الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR.YAHIA FARES DE

MEDEAFACULTE DE LA TECHNOLOGIE
Département du Génie Électrique



Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme de
Master en Génie Électrique

Option: *automatique des systèmes*

THEME

Commande d'un moteur a courant continu en utilisant un circuit FPGA

Proposé par

Dr. KABACHE NADIR

Préparé et présenté
Par
LEBATT AHMED

NADJEM MOUSSA NADJEM

2016 /2017

Dédicaces

Ahmed

Je dédie ce travail a :

Mes parents Nul mot ne peut exprimer ma gratitude envers eux, je l'ai dit une fois dans le temps, je la répéterai toujours avec la même conviction.

A mon Frère pour son énorme soutien et ses encouragements

A tous les membres de ma famille pour leur soutien et leur affection.

A tous mes amis en honneur à la grande amitié qui nous unit, aux souvenirs et moments agréables passés ensemble.

À tous je vous dis que ce modeste travail soit une guise de reconnaissance pour votre estime et votre soutien.

Nadjem

Je dédie ce travail a :

Mes parents Nul mot ne peut exprimer ma gratitude envers eux, je l'ai dit une fois dans le temps, je la répéterai toujours avec la même conviction.

A mes Frères et ma sœur pour son énorme soutien et ses encouragements

A tous les membres de ma famille pour leur soutien et leur affection.

A tous mes amis en honneur à la grande amitié qui nous unit, aux souvenirs et moments agréables passés ensemble.

Remerciements

Nous remerciant ALLAH qui nous aide et nous donne la patience et le courage

Durant ces langues années d'étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personne qui

Nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire

Ainsi qu'à la réussite de cette formidable année universitaire.

Nous tenons à remercier sincèrement Mr KABACHE NADIR, en tant que Encadreur, qui à toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.

Merci à tous

Sommaire

Introduction générale ……..…….....……... 1

Chapitre 1 : Modélisation et commande d'un MCC

1.1 Introduction..... 2

1.2 Constitution..... 2

1.2.1 L'inducteur..... 3

1.2.2 Le rotor (l'induit)..... 3

1.2.3 Le collecteur et les balais..... 3

1.3. Principe de fonctionnement..... 4

1.4. Les différents types de moteurs à courant continu..... 6

1.4.1. Les moteurs à inducteur à aimant permanent..... 6

1.4.2. Les moteurs à inducteur bobiné..... 6

1.5. Le modèle du moteur à excitation séparée..... 7

I.6. La commande en vitesse d'un MCC à excitation séparée..... 9

1.7.Résulta de simulation..... 12

1.7.1. Résulta de simulation d'une commande avec hacheur idéale..... 12

I.7.2. Résulta simulation de commande avec convertisseur (hacheur)……..... 14

1.8 .Conclusion..... 15

Chapitre 2 : circuits FPGA

2.1 Introduction ……..... 16

2.2 Les circuits logiques programmables ……..... 16

2.3 Les FPGA……..... 17

2.3.1 L'architecture des circuits FPGA..... 18

2.3.2 Technologie de programmation des circuits FPGA 19

2.3.3Domaines d'applications d'FPGA..... 20

2.3.4 Avantages et inconvénients de FPGA..... 21

2.4.1 Historique de VHDL..... 21

2.4.2 Structure d'une description VHDL 22

2.4.2.1 Création du code VHDL d'un développement..... 23

2.4.2.2 Synthèse..... 23

2.4.2.3 Simulation..... 25

2.4.2.4 .Optimisation, placement et routage..... 25

2.4.3 Différences entre un langage de programmation ordinaire et VHDL..... 26

2.5.1 Adaptation et initialisation des signaux..... 27

2.5.2 Description VHDL de l'algorithme de commande du moteur..... 28

2.5.3 Simulation de l'algorithme de commande..... 29

2.6.Résultats de simulation Hardware …….....…….....…….. 30

2.7.Conclusion…….... 31

Chapitre 3 : Implémentation et résultats

3.1 Introduction..... 32

3.2 Présentation de l'outil de développement ISE XILINX 32

3.3 Présentation de L'analyseur Chip Scope..... 33

3.4 Commande MLI d'un hacheur..... 33

3.4.1 Génération d'un signal carré..... 34

3.4.2 Génération d'un signal MLI par signal triangulaire..... 34

3.4.3 Résultat de simulation software..... 36

3.4.3 La simulation hardware de la technique de command MLI …….....…..... 37

3.5 Conclusion…….....… 46

Conclusion générale …….……...….. 47

Références bibliographiques …….....…….....……... 48

Liste des figures

Figure 1.1: Description de la machine à courant continu 02

Figure 1.2 : L'inducteur de la machine à courant continu 03

Figure 1.3 : Le rotor de la machine à courant continu 03

Figure 1.4 : Le dispositif collecteur / balais 04

Figure 1.5: Principe de fonctionnement d'un moteur à courant continu 05

Figure 1.6: Moteur à aimant permanent 06

Figure 1.7 : Modèle électrique du moteur à excitation indépendante 07

Figure 1.8 Caractéristiques d'un moteur à excitation séparée 08

Figure 1.9: Le couple en fonction de courant d'alimentation 08

Figure1.10 Schéma-bloc d'une machine à courant continu à excitation indépendante 09

Figure. 1.11: Schéma du la commande MLI 10

Figure 1.12: Structure du régulateur PI utilisé 11

Figure. 1.13: Schéma principale de commande MCC 11

Figure :1.14: Caractéristique de la vitesse en charge 12

Figure 1.15: Caractéristique du courant en charge 13

Figure 1**.16 :** Caractéristique du couple en charge 13

Figure 1**.17 :** Caractéristique de vitesse en charge **14**

Figure 1**.18 :** Caractéristique de courant en charge 14

Figure1.19 : Caractéristique de couple en charge 15

Figure2.1 : Diagramme des différents types de circuits logiques programmable.16

Figure 2. 2 : Structure d'un circuit FPGA 18

Figure 2.3 : Structure d'une cellule SRAM 19

Figure 2.4 : Organisation fonctionnelle de développement d'un projet sur circuit FPGA 22

Figure 2.5 : *Vue d'ensemble du logiciel « Xilinx Project Navigator*» 23

Figure 2.6 : Aperçu de l'outil « View RTL Schematic » 23

Figure 2.7 : Aperçu de l'outil d'affectation des broches d'entrées/sorties 24
Figure 2.8: Aperçu de l'outil « FPGA Editor » 25
Figure 2.9 : Exemple de choix de nombre de bits pour les signaux utilisés 27
Figure 2.10: Un partie de la description VHDL de l'algorithme de commande 28
Figeur 2.11 : Schema technologique de l'algorithme de commande élaboré 29
Figure 2.12 : Comparaison entre résultat en régime permanant 30
Figure 2.13 : Comparaison entre résultat en régime transitoire 31
Figure 3.1 : analyseur Chip Scope 33
Figure 3.2 : Description VHDL pour la génération des signaux MLI 35
Figure 3.3: Description VHDL pour la variation du rapport cyclique 36
Figure 3.4 : simulation des techniques de commande 37
Figure 3.5 : Banc d'expérimentation utilisé 38
Figure 3.6 : la tentions de sortie avec un rapport cyclique de 20 % 39
Figure 3.7 : la tentions de sortie avec un rapport cyclique de 50 % 39
Figure 3.8 : la tentions de sortie avec un rapport cyclique de 90 % 40
Figure 3.9 : la tentions de sortie avec un rapport cyclique de 100 % 41
Figure 3.10 : vitesse correspondante a 50% du rapport cyclique 42
Figure 3.11: vitesse correspondante a 75% du rapport cyclique 43
Figure 3.12: vitesse correspondante a 100% du rapport cyclique 44
Figure 3.13 : Banc de montage 45

Nomenclature

MCC : machine a courant continu

FPGA: Field Programmable Gate Array

VHDL:Vhsic (Verry Hight Speed Integrated Circuit)Hardware Description Language

ASIC: application specific integrated circuit

CAO: conception assistée par ordinateur

CLB: configurable logique bloc

CPLD: complexe programmable logique devise

CORDI :coordinaite rotation digitale computer

DSP :digital signal processeur

EPROM :erasable programmable read only memory

EEPROM: electrically erasable programmable read only memory

IOB: input output bloc

MLI: modélisation de largeurs d'impulsion

PROM: programmable read only memory

PI : proportional integral

PAL: programmable array logic

PLD: programmable logic device

ROM: read only memory

SRAM: static random access memory

Chapitre I : modélisation et commande d'un MCC

Introduction générale

Ce mémoire a été réalisé dans le cadre de la formation Master II, option automatique des systèmes, au sein du laboratoire d'électrotechnique de Médéa. L'objectif de ce projet est la commande d'un moteur à courant continu via une carte FPGA.

Les moteurs et les générateurs à courant continu furent les premières machines électriques utilisées par les ingénieurs au milieu du 19^{ième}siècle pour produire de la puissance motrice particulièrement dans le transport ferroviaire (tramways, TGV), et maintenant restent très utilisés dans le domaine de l'automobile (ventilateurs, lève-vitre, etc.)

Les FPGAs, Field Programmable Gate Array ou réseau de portes programmables.

Ce sont des circuits intégrés programmables (ou plutôt reconfigurables) plusieurs fois et de plus en plus dynamiquement! Ils permettent d'émuler un circuit afin de le valider avant la gravure silicium.

Ce mémoire présente notre projet qui consiste à commander un moteur à courant continu par un signal modulé en largeur d'impulsion (MLI). La commande a été programmée en langage VHDL en utilisant une carte FPGA.

Ce mémoire est divisé en trois chapitres, organisés comme suit:

- Dans le premier chapitre nous avons présentés une étude théorique du moteur à courant continu et aussi des convertisseurs continu/continu ou bien les Hacheurs.
- Dans le deuxième chapitre nous avons présenté des notions générales sur les FPGAs, son langage de programmation VHDL.
- Dans le troisième chapitre, nous avons présenté notre application concernant la commande d'un moteur à courant continu via une carte FPGA. Une description détaillée des parties matérielles et logicielles de l'application a été donnée, ainsi qu'aux différents résultats pratiques obtenus par les deux cartes spartane 3E et DSPace. Ces résultats concernent les deux solutions (Simulink et VHDL-Sysgen) pour générer le signal PWM.
- Enfin, dans la dernière partie, nous terminons par une conclusion générale sur cette étude ainsi que les perspectives envisageables pour un travail futur

1.1 Introduction :

Les moteurs à courant continu sont très utilisés dans les systèmes automatiques qui nécessitent une variation précise de la vitesse de rotation. Les machines à courant continu sont des convertisseurs électromécaniques d'énergie. Soit, ils convertissent l'énergie électrique absorbée en énergie mécanique lorsqu'ils sont capables de fournir une puissance mécanique suffisante pour démarrer puis entraîner une charge en mouvement. On dit alors qu'ils ont un fonctionnement en moteur. Soit, ils convertissent l'énergie mécanique reçue en énergie électrique lorsqu'ils subissent l'action d'une charge entraînante. On dit alors qu'ils ont un fonctionnement en générateur.

Dans ce chapitre, on présente la constitution de la machine à courant continu, son principe de fonctionnement ainsi que son schéma de commande.

1.2 Constitution [9]:

La machine à courant continu est constituée de trois parties principales (**Fig.1.1**):

- L'inducteur.
- L'induit.
- le dispositif collecteur / balais.

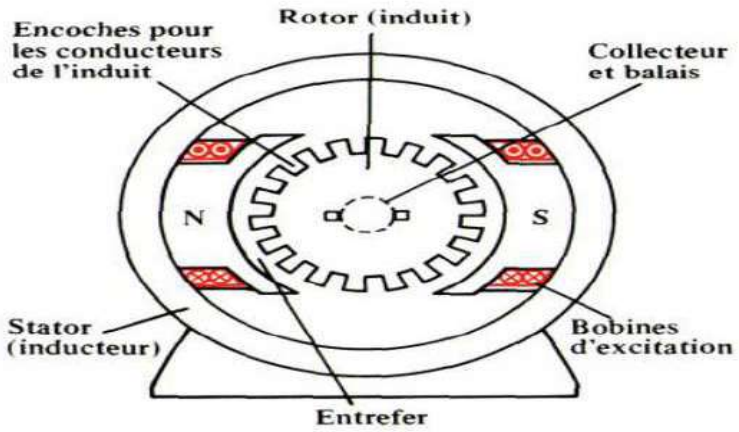


Figure 1.1: Description de la machine à courant continu

1.2.1 L'inducteur :

L'inducteur est la partie fixe du moteur. Il est constitué d'un aimant permanent ou d'un électroaimant alimenté par un courant continu d'excitation (**figure 1.2**).



Figure 1.2 : L'inducteur de la machine à courant continu

1.2.2 Le rotor (l'induit):

Le rotor est constitué d'encoches dans lesquelles est enroulé un bobinage de (N)Conducteurs alimentés en courant continu (I) via le collecteurfigure(I.3).

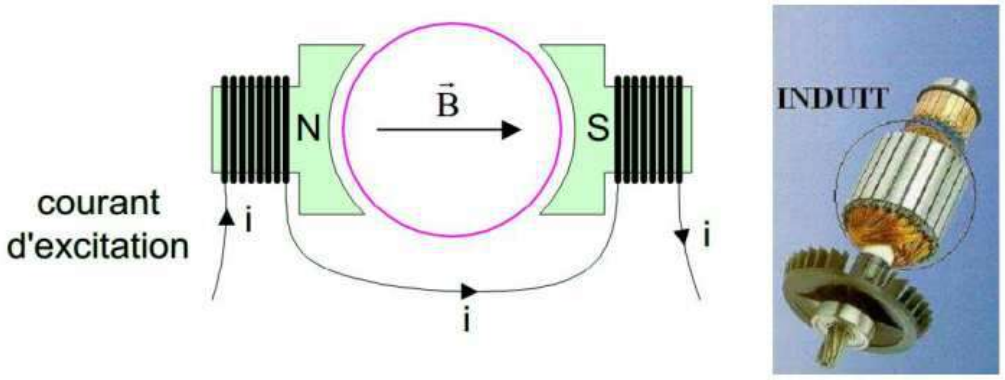


Figure 1.3 : Le rotor de la machine à courant continu

1.2.3 Le collecteur et les balais : [2]

Le collecteur est un ensemble de lames de cuivre où sont reliées les extrémités du bobinage de l'induit . Les balais (ou charbons) sont situés au stator et frottent sur le collecteur en rotation comme l'indique la figure (**figure 1.4**)



Figure 1.4 : Le dispositif collecteur / balais

1.3. Principe de fonctionnement [9] :

Le fonctionnement du moteur à courant continu est basé sur le principe des forces de Laplace.Un conducteur de longueur (L), placé dans un champ magnétique etparcouru par un courant, est soumis à une force électromagnétique. Le champ crée par l'inducteur agit sur les conducteurs de l'induit. Chacun des (N) conducteurs de longueurs (L) placé dans le champ (B) et parcouru par un courant (I) est le siège d'une force électromagnétique perpendiculaire au conducteur (Fig. 1.5). La force électromagnétique est donnée par :

$$F = B. I. L. \sin \alpha \qquad (1.1)$$

Ces forces de Laplace exercent un couple proportionnel à l'intensité (I) et au flux (Φ) sur le rotor. Le moteur se met à tourner à une vitesse proportionnelle à la tension d'alimentation (V) et, inversement proportionnelle au flux (Φ).Au passage de tout conducteur de l'induit sur la ligne neutre, le courant qui le traverse change de sens grâce au collecteur. Le moteur conserve le même sens de rotation. Pour inverser le sens de rotation du moteur, il convient d'inverser le sens du champ produit par l'inducteur par rapport au sens du courant circulant dans l'induit :

- Soit on inverse la polarité de la tension d'alimentation de l'induit.
- Soit on inverse la polarité d'alimentation du circuit d'excitation.

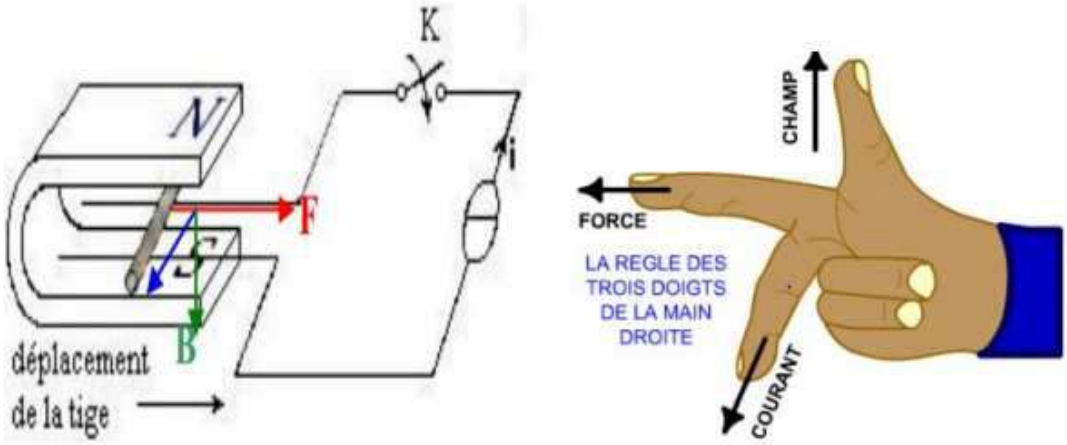


Figure 1.4: Principe de fonctionnement d'un moteur à courant continu

D'une autre part, la force électromotrice (E), qui est la tension produite par le rotor (l'induit) lors de sa rotation dans le flux magnétique produit par la partie fixe (l'inducteur), dépend des éléments de construction de la machine. Elle se donne par :

$$\boldsymbol{E = \frac{P}{a} N \frac{n}{60} \Phi} \qquad \textbf{(}1.2\textbf{)}$$

Avec :

P : nombre de pairs de pôles de la machine.

N : nombre de conducteurs actifs de la périphérie de l'induit.

a : nombre de paires de voies de l'enroulement entre les deux balais.

n : fréquence de rotation de l'induit (en t/s).

Φ: flux sous un pôle de la machine en Webers.

Finalement :

$$\mathbf{E = K.\ \Omega.\ \Phi} \qquad \textbf{(1.3)}$$

Avec :

$$\mathbf{K=\frac{P}{2\pi a} N} \qquad (1.4)$$

1.4. Les différents types de moteurs à courant continu [3] :

On distingue deux types de moteurs à courant continu :

- Les moteurs à inducteur à aimant permanent
- Les moteurs à inducteur bobiné

1.4.1. Les moteurs à inducteur à aimant permanent :

Il n'y a pas de circuit inducteur, le flux inducteur est produit par un aimant permanent (Fig. 1.6). Tous les moteurs à courant continu de faible puissance et les micromoteurs sont des moteurs à aimant permanent. Ils représentent maintenant la majorité des moteurs à courant continu due à leur utilisation très simple.

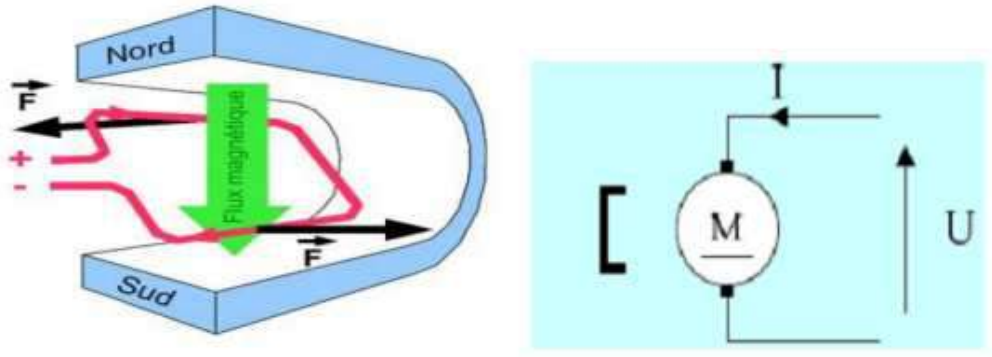


Figure 1.6: Moteur à aimant permanent

1.4.2. Les moteurs à inducteur bobiné :

Il existe 4 types différents de moteurs électriques qui sont classés en fonction du type d'excitation qui est employé, qui sont :

- ❖ Le moteur à excitation séparée.
- ❖ Le moteur à excitation shunt.
- ❖ Le moteur à excitation série.
- ❖ Le moteur à excitation composée.

Dans la suite de ce travail, nous intéresserons aux moteurs à excitation séparée.

1.5. Le modèle du moteur à excitation séparée :

Dans un moteur à excitation séparée ou indépendante, le circuit d'excitation est séparé du circuit d'induit. Si l'inducteur est un aimant permanent, le flux (Φ) est constant. Si l'inducteur est un électroaimant alimenté par une source de tension continue réglable, le flux (Φ) ne dépend que du courant dans l'inducteur appelé courant d'excitation (Ie).Le courant crée un champ qui, de sa part, crée à travers une spire un flux. Si la tension (Ve) est constante, le courant d'excitation (Ie) est constant et le flux (Φ) est constant. Dans ces conditions, la force contre électromotrice (E) ne dépend que de la fréquence (n) de rotation. Donc la fréquence de rotation (n) est proportionnelle à la tension (V) d'alimentation du moteur (Fig. 1.7).

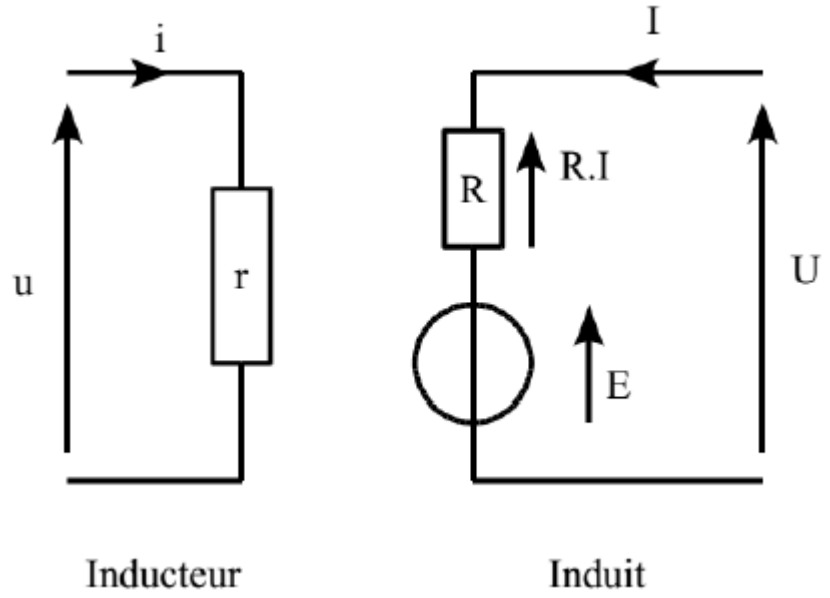


Figure 1.7 : Modèle électrique du moteur à excitation indépendante

Compte tenu de la conservation de la puissance d'une machine (en négligeant les pertes), la puissance électrique absorbée est :

Pa= UI **(**1.5**)**

La puissance mécanique fournie est :

$$\mathbf{P_u = T_u .\Omega = T_u .2\pi n} \qquad (1.6)$$

Si (V) est proportionnelle à (n) cela signifie que ($\mathbf{T_u}$) est proportionnel à (I).Donc le couple utile ($\mathbf{T_u}$) est proportionnel à l'intensité du courant d'alimentation du moteur.

Pour un moteur à excitation séparée, le flux (Φ) est constant donc la fréquence de rotation (n) est proportionnelle à la tension d'alimentation V (Fig. 1.8) :

$$\mathbf{n=\frac{E}{K\Phi}=\frac{V-RI}{K\Phi}=k.\ V} \qquad (1.7)$$

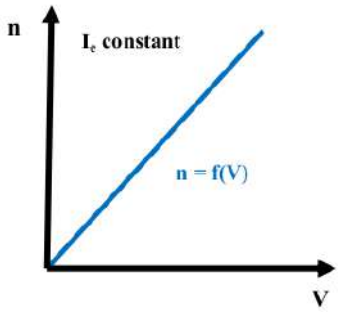


Figure 1.8 : Caractéristiques d'un moteur à excitation séparée

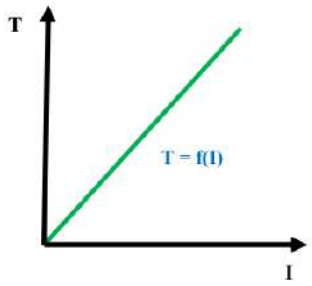


Figure 1.9: Le couple en fonction de courant d'alimentation

Et le couple (Tu) est proportionnel à l'intensité du courant d'alimentation (Fig. 1.9):

$$\mathbf{T=\frac{EI}{\Omega}=K\Phi I=k'v} \qquad (1.9)$$

En considérant le modèle électrique du moteur (Fig. 1.7), les équations caractérisant le comportement du moteur sont données comme suivant.

- Équation de l'inducteur :

$$\mathbf{V_e=r.i+Le\frac{di}{dt}} \qquad \mathbf{(}1.10\mathbf{)}$$

- Équation de l'induit :

$$\mathbf{V_a=R_a.I_a+L_a\frac{dIa}{dt}+E=R_a.I_a+L_a\frac{dIa}{dt}+k\,\Omega\Phi} \qquad (1.11)$$

Avec Ω vitesse de rotation en radians/seconde.

La figure (1.10) représente le modèle du moteur établit en utilisant l'interface Simulink du Matlab.

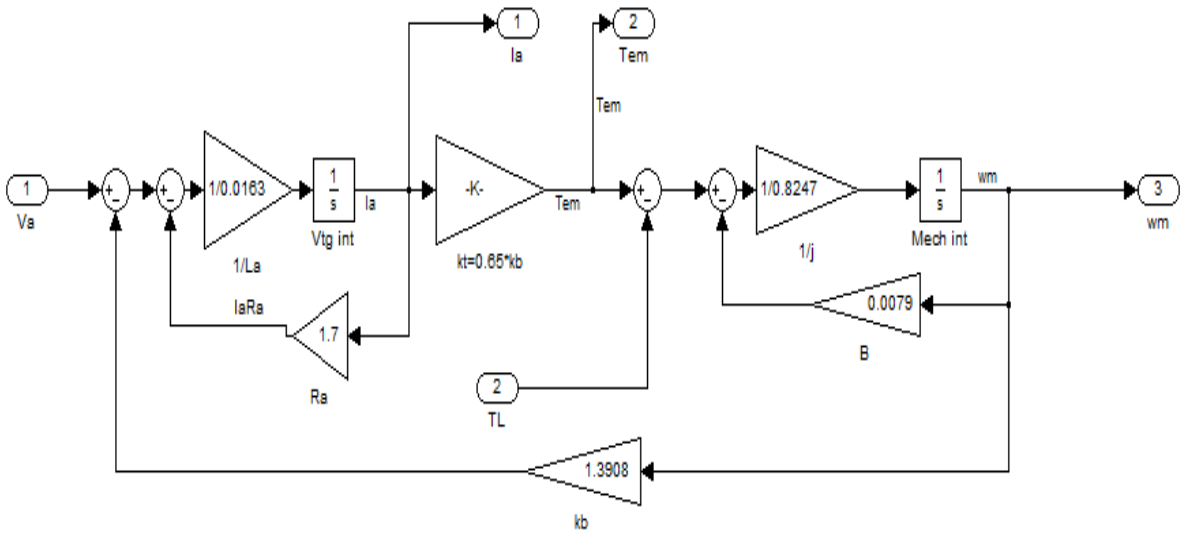


Figure 1.10. Schéma-bloc d'une machine à courant continu à excitation indépendante

I.6. La commande en vitesse d'un MCC à excitation séparée :

Pour faire varier la vitesse d'un moteur à courant continu, on peut agir sur la tension aux bornes de l'induit. La tension d'induit est directement proportionnelle à la vitesse de rotation. La puissance varie mais le couple reste constant. On dit alors que l'on fait de la variation de vitesse à couple constant. Par conséquent, il est possible d'utiliser la tension moyenne aux bornes de l'induit pour commander la vitesse du moteur.

Pour atteindre cet objectif, le hacheur est le convertisseur statique adéquat pour cette commande. Les convertisseurs statiques sont utilisés dans des domaines très variés de la conversion d'énergie électrique. L'évolution importante des semi-conducteurs de puissance et la variété des techniques de conception des circuits de commande et de réglage apportent des solutions très avantageuses concernant l'encombrement, la fiabilité, le rendement et l'entretien des convertisseurs.

Comme indiquée ci-dessus, pour faire varier la vitesse d'un moteur on peut faire varier la tension d'alimentation à ses bornes en utilisant. Dans ce cas, le hacheur agit sur la valeur moyenne de la tension d'alimentation du moteur.

Le principe du hacheur consiste à établir puis interrompre périodiquement la liaison source-charge à l'aide d'un interrupteur électronique. L'un des paramètres caractéristiques du hacheur est le rapport cyclique. Il détermine le rapport du temps pendant lequel l'interrupteur est fermé par rapport à une période de fonctionnement du montage (T).

$$\alpha = \mathbf{t_f/T} \qquad (1.18)$$

La valeur de rapport cyclique : $0 \leq \alpha \leq 1$

La qualité de la tension de sortie d'un hacheur dépend largement de la technique de commande utilisée pour commander les interrupteurs de ce hacheur. Il existe plusieurs techniques de commande et le choix d'une technique parmi toutes les possibilités dépend essentiellement du type d'application auquel l'appareil est désigné. Parmi ces techniques, on distingue la technique de la Modulation de la Largeur d'Impulsion (MLI).

Les impulsions de commutation sont générées en comparant le signal prévenant du circuit de commande qui représente la tension de référence et un signal triangulaire d'une fréquence appropriée choisie pour avoir le rapport cyclique voulu (Fig. 1.11).

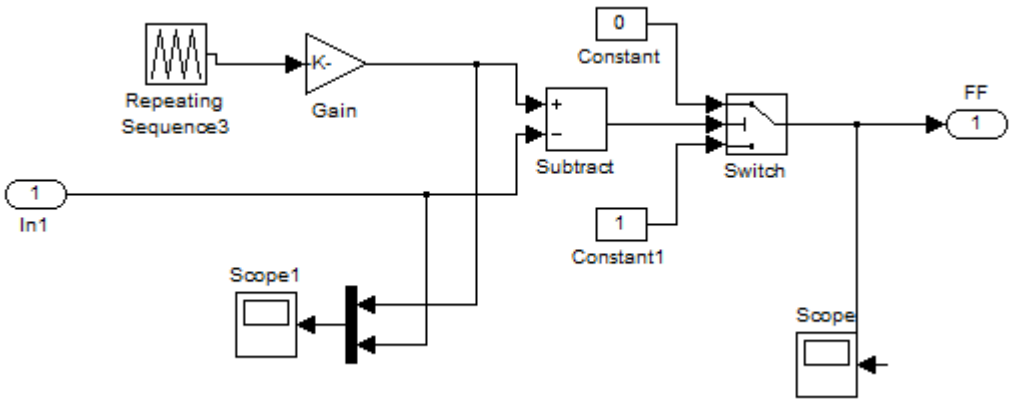


Figure. 1.11 : schéma du la commande MLI

pour le réglage de la vitesse, un régulateur PI a été utilisé dans la structure et les paramètres sont sélectionnés en se basant sur les équation de modèle du moteur établis précédemment. La figure (1.12) donne la structure du régulateur PI utilisé.

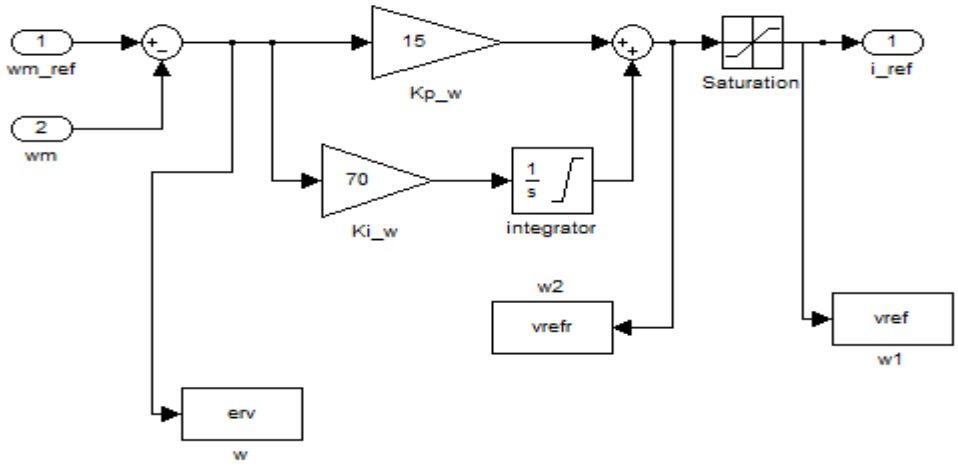


Figure 1.12 : structure du régulateur PI utilisé

Le schéma bloc de la commande du moteur, dans l'environnement Simulink est donné sur la figure (1.13).

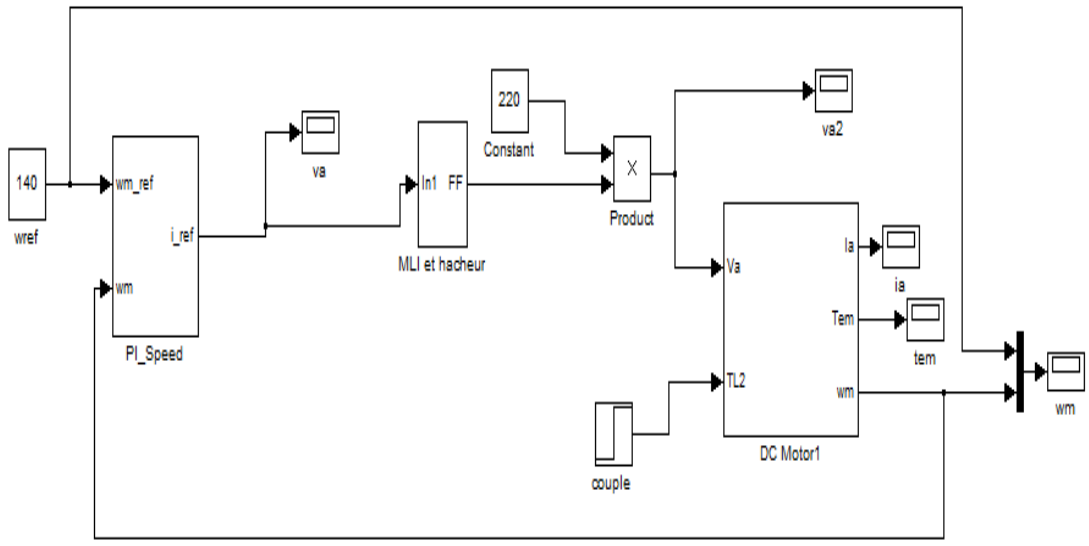


Figure. 1.13 : schéma principale de commande MCC

1.7.Résulta de simulation:

Le diagramme de commande établit a été utilisé pour simuler le comportement du moteur pour différentes condition de fonctionnement. La simulation a été effectuée avec un moteur à courant continu avec les paramètres suivant : La=0.0163, Ra=1.7, $K_{f=}$ 0.0079, J=0.8247, Ki=1.3908,T_{em}=k.I_a=0.65*Ki

1.7.1. Résulta de simulation d'une commande avec hacheur idéale

On remarque que la vitesse prend un temps pour se stabilise. Si on applique une charge en régime stable on constate une baisse de vitesse et se stabilise sous l'effet de la commande (Fig. 1.14).

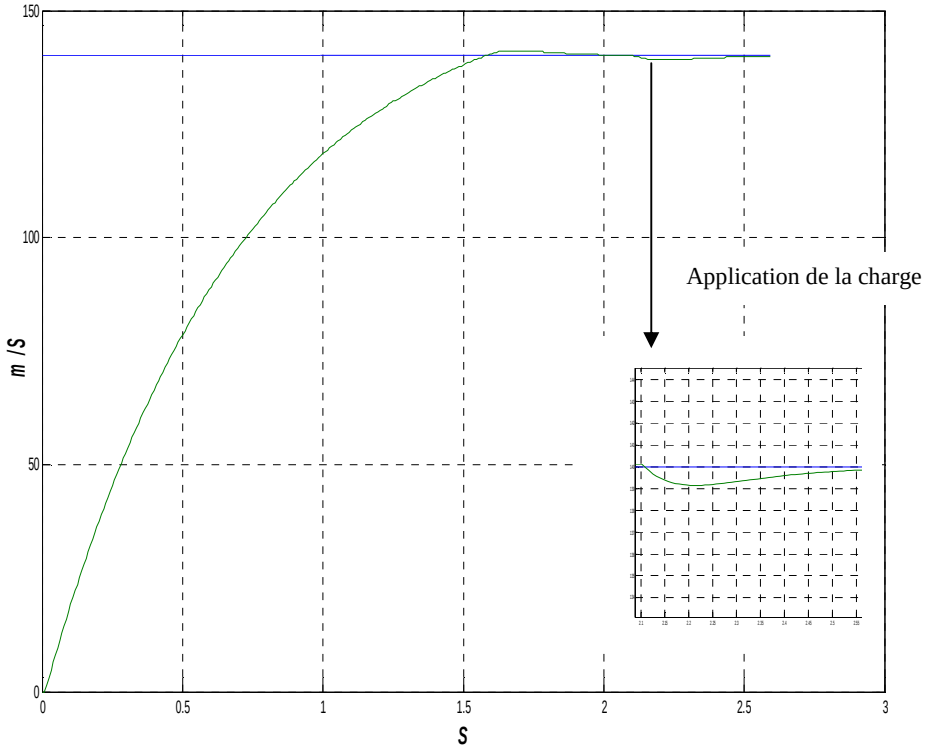


Figure :1.14 caractéristique de la vitesse en charge

Pour le couple on a T=k.Φ.Ia, donc le couple (Fig. 1.15) et le courant (Fig. 1.16)on la même forme. Avec l'application de la charge, on constate que le couple et le courant s'augment progressivement.

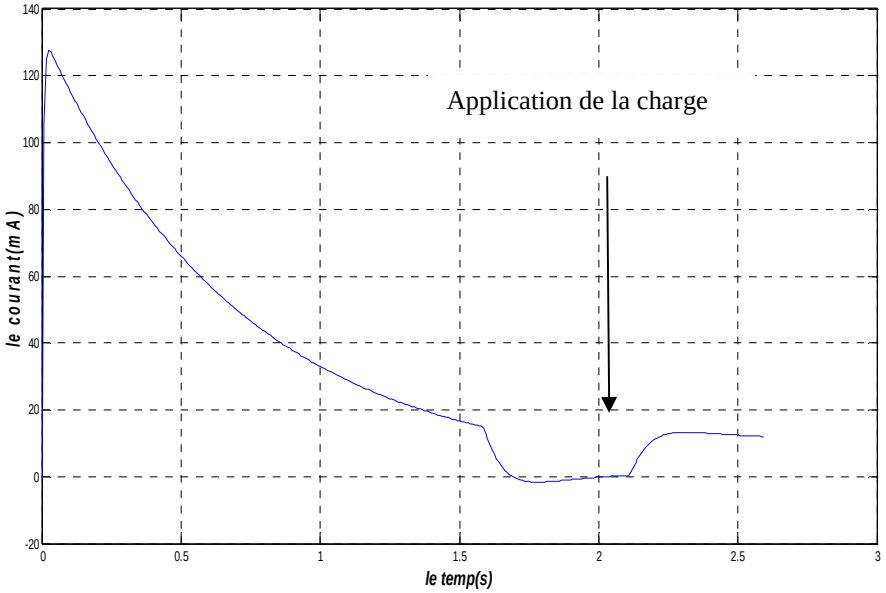


Figure I.15 : Caractéristique du courant en charge

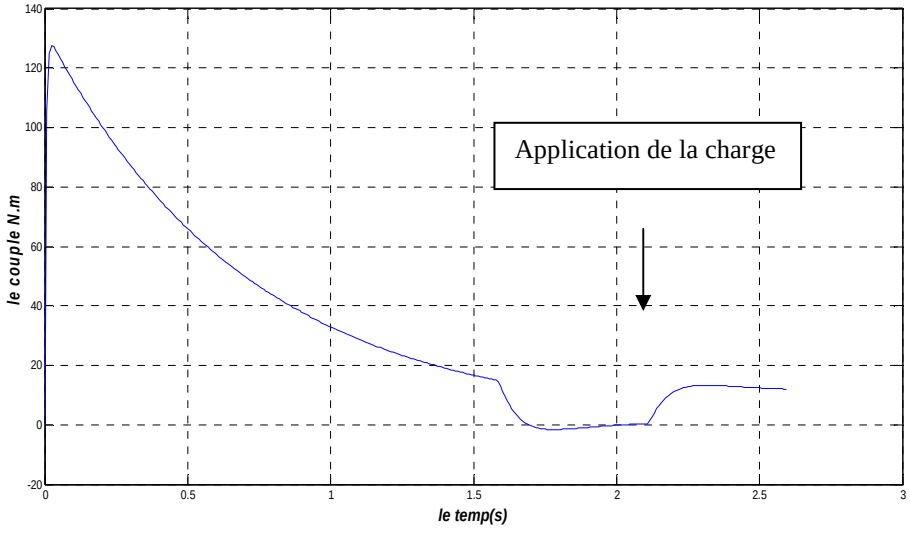


Figure I.16 : Caractéristique du couple en charge

I.7.2. Résulta simulation de commande avec convertisseur (hacheur) :

On observant les figures (1.17, 1.18 et 1.19), on peut constater que les performances sont identiques à celle obtenue sans hacheur sauf que, comparés aux résultats sans hacheur, on observe certaines fluctuations dans les résultats, en particulier pour le couple et le courant. Cet effet est due à la commutation ou à la l'alimentation découpée assuré par le hacheur.

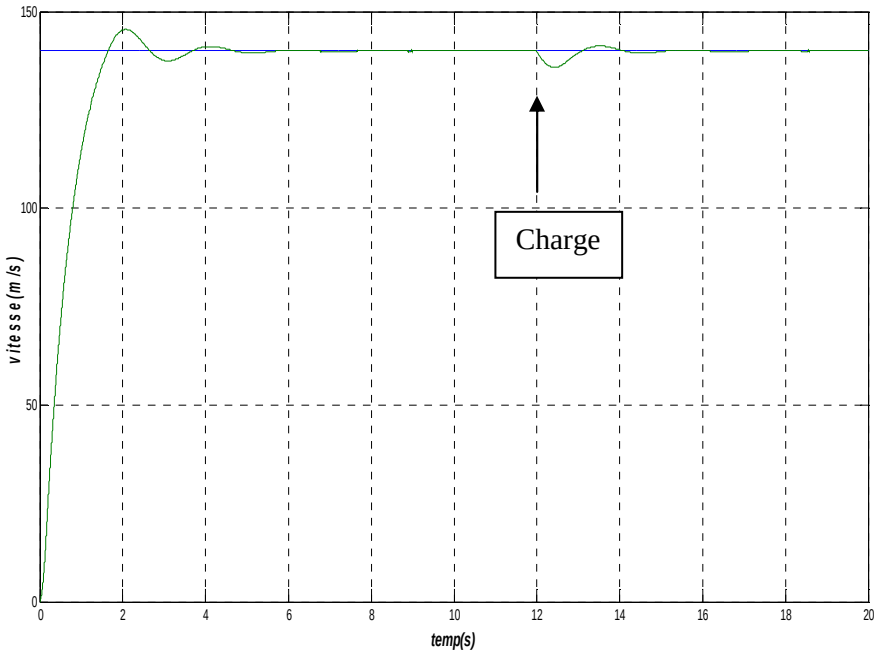


Figure 1.17 Caractéristique de vitesse en charge

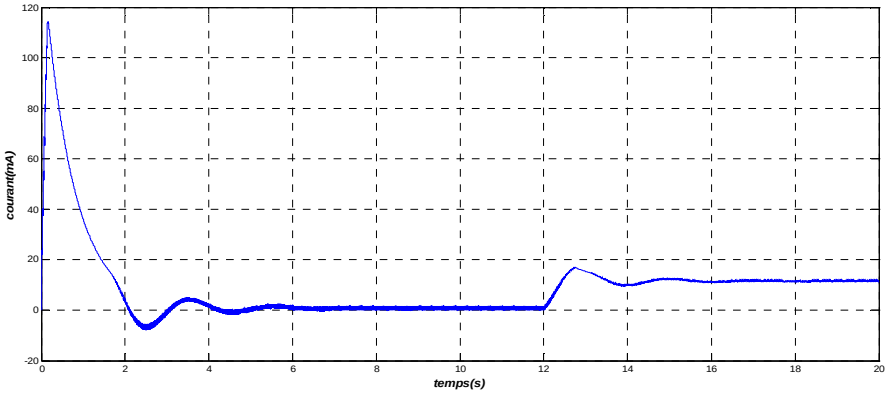


Figure I.18 : Caractéristique de courant en charge

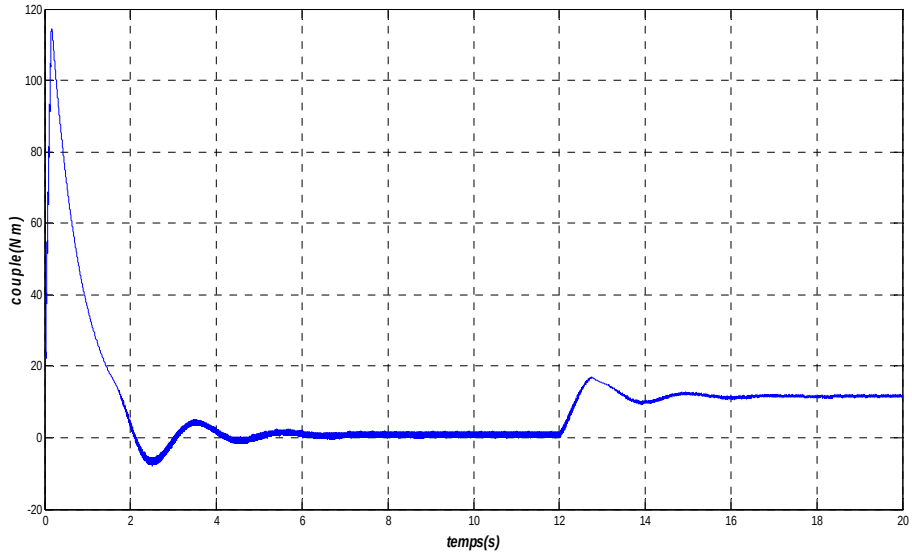


Figure1.19 : Caractéristique de couple en charge

1.8 .Conclusion

Nous avons présenté dans ce chapitre des généralités sur la machine à courant continu à excitation séparée ainsi que sa commande. Pour la partie commande, un hacheur avec commande MLI a été utilisé. Le modèle établis a été simulé pour différentes conditions de fonctionnement avec et sans hacheur. En effet, cet étape a été effectué pour la comparé avec une autre type de simulation dite hardware en utilisant les circuits FPGA qui sera l'objet du chapitre suivant.

Conclusion générale

Ce travail a été consacré à une commande en boucle ouvert d'un moteur à courant continu en utilisant un circuit FPGA.

D'après l'étude des différents types de moteur et suivant ces caractéristiques, on a vu que le moteur à excitation indépendante c'est lui qui s'adapte pour notre projet, il est caractérisé par une vitesse réglable par tension et indépendante de la charge. En association avec un convertisseur (hacheur) fournissant une tension réglable, la vitesse peut varier sur une large plage. Il fournit un couple important à faible vitesse.

Dans ce travaille, on effectuée trois taches complémentaire.

Pour la première tâche, la commande de la vitesse du moteur à courant continu a été achevée un utilisant un régulateur PI et un Hacheur MLI.

Pour la deuxième tache, on pour l'élaboration de la description hardware de l'algorithme de commande du moteur. La description hardware a été testée par une simulation hardware.

Pour la dernière, la commande en boucle ouverte du moteur à courant continue a été effectuée en utilisant un circuit FPGA. Un algorithme a été élaboré pour générer les impulsions MLI du Hacheur avec un mécanisme permet l'automatisation de la variation du rapport cyclique afin de varier la vitesse. Le schéma de commande a été testé et validé par simulation et par expérimentation.

Les résultats obtenus sont satisfaisants et nous encourage à tracer d'autres perspectifs pour terminer le présent projet telle la commande en boucle fermée du moteur.

Chapitre II :

CIRCUITS FPGA

Chapitre 2 les Circuits FPGA

2.1 Introduction :

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

Le présent chapitre se compose de deux parties. Dans la première partie on présente un aperçu sur les circuits FPGA, leur composition, leur principe et leur prototypage. Dans la deuxième partie, l'algorithme de la commande du moteur à courant continu est refait en utilisant le langage VHDL et, ensuite, une simulation hardware de cet algorithme est achevée.

2.2 Les circuits logiques programmables :

Alors qu'auparavant la distinction était nette entre le logiciel et le matériel, le circuit logique programmable FPGA est venu s'introduire comme un hybride entre les deux approches. Les circuits logiques programmables et reprogrammables, architecturalement, sont classifiés en trois grandes familles les PLD, CPLD et FPGA. La figure (2.1) illustre l'arborescence des différents types des circuits programmables.

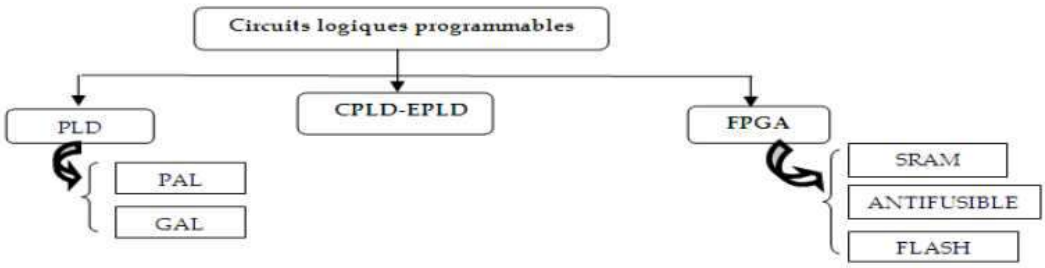


Figure 2.1 Diagramme des différents types de circuits logiques programmable.

Le PLD (Programmable LogicDevice) représente la version initiale des circuits programmables. Il contient des éléments distribues de logique combinatoire et un réseau d'interconnexion, parfois on trouve des éléments de logique séquentielle.Le CPLD est un circuit programmable avec une densité d'intégration relativement importante. Il contient en ensemble de PLD avec un bus de connexion unique permettant de réaliser les deux logiques combinatoire et séquentielleLes FPGA (Field Programmable Gâte Array) : ces circuits sont une évolution des CPLD. Récemment, ils intègrent également des mémoires entières, des multiplieurs et même des noyaux de processeur. Ils ont une densité d'intégration très élevée.

2.3 Les FPGA [7] :

Les FPGA (Field Programmable Gate Array) sont des circuits numériques configurables, dédiés à l'électronique numériques. A l'état initial ils ne peuvent rien faire mais disposent d'une importante quantité de ressources matérielles opérationnelles dont on peut configurer la fonction. Ces ressources sont, principalement, des blocs élémentaires logiques (pour réaliser des fonctions booléennes), des mémoires RAM, des opérateurs arithmétiques (qui travaillent en virgule fixe), des ressources de routage interne et des entrées/sorties. Ces ressources configurables sont reliées par un réseau dense de lignes de routage et de lignes de transport des horloges. Ces lignes de routage sont aussi configurables.

En plus de ces ressources, un FPGA est composé d'une mémoire interne de configuration. Chaque point de cette mémoire correspond à la configuration d'un élément d'une des ressources opérationnelles. Pour réaliser une application avec un FPGA, il faut décrire le circuit électronique à réaliser avec un langage de description matérielle comme le VHDL (Very High Speed Integrated Circuit Hardware Description Langage). Puis il faut synthétiser cette description en circuit électronique. Cette étape et les suivantes peuvent se faire avec des logiciels gratuitsfournies par le fabricant de circuit. Enfin après une étape de placement et routage qui prend en compte l'architecture du FPGA, un fichier de configuration appelé bitstream est généré. Celui-ci permet de spécifier au FPGA lors de la configuration la position des points de la mémoire de configuration. Parmi les principaux fabricants de FPGA dans le monde on peut citer : Xilinx (N°1 du marché des FPGA) et Altera.

2.3.1 L'architecture des circuits FPGA [8] :

Les FPGA sont composés d'éléments logiques de base, constitués de portes logiques, présentes physiquement sur le circuit. Ces portes sont reliées par un ensemble d'interconnexions modifiables, d'où l'aspect programmable du circuit, figure (2.2)

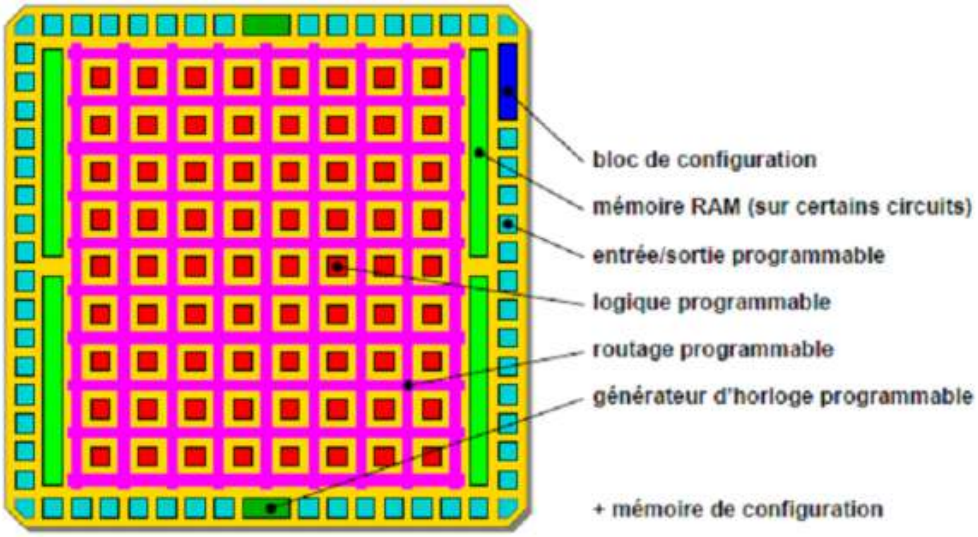


Figure 2.2 Structure d'un circuit FPGA

Typiquement, le circuit FPGA se compose des éléments suivants :

- De cellules d'entrées sorties modifiables qui servent d'interfaces entre les broches du circuit et le cœur du FPGA pour adapter les signaux suivants :
  - Alimentation
  - Signaux d'horloge
  - Signaux de configuration du FPGA
  - Signaux de test
- Des blocs logiques ou éléments logiques contenant les fonctions logiques combinatoires et séquentielles.
  - La partie combinatoire permet de réaliser des fonctions de complexité moyenne avec des portes classiques (ET, OU, etc.) de deux à une dizaine d'entrées.

- La partie séquentielle comporte une ou deux bascules généralement de type D. Compte tenu du nombre d'éléments logiques et de leur structure, leur association permet de réaliser tous les types de bascule. L'intérêt est de créer des mémoires élémentaires à un bit

- De réseaux d'interconnexions : Ces réseaux relient entre eux les blocs logiques et les blocs d'entrées/sorties.

2.3.2 Technologie de programmation des circuits FPGA :

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

- Technologie de programmation « SRAM ».
- Technologie de programmation «EEPROM ou FLASH ».
- Technologie de programmation « ANTI -FUSIBLE»

La technologie à base de SRAM (Xilinx et Altera)permet d'avoir une reconfiguration rapide des FPGA. Les points de connexions sont des ensembles de transistors commandés. L'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du FPGA dans une autre mémoire Flash, la Figure (2.3) représente la structure d'une cellule SRAM.

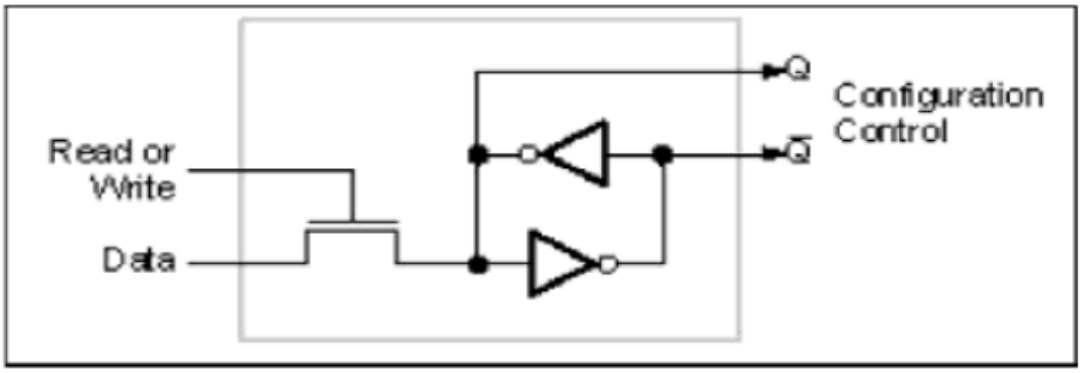


Figure 2.3 : Structure d'une cellule SRAM

Pour la technologie à base d'EEPROM ou FLASH, le circuit garde sa configuration mais on a besoin d'un dispositif spécial pour la programmation du circuit. Un autre inconvénient, c'est le nombre limité de configurations avec un temps de configuration important.

Pour la technologie à base d'ANTI-FUSIBLE, les points de connexions sont du type ROM, c'est–à-dire que la modification du point est irréversible. 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.

2.3.3 Domaines d'applications d'FPGA :

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

- Informatique : Périphériques spécialisés.
- Machinerie industrielle : Contrôleur pour machines.
- Télécommunications : Traitement d'images, Filtrage.
- Instrumentation : Équipement médical, Prototypage.
- Transport : Contrôle d'avions et métros.
- Aérospatiale : Satellites.

2.3.4 Avantages et inconvénients de FPGA :

- **avantages**
  - Technologie facile
  - Reprogrammable
  - Cout peut élève
  - Reconfiguration en temps réel

- **inconvénients**
  - Performance non optimise
  - Tempe de réponse longue par rapport aux ASIC
  - Rapidité due aux parallélismes par rapport aux circuits numériques standards

2.4 Programmation des circuits FPGA :

Pour la programmation des circuits FPGA, on trouve plusieurs langages dire de description hardware, parmi les quels Le langage VHDL (Very high speed integrated circuits Hardware Description Language). Ce langage est utilisé pour la modélisation et la synthèse de circuits numériques. La modélisation permet de simuler le comportement d'un circuit numérique, tandis que la synthèse permet de le programmer.

2.4.1 Historique de VHDL [4] :

Le VHDL a été développé dans les années 80 aux USA et est devenu une norme IEEE (n° 1076) en 1987. Révisée en 1993 pour supprimer quelques ambiguïtés, cette norme est vite devenue un standard en matière d'outils de description de fonctions logiques. Le VHDL est principalement utilisé pour concevoir des ASIC, programmer des composants du type PLD, CPLD, FPGA…, et concevoir des modèles de simulation numérique ou des bancs de test.

2.4.2 Structure d'une description VHDL

Le développement en VHDL nécessite l'utilisation de deux outils :

- le simulateur va nous permettre de simuler notre description VHDL avec un fichier de simulation appelé « test-bench » ; cet outil interprète directement le langage VHDL et il comprend l'ensemble du langage.
- le synthétiseur: traduire le comportement décrit en VHDL en fonctions logiques de bases, celles-ci dépendent de la technologie choisie ; cette étape est nommée « ***synthèse***». L'intégration finale dans le circuit cible est réalisée par l'outil de placement et routage. Celui-ci est fourni par le fabricant de la technologie choisie.

En plus de la simulation et la synthèse, la réalisation d'un projet sur un FPGA passe par déférents étapes (Fig. 2.4)

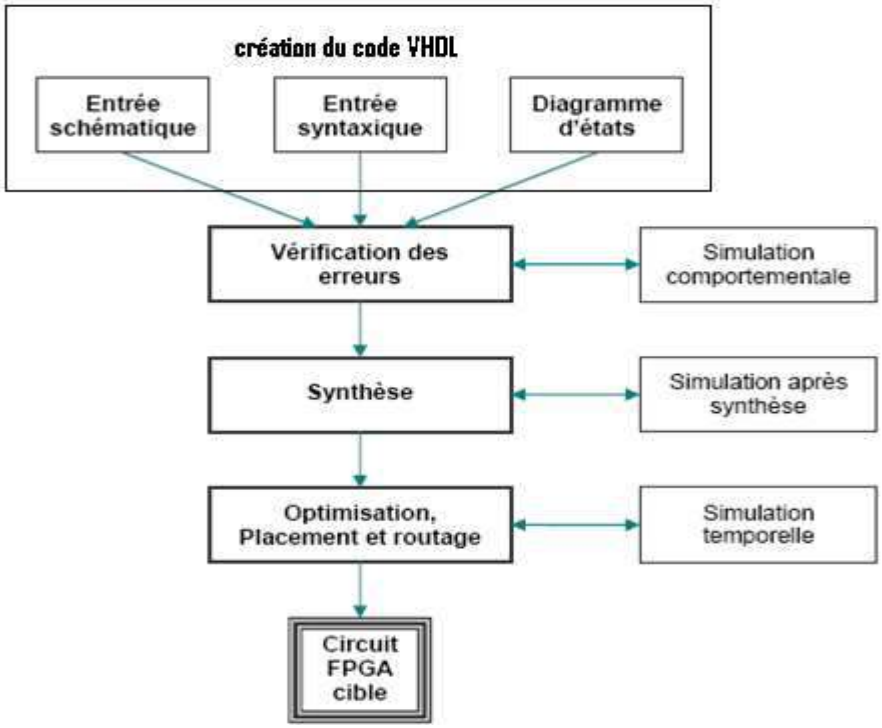


Figure 2.4 : Organisation fonctionnelle de développement d'un projet sur circuit FPGA

2.4.2.1 Création du code VHDL d'un développement :

Il existe défirent environnement pour crée une VHDL parmi lequel on trouve celui proposé par Xilinx***« ISE Xilinx Project Navigator»***. Ce logiciel propose une palette d'outils permettant d'effectuer toutes les étapes nécessaires au développement d'un projet sur circuit FPGA. La Figure (2.5) présente le mode textuel d'une description VHDL. Avant de passer aux autres étapes, la vérification de la syntaxe de la description est nécessaire.

Cette phase permet donc de valider la syntaxe du programme et de générer la ***« netlist»,*** qui est un fichier contenant la description de l'application sous forme d'équations logiques.

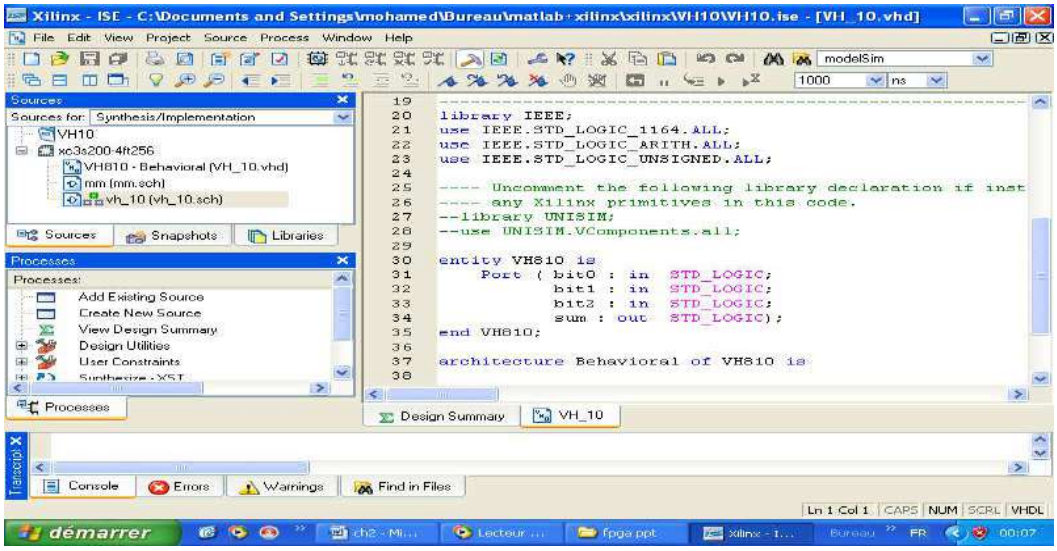


Figure 2.5 : *Vue d'ensemble du logiciel « Xilinx Project Navigator*»

2.4.2.2 Synthèse

La synthèse permet de réaliser l'implémentation physique d'un projet. Le synthétiseur a pour rôle de convertir le projet, en fonction du type du circuit FPGA cible utilisé, en portes logiques et bascules de base. L'outil « ***View RTL Schematic*** » permet devisualiser les schémas électroniques équivalents générés par le synthétiseur (Figure 2.6**)**

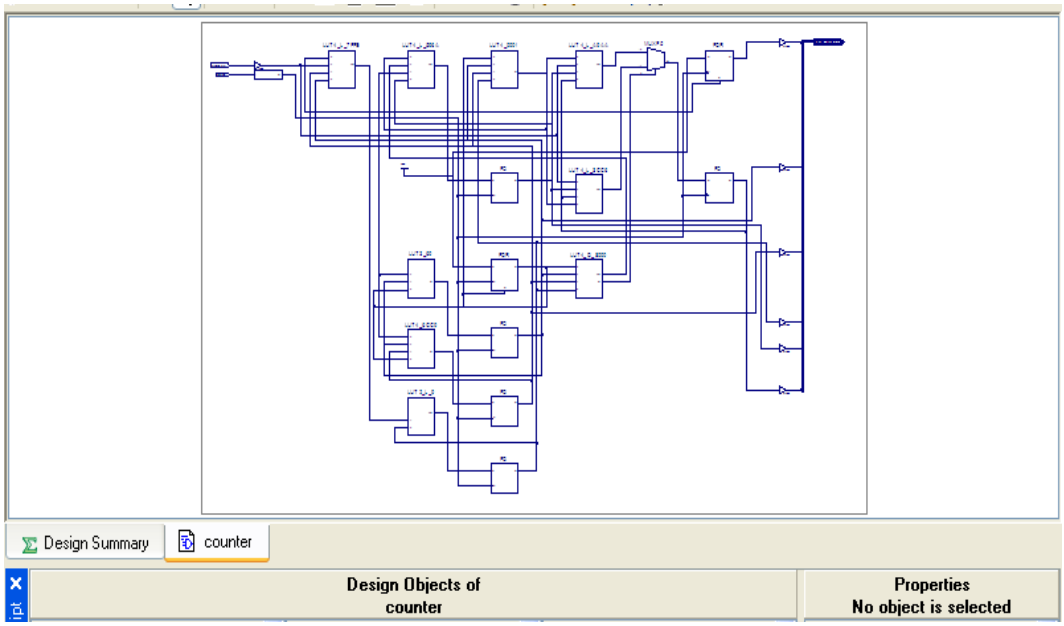


Figure 2.6 : Aperçu de l'outil « View RTL Schematic »

De plus, le synthétiseur permet à l'utilisateur d'imposer des contraintes detechnologie ***(User constraints***) : par exemple fixer la vitesse de fonctionnement (***CreateTiming Constraints***), délimiter la zone du circuit FPGA dans laquelle le routage doit se faire***(Create Area constraints)*** ou affecter les broches d'entrées/sorties ***(Assign Package Pins)*** (Fig. 2.7**)**

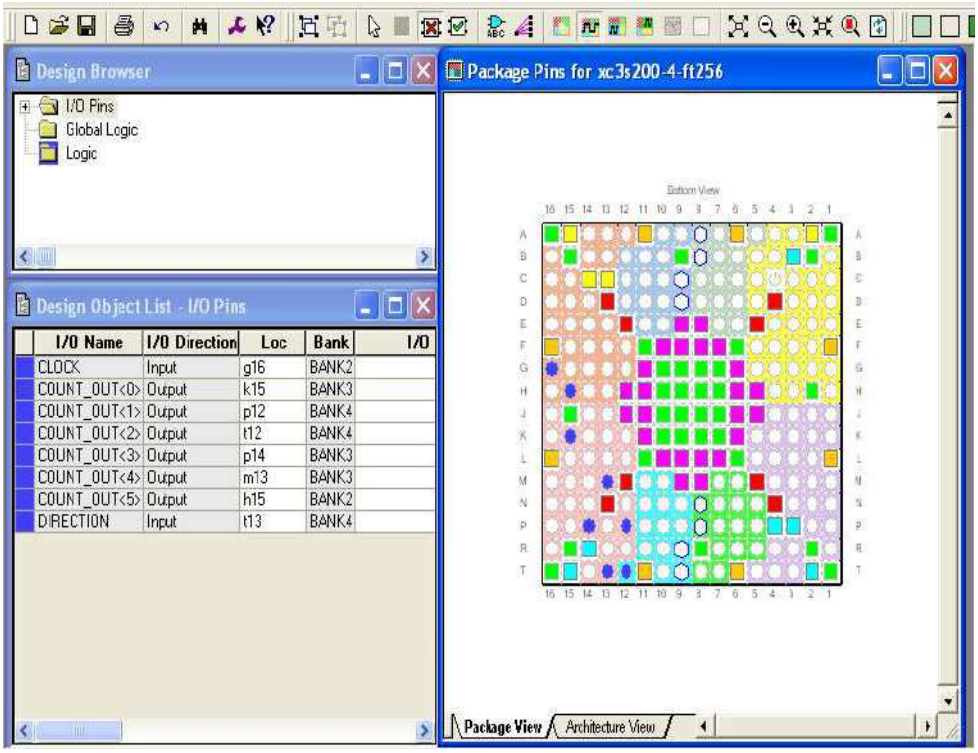


Figure 2.7 : Aperçu de l'outil d'affectation des broches d'entrées/sorties

2.4.2.3 Simulation

La simulation permet de vérifier le comportement d'un design avant ou après implémentation dans le composant cible. Elle représente une étape essentielle qui nous fera gagner du temps lors de la mise au point sur la carte. Il faut juste noter qu'un projet peut être simulé même s'il n'est pas synthétisable.

Lors de l'étape de simulation comportementale, on valide l'application indépendamment de l'architecture et des temps de propagation du futur circuit cible. La phase de simulation après synthèse valide l'application sur l'architecture du circuit cible, et enfin la simulation temporelle prend en compte les temps de propagation des signaux à l'intérieur du circuit FPGA cible.

2.4.3.4 .Optimisation, placement et routage

Pendant l'étape d'optimisation, l'outil cherche à minimiser les temps de propagation et à occuper le moins d'espace possible sur le circuit FPGA cible. Le placement et routage permettent de tracer les routes à suivre sur le circuit afin de réaliser le fonctionnement attendu (Fig.2.8**)**

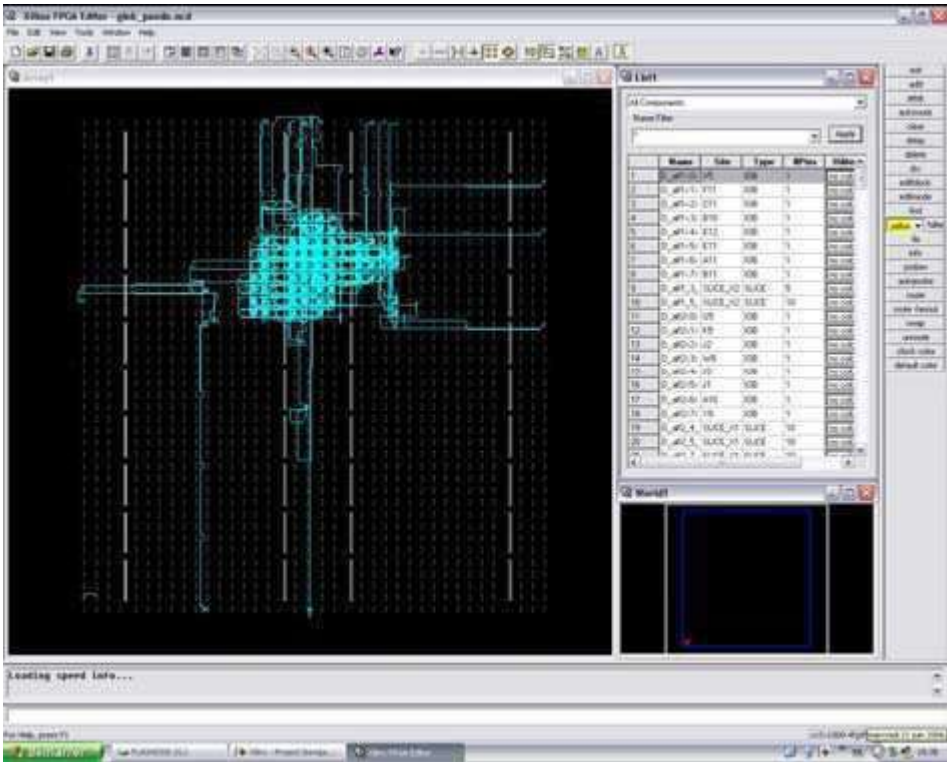


Figure 2.8 : Aperçu de l'outil « FPGA Editor »

24.2.5 Programmation du composant et test

Dans cette dernière étape (***GenerateProgramming Files***), on génère le fichier à charger sur le circuit FPGA à travers l'interface JTAG. Une fois le programme chargé sur le circuit, on peut tester et visualiser les résultats directement sur la carte de développement

2.4.3 Différences entre un langage de programmation ordinaire et VHDL [5] :

VHDL n'est pas un langage de programmation comme le C, c'est un langage de description matériel. Il est important de bien comprendre la différence entre les deux :

- Un langage de programmation est destiné à être traduit en langage machine puis à être exécuté par un microprocesseur.

- Un langage de description matériel comme VHDL décrit une réalité matérielle, c'est-à-dire le fonctionnement d'un système numérique. Il va être traduit (synthétisé) en un ensemble de circuits logiques combinatoires et séquentiels qui vont être implémentés dans un circuit intégré. Il n'y a aucun microprocesseur pour exécuter la description VHDL dans un circuit intégré.

2.5. Simulation hardware de l'algorithme de commande d'un moteur a CC

L'établissement d'une description hardware de l'algorithme de commande passe par différentes étape parmi les quels :

- L'adaptation des signaux du système avec l'environnement hardware
- La description VHDL de l'algorithme de commande
- Simulation de l'algorithme

2.5.1 Adaptation et initialisation des signaux :

Il est nécessaire de comprendre qu'en hardware les signaux sont dimensionnés en nombre de bits. Par conséquent et, avant d'entamer une description VHDL, il faut choisir une résolution (nombre de bits) appropriée pour les différents signaux utilisés dans la description. Cette phase nécessite la connaissance précise de la taille des signaux réels du système afin d'attribuer le nombre convenable de bits pour chaque un d'eux.

Un problème à résoudre, c'est les variables avec virgule. En effet, en Matlab on est très alise puisque en virgule flottante on peut aller à des précisions infinies. C'est pas le cas pour les circuits FPGA où deux problèmes s'imposent :

- Le problème la précision à considérer, autant importante autant qu'un important nombre de ressources sont nécessaires. Il faut faire donc un compromis entre la précision et le nombre de ressources utilisé. Pour notre cas, on considère les variables en virgule fixe avec une précision de trois chiffres après la virgule (10^{-3}).
- Le problème de la manipulation de la virgule fixe elle-même. En effet, le vocabulaire des circuits FPGA ne reconnait pas une variable avec virgule. Our traiter une telle variable il faut des codes assez complique et assez encombrant en terme de temps ou en terme de ressources. Pour surmonter ce

problème, dans notre projet on opter pour l'élimination de cette virgule er multipliant tous les signaux par 1024 ($2^{10} \cong 1000$). Le choix d'un nombre multiple de deux n'est pas abusif mais parce que les opérations de multiplication et de division par telle type de nombre n'est que des opérations de décalage logique à gauche et à droite. En circuit FPGA, ces opérations ne consomment aucunes ressources. La figure (2.9) illustre un exemple de choix des signaux de notre description.

```
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24  Use ieee.numeric_std.all;
25  Use ieee.std_logic_unsigned.all;
26  entity PI0 is
27      Port ( ck : in  STD_LOGIC;
28           er_v : in  STD_LOGIC_VECTOR (18 downto 0);
29           vref : out  STD_LOGIC_VECTOR (31 downto 0);
30          vref1 : out  STD_LOGIC_VECTOR (31 downto 0));
31  end PI0;
32  architecture arch of PI0 is
33  signal ki : std_logic_vector(14 downto 0) :="000000001000110"; ----1000110
34  signal kp : std_logic_vector(12 downto 0) := "0000000001111";
35  -- matlab h=1/2^14   ----->    xilinx h=1  Signal *14bit ;
36  signal up,up1,up2: std_logic_vector(31 downto 0) := ( others => '0');
37  signal ui : std_logic_vector(31 downto 0) := ( others => '0');
38  signal ui1 : std_logic_vector(32 downto 0) := ( others => '0');
39  signal ui2 : std_logic_vector(47 downto 0) := ( others => '0');
40  signal yp, yp1 : std_logic_vector(31 downto 0) := ( others => '0');
41  signal c: STD_LOGIC_VECTOR (31 downto 0) ;
42  signal fa: STD_LOGIC ;
```

Figure 2.9: Exemple de choix de nombre de bits pour les signaux utilisés

2.5.2 Description VHDL de l'algorithme de commande du moteur :

L'algorithme de commande du moteur comprend deux block :

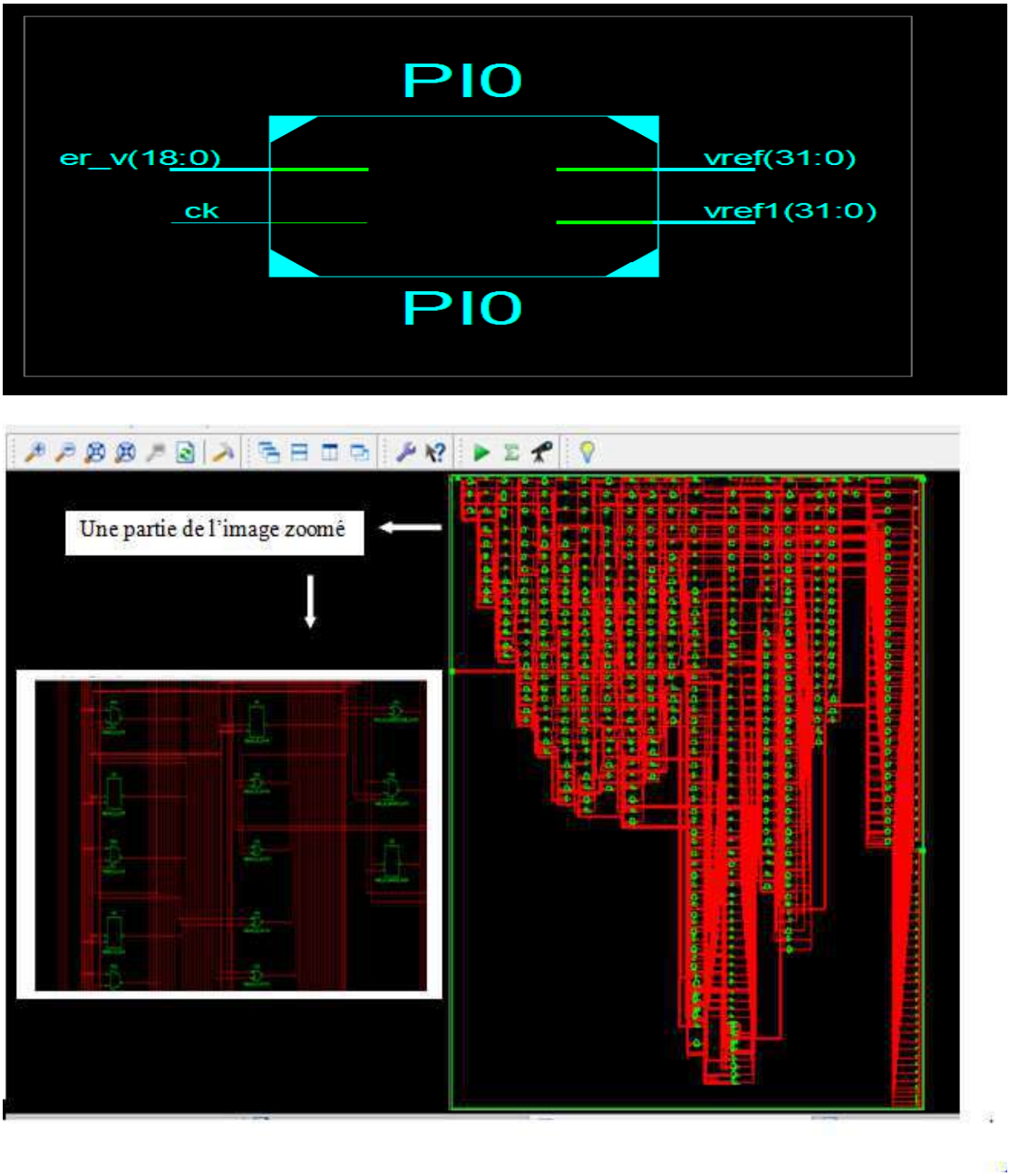
- Un régultaeur PI
- Un block pour le calcule de la tension de référence en se basans sur la sortie du régulateur.

Pour le régulateur PI, la méthode d'Euler a été utilisé pour simplifier la description. Dans ce cas l'action integrale est réalise par un accumulateur. Au cours de la description, pour chaque bloc, on vérifie sa sortie en utilisant un ensemble d'entrées/sorties prévenant de Matlab dans une phase de simulation. La figure (2.10), donne un exemple de la description d'une partie de l'algorithme de commande.

```
45  begin
46  process(ck)  -- programmable de les equation MCC-----
47  begin
48     if rising_edge(ck) then
49
50          -- action prportionnelle
51
52         up1<= kp*er_v ;
53        --- up2<= "0000000000" & up1(31 downto 10);-----------
54         up<= up1 ;
55
56          -- action integrale
57
58         ui1<=ui1 + ("00000000000000"&er_v) ;
59         ui2<= ki*ui1;
60         ui <= ui2(47 downto 16);
61         -- sortie du regulateur PI
62          yp1<= ui + up   ;
63
64        ------------------------------------
65
66           end if;
67  end process;
68   vref1<=yp1;
69    process(ck)
70     begin
71           if rising_edge(ck) then
72    if (yp1 > "00000000000000011011100000000000") then --------limitation
73    yp <= "00000000000000011011100000000000";
74    else
75    yp <= yp1;
76    ------------------------------------
77    end if;
78    end if;
79    end process;
80     vref  <= yp ;
```

Figure 2.10: Une partie de la description VHDL de l'algorithme de commande

La figure (2.11) donne le schema bloc de la description faite ainsi que sont développment détaillé. Ce schéma, appelé schéma technologique, présente l'image du circuit correspondant à l'algorithme de commande qui sera implémenté sur le circuit FPGA.



Figeur 2.11 : schema technologique de l'algorithme de commande élaboré

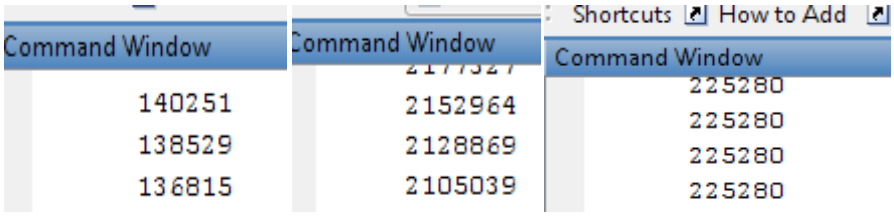
2.5.3 Simulation de l'algorithme de commande

Pour vérifier le comportement de la description VHDL élaborée, une étape de teste et validation est nécessaire. Pour ce faire, on a utilisé les données du modèle de Matlab. Les mêmes entrées du modèle Simulink ont été utilisées avec le modèle hardware élaboré et les sorties développées ont été comparées avec celle obtenues par Matlab.

2.6.Résultats de simulation Hardware :

Après avoir importer les resultats de simulation dans l'environnement Matlab/Simulink pour les régimes transitoires et permanent, un test de validation a été appliqué sur le model hardware établit. Les résultats obtenus prove que le modèle hardware donne des résultats très proche de ceux obtenu par matlab. Les légéres diffférence entre les valeurs prévenant de Matlab et ceux de Xilinx sont dues aux la précision considére. En effet le matlab travaille avec une virgule flottante, tandis que pour notre projet, une précision fixe de trois chiffre apprés la virgule a été considérée. Les résultas obtenus (Fig. 12 et 13)permet de confirmer la validité du modèle hardwarr élaboré.

L'erreur (er_v).vref1 avant l'imitateur.vref après l'imitateur.



Résultat VHDL :

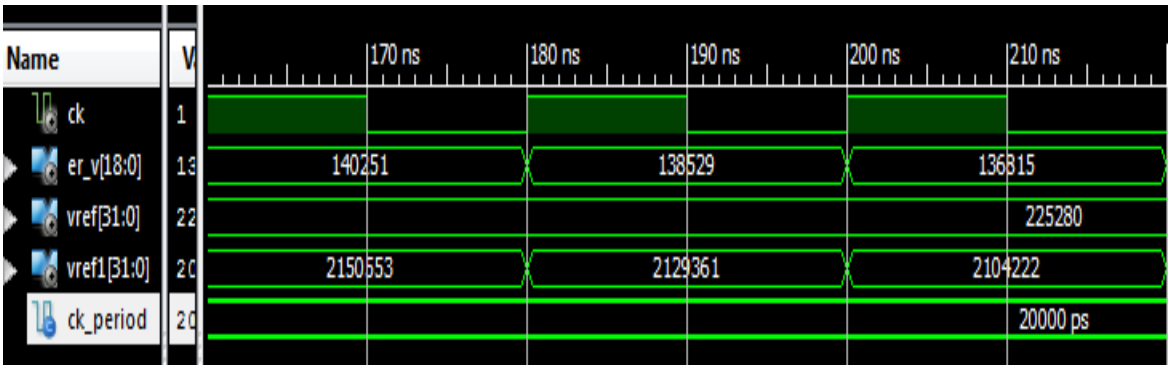
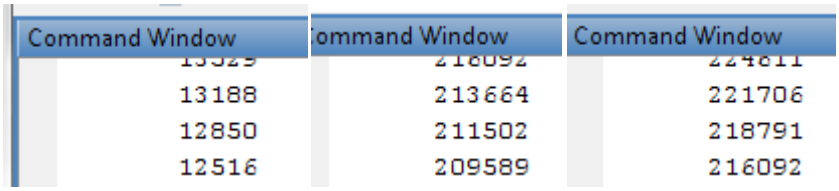


Figure 2.12 : Comparaison entre résultat en régime permanant

Les erreur (er_v). vref1.vref



Résultat VHDL :

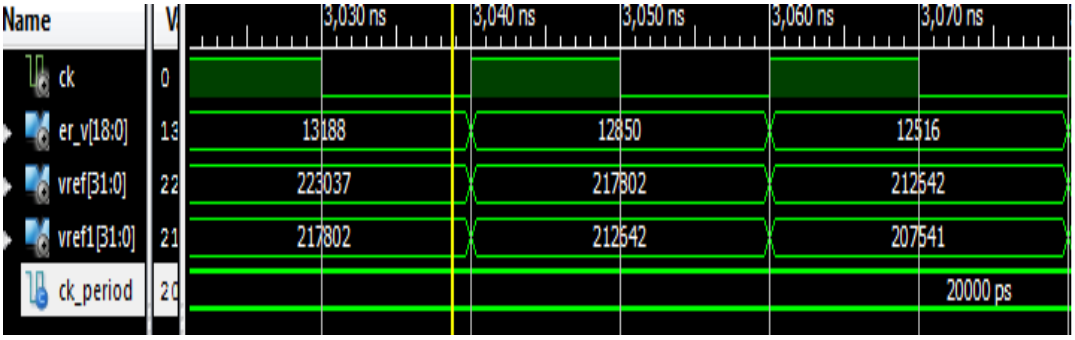


Figure 2.13: Comparaison entre résultat en régime transitoire

2.10. Conclusion :

Dans ce chapitre, on a donné, au premier, un aperçu sur les circuits FPGA concernant leur structure, leur principe, leur programmation ainsi que sur le langage de description VHDL. Ensuite, on a opté pour la description hardware de l'algorithme de commande du moteur CC. Les résultats de simulation hardware effectuée prouve la validité de la description hardware élaborée.

Chapitre III : IMPLEMENTA-TION ET RESULTATS

CHAPITRE 3 : implémentation et résultats

3.1 Introduction :

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

3.2 Présentation de l'outil de développement ISE XILINX :

Le logiciel ISE XILINX (Integrated Software Environnement) est un environnement de développement qui possède différent outils de CAO. Tous les fabricants de FPGAproposent des outils de CAO pour configurer leurs circuits. L'offre logicielle dans le domaine de conception des circuits numériques est très varié et l'un parmi ces environnement que nous allons exploiter au cours de ce travail est le ISE XILINX. C'est un logiciel multitâche qui possède dans son soft différents outils permettent la création des systèmes ou circuits numériques. XILINX ISE permet la simulation de la description et la synthèse du circuit logique équivalent puis placer et router ce circuit sur un prototype correspondant à une technologie FPGA bien précise. Enfin lorsque toute les vérifications sont faites, vient l'implantation sur une carte réelle ce qui correspond à générer un fichier (bit Stream) correspondant au circuit logique conçu. D'une manière générale, l'environnement XIILINX ISE met en œuvre quatre outils pour réaliser une conception :

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

3.3 Présentation de L'analyseur Chip Scope :

L'analyseur Chip Scope de Xilinx est un ensemble d'outils fabriqués par Xilinx qui nous permet de sonder facilement les signaux internes de la conception dans un circuit FPGA, mais moins que comme on peut faire avec un analyseur logique. Par exemple, si notre conception est exécutée sur le FPGA, nous pouvons le déclencher lorsque certains événements se produisent et afficher l'un des signaux internes de notre conception parce que la logique de l'analyseur de Chip Scope est implémentée dans le FPGA. En autre terme, le Chip Scope représente une version d'un Oscilloscope numérique (fig. 3.1).



Figure 3.1 analyseur Chip Scope

3.4Commande MLI d'un hacheur [1] :

Dans le cas de la commande MLI on fait varier l'état de l'interrupteur à une cadence qui ne dépend pas de la manière dont évoluent les grandeurs relatives aux systèmes interconnectés par le convertisseur électronique de puissance, cette cadence étant fixée essentiellement en fonction de la vitesse de commutation de l'interrupteur.

Comme indiquer dans le premier chapitre, la vitesse de rotation d'un moteur à courant continu, est proportionnelle à la tension d'alimentation de l'induit. Pour varier cette vitesse, on peut agir sur la valeur moyenne de la tension d'alimentation. A travers la technique MLI, on agit sur le rapport cyclique pour varier la tension.

Pour ce faire, en VHDL, nous proposons deux variantes :

- La génération d'un signal carré avec un rapport cyclique variable
- La comparaison entre deux signaux : un signal triangulaire et un signal fixe représente la valeur moyenne de la tension à reproduire

3.4.1Génération d'un signalcarré :

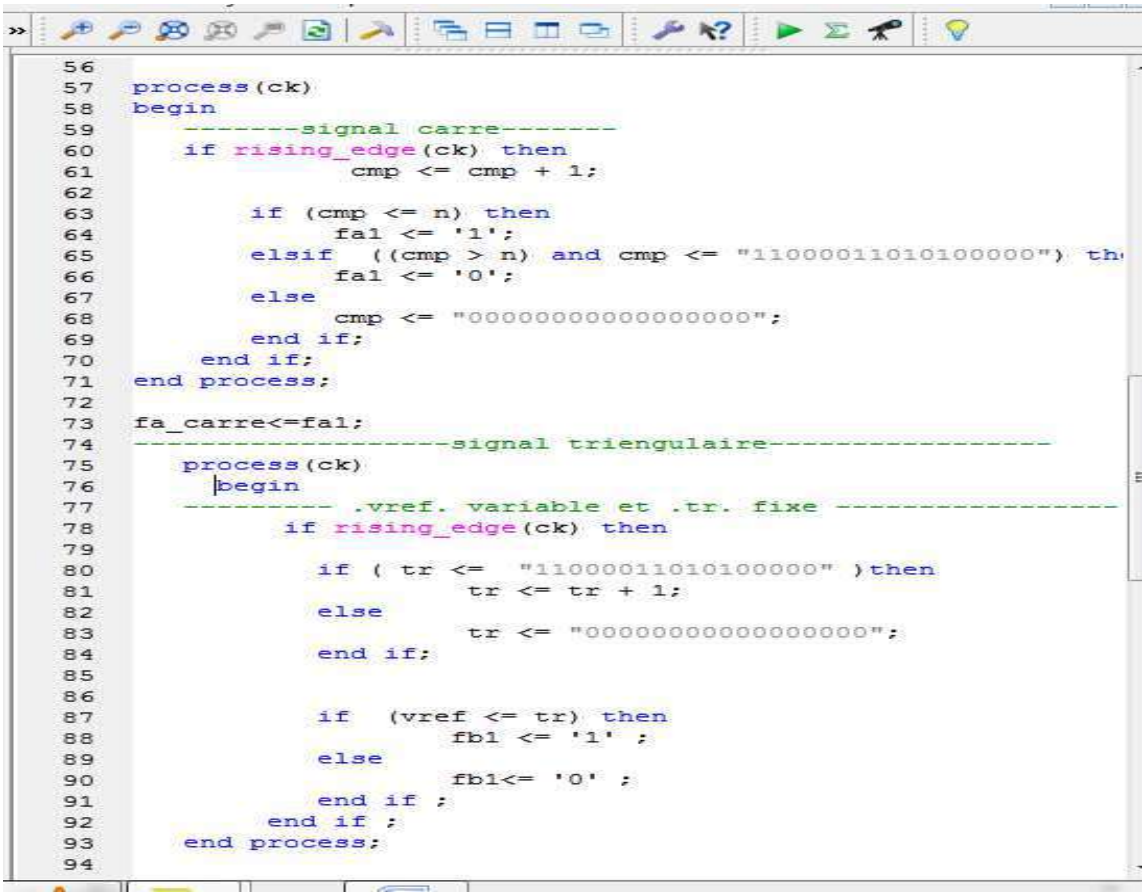
Par principe, on fixe une période de travail approprié, cette période est divisée en deux cycles :

- Cycle correspondant à un transistor fermé, état 1.
- Cycle correspondant à un transistor ouvert, état 0

En VHDL, la période de travail est évaluée en nombre de cycle d'horloge. Pour assurer cette période, on utilise un compteur incrémenté dont la limite maximale correspond à un nombre de cycles d'horloge équivalent à la période de travail. Pour les états ouvert et fermé, on prend une valeur intermédiaire du compteur correspondant au rapport cyclique voulu. Cette valeur est comparée avec l'état du compteur pour générer les états 1 et 0 (Fig. 3.2)

3.4.2Génération d'un signalMLI par signal triangulaire :

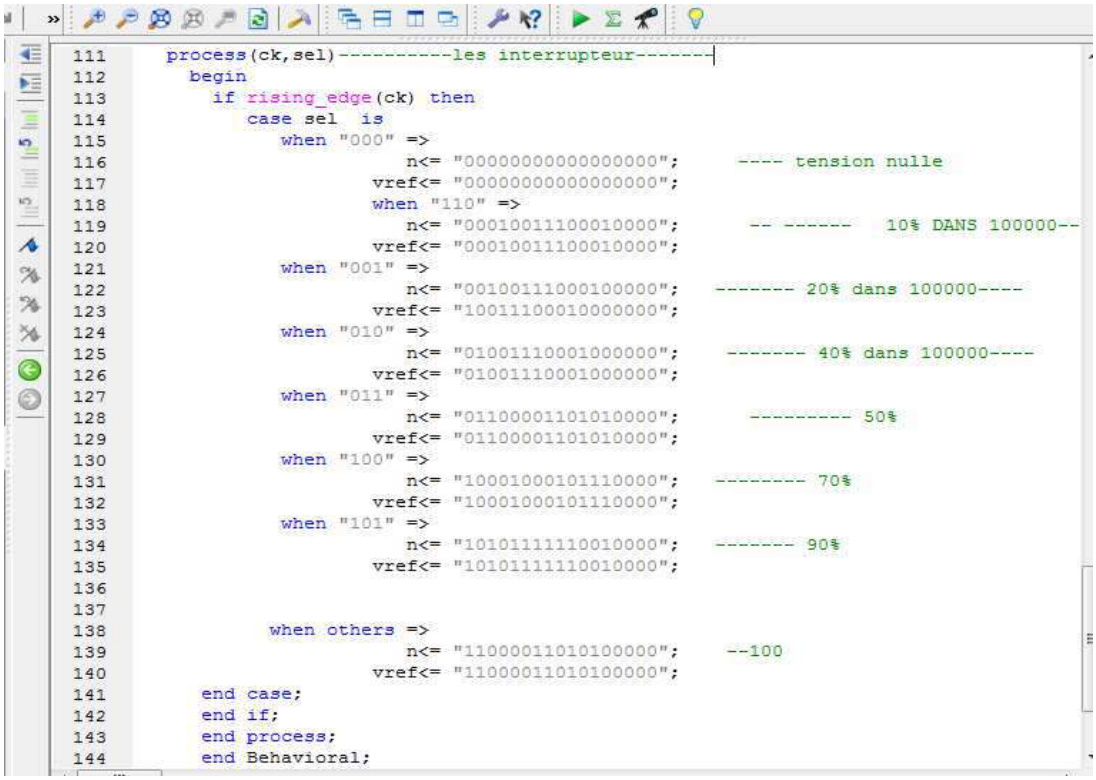
Dans ce cas, un signal triangulaire d'une fréquence appropriée est généré. Les impulsions sont générées en comparant le signal triangulaire avec une tension fixe. Le rapport cyclique entre état ouverte et état fermée est varié en modifiant la valeur de la tension fixe, (Fig. 3.2).



```
56
57  process(ck)
58  begin
59     -------signal carre--------
60     if rising_edge(ck) then
61                 cmp <= cmp + 1;
62
63          if (cmp <= n) then
64                fa1 <= '1';
65          elsif  ((cmp > n) and cmp <= "1100001101010000") th
66                fa1 <= '0';
67          else
68                cmp <= "0000000000000000";
69          end if;
70       end if;
71  end process;
72
73  fa_carre<=fa1;
74  --------------------signal triengulaire-----------------
75     process(ck)
76       begin
77     --------- .vref. variable et .tr. fixe ------------------
78           if rising_edge(ck) then
79
80              if ( tr <=  "1100001101010000" )then
81                          tr <= tr + 1;
82              else
83                          tr <= "0000000000000000";
84              end if;
85
86
87              if  (vref <= tr) then
88                        fb1 <= '1' ;
89              else
90                        fb1<= '0' ;
91              end if ;
92          end if ;
93     end process;
94
```

Figure3.2 Description VHDL pour la génération des signaux MLI

Pour automatiser, le processus de la variation de la vitesse, on exploite une combinaison d'interrupteurs disponibles sur la carte. Pour chaque combinaison des états des interrupteurs, on attribue une valeur pour le rapport cyclique. En effet, pour chaque rapport correspond une tension d'alimentation et, par conséquent, une vitesse du moteur (Fig. 3.3).



```
111    process(ck,sel)----------les interrupteur-------
112      begin
113        if rising_edge(ck) then
114           case sel  is
115              when "000" =>
116                        n<= "00000000000000000";     ---- tension nulle
117                     vref<= "00000000000000000";
118                     when "110" =>
119                        n<= "00010011100010000";      -- ------   10% DANS 100000--
120                     vref<= "00010011100010000";
121              when "001" =>
122                        n<= "00100111000100000";    -------- 20% dans 100000-----
123                     vref<= "10011100010000000";
124              when "010" =>
125                        n<= "01001110001000000";     -------- 40% dans 100000-----
126                     vref<= "01001110001000000";
127              when "011" =>
128                        n<= "01100001101010000";      ---------- 50%
129                     vref<= "01100001101010000";
130              when "100" =>
131                        n<= "10001000101110000";    --------- 70%
132                     vref<= "10001000101110000";
133              when "101" =>
134                        n<= "10101111110010000";    -------- 90%
135                     vref<= "10101111110010000";
136
137
138              when others =>
139                        n<= "11000011010100000";     --100
140                     vref<= "11000011010100000";
141        end case;
142        end if;
143        end process;
144        end Behavioral;
```

Figure3.3 Description VHDL pour la variation du rapport cyclique

3.4.2Résultat de simulation software :

Pour vérifier la forme des signaux MLI générés, les deux algorithmes proposés ont été testé par simulation. Pour chaque combinaison des trois interrupteurs (3bits), on attribue un rapport cyclique. Par exemple : les combinaisons (001;010; 011; 100 ;101 ;110;111) correspondent, respectivement, au rapport cyclique : 10%,20%,40%, 50%, 70%, 90% et 100%.

Les résultats obtenus sont illustrée sur la figure (3.4). On peut constater, que les signaux MLI générées par les deux algorithmes correspondent aux situations prescrites.

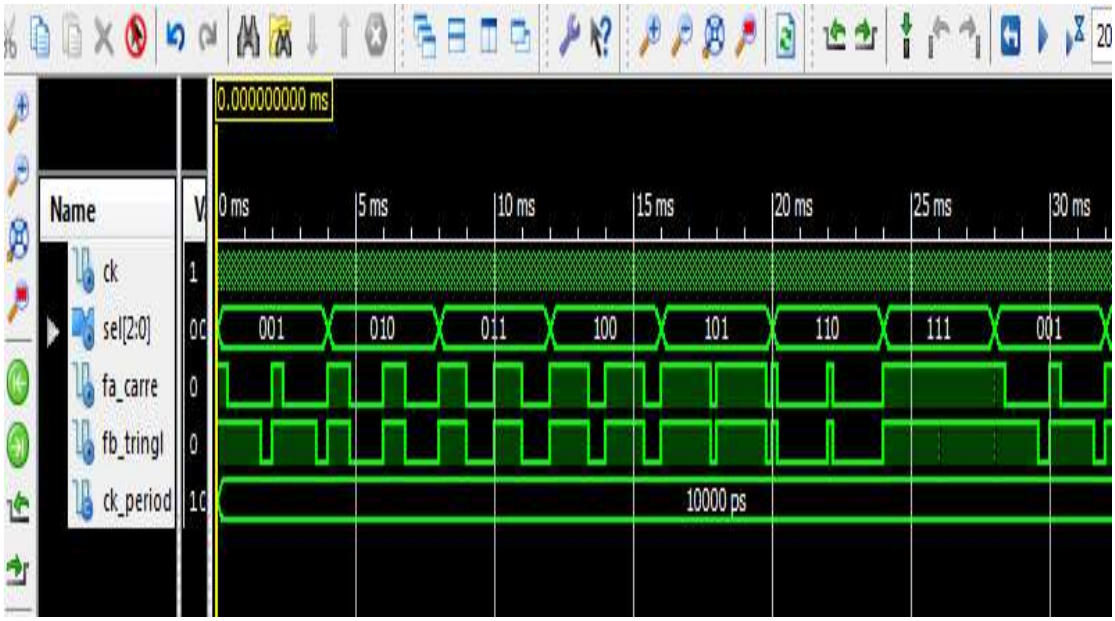


Figure 3.4 : simulationdes techniques de commande

3.5.2La simulation hardware de la technique de command MLI :

Avant de passer à l'exploitation réelle des signaux MLI généré, reste une dernière étape qui consiste à vérifier la forme réelle de ces signaux. Dans ce cas, la description est compiler afin de généré le fichier bitStream. Ensuite, le fichier généré est chargé dans la carte via la liaison JETAG (Fig. 3.5). Le rapport cyclique est varié en agissant sur les interrupteurs du circuit de la carte FPGA comme indique dans la description VHDL.

Pour visualiser les signaux, on utilise l'analyseur Chip Scope de Xilinxqui nous permet de sonder facilement les signaux internes de la conception dans le circuit FPGA. Les sorties de la conception peuvent être aussi visualisé en utilisant un oscilloscope (Fig. 3.5).

Les résultats obtenus pour différents rapports cycliques sont donnés sur les figures (3.6 à 3.9). On peut voir clairement la correspondance entre les signaux MLI générés et les rapports cyclique sélectionnés. Chose qui prouve la validité de la conception ainsi que l'implémentation réalisée.

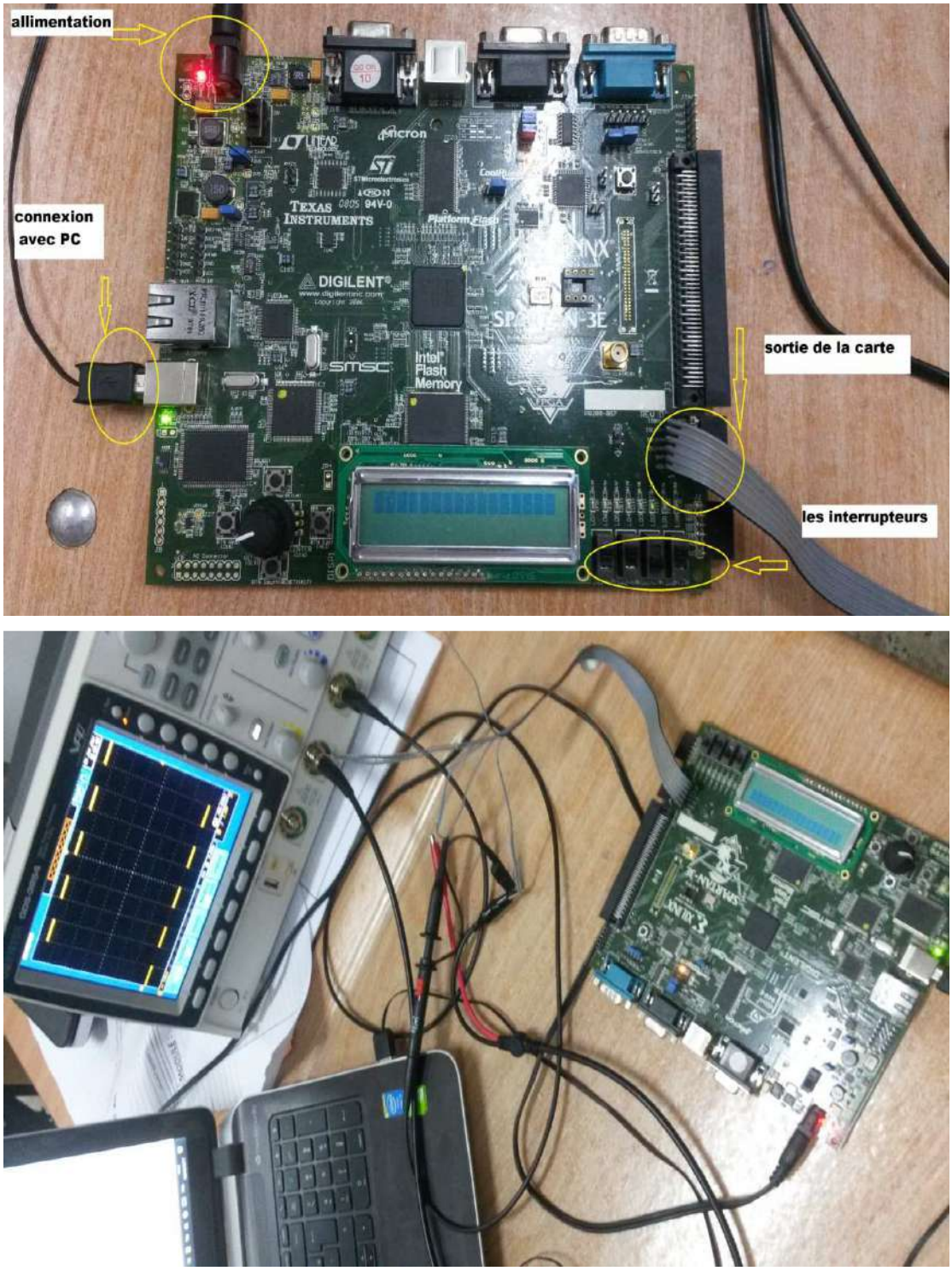


Figure3.5 Banc d'expérimentation utilisé

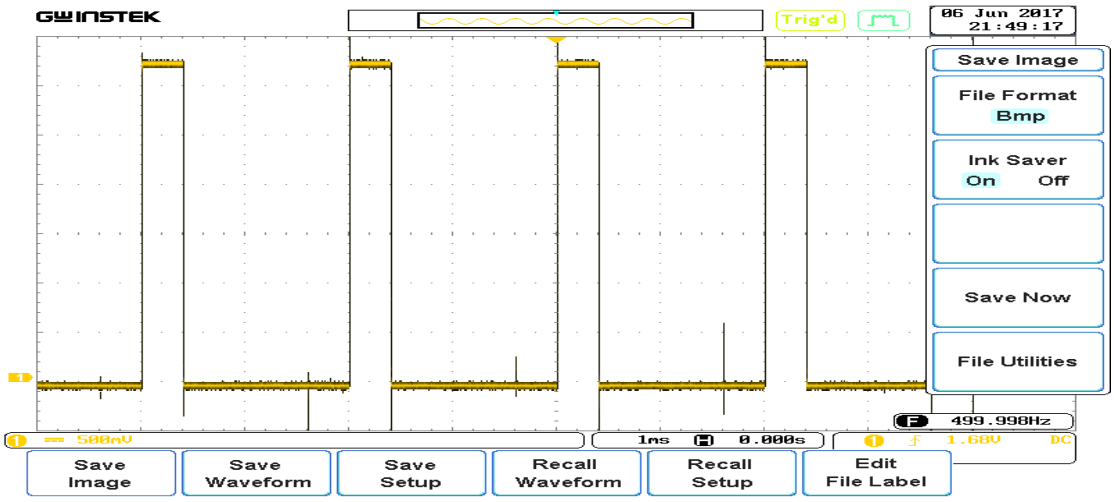


Figure 3.6la tentions de sortie avec un rapport cyclique de 20 %

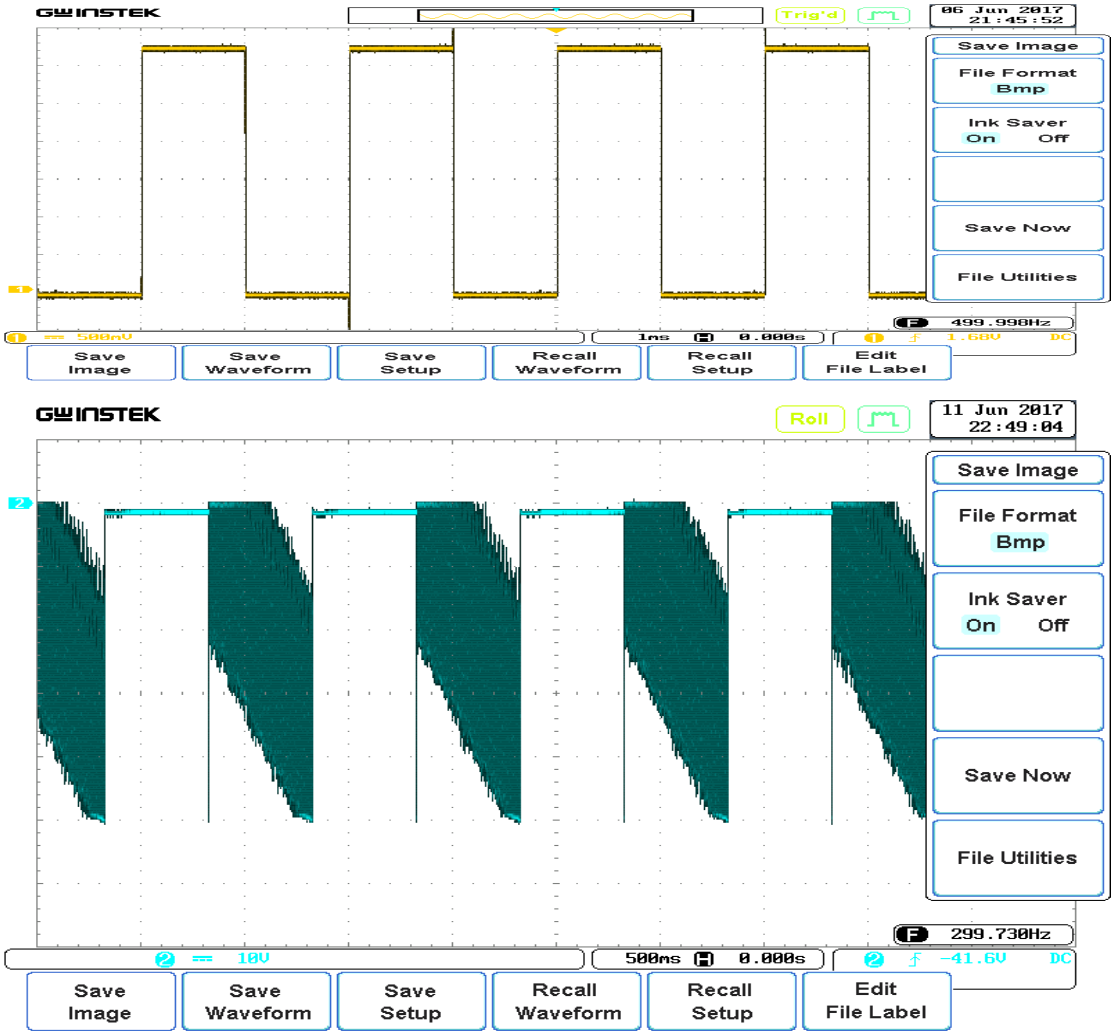


Figure 3.7 la tentions de sortie avec un rapport cyclique de 50 %

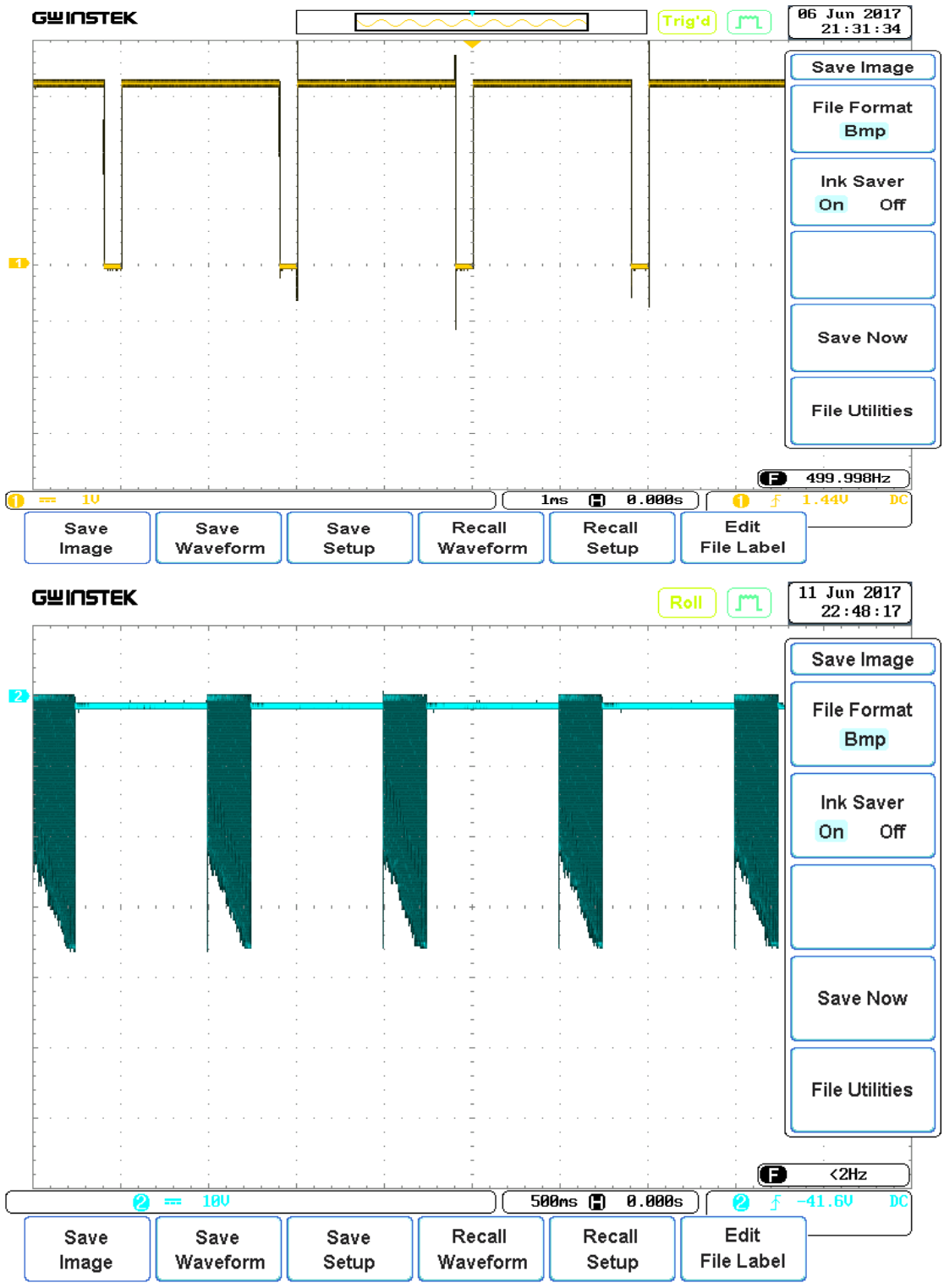


Figure 3.8 la tentions de sortie avec un rapport cyclique de 90 %

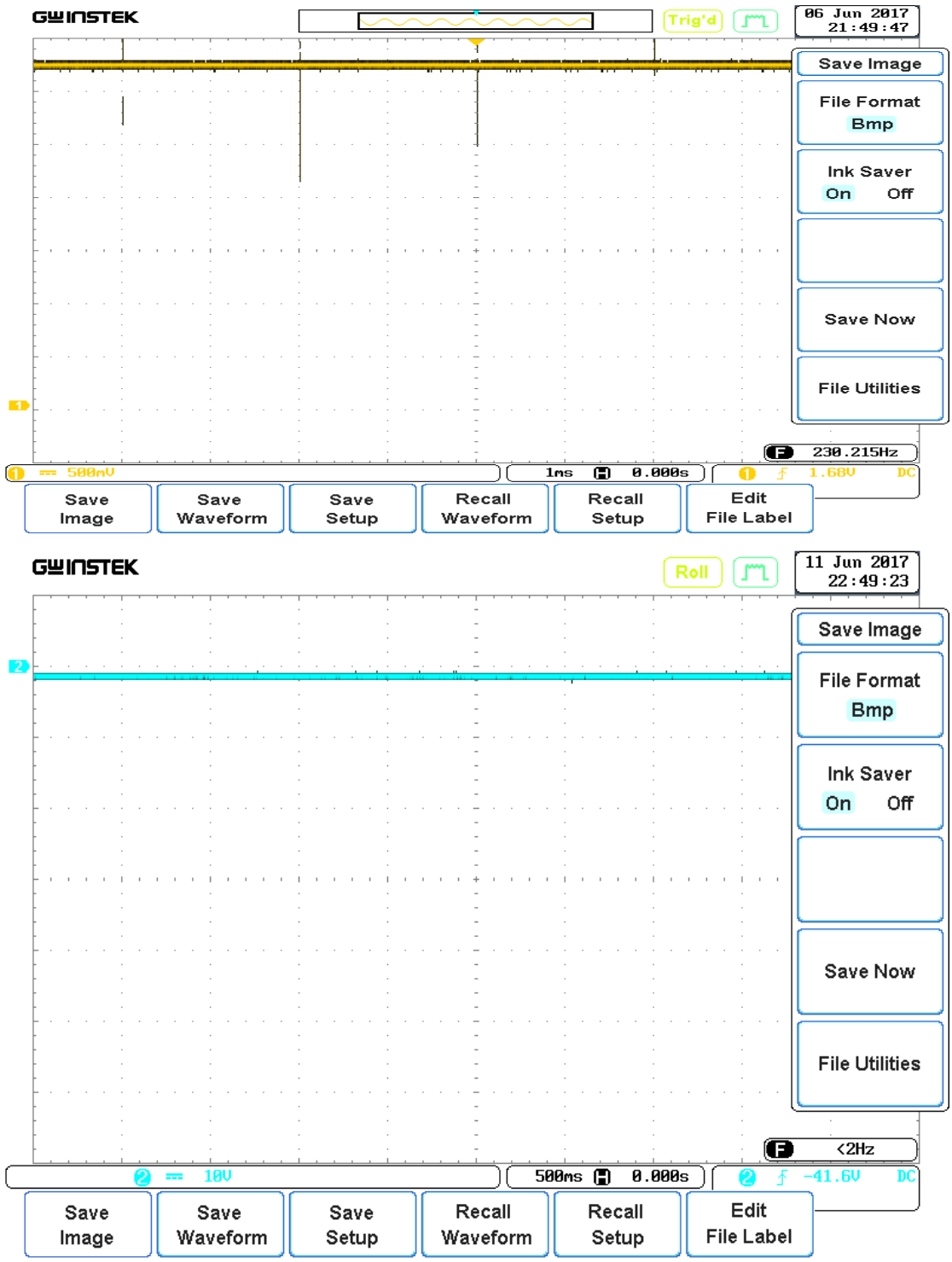


Figure 3.9la tentions de sortie avec un rapport cyclique de 100 %

- Interprétation

  D'après avoir faire le montage, la réalisation d'une commande de la machine a courant continu avec convertisseur

  On 'a pu commander la machine à travers des interrupteurs de la même manier que ce que on 'a fait en software voire les figures (3.6 à 3.9)

On voit qu'il y'a une ressemblance entre les signaux MLI générés et les rapports cyclique sélectionnés. Ce qui donne la validité de la conception faite ainsi que l'implémentation réalisée. Dans ce qui suit on a visualisé la vitesse de rotation de la machine dans l'interface chip scope .Et on peut constater que si on veut varier la vitesse on joue sur le rapport cyclique Et les résultats obtenus sont donnée dans les figures suivant :

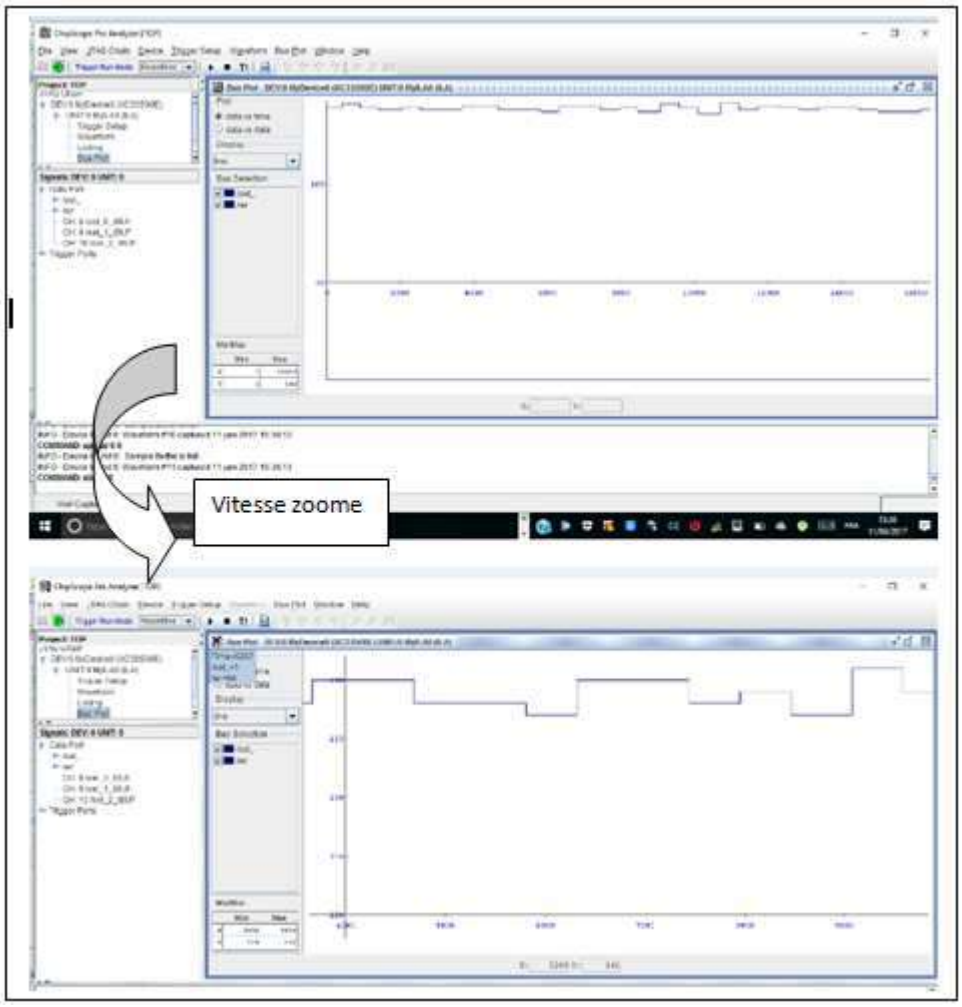


Figure 3.10 : vitesse correspondante a 50% du rapport cyclique

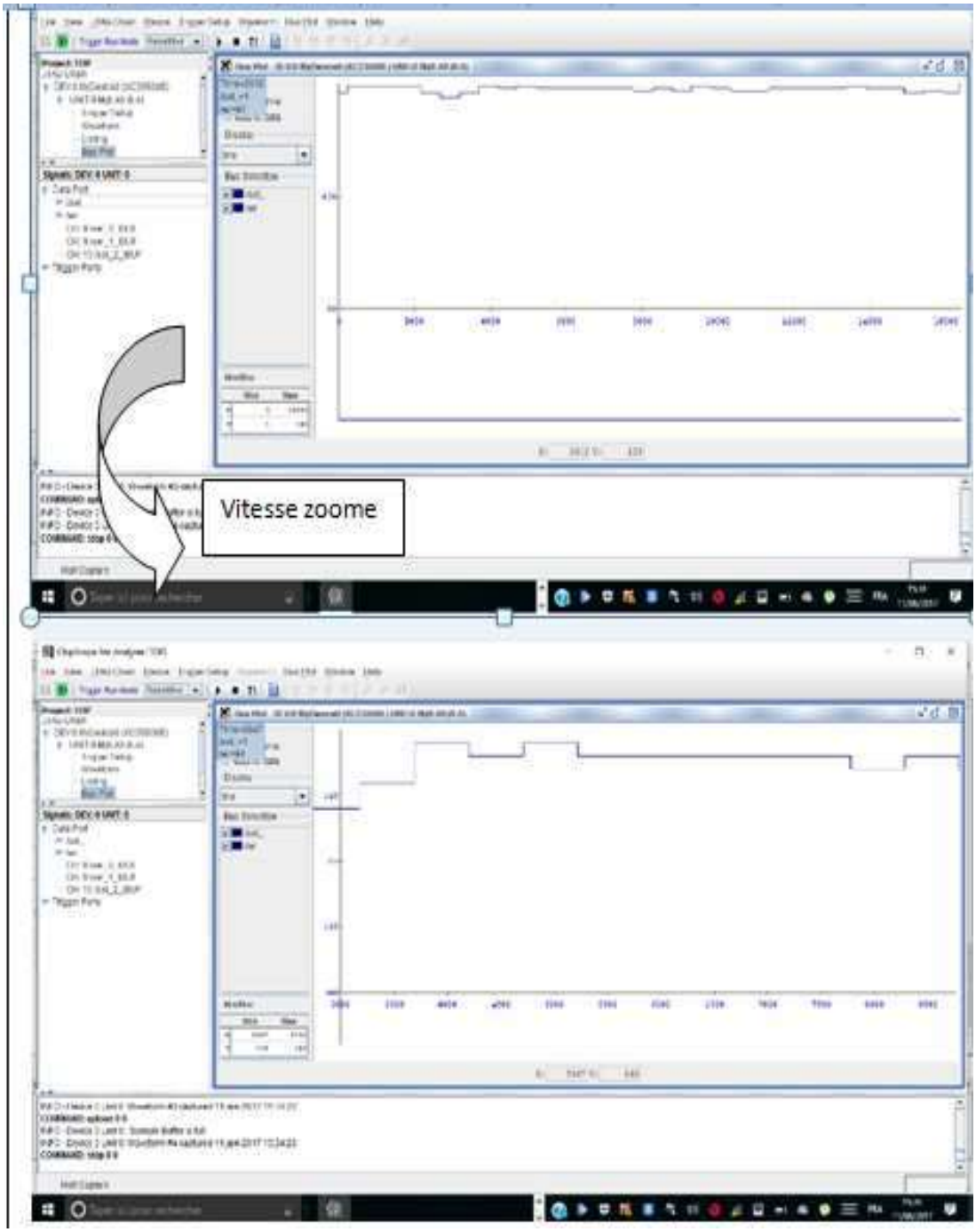


Figure 3.11:vitesse correspondante a 75% du rapport cyclique

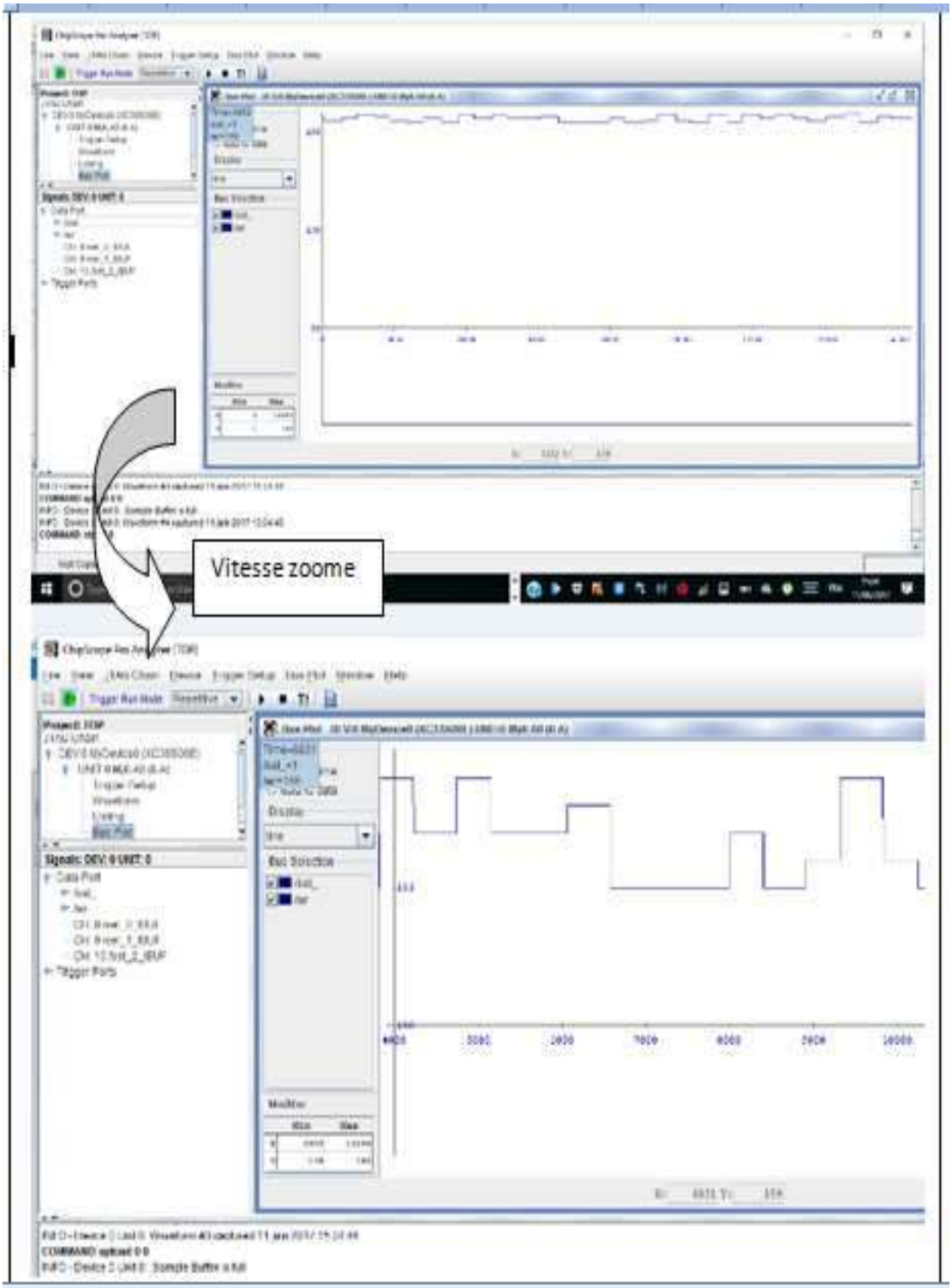


Figure 3.12 :vitesse correspondante a 100% du rapport cyclique

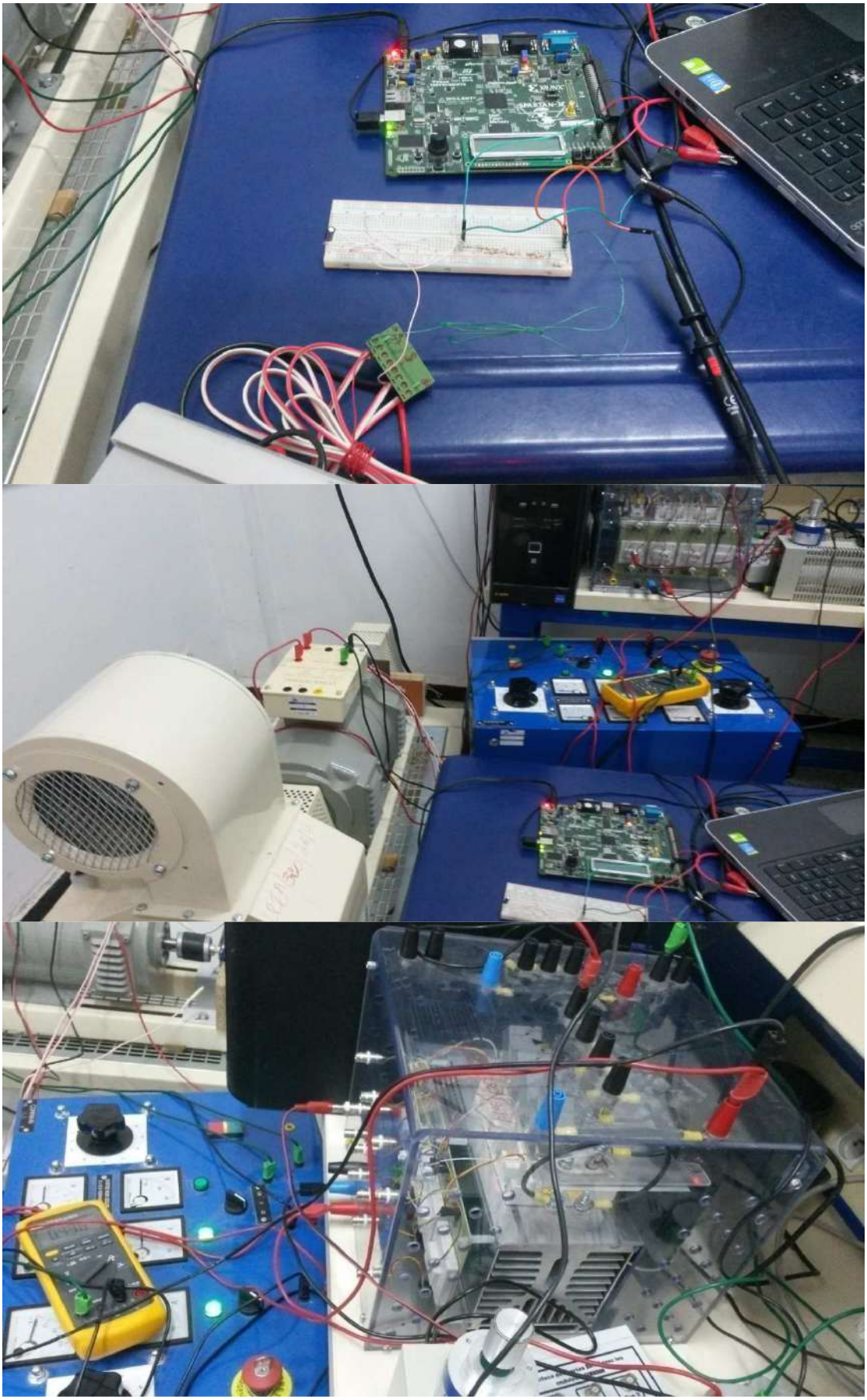


Figure.3.13: Banc de montage

3.4 Conclusion :

Dans ce chapitre nous avons implémenté une commande MLI d'un hacheur qui permet, de sa part, de commander la vitesse du moteur à courant continu. Les résultats obtenus montrent que la vitesse de rotation d'un moteur à courant continu dépend sa tension d'alimentation, la variation du rapport cyclique varie la tension ou borne de moteur à courant continu. La validation pratique a été effectuée en utilisant un circuit FPGA de type Spartane-3E.

CONCLUSIONS GÉNÉRALES

Bibliographie

[1] Michel Pinard, « Convertisseurs et Electronique de Puissance », Dunod, 2007.

[2] Luc Lasne, Notions de base et machine électrique, Dunod, Pairs, 2005.

[3] Fabrice DESCHAMP, « Cours BAC S SI – Convertir L'énergie – Machine à Courant Continu » Sciences de L'ingénieur, LYCEE JACQUES PREVERT.

[4] Lilian BOSSUET, Les circuits matériels reconfigurables – FPGA 'Les FPGA, Technologie, Architecture et utilisation', 2010

[5] Johanna MARIANI, Programmation et Utilisation du FPGA pour la validation et la vérification de Circuits électroniques, Mémoire d'ingénieur, 2011

[6] Denis RABASTE, Programmation des CPLD et FPGA en VHDL avec Quartus II, aide mémoire, IUFM d'Aix-Marseille, 2011

[7] Kamel MESSAOUDI, Cours VHDL (Informatique industrielle), Master I Electrotechnique, Université de Constantine, 2013

[8] C.ALEXANDRE, FPGA EPRM, Conservatoire National des Arts et Métiers, Support de cours 2013/2014

[9] www.elec.jbd.free.fr, « Machine électromagnétique », Chapitre 13

Résumé

L’évolution des technologies conduit à utiliser des machines nécessitant des vitesses de rotation précises et variables pour l’entraînement de TGV par exemple. L’avantage principal des machines à courant continu réside dans leur adaptation simple aux moyens permettant de régler ou de faire varier leur vitesse, leur couple et leur sens de rotation ainsi que leur raccordement direct à une source d'énergie (batteries d'accumulateur, piles, etc.) Malgré leur principal problème qui se pose au niveau de la liaison entre les balais, ou « charbons » et le collecteur rotatif, le moteur à courant continu continue d’exister à travers de nombreuses applications. Dans ce travaille, on présente plusieurs cas que ce soit étude du moteur à courant continu, puis la technique de la variation de vitesse, et enfin la simulation de notre model sur logiciel de simulation MATLAB et réalisation d’un circuit de commande avec la carte FPGA.

Mots clés : Machine à courant continu, les convertisseurs statiques, la technique MLI

Abstract

The evolution of technologies leads to use machines requiring precise and variable speeds for the TGV drive for example. The main advantage of DC machines lies in their easy adaptation to ways to adjust or vary their speed, torque and direction of rotation as well as their direct connection to a power source (accumulator, batteries, etc.) Despite their major problem at the connection between the brushes, or "coal" and the rotary collector, the DC motor continues to exist across many applications. In this work, we present several cases whether study of DC motor, then the technique of the speed variation, and finally the simulation of our model on MATLAB simulation software and implementation of a control circuit with the card FPGA.

Keywords: DC machine, static converters, PWM technique.

ملخص

إن تطور التكنولوجيا يؤدي إلى استخدام الآلات التي تتطلب سرعات دقيقة ومتغيرة على سبيل المثال قطار فائق السرعة الميزة الرئيسية لمحرك التيار المستمر تكمن في تكيفها السهل في سبل تعديل او تغير سرعتها وعزم دورنها واتجاه الدوران بالإضافة إلى الاتصال المباشر لمصدر الطاقة .

لا يزال محرك التيار المستمر يتواجد عبر العديد من التطبيقات وفي هذا العمل نقوم بالتحكم في محرك التيار المستمر ثم أساليب تغير السرعة وأخيرا محاكاة نموذجنا على برنامج المحاكاة ماتلب وتنفيذ دارة تحكم بواسطة بطاقة FPGA

الكلمات المفتاحية

ركات التيار المستمر .المحولات الثابتة .تقنية MLI