Université MOULOUD MAMMERI de TIZI-OUZOU
Faculté de génie électrique et d'informatique
Département d'Electronique

Mémoire

Pour l'obtention du diplôme de

MAGISTER

En Automatique

Option : Automatique des systèmes continus et productique

Présenté Par

Zahir Ait Ouali

Application des FPGA à la commande d'un moteur asynchrone

JURY

Mr Djennoune Said	UMMTO	Professeur	Président
Mr Benfdila Arezki	UMMTO	Professeur	Rapporteur
Mr Mellah Rabah	UMMTO	Maître de conférences	Examinateur
Mr Bensidhoum M-Outahar	UMMTO	Maître de conférences	Examinateur
Mr Daoui Mohamed	UMMTO	Maître de conférences	Examinateur

RESUME

Ce travail de mémoire s'inscrit dans le cadre de développement d'un système embarqué sur FPGA (Field Programmable Gate Array). Une implémentation hardware dans l'objectif est de piloter un moteur asynchrone triphasé. C'est une nouvelle approche pour améliorer les performances du contrôle numérique des machines tournantes en temps réel. La technologie des circuits logiques programmables FPGA est une alternative innovante par rapport aux processeurs de contrôle conventionnels (DSP /microprocesseurs).C'est pour cette raison que les FPGA sont l'objet de recherches actives. A cet effet, nous nous sommes posés comme objectif de concevoir la circuiterie de commande vectorielle du moteur asynchrone triphasé par une logique câblé configurable sur FPGA. La première partie de ce document est dédiée à l'état d'art des FPGA. Nous exposons les principaux types de circuits intégrés programmables , allant de la simple mémoire programmable au circuit FPGA en suite nous citerons l'apport des FPGA sur la commande des machines électriques. La deuxième partie sera consacrée pour la modélisation du moteur asynchrone et la structure de commande. Nous choisissons la méthode vectorielle et la stratégie de modulation par largeur d'impulsions PWM afin de l'implémenter sur le circuit intégré FPGA où ce dernier s'occupe de générer les séquences de pulsations désirées. La troisième et quatrième partie s'intéressent aux différents aspects de la conception des systèmes intégrés : description, simulation et synthèse. Nous utiliserons le langage de description matériel VHDL pour la modélisation comportementale des composants et un environnement logiciel pour la conception et la simulation. Cet environnement de la firme XILINX est nommé XILINX ISE 10.1 dédié à la CAO des circuits numériques accompagné de son simulateur intégré ISE SIMULATOR.

Nous clôturons ce mémoire par une conclusion générale et quelques perspectives de recherches.

Mots-clés *:* FPGA, Modélisation, Commande, Conception, Numérique, Description, Synthèse, Logique câblée, Programmation hardware, Bloc logiques, SOC, VHDL, PWM, CAO.

AVANT-PROPOS

Le travail présenté dans ce mémoire a été préparé au sein du laboratoire de microélectronique au département d'automatique de la faculté de Génie Electrique et d'Informatique à l'université MOULOUD MAMMERI de TIZI OUZOU sous la direction du Professeur AREZKI BENFDILA.

A cet effet, Je suis très honoré que Monsieur le Professeur AREZKI BENFDILA du laboratoire de microélectronique à accepter d'être mon promoteur. Je tiens à le remercier très vivement et lui exprimer ma profonde reconnaissance pour m'avoir proposé ce sujet de recherche qui est pluridisciplinaire par excellence et m'avoir accueilli au sein de son laboratoire. Aussi, je tiens ainsi à le remercier pour sa rigueur scientifique et ses qualités humaines ainsi pour ces conseils didactiques, ces nombreux encouragements et la très grande liberté qu'il ma accordée dans l'orientation de mes travaux de recherche.

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

A titre de reconnaissance, je dédie ce travail à tous les gents qui se donnent à fond à la recherche scientifique.

Je dis également un grand merci à mes amis, mes collègues et mes enseignants du laboratoire de commande des systèmes continus, et spécialement à NACER AHMIM de l'université de Boumerdes Option mécatronique qui, tant comme ami que collègue, m'a apporté son soutien moral et technique quand j'en ai eu besoin.

Finalement, je réserve une place singulière à toutes les personnes qui ont contribué, d'une façon directe ou indirecte à l'achèvement de ce travail et a tous ceux et celles qui m'ont aidé et soutenu durant tout mon parcours trouvent ici l'expression de mes remerciements les plus sincères car le chemin que j'ai parcouru depuis le début de mes études est le fruit de relations avec de nombreuses personnes sans les nommer. C'est pourquoi je tiens à terminer en présentant mes excuses à ceux qui ne se retrouveront pas dans cette liste, qui est bien loin d'être exhaustive.

ZAHIR AIT OUALI

SOMMAIRE

TABLE DES MATIERES

CHAPITRE II

MODELISATION ET COMMANDE DU MOTEUR ASYNCHRONE TRIPHASE

CHAPITRE III

APPROCHE NUMERIQUE DE LA COMMANDE VECTORIELLE

CHAPITRE IV

SIMULATION ET SYNTHESE DU CIRCUIT DE COMMANDE

LISTE DES FIGURES ET TABLEAUX

LISTE DES ACRONYMES ET NOMENCLATURE

Les notations utilisées sont très variées car le sujet est pluridisciplinaire. Les principales utilisées dans ce mémoire sont rapportées ci-dessus, d'autres significations se trouvent explicitées dans le texte.

LISTE DES ACRONYMES

ASIC: *Application Specific Integrated Circuit.*

CAO **:** *Conception Assistée par Ordinateur.*

CLB **:** *Configurable Logique Bloc.*

CPLD **:** *Complex Programmable Logique Device.*

CORDIC: *COordinate Rotation DIgital Computer.*

CPLD*: Complex Programmable Logic Device.*

DSP: *Digital Signal Processor.*

EPROM: *Erasable Programmable Read Only Memory.*

EEPROM: *Electrically Erasable Programmable Read Only Memory.*

EEPLD*: Electrically Erasable Programmable Logic Device.*

FPGA: *Fieled Programmable Gate Array.*

IOB: *Input Output Bloc.*

IP: *Intellectual Properties.*

LUT : *Look Up Table.*

MLI: *Modulation de Largeurs d'Impulsions.*

PLD: *Programmable Logique Device.*

PROM: *Programmable Read Only Memory .*

PI **:** *Proportionnel Intégral.*

PAL **:** *Programmable Array Logic.*

PLD: *Programmable Logic Device.*

ROM: *Read Only Memory.*

RTL: *Register Transfer Level.*

SRAM: *Static Random Access Memory.*

***SOC*: *S*ystems *O*n *C*hips.**

***VHDL*: *V*ery high speed integrated circuit *H*ardware *D*escription *L*anguage.**

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 lié au champ tournant.*

α,β : *Indices correspondant au référentiel fixe (lié au stator).*

Rs ,Rr : *Résistances du stator et du rotor.*

Ls , Lr : *Inductances cycliques du stator et du rotor.*

M : *Inductance mutuelle cyclique entre le stator et le rotor.*

p : *Nombre de paires de pôles.*

Ω : *Vitesse mécanique.*

ωs : *Pulsation statorique.*

ωr : *Pulsation rotorique.*

g : *Glissement.*

θ : *Position du référentiel par rapport au stator.*

J : *Moment d'inertie.*

f : *Coefficient de frottement visqueux.*

Cem : *Couple électromagnétique.*

Cr : *Couple résistant.*

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Les concepteurs de 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 Conception Assistée par Ordinateur (CAO) est plus qu'indispensable. L'avantage des méthodes de CAO 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. Dans des approches pluridisciplinaires en particulier la mécatronique, il est indispensable de mettre en place des méthodes et des outils facilitant l'intégration de solutions analogiques, numériques et mixtes. L'introduction et l'accumulation d'innovations technologiques sur les circuits intégrés ainsi que leurs expansion a permis l'automatisation des tâches considérées 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, les entraînements ont pris une importance considérable ces dernières décennies et constituent un domaine de recherche très dynamique. Mais la complexité croissante des algorithmes de commande des systèmes électriques ne cessent d'augmenter a cause des contraintes liées aux modèles (non-linéarités, couplage, variation de paramètres…etc.) et d'autres contraintes liées aux performances (temps d'exécution, précision...etc.). Motivé par les exigences grandissantes en puissance de traitement et afin de répondre aux exigences strictes concernant les performances exigeantes en puissance de calcul, les FPGA représentent une alternative pour compenser les DSP qui sont devenus classique avec une grande souplesse et de bonnes performances (parallélisme de traitement, vitesse, surface, consommation….etc.). 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. Alors, les FPGA présentent beaucoup de perspectives pour l'implantation d'algorithmes de contrôle des machines.

Pour ces FPGA, les outils de conception assistée par ordinateur 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 telque les ASIC ou FPGA.C'est

ce qui nous a permis d'opter aux circuits intégrés programmables FPGA pour la commande des moteurs plus précisément le moteur asynchrone comme sujet d'application de notre travail dans l'intérêt est de pouvoir faire une recherche plus approfondie concernant la dynamique d'innovation technologique du contrôle numérique des procédés.

Ce travail de thèse a pour objectif l'élaboration d'une étude complète avec conception, simulation et réalisation d'un circuit numérique d'une commande pour un moteur asynchrone triphasé à base d'un circuit reconfigurable FPGA en vue d'améliorer les performances et la fiabilité du circuit de commande en tenant compte des spécificités structurelles des circuits reconfigurables FPGA. C'est dans ce contexte que cette thèse est élaborer avec la mise au point d'une solution architecturale pour des algorithmes en temps réel.

La démarche scientifique adoptée dans cette thèse suit cette structuration:

-Le premier chapitre est consacré à l'état d'art des circuits reconfigurables FPGA. On traitera l'architecture interne de ces derniers et une comparaison des FPGA et les circuits séquentiels classiques puis la contribution des FPGA pour la commande des machines.

-Le second chapitre présente brièvement des rappels sur le modèle de la machine asynchrone et du variateur de vitesse et met en évidence une technique de commande du moteur asynchrone qui est la commande vectorielle qui sera implémenté.

-Le troisième chapitre concerne la conception du circuit de commande numérique à base de circuits logiques élémentaires inspirés a partir de l'algorithme analytique de contrôle avec une décomposition en éléments simples.

-Le quatrième et dernier chapitre est consacré à pour la validation et l'implémentation de notre solution avec un langage de description matériel VHDL. Une introduction au VHDL sera introduite afin de se familiariser avec la transcription d'un circuit numérique en description textuelle dans le but est de faire la simulation, la synthèse et l'implantation du circuit de commande sur un circuit cible FPGA. En suite, une petite présentation du logiciel Xilinx ISE permettant de valider notre approche par sa réalisation ainsi la visualisation des résultats de simulations obtenus qui serons présentées et commentées.

Enfin, nous clôturons ce document avec une conclusion générale ainsi que des perspectives d'application et d'exploitation de la technologie programmable FPGA qui est prometteuse et en pleine maturité.

CHAPITRE I

Etat d'art des FPGA et leurs apports à la commande des machines

OBJECTIF

L'objectif de ce chapitre, avant d'aborder le vif du sujet, est de présenter les *FPGA* (*Field-Programmable Gate Array*) afin de se familiariser avec ce type de circuits et en fin, nous présentons leurs contributions à la commande des machines tournantes.

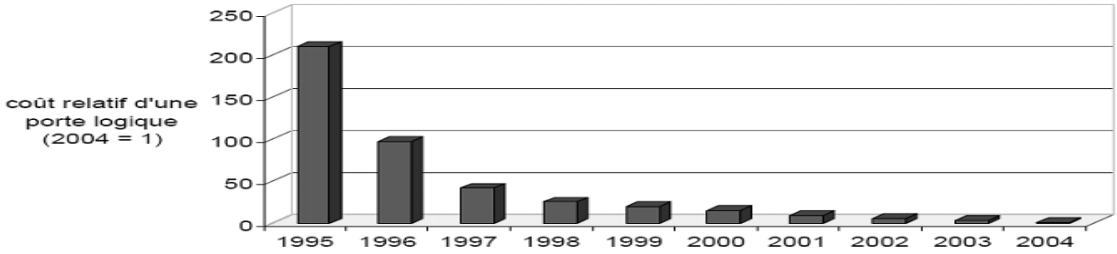
I.1) – INTRODUCTION

L'histoire des circuits intégrés a commencé après l'invention du transistor en *1947* par les laboratoires *Bell*. Relativement, l'apparition du premier circuit intégré est au cours des années *1958* et *1959* grâce a un jeune ingénieur du nom de *Jack Kilby* de la firme *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 en *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 permis de prédire que le taux d'intégration des transistors dans ces circuits double tous les deux ans. A nos jours, ce constat reste vérifiée 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 a cause de la complexité des circuits et des fonctions à intégrer qui demandent divers efforts et compétences. Par conséquence des coûts élevés et cette technologie est inaccessible au grand public.

Aujourd'hui, l'avènement des dernières générations d'*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 proposés 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 reconfigurés (Architecture programmée modifiable) partiellement ou entièrement suivant l'application. A cet effet, 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 commande des moteurs.

I.2)-FACTEURS D'EVOLUTION DES CIRCUITS NUMERIQUES

Les circuiteries numériques (où la porte logique représente l'unité de base) reposent sur les trois aspects des circuits logiques qui sont : le combinatoire, le séquentiel et l'hybride des deux. Des statistiques qui ont été faites ces dernières années ont montrés qu'au bout de *10* ans, le prix d'une porte logique a été divisé par *200* où l'intérêt économique du numérique. A cet effet, le prix d'une porte logique est divisé se réduit de *40%* par an. Le diagramme suivant montre ces statistiques d'évolution des couts durant ces dernières années:



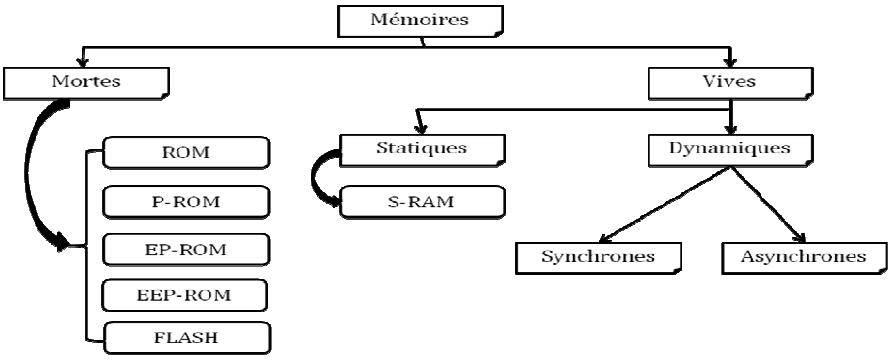
***Figure(I.01)** : Diagramme d'évolution des coûts ces dernières années.*

Cette évolution est le fruit d'un ensemble de facteurs qui sont récapitulés comme suit :

1. **B**esoins croissants en circuits spécialisés.
    - **P**roduits de plus en plus complexes.
    - **C**ontraintes (performance, coût…).
2. **E**volution rapide de la technologie.
    - **G**énéralisation des « Systèmes sur Puce ».
    - **E**space de conception trop grand.
3. **E**volution des outils de conception "haut-niveau".
    - **P**roduire une architecture à partir d'un algorithme.
    - **E**xploration automatique de l'espace de conception.
    - **O**utils d'estimations (performances, surface, etc.).

I.3)-LES TECHNOLOGIES DE MEMORISATION

Voici d'une manière générale, l'arborescence des mémoires disponibles à nos jours :



***Figure(I.02)** : Les différents types de mémoires.*

L'ensemble des caractéristiques de ces mémoires sont récapitulées comme suit :

- **Les *ROM (Read Only Memory):*** Mémoires figées par le concepteur a lecture seule et non modifiables.
- **Les *P-ROM (Programmable Read Only Memory):*** Mémoires programmables une fois par l'utilisateur avec un équipement spécialiser (tableau de fusibles).
- **Les *EP-ROM (Erasable Programmable Read Only Memory):*** Mémoires programmables électriquement et effacement par des rayons ultra-violets au bout d'un certain temps (Quelques minutes).
- **Les *EEP-ROM (Electrically Erasable Programmable Read Only Memory):*** Mémoires programmables électriquement à lecture seule, effaçables électriquement (Quelques millisecondes).
- **Les *mémoires FLASH:*** Elles sont une version plus évoluer des *EEP-ROM* avec avantage d'être plus facile à programmer et à effacer.
- **Les *S-RAM (Static Random Mémory):***Mémoires volatiles avec cellule de base à plusieurs transistors (accès rapide, consommation plus, coûteux). La volatilité correspond au non disponibilité de l'information lorsqu'il n'y a pas d'alimentation.
- **Les *RAM dynamiques*:** Mémoires volatiles qui nécessitent rafraichissement périodique de l'information afin de la conserver avec cellule de base à un transistor (densité forte, accès lent).

I.4)-LES CIRCUITS PROGRAMMABLES

La plupart des circuits numériques programmables sont issues de la célèbre architecture« *Architecture 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 où le facteur temps d'exécution est le plus important. Cette architecture nommée « *Architecture **H**arvard »*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 des deux précédentes architectures. Cette solution réside dans le mode de programmation qui est devenu architectural à logique câblée inversement aux deux premières architectures précédentes où la programmation est séquentielle.

D'une manière générale, il existe deux alternatives ou solutions qui sont:

- **Une solution logicielle**: Elle est nommée aussi solutions programmables du type « *Processeur* » où un traitement séquentiel relativement lent et programmation dépendante du composants (*DSP, Microprocesseur et Microcontrôleur…).*
- **Une solution matérielle***:* Elle est nommée aussi solutions programmables du type « Logique » où un traitement parallèle en temps réel et une programmation architecturale avec un langage de description matériel *HDL* (Méthodologie de conception *CAO*) indépendante du composant (*ASIC* et *FPGA…).*

Les caractéristiques de ces circuits sont :

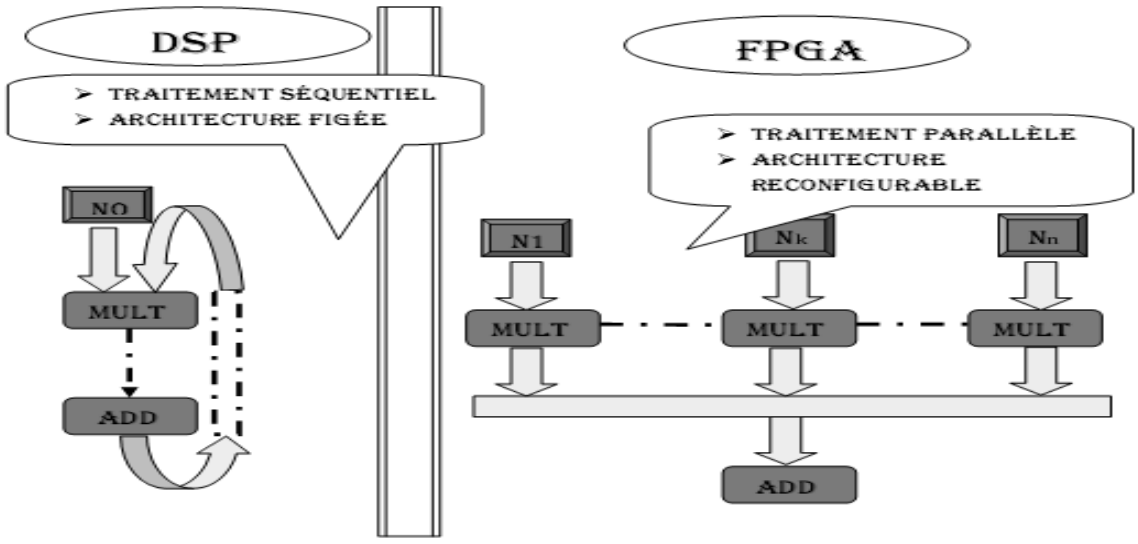
1. **Les circuits du type DSP/Microprocesseurs** : Un rapport performance/coût faible, un temps de conception très court et une grande souplesse d'utilisation.
2. **Les circuits du type spécialisé ASIC** : Très performants mais avec un cycle de conception long et une architecture figée.
3. **Les circuits du type FPGA**: Des performances proches des *ASIC***,** un coût unitaire intermédiaire et un cycle de conception moyen et une architecture modifiable.

Voici un tableau récapitulatif de comparaison des différentes solutions numériques :

	ASIC	FPGA	DSP
Performances	Très élevées	Elevées	Faibles
Taille	Faible	Moyenne	Elevée
Consommation	Faible	Moyenne	Très élevées
Intégration	Système sur puce	Système sur puce	Composants supplémentaires
Souplesse	Fonctions figées	Reconfigurable	Programmable
Mise en œuvre	Complexe	Complexité moyenne	Complexité moyenne
Coût du composant	Très élevé	Moyen	Faible

***Tableau(I.01)**: C*omparaison des différentes solutions numériques.

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

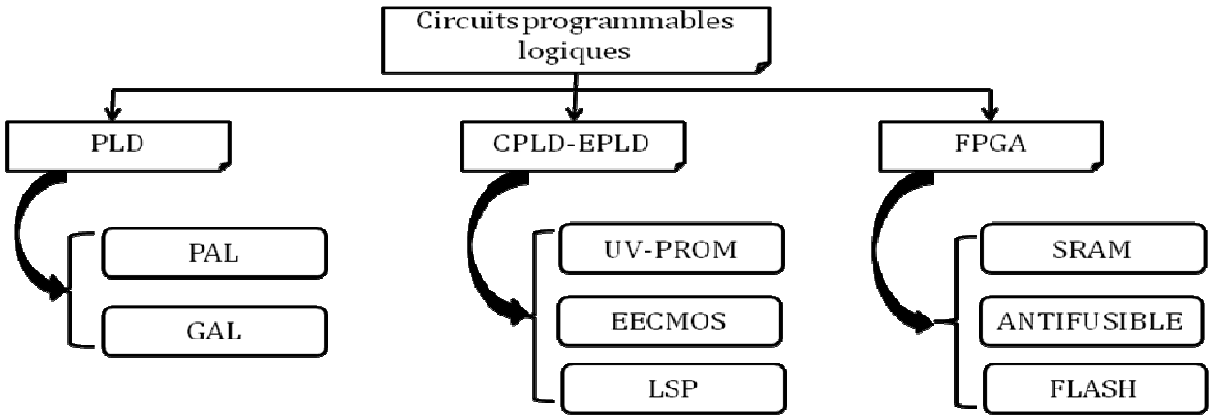


***Figure(I.03)**: Schéma comparatif d'un **DSP** et d'un **FPGA**.*

Les circuits séquentiels ont une architecture matérielle figée et inversement pour les circuits logiques reconfigurables *FPGA*, on fait une adaptation de l'architecture du composant en fonction des l'algorithme.

1.5)-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.* L'arboressance suivant illustre les différents types suivant la technologie utilisée.



***Figure(I.04)** : Diagramme des différents types de circuits logiques programmables.*

- **Les PLD (Programmable Logic Device):**Famille des circuits programmables qui comprend les *PAL*, *GAL*.
  - ✓ **PAL (Programmable Array Logic):** Circuits logiques programmables dans lesquels seules les fonctions *ET* sont programmables, les fonctions *OU* ne le sont pas.
  - ✓ **GAL (Generic Array Logic):**Circuits logiques *PAL* reprogrammables à technologie *CMOS.*
- **Les CPLD ou EPLD (Erasable Programmable Logic Device**):Circuits logiques reprogrammables.
  - ✓ **ISP (In System Programmable):**Circuit que l'on peut programmer même lorsqu'il est en place sur l'application.
- **Les FPGA (Field Programmable Gate Array)** *:* Ces circuits sont une évolution des *CPLD.* Récemment, ils intègrent également des mémoires entières, des multiplieurs et même des noyaux de processeur.

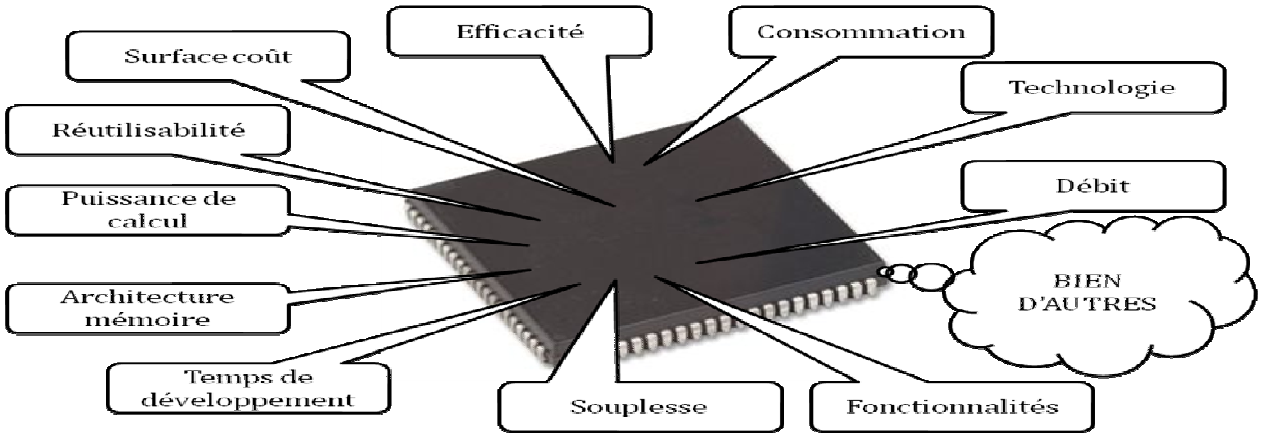
I.6)-LES CIRCUITS LOGIQUES PROGRAMMABLES DU TYPE FPGA

I.6.1)-Critères de chois du circuit programmable FPGA

Les *FPGA* sont développés récemment grâce aux progrès de la technologie *VLSI*, l'apparition de ce type de circuits est une révolution des systèmes digitaux et ouvrants des perspectives de traitement numérique inaccessibles auparavant. 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 coût 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, il est nécessaire de respecter un certain nombre de règles comme :

- **B**ien connaître les caractéristiques du *FPGA* ciblé pour assurer son adéquation avec les besoins du projet.
- **E**laborer une méthodologie de conception.
- **M**aîtriser les outils d'implémentation et de choisir des outils de synthèse de qualité.

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.



***Figure(I.05)** : Critères de chois du circuit logique programmable **FPGA**.*

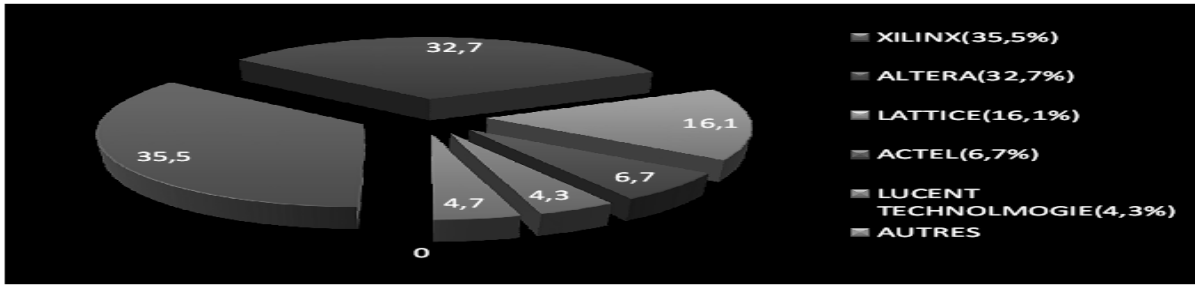
I.6.2)- Différent domaines d'application des FPGA

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

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. **Aérospatiale** *:* Satellites. **Militaire** *:* Radar, Communication protégée, la détection ou la surveillance. **Autres**.

I.6.3)-Principaux fondeurs d'FPGA

Les fabricants des *FPGA* ne cessent pas d'améliorer leurs produits par l'efficacité et la puissance. L'ensemble des firmes (*Principaux fondeurs*) qui conçoivent ce type de circuits sont : ***Actel ,Altera , Atmel ,Cypress, Lattice, Minc , QuicLogic, Xilinx*** et d'autres.

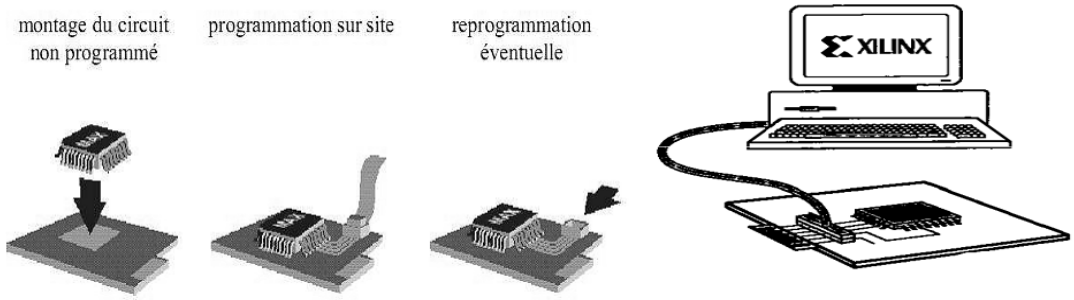


***Figure(I.06)** : Statistiques du marché occupé par les vendeurs d'**FPGA**.*

1.6.4)-Configuration et reconfiguration des FPGA

Un système reconfigurable est un système qui est constitué de composants ou entités à architecture modifiable afin de répondre à un objectif bien déterminé. Ce système reconfigurable dispose d'un mécanisme permettant de choisir une nouvelle configuration et de la mettre en place dans le cadre du processus de reconfiguration. Les circuits *FPGA* sont un type de ces circuits reconfigurables. Ils sont programmables ou configurables sur les cartes

sur quelles ils sont implantés par l'utilisateur. 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.



Figure(I.07)**: Reprogrammabilité sur site d'un **FPGA.

1.6.5)-Technologies de programmation des FPGA

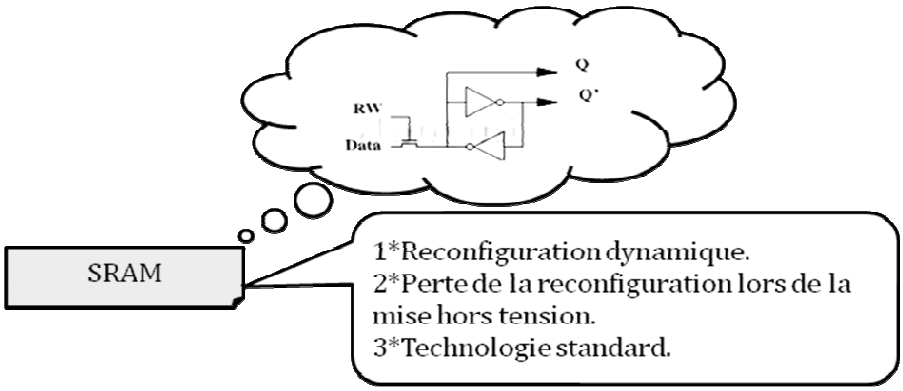
Pour franchir les inconvénients susmentionnés des mémoires, et dans le but de faire un ensemble de technologies complémentaire adaptable suivant l'environnement des cahiers de charges, il existe trois types d'*FPGA* reprogrammables suivant la technologie de mémorisation pour répondre aux différentes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

Ces trois principales technologies d'*FPGA* sont :

- **T**echnologie de programmation par ***RAM***.
- **T**echnologie de programmation par ***EEPROM*** ou ***FLASH***.
- **T**echnologie de programmation par ***ANTI-FUSIBLE.***

I.6.5.1)-Technologie à base de RAM (XILINX et ALTERA)

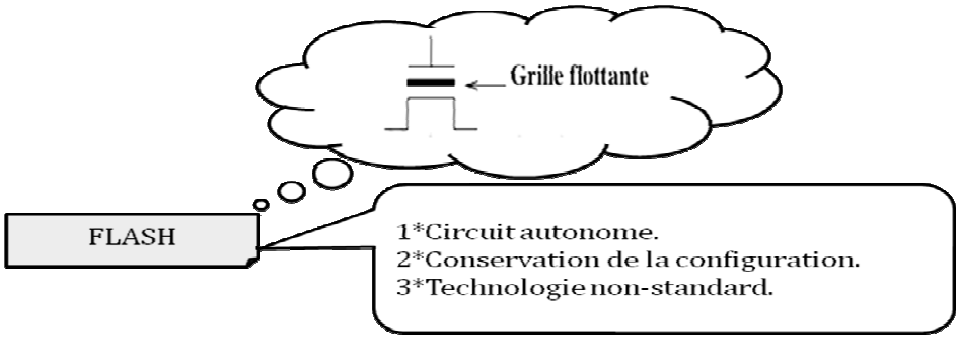
Cette technologie permet d'avoir une reconfiguration rapide des *FPGA* .Les points de connexions sont des ensembles de transistors commandés. L'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du *FPGA* dans une autre mémoire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



***Figure(I.08)**: Caractéristiques des technologies **SRAM**.*

I.6.5.2)- Technologie à base d'EEPROM ou FLASH (LATTICE et ACTEL)

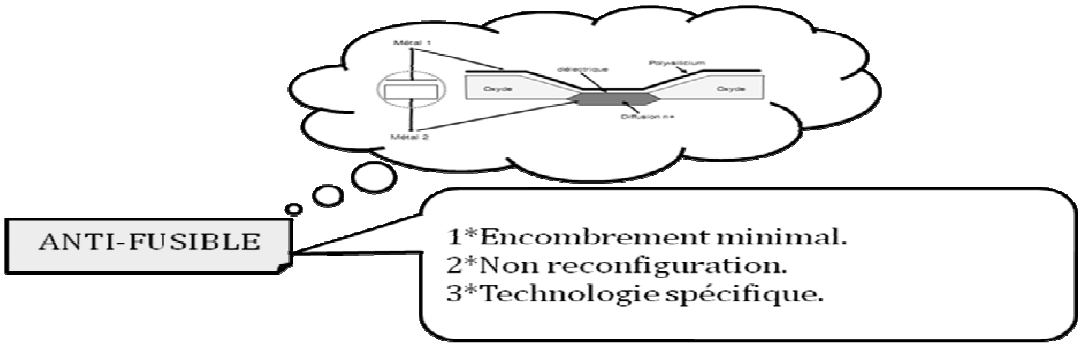
Cette technologie garde sa configuration mais un nombre limité de configuration avec une configuration plus lente par rapport à *SRAM*.



***Figure(I.09)**: Caractéristiques des technologies **FLASH**.*

I.6.5.3)- Technologie à base d'ANTI-FUSIBES(ACTEL)

Les points de connexions sont du type *ROM*, c'est-à-dire que la modification du point est irréversible. Pour comprendre le mécanisme de connexion 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.

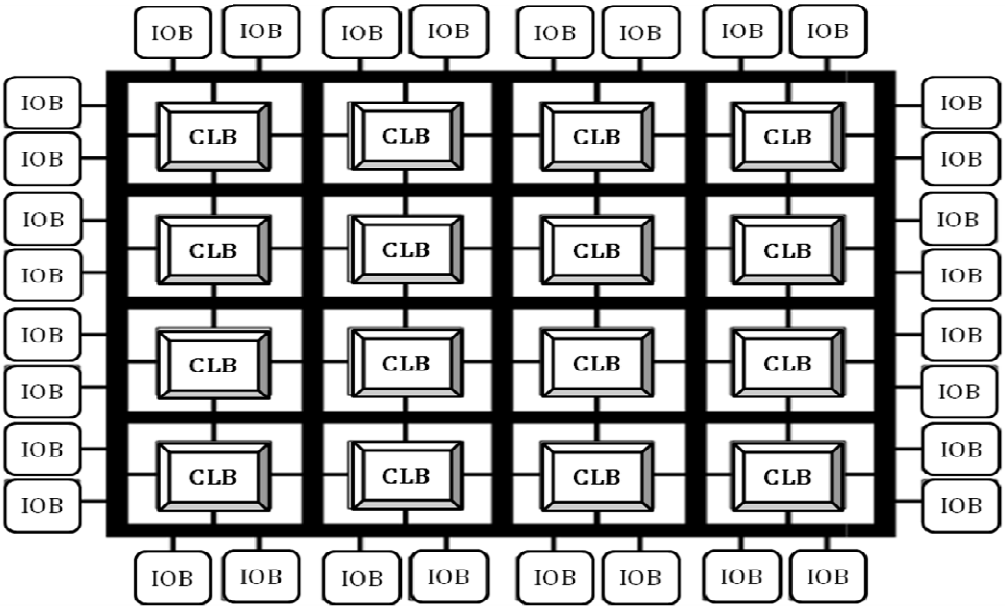


***Figure(I.10)**: Caractéristiques des technologies **ANTI-FUSIBLES**.*

I.6.6)-Architecture interne des FPGA

On appelle les *FPGA* quelques fois *LCA*, abréviation anglaise de « ***L**ogic **C**ell **A**rray* » signifiant réseau de cellules logiques. Pour réussir à implanter un système dans un *FPGA* de manière efficace, il est indispensable de bien connaître sa structure interne et ses limites du point de vue performances. Les composants logiques programmables sont des circuits composés de nombreuses cellules logiques élémentaires librement assemblables. Celles-ci sont connectées de manière définitive ou réversible par programmation afin de réaliser les ou les fonctions numériques désirées. Un *FPGA* (***F**ield-**P**rogrammable **G**ate **A**rray*) est un circuit intégré avec une structure adaptable par l'utilisateur et composée d'un réseau de cellules élémentaires ou d'éléments logiques programmables *CLB* et *IOB* répartis régulièrement et reliés entre eux grâce à des connections qui forment une matrice de routage programmable pour obtenir un comportement spécialisé du circuit dans sa globalité. Puisque tous les éléments sont programmables, le *FPGA* peut émuler et réaliser n'importe quel circuit a l'unique condition que celui-ci n'épuise pas les ressources logiques et de routage du *FPGA*. L'ensemble des systèmes reconfigurables *FPGA* est subdivisé en trois catégories suivant les fonctions préexistantes et des possibilités de les interconnectées. Ces catégorie sont : des systèmes reconfigurables nommés "***grain fin***", des systèmes reconfigurables nommés "***grain moyen***" et des systèmes reconfigurables nommés " ***grain large***".

L'architecture interne des *FPGA* est différente d'un fondeur 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 suivante :



***Figure(I.11)**: Architecture interne d'un **FPGA**.*

❖ Les macro-cellules internes sont appelées :

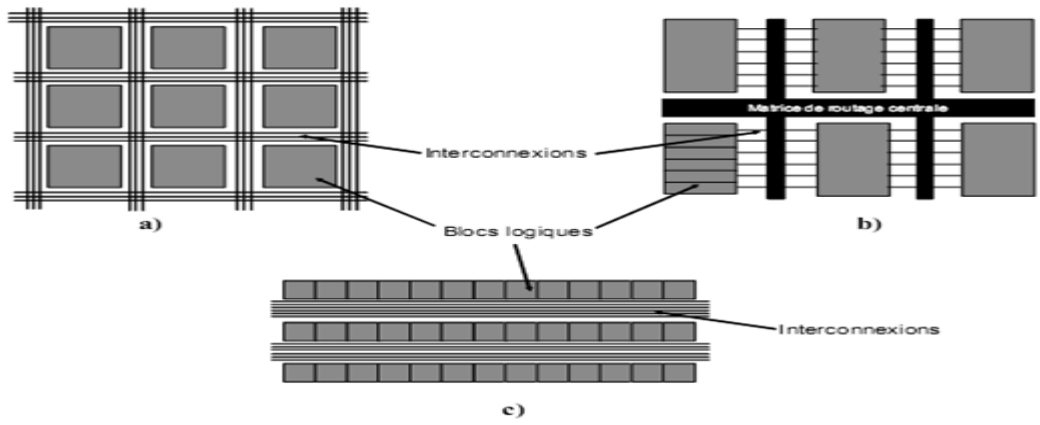
- **S**oit ***CLB*** qui est la dénomination adoptée par *Xilinx* et abréviation anglaise de*« **C**onfigurable **L**ogic **B**lock »,* signifiant bloc logique configurable.
- **S**oit ***LC*** qui est le nom choisi par *Cyprès* et abréviation anglaise de *« **L**ogic **C**ell »,* signifiant cellule logique.
- **S**oit ***LE*** qui c'est l'appellation d'Altéra abréviation anglaise de *« **L**ogic **E**lement»* signifiant élément logique.

❖ Les macro-cellules sur la périphérie sont appelées : ***IOB*** abréviation anglaise de« ***I**nput **O**utput **B**lock* », signifiant bloc logique d'entrées sorties.

❖ L'ensemble des points de connexion est appelé ***PIP***, abréviation anglaise de « ***P**rogramme **I**nterconnect **P**oints* ».

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

Les architectures existantes peuvent être regroupées en trois grandes catégories suivant la manière dont les blocs logiques sont organisés comme le montre la figure suivante:



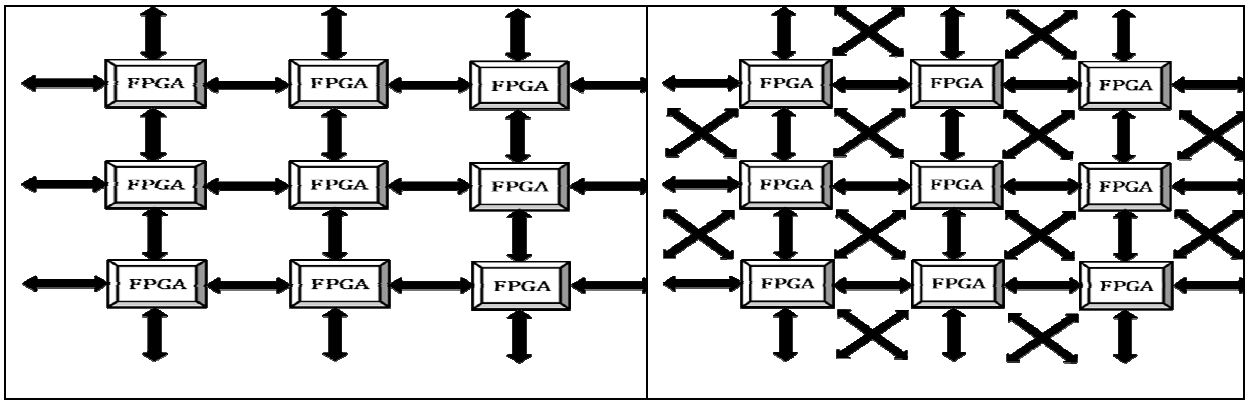
***Figure(I.12)**: Différentes architectures des **FPGA**.*

I.6.7)- Architecture MULTI-COMPOSANTS

Dans un environnement multi-composant,plusieurs structures sont possibles pour un certain nombre d'*FPGA* qui admettent l'association et comme exemple:

I.6.7.1)- Association de plusieurs FPGA

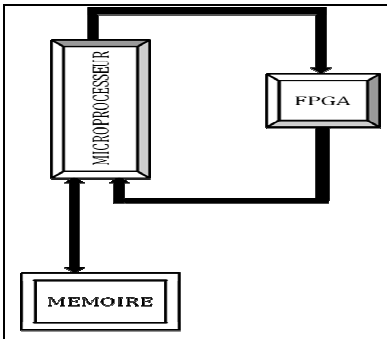
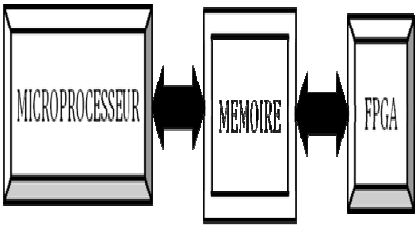
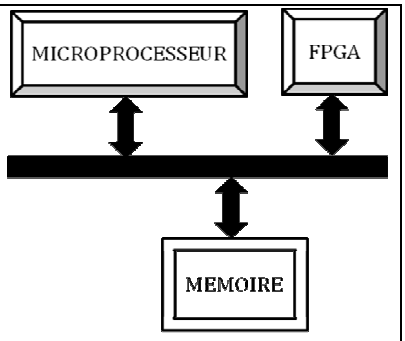
L'assocition de plusieurs *FPGA* prend différentes architectures suivant le besoin voici deux exemples d'association :



***Figure(I.13)**: Exemples d'association entre **FPGA**.*

I.6.7.2)- Association d'un microprocesseur et d'un FPGA

La combinaison entre un ***FPGA*** et un processeur est possible dans certains cas comme accélérateurs matériels où le système reconfigurable est directement couplé à un processeur ce qui constitue un *SOC***.** Le microcontroleur ou le microprocesseur est généralement le hote (organise,initialise,charge les programmes…..).

		
Unité intégrée	Périphérique	Co-processeur

***Figure(I.14)**: Exemples d'association entre **FPGA et Microprocesseur**.*

1.6.8)-Avantages et inconvénients des FPGA

Les avantages et les inconvénients des *FPGA* sont multiples on trouve :

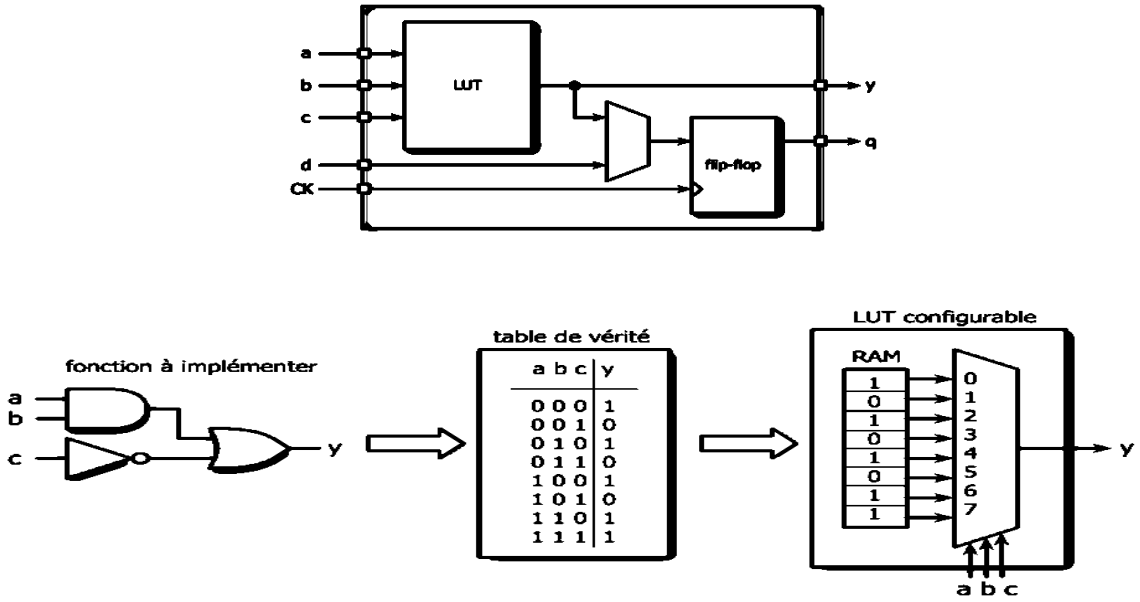
Avantages	Inconvénients
-**T**echnologie « facile » à maîtriser. -**T**emps de développement réduit. -**R**eprogrammable. -Idéal pour le prototypage. -**C**oût peu élevé. -**P**arallélisme de traitement. -**F**lexibilité et la possibilité de réduire fortement les délais de développement et commercialisation. -**L**a reconfiguration, parfois en temps réel.	- **P**erformances non optimisées. -**T**emps de réponse long par rapport aux ***ASIC***.

I.6.9)-Les deux grandes familles architecturales d'FPGA

Les familles des *FPGA* peuvent se regroupées en deux groupes :

I.6.9.1)-Les circuits FPGA à base de « LUT » (Look Up Tables)

Les *LUT* (***L**ook **U**p **T**ables*) ressemblent aux tables de vérité des fonctions logiques et réalisables par des mémoires de type *SRAM*. Aujourd'hui, la structure la plus utilisée est basée sur ce type (Look-Up Table) d'*FPGA*. Les possibilités offertes par les circuits programmables *FPGA a SRAM* permettent par ailleurs de mettre en œuvre le concept de prototypage (ou maquette) pour la vérification fonctionnelle de systèmes sur puce pour certaines applications. La fonction de la *LUT* est de stocker la table de vérité de la fonction combinatoire à implémenter comme le montre la figure suivante.

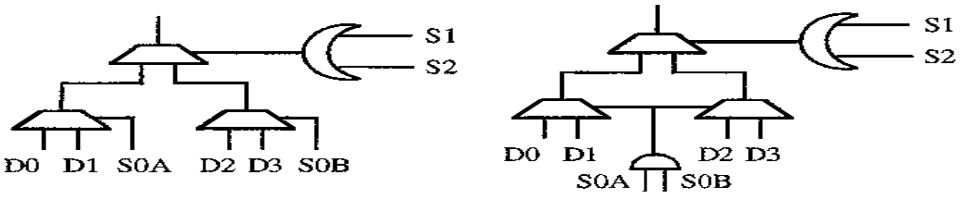


***Figure(I.15)**: Exemple d'implémentation sur **LUT**.*

I.6.9.2)-Les circuits FPGA a base de multiplexeurs « MUX »

Les *FPGA* à base de multiplexeurs qui sont des microcellules a trois entrées capable de réaliser la fonction suivante :

Equation logique	*Symbole*
$F = (a\ AND\ b) OR (\bar{a}\ AND\ b)$	



***Figure(I.16)**: Exemple d'implémentation sur des multiplexeurs.*

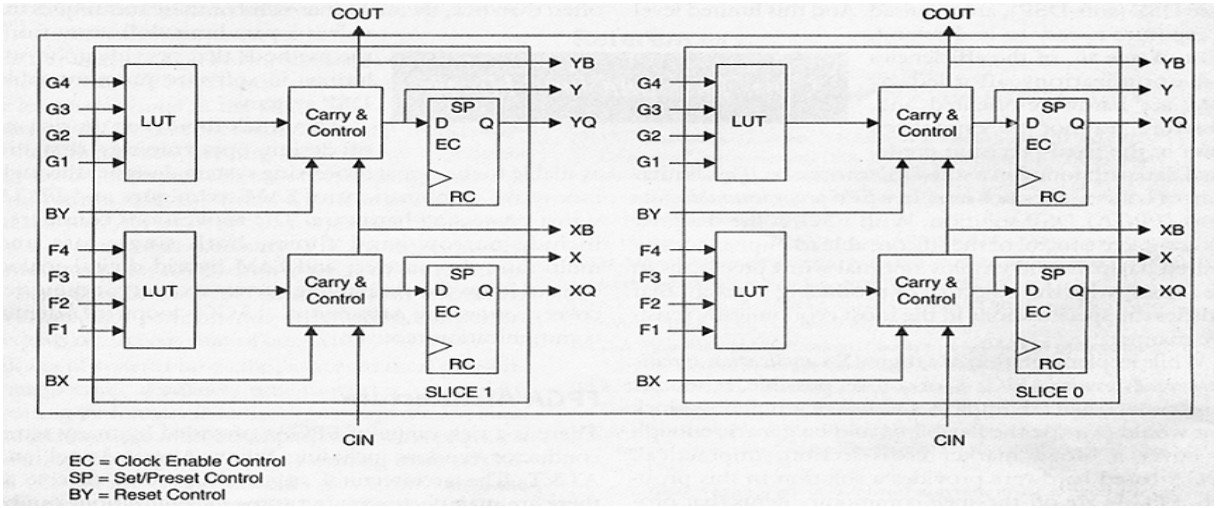
I.6.10)- Technologie du premier fondeur d'FPGA « XILINX »

Le plus gros constructeur du marché est *XILINX* qui a introduit la série *XC2000* (de *600* à *1500* portes) entre *1984* et *1985* avec des fréquences de fonctionnement pouvant atteindre les *420MHZ*. Depuis, d'autres séries sont apparues comme les *XC3000, XC4000, XC5200* et *XC6200*, ainsi que les *XC9500* et la mise sur le marché du 1er *FPGA XILINX* en *1985*.Puis l'apparition en *1992* du premier *FPGA* du constructeur *ALTERA* qui est le concurrent le plus important de *XILINX*, avec un type de circuits assez différent le *FLEX 8000* (*15 000* portes max).Rapidement l'exploitation de la technologie *EEPROM* un an plus tard en *1993* puis le lancement du *VIRTEX II /XILINX* (jusqu'à *10* millions de portes) en *2001* et des *FPGA* de capacités supérieures à 50 millions de portes fonctionnant à des fréquences dépassant les *500 MHz* en *2005***.**

Le principe des *FPGA* de *XILINX* est de stocker la configuration dans une mémoire vive statique *SRAM*. Aujourd'hui des blocs des fonctionnalités supplémentaires dans quelques versions évoluées sont ajoutés et dédiées à des applications spécifiques. Ces fonctionnalités supplémentaires qui sont : Mémoire *RAM*, Petits multiplieurs, Blocs *DSP*, Cœurs de processeurs *RISC* et arbres de distribution d'horloge pour générer différent domaines d'horloge pour un bon synchronisme en cas d'*FPGA* a grande capacité.

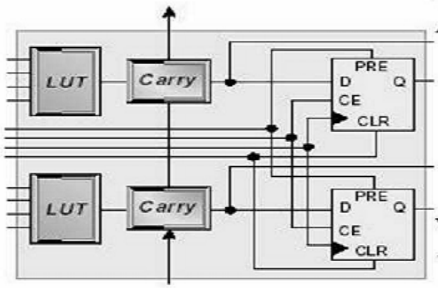
Dans le cas du *Virtex-II 1000*, la matrice de logique programmable est de *40 × 32 = 1280 CLB*, soit *5120* slices, soit enfin *10240* cellules logiques. Les *LUT* sont constitués de quatre slices. Les bascules dans chaque slice sont initialisées par défaut à valeur '*0*' et chaque bascule bénéficie de broches de contrôle. Chaque table *LUT* permet la conception d'une mémoire synchrone *16×1* bits. Les deux *LUT* d'une tranche offrent une mémoire synchrone 16×2 bits, *32×1* bits ou *16×1* bits. Une *LUT* fonctionne également comme un registre à décalage de seize bits.

❖ **Structure des CLB**



***Figure(I.17)**: Architecture d'un **CLB** de familles **Virtex**.*

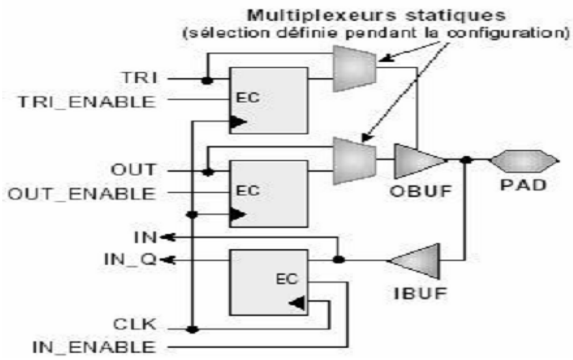
L'architecture simplifiée d'un slice est comme suit :



***Figure(I.18)**: Architecture d'un **SLICE** de familles **Virtex**.*

❖ **Structure des IOB**

Les blocs d'entrées/sorties disposent de bascules aussi de contrôle à trois-états.



***Figure(I.19)**: Architecture d'un **IOB** de familles **Virtex**.*

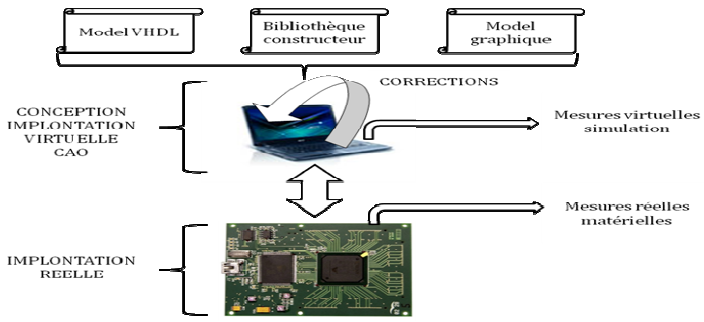
I.7)- LA CONFIGURATION DES FPGA PAR LES OUTILS CAO

I.7.1)- De l'algorithmique à la conception CAO

L'origine du mot algorithme provient du nom latinisé d'*Al-Khawarizmi (Abou Jafar Muhammed Ibn Musa al-Khawarizmi* médecin arabe du moyen âge). Un algorithme est une suite ou séquence de raisonnements réalisé par un nombre fini d'opérations en termes de temps et de support matériel afin de fournir une solution a certains problèmes. Un algorithme est présenté sous forme d'une prescription qui peut avoir différentes formes (textuelle, graphique, formule mathématique, diagramme de séquence…..etc.). La plupart des algorithmes existants sont orientés vers une implémentation logicielle ce qui est contraire à notre application qui s'agit d'une implantation matérielle. L'objectif des algorithmes sont récapitulés comme suit:

- **T**ransmettre un savoir faire.
- **D**écrire les étapes à suivre pour réaliser un travail.
- **E**xpliciter clairement les idées.

Les techniques de conception *CAO (**C**onception **A**ssistée par **O**rdinateur)* sont aujourd'hui très éprouvées et largement employées afin de concevoir des circuits électroniques nécessaires a 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.



***Figure(I.20)**:Mode d'exécution matériel de la **CAO**.*

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.

I.7.2)- Méthodologie de conception

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. Nous pouvons traiter un système complexe de deux manières qui sont :

- **L**'approche dite *« descendante » (ou « **top-down** » en anglais).*
- **L**'approche dite *« ascendante » (ou « **bottom-up** » en anglais).*

REMARQUE: Il y a une grande similitude de conception pour les FPGA, les CPLD et les ASIC.

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

1)-Spécification du design

- **L**e nombre de broches d'entrée-sortie et leur localisation dans la puce *FPGA*.
- **L**a spécification de la fréquence d'horloge du système.
- **L**a spécification de la mémoire requise pour l'application.

2)-Développement du design

- **S**pécification de la méthodologie de design (Outil de développement utilisé).
- **L**a saisie du circuit Codage *RTL* (*VHDL, Verilog*...)
  - **G**raphique *(Machine à états).*
  - **S**aisie HDL *(**H**ardware **D**escription **L**anguage).*
- **L**a simulation (Pré et Post synthèse).

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.

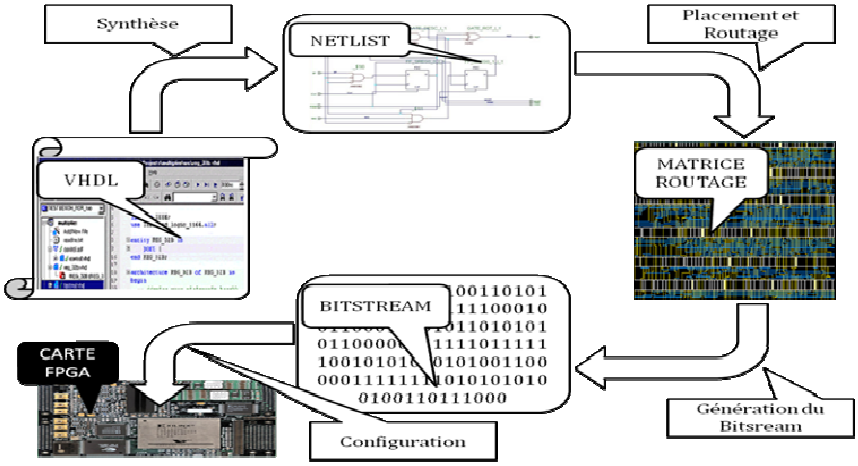
4)-Placement et routage

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 a 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***.

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-a-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 suivante résume un peut l'ensemble de ces étapes.



***Figure(I.21)**:Etapes de conception sur **FPGA**.*

I.8)- L'APPORT DES FPGA A LA COMMANDE DES MACHINES

Sous l'impulsion de progrès rapide et extraordinaire du numérique ; l'amélioration de la qualité et des performances a toujours été une préoccupation constante chez les concepteurs de circuits numériques et c'est dans ce contexte que nous exploitons les *FPGA* afin de faire une contribution à l'amélioration de la commande des machines en temps réel. Les contraintes majeures pour la commande des machines sont la satisfaction du compromis rapidité /précision d'une part et un taux de calculs élevé d'autre part. Une approche efficace pour résoudre ce genre de problème est aujourd'hui disponible et concrétiser par ces circuits logiques *FPGA*. L'introduction des *FPGA* est un remède à la complexité des algorithmes de commande ainsi qu'a la vitesse de traitement.

Le *FPGA* assure toute la partie algorithmique de la commande grâce à ces caractéristiques notamment le parallélisme de traitement. Parmi les caractéristiques de ce circuit que le contrôle des machines électriques peut se bénéficier réside : la possible d'implanter des fonctions avancées irréalisables dans le domaine analogique, aucun impacte des perturbations externes sur les algorithmes, la réalisation de systèmes sûrs et efficaces avec précision, la reprogrammabilité sur site sans changer de composant ni câblage et un encombrement minimal où tout les algorithmes de contrôles sont intégrés sur une puce de quelques millimètres carrés. L'ensemble de ces caractéristiques est un acquis pour la commande des moteurs par une optimisation du rendement des convertisseurs statiques.

Finalement l'apport des *FPGA* sur la commande des machines peut etre résumé dans les qulques points suivant :

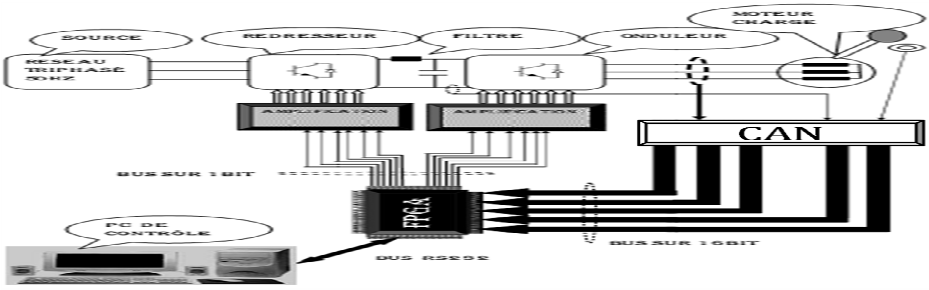
- **L**es *FPGA* sont des solutions numériques qui permettent d'approcher les avantages de l'analogique et de garder au même temps les avantages du numérique avec l'implantation d'algorithmes complexes, un temps de calcul réduit a quelques microsecondes et une bonne précision.

- **E**viter l'inconvénient majeur des solutions analogiques classiques qui réside dans l'influence des variations paramétriques engendré par la sensibilité aux perturbations externes comme la chaleur.
- **P**as d'entretien qui nécessite du temps et des pertes d'ordres économique a l'inverse des solutions analogiques.
- **I**mplémentation de fonctionnalités supplémentaires qui ne sont pas réalisable en continu.
- **A**ugmentation de la bande passante vis a vis des autres solutions numériques comme les *DSP* et microcontrôleurs ou microprocesseur traditionnels.
- **L**'intégration sur une seule puce de plusieurs algorithmes de contrôle grâce à la configuration dynamique avec une grande flexibilité pour un changement de la structure de contrôle
- La possibilité de réduire fortement les délais de développement et de commercialisation.
- **L**'utilisation des *FPGA* dans le contrôle des machines ne nécessite pas d'espace ce qui est équivalent a un encombrement minimale car c'est une technologie embarquée hautement intégrée avec une consommation d'énergie ultra-basse.

Avant l'apparition des *FPGA* qui sont disponibles aujourd'hui, les méthodes classiques utilisées avec *DSP* et microcontrôleur, permettent l'obtention d'un temps de cycle moyen proche de *100 (us)* ce qui est équivalent a une fréquence de commutation aux alentours de *1-5 kHz*. Avec l'introduction des *FPGA* a des prix raisonnables, la fréquence de commutation est devenue aux alentours de *10-15 (KHz)* et aujourd'hui d'ordre de *50(KHz)*. Il est même possible d'obtenir des fréquences aussi élevées que *100 (KHz)* mais malheureusement les limites des dispositifs d'électronique de puissance sont atteintes sachons que l'énergie dissipée dans ces convertisseurs est proportionnelle a la fréquence de commutation ce qui représente une contrainte très pesante.

I.9)- LA STRUCTURE MATERIELLE DE COUPLAGE (FPGA /MACHINE)

La recherche dans le domaine de conception des circuits numériques de commande pour les systèmes automatisés en temps réel est difficile, car elle nécessite non seulement une parfaite maitrise des technologies employées mais aussi une très bonne connaissance des caractéristiques d'application et de la nature de sont environnement. La structure générale de couplage FPGA/Moteur est schématisée dans la figure suivante :



***Figure(I.22)**:Structure de couplage **FPGA**/Moteur triphasée.*

L'élément microélectronique *FPGA* agit comme le « cerveau » du système. Il reçoit des données, les traite, et prend des décisions. Donc, comme on le voie sur la figure précédente, Le *FPGA* va permettre de cadencer les interrupteurs des convertisseurs statiques.

I.10)- CONCLUSION

Ce premier chapitre est dédié à la présentation du contexte général de la thèse et permet d'en déterminer les principaux objectifs. Au départ, nous avons présenté un survol des circuits programmables puis nous avons étudié l'état d'art des *FPGA* ce qui nous a permis de conclure que la technologie *FPGA* s'inscrit au sommet de l'évolution des composants logiques et le besoin croissant de composants plus performants, plus économiques et disponibles en grandes quantités est les grands axes du progrès qui sont disponibles dans les *FPGA. Les FPGA* ouvrent de grandes perspective en matière de contrôle en temps réel. La réalisation d'un système de contrôle en temps réel nécessite aussi une bonne maîtrise des outils fournis par la théorie de l'automatique lors de la phase de modélisation et de simulation, ainsi qu'une bonne maîtrise de l'informatique temps réel lors de la phase d'implantation et l'ensemble de ces point seront traiter au cours des prochains chapitres puis sera plus particulièrement détaillé et appliqué sur un moteur asynchrone triphasé qui est la cible d'application des *FPGA* et sujet de nôtre application.

CHAPITRE II

Modélisation et commande du moteur asynchrone triphasé

OBJECTIF

C'est dans la tendance que le matériel doit être parfaitement connu pour être exploité par un logiciel *CAO* que ce chapitre sera consacré. Donc, dans le contexte de modélisation de l'ensemble convertisseur statique-machine asynchrone en vue de la commande qui sera appliquée sur les grandeurs statoriques que ce chapitre sera entièrement dédié.

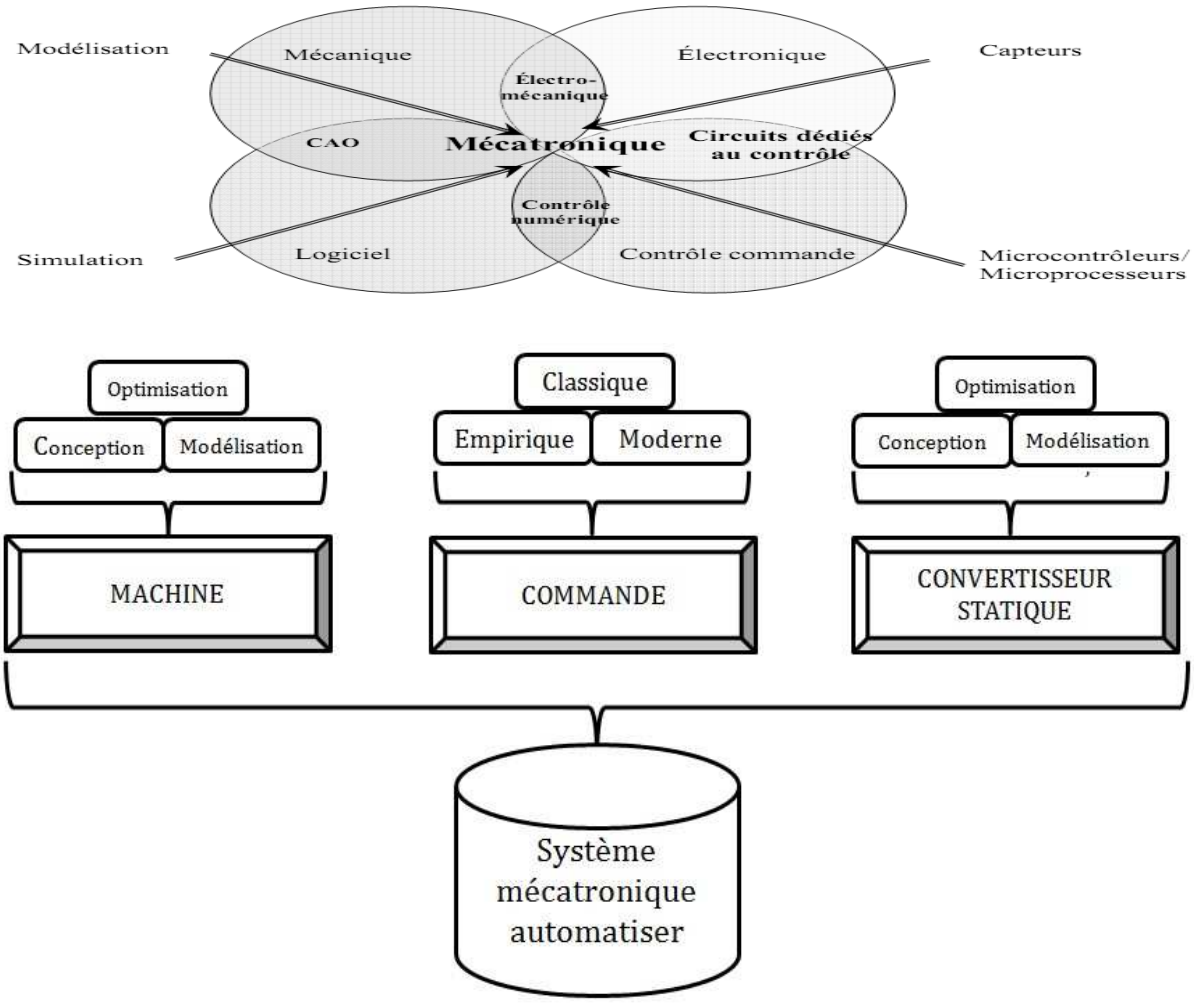
II.1)-INTRODUCTION

Les machines électriques ou machines tournantes sont des dispositifs électriques qui permettent de générer un mouvement ou une énergie mécanique à partir d'une énergie électrique et inversement. Elles occupent une place prépondérante dans tous les secteurs industriels. Cette dernière décennie a vue l'exploitation à grande échèlle de ces machines dans l'industrie ce qui a engendrer un progrès particulier pour ce qui concerne la recherche de techniques de contrôle de ces machines. Les machines électriques tournantes sont classées en trois catégories qui sont les machines à courant continu, les machines synchrones et les machines asynchrones. Le chois de la machine est en relation directe avec le type d'application et de son environnement.

Depuis son invention et sa découverte par *NICOLA TESLA*, la machine asynchrone à attirer une attention particulière des industriels et elle est devenue l'actionneur le plus important parmi les machines tournantes de nos jours. Elle est caractérisée par sa simplicité de conception, de fabrication, d'entretien, de robustesse et peu coûteuse avec un excellent rendement. 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 ce qui est un comportement inverse de sa prédécesseur la machine a courant continu. Le moteur asynchrone a des avantages assez connus et assez nombreux. Par contre, il est difficile et onéreux d'en faire une variation de sa vitesse lors du fonctionnement. C'est la raison pour laquelle, le moteur à courant continu est resté longtemps le plus utilisé dans les entraînements à vitesse variable. Avec le développement de l'industrie et les techniques de commande numérique en temps réel, le moteur asynchrone suscite de plus en plus d'intérêts. Nous consacrerons ce chapitre au moteur asynchrone triphasé qui est notre application. Le principe de fonctionnement est d'abord étudié puis les éléments de contrôle et les boucles de régulation seront décrits et finalement nous présenterons la commande qui pourrait être implantable sur les *FPGA*.

II.2)-L'ENVIRONNEMENT MECATRONIQUE

Les systèmes mécatroniques sont des systèmes complexes faisant intervenir des technologies différentes dans le but de construire un système automatisé. Ce dernier est subdivisé en deux sous systèmes complémentaires qui sont : « un système contrôlé » et « un système de contrôle ». Le système contrôlé composé d'une partie matérielle (Moteur/Variateur) et le système de contrôle ou de commande, composé d'une partie matérielle et une autre logicielle (Calculateur pour exécuter l'algorithme de contrôle comme exemple « *FPGA ; CAO* »). Ces deux parties du système automatisé communiquent entre elles par l'intermédiaire de capteurs pour acquérir l'état de cet environnement. L'étude d'un système comme celui-ci nécessite un développement de modèles complexes, fortement multidisciplinaires et afin d'illustrer ces propos, le schéma suivant résume l'ensemble de ces liens :



Figure(II.01): L'environnement mécatronique des machines.

II.3)-DESCRIPTION DU MOTEUR ASYNCHRONE

Le moteur asynchrone triphasé est le plus utilisé pour assurer la variation de vitesse et du couple dans les mécanismes industriels. Il est constitué du stator et du rotor avec trois enroulements (bobines) parcourus par des courants alternatifs triphasés qui représentent le stator et un autre ensemble de trois enroulements qui sont court-circuités et qui forme le rotor. Et dans une machine asynchrone à 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. Les courants alternatifs dans le stator créent un champ magnétique tournant à la pulsation de synchronisme et le rotor tourne à une vitesse plus petite que la vitesse de synchronisme .On dit que le rotor «"glisse"» par rapport au champ tournant et ce glissement dépend de la charge.

II.4)- AVANTAGES ET INCONVENIENTS DU MOTEUR ASYNCHRONE

Les avantages et les inconvénients du moteur asynchrone sont assez nombreux mais les principaux sont résumés dans le tableau suivant :

Avantages	Inconvénients
-**L**a robustesse. -**L**a simplicité de construction. -**L**eurs bas coûts. -**U**n rendement excellent.	-**N**on découplage naturel. -**N**on linéarités.

II.5)-MODELISATION DU MOTEUR ASYNCHRONE

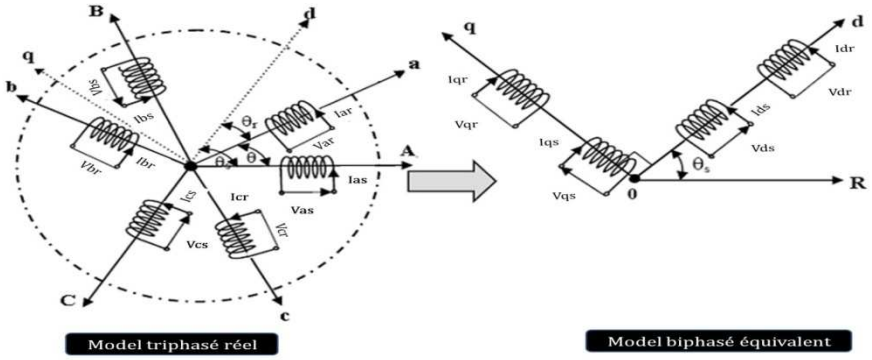
L'établissement des diverses relations qui constituent le système est en vue de la commande de ce dernier car la conception d'un système de commande en temps réel passe nécessairement par une étape de modélisation. La théorie généralisée des machines électriques triphasées classiques, est basée sur la transformation de ***Park*** qui rapporte les équations électriques statoriques et rotoriques à deux axes perpendiculaires électriquement (direct et en quadrature). L'étude et l'analyse du moteur asynchrone consiste a obtenir un modèle mathématique représentatif du fonctionnement de ce moteur et qui permet de prévoir le comportement de ce système et l'évolution des variables d'état de ce dernier sous l'action d'un événement particulier avec une prise 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.

II.5.1)-Hypothèses simplificatrices

Avant d'établir le modèle de la machine asynchrone en vue de sa commande, nous rappelons brièvement les hypothèses simplificatrices désormais classiques, retenues. Les hypothèses faites sont les suivantes :

- Une parfaite symétrie.
- Le système à trois phases est considéré équilibré.
- Non saturation des circuits magnétiques.
- Les pertes fer sont négligées.
- Il n'y a pas d'effet de peau.
- L'effet des encoches est négligé.
- La répartition de la force magnétomotrice est sinusoïdale.

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



Figure(II.02): Représentation schématique de l'ensemble (stator/ rotor).

Définition des angles :

$$(O_{as};O_d)=\theta_s\ ;\ \ (O_{ar};O_d)=\theta_r\ ;$$

$$(O_{as}-O_{ar})=\theta_s-\theta_r=\theta;\ \left(O_q;O_d\right)=\frac{\pi}{2}$$

$$(O_{bs};O_d)=\theta_s-\frac{2\pi}{3};\ (O_{br};O_d)=\theta_r-\frac{2\pi}{3}$$

$$(O_{cs};O_d)=\theta_s-\frac{4\pi}{3}\ \ (O_{cr};O_d)=\theta_r-\frac{4\pi}{3}$$

$$\omega_r=\frac{d\theta_s}{dt}-\frac{d\theta_r}{dt}=\frac{d\theta}{dt}$$

Nous aurons dans ce qui suit un aperçu sur les relations mathématiques se retrouvent dans la littérature technique de modélisation consacrée aux machines électriques triphasées asynchrones a rotor bobiné.

II.5.2)-Equations électriques sur les axes « a,b,c »

Les tensions statoriques et tensions rotoriques dans une machine asynchrone sont données d'après la loi de *FARADY* comme suit :

$$\begin{cases} v_{as} = -\frac{d\varphi_{as}}{dt} - R_s.i_{as} \\ v_{bs} = -\frac{d\varphi_{bs}}{dt} - R_s.i_{bs} \\ v_{cs} = -\frac{d\varphi_{cs}}{dt} - R_s.i_{cs} \end{cases} \qquad \begin{cases} v_{ar} = -\frac{d\varphi_{ar}}{dt} + R_r.i_{ar} \\ v_{br} = -\frac{d\varphi_{br}}{dt} + R_r.i_{br} \\ v_{cr} = -\frac{d\varphi_{cr}}{dt} + R_r.i_{cr} \end{cases} \qquad \textbf{EQ (II: 01)}$$

II.5.3)-Ecriture matricielle des équations électriques

$$(v_s) = -\frac{d(\varphi_s)}{dt} - [\boldsymbol{R_s}](i_s) \qquad (v_r) = -\frac{d(\varphi_r)}{dt} - [\boldsymbol{R_r}](i_r) \qquad \textbf{EQ (II: 02)}$$

Telque :

$$(v_s) = (v_{as}\ \ v_{bs}\ \ v_{cs})^t \qquad (v_r) = (v_{ar}\ \ v_{br}\ \ v_{cr})^t$$

$$(i_s) = (i_{as}\ \ i_{bs}\ \ i_{cs})^t \qquad (i_r) = (i_{ar}\ \ i_{br}\ \ i_{cr})^t$$

$$[R_s] = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \qquad [R_r] = \begin{bmatrix} R_r & 0 & 0 \\ 0 & R_r & 0 \\ 0 & 0 & R_r \end{bmatrix}$$

II.5.4)-Equation mécanique

Le comportement mécanique de la machine asynchrone dépend de l'inertie J, du couple électromagnétiqueCe, du couple mécanique résistant Cr et du couple de frottement $C_f(t) = f_v\Omega(t)$ ou f_v est la constante de frottement. L'équation mécanique est donnée par :

$$J\frac{d\Omega(t)}{dt} + f_v\Omega(t) = Ce(t) - Cr(t) \qquad \textbf{EQ (II: 03)}$$

Le modèle de représentation de la machine asynchrone que nous venons de présenter 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 magnétiques contiennent des

éléments variables en fonction de l'angle de rotation $\boldsymbol{\theta}$. Une solution pour obtenir des coéfficients constants consiste à appliquer une transformation mathématique au système et cette transformation est plus connue sous le nom de transformation de *Park*.

II.5.5)-Transformation de « PARK »

L'objectif primordiale de la transformé de *Park* est rendre le système (modèle de la machine) linéaire. On utilise alors cette transformation mathématique qui permet de décrire le comportement de la machine à l'aide d'équations différentielles à coefficients constants. Elle consiste à faire des projections sur deux axes orthogonaux de toutes les grandeurs du système. La condition qui permet de remplacer la machine triphasée par son modèle biphasé est sa symétrie. La matrice de ***Park*** et sont inverse sont les suivantes:

$$P(\theta_s) = \frac{2}{3}\begin{bmatrix} cos(\theta_s) & cos\left(\theta_s - \frac{2\pi}{3}\right) & cos\left(\theta_s - \frac{4\pi}{3}\right) \\ sin(\theta_s) & sin\left(\theta_s - \frac{2\pi}{3}\right) & sin\left(\theta_s - \frac{4\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad \textbf{EQ (II: 04)}$$

$$P^{-1}(\theta_s) = \frac{2}{3}\begin{bmatrix} cos(\theta_s) & cos\left(\theta_s - \frac{2\pi}{3}\right) & 1 \\ sin(\theta_s) & sin\left(\theta_s - \frac{2\pi}{3}\right) & 1 \\ cos\left(\theta_s - \frac{4\pi}{3}\right) & sin\left(\theta_s - \frac{4\pi}{3}\right) & 1 \end{bmatrix} \qquad \textbf{EQ (II: 05)}$$

Cette transformation triphasée-biphasée qui permet de modéliser chaque grandeur par deux coordonnées au lieu de trois par conséquent une réduction du nombre d'équations nécessaires a la modélisation du système.

La transformation de *Park* pour les courants est :

$$(i_{ps}) = P(\theta_s)(i_s) \qquad (i_s) = P^{-1}(\theta_s)(i_{ps})$$

Avec :

$(i_s) = (i_{as}\ \ i_{bs}\ \ i_{cs})^t$

$Composantes\ a, b, c\ du\ courant\ statorique$

$(i_{ps}) = (i_{ds}\ \ i_{qs}\ \ i_{0s})^t$

$Composantes\ de\ Parck(d, q; 0)\ du\ courant\ statorique$

REMARQUE : Cette transformation est valable aussi pour les tensions et les flux. Pour les grandeurs rotoriques, il suffit de remplacer l'indice « s » par l'indice « r ».

II.5.6)- Equations magnétiques sur les axes « a,b,c »

L'absence de saturation et la limitation au premier harmonique d'espace nous permettent d'écrire les expressions des flux statoriques et rotoriques comme suit :

$$\begin{cases}(\varphi_s) = [L_{ss}](i_s) + [M_{sr}](i_r) \\ (\varphi_r) = [L_{rr}](i_r) + [M_{rs}](i_r)\end{cases} \qquad \textbf{EQ (II: 06)}$$

Où :

$$(\varphi_s) = (\varphi_{as}\ \ \varphi_{bs}\ \ \varphi_{cs})^t \qquad (\varphi_r) = (\varphi_{ar}\ \ \varphi_{br}\ \ \varphi_{cr})^t$$

$$(i_s) = (i_{as}\ \ i_{bs}\ \ i_{cs})^t\ \ (i_r) = (i_{ar}\ \ i_{br}\ \ i_{cr})^t$$

Les quatre sous matrices d'inductances s'écrivent :

$$[\boldsymbol{L}_{ss}] = [L_s] = \begin{bmatrix} l_{as} & m_{as} & m_{as} \\ m_{as} & l_{as} & M_{as} \\ m_{as} & m_{as} & l_{as} \end{bmatrix} \qquad [\boldsymbol{L}_{rr}] = [L_r] = \begin{bmatrix} l_{ar} & m_{ar} & m_{ar} \\ m_{ar} & l_{ar} & M_{ar} \\ m_{ar} & m_{ar} & l_{ar} \end{bmatrix}$$

$$[\boldsymbol{M}_{sr}] = M_{rs}\begin{bmatrix} cos\,\theta & cos\left(\theta - \frac{4\pi}{3}\right) & cos\left(\theta - \frac{2\pi}{3}\right) \\ cos\left(\theta - \frac{2\pi}{3}\right) & cos\,\theta & cos\left(\theta - \frac{4\pi}{3}\right) \\ cos\left(\theta - \frac{4\pi}{3}\right) & cos\left(\theta - \frac{2\pi}{3}\right) & cos\,\theta \end{bmatrix}$$

$$[\boldsymbol{M}_{rs}] = [\boldsymbol{M}_{sr}]^t$$

Avec les différentes inductances qui désignent:

L_{as}: Inductance propre d'une phase statorique.

L_{ar}: Inductance propre d'une phase rotorique.

M_{as}: Inductance mutuelle entre deux phases statoriques.

M_{ar}: Inductance mutuelle entre deux phases rotoriques.

M_{rs}: Inductance mutuelle maximale entre une phase statorique et une phase rotorique.

On remarque que chaque composante du flux magnétique est une combinaison ou interaction des courants de toutes les phases.

Dans ce qui suit, nous allons donner les équations électriques de la *MAS* dans le système biphasé en appliquant la transformation de Park aux équations.

II.5.7)- Equations magnétiques sur les axes « d »et « q »

C'est a partir de ce niveau de modélisation que commence à apparaitre l'utilité de la transformée de *Park* sur les équations des flux. Pour les flux statoriques et avec projection sur les axes « $\boldsymbol{d}$ »et « $\boldsymbol{q}$ » on aura :

$$(\varphi_s) = [\boldsymbol{L}_{ss}](i_s) + [\boldsymbol{M}_{sr}](i_r)$$

$$P(\theta_s)^{-1}(\varphi_{ps}) = [\boldsymbol{L}_{ss}]P(\theta_s)^{-1}(i_{ps}) + [\boldsymbol{M}_{sr}]P(\theta_s)^{-1}(i_{pr})$$

On multiplie par $P(\theta_s)$:

$$(\varphi_{ps}) = \{P(\theta_s)[\boldsymbol{L}_{ss}]P(\theta_s)^{-1}\}(i_{ps}) + \{P(\theta_s)[\boldsymbol{M}_{sr}]P(\theta_s)^{-1}(i_{pr})\,(\varphi_{ps})\} \quad \textbf{(II:07)}$$

Pour les flux rotoriques la même chose on aura :

$$(\varphi_r) = [\boldsymbol{L}_{rr}](i_r) + [\boldsymbol{M}_{rs}](i_s)$$

$$P(\theta_r)^{-1}(\varphi_{pr}) = [\boldsymbol{L}_{rr}]P(\theta_r)^{-1}(i_{pr}) + [\boldsymbol{M}_{rs}]P(\theta_s)^{-1}(i_{ps})$$

On multiplie par $P(\theta_r)^{-1}$:

$$(\varphi_{pr}) = \{P(\theta_r)[\boldsymbol{L}_{rr}]P(\theta_r)^{-1}\}(i_{pr}) + \{P(\theta_r)[\boldsymbol{M}_{rs}]P(\theta_s)^{-1}\}(i_{ps}) \quad \textbf{EQ (II: 08)}$$

En effectuant les quatre produits matriciels on trouve :

$$\begin{pmatrix}\boldsymbol{\varphi_{ds}}\\ \boldsymbol{\varphi_{dr}}\end{pmatrix} = \begin{bmatrix}L_s & M\\ M & L_r\end{bmatrix}\begin{pmatrix}i_{ds}\\ i_{dr}\end{pmatrix} \qquad \begin{pmatrix}\varphi_{qs}\\ \varphi_{qr}\end{pmatrix} = \begin{bmatrix}L_s & M\\ M & L_r\end{bmatrix}\begin{pmatrix}i_{qs}\\ i_{qr}\end{pmatrix} \quad \textbf{EQ (II: 09)}$$

$\varphi_{0s} = L_{0s}i_{0s}$ $\qquad$ $\varphi_{0r} = L_{0r}i_{0r}$

Avec :

$L_s = L_{as} - M_{as}$: Inductance propre cyclique du stator.

$L_r = L_{ar} - M_{ar}$: Inductance propre cyclique du rotor.

$M = \frac{3}{2} M_{rs}$: Inductance mutuelle cyclique entre stator et rotor.

$L_{0s} = L_{as} + 2M_{as} \quad L_{0r} = L_{ar} + 2M_{ar}$

On remarque que d'après l'équation des flux précédente que les matrices inductance deviennent diagonale avec peut d'élément et ces derniers ne dépend plus de $\boldsymbol{\Theta}$ d'où l'utilité de la transformée de *Park* a ce niveau.

II.5.8)- Equations électriques sur les axes « d »et « q »

Pour les flux statoriques et avec projection sur les axes « ***d*** »et « ***q*** » on aura :

$$(v_s) = -\frac{d\varphi_s}{dt} - [\boldsymbol{R}_s](i_s)$$

$$\left(P(\theta_s)^{-1} v_{ps}\right) = -\frac{d\left(P(\theta_s)^{-1}\varphi_{ps}\right)}{dt} - [\boldsymbol{R}_s]\left(P(\theta_s)^{-1} i_{ps}\right)$$

$$\left(P(\theta_s)^{-1} v_{ps}\right) = -P(\theta_s)^{-1}\frac{d\left(\varphi_{ps}\right)}{dt} - \frac{d(P(\theta_s)^{-1})}{d\theta_s}\frac{d\theta_s}{dt}\varphi_{ps} - [\boldsymbol{R}_s]\left(P(\theta_s)^{-1} i_{ps}\right)$$

$$\left(v_{ps}\right) = \frac{d(\varphi_{ps})}{dt} + \left\{P(\theta_s)\frac{d(P(\theta_s)^{-1})}{d\theta_s}\right\}\frac{d\theta_s}{dt}\varphi_{ps} + [\boldsymbol{R}_s] i_{ps} \qquad \textbf{EQ (II: 10)}$$

Idem pour le flux rotoriques la même chose on aura :

$$(v_r) = -\frac{d\varphi_r}{dt} - [\boldsymbol{R}_r](i_r)$$

$$\left(P(\theta_r)^{-1} v_{pr}\right) = -\frac{d\left(P(\theta_r)^{-1}\varphi_{pr}\right)}{dt} - [\boldsymbol{R}_r]\left(P(\theta_r)^{-1} i_{pr}\right)$$

$$\left(P(\theta_r)^{-1} v_{pr}\right) = -P(\theta_r)^{-1}\frac{d\left(\varphi_{pr}\right)}{dt} - \frac{d(P(\theta_r)^{-1})}{d\theta_r}\frac{d\theta_r}{dt}\varphi_{pr} - [\boldsymbol{R}_r]\left(P(\theta_r)^{-1} i_{pr}\right)$$

$$\left(v_{pr}\right) = \frac{d(\varphi_{pr})}{dt} + \left\{P(\theta_r)\frac{d(P(\theta_r)^{-1})}{d\theta_r}\right\}\frac{d\theta_r}{dt}\varphi_{pr} + [\boldsymbol{R}_r] i_{pr} \qquad \textbf{EQ (II: 11)}$$

On effectue les produits matriciels :

$$P(\theta_s)\frac{d(P(\theta_s)^{-1})}{d\theta_s} = P(\theta_r)\frac{d(P(\theta_r)^{-1})}{d\theta_r} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Finalement :

$$\begin{cases} v_{ds} = -\frac{d\varphi_{ds}}{dt} - \varphi_{qs}\frac{d\theta_s}{dt} - R_s i_{ds} \\ v_{qs} = -\frac{d\varphi_{qs}}{dt} - \varphi_{ds}\frac{d\theta_s}{dt} - R_s i_{qs} \\ v_{0s} = -\frac{d\varphi_{0s}}{dt} - R_s i_{0s} \end{cases} \quad \textit{EQ (II: 12)}$$

$$\begin{cases} v_{dr} = -\frac{d\varphi_{dr}}{dt} + \varphi_{qr}\frac{d\theta_r}{dt} + R_r i_{dr} \\ v_{qr} = -\frac{d\varphi_{qr}}{dt} - \varphi_{dr}\frac{d\theta_r}{dt} + R_r i_{qr} \\ v_{0r} = \frac{d\varphi_{0r}}{dt} + R_r i_{0r} \end{cases} \quad \textit{EQ (II: 13)}$$

A ce stade, nous avons exprimé les équations électriques de la machine sur le nouveau repère (***d***,***q***) qui sont très importante pour la commande qu'on va abordé plus tard.

II.5.9)-Le chois du repère pour exprimer le modèle

Le repère diphasé orthonormé (***d,q***) peut être fixe ou tournant par rapport aux armatures de la machine. Judicieusement, il existe trois systèmes d'axes de référence ayant des spécificités distinctes et intéressantes :

- Si le référentiel est fixe par rapport au stator $\omega_s = 0$, on obtient un système électrique où les grandeurs statoriques sont purement alternatives et avec la fréquence d'alimentation. La simulation de la machine asynchrone dans ce repère n'exige donc aucune connaissance de la position du rotor, ce qui constitue un avantage pour la commande sans capteur de position. L'inconvénient majeur est la manipulation de signaux à fréquence élevée.
- Si le référentiel tourne à la vitesse de synchronisme $\omega = \omega_s = 2\pi f_s$, on obtient un système électrique purement continu qui est très bien adapté aux techniques d'identification. Cependant la position du champ tournant doit être reconstituée à chaque instant d'échantillonnage.
- Si le référentiel est fixe par rapport au rotor $\omega_r = \omega$, les signaux électriques sont alors quasi-continus. La pulsation des grandeurs électriques est alors égale à $g.\omega$ (où $g = \frac{\omega - \omega_r}{\omega}$ est le glissement de la machine) qui est faible dans les conditions réelles de fonctionnement. Lorsqu'on a accès à la position mécanique, ce repère est privilégié du fait de la quasi-continuité des grandeurs électriques.

II.5.9.A)- Le repère lié au stator

Les équations liées au référenciel immobile par rapport au stator sont :

$$\begin{cases} \frac{d\theta_s}{dt} = 0 \ ==> \frac{d\theta_r}{dt} = -\omega_r \\ v_{ds} = -\frac{d\varphi_{ds}}{dt} - R_s i_{ds} \\ v_{qs} = -\frac{d\varphi_{qs}}{dt} - R_s i_{qs} \end{cases} \begin{cases} v_{dr} = -\frac{d\varphi_{dr}}{dt} - \varphi_{qr}\omega_r + R_r i_{dr} \\ v_{qr} = -\frac{d\varphi_{qr}}{dt} + \varphi_{dr}\omega_r + R_r i_{qr} \end{cases} \quad \textit{EQ (II: 14)}$$

II.5.9.B)- Le repère lie au rotor

Les équations liées au référenciel immobile par rapport au rotor sont :

$$\begin{cases} \frac{d\theta_s}{dt} = \omega_r \ ==> \frac{d\theta_r}{dt} = 0 \\ v_{ds} = -\frac{d\varphi_{ds}}{dt} - \varphi_{qs}\omega_r - R_s i_{ds} \\ v_{qs} = -\frac{d\varphi_{qs}}{dt} + \varphi_{ds}\omega_r - R_s i_{qs} \end{cases} \begin{cases} v_{dr} = -\frac{d\varphi_{dr}}{dt} + R_r i_{dr} \\ v_{qr} = -\frac{d\varphi_{qr}}{dt} + R_r i_{qr} \end{cases} \quad \textit{EQ (II: 15)}$$

II.5.9.C)- Le repère lié au champ tournant

Les équations liées au référenciel immobile par rapport au chant tournant sont:

$$g = \frac{\omega - \omega_r}{\omega} = glissement$$

$$\frac{d\theta_s}{dt} = \omega ==> \frac{d\theta_r}{dt} = g\omega$$

$$\begin{cases} v_{ds} = -\frac{d\varphi_{ds}}{dt} - \varphi_{qs}\omega - R_s i_{ds} \\ v_{qs} = -\frac{d\varphi_{qs}}{dt} + \varphi_{ds}\omega - R_s i_{qs} \end{cases} \begin{cases} v_{dr} = -\frac{d\varphi_{dr}}{dt} + \varphi_{qr}.g.\omega + R_r i_{dr} \\ v_{qr} = -\frac{d\varphi_{qr}}{dt} - \varphi_{dr}.g.\omega + R_r i_{qr} \end{cases} \quad \textit{EQ (II: 16)}$$

I.5.10)-Expression du couple électromagnétique instantané

L'expression du couple électromagnétique peut être obtenue à partir de la dérivée d' énergie magnétique par rapport à l'angle θ ou par le bilan de puissance. Le couple peut être réglé entre autres par le flux statorique (via la tension d'alimentation par exemple) ou la pulsation ω_s d'alimentation et le couple s'écrit :

$$C_{em} = [I_s]^t \frac{\partial[M_{sr}(\theta)]}{\partial\theta}[I_r] \quad \textit{EQ (II: 17)}$$

Is et Ir désignent respectivement les courants statoriques et rotoriques.

Dans le modèle de Park, l'expression du couple s'écrit sous une forme parmis ces formes équivalentes:

$$C_{em} = p\left(\varphi_{ds} i_{qs} - \varphi_{qs} i_{ds}\right) = p\left(\varphi_{qr} i_{dr} - \varphi_{dr} i_{qr}\right) = pM\left(i_{qs} i_{dr} - i_{ds} i_{qr}\right)$$

$$C_{em} = p\frac{M}{L_r}\left(\varphi_{dr} i_{qs} - \varphi_{qr} i_{ds}\right) \qquad \textbf{EQ (II: 18)}$$

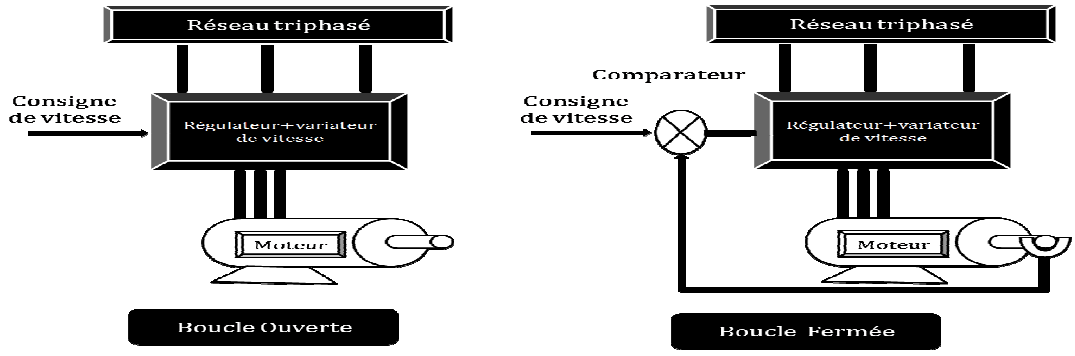
Où : p est le nombre de paires de pôles.

Ces formules sont très intéressantes pour la commande vectorielle avec découplage des grandeurs. Elles nous permis d'inspirer une commande analogue à celle du moteur à courant continu avec quelques considérations a prendre pour le chois du repère (***d,q***) et son orientation suivant l'axe du flux magnétique et qui sera traiter en détaille à la fin de ce chapitre dans la commande vectorielle. Pour plus de détails et plus d'informations sur la modélisation voir la bibliographie.

II.6)-COMMANDE EN BOUCLE OUVERTE OU FERMEE

La commande de ces systèmes est réalisée par deux manières qui sont :

- Variation directe de vitesse (onduleur commandé en boucle ouverte).
- Régulation de la vitesse (onduleur commandé en boucle fermée).



Figure(II.03) : Schéma d'alimentation et de commande du moteur.

La régulation de ces systèmes se fait à l'heure actuelle systématiquement de manière numérique. Les signaux d'entrée-sortie sont discrétises (échantillonnage, blocage), aussi bien dans l'espace des valeurs que dans le temps. Mais, on ne peut parler de la commande de la machine asynchrone, sans qu'on parle convertisseur qui lui est associé, de son alimentation et de sa commande.

II.7)- PRINCIPE DES CONVERTISSEURS STATIQUES

Le convertisseur d'électronique de puissance est un élément statique positionné entre deux sous-systèmes électriques et constitué d'une matrice d'interrupteurs. Ces deux sous systèmes sont un générateur d'énergie électrique et un récepteur de cette dernière. Le transfert d'énergie entre les bornes de sortie du générateur et les bornes d'entrée du récepteur est établi par les connexions assuré par ces interrupteurs à semi-conducteurs (*GTO*, Transistor *MOS, IGBT, etc*.) de puissance. Les signaux nécessaires à la commande des interrupteurs sont élaborés par une électronique de commande et de régulation qui détermine les intervalles de conduction(La séquence de conduction et blocage) des interrupteurs à partir de consignes reçues de l'extérieur et des mesures prélevées sur l'état du système.

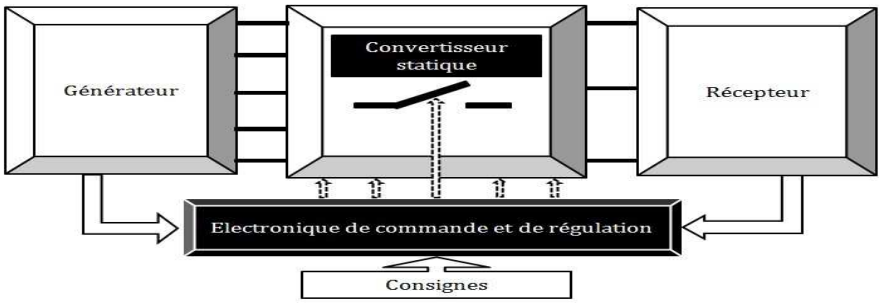


Figure (II.04) : Schéma du principe de commande des convertisseurs statiques.

II.8)-LA FONCTION DE MODULATION MLI ou PWM

II.8.1)-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 distincts qui sont :

- **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 de 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.

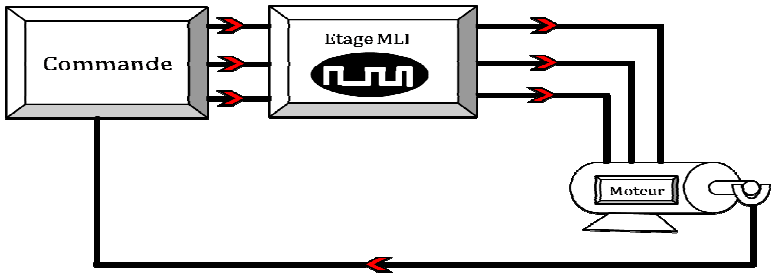


Figure (II.05) : Schéma de position de ***MLI*** sur la chaine de régulation du moteur.

L'ensemble non-exhaustif de ces stratégies sont :

- **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 a 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.8.2)-La fonction de modulation MLI

La technique de modulation en largeur d'impulsions *MLI* (**M**odulation de **L**argeur d'**I**mpulsions ou *PWM* **P**ulse **W**idth **M**odulation, en anglais) est l'essor et le fruit du développement l'électronique de puissance à la fin du siècle dernier. Elle est le cœur du contrôle des convertisseurs statiques. Le chois de la technique *MLI* pour contrôler l'onduleur de tension est en vue d'avoir une réponse rapide et des performances élevées. Le chois de la technique 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 à implanter grossièrement du coût et performances désirées. La *MLI* est composés d'impulsions dont la largeur dépend des choix effectués pour la stratégie de modulation.

Il existe plusieurs types de méthodes ou fonctions *MLI* et une description non-exhaustive de l'ensemble de ces stratégies sont résumés comme suit :

- **MLI Intersective (modulation par porteuses)** :

C'est une stratégie *MLI* triphasée « *classique* » simple à réaliser en analogique initialement conçues en monophasé et son implantation numérique est plus compliquée tel que un grand nombre d'échantillons de la modulante doit être sauvegardé dans une mémoire *ROM* pour pouvoir obtenir une bonne précision du signal modulé. Sont principe est simple avec une simple comparaison, pour chaque bras, entre un signal de référence (*la modulante*) et un signal triangulaire « *dent de scie* » de fréquence plus élevée (*la porteuse*). 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. Parmi les variantes de la *MLI* intersective et la plus populaire la modulation sinusoïdale « *Modulation sinus-triangle SPWM (Sinusoïdal PWM)* ».

- **MLI Précalculée(Modulation pré-calculée):**

Le développement des technologies numériques permet le recours à des stratégies de modulation triphasée spécifiques, non déduites des techniques analogiques .Elle est appelée aussi la technique directe numérique (*DDT – **D**irect **D**igital **T**echnique*) ou technique sans porteuse. Son principe est de générer des impulsions grâce à des séquences préalablement calculées et stockées dans une mémoire.

- **MLI Vectorielle (modulation poste calculer**):

La modulation vectorielle (***S**pace **V**ector **M**odulation, en anglais*) est une technique numérique. Les ordres de commutation des interrupteurs sont déterminés par un algorithme et sont calculés analytiquement à travers des équations mathématiques avec un vecteur tension de contrôle est calculé globalement et approximé sur une période de modulation, par un vecteur tension moyen, puis les ordres de commande adéquats sont appliqués aux interrupteurs. Contrairement à d'autres méthodes, la *MLI* vectorielle ne s'appuie pas sur des calculs séparés des modulations pour chacun des bras de l'onduleur afin d'obtenir en valeur moyenne une tension de référence à partir des états de commutation de l'onduleur et en fin les vecteurs à appliquer et les temps d'application de ces vecteurs.

Les deux *MLI* les plus utilisées sont *PWM* sinusoïdale et *PWM* vectorielle. Cette dernière est certainement la méthode de *MLI* la mieux adaptée au contrôle des moteurs asynchrones. On va adopter *MLI* Vectorielle (*SVPWM*) dans la suite pour le contrôle du moteur asynchrone et cette méthode diffère par rapport à d'autres techniques par le fait que les signaux de commandes sont élaborés en tenant compte de l'état des trois bras de l'onduleur en même temps. Nous tenons à signaler que la *SVPWM* est très lourde aux circuits numériques classiques mais aujourd'hui une amélioration est remarquable avec l'avènement des *FPGA* de dernière génération permettant d'implémenter la *SVPWM* de manière efficace.

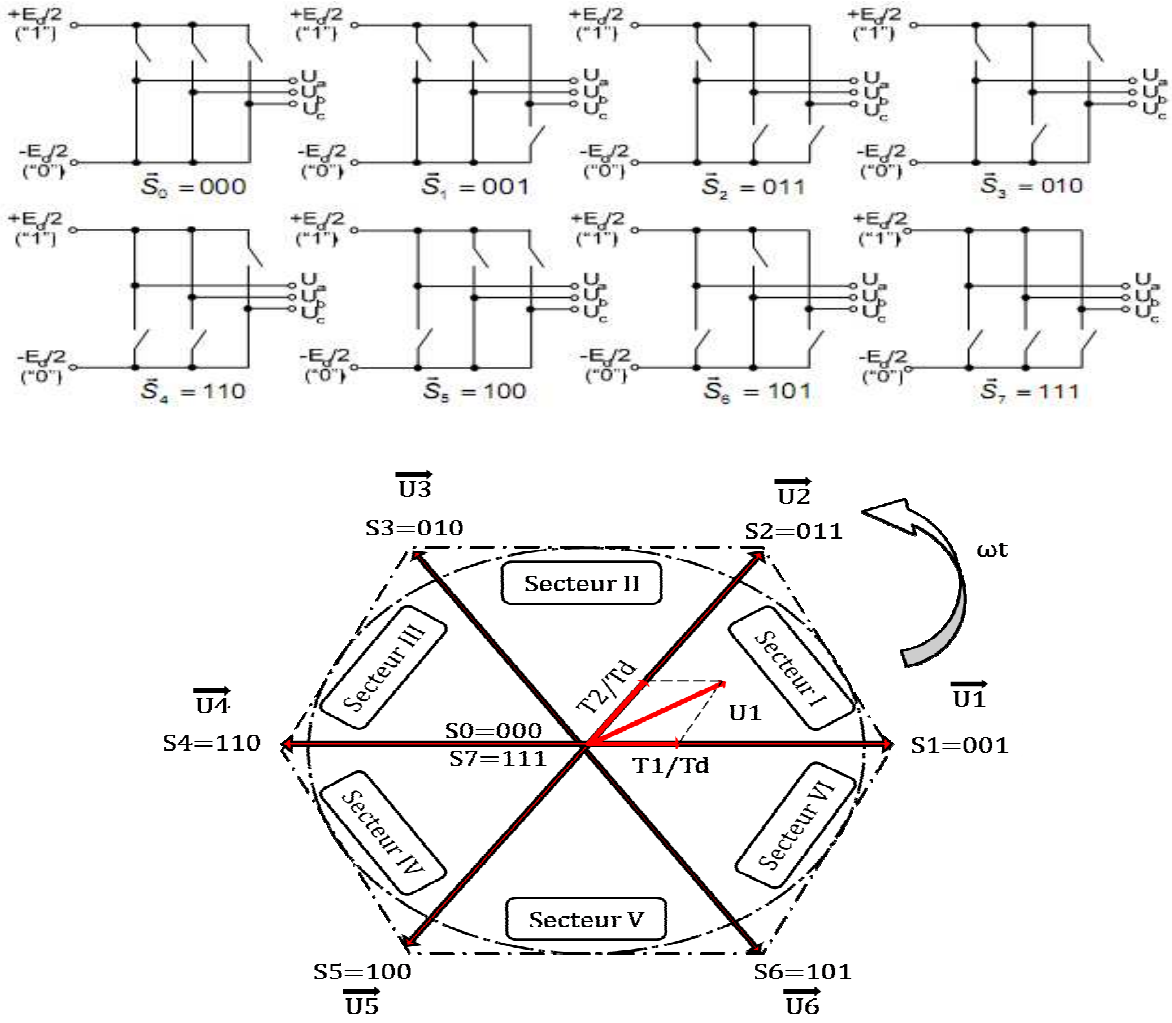


Figure (II.06) : Schéma descriptif de la ***MLI*** vectorielle.

Les vecteurs à appliquer sont déterminés comme suit :

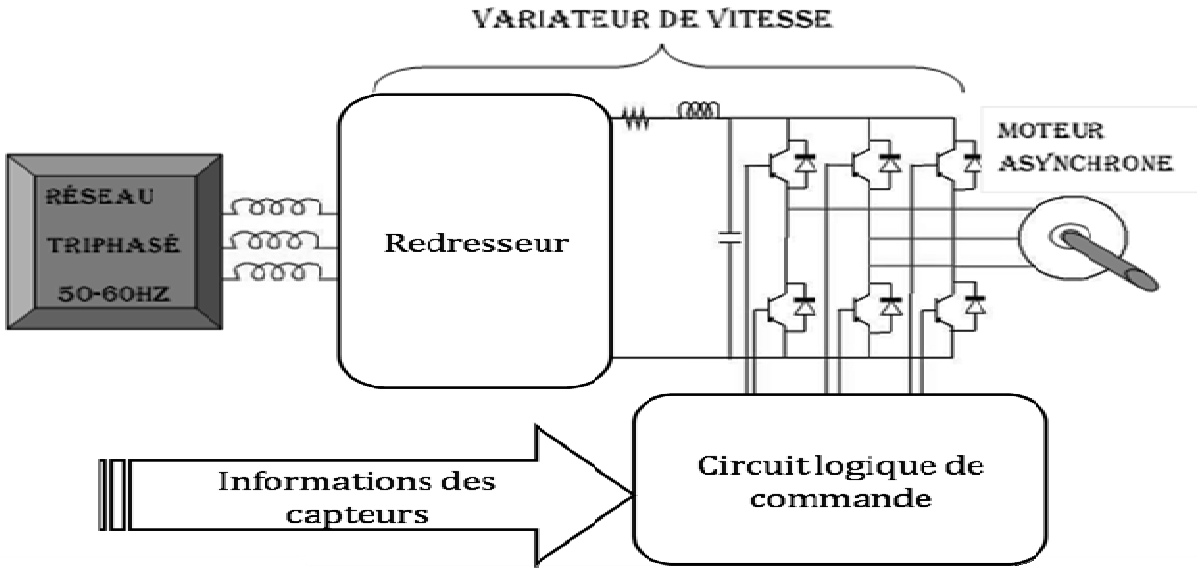
$$\begin{cases} \vec{U}_I = \frac{T_1}{T_d}\vec{U}_1 + \frac{T_2}{T_d}\vec{U}_2 + \frac{T_d - T_1 - T_2}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \\ \vec{U}_{II} = \frac{T_2}{T_d}\vec{U}_2 + \frac{T_3}{T_d}\vec{U}_3 + \frac{T_d - T_2 - T_3}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \\ \vec{U}_{III} = \frac{T_3}{T_d}\vec{U}_3 + \frac{T_4}{T_d}\vec{U}_4 + \frac{T_d - T_3 - T_4}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \\ \vec{U}_{IV} = \frac{T_4}{T_d}\vec{U}_4 + \frac{T_5}{T_d}\vec{U}_5 + \frac{T_d - T_4 - T_5}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \\ \vec{U}_V = \frac{T_5}{T_d}\vec{U}_5 + \frac{T_6}{T_d}\vec{U}_6 + \frac{T_d - T_5 - T_6}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \\ \vec{U}_{VI} = \frac{T_6}{T_d}\vec{U}_6 + \frac{T_1}{T_d}\vec{U}_1 + \frac{T_d - T_6 - T_1}{2T_d}\left(\vec{U}_0 + \vec{U}_7\right) \end{cases} \quad \textbf{EQ (II:}$$

19) **II.9)-MODELIATION DU SYSTEME D'ALIMENTATION**

II.9.1)-Alimentation

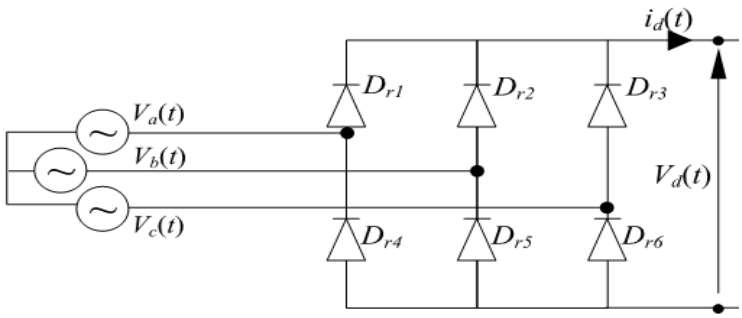
Cette partie est dédiée pour la présentation du système d'alimentation à convertisseurs de puissance. Une des applications des convertisseurs statiques de puissance est l'alimentation des moteurs électriques via les variateurs de vitesse qui permettent de commander les machines tournantes en contrôlant précisément leurs couples et leurs vitesses. Ce convertisseur statique représente un nouveau mode de conversion d'énergie où le rôle de ce dispositif est de permettre la modification de la forme de l'énergie électrique qu'il transmet.

Le modèle complet de ce convertisseur statique qu'on a choisi pour le moteur asynchrone triphasé est une simple mise en cascade d'un redresseur, filtre et onduleur. Afin de faire varier la vitesse et contrôler le couple des moteurs asynchrones il est nécessaire de faire varier simultanément l'amplitude et la fréquence de la tension (ou courant) d'alimentation. La variation de l'amplitude et de la fréquence est assurée par un onduleur commandé. L'onduleur de tension qui reçoit son énergie des batteries ou d'un réseau alternatif redressé via un redresseur .



Figure(II.07) : Schéma d'alimentation et de commande du moteur.

II.9.1.1)-Modélisation du redresseur



Figure(II.08) : Schéma de redresseur triphasé à diodes.

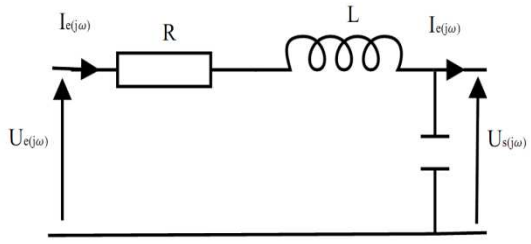
Ce type de redresseur comporte trois diodes à cathode commune assurant l'aller du courant *Id(t) :Dr1 ; Dr2 ; Dr3* et trois diodes à anode commune assurant le retour du courant *Id(t) :Dr4 ; Dr5 ; Dr6*. Si l'effet d'empiètement est négligé, la valeur instantanée de la tension redressée peut être exprimé par :

$$v_d(t) = \boldsymbol{max}\big(v_a(t), v_b(t), v_c(t)\big) - \boldsymbol{min}\,(v_a(t), v_b(t), v_c(t)) \qquad \textbf{EQ (II: 20)}$$

II.9.1.2)-Modélisation du filtre

Le filtrage *RLC* élimine les phénomènes d'ondulation de la tension en sortie du redresseur.

Cas générale *RLC* :



Figure(II.09) : Schéma du filtre ***RLC***.

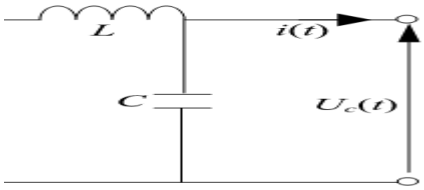
La structure du pont diviseur de tension permet de déduire l'expression de en $Us(j\omega)$ fonction de celle de Ue(jω) : $Us(j\omega) = \frac{\frac{1}{jc\omega}}{R+j\left(L\omega-\frac{1}{c\omega}\right)} Ue(j\omega)$

$$H(j\omega) = \frac{1}{1+\frac{1}{Q\omega_0}j\omega-\left(\frac{\omega}{\omega_0}\right)^2} \qquad \textbf{EQ (II: 21)}$$

$$H(\omega) = \frac{1}{\sqrt{\left(1-\left(\frac{\omega}{\omega_0}\right)^2\right)^2+\frac{1}{Q^2}\left(\frac{\omega}{\omega_0}\right)^2}} \quad et \quad \varphi(\omega) = -tan^{-1}\frac{\frac{1}{Q}\frac{\omega}{\omega_0}}{1-\left(\frac{\omega}{\omega_0}\right)^2} \qquad \textbf{EQ (II: 22)}$$

Cas particulier *LC* :

Afin de réduire le taux d'ondulation de la tension redressée et de réduire la chute de tension engendrer par la résistance ***R*** on utilise un filtre passe bas(***LC***) avec une résistance faible et négligeable.



Figure(I.10) : Schéma du filtre *LC*.

Le filtre passe bas *LC* est caractériser par mes équations différentielles suivantes :

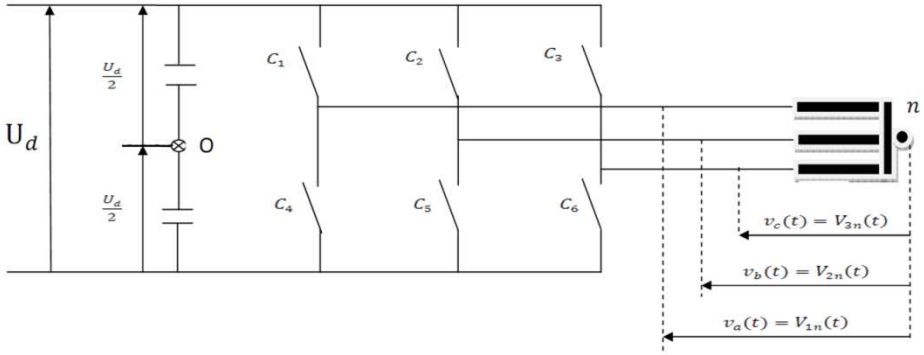
$$\begin{cases} \frac{di_d(t)}{dt} = \frac{1}{L}(v_d(t) - U_c(t)) \\ \frac{dU_c(t)}{dt} = \frac{1}{c}(i_d(t) - i(t)) \end{cases} \qquad \textbf{EQ (II: 23)}$$

Pour dimensionner le filtre, on doit tout simplement placer ca fréquence de coupure au dessous de la fréquence de la première harmonique de *Vd(t)*, cette condition nous permit de déterminer *L* et *C*.

II.9.1.3)-Modélisation d'onduleur de tension

L'onduleur de tension est largement décrit dans la littérature il se comporte comme un commutateur de tension en appliquant alternativement sur chaque borne du moteur les polarités positive et négative de la source et ça tâche est de transformer une source de tension continue constante en une tension alternative polyphasée de fréquence et d'amplitude variables. Il permet de faire la variation de la fréquence et de l'amplitude et même la forme de la tension appliquée au moteur électrique, ce qui permet la variation de sa vitesse de rotation. L'onduleur de tension est constitué de trois bras dont chacun possède deux interrupteurs (*cellules de commutation*) à base de semi-conducteurs (*IGBT à titre d'exemple*) montées en série et qui ne fonctionnent pas simultanément. Dans ce cas, chaque cellule est assimilée à un interrupteur contrôlable à l'ouverture et à la fermeture.

La vitesse de rotation du rotor dépend de la fréquence statorique et de la fréquence des courants rotoriques et l'onduleur est un convertisseur statique qui permet de faire varier la fréquence de la tension d'alimentation et donc de faire varier la vitesse de la machine. Cet onduleur est commandé par une *MLI* qui contrôle et impose la largeur des impulsions obtenues par hachage de la tension *Ud.* La *MLI* consiste à faire reproduire la valeur moyenne et d'approcher les trois tensions instantanément(*Les pertes dans les interrupteurs ne sont pas considérées*) par action d'ouverture ou fermeture des interrupteurs à chaque période de commutation en jouant sur la durée d'application des tensions positives et négatives.



Figure(II.11) : Schéma d'un onduleur de tension commandé.

$$Sc_i(t) = \begin{cases} 1 <=> C_i = On \\ 0 <=> C_i = Off \end{cases} \quad i \in \{1, \dots, 3\} \qquad \textbf{EQ (II: 24)}$$

On rappelle la définition d'un rapport cyclique $\alpha_i(k)$ pour l'intervalle de temps $[kT_d; (k+1)T_d]$:

$$\alpha_i(k) = \frac{[Temps\ de\ mise\ en\ On\ du\ bras\ i\ durant\ une\ période\ d'échantillonnage T_d]}{[Période\ d'échantillonnage T_d]}$$

$$\alpha_i(k) = \frac{[La\ valeur\ moyenne\ de\ la\ tention\ du\ bras\ i\ sur\ une\ période\ T_d]}{[La\ valeur\ moyenne\ de\ la\ tention\ \ redressée\ de\ charge]}$$

$$\alpha_i(k) = \langle Sc_i(t) \rangle_{Valeur\ moyenne} = \frac{1}{T_d} \int_{kT_d}^{(k+1)T_d} Sc_i(t)\, dt \qquad \textbf{EQ (II: 25)}$$

$$\alpha_i(k) = \frac{T_i(k)}{T_d} \ \forall (i \in \{1,2,3\}; k) \quad 0 \leq \alpha_i(k) \leq 1 \qquad \textbf{EQ (II: 26)}$$

$V_{io}(t)$ Peut prendre deux valeurs $\frac{U_d}{2}$ ou $-\frac{U_d}{2}$.

$$V_{io}(t) = \frac{U_d}{2}\left(2S_{c_i}(t) - 1\right) \quad i \in [1, \dots, 3] \qquad \textbf{EQ (II: 27)}$$

$$\begin{cases} V_{1o}(t) = \frac{U_d}{2}\left(2S_{c_1}(t) - 1\right) \\ V_{2o}(t) = \frac{U_d}{2}\left(2S_{c_2}(t) - 1\right) \\ V_{3o}(t) = \frac{U_d}{2}\left(2S_{c_3}(t) - 1\right) \end{cases} \qquad \textbf{EQ (II: 28)}$$

$$\begin{pmatrix} V_{1o}(t) \\ V_{2o}(t) \\ V_{3o}(t) \end{pmatrix} = U_d . \begin{pmatrix} S_{c_1}(t) \\ S_{c_2}(t) \\ S_{c_3}(t) \end{pmatrix} - U_d . \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \qquad \textbf{EQ (II: 29)}$$

$$\begin{cases} V_{1n}(t) = V_{1o}(t) - V_{no}(t) \\ V_{2n}(t) = V_{2o}(t) - V_{no}(t) \\ V_{3n}(t) = V_{3o}(t) - V_{no}(t) \end{cases} \qquad \textbf{EQ (II: 30)}$$

Le système triphasé est équilibré donc :

$V_{1n}(t) + V_{2n}(t) + V_{3n}(t) = 0$

$$\begin{pmatrix} V_{1n}(t) \\ V_{2n}(t) \\ V_{3n}(t) \end{pmatrix} = \frac{U_d}{3} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{pmatrix} S_{c_1}(t) \\ S_{c_2}(t) \\ S_{c_3}(t) \end{pmatrix} \qquad \textbf{EQ (II: 31)}$$

$$\begin{pmatrix} V_{1n}(t) \\ V_{2n}(t) \\ V_{3n}(t) \end{pmatrix} = \frac{1}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}\begin{pmatrix} V_{1o}(t) \\ V_{2o}(t) \\ V_{3o}(t) \end{pmatrix} \quad \textbf{EQ (II: 32)}$$

$$i_d = (S_{c_1}(t) \quad S_{c_2}(t) \quad S_{c_3}(t))\begin{pmatrix} i_a(t) \\ i_b(t) \\ i_c(t) \end{pmatrix} \quad \textbf{EQ (II: 33)}$$

En fonction de la tension continue *Ud* et des fonctions logiques générées par le système de contrôle, les tensions de sortie s'écrivent :

Avec:

$S_{c_i}(t)$; i=1, 2,3 : Les états binaires des cellules de commutation.

$v_{in}(t)$: Les tensions simples aux bornes du stator.

Où $S_{c_1}(t)$, $S_{c_2}(t)$ et $S_{c_2}(t)$ sont les fonctions de commutation à tout instant qui sont l'état de connexion des interrupteurs découpent la tension d'entrée en impulsions de largeur variable. Donc l'ensemble de ces fonctions sont instantanées et nous avons besoin de fonction discrétisées en numérique. Dans le cas continu nous aurons:

$$\begin{pmatrix} V_{1n}(t) \\ V_{2n}(t) \\ V_{3n}(t) \end{pmatrix} = \frac{U_d}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}\begin{pmatrix} S_{c_1}(t) \\ S_{c_2}(t) \\ S_{c_3}(t) \end{pmatrix} \quad \textbf{EQ (II: 34)}$$

On fait la moyenne sur une période d'échantillonnage $[kT_d; (k+1)T_d]$ des deux termes de cette équation et on aboutira à l'équation suivant en fonction des rapports cycliques:

$$\begin{pmatrix} V_{1n}(k) \\ V_{2n}(k) \\ V_{3n}(k) \end{pmatrix} = \frac{U_d}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}\begin{pmatrix} \alpha_1(k) \\ \alpha_2(k) \\ \alpha_3(k) \end{pmatrix} \quad \textbf{EQ (II : 35)}$$

$$\forall (i \in \{1,2,3\}; k) \quad 0 \leq \alpha_i(k) \leq 1$$

Où ***k*** est le nombre de périodes à l'instant d'échantillonnage. $V_{1n}(k)$, $V_{2n}(k)$ *et* $V_{3n}(k)$ sont les tensions moyennes sur une période a la *k*-ième période. $\alpha_1(k)$, $\alpha_2(k)$ *et* $\alpha_3(k)$ sont les rapports cycliques de chaque bras de l'onduleur sur la *k*-ième période d'échantillonnage.

Si on désire avoir des tensions de références aux bornes de chaque phase statorique du moteur et le point neutre commun on aura :

$$\begin{pmatrix} V_{1n}(k) \\ V_{2n}(k) \\ V_{3n}(k) \end{pmatrix} = \begin{pmatrix} V_{1nRef}(k) \\ V_{2nRef}(k) \\ V_{3nRef}(k) \end{pmatrix} \qquad \textbf{EQ (II: 36)}$$

On a les équations de tensions entre sorties d'onduleur et le point de référence **O** :

Cas continu :

$$\begin{pmatrix} V_{1o}(t) \\ V_{2o}(t) \\ V_{3o}(t) \end{pmatrix} = U_d \begin{pmatrix} S_{c_1}(t) \\ S_{c_2}(t) \\ S_{c_3}(t) \end{pmatrix} - \frac{U_d}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \textbf{EQ (II: 37)}$$

On valeurs moyennes sur une période d'échantillonnage:

$$\begin{pmatrix} V_{1o}(k) \\ V_{2o}(k) \\ V_{3o}(k) \end{pmatrix} = U_d \begin{pmatrix} \alpha_1(k) \\ \alpha_2(k) \\ \alpha_3(k) \end{pmatrix} - \frac{U_d}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \textbf{EQ (II: 38)}$$

D'après cette équation on tire les rapports cycliques en si on inject les tensions de référence par rapport au point ***O*** de l'équation **(I:29)** dans l'équation **(I:31)** et réarrangement des termes:

$$\begin{pmatrix} \alpha_1(k) \\ \alpha_2(k) \\ \alpha_3(k) \end{pmatrix} = \frac{1}{U_d} \begin{pmatrix} V_{1oRef}(k) \\ V_{2oRef}(k) \\ V_{3oRef}(k) \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \textbf{EQ (I: 39)}$$

Et par rapport au point n on aura :

$$\begin{bmatrix} \alpha_1(k) \\ \alpha_2(k) \\ \alpha_3(k) \end{bmatrix} = \frac{1}{U_d} \left(\begin{bmatrix} V_{1nRef}(k) \\ V_{2nRef}(k) \\ V_{3nRef}(k) \end{bmatrix} + V_{noRef}(k) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) - \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \textbf{EQ (II: 40)}$$

Avec : $\forall k \begin{cases} V_{noRef}(k) = \frac{1}{2} V_{ho}(k) \\ |V_{ho}(k)| = Min\left(|V_{1nRef}(k)|; |V_{2nRef}(k)|; |V_{3nRef}(k)|\right) \end{cases}$

En faisant varier la largeur des impulsions (*leur amplitude étant fixée par la tension d'alimentation continue*), on peut modifier l'amplitude et la fréquence du fondamental, donc de la tension d'alimentation du moteur.

II.10)- LES TECHNIQUES DE CONTROLE DES MACHINES

II.10.1)-Généralités

Les premières applications du moteur asynchrone est le réglage de la vitesse en régime permanent ; très vite on s'est intéressé aux performances durant les régimes transitoires ; à savoir: le démarrage, le freinage ainsi que ceux qui apparaissent lors de l'application brusque d'une charge. La disponibilité des convertisseurs statiques de puissance efficaces assurant la mise en forme de l'énergie électrique et la programmation des lois de commande dans des processeurs numériques très rapides tel que le Microcalculateur , *DSP ,ASIC , FPGA* ou autre circuit numérique, a permet d'utiliser le moteur asynchrone dans des applications à vitesse variable. L'objectif de contrôler et maitriser le régime dynamique transitoire est d'améliorer les performances et éviter les risques de ce régime qui se manifeste comme des changements brusques des tentions. De très nombreux travaux ont été menés dans ce domaine et plusieurs algorithmes de contrôles des machines sont développés et utilisés suivant pour développer des systèmes de commande de très hautes performances suivant l'application et les spécificités du cahier des charges comme le témoigne l'importante production scientifique associée.

D'une manière générale, Il existe différents algorithmes de contrôle qui dépend de l'objectif désirer, on peut citer ; le contrôle en courant, en position, en vitesse, en couple et en puissance. Il existe nos jours trois catégories de techniques de commande qui sont : les techniques empiriques, les techniques classiques et les techniques modernes.

- Les techniques empiriques sont basées en générale sur les algorithmes d'identification des procédés.
- Les techniques classiques sont la commande scalaire, la commande vectorielle directe et indirecte, la commande par mode glissement et bien d'autre.
- Les techniques modernes sont a base d'algorithmes génétiques, la logique floue et les réseaux neurones qui représentent des techniques de calcules numériques à base d'intelligence artificielle qui se caractérise par les calcules d'approximation.

Même s'il nous est impossible de traiter ici tous ces types de commande, il est nécessaire pour effectuer le contrôle des systèmes, de choisir une commande pour l'entraînement en question. Dans le but d'obtenir une commande similaire à la machine à courant continue, nous utilisons la commande vectorielle. La commande vectorielle élaborée sur le modèle de *Park* de la machine et qui ne dissocie pas le traitement des trois phases séparément mais elle

exploite les grandeurs en diphasés sur les axes ***(d,q)*** après avoir subir la transformé de *Park* et avant l'inversion de celle-ci. Nous présentons dans ce qui suit cette solution que nous avons retenue pour réaliser la commande où de nombreuses études ont évalué l'efficacité de cette commande.

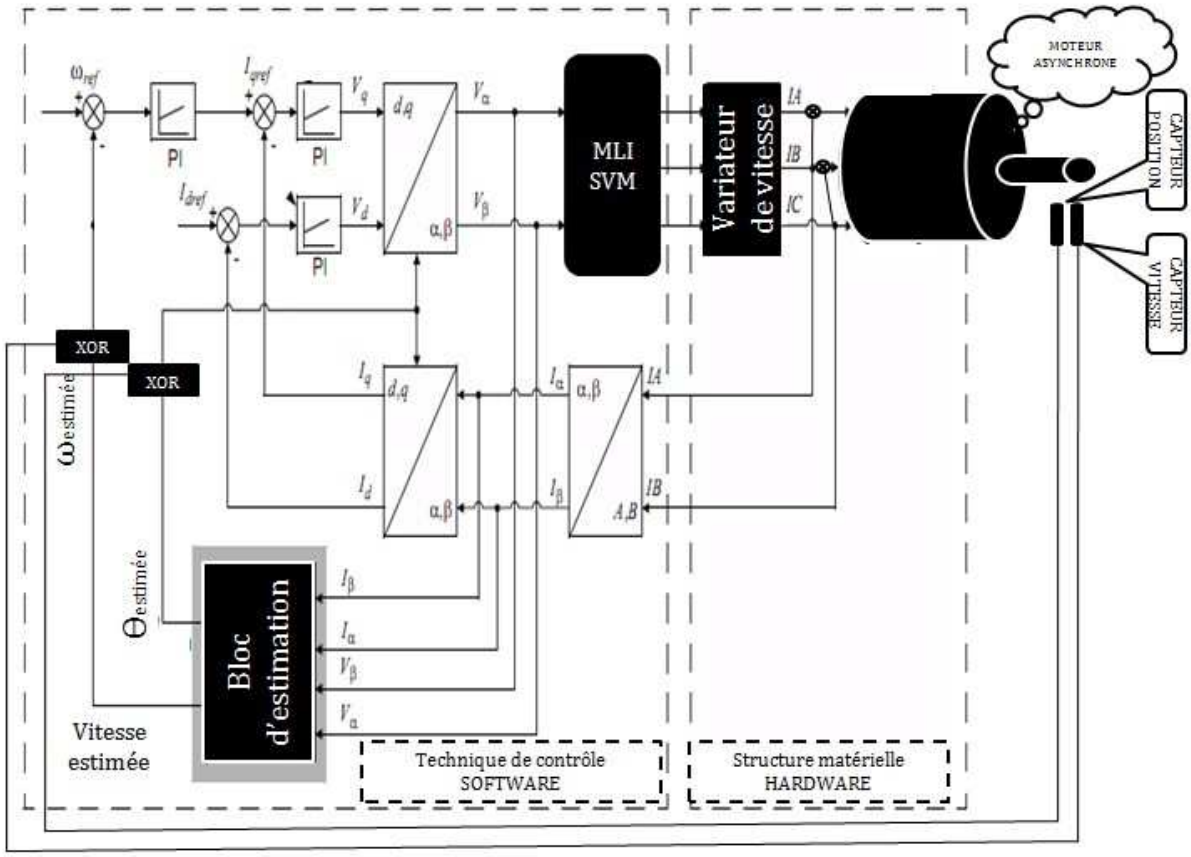
II.10.2)-La commande vectorielle

Depuis son apparition, la commande vectorielle des machines asynchrones n'a cessé d'intéresser les chercheurs dans le domaine des entraînements électriques à vitesse variable. Elle est devenue la référence universelle et industrielle pour contrôler le couple et la vitesse des moteurs à induction. De nombreux travaux de recherche ont été effectués dans ce domaine et depuis son apparition, la commande vectorielle des machines asynchrones est la colonne vertébrale de la commande des machines asynchrones à vitesse variable ou elle reste d'intéresser les chercheurs du domaine. La différence majeure entre la commande scalaire et la commande vectorielle c'est que cette dernière est plus complexe mais en contre partie, elle permet d'avoir de meilleures performances en régime transitoire. Les techniques de contrôle "vectoriel" de machine à courant alternatif sont destinées au contrôle instantané d'amplitude et position du vecteur flux que se soit le régime transitoire ou permanent. Il existe plusieurs types de commande vectorielle et on générale, la commande vectorielle a trois grandeurs à contrôler qui sont:

- Les flux rotoriques.
- Les flux statoriques.
- Les flux d'entrefer.

Le temps d'exécution du contrôle vectoriel est de quelques ***us***, car l'électronique de puissance des onduleurs commandés nous impose des contraintes de commutation d'ordre de 25 ***us.*** Le choix du contrôle vectoriel n'est pas un fait du hasard mais un chois judicieux non seulement par ces caractéristiques de contrôle qui sont largement éprouvées par la communauté scientifique mais aussi c'est sa complexité et sa modularité qui offrent un défi intéressant comme application ainsi qu'un bon terrain de conception et d'implantation sur *FPGA*.

On peut alors schématiser la commande vectorielle comme le montre la figure suivante sous forme de blocs. Chacun de ces blocs va produire des sorties en appliquant les entrées aux équations du bloc.



Figure(I.12) : Schématisation du contrôle vectoriel d'un moteur asynchrone triphasé.

$$I_\alpha = I_A \;\; et \;\; I_\beta = \frac{1}{\sqrt{3}}(2 * I_B + I_A) \quad Avec \quad \frac{1}{\sqrt{3}} = 0.577 = (0.100100111011)_{Binaire}$$

$$I_{sd} = I_\alpha . \cos(\theta) + I_\beta . \sin(\theta) \qquad et \qquad I_{sq} = -I_\alpha . \sin(\theta) + I_\beta . \cos(\theta)$$

EQ (II: 41)

Idem pour les tensions, ces dernières équations sont de même forme que celles des tensions en changent seulement les courants (I) par les tensions(V).

Ce schéma contient :

- **Des boucles de régulation :** Le rôle des boucles de régulation est de maintenir les grandeurs de sorties le lus proche possible des consignes.
- **Correcteurs PI :** Le correcteur **PI** classique est composé d'un terme proportionnel et d'un terme intégral. Ce correcteur est très utilisé dans les applications de contrôle de signaux continus ou discrets.
- **La stratégie de modulation** : La modulation vectorielle (*Space Vector Modulation, en anglais*) qui traite les signaux directement dans le plan diphasé de la transformée de *Park* où les angles de commutation sont déterminés en temps réel. Cette modulation

est utilisée dans les commandes modernes des machines asynchrones pour obtenir des formes d'ondes arbitraires non nécessairement sinusoïdales.

- **Le block d'estimations *:*** Ce block n'est pas toujours nécessaire et il peut ne pas être en cas de disponibilité de capteurs de vitesse et de position car afin d'asservir la vitesse de la charge dans la commande vectorielle, il faut mesurer celle-ci par l'intermédiaire d'un capteur mécanique. D'une manière générale, l'emploi de ce block est en cas où seules les variables statoriques sont mesurées.

II.11)-CONCLUSION

Ce chapitre a été consacré en premier lieu à modéliser l'association machine-convertisseur-commande. Il a mis l'accent sur la modélisation par équations différentielles qui régissent le comportement dynamique de ce système et la mise en évidence des propriétés des du variateur de vitesse. Cette modélisation a un intérêt primordial pour deux raisons qui sont ; la compréhension du comportement des actionneurs et en suite l'exploitation des modèles afin de servir pour synthèse des algorithmes et lois de commande. Nous avons dévoilé les différents types de commandes et nous avons choisi une pour être implanté sur un support physique. Dans le chapitre suivant, nous présentons la technique d'intégration de la commande sur un support programmable *FPGA* sous forme d'un circuit logique.

CHAPITRE III

Approche numérique de la commande vectorielle

OBJECTIF

Nous consacrons ce chapitre pour une étude plus détaillée des aspects arithmétiques de la commande vectorielle qui est abordée au précédent chapitre. Donc, la partie logique sera discutée en détail et nous tiendrons compte du compromis simplicité et efficacité du circuit à synthétiser indépendamment du fournisseur d'*FPGA*.

III.1)-INTRODUCTION

Aujourd'hui, concevoir un nouveau produit revient à imaginer une description matérielle possible de ce dernier. Alors, la tache confiée aux concepteurs de circuits numériques est de proposer des circuits logiques satisfont les cahiers de charges. Suite à ce que nous venons de dire, la conception d'un système et sa validation sur un circuit programmable *FPGA* exige généralement un important travail de conception et une grande expertise pour fixer l'architecture adéquate. Aujourd'hui, une méthodologie de conception rigoureuse est incontournable et la réussite dépend du savoir-faire du concepteur. Il est d'abord intéressant de noter que le processus de développement algorithmique est dissocié de l'implantation sur le circuit programmable *FPGA*. Pour faire introduire un algorithme de contrôle d'une machine sur un support physique *FPGA*, il est nécessaire de maitriser parfaitement le travaille d'adéquation entre l'algorithme analytique et l'architecture matérielle qui sera intégrer. Dans ce qui suit, nous présentons le développement d'une méthodologie systématique de conception numérique pour une commande vectorielle du moteur asynchrone. Notre approche sera une tentative de généraliser le processus de conception avec une évaluation de fonctions élémentaires par opérateurs simples réalisables en circuits logiques. Alors la démanche que nous suivrons durant ce chapitre est la conception des unités de calcul réutilisables dans plusieurs applications et sur plusieurs technologies.

III. 2)-L'ARTHMETIQUE DES CALCULATEURS

L'arithmétique est un processus de traitement accordé aux systèmes de représentation des nombres et les algorithmes de calcul associés. Un système arithmétique peut être conçu en matériel, en logiciel ou les deux au même temps (*Mixte*). L'arithmétique des ordinateurs est une discipline qui a comme sujet la manipulation des nombres avec leurs représentations ce qui correspond aux traitements algorithmiques de ces données. En matériel, ces algorithmes s'implantent sous forme de circuits combinatoires ou séquentiels qui sont très efficaces.

Les trois points fondamentaux d'arithmétique sont :

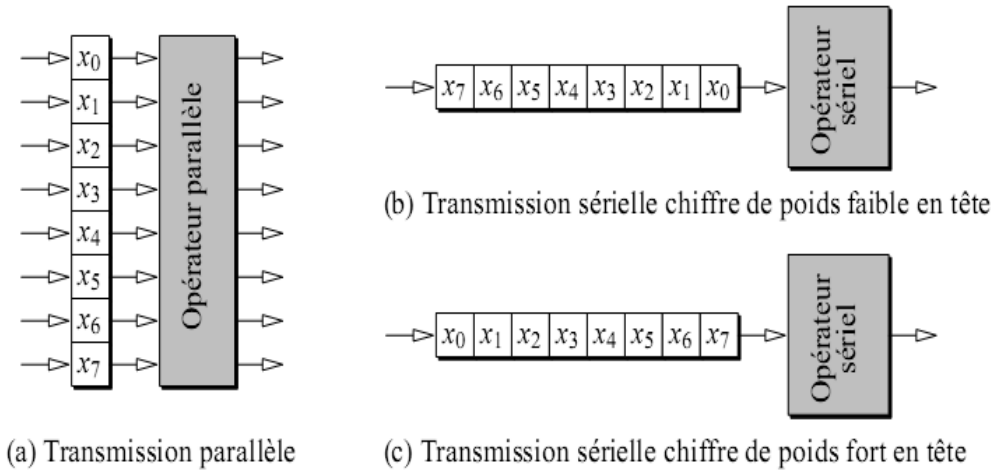
- **Les systèmes de représentation des nombres :** C'est la quantification et le codage de valeurs où grandeurs (Code machine).
- **Les algorithmes de calcul et d'évaluation arithmétiques :** C'est de décrire l'enchaînement des tâches a réalisées avec un langage approprier.
- **L'implantation matérielle ou logicielle :** C'est la manière d'introduire des fonctionnalités et des taches sur un support physique.

III.4)-TRANSMISSION DE DONNEES BINAIRES

D'une manière générale, il existe deux manières de transmettre les données entre deux composants. Souvent ces composants sont des registres qui sont nécessaires à la conservation de données de calcul. Alors les deux systèmes de transmissions sont :

- **La transmission parallèle (La plus utilisée) :** Les « *n* » bits de données à transmettre sont envoyés simultanément sur une ligne (*bus*) de transmission.
- **La transmission sérielle (La moins utilisée) :** Les bits d'un mot de données qui sont à transmettre seront envoyés les uns après les autres (*chiffre par chiffre*) à travers les modules et sur un seul fil de liaison.

Une complémentarité est observée entre ces deux modes de transmission. Un gain au niveau de surface requise par un opérateur sériel mais en contre partie il a besoin d'un nombre important de cycles d'horloge pour transmettre une donnée ce qui est inverse pour un opérateur parallèle.

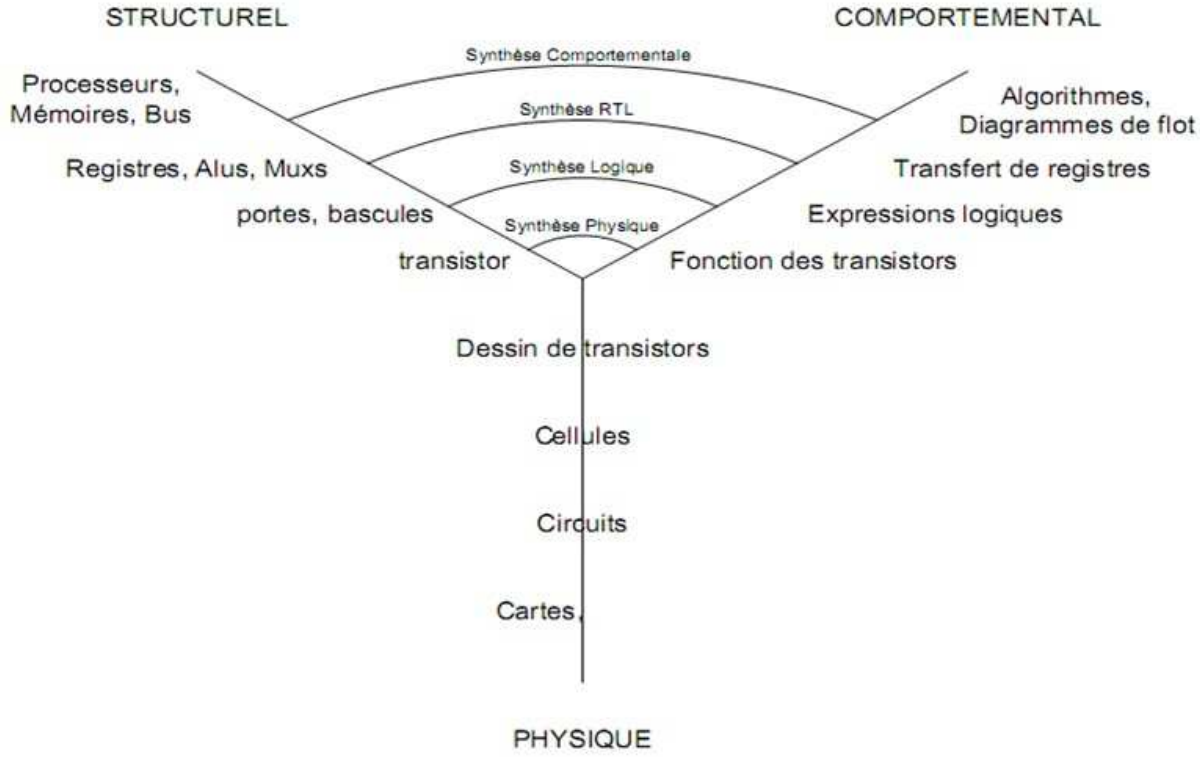


***Figure(III.01)**: Architecture de transmission parallèle / série.*

Dans les applications sur *FPGA*, le mode parallèle présente un gain de vitesse et le mode sériel présente un gain de surface. Dans notre travaille on s'intéresse au premier mode qui est le mode parallèle.

III.5)-CONCEPTION DU CIRCUIT LOGIQUE

On générale, les algorithmes sont introduits sur le matériel dans le formalisme le plus approprié (*Grafcet, RTL, Texte, Equations booléennes…etc.*). Le passage d'un niveau à un autre est un transite entre les différents niveaux d'abstraction. A cet effet, allons du niveau algorithme, il existe de différents niveaux avant d'arriver au niveau porte logique comme le montre le diagramme de *GAJSKI* qui suit.



***Figure(III.02)**: Le diagramme de **GAJSKI** reliant les niveaux d'abstraction.*

La synthèse d'un circuit numérique est un passage d'un niveau d'abstraction à un autre. Le niveau *RTL* (*Register Transfert Level*) est le plus adopter par les langages de description matérielle. Comme le montre le diagramme de *GAJSKI*, ce niveau est un niveau intermédiaire entre le niveau le plus élevé (*Niveau algorithmique*) et le niveau le plus bas (*Niveau portes logiques*).La conception des circuits digitaux peut se faire dans plusieurs niveaux et actuellement il existe des outils informatiques qui permettent de passer d'un niveau à un autre pour mieux manipuler la conception.

La conception d'un système numérique passe par quatre étapes:

- **Réflexion.**
- **Réalisation.**
- **Vérification.**
- **Intégration.**

On générale pour faire évaluer des fonctions au stade matériel, il existe de différentes manières pour les estimées qui sont récapitulées comme suit: Méthodes à base de tables, Méthodes à base d'algorithmes à récurrence, Méthodes à base d'approximations polynomiales ou rationnelles, ou encore des combinaisons de ces méthodes.

- **Les méthodes à base de tables :** Il est souvent possible de remplacer un circuit logique par une mémoire.
- **Les algorithmes à récurrence :** Comme il est bien connu, les résultats sont obtenus par itérations successives jusqu'a satisfaire un critère donné et souvent ce critère représente la précision ou l'erreur relative. Ces algorithmes s'articulent sur quelques operateurs le plus souvent des additionneurs pour le calcul de chaque itération. L'avantage principal de ces algorithmes récurrents c'est d'avoir un gain d'espace avec de petits operateurs. Mais en contre partie, la récurrence génère et accumule des pertes de temps ce qui est inacceptable dans certaines applications. Beaucoup d'algorithmes de ce genre sont développés mais les plus connus en conception matérielle on trouve sont :
  - L'algorithme *CORDIC* pour les fonctions élémentaires telque les fonctions trigonométriques et l'exponentiel ou le logarithme.
  - L'algorithme *E-METHODE* pour les opérations matricielles.
  - L'algorithme *SRT* en générale pour évaluer les fonctions algébriques de base comme la division, racine carrée et bien d'autres.
- **Les approximations polynomiales et rationnelles :** Les approximations polynomiales et rationnelles sont un moyen efficace d'approcher et d'évaluer les fonctions mathématiques. Le principal avantage de ces approximations est la couverture de la majorité des fonctions mathématiques et la précision manipulable en fonction du besoin. Mais en contre partie, les opérateurs de base de cette méthode sont la multiplication et la division qui consomme beaucoup d'espace sur le plan matériel.

Dans le domaine de la conception des systèmes numériques, le besoin nous oriente à faire la combinaison de ces différentes méthodes d'évaluation. C'est nécessaire pour évaluer des algorithmes qui ne cessent d'augmenter en complexité.

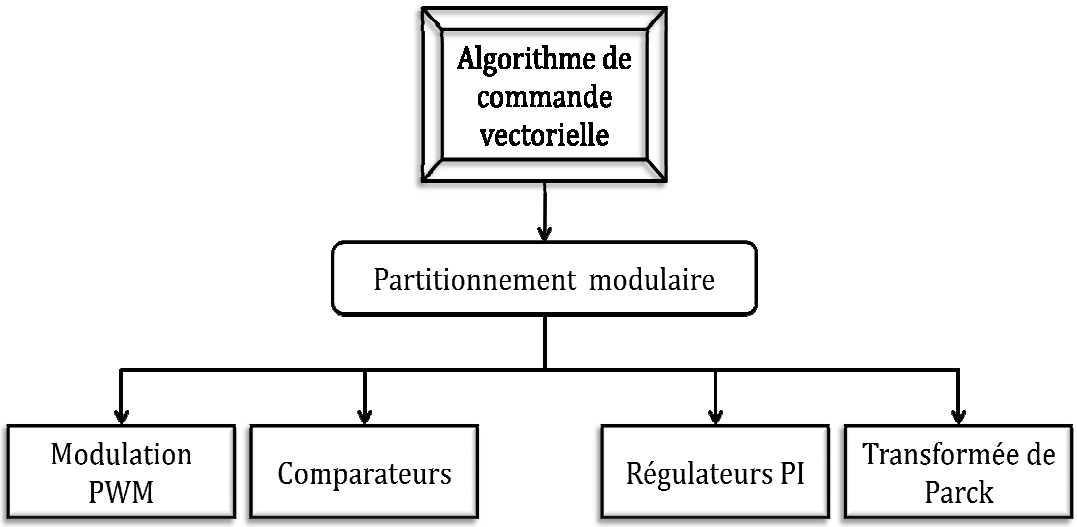
III.6)-APPROCHE MODULAIRE

Nous assistons à l'émergence de plusieurs tendances techniques et hiérarchiques pour manipuler la complexité des circuits. Ces tendances sont susceptibles d'influencer sur l'architecture matérielle des circuits dans le but d'amélioration les performances de ces derniers lors d'une implantation matérielle de fonctions numériques.

Parmi ces approches on trouve :

- L'approche architecturale automatique.
- L'approche architecturale dédiée.
- L'approche architecturale modulaire.

Pour mieux gérer la complexité algorithmique et de partager les tâches algorithmiques ; il est préférable de faire adopter l'approche modulaire pour l'intégration de commandes des systèmes électriques sur *FPGA* où principe de la modularité ressemble la philosophie de diviser pour reigner. La figure suivante montre comment le contrôle vectoriel est scindé sous forme modulaire.



***Figure(III.03)**: Architecture modulaire des éléments de base de la commande.*

Danc, le contrôle vectoriel est amorcé en se basant sur les équations théoriques de ce type de contrôle.

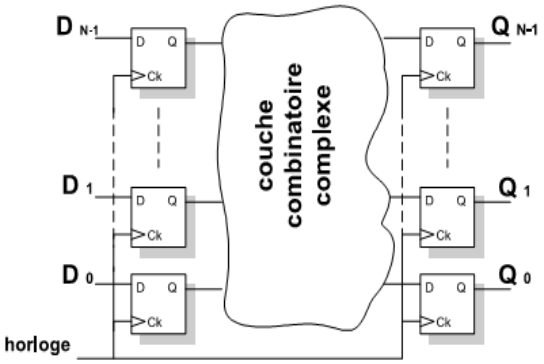
III.7)-ELEMENTS DE BASES POUR LA CONCEPTION

L'objectif finale de ce chapitre est d'établir une décomposition de la commande sous forme d'operateur matérielle simples à implanter sur un *FPGA*. Du moment qu'un ordinateur est incapable de concevoir un circuit car une machine est dépourvue d'intelligence et de réflexions, la conception est une tâche de création confier aux concepteurs dotés de certaines connaissances du domaine. Nous considérons que le problème crucial est de proposer une architecture simple et fiable et c'est dans ce sens que nous proposons alors une solution simple à appliquer par la suite. La nature de notre application qui est la commande vectorielle nous offre un très bon degrés de parallélisme qui va nous aider par la suite, ce qui n'est pas forcément évident pour d'autres applications. La majorité des algorithmes de commande des machines sont développés pour être exécuté par des processeurs séquentiels et non des solutions câblées. Les modèles développés jusqu'ici sont analytiques basés sur les équations mathématiques alors nous essayons d'exploiter au maximum cette approche analytique à un autre niveau d'abstraction pour déterminer les composants logiques permettant de répondre au besoin de notre système mécatronique. La structure de la commande vectorielle peut être vu comme un ensemble de blocs logiques qui exécutent des calculs et communiquent entre eux à chaque front d'horloge. Un problème potentiel de conception des systèmes sur puce est la taille de la circuiterie où les pionniers des systèmes digitaux essayent de le déceler et de l'optimiser.

Nous tenons à signaler l'ensemble de points que nous avons pris en considération :

- Le signal d'horloge, qui cadence tout le fonctionnement, doit être distribué, en phase, en tout point du circuit. Donc, le signal d'horloge issu du quartz est traité par un diviseur de fréquence, afin d'obtenir la fréquence de commutation désirée. Ce signale doit prendre en compte les problèmes de discrétisation liés à la réalisation numérique du système de contrôle.
- Chaque contacteur de l'onduleur est commandé directement par une bascule « *D* » (Ou registre et Mémoire) afin mémoriser son état au cours d'une période.
- Ajustage des largeurs de données.

Des registres et bascules d'entrée et de sortie sont insérés avant et après le circuit de traitement comme le montre la figure suivante.



***Figure(III.04)**: Architecture générique des éléments logiques de base adoptée.*

REMARQUE : Le choix de la période d'échantillonnage *Te* est essentiel et elle sera fixée en tenons compte du temps maximale de traitement du circuit.

III.8)-DECOMPOSITION MATERIELLE DE LA COMMANDE

Notre approche est de proposer une architecture intuitive très proche du matériel. Il s'agit ici d'étudier l'arsenal des modules algorithmiques existant afin d'identifier les composants numériques. Dans le contexte du développement numérique actuel, où les critères prédominants sont la rapidité de développement et la maîtrise des coûts, l'approche modulaire est la méthode la plus répondue. Comme nous l'avons déjà signalé, elle consiste à dégrossir un problème en le décomposant en une somme bien déterminer de sous problèmes. Cette méthode de conception permet de réduire le facteur d'erreur humaine. Pour la suite on considère que le circuit complexe d'algorithme de la commande vectorielle va être subdivisé en blocs. Chacun de ces blocs contiens des operateurs et fonctions logiques nécessaires pour réaliser une tâche bien précise qui est une partie de la commande vectorielle. D'après le schéma de la commande vectorielle vue dans le précédent chapitre on constate qu'il faut concevoir un ensemble de blocs logiques qui sont : Cinq comparateurs, quatre régulateurs *PI*, un bloc pour la transformée de *PARK* et un bloc pour l'étage *MLI*. Sans oublier les six bascules *D* pour attaquer chaque interrupteur de l'onduleur. On aura besoin d'un certain nombre de signaux pour le synchronisme en générale comme :

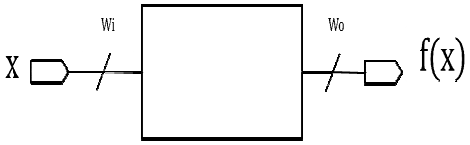
CLK : Afin de synchroniser le système suivant les fronts d'horloges.

RESET : Afin de remettre les registres à zéro (*Initialisation*).

Souvent, comme dans le cas de la commande vectorielle, le défi est alors de bien synchroniser les blocs entre eux. C'est ici que des signaux « *CLK* » et « *RESET* » peuvent être implémentes afin de s'assurer que le traitement des données se fait dans le bon ordre. Il ne s'agit plus ici d'un problème d'ordre mathématique, mais plutôt d'ordre logistique.

III.8.1)- La structure matérielle des operateurs

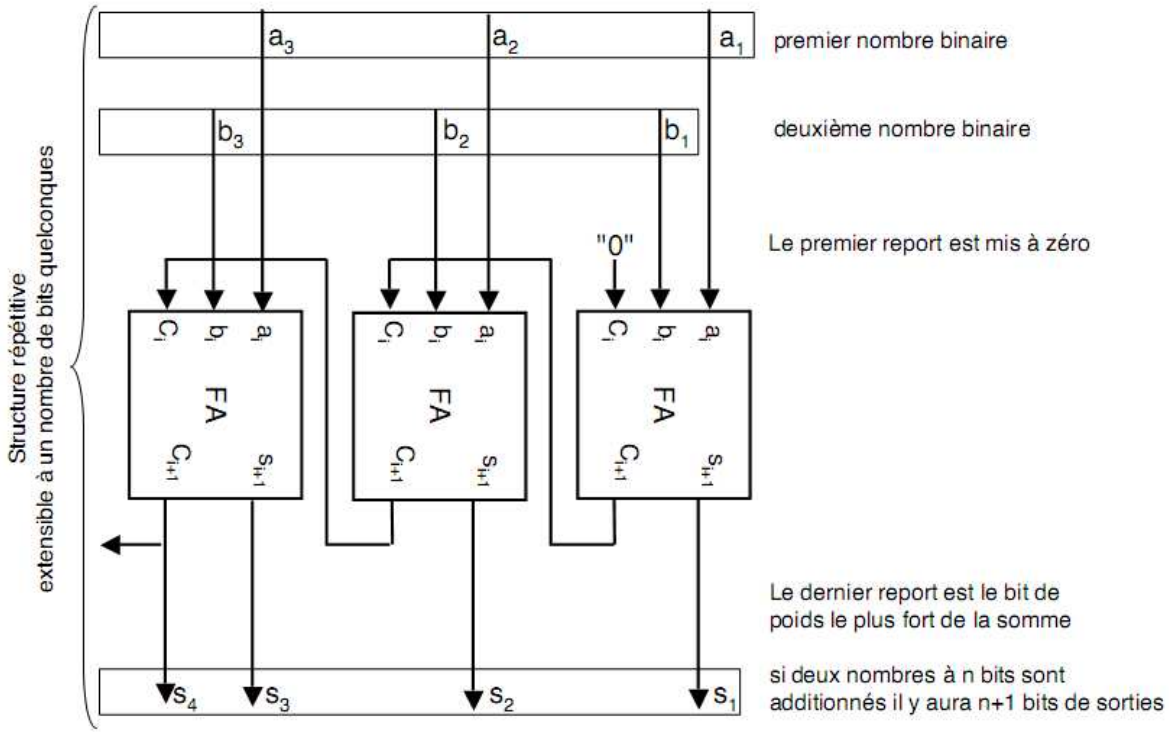
Les opérateurs logiques de base sont très utilisés dans les circuits complexes comme c'est le cas de notre système. Alors le rappel sur les additionneurs, soustracteurs et multiplieurs nous sera utile lors de la conception du système complet. Nous les assemblant comme une bibliothèque de composants ou d'operateurs arithmétiques (*Fonctions élémentaires*) a développée.



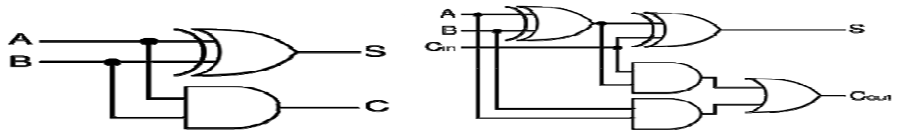
***Figure(III.05)**: Operateur d'évaluation d'une fonction élémentaire.*

III.8.1.1)-Les additionneurs

Le principe d'un additionneur binaire est illustré dans les deux figures qui suivent :



***Figure(III.06)**: Schéma et principe d'un additionneur logique.*

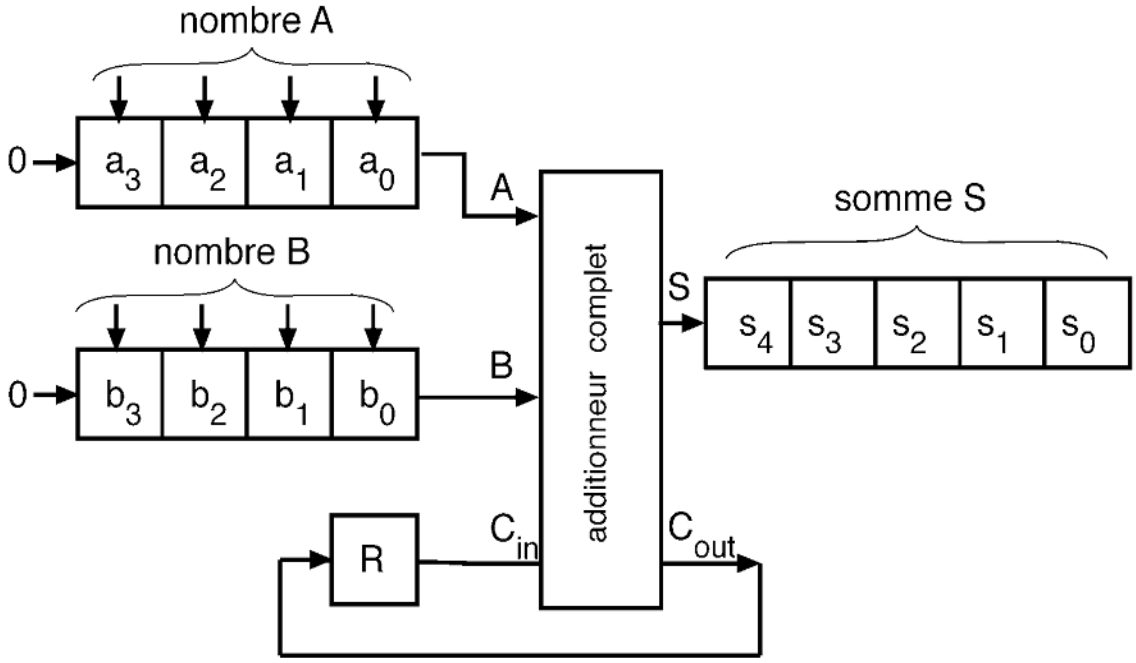


***Figure(III.07)**:Cellules logiques d'un additionneur.*

$$S = (A \oplus B) \oplus C_i$$

$$C_o = (A \cdot B) + (C_i \cdot (A \oplus B)) = (A \cdot B) + (B \cdot C_i) + (C_i \cdot A)$$

EQ (III: 01)



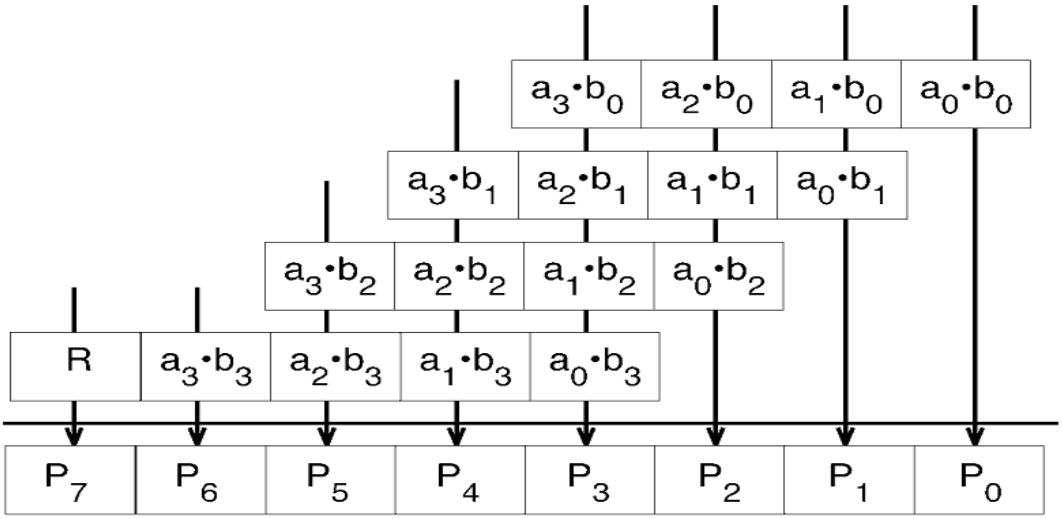
***Figure(III.08)**: Schéma de réalisation séquentielle d'opération Additionneur.*

III.8.1.2)-Les soustracteurs

Pour concevoir un soustracteur de deux nombres binaires il est plus judicieux d'exploiter la structure de l'additionneur. On peut obtenir un soustracteur à base d'un additionneur par une complémentation du nombre soustrait au compliment à ***2***.

III.8.1.3)-Les multiplieurs

La multiplication est l'une des opérations les plus coûteuses en consommation de ressources par rapport aux autres operateurs arithmétiques ainsi parmi les plus lentes à être implanter sur *FPGA*.



***Figure(III.09)**: Schéma de principe de l'opération de multiplication.*

III.8.2)- La structure matérielle de la transformé de PARK

Nous allons réaliser un système numérique permettant d'émuler le comportement le la matrice de *PARK*. La fonction sinusoïdale est très importante car elle intervient souvent dans les théories de la commande comme c'est le cas de notre application où la matrice de *PARK* est totalement constituée d'éléments sinusoïdaux. D'une manière générale pour générer une fonction trigonométrique on utilise différentes méthodes qui sont à base de mémoire *ROM* ou bien des algorithmes spécifiques telque *CORDIC*. Alors pour faire évaluer la fonction sinus il existe deux solutions qui sont :

- **Solution statique par mémorisation *(PROM)* :**

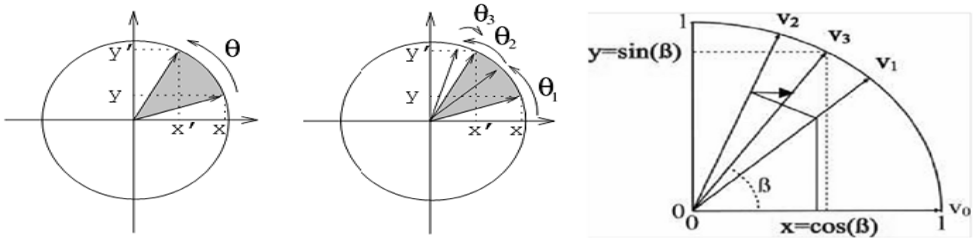
***Figure(III.10)**: Génération de sinus a base de mémoire **PROM**.*

En effet, plus il y a d'échantillons calculés, plus l'intervalle de phase entre deux échantillons successifs est faible. Malheureusement cette méthode présente beaucoup de restrictions sur la fréquence d'horloge maximale que peut atteindre le circuit ainsi un grand gaspillage de ressources (*La surface occupée*). Une technique intuitive est exploitée pour réduire l'espace mémoire nécessaire qui consiste à faire prendre des échantillons de la fonction sur un seul quadrant du cercle trigonométrique et même moins et de générer par la suite le reste des valeurs du cercle par jeux de signe seulement.

- **Solution dynamique par récurrence algorithmique*(CORDIC) :***

Il existe cependant une alternative intéressante pour remplacer la méthode tabulaire et l'utilisation des mémoires pour générer des fonctions mathématiques qui permit éviter cet inconvénient de ressources limitées. Cette alternative réside dans l'exploitation des algorithmes spécifiques. C'est en effet ce qui justifie le recours à l'algorithme *CORDIC* qui est un calculateur numérique à rotations de coordonnées. Le *CORDIC* (***CO****ordinate* ***R****otation* ***D****igital* ***I****ntegrate* ***C****ircuit*) est introduit pour la première fois en *1959* par l'américain *JACK E.VOLDER* en suite il est suivi d'un développement théorique approfondi par *Walter* en *1971*. C'est la clé de réussite de certaines calculatrices scientifiques commerciales.

Il est particulièrement bien adapté aux composants programmables car cet algorithme est caractériser par la simplicité du circuit numérique qui lui correspond conçu avec quelques éléments de base telque les additionneurs, les soustracteurs et les registres a décalages. Il permet d'économiser la surface et de calculer d'une façon itérative toute une gamme de fonctions comme les fonctions trigonométriques, hyperboliques, logarithmiques et exponentielles.



***Figure(III.11)**: Principe de décomposition des miro-rotations du **CORDIC**.*

D'après la figure, le vecteur $\boldsymbol{v}$ subit une rotation d'angle $\boldsymbol{\theta}$. cette rotation peut être transcrite sous forme matricielle avec une conversion de coordonnées « *Polaires-Cartésiennes* » comme est comme suit :

$$\overrightarrow{v'} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \vec{v} = \cos\theta \,.\, \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix} \vec{v} \qquad \textbf{EQ (III: 02)}$$

$$\vec{v} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \; et \; \overrightarrow{v'} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$$

Pour simplifier les fonctions trigonométriques complexes et faire une approche binaire pour l'angle de rotation, on ne doit considérer qu'un sous ensemble particulier d'angles qui sont une multiplication par une puissance de deux. Avec cette simplification l'opération de rotation correspond à un simple décalage binaire à droite ou à gauche.

$$\alpha = \tan^{-1}(2^i) \ , i \in Z$$

L'équation précédente peut réécrite par:

$$\overrightarrow{v'} = \cos(\tan^{-1}(2^i)) . \begin{pmatrix} 1 & -2^i \\ 2^i & 1 \end{pmatrix} \vec{v} \qquad \textbf{EQ (III: 03)}$$

Pour ne pas exclure les angles qui ne sont pas puissance de deux, *VOLDER* a démontré que quelque soit de θ prise dans l'intervalle $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ il existe une suite de valeurs $d_i \in \{-1,1\}$ qui permet de décomposer l'angle θ en une série de micro-rotations successives telque :

$$\theta = \sum_{i=0}^{\infty} d_i . tan^{-1}(2^{-i}) \qquad \textbf{EQ (III: 04)}$$

Ainsi, effectuer une rotation d'angle θ peut s'écrire sous la forme matricielle suivante :

$$\overrightarrow{v'} = \prod_{i=0}^{\infty} cos(d_i . tan^{-1}(2^{-i})) . \prod_{i=0}^{\infty} \begin{pmatrix} 1 & -d_i . 2^i \\ d_i . 2^i & 1 \end{pmatrix} \vec{v} \qquad \textbf{EQ (III: 05)}$$

Pour simplifier cette expression, on peut poser :

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} \cos(\tan^{-1}(2^{-i})) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1+2^{-2i}}} \qquad \textbf{EQ (III: 06)}$$

$$K = \lim_{n \to \infty} K(n) \approx 0.6073$$

Finalement les itérations de l'algorithme de *CORDIC* sont décrites par les quatre relations suivantes :

$$\begin{cases} x_{k+1} = x_k - d_k . y_k . 2^{-i} \\ y_{k+1} = y_k + d_k . x_k . 2^{-i} \\ z_{k+1} = z_k - d_k . y_k . tan^{-1}(2^{-i}) \\ d_{i+1} = \pm 1 \end{cases} \qquad \textbf{EQ (III: 07)}$$

En faite, il existe deux modes pour interpréter puis exploiter cet algorithme qui sont :

- Le mode vectoriel.

$$\begin{aligned} x_{k+1} &= x_k - d_k.y_k.2^{-i} \\ y_{k+1} &= y_k + d_k.x_k.2^{-i} \\ z_{k+1} &= z_k - d_k.y_k.tan^{-1}(2^{-i}) \\ d_k &= \begin{cases} -1 & si\ y_k < -1 \\ +1 & si\ non \end{cases} \end{aligned} \qquad \textbf{EQ (III: 08)}$$

Alors comme résultat d'itérations de (*k=0*) jusqu' (*k=n*) de ce mode vectorielle, nous aurons:

$$\begin{cases} x_n = A_n\sqrt{{x_0}^2 + {y_0}^2} \\ y_n = 0 \\ z_n = z_0 + tan^{-1}\left(\frac{y_0}{x_0}\right) \\ A_n = \prod_{i=0}^{i=n}\sqrt{1 + 2^{-2i}} \end{cases} \qquad \textbf{EQ (III: 09)}$$

Comme il est clair dans ces dernières équations, ce mode vectoriel a pour objectif de déterminer par approche, l'angle de rotation et l'amplitude avec itérations sur les coordonnées.

- Le mode rotationnel.

$$\begin{aligned} x_{k+1} &= x_k - d_k.y_k.2^{-i} \\ y_{k+1} &= y_k + d_k.x_k.2^{-i} \\ z_{k+1} &= z_k - d_k.y_k.tan^{-1}(2^{-i}) \\ d_k &= \begin{cases} -1 & si\ z_k < -1 \\ +1 & si\ non \end{cases} \end{aligned} \qquad \textbf{EQ (III: 10)}$$

Alors comme résultat d'itérations de (k=0) jusqu' (k=n) de ce mode rotationnel, nous aurons:

$$\begin{cases} x_n = A_n(x_0 \cos z_0 - y_0 \sin z_0) \\ y_n = A_n(y_0 \cos z_0 - x_0 \sin z_0) \\ z_n = 0 \\ A_n = \prod_{i=0}^{i=n}\sqrt{1 + 2^{-2i}} \end{cases} \qquad \textbf{EQ (III: 11)}$$

Comme il est clair dans ces dernières équations, ce mode rotationnel a pour objectif de déterminer par approche, les coordonnées avec itérations sur l'angle de rotation.

Les deux précédents modes sont équivalent mais le chois d'un parmi eux dépend de la fonction trigonométrique ou autre à approximée. Pour notre cas, nous choisissons le deuxième mode qui est le mode rotationnel afin d'approximer les deux fonctions trigonométrique Sinus et Cosinus.

Le principe itératif du *CORDIC* par approche rotationnelle angulaire qui consiste à pivoter dans le sens approprié le vecteur de rotation par un angle de plus en plus petit jusqu'à ce que l'angle θ entre deux vecteurs successifs, soient approximativement égales à 0. Comme le signale le principe itératif du *CORDIC*, la troisième équation (z_k) de l'algorithme de *CORDIC* permet de garder une trace de l'angle de rotation accumulé durant les micro-rotations successives.

$$z_n = z_0 - \sum_{i=0}^{n-1} d_i . tan^{-1}(2^{-i}) \qquad \textbf{EQ (III: 12)}$$

Sachont z_n est la déférence entre l'angle de départ et la somme totale des angles de rotation accumulés. Cette suite récurrente converge vers la matrice solution du problème initial.

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

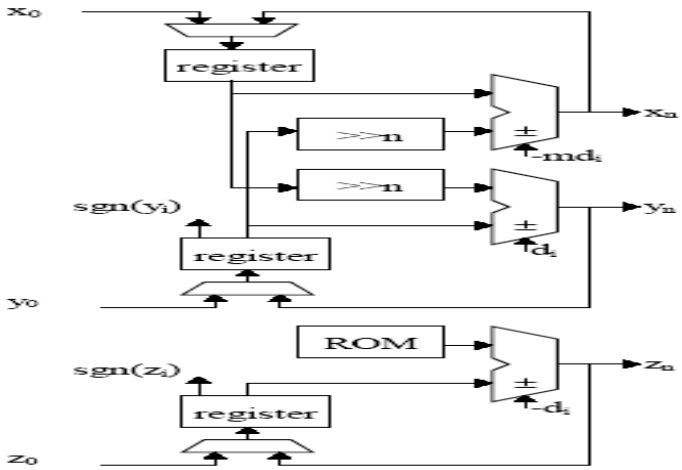
$$x_0, y_0 \; et \; z_0 = \theta$$

Donc pour le calcule du sinus et cosinus nous choisirons le vecteur initial sur l'axe des ordonnées comme le montre la figure précédent et nous aurons par conséquent après un nombre jugé suffisant d'itérations :

$$\begin{cases} x_n = A_n(x_0 \cos\theta - y_0 \sin\theta) = A_n \cos\theta \\ y_n = A_n(y_0 \cos\theta - x_0 \sin\theta) = -A_n \sin\theta \\ z_n = 0 \end{cases} \qquad \textbf{EQ (III: 13)}$$

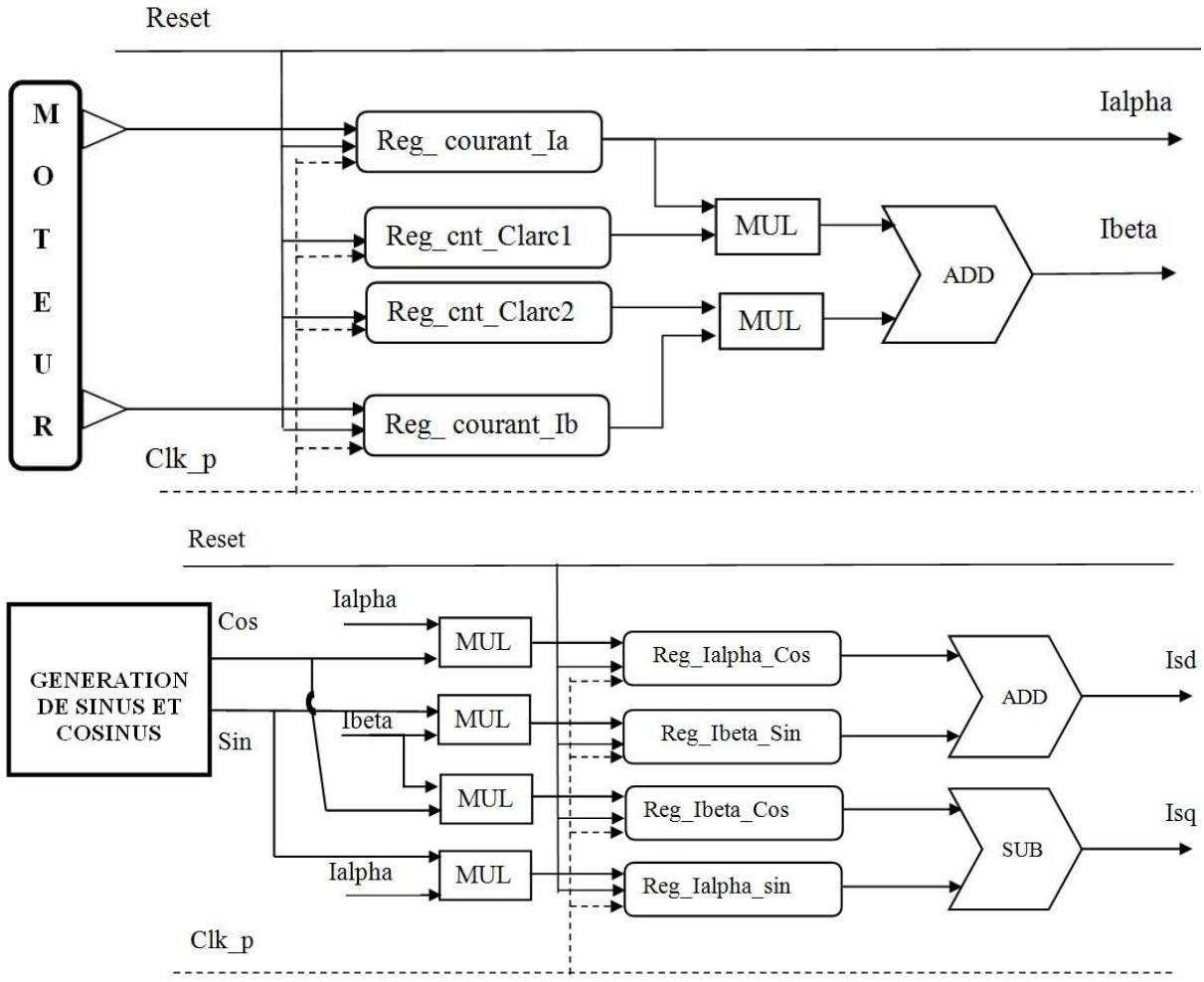
$$avec \;\; x_0 = 1; y_0 = 0 \; ; \; z_0 = \theta \;\; et \; A_n = \frac{1}{K_n} \qquad \lim_{n\to\infty} K(n) \approx 0.6073$$

La réalisation du circuit logique qui reflète le comportement du *CORDIC* peut se faire de différentes façons comme la réalisation parallèle et la réalisation sérielle. Nous retenons l'architecture de la figure suivante :



***Figure(III.12)**: Bloc **CORDIC** au niveau RTL.*

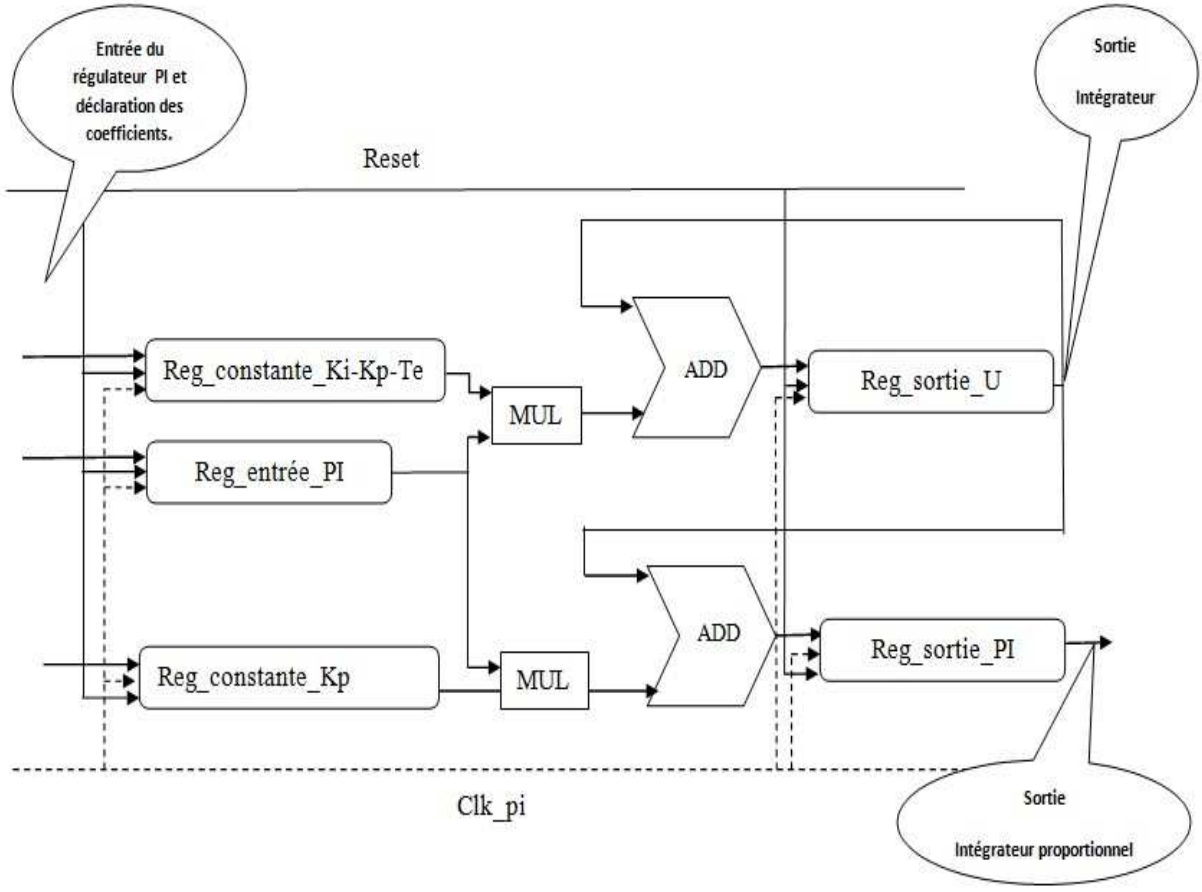
L'approche au niveau *RTL* de la transformation de *PARK* est comme le montre la structure suivante :



***Figure(III.13)**: Schéma du bloc de la transformé de **PARK** au niveau RTL.*

III.8.3)- LA SRUCTURE MATERIELLE DU REGULATEUR PI

Nous allons réaliser un système numérique permettant de réaliser un correcteur *PI*. Comme il est connu que les correcteurs à base d'intégrateurs, apparaissent comme la structure la mieux adaptée dans la régulation face en générale à des erreurs paramétriques grâce a leur robustesse. Pour implémenter un correcteur *PI* dans un *FPGA* on doit transformer sa forme analytique continue en une forme exploitable et synthétisable avec des additions, multiplications et des décalages où les coefficients seront stockés dans des registres ou mémoires.

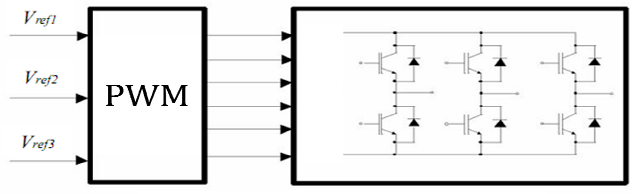


***Figure(III.14)**: Architecture générique des **PI**.*

L'implémentation du *PI* peut se faire à base de trois composants logique qui sont : additionneurs, multiplieurs et registres. Ces registres contiennent le coefficient du terme *e(k)* qui prend en compte le coefficient proportionnel *Kp*, intégrale *Ki* et la période d'échantillonnage *Te* comme le montre la figure précédente. Finalement un régulateur *PI* n'est qu'un accumulateur au stade logique qui représente l'intégrateur plus a cet accumulateur un terme de proportionnalité.

III.8.4)- LA SRUCTURE MATERIELLE DU BLOC PWM

Nous allons réaliser un système numérique permettant de générer ce qui représente le cœur de la structure de commande .Ce bloc génère des impulsions modulées par la méthode *PWM* vectorielle comme le montre la figure suivante.



***Figure(III.15)**: Position de la **PWM** dans la chaine de régulation.*

Deux étapes fondamentales qui sont :

- La détermination du secteur convenable parmi les six secteurs.
- Le calcule de durées (des impulsions) d'application de chaque vecteur des huit combinaisons possible ou seulement les six qu'on a présenté au chapitre deux.

❖ ***Détermination du secteur :***

Il est possible de déterminer le secteur par différentes manières. La plus simple à intégrer sur *FPGA* est basée sur la comparaison des deux tensions $\boldsymbol{V_\alpha}$ et $\boldsymbol{V_\beta}$ comme le montre le tableau suivant :

Secteur	Intervalle	$V_\beta > 0$	$V_\beta > \sqrt{3}.V_\alpha$	$V_\beta > -\sqrt{3}.V_\alpha$
I	$(0^0, 60^0)$	**1**	**0**	**1**
II	$(60^0, 120^0)$	**1**	**1**	**1**
III	$(120^0, 180^0)$	**1**	**1**	**0**
IV	$(180^0, 240^0)$	**0**	**1**	**0**
V	$(240^0, 300^0)$	**0**	**0**	**0**
VI	$(300^0, 360^0)$	**0**	**0**	**1**

***Tableau(III.01)**:Table de détermination des secteurs de la **PWM**.*

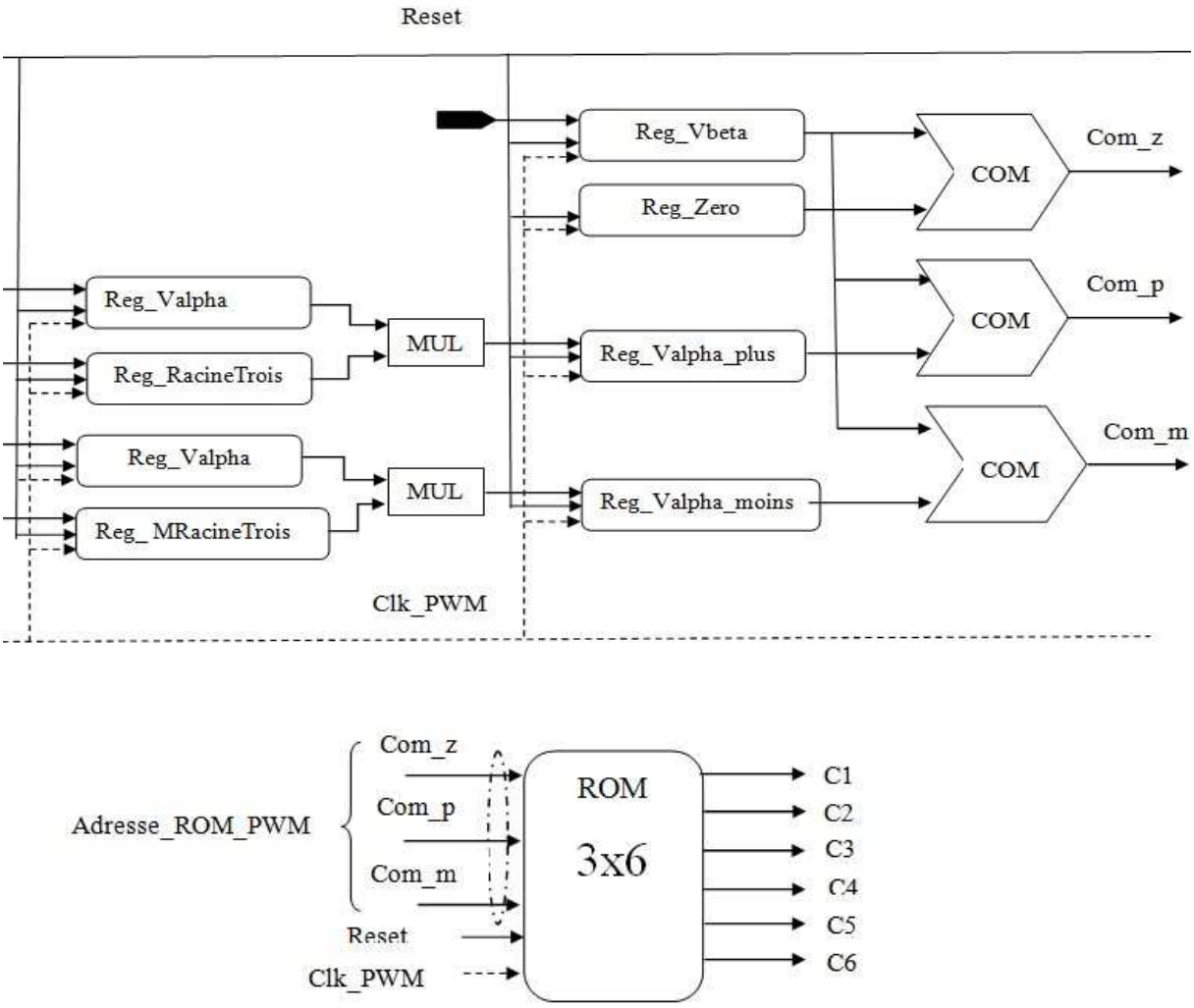
❖ ***Détermination des durées d'impulsions:***

Comme nous l'avons vu au précédent chapitre de modélisation, pour déterminer le durée des impulsions il faut faire des projections du vecteur générer par les deux composantes V_α et V_β sur les deux vecteur proches qui délimitent le secteur concerné. Après projection, le rapport cyclique de chaque vecteur tension à applique est égal à la valeur algébrique de cette projection. Une fois que les rapports cycliques sont déterminés alors les durées d'application de chaque vecteur sur une période d'échantillonnage seront déterminer car le rapport cyclique

représente la proportionnalité entre le temps d'application d'un vecteur qu'on cherche et la période d'échantillonnage (*Ti/Te*). Malheureusement pour implanter cette technique sur *FPGA* il faut consommer beaucoup d'espace. Pour résoudre le problème nous avons opté pour faire des simplifications de telle manière à appliquer un seul vecteur tension sur une période d'échantillonnage Te comme le présent le tableau suivant.

Secteur	Intervalle	Entrées	Sorties
I	$(0^0, 60^0)$	**101**	**001**
II	$(60^0, 120^0)$	**111**	**010**
III	$(120^0, 180^0)$	**110**	**011**
IV	$(180^0, 240^0)$	**010**	**100**
V	$(240^0, 300^0)$	**000**	**101**
VI	$(300^0, 360^0)$	**001**	**110**

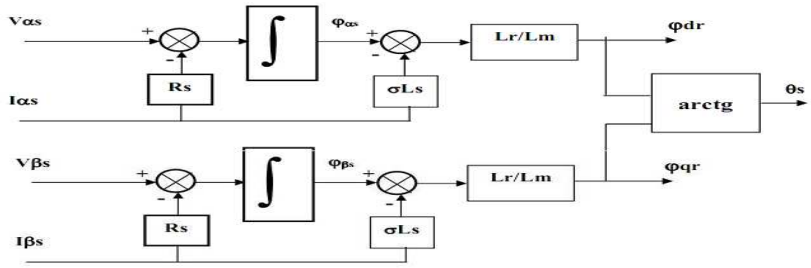
***Tableau(III.02)**: Table simplifié pour la **PWM**.*



***Figure(III.16)**: Schéma au niveau RTL du bloc **PWM**.*

III.8.5)- LA SRUCTURE MATERIELLE DU BLOC ESTIMATION

Nous parlons ici d'estimateur connu dans la théorie de commande où il est indispensable de remplacer les grandeurs non commandables ou non observables par des modèles algorithmiques. Le deuxième point d'utilisation de cet estimateur est pour réduire le nombre de capteurs. Pour notre application qui est le moteur asynchrone triphasé, la structure interne de cet estimateur est représentée dans la figure suivante :



***Figure(III.17)**: Schéma bloc d'estimation.*

On est sensé dans cette partie de construire un système numérique permettant de refléter le comportement de cet estimateur qu'on vient de présenter, où les grandeurs à estimer en temps réel. Comme le montre la figure précédente, le bloc d'estimation est assez complexe et demande beaucoup de ressources en circuits logiques où la provenance de l'erreur à l'intérieur du bloc est souvent difficile à identifier. L'implantation de cet estimateur sur un FPGA va épuiser les ressources de ce dernier. Sachont que ce bloc n'est pas nécessaire en cas de disponibilité du capteur de vitesse ou position, nous allons seulement décrire brièvement une méthodologie pour le transcrire en circuit logique. Pour chaque intégrateur correspond un accumulateur logique comme on a fait précédemment. L'ensemble des coefficients sera codé et stocké dans des registres et les comparateurs sont équivalent aux soustracteurs binaires. Finalement la fonction trigonométrique arc-tangente « *arctg* » sera réalisable à base d'algorithme *CORDIC* qu'on a présenté précédemment .Il suffit d'exprimer arc-tangente en fonction de sinus et cosinus ou bien de manipuler judicieusement l'algorithme *CORDIC*.

III.8.6)-CONCEPTION ET SYNTHESE DU SIGNAL D'HORLOGE

Il existe deux types d'horloges qui sont l'horloge de cadencement et l'horloge d'échantillonnage. Le nombre de pas nécessaire pour le traitement n'est pas forcément le même pour deux blocs, ce qui engendre l'insertion de blocs supplémentaires pour synchroniser les résultats .Pour résoudre le problème de « timing » et à base d'un conteur on peut générer une fréquence adaptée et inferieur à la fréquence d'oscillateur ou horloge

principale. C'est ce qui signifie que l'usager doit modifier le mécanisme de synchronisation entre les deux blocs.

III.9)-LE CIRCUIT NUMERIQUE DE LA COMMANDE VECTORIELLE

A ce stade, nous avons développé des architectures hautement parallèles afin d'aboutir à une implantation efficace. Cette architecture qu'on vient d'élaborer est un support garanti en cas de modifications futures éventuelles des exigences ou de la configuration du système car les circuits logiques programmables *FPGA* permettent de rajouter de nouveaux blocs de traitement si le concepteur le juge utile et dans les limites des capacités de ces circuits *FPGA*. On peut alors passer à la simulation et la réalisation matérielle par des outils *CAO* de cette architecture avec un chois convenable aux valeurs des paramètres de la machine asynchrone.

III.10)-CONCLUSION

Dans ce chapitre, nous avons présenté l'état d'art sur la conception numérique en générale et plus particulièrement la commande vectorielle. Cette conception que nous venons de dresser, nous a permis de mettre en lumière une méthodologie de conception d'un système de commande numérique dédié a la commande du moteur asynchrone. L'analyse de cette conception rend compte de la multitude des situations auxquelles le concepteur est confronté, dans son projet ou face à de multiples compromets a satisfaire. Ce chapitre apporte une solution simple et systématique pour la conception et la synthèse de la structure logique de la commande vectorielle en utilisant les éléments théoriques détaillés dans les chapitres précédents et nous a permit de conclure que l'entraînement électronique des machines à courant alternatif nécessite une électronique qui reste relativement complexe. Au cours de ce chapitre nous avons développé une architecture hautement parallèles et modulaire afin d'aboutir à une implantation simple et efficace sans erreurs. Ce chapitre est la pierre angulaire des travaux présentés dans ce mémoire car il présente l'approche interne du circuit logique de commande qui permet une étude des fonctions techniques assurées par les constituants matériels. A ce niveau de conception le circuit peut être décrit en *VHDL* et en suite implémenté sur différentes technologies d'*FPGA*. La simulation peut être effectuée après la génération du code *VHDL* correspondant au circuit de commande au cours du prochain chapitre.

CHAPITRE IV

Simulation et synthèse du circuit de commande

OBJECTIF

L'objectif de cette partie est de coder en *VHDL* puis simuler les différentes architectures retenues pour garantir les performances. Enfin après avoir évalué leurs performances, nous injectons la solution sur une cible *FPGA* avec un environnement *CAO*.

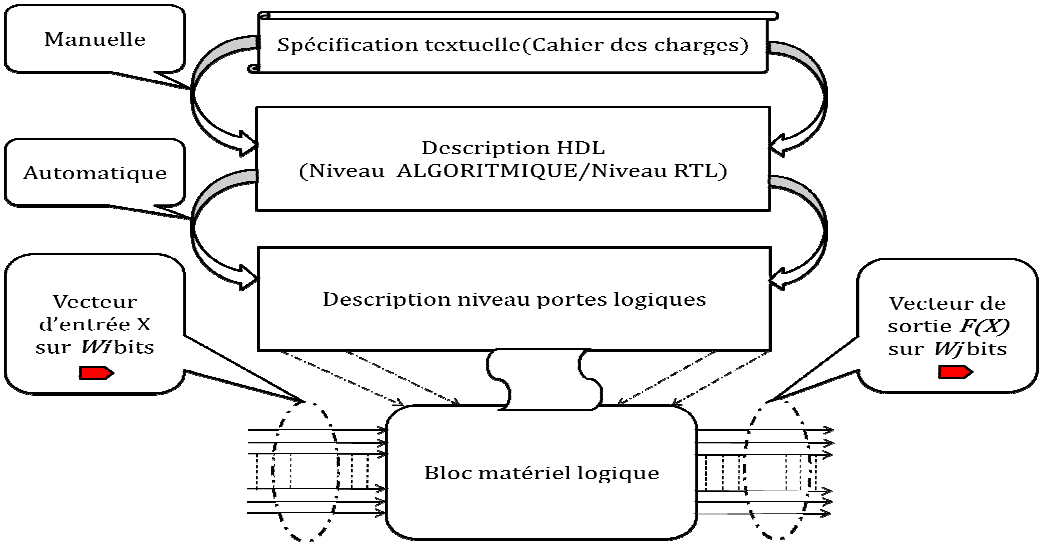
IV.1)-INTRODUCTION

Comme nous l'avons dit au précédent chapitre, 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 sûreté 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 de *CAO* (***C****onception* ***A****ssistée par* ***O****rdinateur*) microélectronique indispensable. Les outils de *CAO* ne se limitent pas à la simulation mais aussi à la synthèse des circuits et assurent les transitions entre les différents niveaux d'abstraction sachons que le passage de l'algorithme vers l'architecture est une synthèse automatique confiée aux logiciels *CAO* qui génèrent l'architecture adéquate du système. 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 la description et de son circuit synthétisé. Cette partie du travaille qui s'agit de concevoir, à partir du langage *VHDL,* un modèle équivalent au circuit de commande sachons qu'un modèle d'un circuit n'est qu'une abstraction de son comportement. Ce modèle va nous permettre de tester les lois de commande et de nous assurer de la validité de chaque partie grâce à l'utilisation d'un logiciel de simulation intégrer dans l'environnement *CAO*. Il est fondamentale d'apporter la preuve mathématique du bon fonctionnement du circuit ou bien d'émuler sont fonctionnement. La simulation du système est faite pour différentes raisons non seulement de vérifier la validité du code mais aussi un outil d'analyse permettant de prévoir le comportement du système sous l'action d'un événement particulier et la visualisation de son évolution temporelle. La dernière phase cette conception consiste à faire migrer cet algorithme sur un démonstrateur matériel *FPGA* afin de vérifier sa faisabilité technique et d'évaluer ses performances. Donc, ce chapitre s'occupe de la conversion vers un format exicutable(*langage de description materiel*) de la commande vectorielle puis la verification du circuit integré correspondant à cette commande en utilisant des outils *CAO* de *XILINX*.

IV.2)- DESCRIPTION EN VHDL DU CIRCUIT DE COMMANDE

On est sensé d'élaborer une plate-forme matérielle dédiée à la commande du moteur asynchrone basée sur des composants du type « *System On Chip* » ou « *System On Programmable Chip* », entièrement portable puisque elle sera décrite en langage *VHDL* qui est indépendant vis-à-vis de la technologie matériel cible (portabilité). Nous allons adopter une méthodologie de conception modulaire qui est à base de composant génériques (*Une bibliothèque de modules IP réutilisables*). Cependant, il existe différentes manières de faire décrire un circuit numérique sur différents niveaux d'abstraction. On peut mélanger des blocs de niveaux différents pour la simulation dans certains outils les plus élaborés.

- **La description comportementale *:*** La modélisation ou la description comportementale consiste à faire décrire le fonctionnement attendu d'un circuit explicitement et indépendamment de l'architecture matérielle du système. Ce genre de description convient pour une simulation purement logicielle du circuit mais déconseillée pour la synthèse de circuits.
- **La description structurelle :** Contrairement aux descriptions comportementales cette description est inspirée des structures materielles où les objets manipulés sont des objets qui decrivent directement le matériel.
- **La description bas-niveau :** C'est une description tres détaillée au niveau de portes logiques. Elle est interessante dans le cas où le circuit est simple (des fonctions combinatoires et des registres) et déconseillée en cas de circuits complexes.



Figure(IV.01)*:Description d'un opérateur matériel élémentaire.*

La démarche que nous avons suivie pour décrire et vérifier le circuit est résumé comme suit :

- Analyse et décomposition du système de commande vectoriel en simples blocs.
- Description en *VHDL* et validation de chaque bloc.
- Rassembler les blocs élémentaires du système.

Nous avons modélisé chaque composant séparément et nous avons stocké les modèles dans des bibliothèques de composants réutilisables afin d'économiser le temps de développement. Des modules supplémentaires peuvent être ajoutés si nécessaire car le langage de description matérielle *VHDL* facilite la modification et la réutilisation d'un design.

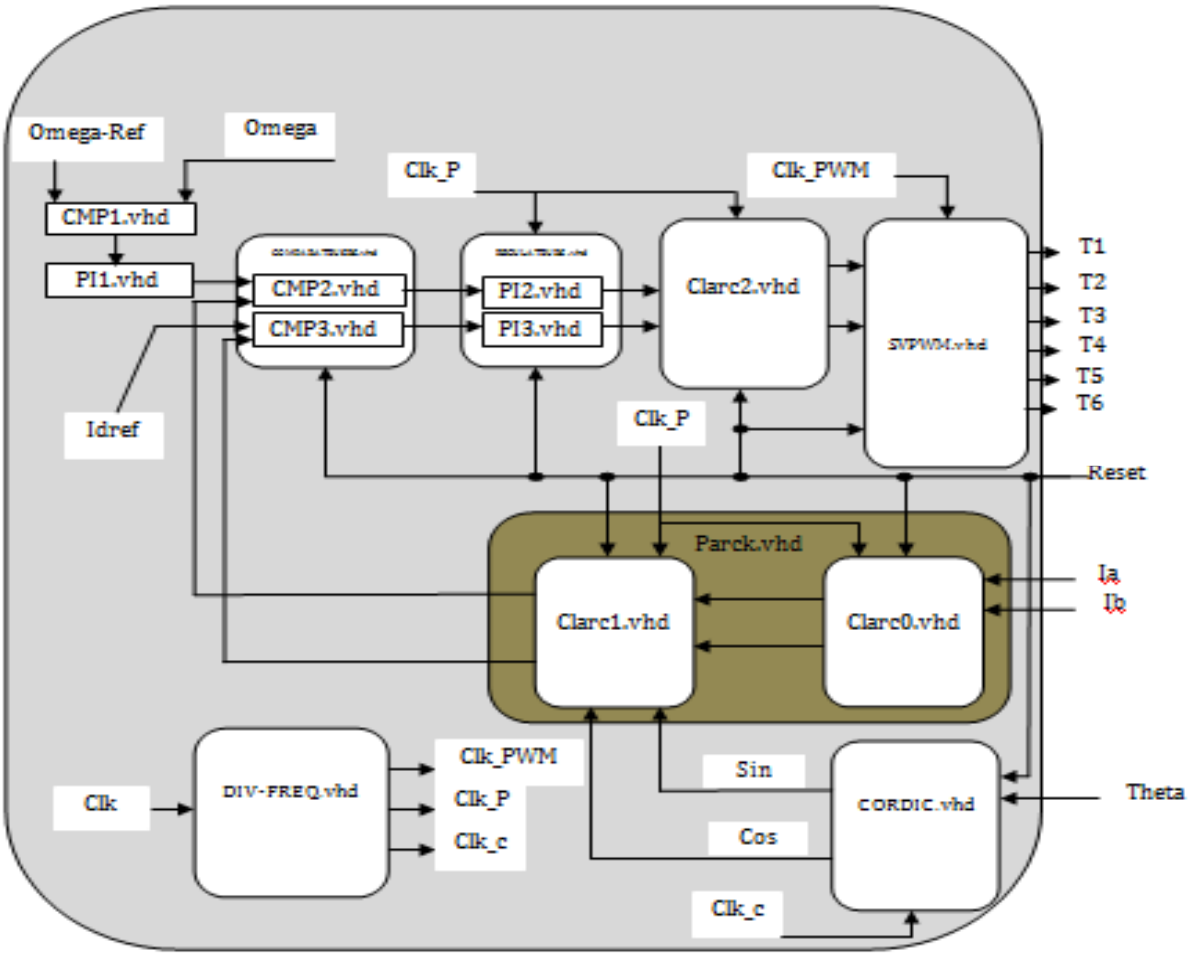


***Figure (IV.02)**:Schéma hiérarchique des modules de la commande développés en VHDL.*

Pour ce model en *VHDL* du contrôle vectoriel, les blocs individuels sont développés et testés et nous allons exposer les résultats dans ce qui suit dans ce travail.

Nous avons respecté d'adopter le même nom pour un signal injecté entre les différents blocs afin que le model complet sera assemblé. Nous avons aussi respecté le mode de chaque signal lors de sa déclaration dans le programme car c'est fondamentale en VHDL de déclarer le type du signal **(*Les modes in et out*, *Le mode buffer, Le mode inout*)**.Enfin, Nous avons opté de représenter sur *16 bits* les signaux d'entrées/sorties de tout le système (courants, tensions, vitesse…etc.) afin d'avoir une bonne précision.

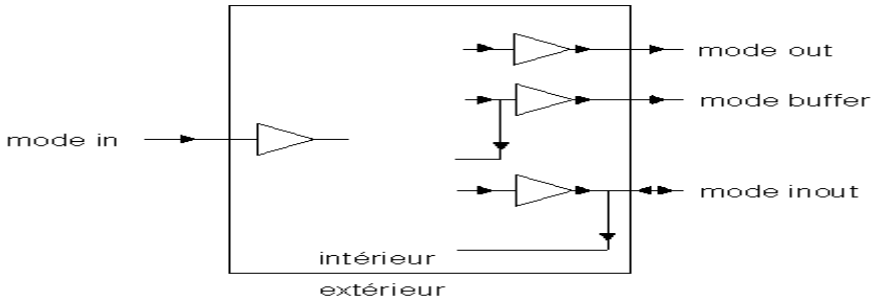


***Figure (IV.03)**:Schéma des différents modes du signal.*

IV.3)-PRESENTATION DU LOGICIEL XILINX ISE

Le logiciel *XILINX ISE (**I**ntegrated **S**oftware **E**nvironment)* est un environnement de développement qui possède différent outils de *CAO*. Les sociétés spécialisées en *CAO* microélectronique fournissent des environnements logiciels spécialisés. Tous les fabricants de *FPGA* proposent des outils de *CAO* pour configurer leurs circuits (*XILINX* c'est *ISE - Foundation* pour *ALTERA* c'est *QUARTUS* ou *MAX + II*). 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 *XILINX ISE* qui est un logiciel de création et de gestion de projets *CAO* où un environnement de conception. C'est un logiciel multitâche qui possède dans son soft différents outils permettant la création de 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*) sachons que la saisie graphique est une alternative à la saisie textuelle mais limitée. Ce logiciel *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 a une technologie *FPGA* bien précise et enfin lorsque toute les vérification sont faites vient l'implantation sur un *FPGA* réel ce qui correspond a générer le fichier de configuration du circuit cible choisi afin d'établir les interconnections des cellules logiques correspondantes 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 *FPFG* de *XILINX* et même pour d'autres circuits programmables telque les *CPLD*.

La conception de circuits sur *XILINX ISE* met en œuvre quatre outils : un éditeur de texte ou entrée graphique, un simulateur, un synthétiseur et un placeur-routeur. L'éditeur de texte ou entrée graphique est pour faire introduire la description dans les logiciels *CAO* c'est à dire de dessiner ou décrire le circuit avec une interface graphique ou textuelle. La simulation du système est faite pour vérifier la validité du code avant-synthèse, après-synthèse et même après placement-routage. Les deux étapes synthèse et routage succéderons par la suite où la synthèse consiste à faire la transcription de la description d'une forme texte vers une autre graphique(*RTL*) à base de portes logiques et pour la deuxième étape nommée routage, n'est qu'une adaptation du circuit logique synthétisé sur les ressource disponible dans le circuit *FPGA* ciblé.

IV.4)-SYNTHESE ET SIMULATION DU CIRCUIT

Comme nous l'avons vu au premier chapitre, la conception du circuit met en œuvre quatre outils : un éditeur de texte/entrée graphique, un simulateur, un synthétiseur et un placeur-routeur. La description du circuit 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 au troisième chapitre pour le circuit numérique de commande vectorielle. Par la suite, nous allons réaliser une simulation manuelle en appliquant des valeurs sur les entrées et en vérifiant visuellement l'état des sorties.

IV.4.1)-Synthèse et simulation des éléments logiques de base

IV.4.1.1)- Additionneur 16Bits

A) Résultat de synthèse

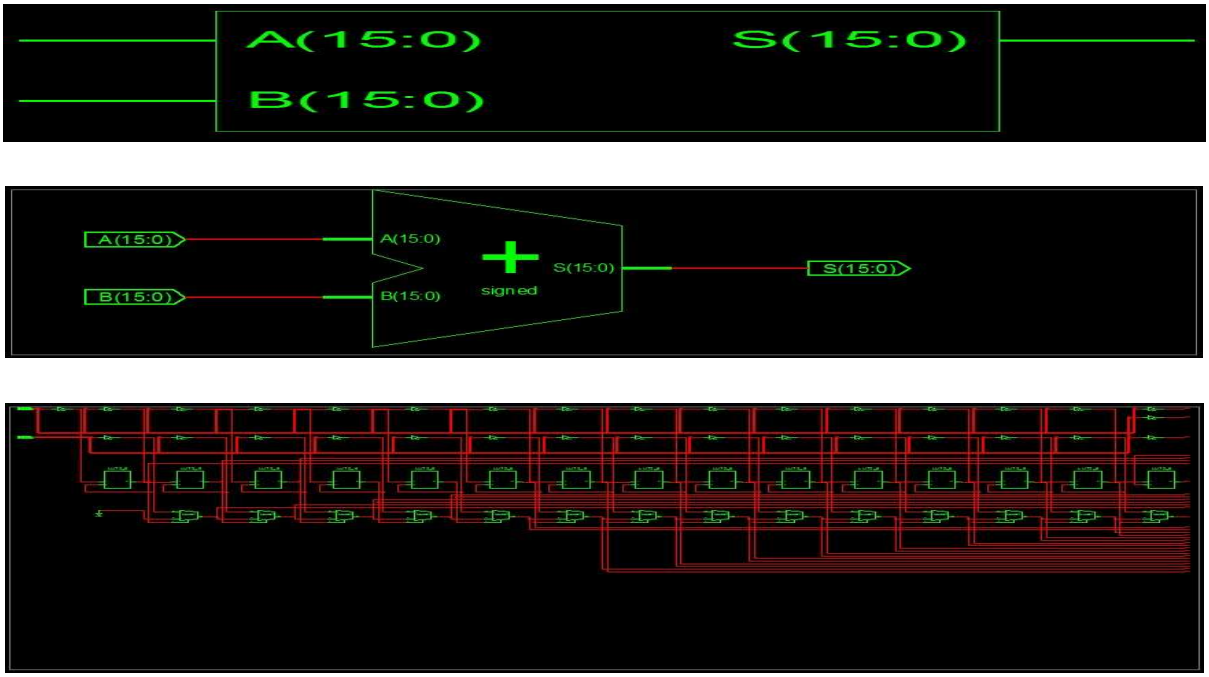


***Figure (IV.04)**:Vue externe, interne(RTL) et technologique du module additionneur à 16 bits.*

B) Simulation

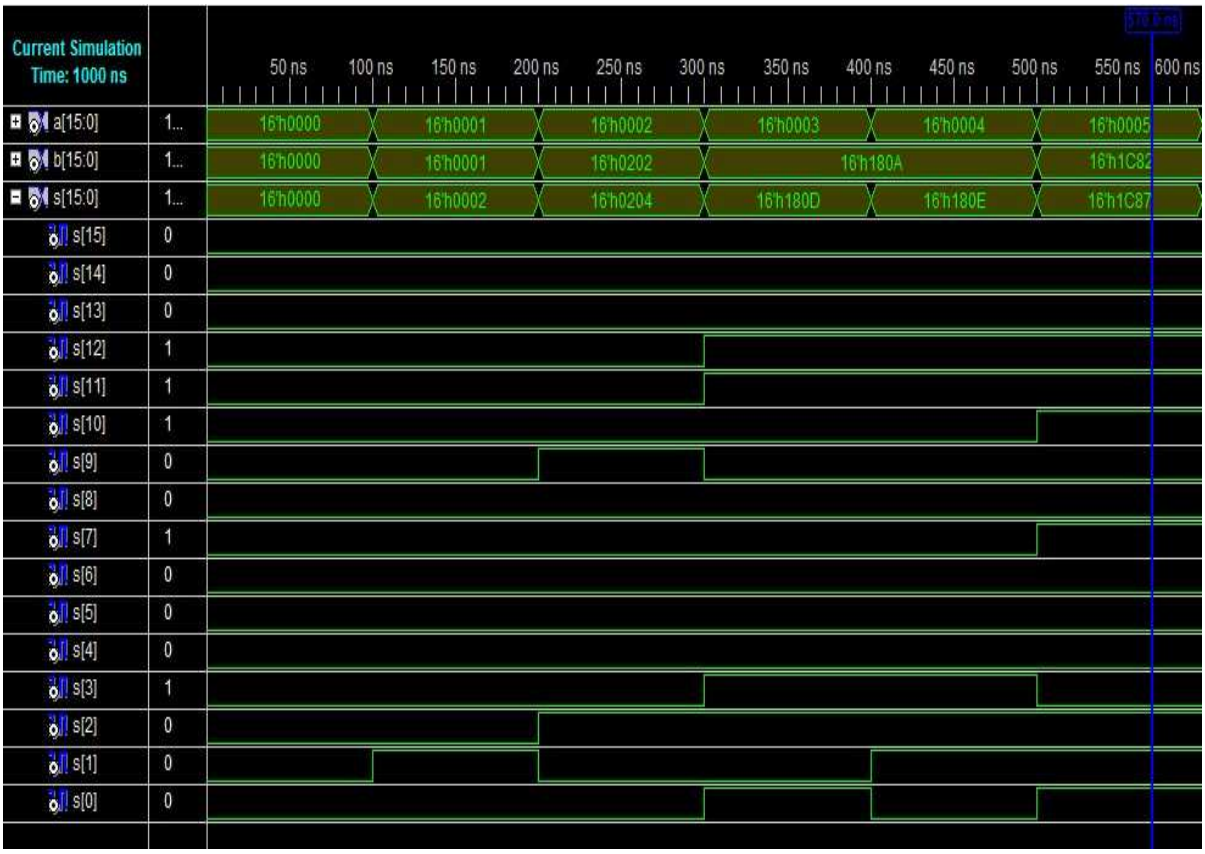


***Figure (IV.05)**:Résultats de simulation du module additionneur signé à 16 bits.*

C) Interprétation des résultats

On remarque que le chronogramme est subdivisé en deux partie la première en haut représente les entrées/sorties sous forme de bus codé en hexadécimal car les données ne sont pas sont des paquets de bits que la fenêtre de simulation ne peut pas les contenir en représentant chaque bit dus bus. La deuxième est la partie détaillée du bus de sortie encodé dans notre cas sur *16 bits*. C'est nous qui avons ouvert le bus de sortie pour expliciter les résultats. Nous avons testé l'additionneur en lui présentons six valeurs en entrées ***a*** et ***b*** pour être sommés. Comme le montre le diagramme sachons que les valeurs sont en hexadécimales :

*(0000+0000=0000), (0001+0001=0002), (0002+0202=0204), (0003+1801=180D), (0004+1801=180E), (0005+1C82=1C87).*Donc effectivement ce composant fait la somme de deux nombres sur seize bits.

IV.4.1.2)-Soustracteur(SIGNE)

A) Résultat de synthèse

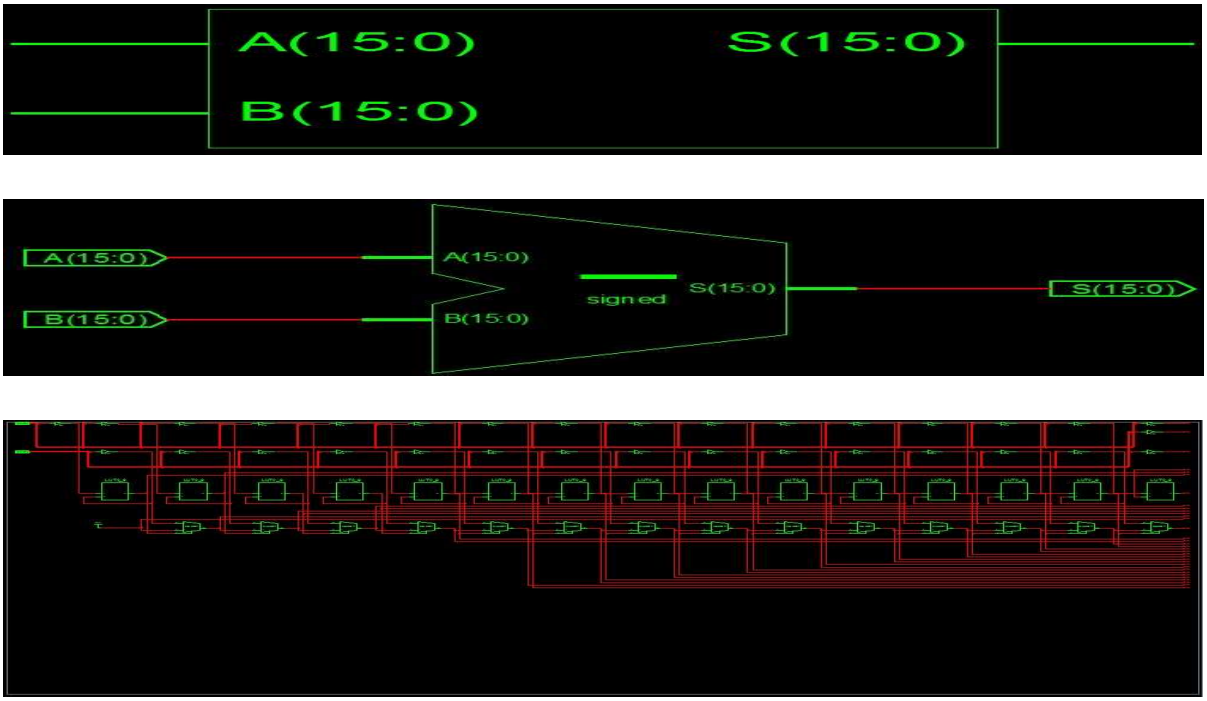


***Figure (IV.06)**:Vue externe, interne(RTL) et technologique du module soustracteur signé à 16 bits.*

B) Simulation

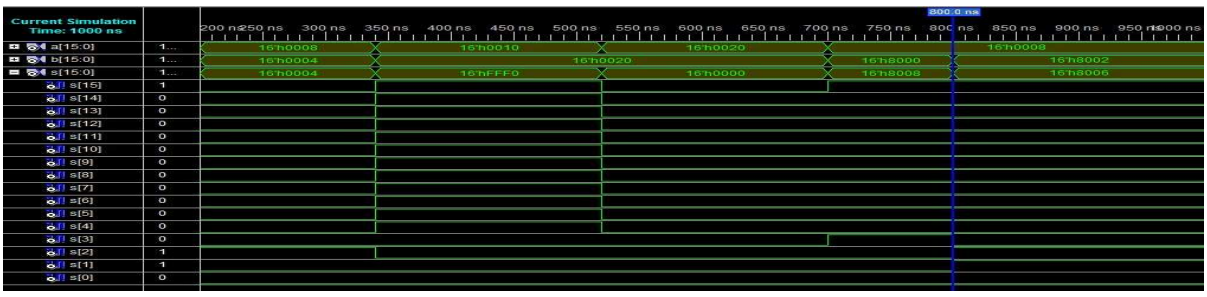


***Figure (IV.07)**:Résultats de simulation du module soustracteur signé à 16 bits.*

C) Interprétation des résultats

D'après la figure de simulation, nous avons donné des valeurs aux entrées *(a)* et *(b)* et nous avons récupérer la sortie*(s)*.Les valeurs sont en hexadécimale et comme exemple *(0008)h –(0004)h=(0004)h* et *(0010)h-(0020)h=(FFF0)h* ce dernier résultat est représenter en complément à deux car il est négatif .Enfin, même raisonnement pour les autres valeurs.

IV.4.1. 3)-MULTIPLIEUR(SIGNE)

A) Résultat de synthèse

***Figure (IV.08)**:Vue externe, interne(RTL) et échantillon technologique du module multiplieur signé à 16 bits.*

B) Simulation

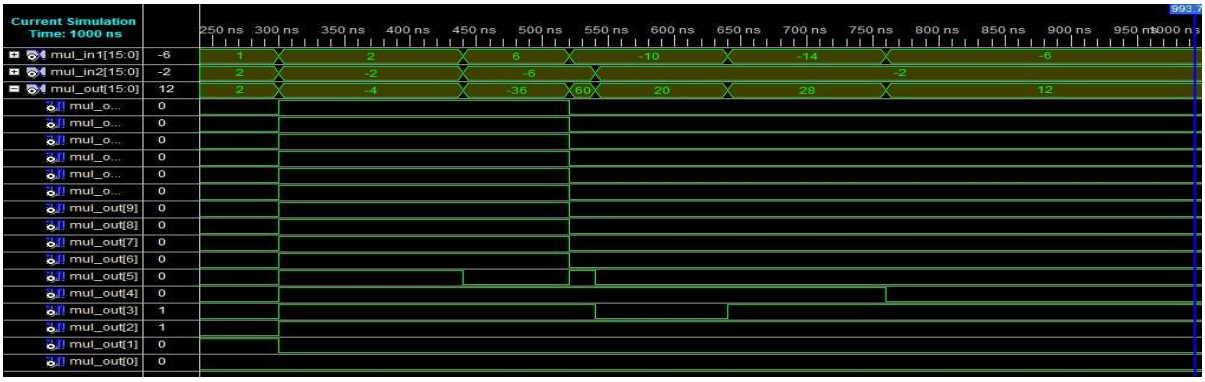


***Figure (IV.09)**:Résultats de simulation du module multiplieur signé à 16 bits.*

C) Interprétation des résultats

Comme le montre la simulation, les deux entrées *(MUL_in1 et MUL_in2)* sont multipliées et donnent la sortie*(MUL_out)*.Les valeurs sont en décimale et comme exemple *(1 x 2 =2), (2 x -2=-4), (6 x -6=-36), (-10 x -6=60), (-10 x -2=20) ….etc.*

IV.4.1.4)- Bascule D

A) Résultat de synthèse

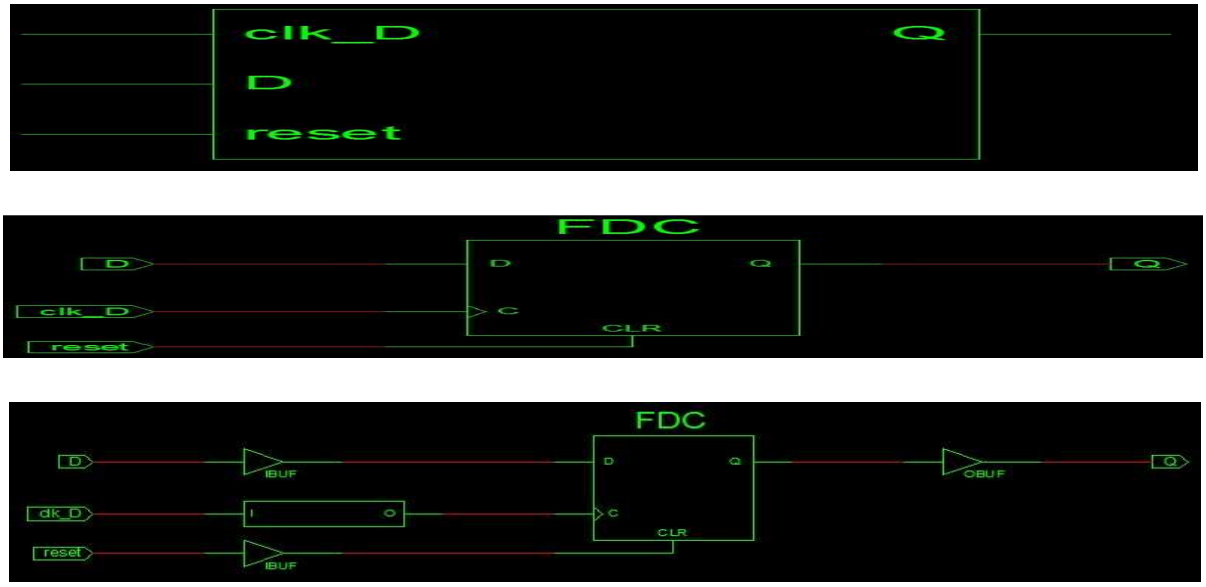


***Figure (IV.10)**:Vue externe, interne(RTL) et technologique du module de bascule **D**.*

B) Simulation

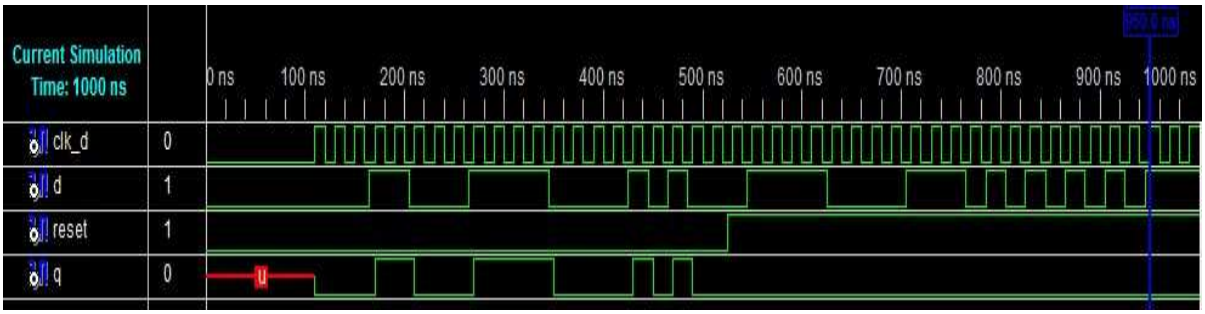


Figure (IV.11)**:Résultats de simulation du module de bascule **D.

C) Interprétation des résultats

D'après le chronogramme de simulation pour la bascule ***D***, on voie que la simulation du composant commence à partir de *100ns*. Le signal de remise a zéro ***reset*** est activé a partir de *510ns* quelque soit l'état de l'entre et le signal d'horloge la sortie ***q*** est mise a zéro. Durant la période entre*100ns* et *510ns* le signal **reset** n'est pas activé danc c'est le domaine ou la bascule peut récupérer les données d'entrée ***d***. Alors durant cette période la sortie ***q*** récupère exactement la donnée de l'entre d au front montant d'horloge ***clk_d.*** C'est ce qui correspond au comportement théorique de la bascule.

IV.4.1.5)- Registre 16Bits

A) Résultat de synthèse

***Figure (IV.12)**:Vue externe, interne(RTL) et technologique du module registre simple.*

B) Simulation

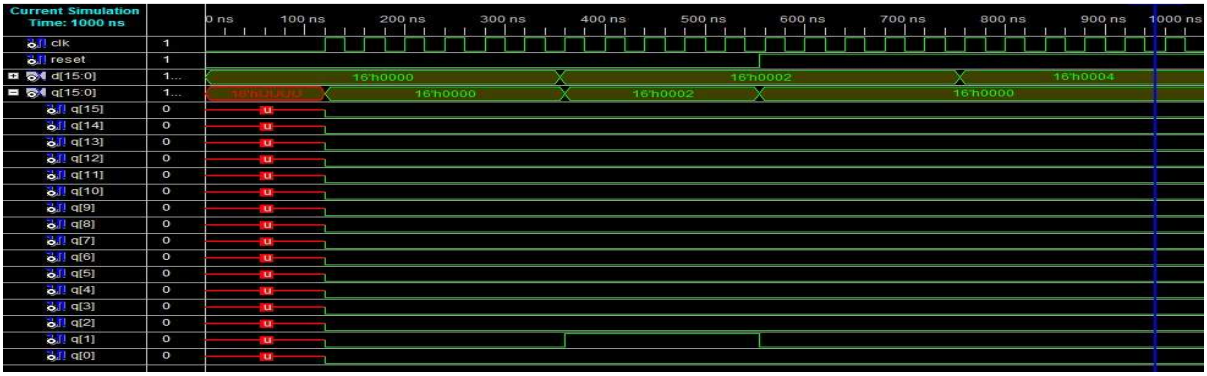


***Figure (IV.13)**:Résultats de simulation du module registre simple.*

C) Interprétation des résultats

La simulation commence a partir de *120ns* et se termite a *545ns* avec l'activation de remise a zéro ***reset***. Comme le chronogramme le montre les deux données en entrées *(0000)* et *(0002)* sont récupérer immédiatement en sortie au plus proche front montant d'horloge.

IV.4.1.6)- Registre a décalage 16Bits

A) Résultat de synthèse

***Figure (IV.14)**:Vue externe, interne(RTL) et technologique du module registre à décalage a 16Bits.*

B) Simulation

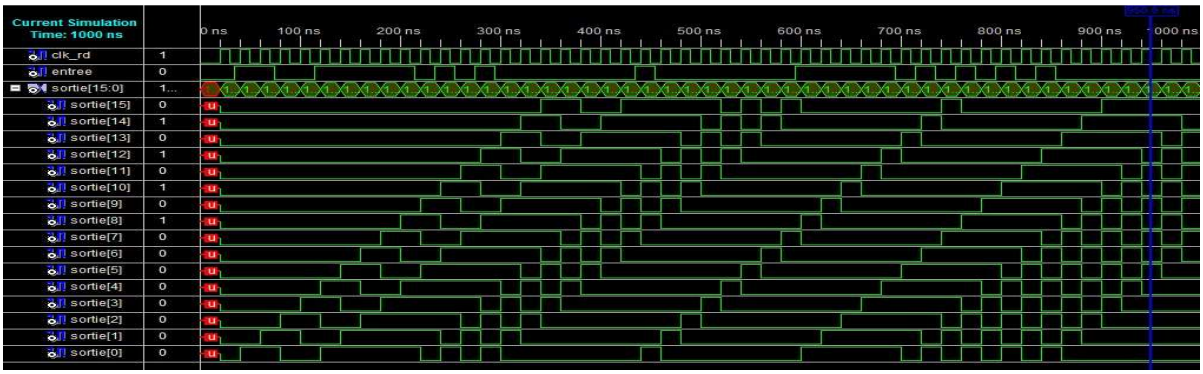


***Figure (IV.15)**:Résultats de simulation du module registre a décalage a 16Bits.*

C) Interprétation des résultats

La simulation montre qu'a chaque front montant d'horloge*(clk_rd)* la sortie*(sortie)* se décale d'une unité et la valeur de l'entrée*(entree)* est injectée au bit(0) de la sortie*(sortie)*.

IV.4.1.7)-Mémoire RAM

A) Résultat de synthèse

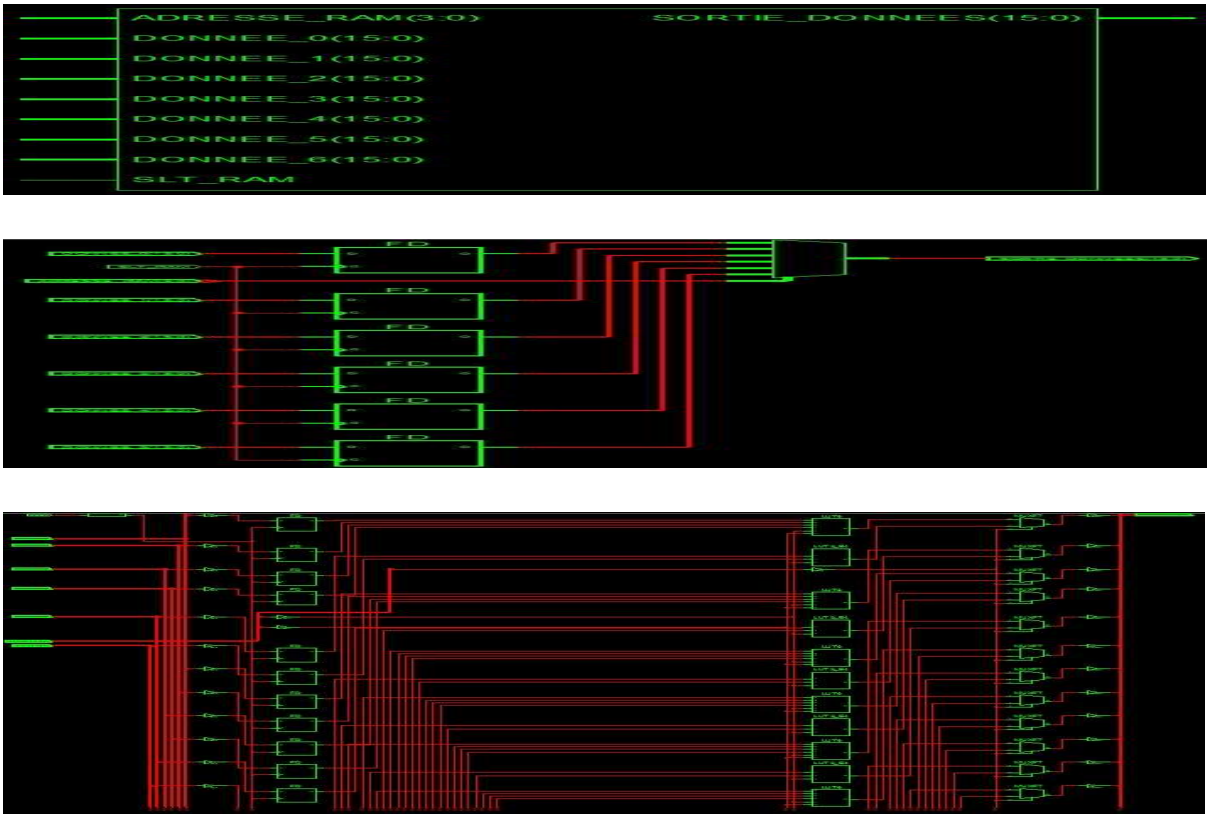


***Figure (IV.16)**:Vue externe, interne(RTL) et technologique du module RAM a 16Bits.*

B) Simulation

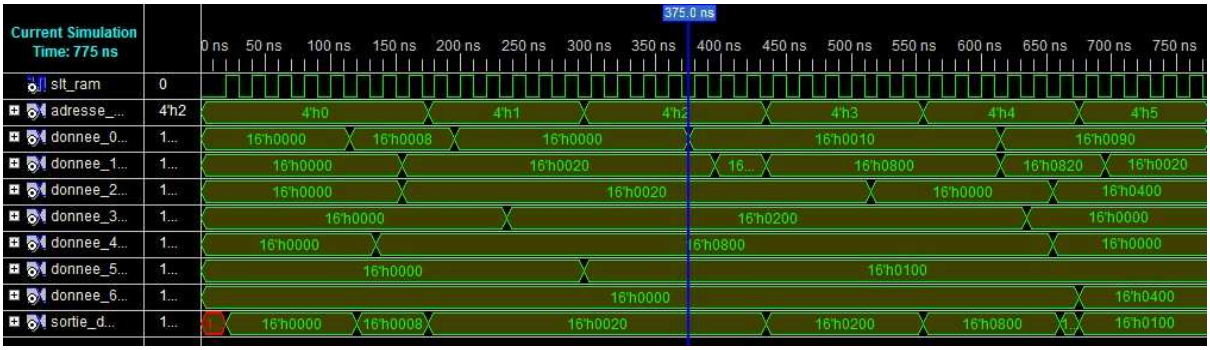


***Figure (IV.17)**:Résultats de simulation du module RAM à 16Bits.*

C) Interprétation des résultats

Une fois que la *RAM* est sélectionnée *(slt_ram)*, la donnée correspondante à l'adresse choisie sera récupérée en sortie.

IV.4.1.8)- Conteur 16Bits(Temporisateur)

A) Résultat de synthèse

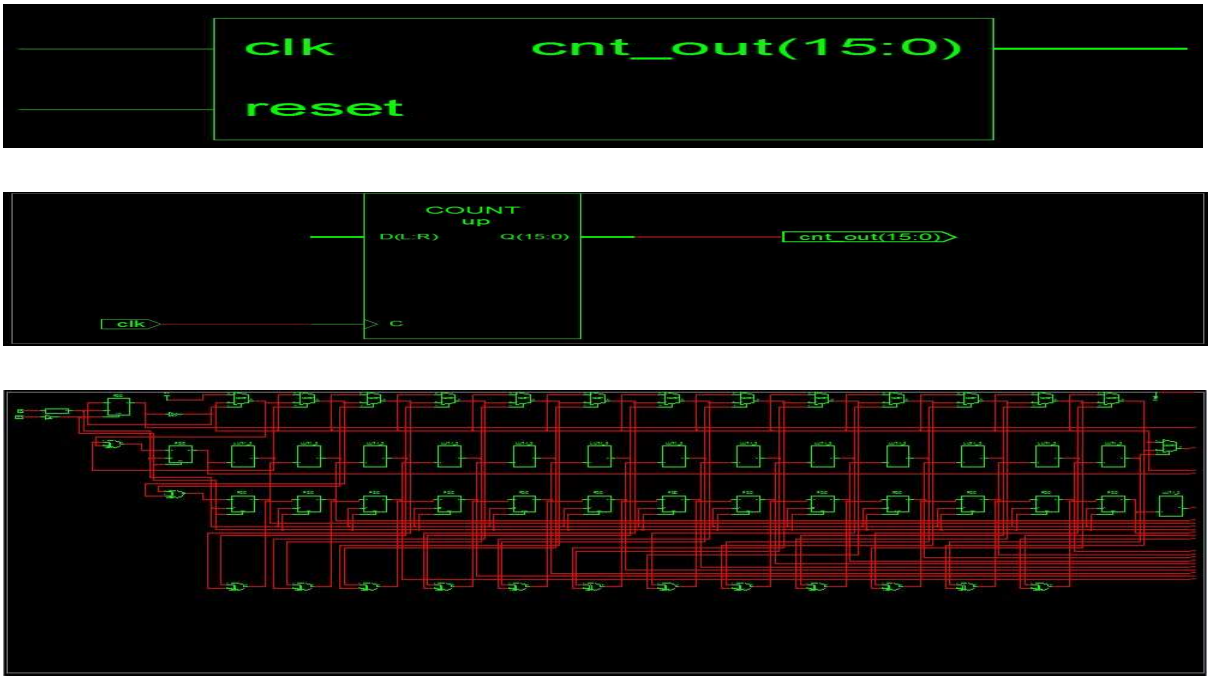


***Figure (IV.18)**:Vue externe et interne(RTL) du module de conteur d'impulsions à 16 Bits.*

B) Simulation

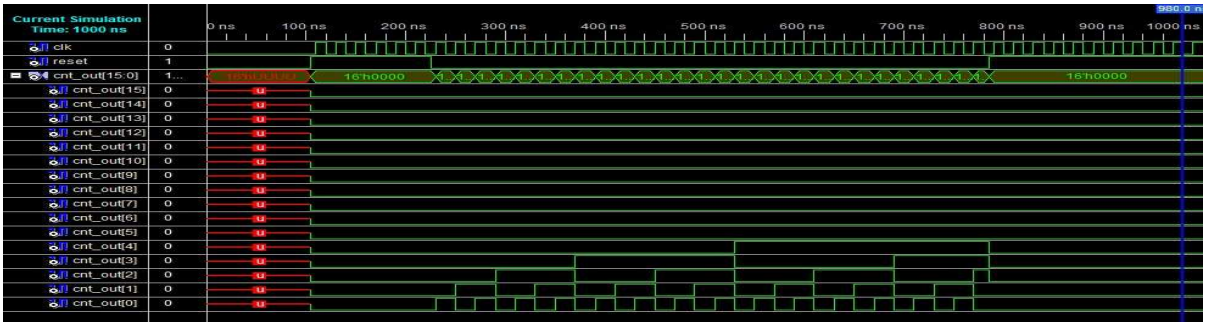


***Figure (IV.19)**:Résultats de simulation du module de conteur d'impulsions.*

C) Interprétation des résultats

Nous remarquons que la simulation commence à *100ns* et elle est remise à zéro durant deux périodes de temps (*100ns* a *225ns*) et après *780 ns*. Durant la période de fonctionnement le conteur fait sont travail en binaire qui est comme suit :(*0….0000*) après (*0….0001*) après *(0….0010)* après *(0….0011)* après *(0….0100)* après *(0….0101)* après *(0….0110)* après (*0….0111)*….etc. C'est ce qui corresponde au comptage décimal : *1, 2, 3, 4, 5,6, 7,…* etc.

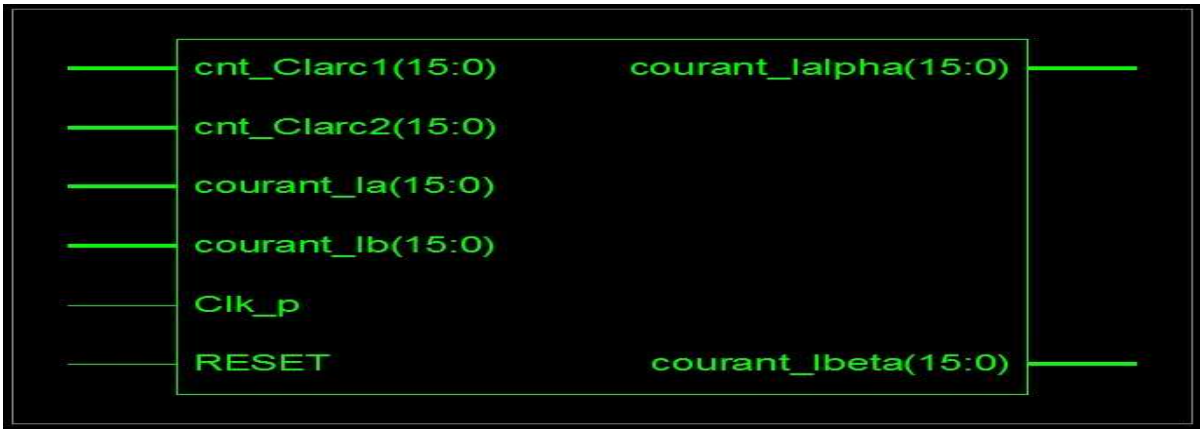
IV.4.2)-Synthèse des éléments de la commande vectorielle

Apres avoir réussir de synthétiser les éléments logiques de base, nous allons les exploiter pour synthétiser les blocs de la commande vectorielle que nous avons élaboré au précédent chapitre sachons que le synthétiseur du logiciel *XILINX ISE10* peut améliorer la synthèse du circuit logique. Pour ce qui concerne la simulation du circuit de commande vectorielle, nous allons simuler chaque blocs car plus le test des blocs individuels est rigoureux, moins le test du système complet est laborieux.

IV.4.2.1)-Synthèse du bloc de la transformé de PARK

IV.4.2.1 .1)-Sous bloc Clarc0

A) Résultat de synthèse



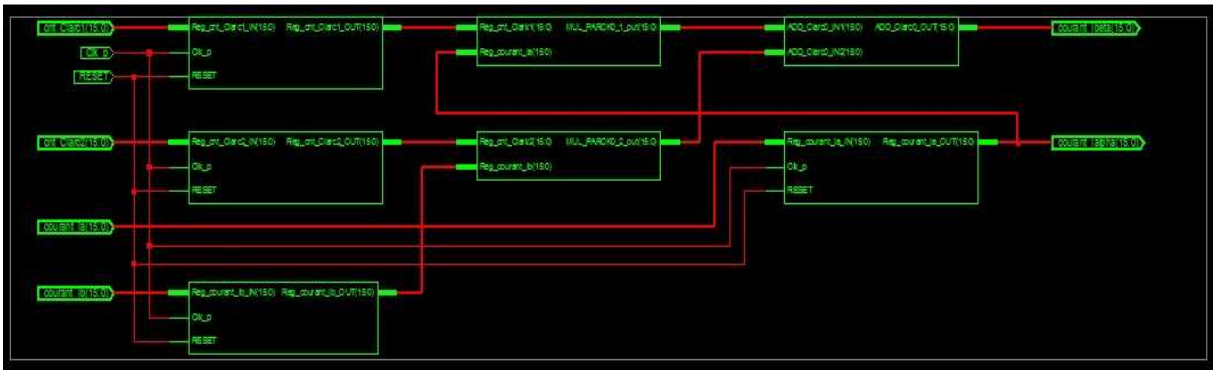


***Figure (IV.20)**:Vue externe, interne(RTL) et échantillon technologique du sous bloc Clarc0.*

IV.4.2.1 .2)-Sous bloc Clarc1

A) Résultat de synthèse

***Figure (IV.21)**:Vue externe, interne(RTL) et échantillon technologique du sous bloc Clarc1.*

IV.4.2.1 .3)-Bloc générale de Park

A) Résultat de synthèse

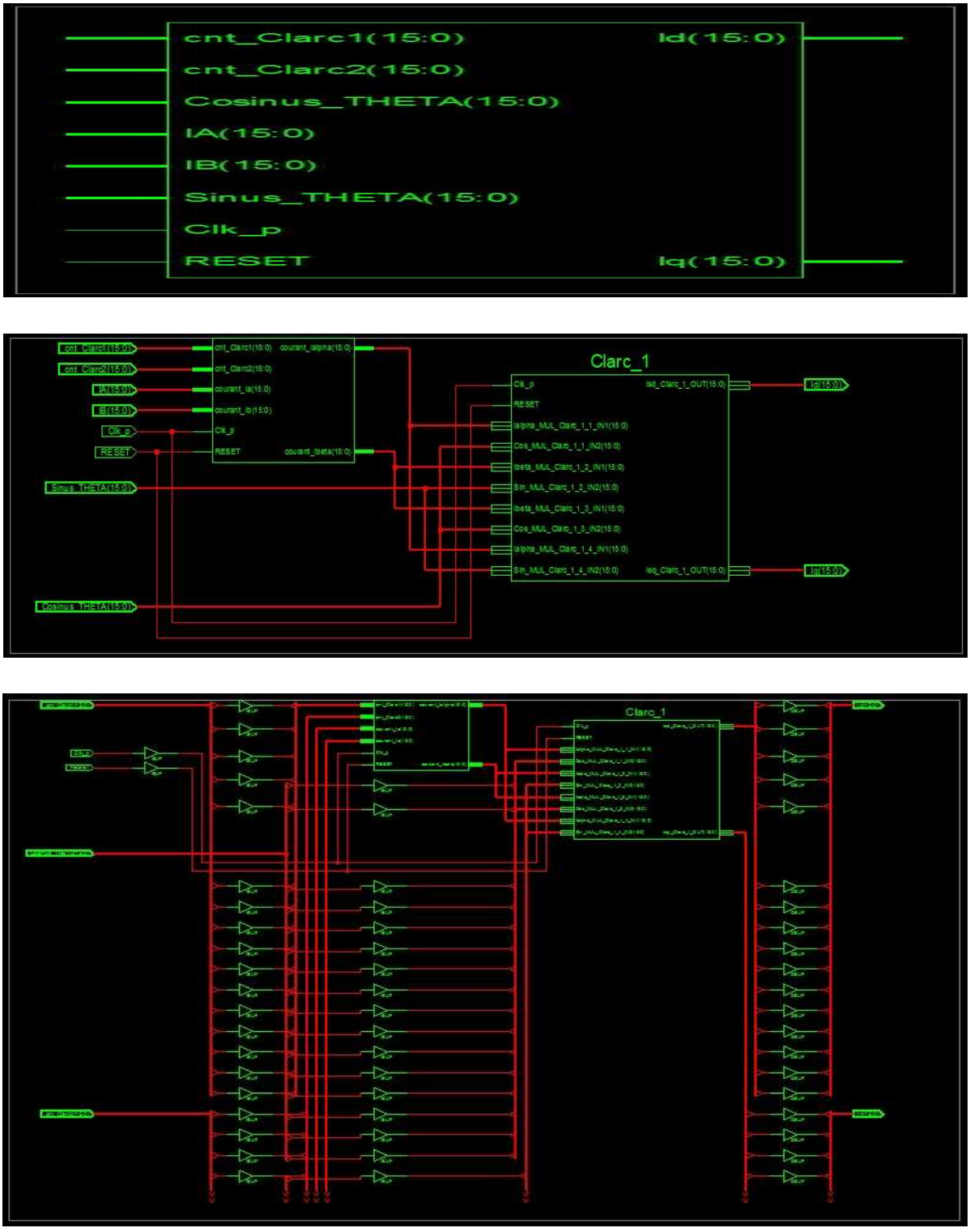


***Figure (IV.22)**:Vue externe, interne(RTL) et échantillon technologique du sous bloc matrice de Park.*

IV.4.2.2)-Synthèse des blocs de régulateurs « PI »

Comme la structure des régulateurs *PI* qui sont dans la commande est la même, nous présenterons les résultats d'un parmi eux.

A) Résultat de synthèse

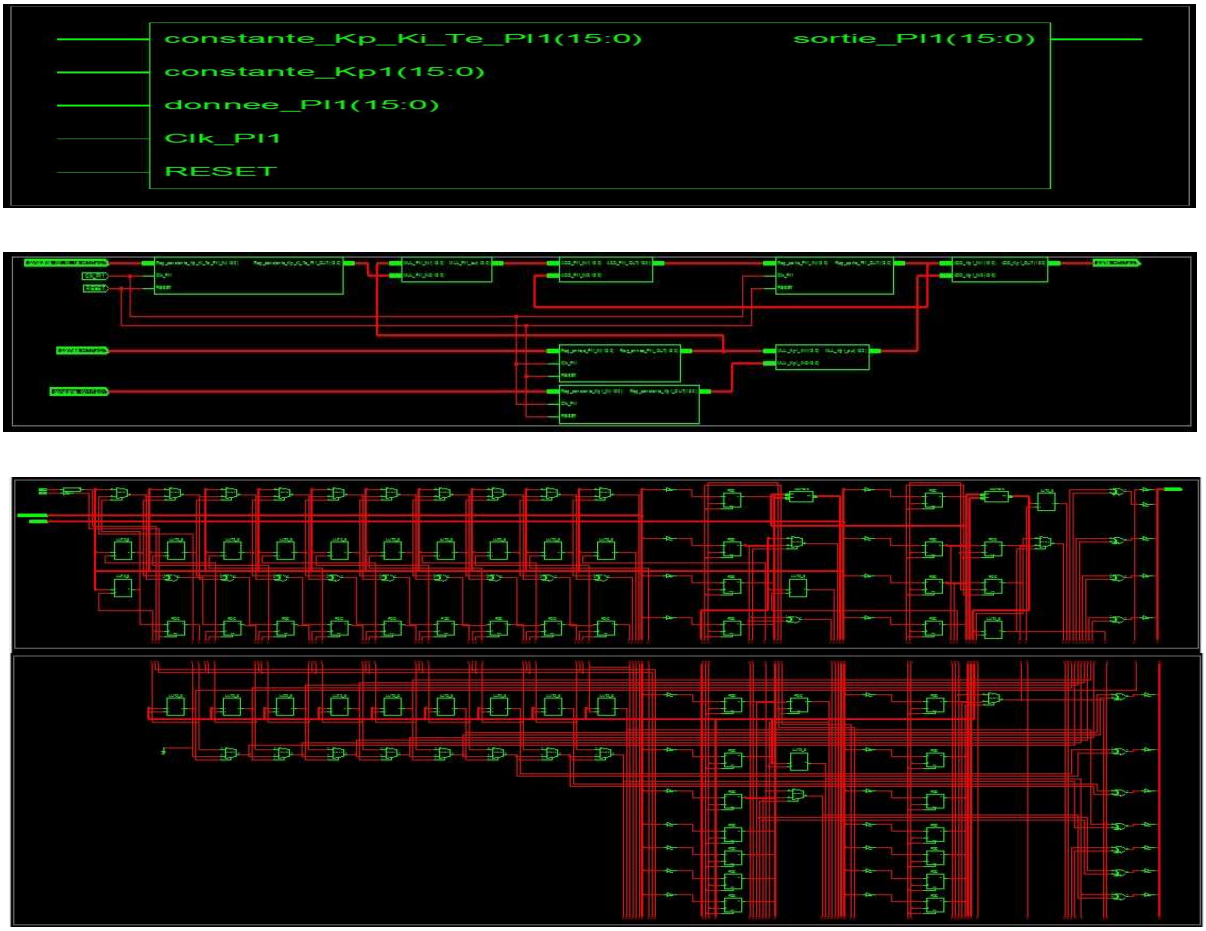


***Figure (IV.23)**:Vue externe, interne(RTL) et échantillon technologique du module régulateur PI1.*

IV.4.2.3) -Synthèse du bloc de modulation « SVPWM »

IV.4.2.3 .1) -Synthèse et simulation de la ROM de SVPWM

A) Résultat de synthèse

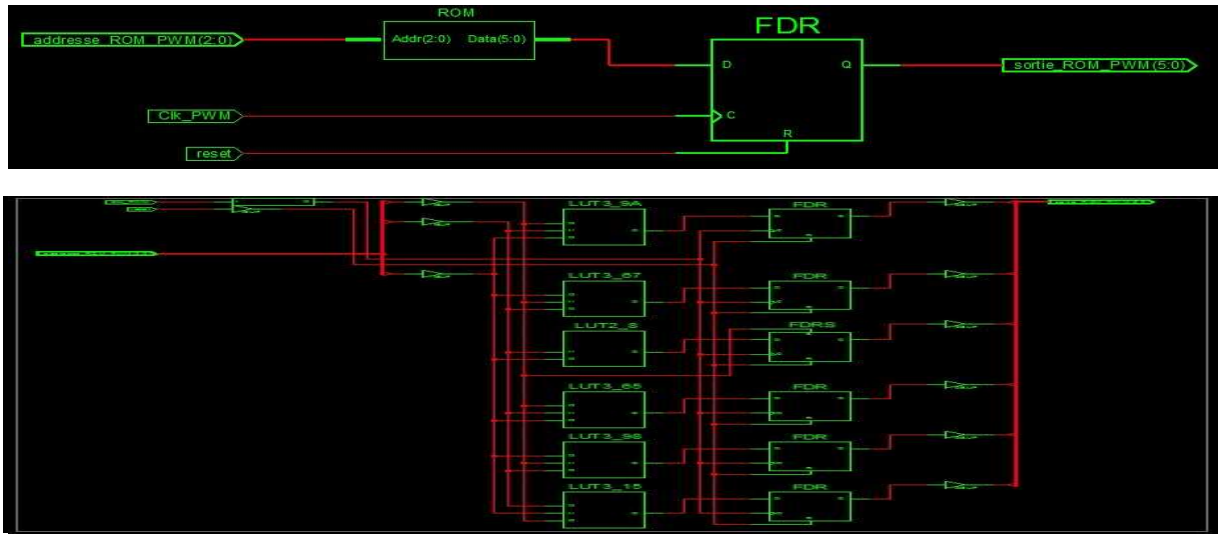


***Figure (IV.24)**:Vue externe, interne(RTL) et technologique du module ROM de la SVPWM.*

B) Simulation

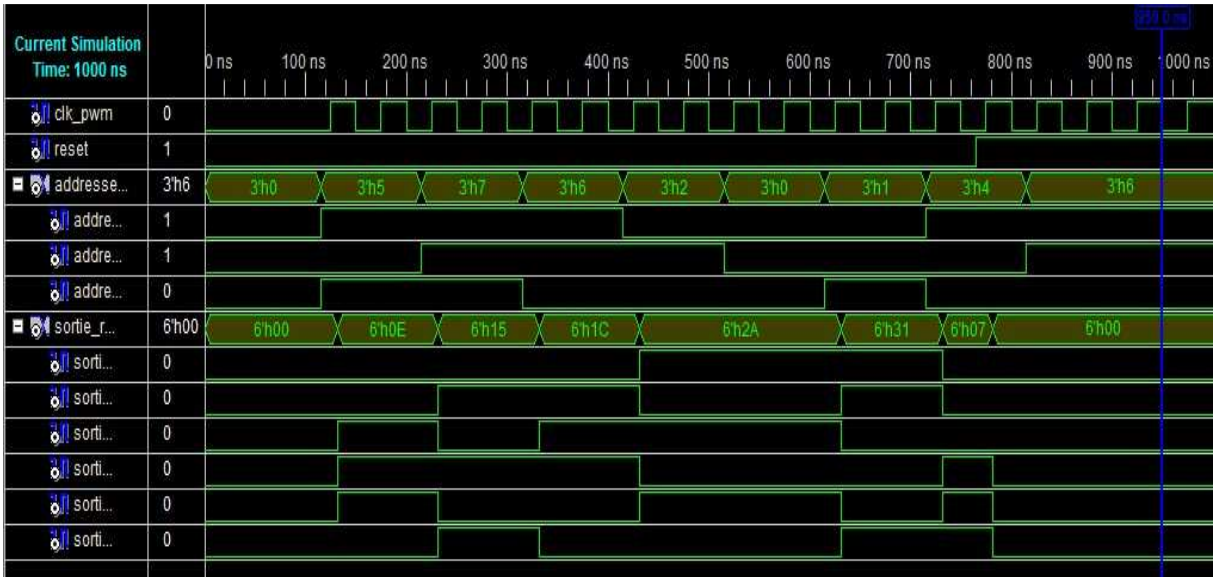


***Figure (IV.25)**:Résultats de simulation du module ROM de la SVPWM.*

C) Interprétation des résultats

Comme le montre la figure, la sortie présente une symétrie entre les trois bits du haut et les trois dessous ce qui correspond a la complémentarité des deux interrupteurs d'un bras d'onduleur. Lorsqu'on sélectionne une adresse, une sortie sera correspondante et ce processus suit les tableaux de la SVPWM simplifiée qui est présenté au précédent chapitre. Comme exemple ((5)h=(101)b)correspond a la sortie ((0E)h=(001 110)b)….etc.

IV.4.2.3 .2) -Synthèse de la SVPWM

A) Résultat de synthèse

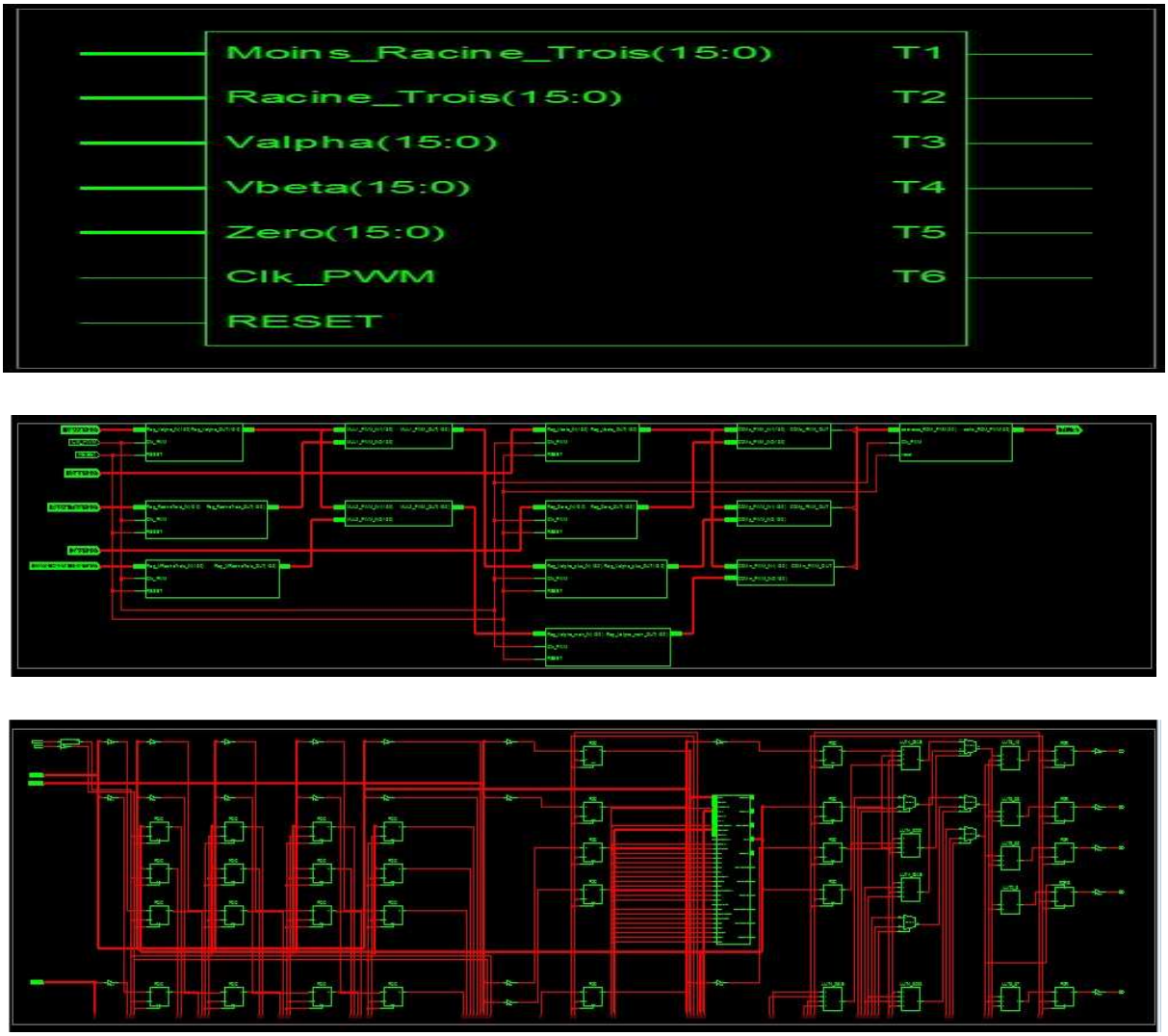


***Figure (IV.26)**:Vue externe, interne et échantillon technologique(RTL) du module général de la SVPWM.*

IV.4.3)-Synthèse du bloc CORDIC

A) Résultat de synthèse

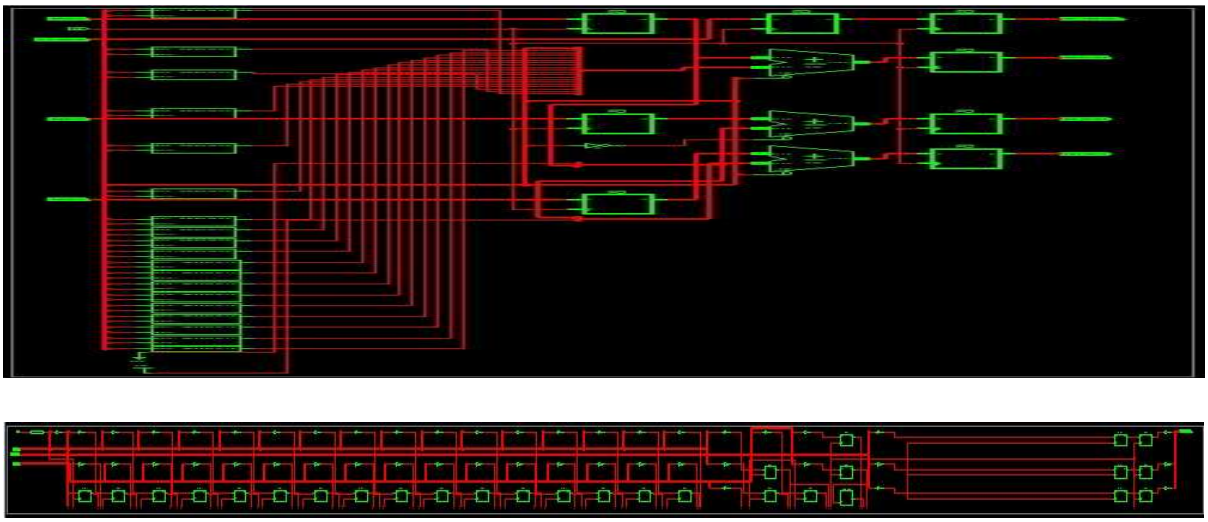
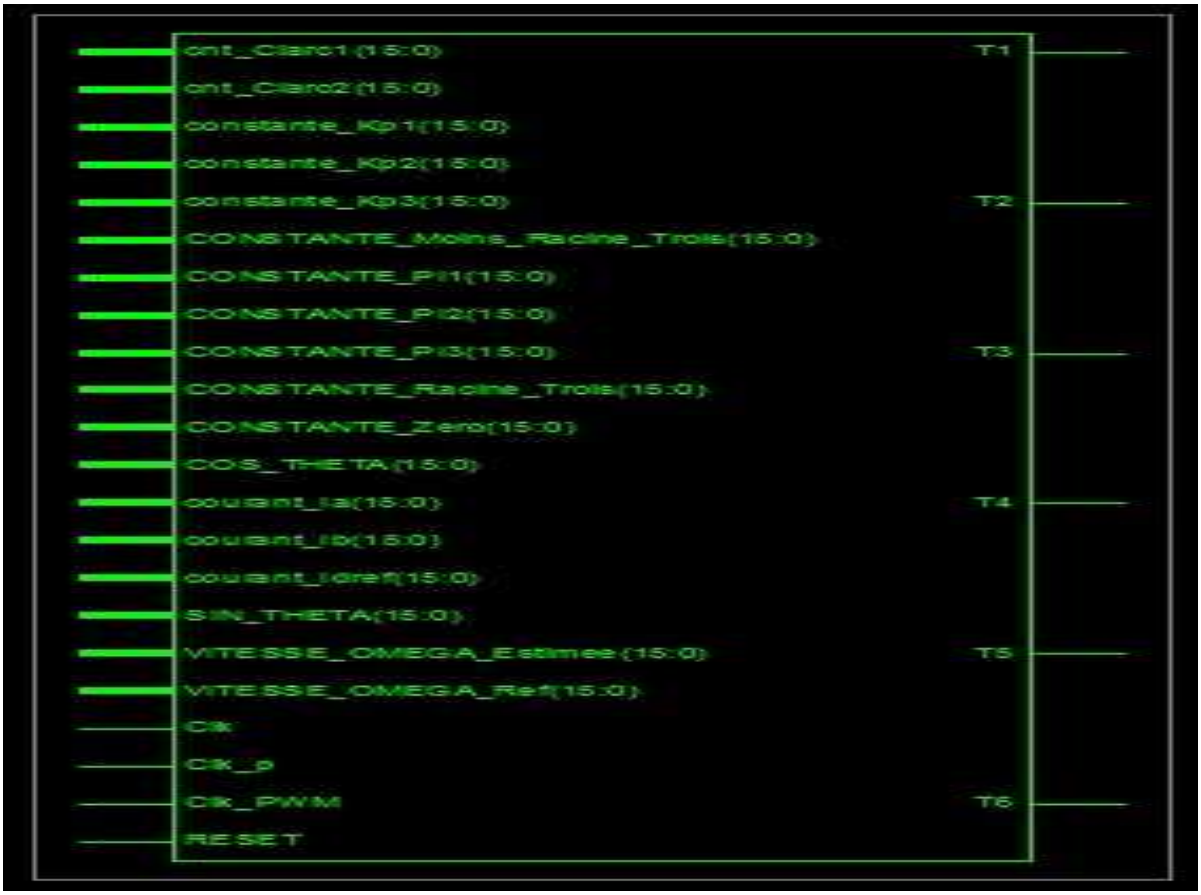


Figure (IV.27):*Vue externe et interne(RTL) et échantillon technologique du module du Cordic à 16 Bits.*

IV.4.4)-Synthèse du bloc générale de la commande vectorielle

A) Résultat de synthèse



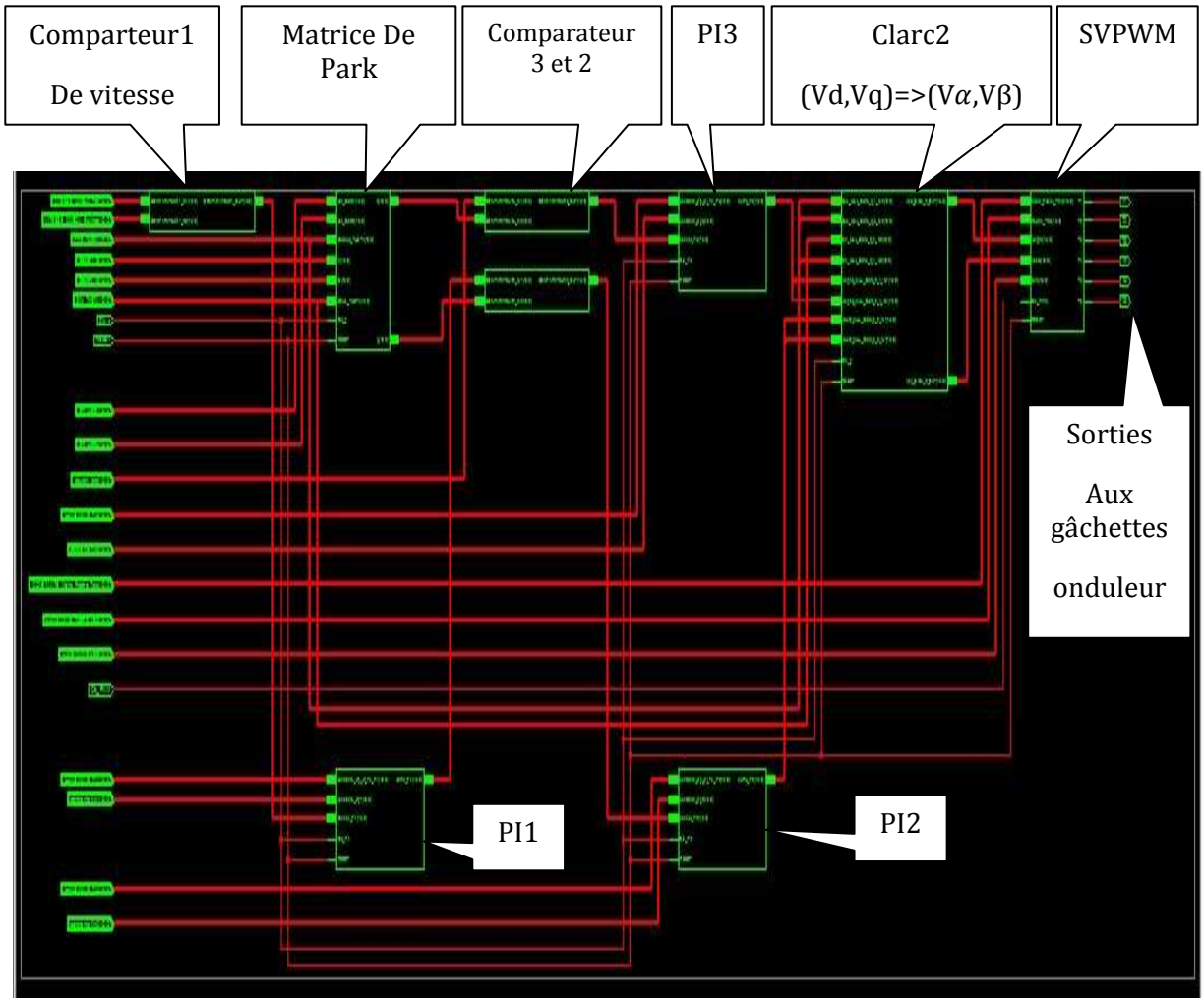


***Figure (IV.28)**: Vue externe et interne(RTL) du module générale de la commande vectorielle.*

IV.5)-LE RAPPORT DE CONSOMMATION DES RESSOURCES

COMMANDE_VECTORIELLE Project Status			
Project File:	COMMANDE_VECTORIELLE.ise	**Current State:**	Synthesized
Module Name:	commande_vectorielle	**• Errors:**	No Errors
Target Device:	xc2v500-4fg456	**• Warnings:**	No Warnings
Product Version:	ISE 10.1 - WebPACK	**• Routing Results:**	
Design Goal:	Balanced	**• Timing Constraints:**	
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	

COMMANDE_VECTORIELLE Partition Summary
No partition information was found.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	320	3072	10%
Number of Slice Flip Flops	454	6144	7%
Number of 4 input LUTs	214	6144	3%
Number of bonded IOBs	249	264	94%
Number of MULT18X18s	15	32	46%
Number of GCLKs	2	16	12%

***Tableau (IV.01)**: Ressources consommées par la commande vectorielle.*

IV.6)-TRANSFERT DE LA SOLUTION VERS UN SUPPORT PHYSIQUE FPGA

Cette étape est la dernière dans le processus de conception en général et en particulier de notre projet. Elle va nous permettre de charger le fichier de description *VHDL* sur un support *FPGA* (*Migration de la solution vers la cible hardware*) choisi convenable. Pour notre cas, une interface fournie par *XILINX* pour la configuration du *FPGA* par un fichier en format *BIT*. Le transfert de la solution finale peut être assuré par un câble *JTAG* (*Joint Test Action Group*) qui relie le micro-ordinateur*(PC)* et la carte *FPGA*.

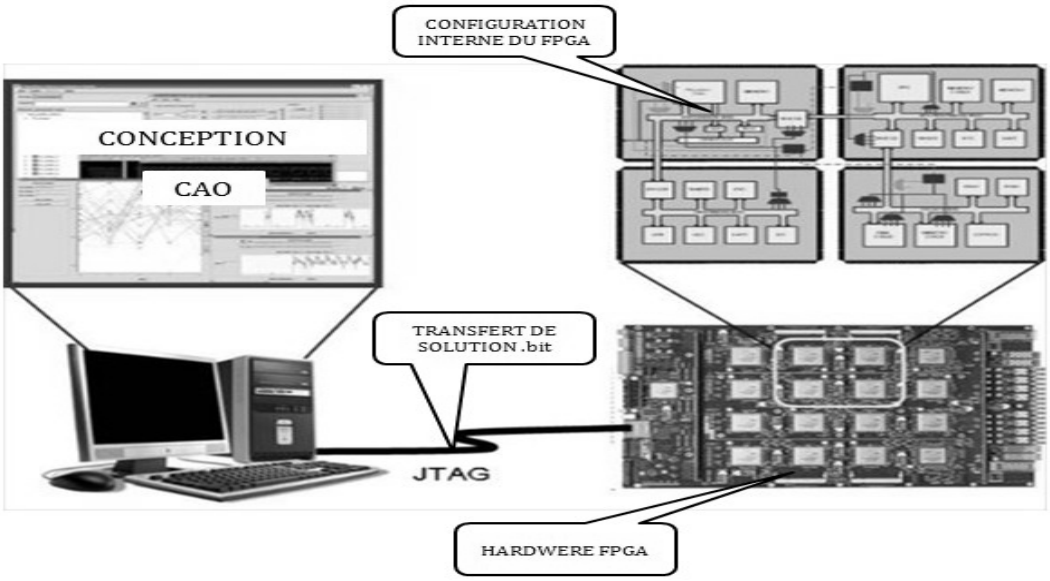


***Figure (IV.29)**:Configuration du FPGA par un câble JTAG.*

IV.7)-CONCLUSION

Ce dernier volet de cette étude nous permet de conclure que les résultats obtenus démontrent la justesse des modèles retenus sachons que cette méthode de conception permet de déceler les problèmes de vérification puis de réduire le facteur d'erreur humaine et les risques d'interruption du système. La conception et la simulation du circuit numérique menée par l'appui d'outils informatiques spécialisés et l'utilisation du langage *VHDL* comme outil de description pour représenter le comportement et l'architecture du dispositif numérique nous a permis d'obtenir un certain niveau de réutilisabilité des différents blocs de l'architecture mais toute la difficulté est de savoir ce qui est synthétisable ou non ainsi la difficulté de construire le « *Test-Bench* » afin de simuler la description.

CONCLUSION GENERALE ET PERSPECTIVES

CONCLUSION GENERALE

Nous constatons que la recherche dans ce domaine (la conception de commandes implémentables sur les circuits numériques à architecture reconfigurable *FPGA*)est complexe, car elle nécessite non seulement une maitrise des technologies relatives aux *FPGA* mais aussi une très bonne connaissance des applications et de leurs environnements.

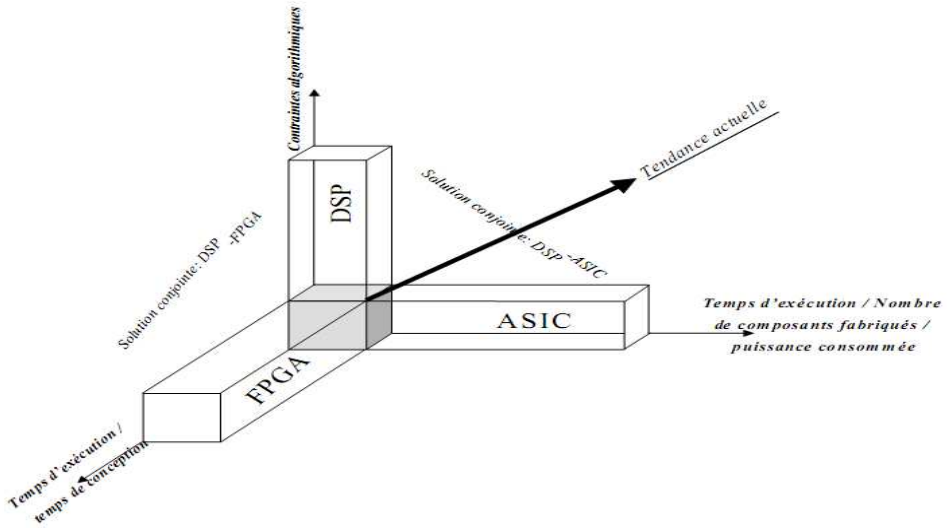
Dans le cadre du projet nous avons développés une méthodologie de conception d'une architecture câblée pour un système mécatronique. La démarche théorique suivie est récapitulée comme suis : « spécification du cahier de charge avec une étude théorique, conception architecturale détaillée, test, intégration et validation ». Durant le processus de conception, nous avons constaté que malgré les progrès réalisés dans le domaine des architectures reconfigurables, les outils de programmation non pas atteignent leurs maturité et que les algorithmes de programmation matérielle s'écartent significativement des algorithmes logiciels. Nous avons démontré dans ce mémoire la faisabilité d'implantation numérique câblée de la commande vectorielle. Sachons que le développement des systèmes électriques commandés reste encore très dépendant de la technologique d'implantation, nos résultats attestent que les *FPGA* constituent une alternative sérieuse aux *DSP* et aux *ASIC*. Danc, les *FPGA* est un moyen d'améliorer les performances de contrôle avec un gain économique et un autre du temps de développement.

Finalement, l'approche traitée au cours de ce travail peut-être avantageusement améliorée et facilement étendue sur d'autres types de machines et les chercheurs du domaine vont donc devoir relever des défis encore plus importants.

PERSPECTIVES

Dans le cadre des systèmes à hautes performances, ce travail présente une contribution à la conception et à l'implantation de commandes sur *FPGA*. Plusieurs travaux peuvent venir compléter le travail qui a été fait jusqu'ici et particulièrement de prendre en compte différents points qui n'ont pas étés abordés dans ce manuscrit. Malgré l'importance quantitative des travaux scientifiques, et malgré les immenses progrès réalisés dans les architectures numériques de nombreuses problématiques restent à explorer. A cet effet, nous envisagerons comme perspectives, quelques axes de recherche pour améliorer les performances de conception pour la commande des machines. Ces axes sont récapitulés comme suit :

- L'adaptation dynamique du design qui signifie par la reconfiguration dynamique des composants *FPGA* (Field Programmable Gate Array) en cas par exemple de changement des paramètres rotoriques de la machine avec développement de modules supplémentaires pour ces nouveaux paramètres.
- Une autre alternative d'amélioration réside dans la conception conjointe logicielle / matérielle (Co-design) qui est apparue comme une première réponse à ces problèmes.



Il reste à espérer que l'on découvre un moyen très efficace de combiner les avantages des deux approches de programmation. Dans ce cas, toute la difficulté consiste à répartir les tâches sur les différents processeurs et à définir les architectures respectives de ceux-ci.

- Des blocs supplémentaires peuvent toujours être ajoutées et même des contraintes ce qui permet d'effectuer séparément la mise à jour des différents modules.

- Les aspects énergétiques des opérateurs arithmétiques seront une bonne source de problématiques de recherche.
- L'optimisation du prototype aussi bien en termes de densité logique que de temps d'exécution.

Finalement, différents points n’ont pas étés abordés dans ce manuscrit et des travaux restent à effectuer sur ces architectures où le parallélisme nous semble être le maître pour les prochaines années et pour des composants toujours plus rapides.

BIBLIOGRAPHIE

REFERENCES BIBLIOGRAPHIQUES

THESES

[01]-**KHELDOUN AISSA :** *Thème « Amélioration des performances d'un variateur de vitesse par moteur asynchrone contrôlé par la méthode à flux orienté ». Thèse de doctorat université de Boumerdès 2007 sous la direction du professeur B. Chetate.*

[02]-**BAGHLI LOTFI :** *Thème « Contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques ».Thèse de doctorat. 1999, Nancy, France / Sous la direction du Pr. A. Rezzoug.*

[03]-**IMENE BEN AMEUR BAZINE:** *Thème « Identification en boucle fermée de la machine asynchrone : Application a la détection de défaut ».Thèse de doctorat 16 juin 2008 l'université de Poitiers directeurs de thèse : J.-C. Trigeassou et K. Jelassi.*

[04]-**JEREMIE DETREY** *: « Arithmétiques réelles sur FPGA virgule fixe, virgule flottante et système logarithmique ». Thèse doctorat École Normale Supérieure de Lyon15 janvier 2007 spécialité : Informatique.*

[05]-**NICOLAS VEYRAT-CHARVILLON :** *« Opérateurs arithmétiques matériels pour des applications spécifiques ». Docteur de l'École Normale Supérieure de Lyon spécialité : Informatique présentée et soutenue publiquement le 28 juin 2007 directeurs de thèse : Jean-Michel Muller et Arnaud Tisserand.*

[06]-**SEBASTIEN SNAIDERO :** *« Modélisation multidisciplinaire VHDL-AMS de systèmes complexes : vers le prototypage virtuel ».Thèse présentée afin d'obtenir le grade de docteur de l'université Louis Pasteur Strasbourg I discipline : électronique, électrotechnique, automatique spécialité : microélectronique soutenue publiquement à l'ENSPS le 03/12/2004 directeur de thèse : M.Yannick Hervé, MdC (HDR), ULP Strasbourg.*

[07]-**LIONEL LELONG :** *« Architecture SoC-FPGA pour la mesure temps réel par traitement d'image. Conception d'un système embarqué : imageur CMOS et circuit logique programmable ».Présentée devant l'université Jean Monnet de Saint-Etienne pour obtenir le grade de docteur spécialité : électronique soutenue le 7 décembre 2005 directeur de thèse : Gérard Jacquet Co-directeur : Guy Motyl.*

[08]-**DAVID GUIHAL :** *« Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires ». Thèse présentée au laboratoire d'analyse et d'architecture des systèmes du CNRS en vue de l'obtention du titre de docteur de l'université Toulouse III école doctorale : Génie électrique, électronique, télécommunications spécialité : conception de circuits microélectroniques et microsystèmes soutenue le 25 Mai 2007.*

[09]-**MOHAMED WISSEM NAOUAR :** ***«*** *Commande numérique à base de composant s FPGA d'une machine synchrone : Algorithmes de contrôle de courant ».Thèse de doctorat a l'école d'ingénieur de Tunis et l'université de Sergy Pontoise. Thèse présentée et soutenue a Tunis le 06 Décembre2007.Directeurs de thèse ERIC MONMASSON et ILHEM SLAMA BELKHODJA.*

[10]- **YOUSEF KEBBATI** *: « Développement d'une Méthodologie de Conception Matériel à Base de Modules Génériques VHDL/VHDL-AMS en Vue d'une Intégration de Systèmes de Commande Electriques » .Docteur de l'Université Louis Pasteur – Strasbourg I Discipline : Sciences pour l'Ingénieur (spécialité Microélectronique) Soutenue publiquement le 16 Décembre 2002. Directeur de thèse : M. Francis BRAUN, Professeur, Université Louis Pasteur, Strasbourg1.*

[11]- **M. SC. MARIUSZ MALINOWSKI:** *« Sensorless Control Strategies for Three - Phase PWM Rectifiers» Ph.D. Thesis Warsaw University of Technology Faculty of Electrical Engineering Institute of Control and Industrial Electronics Poland - 2001. Thesis supervisor Prof. Dr Sc. Marian P. Kaźmierkowski.*

[12]- ***M. KHEMAIES GHALI:*** *« Méthodologie de conception système a base de plateformes reconfigurables et programmables ». Docteur en sciences de l'université PARIS XI ORSAY Soutenue le 01 Mars 2005.*

ARTICLES

[01]- **PHILIPPE POURE, RAFIK KADRI, LUC HEBRARD, FRANCIS BRAUN :** *« Méthodologie d'intégration de commandes numériques pour dispositifs d'Électronique de Puissance basée sur l'utilisation du langage VHDL-AMS ». Laboratoire d'Électronique et de Physique des Systèmes Instrumentaux Université Louis Pasteur.*

[02]- **: J. R. MILLAN-ALMARAZ, R. J. ROMERO-TRONCOSO, L. M. CONTRERAS-MEDINA, A. GARCIA-PEREZ :** *« Embedded FPGA based induction motor monitoring system with speed drive fed using multiple wavelet analysis». Electronics Department – HSP Digital FIMEE - Universidad de Guanajuato Salamanca, Mexico.*

[03]- **TOLE SUTIKNO, Member, IACSIT & IEEE, MOCHAMMAD FACTA, Member, IEEE :** *« An Efficient Strategy to Generate High Resolution Three-Phase Pulse Width Modulation Signal Based on Field Programmable Gate Array».*

[04]- **TOSHIO TAKAHASHI:** *«New Digital Hardware Control Method for High Performance AC Servo Motor Drive – Accelerator TM Servo Drive Development Platform for Military Application». As presented at Military Electronics Conference, Sept 24-25, 2002.*

[05]- **C.BHARATIRAJA1, DR.S.JEEVANANDAM2, PRATIK3:** *«UG Scholar A System on Chip (SOC) - High-performance Power Drive Applications - SVPWM Based Voltage Source Inverter».1-Research Scholar, SRM University, Chennai. 2-Proffessor, Dept of Electrical Eng, Pondicherry University. 3- UG Scholar.*

[06]- **TANYA VLADIMIROVA and HANS TIGGELER**: *«FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm» Surrey Space Centre University of Surrey, Guildford, Surrey, GU2 5XH.*

[07]- **HONGZHI WANG, PIERRE LERAY, JACQUES PALICOT :** *« Architecture reconfigurable CBDRA basée sur l'opérateur CORDIC pour le traitement du signal: Applications aux récepteurs MIMO ». IETR/SUPELEC, Campus de Rennes Av. de la Boulais, CS 47601, 35576 CESSON-SEVIGNE, CEDEX, France.*

LIVRES ET OUVRAGES

[01]*J.WEBER et M.MEAUDRE : *"Circuits numériques et synthèse logique :un outil VHDL",Edition Masson collection technologie.*

[02]*-ETIENNE MESSERLI : *"Manuel VHDL synthèse et simulation», Version partielle septembre 2007.Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (heig-vd).*

[03]*-BARRET PHILIPE : *"Régimes transitoires des machines tournantes électriques", Edition Eyrolles, Paris, 1987.*

[04]*-HADDAD SALAH : *"Régimes transitoires des machines électriques", université Mouloud Mammeri de Tizi Ouzou.*

[05]*-CARLOS CANUDAS de WIT : *"Modélisation contrôle vectoriel et DTC Commande des moteurs asynchrone T1" Edition Hermès.*

[06]* -H. BUHLER : *"Convertisseurs Statiques", Ed. 1991.*

[07]*PETER J. ASHENDEN: *"The VHDL Cookbook"First Edition July, 1990 Dept. Computer Science University of Adelaide South Australia.*

SITES WEB

01-XILINX: *http://www.xilinx.com.*

02-ALTERA: *http://www.altera.com.*

03-MENTOR GRAPHICS : *http://www.mentor.com.*

ANNEXE

ANNEXE : LE LANGAGE DE DESCRIPTION MATERIEL VHDL ET LES SYSTEMES DE REPRESENTATION DES NOMBRES BINAIRES

Cette annexe présente une petite présentation du langage de description matérielle VHDL qui a été développé dans le but de faire la description (documentation), la vérification et la synthèse des circuits puis un bref résumé des systèmes de numération.

A.1)- HISTORIQUE DU LANGAGE DE DESCRIPTION MATERIELLE VHDL

Il existe actuellement différents langages qui permettent la description de circuits numériques. Alors il faut déterminer le langage de modélisation le mieux adapté à nos besoins. Le *VHDL* et le *VERILOG* sont les deux langages qui s'imposent comme standards mondiaux et restent le plus utilisés des langages dans le domaine de la conception des circuits électriques (Entreprises Européennes utilisent majoritairement *VHDL*, Entreprises Américaines utilisent majoritairement *VERILOG*). Le langage de description *VHDL* (*Very High Speed Integrated Circuit Hardware Description Language*) est un langage de description matérielle (comportement et/ou architecture) pour les systèmes numériques très populaire dans le domaine d'industrie de conception. Il est le fruit d'un projet de recherche mené par le groupement *IBM/Texas Instruments/Intermetrics* et conséquence d'un besoin croissant d'outils de conception de haut niveau pour décrire les systèmes numérique et les circuits intègres qui sont de plus en plus complexes. Ce langage est né dans les années *80* comme successeur du langage *ADA (1979)* au département de la défense américain *DOD* qui a lancé un appel d'offre pour créer un langage de description matérielle numérique standard. Ce langage est ouvert au domaine public en *1985* puis il est adopté par *IEEE* (*Institute of Electrical and Electronic Engineers*) comme standard et deviendra une norme en *1987* sous la dénomination *VHDL {IEEE 1076-1987}* puis complété et enrichi en *1993 {IEEE 1076-1993 et IEEE 1164-1993}* sous la dénomination *VHDL'93* avec les extensions *{IEEE 1076.3-1997, IEEE 1076.4-1995 }*puis vient d'importantes extensions qui touchent les signaux analogiques et mixtes en *1999{IEEE 1076.1-1999}* sous le nom *VHDL-AMS* (*1998 pour Verilog-AMS*) où *AMS* est une abréviation en anglais pour(*Analog and Mixed-Signal – AMS*). Le langage matériel *VHDL-AMS* est un standard de description et de modélisation des systèmes a temps continus et a temps discret exploitable pour des fins de simulation. Le *VHDL-AMS* est un langage particulièrement bien adapté à la simulation de problèmes multi-domaines(*multidisciplinaire*) et en *2001* la norme *{IEEE 1076-2001}*et finalement*{IEEE 1076.1-2006}*.

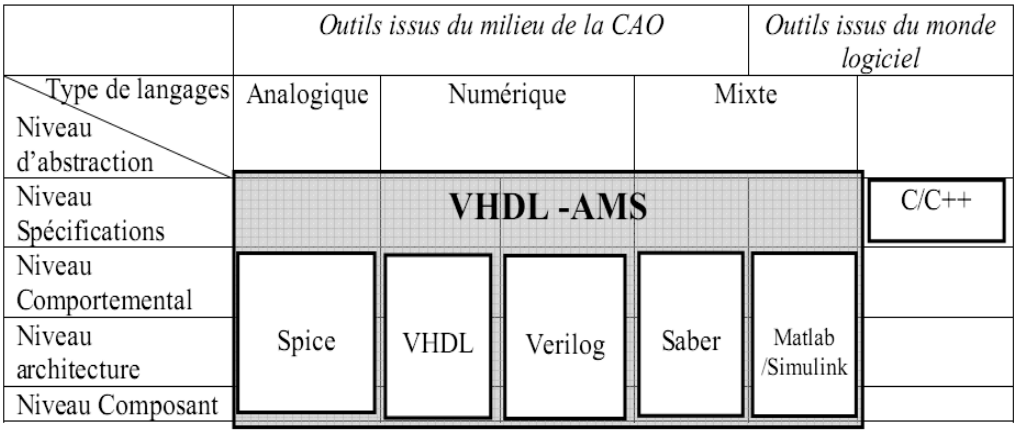
Le langage *VHDL* a été conçu pour être non seulement un langage de description matériel, mais aussi un langage de conception et de synthèse pour les systèmes numériques. Il

est devenu incontournable dans le domaine de la conception des circuits numériques avec une grande capacité de modéliser les circuits digitaux à différents niveaux d'abstraction. Ce langage est basé sur une simulation évènementielle, et non temporelle des systèmes.

A.2)- POSITION DU VHDL AU SEIN DES LANGAGES INFORMATIQUES

Les langages informatiques sont une abstraction qui facilite l'élaboration des algorithmes et un moyen de codé et d'introduire l'information sur un calculateur .Le calculateur s'occupe lui même de convertir ces langages en code machine. La diversité des langages se justifie par la diversité des domaines cibles.C'est ce que nous résumons comme suit :

- Les langages de programmation orientés objet tels *C++, VisualBasic ou Java* .
- Les langages de modélisation numérique tels *VHDL {IEEE 1076-2000}, Verilog {IEEE 1364-2001}, SystemC , SpecC et SystemVerilog.*
- Les langages mathématiques formels explicites *Matlab Simulink* .
- Les langages de modélisation implicites dédiés à l'électronique *SPICE.*
- Les langages de modélisation mixte multi-domaines *VHDL-AMS* avec *Verilog-AMS.*

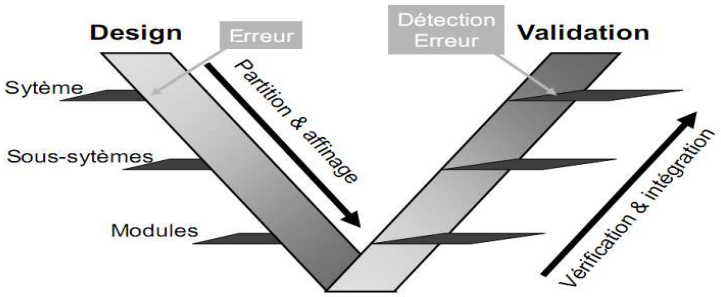


Tableau(A.01): Positionnement du langage *VHDL* au sein des autres langages scientifiques.

Une description ou un modèle en *VHDL* d'un circuit est une abstraction où une représentation du comportement de ce dernier où un fichier *VHDL* contient une seule entité et son architecture (*une ou plusieurs*) avec la déclaration des paquetages.

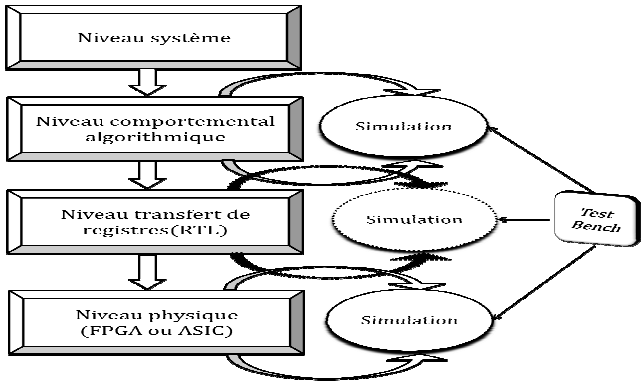
A.3)- ETAPES DE CONCEPTION A BASE DU VHDL

Cette méthodologie contient les principales règles à utiliser pour les différents traitements numériques complexes et la vérification de conformité avec le cahier des charges.



Figure(A.01): Le cycle de conception et de vérification traditionnel en V.

Les étapes fondamentales de conception matérielle à base du *VHDL* ainsi que les simulations tout au long du processus de conception sont illustrées dans le schéma suivant.



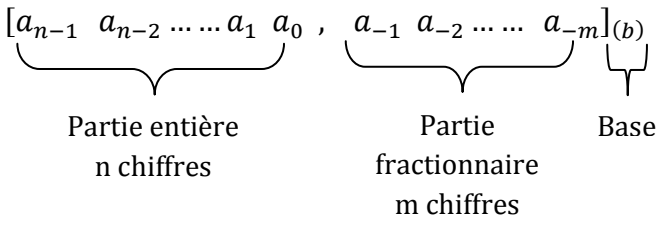
Figure(A.02): Les étapes fondamentales de conception matérielle.

La synthèse est l'étape qui transforme la description *HDL* en portes logiques et qui respecte les contraintes imposer par l'utilisateur (*de superficie, de temps*). Malheureusement, la synthèse de circuits ne concerne qu'un ensemble limité de descriptions *VHDL* qui n'est pas bien déterminée. Certaines structures non synthétisables sont prévisibles comme l'instruction « *After* » mais souvent il est difficile de distingué le synthétisable du non synthétisable. Même la manière dont les instructions sont utilisées et d'un synthétiseur à l'autre peut rendre la description non synthétisable.

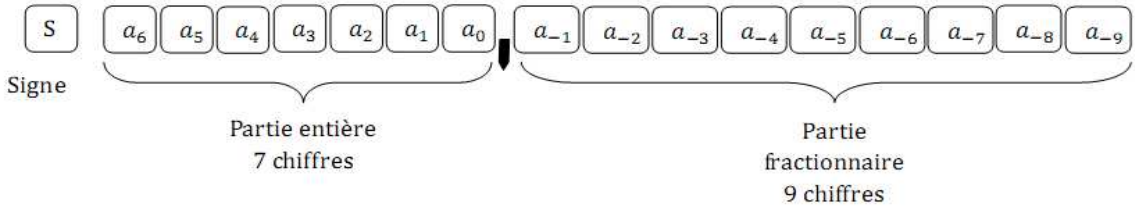
A.4)-SYTEMES DE REPRESENTATION DES NOMBRES

Le nombre de bits utilisés pour effectuer la quantification de l'échantillonnage est un paramètre très important pour l'obtention d'une bonne précision .Son influence est directe sur la simplicité ou la complexité des operateurs arithmétiques.

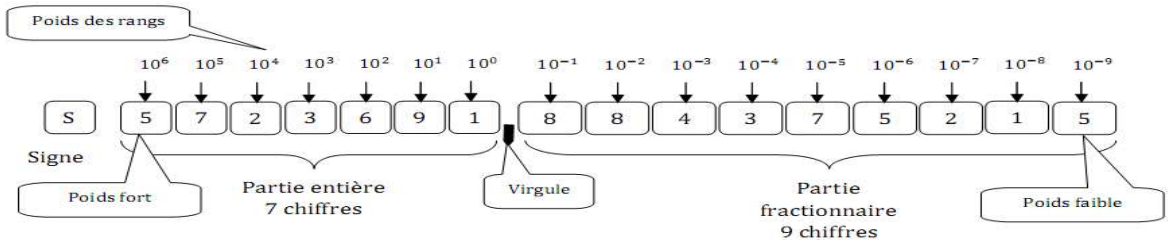
Alor l'utilisation d'un nombre arbitraire de bit pour le codage peut engendrer de lourdes conséquences sur les caractéristiques du circuit. Donc le codage et la quantification des grandeurs peuvent engendrer d'importantes répercussions sur la complexité et la durée des calculs.



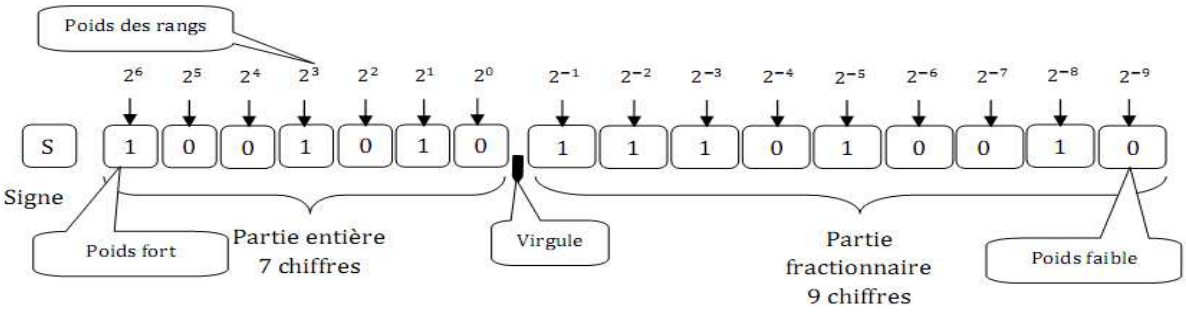
***Figure(A.03)**:* Représentation d'un nombre à base b.



***Figure(A.04)**:* Représentation d'un nombre signé à 16 chiffres.



***Figure(A.05)**:* Exemple de représentation d'un nombre signé à 16 chiffres et sur la base décimale 10.



***Figure(A.06)**:* Exemple de représentation d'un nombre signé à 16 chiffres et sur la base binaire 2.

ANNEXE : LE LANGAGE DE DESCRIPTION MATERIEL VHDL ET LES SYSTEMES DE REPRESENTATION DES NOMBRES BINAIRES

Chaque opération arithmétique est fortement liée au système de représentation des nombres. Dans ce qui suit nous décrivant les trois systèmes de représentation des nombres réels les plus répondus sans approfondir en détails.

A.4.1)-REPRESENTATION BINAIRE EN VIRGULE FIXE

Le système de représentation des nombres réels le plus attractif est la représentation en virgule fixe. Cette représentation est très intéressante du fait de sa simplicité.

- Principe de la représentation :

Si nous prenons un nombre X représenté en virgule fixe, nous constatons qu'il est constitué de deux parties qui ont un nombre de bits fixe. Ces parties sont la partie entière notée I_X et la partie fractionnaire F_X avec un nombre de bit W_I et W_F respectivement.

$$X = I_X + F_X * 2^{-\omega_F} = \overline{I_X F_X} * 2^{-\omega_F}$$

On peut ainsi noter la position de la virgule binaire dans l'écriture du nombre X de la manière suivante : $X = \overline{I_X\ ,\ F_X}$

Enfin, on indexera par leur poids les bits d'un nombre en virgule fixe X :

$$X = \overline{X_{\omega_I - 1} \dots X_1 X_0\ , X_{-1} \dots X_{-\omega_F}} = \sum_{i=-\omega_F}^{\omega_I - 1} X_i\, 2^i$$

Les valeurs négatives seront encodées en complément à deux.

A.4.2)-REPRESENTATION BINAIRE EN VIRGULE FLOTTANTE

En calcul scientifique on a souvent besoin de manipuler des nombres très grands ou très petits. Pour cela on utilise la représentation en virgule flottante. L'intégration sur *FPGA* d'operateurs avec ce système de représentation est possible car la capacité d'intégration des *FPGA* de nos jours a atteint un niveau d'intégration suffisant technologiquement.

- Principe de la représentation :

Si nous prenons un nombre X représenté en virgule flottante, nous constatons qu'il est constitué de deux parties qui sont : la première partie appelée « *mantisse* » et la deuxième partie est le facteur exponentiel souvent appelée « l'exposant ». Nous aboutirons à la représentation suivante pour un nombre X :

$$X = M_X * \beta^{Ex} = (Nombre\ X = Mantisse\ M_X * \beta^{Exposant\ Ex})$$

$$Ou\ \beta = Base\ et\ 2\ en\ binaire$$

Pour pouvoir représenter les nombres positifs comme négatifs, nous ajoutons à la représentation un bit de signe Sx, ce qui donne ainsi :$X = (-1)^{Sx} * \overline{1, F_X} * 2^{Ex}$.

A.4.3)-REPRESENTATION BINAIRE EN SYSTEME LOGARITHMIQUE

Le système logarithmique est introduit par Swartzlander comme une alternative à la virgule flottante.

- Principe de la représentation :

Si nous prenons un nombre X représenté en système logarithmique, nous constatons qu'il ressemble a celui que nous avons vu en virgule fixe avec la seul différence qui réside dans la représentation des nombres par leurs signes et de représenter le logarithme de leurs valeurs absolues en virgule fixe et non la valeur du nombre. Ainsi, la représentation d'un nombre X est : $X = (-1)^{Sx} * \beta^{Lx}$.

REMARQUE : D'après une recherche bibliographique sur l'implantation des différents types d'applications en technologie *VLSI*, on remarque que l'arithmétique à virgule fixe est fortement utilisés dans les systèmes embarqués car elle est beaucoup plus facile à implanter que la représentation en virgule flottante sauf si le cahier de charge exige des contraintes qu'on ne peut pas satisfaire que par d'autre systèmes de représentation.