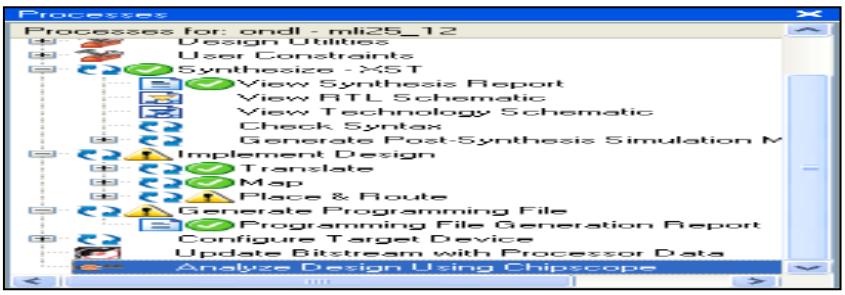


Figure III.16: L'implémentation de la description.

Cette étape est la dernière dans le processus de conception en général et en particulier de notre travail. Elle va nous permettre de charger le fichier BIT STREAM de la description VHDL sur un support FPGA choisi convenable. Le transfert de la solution finale peut être assuré par un câble JTAG qui relie le micro-ordinateur avec la carte FPGA.

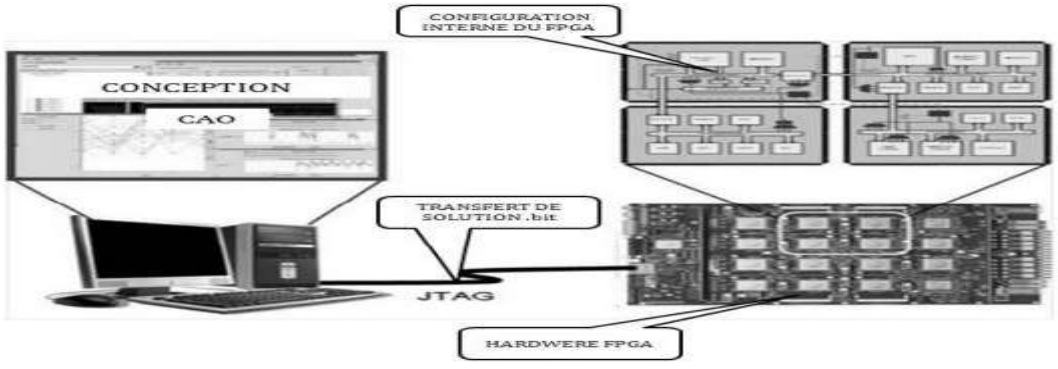


Figure III.17 : Configuration de la carte FPGA par un câble JTAG.

III.5.1. Résultats de l'implémentation pour f=50 Hz

Après l'implémentation de notre description sur la carte FPGA, nous observons les résultats obtenus sur l'analyseur ChipScope (hardware), par la suite on fait un comparaison entre les résultats obtenus avec les résultats de simulateur ISE (software). D'après les chronogrammes de la figure (III.20, 22, 24, 26), on remarque que les résultats sont presque égaux. Quand on augmente le nombre de dents de scie, on remarque que le nombre des points de comparaison augmente, donc les résultats déviants plus précis.

- **Cas d'une porteuse avec 6 dents de scies :**

A. Résultats sur l'analyseur ChipScope :

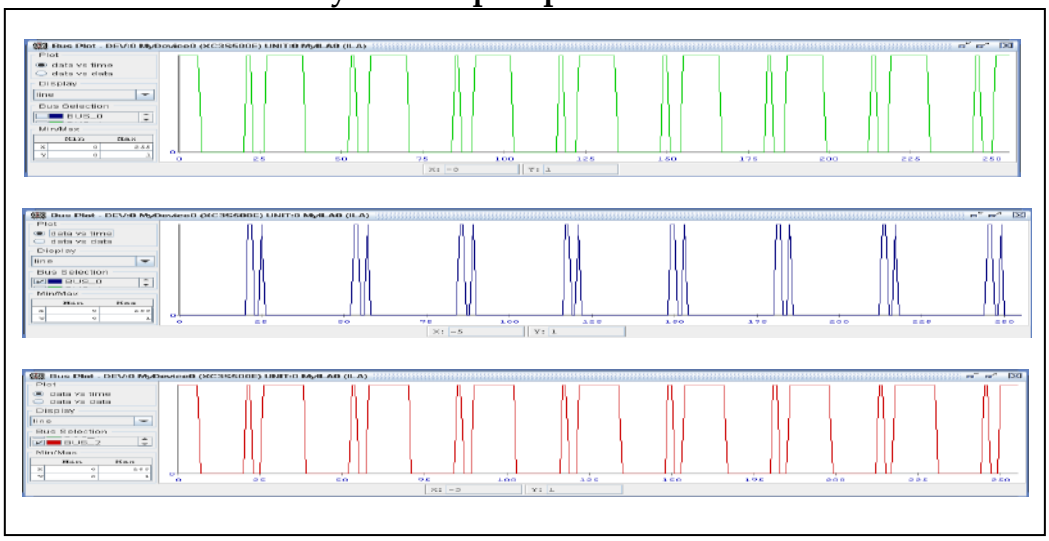


Figure III.18 : Résultats de CNA sur l'analyseur ChipScope (6 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

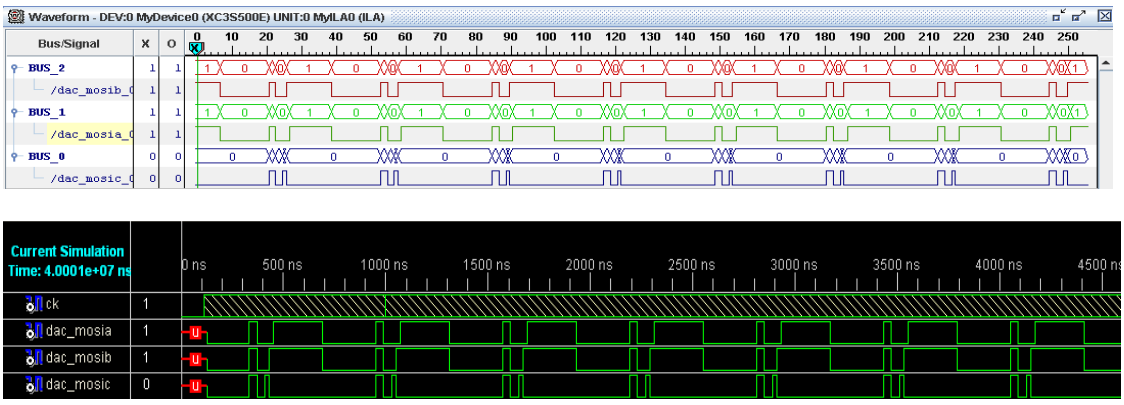


Figure III.19 : Comparaisons des résultats (6 dents de scie).

❖ **Cas d'une porteuse avec 8dents de scies :**

A. Résultats sur l'analyseur ChipScope :

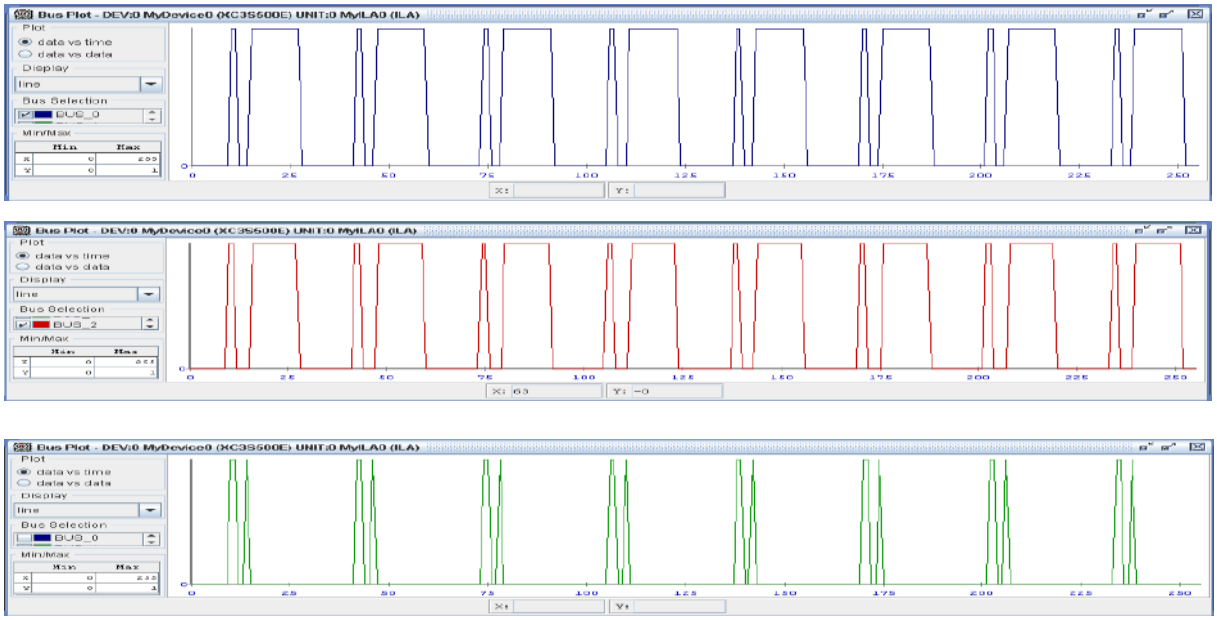


Figure III.20 : Résultats de CNA sur l'analyseur ChipScope (8 dents).

❖ **Comparaison de ChipScope avec le simulateur ISE:**

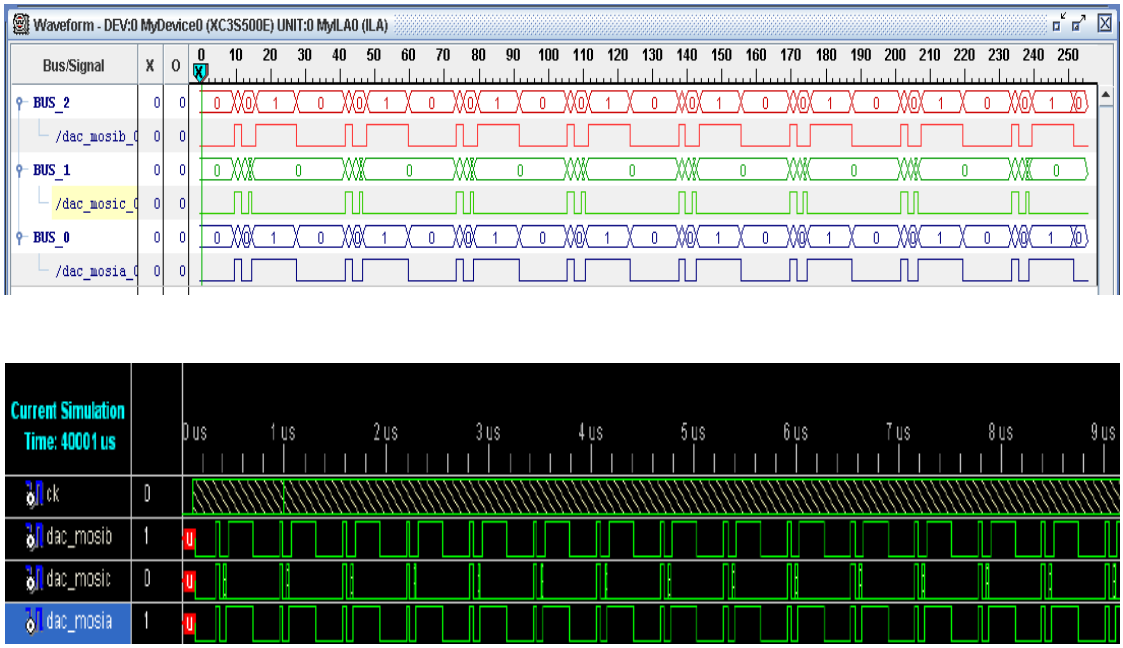


Figure III.21 : Comparaisons des résultats (8 dents de scie).

❖ **Cas d'une porteuse avec 9dents de scies :**

A. Résultats sur l'analyseur ChipScope :

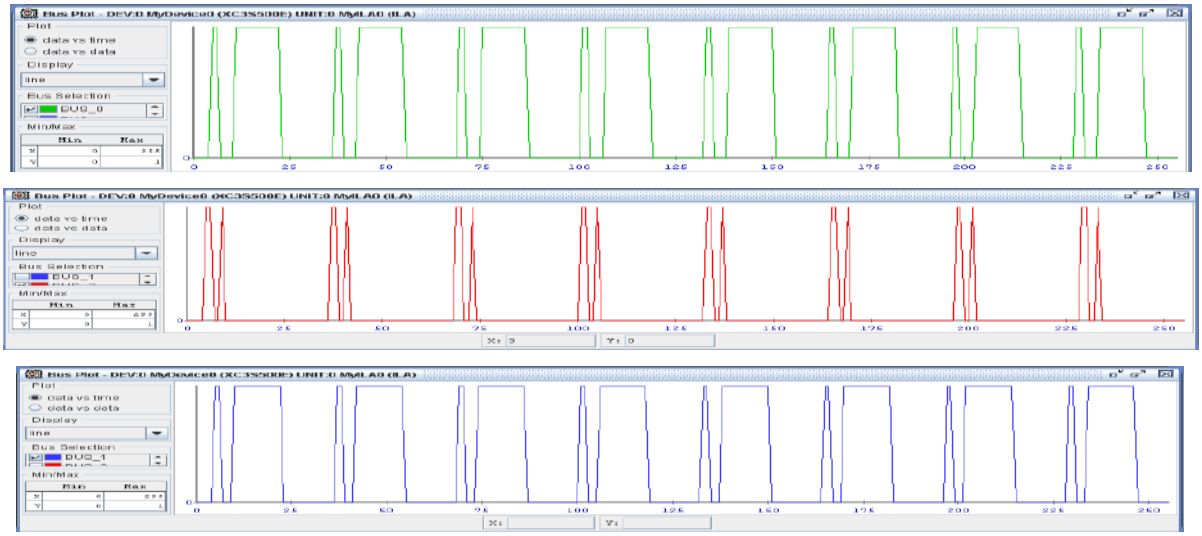


Figure III.22 : Résultats de CNA sur l'analyseur ChipScope (9 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

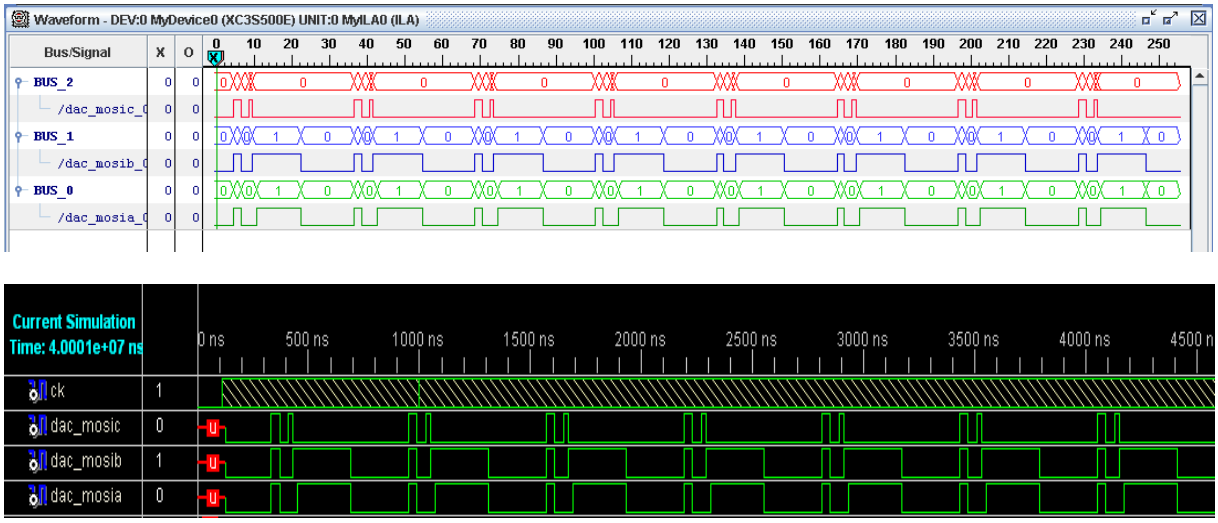


Figure III.23 : Comparaisons des résultats (9 dents de scie).

❖ **Cas d'une porteuse avec 12dents de scies :**

A. Résultats sur l'analyseur ChipScope :

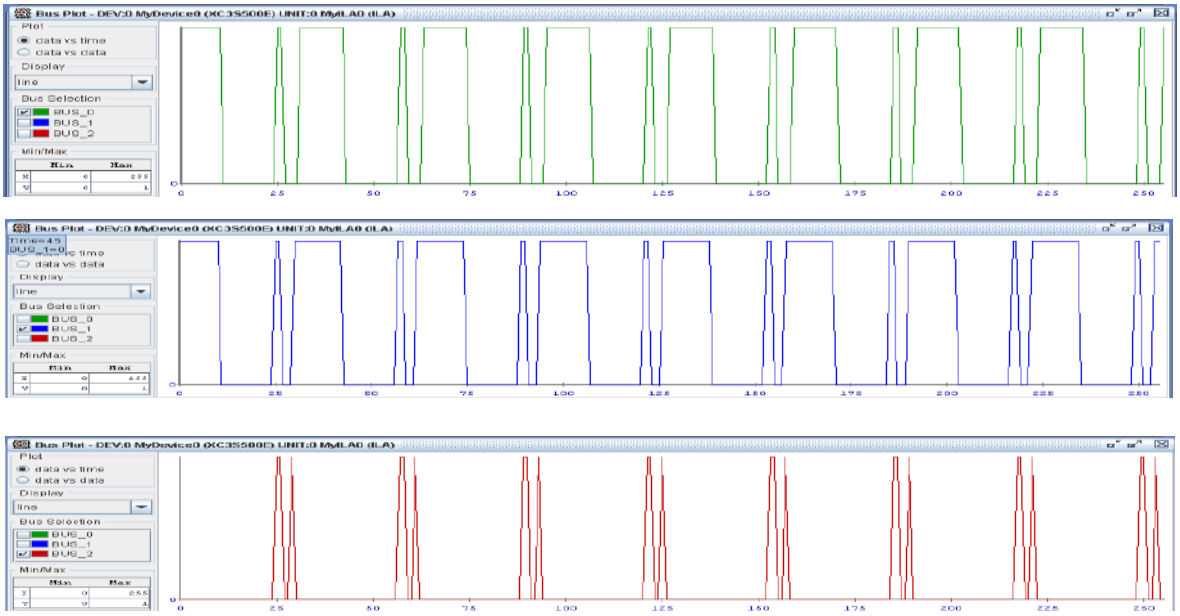


Figure III.24 : Résultats de CNA sur l'analyseur ChipScope (12 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

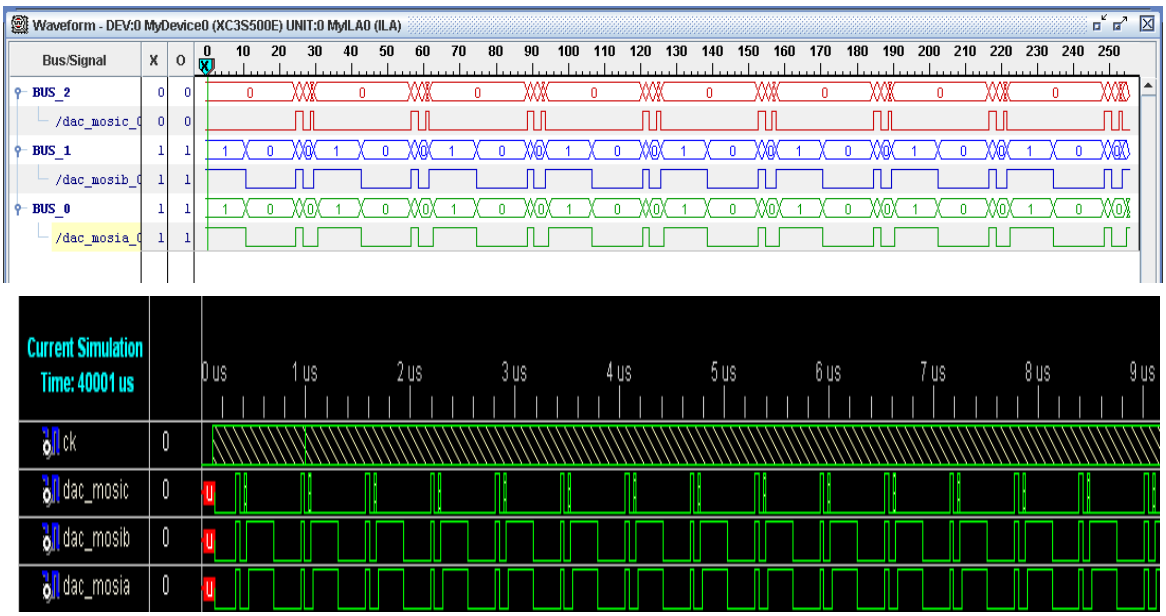


Figure III.25 : Comparaisons des résultats (12 dents de scie).

III.5.2. Résultats de l'implémentation pour f=25Hz

Après l'implémentation de notre description sur la carte FPGA, nous observons les résultats obtenus sur l'analyseur ChipScope (hardware), par la suite on fait un comparaison entre les résultats obtenus avec les résultats de simulateur ISE (software). D'après les chronogrammes de la figure (III.28, 30, 32, 34), on remarque que les résultats sont presque égaux. Quand on augmente le nombre de dents de scie, on remarque que le nombre des points de comparaison augmente, donc les résultats déviants plus précis.

- **Cas d'une porteuse avec 6 dents de scies**

A. Résultats sur l'analyseur ChipScope

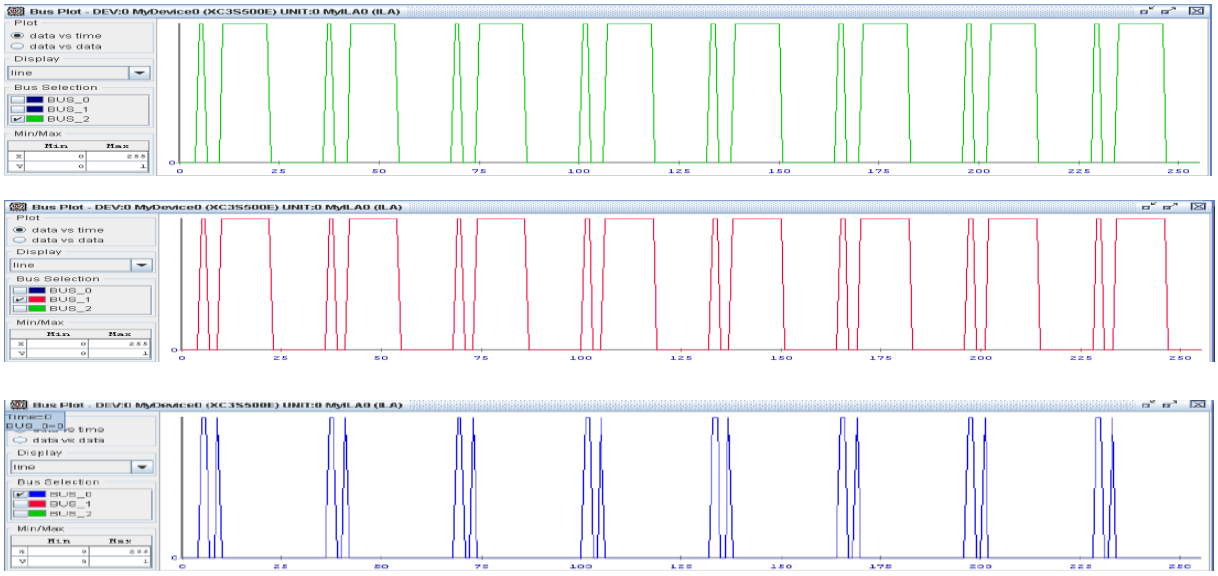


Figure III.26 : Résultats de CNA sur l'analyseur ChipScope (6 dents).

B. Comparaison de ChipScope avec le simulateur ISE

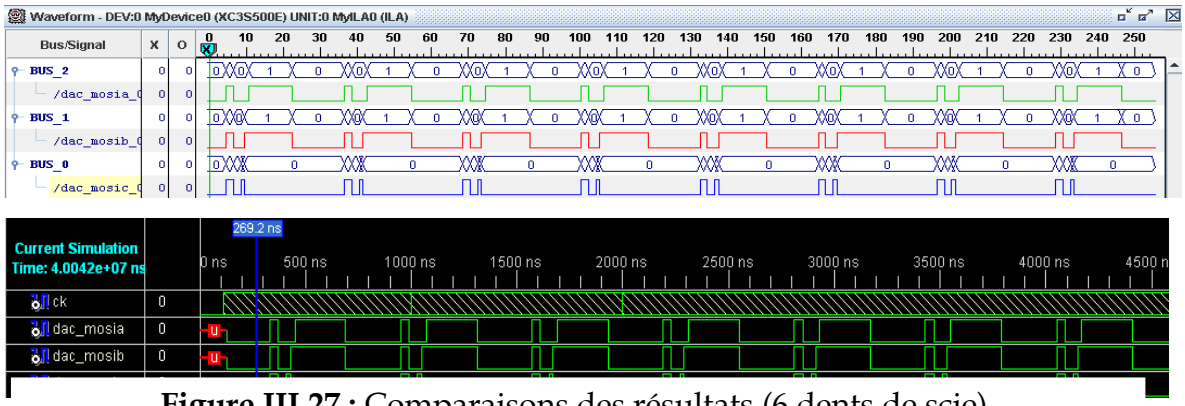


Figure III.27 : Comparaisons des résultats (6 dents de scie).

❖ **Cas d'une porteuse avec 8 dents de scies :**

A. Résultats sur l'analyseur ChipScope :

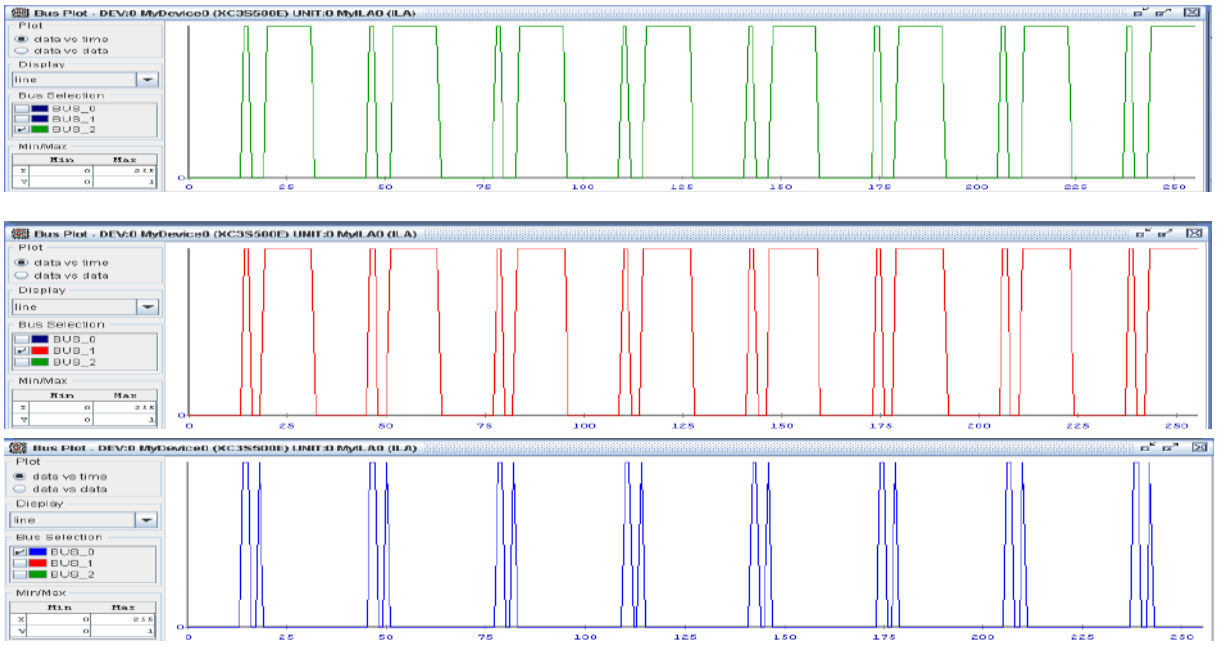


Figure III.28 : Résultats de CNA sur l'analyseur ChipScope (8 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

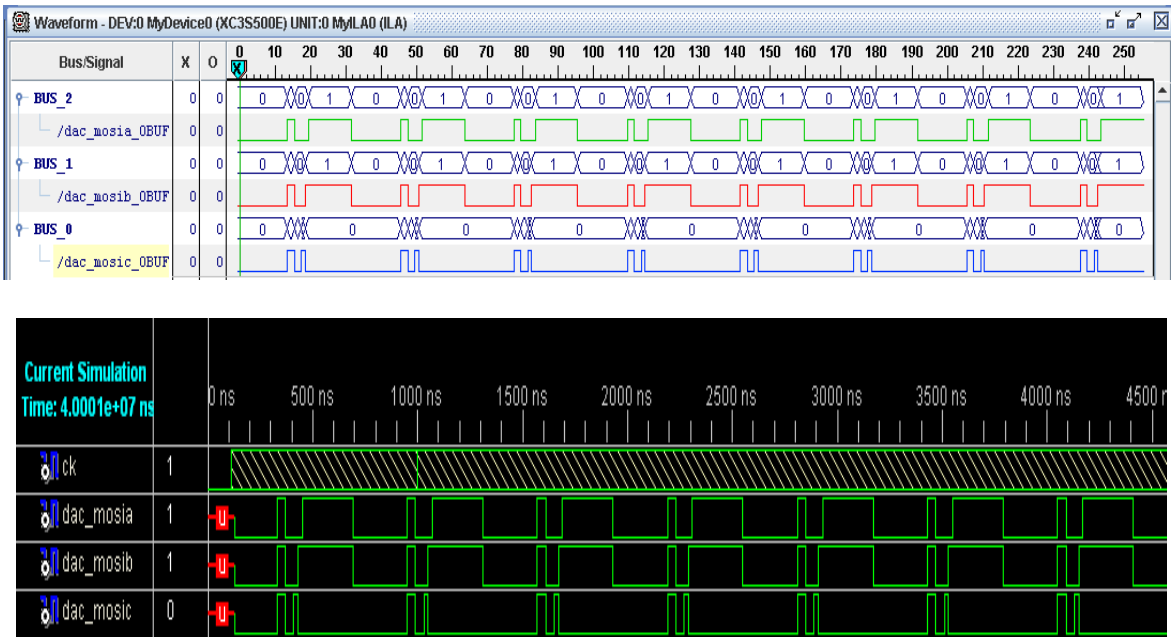


Figure III.29 : Comparaisons des résultats (8 dents de scie).

❖ **Cas d'une porteuse avec 9 dents de scies :**

A. Résultats sur l'analyseur ChipScope :

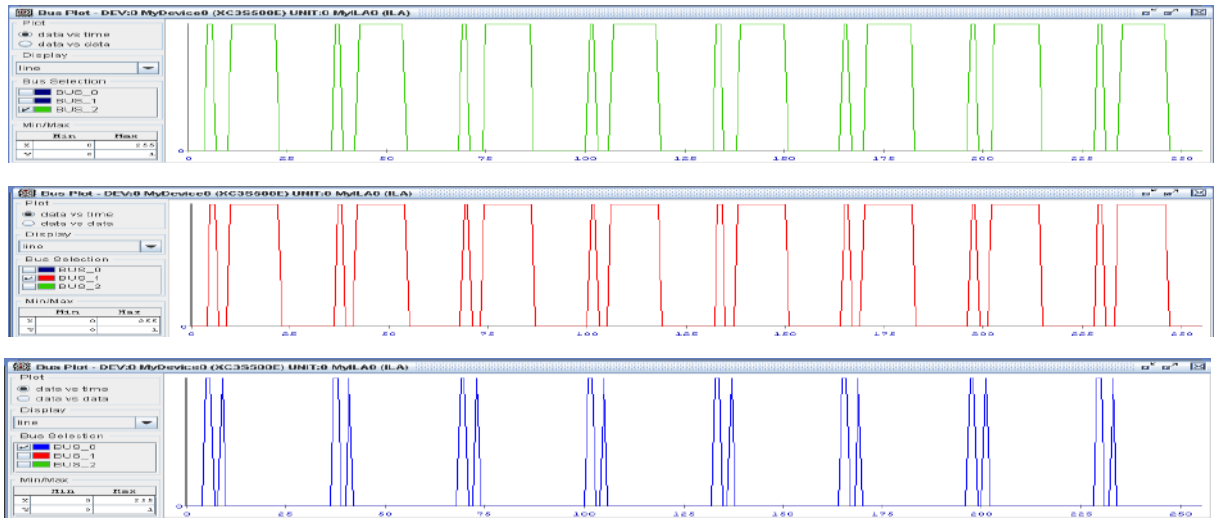


Figure III.30 : Résultats de CNA sur l'analyseur ChipScope (9 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

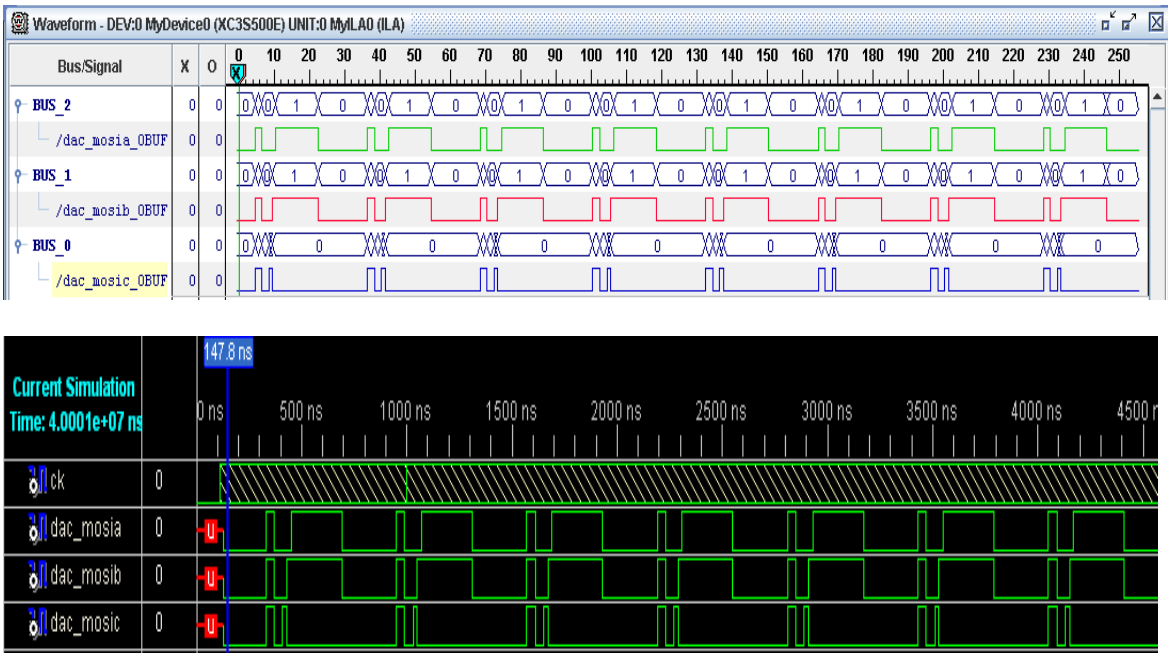


Figure III.31 : comparaisons des résultats.

❖ **Cas d'une porteuse avec 12 dents de scies :**

A. Résultats sur l'analyseur ChipScope :

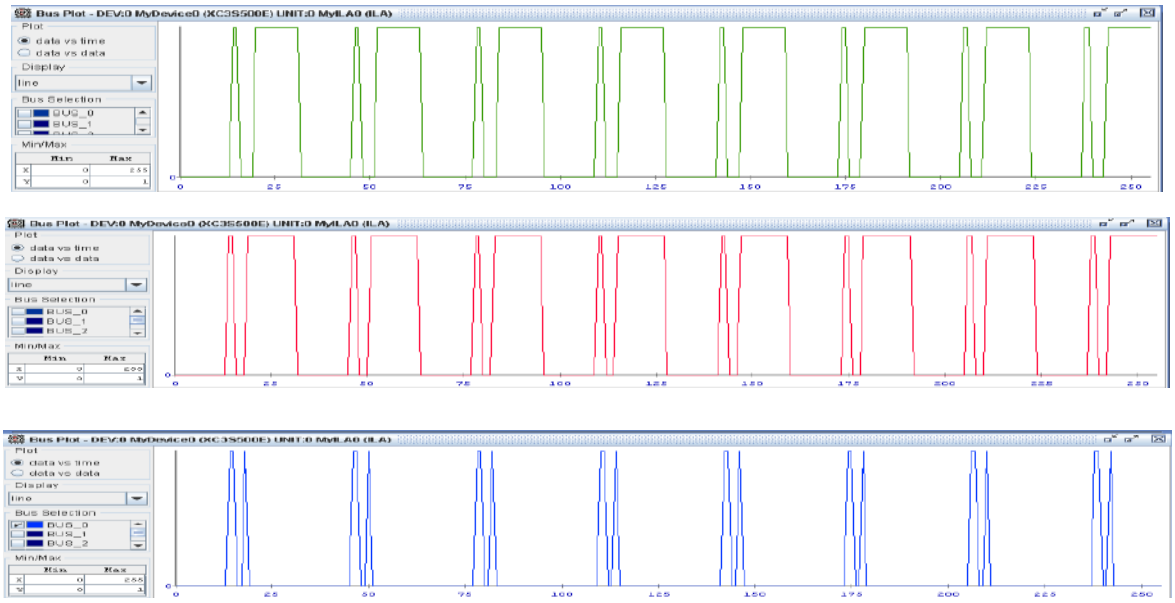


Figure III.32 : Résultats de CNA sur l'analyseur ChipScope (12 dents).

B. Comparaison de ChipScope avec le simulateur ISE:

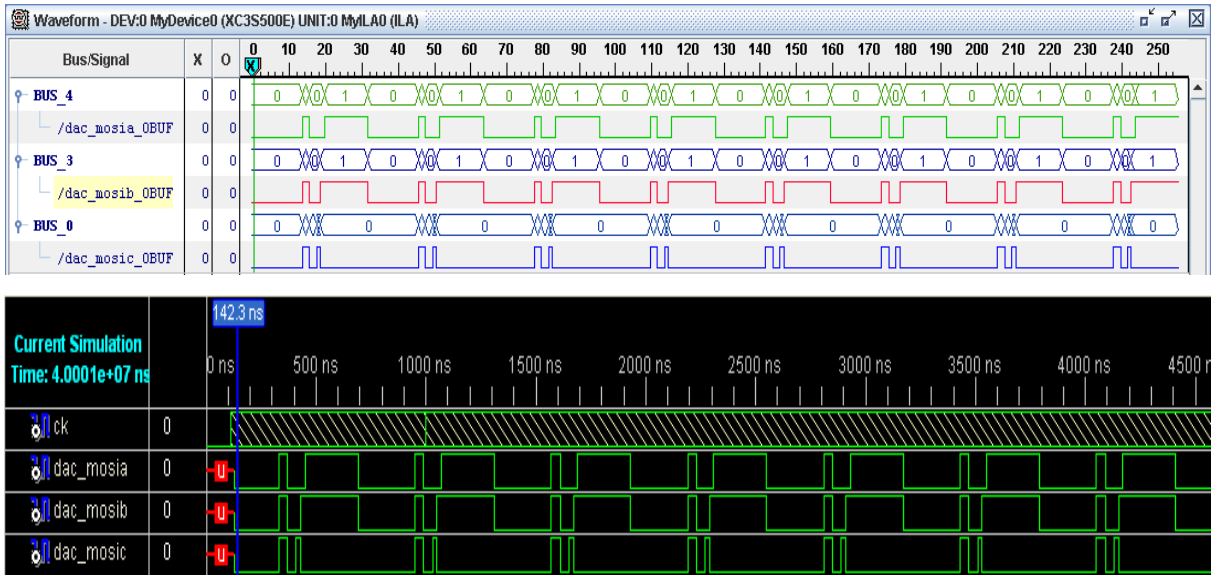


Figure III.33 : Comparaison des résultats (12 dents).

III.6. Rapport de consommation des ressources

Le tableau ci-dessus (Tab : III.1), résume les ressources totales d'FPGA, ainsi que les ressources consommée par notre circuit et les pourcentages d'utilisation (FLIP-FLOP, Luts, IOBs, etc.).

Cette consommation n'a pas une relation avec la description seulement mais l'analyseur ChipScope prend un partie de ces ressources consommées. Puis que le temps de développement est très faible on peut l'ajoute des autres commandes.

mli25_6 Partition Summary	[-]
No partition information was found.	

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	247	9,312	2%	
Number of 4 input LUTs	255	9,312	2%	
Logic Distribution				
Number of occupied Slices	273	4,656	5%	
Number of Slices containing only related logic	273	273	100%	
Number of Slices containing unrelated logic	0	273	0%	
Total Number of 4 input LUTs	316	9,312	3%	
Number used as logic	183			
Number used as a route-thru	61			
Number used as Shift registers	72			
Number of bonded IOBs	4	232	1%	
Number of RAMB16s	4	20	20%	
Number of BUFGMUXs	2	24	8%	
Number of BSCANs	1	1	100%	
Number of RPM macros	12			

Tableau III.1 : Ressources consommés par la stratégie MLI Intersective.

III.7. Conclusion

Dans ce chapitre, on réalise une stratégie avec les outils HDl (utilisation de l'éditeur ISE).

D'abord, nous avons générer les signaux sinusoïdaux et triangulaires, puis nous avons comparé les signaux sinusoïdaux avec le signal triangulaire, après ca nous validons les résultats par une comparaison avec ceux obtenus par MATLAB/SIMULINK

Ensuite, on a injecté les résultats sur les 12 bits de CNA (DAC) réservé pour la donnée, puis l'implémenter sur la carte FPGA et l'analyser par ChipScope.

On obtient des résultats de ChipScope (Hardware) presque égale aux résultats obtenus par le simulateur (software).

Pour cela, on conclure que malgré sa complexité s'effectue dans la description, la configuration des FPGA par les outils HDL est simple à réalisé, est une solution pour configurer un système très performant.

Notre travail a porté sur l'étude de la technique MLI (triangulo-sinusoïdale) et son implémentation sur un circuit logique reconfigurable qui est le FPGA, par l'utilisation de langage de description matérielle (VHDL) sous l'environnement ISE-Simulateur/Matlab.

Dans le chapitre I, nous avons présenté les circuits programmables puis nous avons étudié le type FPGA ce qui nous a permis de conclure que cette technologie (FPGA) s'inscrit au sommet de l'évolution des composants logiques et le besoin croissant de composants plus performants, plus économiques.

Dans le deuxième chapitre, nous avons faire une modélisation de la machine asynchrone triphasée ainsi que la détermination de leurs paramètres électriques et mécaniques et ainsi que la modélisation d'onduleur de tension triphasée.

Dans le dernier chapitre, on réalise une stratégie avec les outils HDl (utilisation de l'éditeur ISE), puis l'implémenter sur la carte FPGA. Notre étude est finalisée par l'implémentation de la technique MLI sur la carte FPGA.

Les résultats obtenus permis ont permis de démontrer le bon fonctionnement des blocs constituant la stratégie MLI, mis en œuvre et de saisir l'importance de programmation des circuits FPGA vue que des modifications peuvent être facilement apportées pour l'amélioration des conceptions réalisé. On peut attester donc que l'utilisation de circuits FPGA est un moyen d'améliorer les performances de contrôle, comme il permet d'apporter des gains en termes de temps de développement aussi bien qu'économique.

De point de vue perspectif l'approche traitée au cours de ce travail peut être:

- ✓ utilisé dans la chaine de commande de machines tournantes.
- ✓ Développé et adapter pour les déférents commandes (Scalaire, vectorielle, adaptatif…)

Dans cette annexe, nous avons présenté une partie contient les signaux utilisés dans la description VHDL, l'architecture de HAVARD, et de VON-NEUMANN. En fin, on présente les paramètres de la machine asynchrone.

A. Les signaux utilisés pour la description VHDL :

- **Sina, Sinb, Sinc :** signaux sinusoïdaux sur 11 bits (sorties).
- **Tri :** signal triangulaire sur 11 bits.
- **Impa, Impb, Impc :** signaux de comparaison sur un seul bit (0 ou 1).
- **Impaa, Impbb, Impcc :** signaux sur 32 bits (31 to 0).
- **Ck :** l'horloge.
- **Dac_mosia, Dac_mosib, Dac_mosic :** Sorties de l'onduleur sur un Bits.

B. L'architecture de HAVARD :

L'architecture dite de Harvard permet de transférer simultanément les données et les instructions à exécuter. Ainsi, l'unité de traitement aura accès simultanément à l'instruction et aux données associées. Ce modèle peut se montrer plus rapide à technologie identique que celui de Von Neumann. Le gain en performance s'obtient cependant au prix d'une complexité accrue de structure.

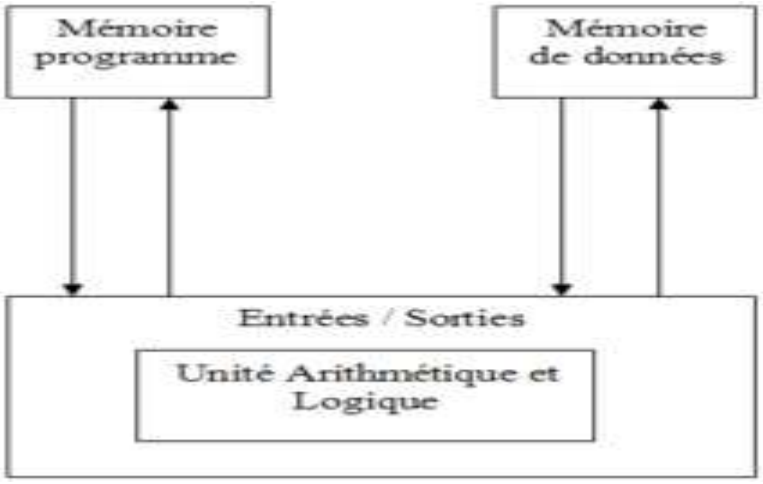
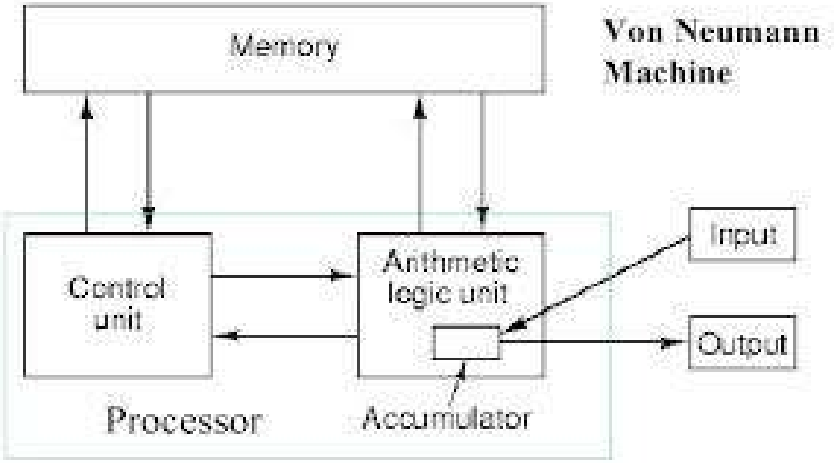


Fig : A.1 : L'architecture de HAVARD.

C. L'architecture de VON-NEUMANN :

L'architecture, dite **architecture de Von Neumann**, est un modèle pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données requises ou générées par le calcul. De telles machines sont aussi connues sous le nom d'ordinateur à programme enregistré. La séparation entre le stockage et le processeur est implicite dans ce modèle.



D. Les paramètres de la machine asynchrone :

Paramètres	Symboles	Valeur	Unité
Résistance de stator	Rs	4.850	Ohm
Résistance de rotor	Rr	3.805	Ohm
Inductance de stator	Ls	0.274	Henry
Inductance de rotor	Lr	0.274	Henry
Inductance Mutuelle	M	0.258	Henry
Moment d'inertie	J	0.031	$Kg*m^2$
Nombre de paire de pole	P	2	/
Couple résistant	Cr	0	N.m
Coefficient de Fréquence	Kf	0.001136	$N*m/rad*s^{-1}$

BIBLIOGRAPHIE

[01]: **Zahir Ait Ouali :** Thème **« Application des FPGA à la commande d'un moteur asynchrone ».** Mémoire de Magister Université de Boumerdes.

[02]: **Roger Woods, John McAllister, Gaye Light body, Ying Yi: "**FPGA-based implementation of signal processing systems", 2008.

[03] : Pong P. Chu : « FPGA PROTOTYPING BY VHDL EXAMPLES **», 2008.**

[04] : **Loutani Amine, Mohammedi Amine :** Thème « **Implémentation d'un estimateur non linéaire sur une carte FPGA en utilisant l'environnement Matlab Simulink-Xilinx system Generator »**, mémoire de Master Université de Médéa.

[05]: **Jacques Weber, Maurice Meaudre: "** le langage VHDL, cours et exercices **», 2003.**

[06] : **Yacine Sehil :** Thème « **Synthèse d'une architecture neuro-floue a base de FPGA pour la linéarisation de la caractéristique d'un capteur non linéaire** ». Université de Médéa.

[07]: RICHARD MUNDEN, ASIC and FPGA Verification A Guide To Component Modeling, Morgan Kaufmann, 2005.

[08] : **Xilinx data book**, disponible au site : www.xilinx.com.

[09] : **M.W.NAOUAR**, thème « **Commande numérique à base de composants FPGA d'une machine synchrone** », Thèse Doctorat , Cergy Pontoise ,2007.

[10] : **ZAIBAK WAIL, ALLOUCHE MUSTAPHA** : thème« Implémentation Du Modèle De La Machine Asynchrone Triphasé Sur Une Carte FPGA En Utilisant L'environnement Matlab-Simulink Xilinx System Generator » Thèse de Master en Génie électrique, Université de Médéa ,2011.

[11] : **HAMIDIA FETHIA**, thème « Implémentation sur un circuit FPGA d'une commande non linéaire appliquée une machine électrique », mémoire de magister, université de Médéa, janvier 2008.

[12] : **Djoudi Ameur, Guerbbouh Med Amine :** thème « Implémentation de la commande scalaire et de la commande vectorielle d'une machine asynchrone à base d'une carte Dspace DS 1104 ». Université de Médéa ,2011.

[13] : **Alain Charbonnel** : « **Le moteur asynchrone triphasé** ».

[14] :**L.Baghli** : thème « **Modélisation et commande de la Machine asynchrone** ». IUFM de Lorraine –UHP.

[15] : **Caron, J. P.; Hautier, J. P**. « Modélisation et commande de la machine asynchrone, Ed Technip, 1995, 279p ».

[16]: **Leonhard, W.,** Control of electrical drives, 2nd ed. Springer-Verlag, 1996, 420p.

[17] **Krause P.C: '**Analysis of Electric Machinery', Edition, McGraw-Hill, New York, 1986.

[18] : **Christophe GOMBERT** : thème « **Simulation temps-réel des dispositifs d'Electronique de Puissance dédiés aux Réseaux d'Energie Electrique** ». Thèse de doctorat, INSTITUT POLYTECHNIQUE DE GRENOBLE.

[19]: **Blaschke, F.** «The principle of field orientation as applied to the new transvektor closed-loop control system for rotating-field machines, » Siemens Review, XXXIX, n°5, pp. 217-220, 1972.

[20] : **Leseinne, J.; Notelet, F.; Séguier, G**, « Introduction à l'électrotechnique

Approfondie, Technique et Documentation, 1981, 247p ».

LISTE DES ACRONYMES ET NOMENCLATURE

LISTE DES ACRONYMES :

ASIC : (**A**pplication **S**pecific **I**ntegrated **C**ircuit),

CAO : **C**onception **A**ssistée par **O**rdinateur.

CLB : (**C**onfigurable **L**ogic **B**lock).

CNA : **C**onvertisseur **A**nalogique **N**umérique.

CPLD : (**C**omplex **P**rogrammable **L**ogic **D**evice, circuit logique programmable complexe).

DAC : **D**igital to **A**nalog **C**onverter

DSP : **D**igital **S**ignal **P**rocessor.

EEPROM : (Electrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory).

EPLD : (**E**rasable **P**rogrammable **L**ogic **D**evice, circuit logique programmable et effaçable).

EPROM :(UVPROM) - (**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory).

FPGA : (**F**ield-**P**rogrammable **G**ate Array, réseau de portes programmables in situ).

IOB : (**I**nput **O**utput **B**loc).

IP : (**I**ntellectual **P**roperty).

LCA : (**L**ogic **C**ells **A**rrays).

LUT : **L**ook-**U**p **T**able.

MLI : **M**odulation de **L**argeurs d'**I**mpulsion.

PAL : (**P**rogrammable **A**rray **L**ogic, réseau logique programmable).

PLA : (**P**rogrammable **L**ogic **A**rray, réseau logique programmable).

PLD :(**P**rogrammable **L**ogic **D**evice, circuit logique programmable).

SRAM : (**S**tatic **R**andom **A**ccess **M**emory).

ROM : **R**ead **O**nly **M**emory.

RTL : **R**egistre **T**ransfer **L**evel.

VHDL : (**V**hsic « **V**ery High Speed Integrated Circuit » **H**ardware **D**escription **L**anguage).

JTAG: Joint **T**est **A**ction **G**roup.

NOMONCLATURE UTILISEE DANS LA MODELISATION DE LA MACHINE ASYNCHRONE

a, b, c : indices correspondant aux trois phases de la machine.

d, q : indices correspondant au référentiel lie au champ tournant.

$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$: indices correspondant au référentiel fixe (lié au stator)

$\mathbf{R_s}$, $\mathbf{R_r}$: résistances du stator et de rotor.

$\mathbf{L_s}$, $\mathbf{L_r}$: inductances cyclique entre le stator et le rotor.

M : inductances mutuelle cyclique entre le stator et le rotor.

P : nombre de paire de pôles.

$\boldsymbol{\Omega}$: vitesse mécanique.

$\boldsymbol{\omega_s}$ **:** pulsation statorique.

$\boldsymbol{\omega_r}$ **:** pulsation rotorique.

g : glissement.

J : moment d'inertie.

f : coefficient de frottement visqueux.

Cem : couple électromagnétique.

Cr : couple résistant.

LISTE DES FIGURES :

- **Figure I.1 :** Classification des circuits numériques……...2
- **Figure I.2 :** Schémas comparatif d'un **DSP** et d'un **FPGA…….....**3
- **Figure I.3 :** Diagramme des différents types de circuits logiques programmables……..4
- **Figure *I.4:*** Reprogrammabilité sur site d'un FPGA **……5**
- **Figure I.5 :** Structure d'une cellule **SRAM** …….6
- **Figure I.6 :** Architecture interne d'FPGA……7
- **Figure I.7 :** Schéma d'une cellule logique (XC 4000 de XILINX)……8
- **FigureI.8 :** Schéma d'un bloc d'entrée/sortie (IOB)……..8
- **Figure I.9 :** Connexions à usage général et détail d'une matrice de commutation……..9
- **Figure I.10 :** Les interconnexions directes……..10
- **Figure I.11 :** Les longues lignes……10
- **Figure I.12 :** Différent partie de langage VHDL…….....12
- **Figure I.13 :** Mode d'exécution matériel de la CAO……..14
- **Figure I.14 :** les étapes de développement une application sur FPGA…….15
- **Figure I.15 :** Hardware Co-simulation XSG……..16
- **Figure I.16 :** Co-simulation et Co-design par FPGA……17
- **Figure II.**1 : Rotor d'une machine asynchrone a cage d'écureuil……21
- **Figure II.2** : principe de fonctionnement d'un MAS……22

- **Figure II.3** : Représentation des enroulements statorique et rotoriques ……23
- **Figure II.4** : repère de Concordia…….....26
- **Figure II.5** : Transformation de Park……..26
- **Figure II.6 :** Schémas de position de MLI sur la chaine de régulation du moteur……...30
- **Figure II.7 :** Principe de la SPWM (MLI sinusoïdale)……31
- **Figure II.8 :** Onduleur de tension triphasé……32
- **Figure II.9 :** Schémas équivalent de modèle de la MAS……33
- **Figure II.10 :** Résultats de simulation de modèle de la MAS……34
- **Figure II.11 :** Schémas équivalent de la technique MLI sur MATLAB/SIMULINK……35
- **Figure II.12 :** Résultats de simulation de modèle de MLI……35
- **Figure II.13 :** Schémas équivalent de modèle de la MAS avec MLI……36
- **Figure II.14 :** Résultats de simulation de modèle de la MAS avec MLI….37
- **Figure III.1 :** L'analyseur ChipScope…….40
- **Figure III.2 :** Principe d'élaboration de la MLI Intersective……42
- **Figure III.3 :** Génération de signal sinusoïdal échantillonné en utilisant Matlab……43
- **Figure III.4 :** l'entité d'une description VHDl…….44
- **Figure III.5** : architecture d'une description VHDl…….45
- **Figure III.6** Principe de création des ram pou stocké les signaux nécessaire……45
- **Figure III.7 :** Convertisseur numérique-analogique et la tête associé……46

- **Figure III.8 :** Protocole de communication dans un **CNA……47**
- **Figure III.9 :** génération des signaux sinusoïdaux, A : résultat de synthèse (RTL), B : résultat de simulation……..48
- **Figure III.10 :** génération des signal triangulaire, A : résultat de synthèse (RTL), B : résultat de simulation……..49
- **Figure III.11 :** comparaison des signaux**,** A : résultat de synthèse (RTL), B : résultat de simulation……50
- **Figure III.12 :** Résultats de la comparaison……50
- **Figure III.13 :** Vue interne (RTL) du MLI……...51
- **Figure III. 14:** Résultats de simulation du bloc générale de la stratégie MLI (50 Hz), A : 6 dents, B : 8 dents, C : 9 dents, D : 12 dents……..52
- **Figure III.15 :** Résultats de simulation du bloc générale de la stratégie MLI (25 Hz), A : 6 dents, B : 8 dents, C : 9 dents, D : 12 dents…….53
- **Figure III.16 :** L'implémentation de la description…….54
- **Figure III.17 :** Configuration de la carte FPGA par un câble JTAG…..54
- **Figure III.18 :** Résultats de CNA sur l'analyseur ChipScope (6 dents)….55
- **Figure III.19 :** Comparaisons des résultats (6 dents de scie)……55
- **Figure III.20 :** Résultats de CNA sur l'analyseur ChipScope (8 dents)….56
- **Figure III.21 :** Comparaisons des résultats (8 dents de scie)……....56
- **Figure III.22 :** Résultats de CNA sur l'analyseur ChipScope (9 dents)….57
- **Figure III.23 :** Comparaisons des résultats (9 dents de scie)……..57
- **Figure III.24 :** Résultats de CNA sur l'analyseur ChipScope (12 dents)….58
- **Figure III.25 :** Comparaisons des résultats (12 dents de scie)……58

- **Figure III.26 :** Résultats de CNA sur l'analyseur ChipScope (6 dents)….59
- **Figure III.27 :** Comparaisons des résultats (6 dents de scie)……..59
- **Figure III.28 :** Résultats de CNA sur l'analyseur ChipScope (8 dents)….60
- **Figure III.29 :** Comparaisons des résultats (8 dents de scie)……..60
- **Figure III.30 :** Résultats de CNA sur l'analyseur ChipScope (9 dents)….61
- **Figure III.31 :** comparaisons des résultats……..61
- **Figure III.32 :** Résultats de CNA sur l'analyseur ChipScope (12 dents)…62
- **Figure III.33 :** Comparaison des résultats (12 dents)……62

LISTE DES TABLEAUX :

- **Tableau I.1 :** Statistiques du marché occupé par le vendeur d ′**FPGA…..18**
- **Tableau I.2 :** Avantages et inconvénients des **FPGA**…….19
- **Tableau II.1 :** Ressources consommés par la stratégie MLI Intersective...63

Résumé :

Dans notre travail, on étudier la faisabilité de l'implémentation de la stratégie MLI sur une carte FPGA pour la commande d'une machine asynchrone. A La première phase on modaliser et simuler le modèle de la stratégie MLI ainsi que le modèle de la machine asynchrone dans un environnement Matlab/Simulink.

La deuxième phase c'est la description VHDL de la stratégie MLI en utilisant les outils HDL (ISE Xilinx). En suite la simulation et l'implémenter sur la cible physique FPGA. En fin la validation des résulta par la comparaison entre celle obtenue par la simulation (MATLAB/ISE Xilinx) et les résultats réel obtenue par l'analyseur ChipScope.

Mots clés :

MLI, FPGA, Matlab/Simulink, VHDL, outils HDL, ISE Xilinx, ChipScope.

الملخص :

في هذه المذكرة قمنا بدراسة إمكانية تنفيذ إستراتيجية (PWM) في بطاقة (FPGA) لقيادة محرك غير متزامن. المرحلة الأولى هي إنشاء نموذج رياضي لإستراتيجية (PWM) لمحاكاته ومحاكاة نموذج المحرك غير المتزامن في برنامج (SIMULINK / MATLAB).
المرحلة الثانية هي إنشاء وصف(VHDL)لإستراتيجية (PWM)باستخدام أدوات HDL (ISE XILINX) ثم محاكاته وتنفيذ هذا الوصف في الهدف المادي بطاقة(FPGA) . في االاخير التحقق من صحة النتيجة عن طريق المقارنة بين تلك التي تحصلنا عليها بالمحاكاة (MATLAB / ISE XILINX) والنتائج الحقيقة التي حصلنا عليها عن طريق المحلل (ChipScope).

المفتاح :

MLI, FPGA, Matlab/Simulink, VHDL, outils HDL, ISE Xilinx, ChipScope.

Summary:

In our work, we have study the possibility of implementing a PWM strategy in an FPGA board for controlling an induction motor .A first phase is to make a mathematical model for PWM strategy and simulate this model with the model of the machine asynchronously in a Matlab / Simulink environment.
The second phase is the hardware description of the PWM strategy using HDL tools in (Xilinx ISE).Then the simulation and the implementation of this description where preferment by using the physical target FPGA. At the end comes the validation of result by the comparison between that obtained by the simulation (MATLAB / Xilinx ISE) and the real one obtained by the ChipScope analyzer.

Keywords:

MLI, FPGA, Matlab/Simulink, VHDL, HDL tools, ISE Xilinx, ChipScope.

SOMMAIRE

Résumé……..…I

Remerciement…….II

Liste des figures…….III

Liste des tableaux……..IV

Liste des acronymes et nomenclature…….V

Introduction générale …….VII

Chapitre I : Présentation des circuits FPGA.

I.1 Introduction 1

I.2.Les circuits programmable 2

I.3. Les circuits logiques programmables 4

I.4. Les circuits logiques programmables de type FPGA 4

I.4.1. Technologie de programmation des FPGA ……5

I.4.1.1.Technologie à base de SRAM …….....6

I.4.1.2. Technologie à base d'EEPROM ou FLASH**…...**.....6

I.4.1.3.Technologie à base d'ANTI-FUSIBLE **……...**……..6

I.5. Architecture interne d'FPGA……..6

I.5.1. Les CLB (Configurable Logic Block)……...7

I.5.2. Les IOB (Input Output Bloc)…….……8

I.5.3. Les différents types d'interconnexion …….9

I.5.3.1.Les interconnexions à usage général …….9

I.5.3.2. Les interconnexions directes …….....10

I.5.3.3. Les longues lignes ……10

I.6. Langage de description matérielle…….11

I.7. Configuration des FPGA …….13

I.7.1. La configuration des FPGA par les outils HDL…….13

I.7.1.1.Spécification du design ……...14

I.7.1.2.Conception du design …….14

I.7.1.3.Synthèse …….14

I.7.1.4.Placement et routage ……...15

I.7.1.5.Intégration et implémentation ……...15

I.7.2. La configuration des FPGA par les outils XSG…….16

I.8. Différents domaines d'applications d'FPGA……..17

I.9. Principaux fabricants d'FPGA ……..18

I.10. Avantages et inconvénients de FPGA…… 19

I.11.Conclusion……..20

Chapitre II : Modélisation de la machine asynchrone triphasé.

II.1.Introduction…….....21

II.2. Description et principe de fonctionnement du MAS …….21

II.3. Modélisation du moteur asynchrone ……22

II.3.1. Modèle d'une MAS dans le référentiel triphasée

(naturelle)……...23

II.3.2.Equations électriques sur les axes « a, b, c » …….....23

II.3.3.Equation magnétique .....….24

II.3.4.Equations mécanique……...24

II.3. 5. Modèle d'une MAS dans le référentiel biphasé.....25

II.4. La fonction de modulation MLI (PWM) ……29

II.4.1.Généralités ……29

II.4.2. La technique de modulation MLI ……..30

II.4.3.MLI Intersective (modulation par porteuses)….....31

II.5. Modélisation de l'onduleur ……32

II.6. Le modèle de MAS sur Matlab/Simulink ……..33

II.7.Conclusion……...38

Chapitre III : Implémentation de la stratégie MLI sur la carte FPGA.

III. Introduction…….39

III.1. Présentation de l'outil de développement ISE XILINX ……39

III.2.Présentation de L'analyseur ChipScope ……...40

III.3.Réalisation et implémentation de la stratégie MLI dans un circuit FPGA……..…41

III.3.1.Descriptions VHDL de la stratégie(MLI) ……41

III.3.2. La Description VHDL……...44

III.3.1.Convertisseur numériques analogique DAC …….46

III.4. Synthèse et simulation de la conception réalisée pour MLI….…….48

III.4.1. Génération des signaux de comparaison (sinusoïdale et triangulaire)……..48

III.4.2. Synthèse du bloc générale de la stratégie MLI …….……...51

III.5. L'implémentation de la description de la stratégie MLI…….54

III.5.1. Résultats de l'implémentation pour f=50 Hz ……....55

III.5.2. Résultats de l'implémentation pour f=25 Hz ……59

III.6. Rapport de consommation des ressources ……63

III.7. Conclusion……64

Conclusion générale……

Annexe……

Bibliographie……..