Dédicaces

A toi seigneur DIEU tout puissant créateur du ciel et de la terre. Je te remercie pour m'avoir donnée la volonté et surtout le courage de mener dans de bonnes conditions ce travail.

A celui qui s'est toujours sacrifié pour me voir réussir, A toi mon père.

A la flamme de mon cœur, ma vie et mon bonheur, maman que j'adore.

Aux personnes dont j'ai bien aimé la présence dans ce jour, a ma grande mère, à mes frères Abdelalim et Abdeldjaoued.

A ma petite sœur Aroua Rimel, sans Oublier mes oncles et mes tantes et tout les membres de la famille BRENKIA et la famille AMEUR

A tous mes amis, exception, a Mahrez, Nadir, Hakim et Hamza.

Aux collègues d'étude. Abdelaziz, Abdelkader et Nidal.

Abdessattar

Liste des tableaux

L'automatique est la science des systèmes. Cette science a une composante théorique constituée des outils de compréhension de la structure des systèmes et de leur comportement, et une composante pratique car l'automatique a pour fin de faire fonctionner des systèmes en minimisant l'intervention humaine pour éviter toutes sortes de tâches fastidieuses, répétitives ou dangereuses, et donc d'en permettre l'automatisation .En effet, l'automaticien s'intéresse essentiellement aux systèmes construits pour l'être humain (une machine a outil ,une rame de métro ,une voiture …ect), touts ces systèmes possèdent des asservissements qui permettent de réguler la position ou la vitesse .

Nous allons essayer de réaliser un système, afin de mieux comprendre comment est-ce -qu'ils marchent, notre système choisi ici est le moteur a courant continu. Ce dernier fonctionne à des vitesses variables. L'objectif ici est de garder cette vitesse stable à une valeur constante prédéterminé (la consigne). C'est-à-dire minimiser l'erreur entre la consigne et la vitesse mesurée.

Ce mémoire présente notre projet qui consiste à commander un moteur à courant continu par deux méthodes de commande (PID et Contrôleur flou). Les algorithmes de commande ont été programmés en langage LABVIEW afin de les implémenter dans le FPGA (Xilinx XC3S500E Spartan-3E).

L'algorithme proportionnel, intégral, dérivé (PID) est l'algorithme de contrôle le plus commun utilisé dans l'industrie. Un régulateur PID est un régulateur à boucle de rétroaction qu'on peut utiliser pour contrôler des processus tels que des systèmes de chauffage et de refroidissement, de surveillance des niveaux de liquides, de contrôle des débits et des pressions.

La logique floue est une méthode de prise de décision basée sur des règles utilisée dans des systèmes experts et pour le contrôle des processus. La logique floue diffère de la logique booléenne traditionnelle en ce qu'elle autorise l'appartenance partielle d'un élément à un ensemble. On peut utiliser la logique floue pour contrôler des processus représentés par des descriptions linguistiques et subjectives, et transmettre des données acquises à un régulateur flou pour prendre des décisions en temps réel ou pour contrôler un système physique. Nous pouvons également utiliser les sorties du régulateur flou avec du matériel de développement (FPGA) à sorties analogiques pour mettre en œuvre un système de contrôle des processus en temps réel.

Ce mémoire est organisé en quatre chapitres :

Dans le premier chapitre-nous présentons une généralité en commençant par la définition des circuits logiques programmables en générale et le FPGA en particulier, en suite nous avons fait une étude détaille des moteurs a courant continu.

Le deuxième chapitre est consacré à l'initiation de l'environnement de programmation graphique LabVIEW, qui est utilisé dans la partie programmation pour implémenter les algorithmes de commande.

Le troisième chapitre est réservé à l'étude d'une méthode de l'intelligence artificielle, il s'agit de la logique floue, en commençant tout d'abord par sa définition, son domaine d'application et enfin on a basculé à l'étude des contrôleurs flous.

Dans le quatrième chapitre nous citons et expliquons la carte de puissance, le circuit d'alimentation et le circuit d'isolation qui sont réalisés sous logiciel EAGL. Nous présentons et expliquons aussi les deux algorithmes implémentes dans la carte FPGA tel que l'algorithme PID et l'algorithme de logique floue. A la fin de ce chapitre on a effectué quelques expériences pour montrer les résultats obtenus.

Le travaille présenté dans ce mémoire nous a poussé à faire des recherches vaste dans plusieurs domaines tel que l'électronique en générale et l'électronique de puissance en particulier, l'asservissement des systèmes automatiques, l'intelligence artificielle, les circuits logique programmable de type FPGA ainsi les programmés via le logiciel LabVIEW. Cette recherche nous a permis d'enrichir nos connaissances et de développer notre base théorique.

Les signaux PWM sont générés par la carte électronique FPGA de Digital Electroniques, cette carte de développement se caractérise par sa fiabilité, sa simplicité et sa robustesse. L'avantage de cette carte est la possibilité de la programmée avec logiciel Xilinx ISE et avec logiciel LabVIEW.

Notre circuit de puissance est commandé par deux signaux PWM, ce circuit est réalisé a base de circuit intégré L6203 qui peut piloter un seul moteur a courant continu de 48 volt maximum et un courant qui arrive jusqu'à 5 ampère.

Nous avons montré que le régulateur PID donne des résultats assez bien, les paramètres Kp, Ki et Kd sont choisit en faisant des essayas.

La commande de notre système par la logique floue suit parfaitement l'allure de la consigne même lorsqu'on exerce une perturbation au système, le contrôleur flou suit toujours la consigne.

Les résultats obtenus nous montrent que la commande par logique floue plus performante que la commande PID.

Nous pouvons confirmer que l'objectif, fixé au départ, a pu être atteint et notre travail satisfait les exigences déterminées au début.

A l'issu de cette étude, nous avons découvert des nouvelles techniques de commande efficace, maitrisé l'implémentation des algorithmes de commande sur le FPGA et appris à mettre au point des réalisations électronique et des circuits électronique imprimé.

Pour les travaux futurs nous préconisons une étude théorique et réalisation pratique d'une autre méthode de commande en utilisant une technique d'intelligence artificielle, il s'agit des réseaux de neurones artificiels et appliquer cet algorithme au même système utilisé dans ce projet pour tester les performances des deux techniques et comparer les résultats.

Chapitre 01 Généralités

Chapitre 02 : Initiation logiciel labVIEW

Chapitre 03 La commande par logique floue

Chapitre 4 Réalisation se la commande du moteur MCC

II.1 Introduction

La Programmation est le procédé qui consiste à produire un groupe d'instructions demandant à l'ordinateur de réaliser une tâche. Cela peut être fait en utilisant différents "langages" de programmation, comme le C++ ou JAVA, et d'autre langage de programmation orienté objet, ces langages sont codés en écrivant des lignes d'instruction de haut niveau. Contrairement les langages de programmation graphiques sont codés en sélectionnant des objets en les connectant et en ajoutant des fonctionnalités, ce chapitre présentLabVIEWet ses fonctionnalités.

II.2 Définition [12]

Le logiciel LabVIEW est une "Plate-forme Expérimentale d'Instruments Virtuels pour Laboratoire" (***Lab****oratory****V****irtual* ***I****nstrument* ***E****ngineering* ***W****orkbench*). C'est un environnement de programmation disponible sur plusieurs systèmes d'exploitationcommercialisé par la société "National Instruments" et possédant un langage graphique (lelangage G), des bibliothèques de fonctions et de sous-programmes, ainsi que des outils de développement. Ce logiciel est dédié a des applications modulaires (notion de sous VI) et extensibles pour la conception d'application, a commande-contrôle des systèmes, traitement de signal et l'acquisition des données.

II.3 HISTORIQUE [13]

1983 : Développement de LabVIEW par NI

1986 : Distribution de la première version pour une plate formeMacOs

1990 : Le languagedeviantcompile

1992 : L'environnement de programmation LabVIEW est porté sur Windows

1994 : Intégration du composant « Application Builder »

1998 : Portage sous Lunix et le module pour cible F.P.G.A

2004 : Introduction de la gestion par objet, les Xcontrols

2009 : NI décide de générer une version de LabVIEW chaque année

2014 : Le logiciel est actuellement disponible sur différents systèmes d'exploitation

II.4 La programmation avec LabVIEW

II.4.1 Les fenêtres du programme [14]

Un programme de LabVIEW comprend 2 fenêtres distinctes : le **panneau avant** servant d'interface avec l'utilisateur et le **diagramme** contenant le programme source en langage graphique G.

a. **Le panneau avant:**

Cette fenêtre, où apparaissent des objets sous forme de commandes d'entrée ou **contrôleur** (*Controls*) ou d'**indicateurs** de sortie (*Indicators*), constitue l'interface interactive du programme. Le « panneau avant » vide apparaissant par défaut lors de la création d'un programme.

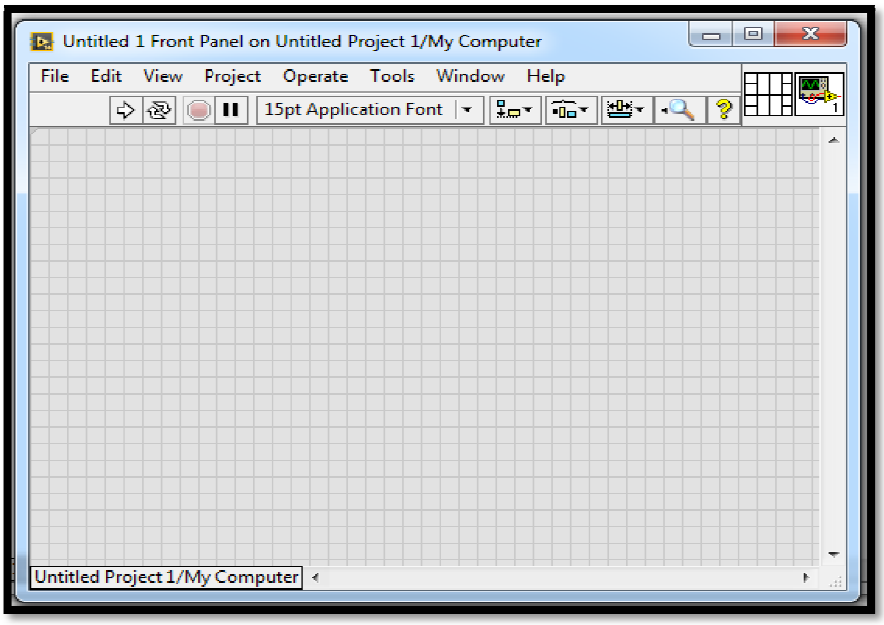


Figure 2-1. : Face avant

b. **Le diagramme**

Cette fenêtre contient le code source graphique représentant le programme écrit en langage G. Le diagramme vide apparaissant par défaut lors de la création d'un programme

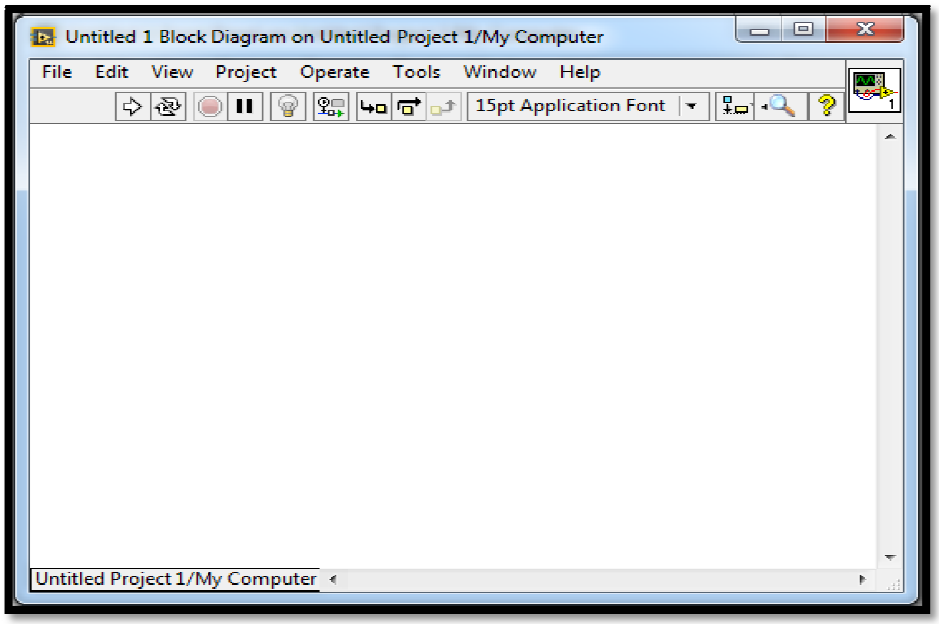


Figure 2-2: Bloc diagramme.

On passe du « panneau avant » au diagramme à l'aide du choix *Show Block Diagramme*dumenu *Window*de la barre de menus. Réciproquement, on retourne au « panneau avant « àpartir du diagramme à l'aide du choix *Show Front Panel* du menu *Window*de la barre demenu.

Chaque VI affiche une icône, dans le coin supérieur droit des fenêtres de la face-avant et du diagramme. Une icône est la représentation graphique d'un VI.

Un connecteur est un ensemble de terminaux correspondant aux commandes et aux indicateurs du VI qui sont accessibles.

II.4.2 Les caractères communs au panneau avant et au diagramme :[14]

a. Barre de titre :

Située en haut de chaque fenêtre, la barre de titre affiche le nom du programme. Ce nom est Untitled i (où i est un nombre > 0) par défaut à l'ouverture d'un nouveau programme, et le reste tant que le programme n'a pas été sauvegardé.

b. Barre de menu :

Située sous la barre de titre elle présente une série de menus déroulant communs aux deux fenêtres. Ces menus à structure hiérarchisée proposent des choix pouvant

entraîner une action immédiate, conduire à des sous-menus, ou aboutir à une fenêtre de dialogue.

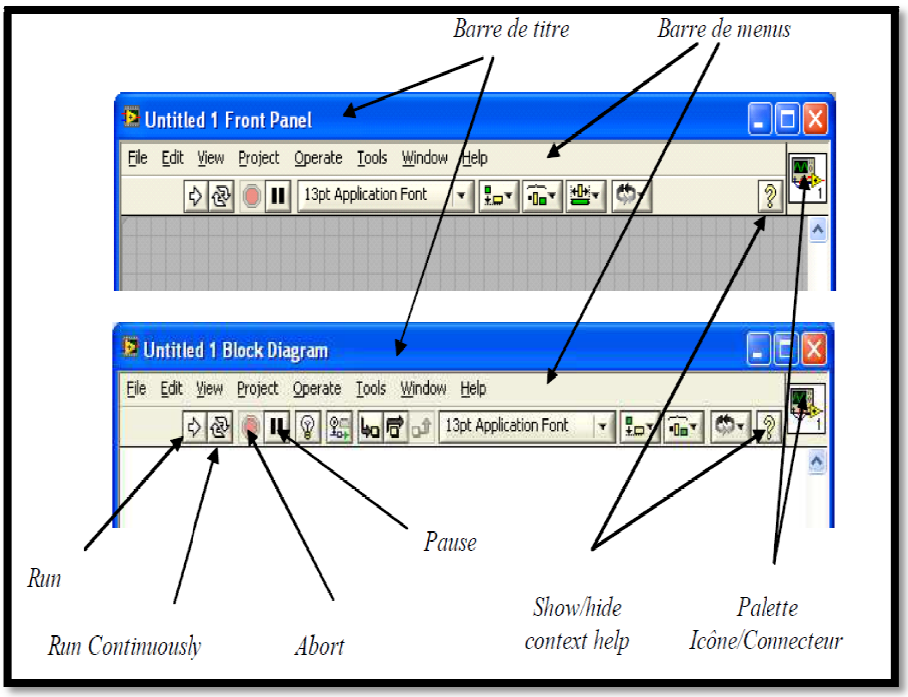


Figure 2-3 : Caractères communs aux deux fenêtres.

c. Palette de boutons de sélection :

Située juste au-dessous de la barre de menus, elle offre de nombreux boutons de sélection, dont au moins quatre boutons de commande communs aux deux fenêtres.

d. Menu surgissant :

La zone située sous la palette des boutons de sélection est spécifique au diagramme ou au « panneau avant ». Lorsqu'on clique dans cette zone avec le bouton droit de la souris, un **menu surgissant** (Pop-up Menu) ou une palette d'objet apparaît à l'endroit pointé. Si l'on a cliqué dans une zone vide, une palette apparaît (Controls ou Functions suivant la fenêtre concernée), permettant d'introduire des objets dans la fenêtre. Si l'on a cliqué sur un objet situé dans la fenêtre, le menu surgissant sur l'objet est spécifique à l'objet et propose plusieurs choix pour modifier, initialiser, … l'objet en question.

II.4.3 Les palettes :[14]

Sous l'environnement LabVIEW, le programmeur dispose d'une palette d'outils et de deux palettes d'objets, apparaissant par défaut en fonction du contexte, selon qu'on travaille sur le « panneau avant » ou sur le diagramme.

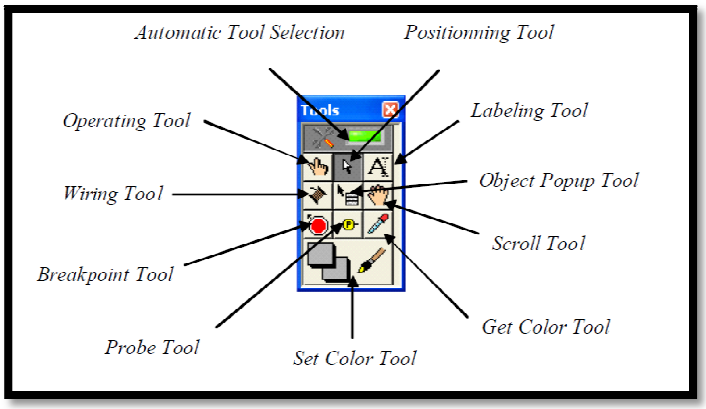


Figure 2-4 : Palette d'outils.

Une **palette d'outils** (*Tools*) utilisables aussi bien pour le diagramme que pour le «panneau avant » apparaît par défaut lorsqu'on ouvre un VI et reste néanmoins toujours accessible par le choix *Tools Palette* du menu *View*si on l'a supprimée. Elle permet de définir divers modes de fonctionnement du curseur.

a. La palette d'objets du « panneau avant » (Controls Palette)

Cette palette apparaît par défaut lorsque la fenêtre du « panneau avant » est active et reste accessible, si on l'a fermée, par le choix *Controls Palette* du menu *View*ou par le bouton droit de la souris sur le « panneau avant ».

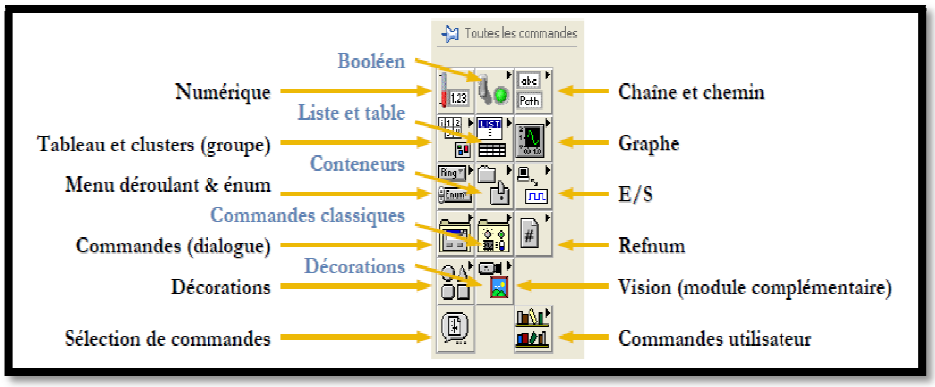


Figure 2-5 : Palette d'objets du panneau avant apparaissant par défaut .

Les principaux choix offerts par cette palette fournissent des objets représentant les entrées sorties du programme. Ils correspondent à des types prédéfinis du langage (*Numeric*, *Boolean*,*String&Path*, *List & Tables*, *Array& Matrix*, *Graph*, *Refnum, …*), à des symboles de décoration divers (*Décorations*) ou à des types définis par l'utilisateur (*Select a Control …*).

b. La palette d'objets du diagramme (Functions Palette) :

Cette palette apparaît par défaut lorsque la fenêtre du diagramme est active et reste accessible, si on l'a fermée, par le choix *Functions Palette* du menu *View*ou par le bouton de droite de la souris sur le diagramme.

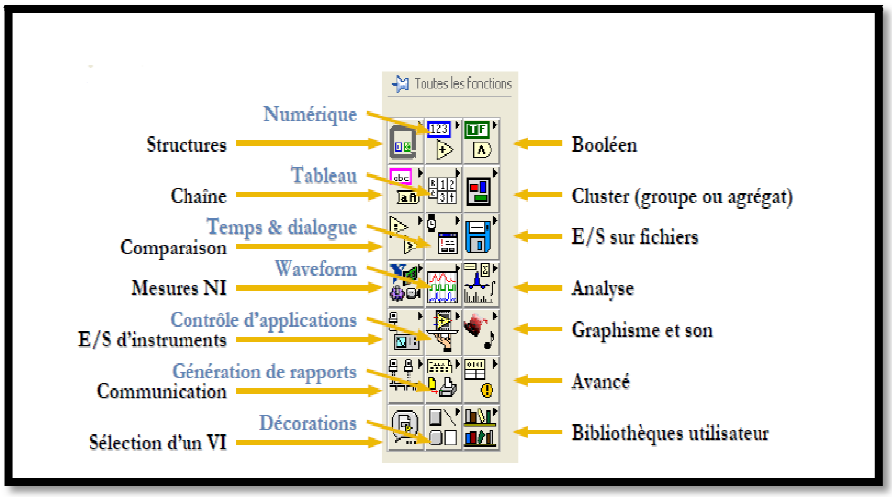


Figure 2-6: Palette d'objets du diagramme apparaissant par défaut.

Les principaux choix offerts par cette palette correspondent à des opérateurs ou des fonctions prédéfinies du langage. Sans panneau avant ni diagramme, les fonctions fournissent du code machine en ligne et sont pour la plupart **polymorphes**, c'est à dire qu'elles s'adaptent aux types (réels, entiers, tableaux de réels, tableaux d'entiers, …) et représentations des données (décimales ou hexadécimales par exemple). La plupart de ces fonctions seront détaillées dans la suite de ce chapitre.

II.4.4 Quelques raccourcis clavier :[12]

Touche clavier	Fonction
Ctrl + B	Élever tout les fils non branchés (fils brisés).
Ctrl + C	Sélectionner pour copies
Ctrl + H	Help
Ctrl + T	Affiché les deux fenêtres cote à cote
Ctrl + espace	Recherche avancée
Ctrl + N	Nouveaux fenêtres
Ctrl + E	Interaction entre face avant et diagramme

Ctrl + L	Liste des erreurs
Ctrl + R	Exécuter (Run)
Ctrl + M	Change le mode d'exécution.
Ctrl + .	Stop l'exécution.
Ctrl + E	Pour passer d'une fenêtre à l'autre.
Ctrl + T	Mosaïque verticale des fenêtres
Ctrl + Z	Annuler (aussi dans le menu Édition)
Ctrl + C	Copier un objet
Ctrl + V	Coller un objet
< Ctrl + cliquer-glisser-	sélectionner pour copies et coller

Tableau 2-1 **:** Liste de raccourcis.

II.4.5 Flux de données :[12]

L'exécution du diagramme dépend du flux de données. Il ne s'exécute pas nécessairement de gauche à droite. L'exécution du nœud se fait quand les données sont disponibles à tous les terminaux d'entrée. Puis les nœuds fournissent des données à tous les terminaux de sortie.

II.4.6 Aide (Help)[12]

Deux possibilités d'aide en ligne :

La fenêtre « d'aide contextuelle » : menu *Aide – Afficher l'aide contextuelle* ou*Ctrl-H :* si cette aide est active, il suffit de placer le curseur sur un élément pour avoir une aide simplifiée.

Pour une aide plus complète :

Menu Aide – référence en ligne (et notamment LabVIEWRéférence – Functions&VIs), ou cliquer sur « aide détaillée » dans la fenêtre d'aide contextuelle.

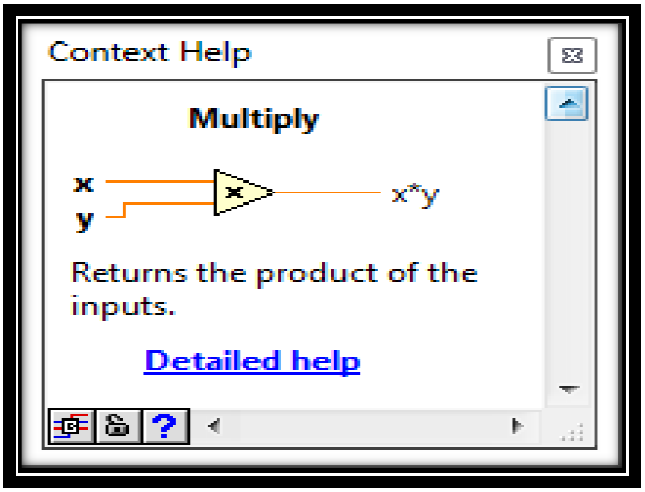


Figure 2-7 :Aide (Help).

II.5 Les principaux objets de LabVIEW [14]

II.5.1 Les objets présents sur le panneau avant et sur le diagramme [14]

a. Les nombre :

☑ **Sur le panneau avant :**

Le contrôleur/indicateur digital (Digital Control/Digital Indicator), accessible dans la sous palette Numeric de la palette Controls, est l'objet fondamental pour la manipulation des nombres. Il se place sur le panneau avant et se présente à l'écran sous forme d'une fenêtre rectangulaire dans laquelle s'affichent les chiffres constituant le nombre, avec une palette d'incrémentation/décrémentation lorsqu'il s'agit d'un contrôleur (un indicateur n'aura évidemment pas cette palette d'incrémentation/décrémentation).

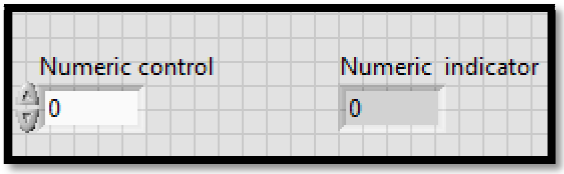


Figure 2-8 : Représentation des variables numériques sur la face avant.

☑ **Sur le diagramme :**

Sur le diagramme, la terminaison associée à un contrôleur digital (ou un contrôleur pseudo analogique) est un rectangle à contour épais. Ce rectangle est à contour mince pour un indicateur. Dans les deux cas, il contient par défaut les trois caractères « DBL », caractéristiques d'un nombre réel en double précision.

Figure 2-9 : Représentation des variables numériques sur le diagramme.

b. Représentation en mémoire :[14]

Lorsqu'on ne veut pas de la représentation par défaut, l'option spécifique représentation du menu surgissant (accessible sur le panneau avant et sur le diagramme) permet de modifier la représentation du nombre en mémoire. Les principales représentations sont

	Symbole	Type de données	octet	Gamme
Les nombres entiers	I8	*Integer*	1	-128... +127
	I16	*Integer*	2	-32768... +32767
	I32	Integer	4	-2147483648..+2147483647
	U8	*Unsigned Integer*	1	0.. 255
	U16	*Unsigned Integer*	2	0.. 65535
	U32	*Unsigned Integer*	4	0.. 4294967295
Les nombresréeles	SGL	Single Précision Real	4	7
	DBL	Double PrécisionReal	8	15

Tableau 2-2 : liste de type de donnée.

c. Les booléens :

LabVIEW fournit des contrôleurs logiques, ou **booléens**, permettant à l'utilisateur de contrôler le déroulement du programme en imposant leur état (via une commande) ou en le recevant (via un indicateur). En mode édition, la sous-palette *Boolean*de la palette *Controls*permet d'accéder à ces contrôleurs booléens.

Les booléens peuvent prendre 2 états logiques, Vraie (*True*) ou Faux (*False*). L'état par défaut est Faux (*False*), mais ceci peut être modifié à l'aide du bouton de manœuvre (*OperatingTool*), puis du choix *MakeCurrent Value Default* de l'option *Data Operations* du menu surgissant sur l'objet.

☑ **La constantbooléenne :**

Elle n'est accessible que sur le diagramme par le choix Boolean de la palette Functions. La valeur de la constante (TRUE ou FALSE) peut être modifiée à l'aide de l'Operating Tool de la palette d'outils.

d. Les chaînes de caractères :

Les contrôleurs et indicateurs alphanumériques, manipulant le texte par chaînes de caractère (*String*), constituent un type de données de base de LabVIEW, à la différence de nombreux langages où le seul type rencontré est le caractère.

☑ **Sur le panneauavant**

En mode édition, la sous-palette String &Path de la palette Controls propose un contrôleur d'entrée de chaîne de caractères (String Control) ainsi qu'un indicateur de sortie (String indicator).

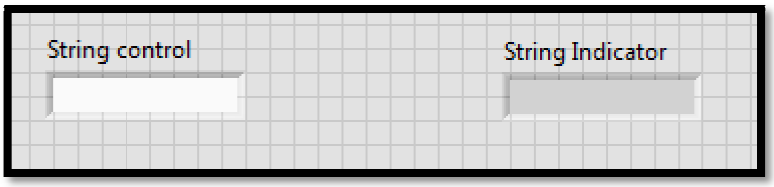


Figure 2-10 :Représentation de chaînes de caractères sur la face avant.

☑ **Sur le diagramme :**

La terminaison du diagramme associée à une chaîne de caractères est un rectangle rose, à contour épais (pour un contrôleur) ou mince (pour un indicateur), contenant les trois caractères « abc », indiquant le type chaîne de caractères.

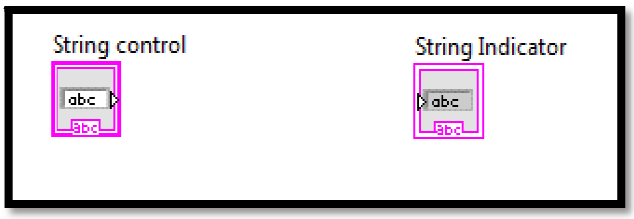


Figure 2-11 : Représentation de chaînes de caractères sur le diagramme.

Les constantes alphanumériques sont accessibles sur le diagramme par le choix String de la palette Functions.

e. Les graphiques

LabVIEW, dans son langage graphique G, offre à l'utilisateur des objets extrêmement performants, correspondant à des types de données numériques inexistants dans les autres langages, permettant l'affichage et la manipulation aisée des graphiques.

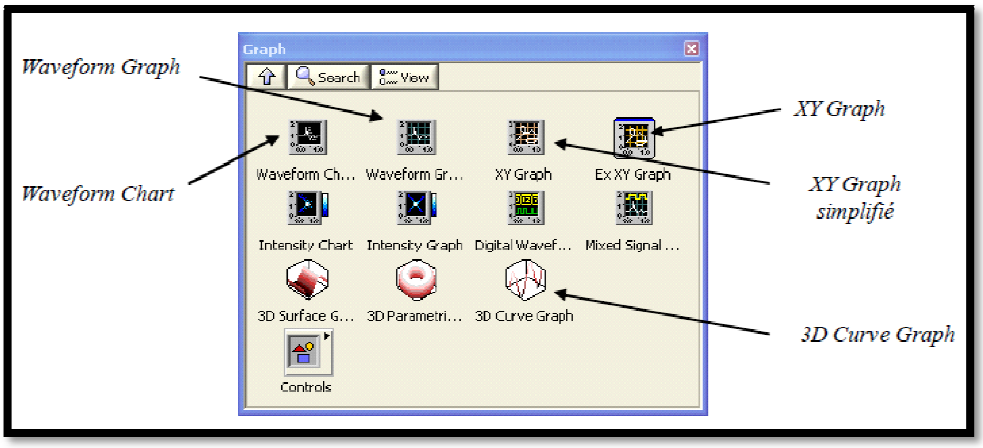


Figure 2-12 :Choix *Graph* de la palette *Controls.*

☑ **Sur le panneauavant :**

Les données reçues sont placées dans une mémoire tampon, associée au graphique (donc au programme dont le panneau avant contient l'afficheur). Une mise à jour de l'affichage **avectracé complet du contenu de la mémoire tampon** s'effectue à chaque ajout d'une nouvelle donnée (d'où une certaine lenteur). La taille de la mémoire tampon (1024 points par défaut) peut être modifiée par le choix ChartHistoryLength du menu surgissant.

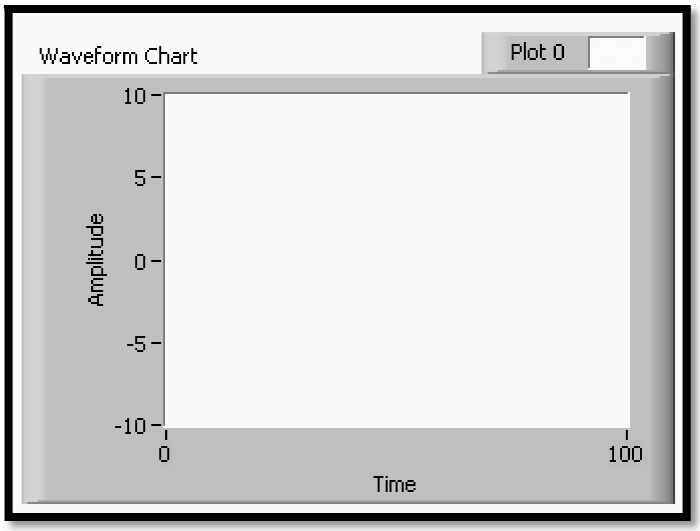


Figure 2-13 : Un *WaveformChart*sur le « panneau avant ».

☑ **Sur le diagramme**

La terminaison associée à l'afficheur graphique sur le diagramme est un rectangle dont l'intérieur va refléter le type des données fournies au graphique par le programme : nombre (entier ou réel), tableau de nombres.

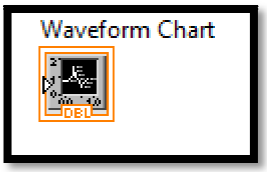


Figure 2-14 : Un *WaveformChart*sur le diagramme.

II.5.2 Boucles et structures [12]

a. Les boucles

a.1. La boucle inconditionnelle POUR (For Loop)

C'est une structure itérative permettant d'exécuter une partie de programme un nombre déterminé de fois. On l'utilise lorsque l'on connaît le nombre d'itérations à exécuter.

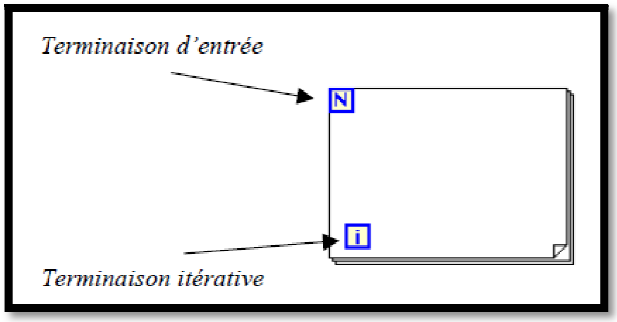


Figure 2-15 : Terminaison de la boucle *For.*

a.2. La boucle conditionnelle TANT QUE (WhileLoop)

C'est une structure permettant d'exécuter une partie de programme tant qu'une condition reste vraie. Elle correspond à la boucle *do … while (condition)* du Pascal par exemple. On l'utilise par exemple lorsqu'on effectue des mesures, sans savoir a priori quand on doit s'arrêter.

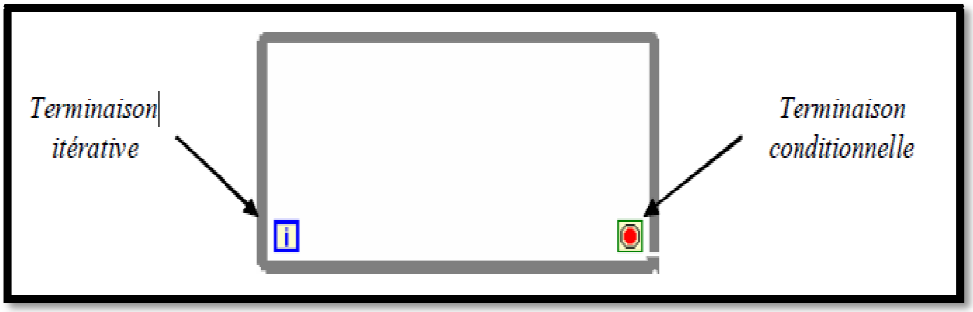


Figure 2-16 : Terminaison de la boucle *While.*

b. Les structures :

b.1. Le *nœud* pour formules (Formula Node)

La programmation graphique du langage G se prête mal aux calculs faisant de nombreux appels à des fonctions élémentaires. C'est pourquoi le langage G propose le **nœud pourformules** (*Formula Node*) permettant d'exécuter des calculs présentés sous la forme d'instructions écrites en C.

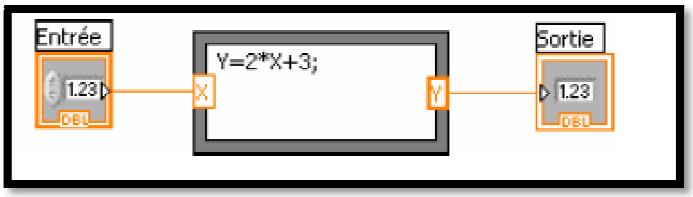


Figure 2-17 : Terminaison du Formula Node.

b.2. La structure séquentielle (StackedSequence Structure)

On est parfois amené à exécuter une partie d'un programme avant une autre (par exemple pour initialiser une carte d'acquisition avant de l'utiliser). Pour cela, la structure séquentielle permet de définir un ordre dans le déroulement du programme. On y accède par le choix *Sequence*de la palette *Structures* Elle est formée d'un cadre ressemblant à de la pellicule photographique que l'on étire de la même façon que le *Formula Node*décrit précédemment.

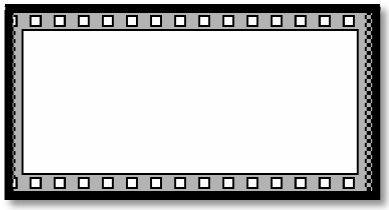


Figure 2-18 : Terminaison de la structure *StackedSequence.*

L'intérieur du cadre est un sous-diagramme destiné à recevoir la partie du programme devant s'exécuter dans la séquence. On ajoute les séquences successives par le choix *Add FrameBefore/After*ou *Duplicate Frame* du menu surgissant sur le bord du cadre. Dès que la structure comporte plus d'une séquence, la partie supérieur du cadre porte un rectangle d'index contenant le numéro de la séquence affichée (0 pour la 1ère séquence).

b.3. La structure de choix (Case Structure)

Cette structure est l'équivalent en langage G des structures *if(condition) then .. else... end if* en Fortran ou *if(condition){..}else {..}*en C, ainsi que de la structure *case* en C. On est amené à l'utiliser lorsque le programme dépend d'une condition.

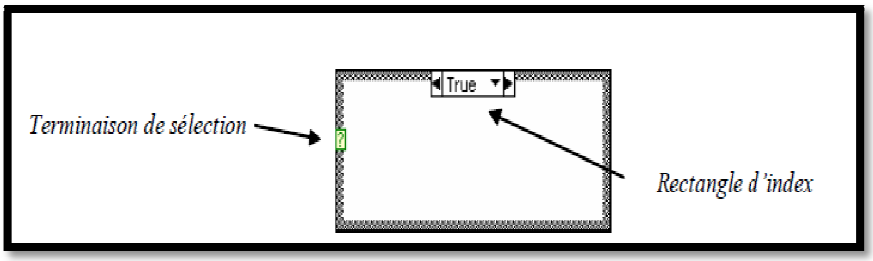


Figure 2-19 : Terminaison de la structure *Case.*

b.4. Matlab Script :

Exécute les scripts LabVIEWMatlab Script. La syntaxe de LabVIEWMatlab Script est similaire à celle du langage MATLAB.

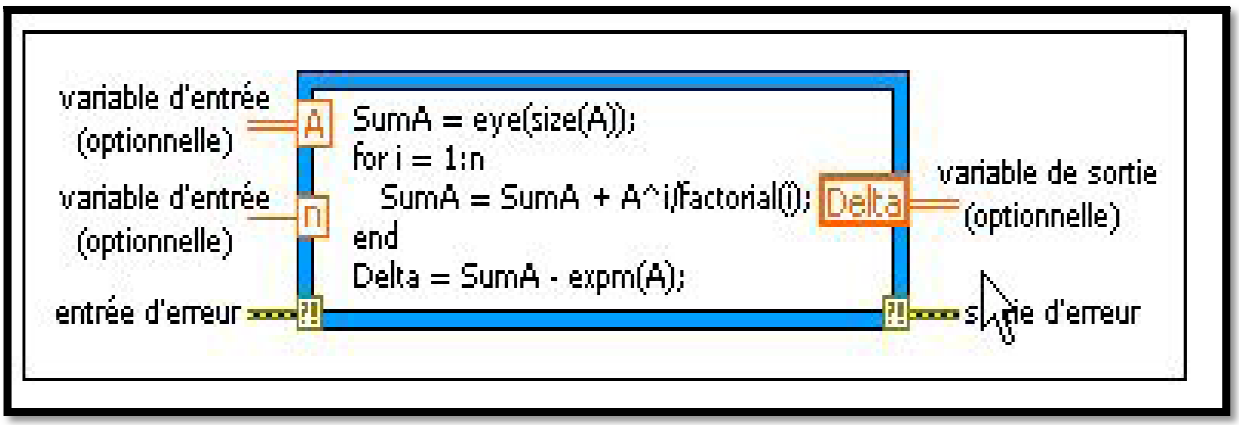


Figure 2-20 : MATLAB script.

II.6 Tunnels [15]

Les tunnels transfèrent les données dans et hors des structures. Les données sortent d'une boucle après la fin de la boucle.

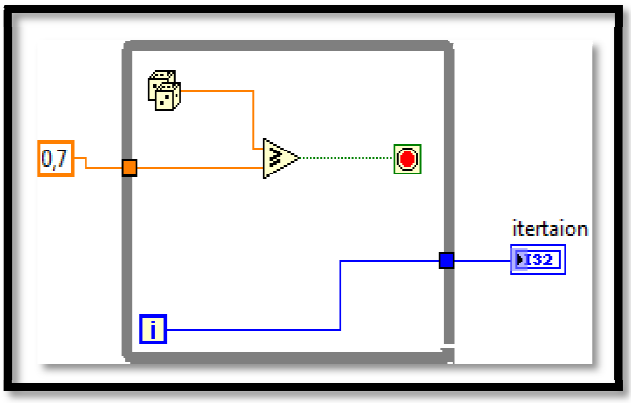


Figure 2-21 : Tunnels.

II.7 Temporisateurs (Timing) [15]

Lorsqu'une boucle termine d'exécuter une itération, elle commence immédiatement à exécuter l'itération suivante, sauf si elle attend une condition d'arrêt. Si vous acquérez des données et que vous souhaitez acquérir les données une fois toutes les 10 secondes, vous avez besoin d'un moyen de chronométrer les itérations de boucle de sorte qu'elles se produisent toutes les 10 secondes. Vous voulez également chronométrer une boucle pour fournir au processeur le temps d'accomplir d'autres tâches, telles que le traitement de l'interface utilisateur.

Attendre jusqu'à ms suivant Multiple Wait Until Next ms Multiple	**Contrôle un compteur milliseconde et attend que le compteur milliseconde atteigne un multiple de la quantité spécifiée. Cettefunctionestsynchronisée avec l'horloge du système.**
Attendre (ms) Wait (ms)	**Attend jusqu'à ce que le compteur milliseconde compte à un montant égal à l'entrée que vous spécifiez.**

Attendre jusqu'à ms suivant Multiple 	**Similaire à la fonction Attendre (ms) avec l'ajout de clusters intégrés.**

Tableau 2-3 : List du différent type de Temporisateur.

II.8 Modularité (Sous VI) [15]

II.8.1 Définition [15]

Lorsqu'on utilise souvent certains VIpersonnels, de les avoir toujoursàportée de main. On pourrait ainsi les placers dans la palette de fonctions.La façon la plus simple consiste à faire apparaître son VIdans la palette« Bibliothèques utilisateur».Un sous VI est un VI utilisé dans un autre VI. Il ressemble à un sous programme (routine) dans les langages textuels.

II.8.2 Etapes de création d'un sous VI [15]

Pour créer un Sous VI il faut suivre les 4 étapes suivantes :

1. Editerl'icône.
2. Choisir le modèle des connecteurs (Pattern).
3. Affectation des connecteurs aux contrôles et indicateurs.
4. Documentation du sous VI.
5. Enregistriez le VI dans Bibliothèques utilisateur.

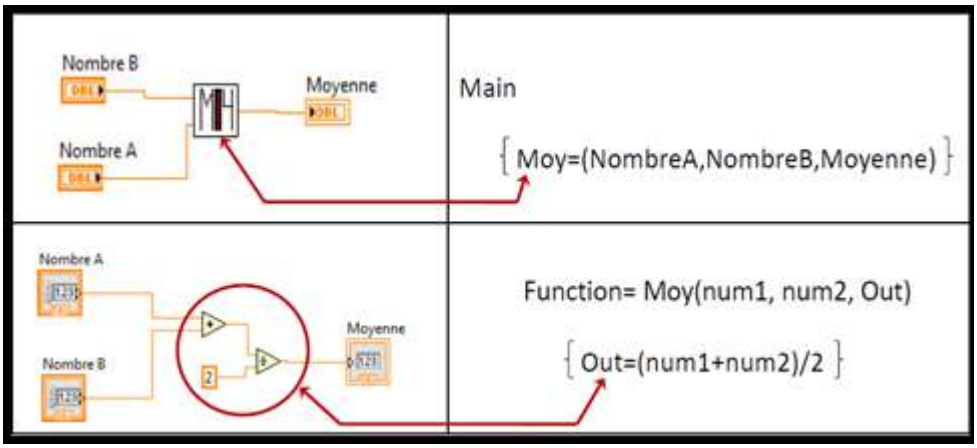


Figure 2-22: sous VI.

II.8.3 Enregistrement le sous VI [15]

- ✓ Après les quatre étapes de création, il suffit d'enregistrer son VI dans le dossier « **user.lib** » de LabVIEW (LabVIEW se trouve généralement, sous Windows, à l'emplacement :

  « C:\Program Files \ National Instruments \ LabVIEW \usir.lib »

- ✓ LabVIEW doit être redémarré pour que la palette soit mise à jour, et présente les VI présents dans ce dossier.
- ✓ Pour ouvre le sous VI enregistré dans la *Bibliothèques utilisateur.* Un clic droit dans la page diagramme :**Functions → User libraires.**

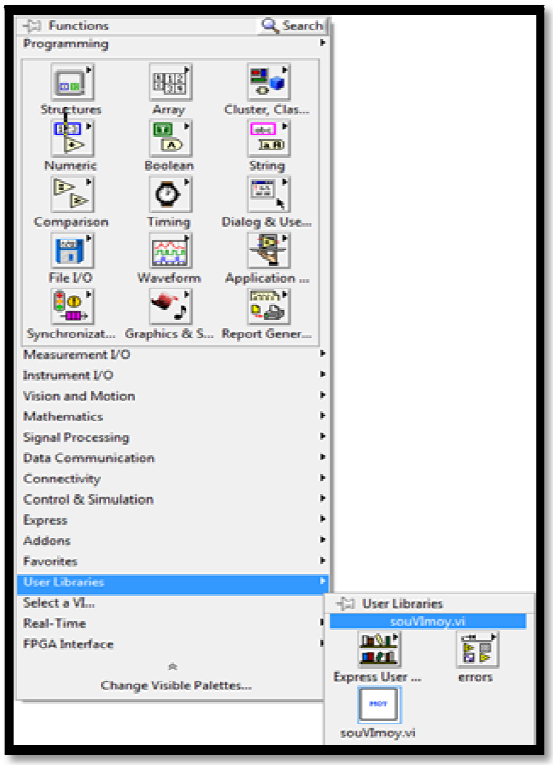


Figure 2-23: Enregistrement du sous VI.

II.9 LA DIFFERENCE ENTRE LABVIEW ET MATLAB [16]

1. Les développeurs de LabVIEW sont NI, alors que les développeurs de MATLAB sont Mathworks.

2.LabVIEW est principalement un langage de programmation graphique pour la visualisation et le réglage des paramètres pour les opérations à distance. MATLAB est un langage informatique pour le développement et la simulation d'algorithmes de contrôle.

3.LabVIEW prend en charge Windows, Linux, MacOs et multiplateforme, tandis queMATLAB prend en charge Windows, Linux et MacOs.

4. MATLAB fournit principalement un environnement informatique mathématique / numérique, tandis que LabVIEW est une plate-forme de conception de système permettant l'acquisition de données, l'automatisation de tests, le contrôle d'instruments et la conception de systèmes intégrés.

5. MATLAB est utilisé pour les simulations en raison de bibliothèques supplémentaires contenant des fonctions de niveau supérieur. LabVIEW est utilisé

lorsqu'une interface graphique fonctionnelle et intuitive ou une interaction avec du matériel est requise.

6. Les extensions de fichier pour LabVIEW sont .vi et MATLAB, 7. 7. Les dernières versions disponibles de LabVIEW sont LabVIEW 2017 SP1 et LabVIEW NXG 2.0, publiées en janvier 2018. La dernière version disponible de MATLAB est matlabR2018a, publiée le 15 mars 2018.

II.10 Conclusion :

Dans ce chapitre nous avons présenté le langage de programmation graphique LabVIEW,Comprendre les avantages de la programmation graphique, une comparaison entre LabVIEW et son concurrent MATLAB a été faite dans ce chapitre.Le prochain chapitre est réservé à l'étude de la logique floue.

III.1. Introduction

Les nombreuses applications de la logique floue qui ont été réalisées dans le monde entier ont prouvé son efficacité pour résoudre divers types de problèmes dans lesquels les connaissances disponibles sont imparfaites. Elle apparaît maintenant comme un outil dont tout développeur de systèmes doit avoir connaissance pour l'utiliser lorsque les conditions le demandent.

Nous présentons dans ce chapitre les notions de bases relatives à la commande par logique floue.

III.2. Définition [17]

La théorie de la logique floue a été développée par Lotfi Zadeh en 1960, c'est une méthodologie de calcule basée non pas sur des valeurs numériques mais sur des variables linguistiques, ces derniers appartenant au langage humain. Cette théorie à des applications dans plusieurs domaines les plus populaires sont dans le domaine d'automatique pour la commande.

III.3. Historique [18]

L'origine de la logique floue remonte au début des années 60 depuis que le professeur Lotfi Zadeh de l'université de Berkeley aux USA, A introduit le concept de sous-ensembles flous. Par la suite, en 1974, Mamdani introduisait la commande floue pour la régulation de processus industriel « moteur à vapeur». La vraie révolution c'est au Japon où la recherche n'est pas seulement théorique mais également très applicative, que la logique floue connaît son véritable essor. Dans l'industrie, le traitement des eaux, les grues portuaires, les métros, les systèmes de ventilation et de climatisation sont touchés. Enfin, des applications existent dans des domaines très différents tels que la finance ou le diagnostic médical. A partir de 1990, c'est en Allemagne que des applications apparaissent en grand nombre ainsi qu'à une moindre échelle aux USA.

III.4. L'utilisation de la logique floue [19]

La logique floue est une technique de résolution de problèmes très puissants avec une large applicabilité dans le control et la prise de décision. Elle est très utile lorsque le modèle mathématique du problème à traiter n'existe pas ou existe mais difficile à implémenter, ou il est trop complexe pour être évaluer assez rapidement pour des opérations en temps réel.

Ou bien lorsque des experts humains sont disponibles pour fournir des descriptions subjectives du comportement du système avec des termes en langage naturel. La logique floue est aussi supposée de travailler dans les situations où il y a de large incertitude et des variations inconnues dans les paramètres et la structure du système.

III.5. Domain d'application [18]

La logique floue est intéressante dans tous les domaines, ou un flou persiste. La liste proposée reste limitée, car les domaines d'applications sont multiples :

☑ Automatisme

Les automatismes ont constitué le domaine d'application par excellence de la logique floue car c'est là ou existe le grand nombre d'applications. On peut classer ses applications par domaines d'utilisation.

- ✓ Automatismes industriels
- ✓ Automatismes d'entreprise et de transport
- ✓ Automatismes dans les appareils grand public

☑ Informatique

De nombreuses disciplines informatiques utilisent également la logique floue.

- ✓ Intelligence artificielle et systèmes experts
- ✓ Programmation et développement

☑ Mathématiques appliquée

- ✓ Statistiques
- ✓ Recherche opérationnelle
- ✓ Reconnaissance des formes

III.6. Les conceptions de bases

III.6.1. Les variables et les valeurs linguistiques [17]

La théorie de la logique floue est basée sur la notion des variables linguistique prenant des valeurs linguistique ou floues. Une variable linguistique représente un état dans le système à régler ou une variable de réglage dans un contrôleur flou, chaque valeur constitue un ensemble flou de l'univers de discours.

Une variable linguistique est généralement caractérisée par :

- ➢ Un nom de la variable linguistique.

- Un ensemble des valeurs linguistique qui peut prendre la variable linguistique
- L'univers de discours, c'est-à-dire l'ensemble des valeurs numérique sur les quelles est définit.

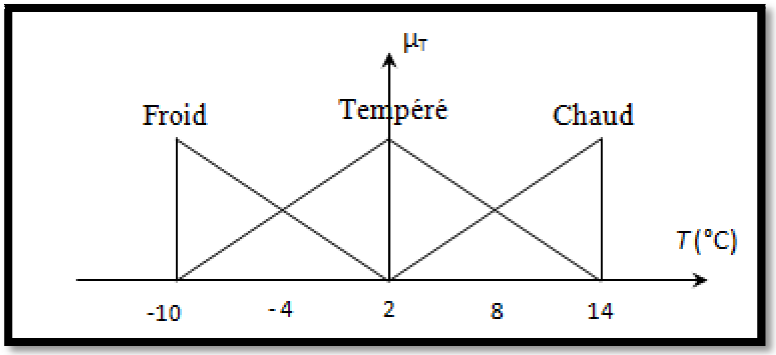


Figure 3-1 : Représentation des variables et des valeurs linguistique.

- Univers de discours : Gamme de température de -10°C à 14°C.
- Variable linguistique : La température.
- Valeurs linguistiques : « Chaud » « Tempéré » « Froid ».

III.6.2. Univers de discours [17]

C'est l'ensemble des valeurs réelles «l'intervalle numérique » qui prendre la variable floue, en générale l'univers de discours est noté par des lettres en majuscule ***U***, ***V*** et ***W***. Les valeurs linguistique seront alors projetées dans l'univers de discours pours définir le sous-ensemble associe, chaque valeur linguistique consiste a un ensemble flou de l'univers de discours.

III.6.3. Fonction d'appartenance [20]

Également appelée fonction indicatrice ou encore fonction caractéristique, On représente les variables linguistiques par leurs fonctions d'appartenances. Donc à chaque sous-ensemble flou ***A*** est associé une fonction d'appartenance $\boldsymbol{\mu_A(x)}$ où ***x*** est la variable linguistique. Tel que, à chaque point ***x*** est associé une valeur précise de $\boldsymbol{\mu_A(x)}$ qui désigne, le degré d'appartenance de ***x*** à la fonction d'appartenance peut être représentée par plusieurs formes : *Triangulaire*, *trapézoïdale*, *sigmoïdale* ou *gaussienne*. On peut définir d'autres formes de fonctions d'appartenance, mais dans le réglage par logique floue, les formes déjà citées et illustrées sur le tableau 3-1 sont largement suffisantes pour délimiter les ensembles flous.

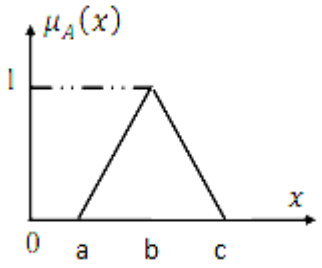
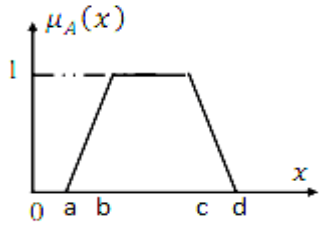
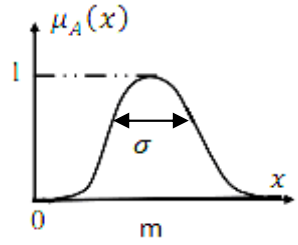
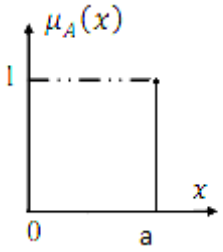
Fonction	Forme algébrique	Forme graphique
Fonction triangulaire	$\mu_A(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x < b \\ \frac{c-x}{c-b} & b \le x \le c \\ 0 & x > c \end{cases}$	
Fonction trapézoïdale	$\mu_A(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x < b \\ 1 & b \le x < c \\ \frac{d-x}{d-c} & c \le x \le d \\ 0 & x > d \end{cases}$	
Fonction gaussien	$\mu_A(x) = \exp\left(-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right)$	
Fonction singleton	$\mu_A(x) \begin{cases} 1 & x = a \\ 0 & x \neq a \end{cases}$	

Tableau 3-1 : Représentation des fonctions d'appartenance.

Le choix des formes des fonctions d'appartenance est arbitraire. Des études comparatives ont montré qu'avec les différentes formes des fonctions d'appartenance, les résultats sont pratiquement similaires en boucle fermée. La forme la plus fréquemment utilisée en commande floue est la forme triangulaire et la forme Trapézoïdale. Le nombre de

fonctions d'appartenance est généralement impair et se répartissent autour de zéro. En général, on introduit pour une variable linguistique trois, cinq ou sept ensembles flous. Le choix du nombre dépend de la précision souhaitée. Les fonctions d'appartenance peuvent être symétriques, non symétriques.

III.7. Les opérations de la logique floue [17]

Comme dans le cas des ensembles «classiques», les opérations logiques d'union (OU), d'intersection (ET) et de complémentation (NON) peuvent être appliquées aux ensembles flous. Leur définition n'est pas unique. Les opérations d'union, d'intersection et de complémentation dans les ensembles flous sont définies à l'aide de leurs fonctions d'appartenance.

Soit deux ensembles flou ***A*** et ***B*** définies sur des univers de discours ***V*** et ***W***, avec une fonction d'appartenance $\boldsymbol{\mu_A}$ et $\boldsymbol{\mu_B}$ respectivement.

III.7.1. Union de l'ensemble flou (Union flou)

L'union de ***A*** et ***B***, que l'on note $\boldsymbol{A \cup B}$, est le sous-ensemble flou constitué des éléments de ***X*** affectés du plus grand des deux degrés d'appartenance $\boldsymbol{\mu_A}$, $\boldsymbol{\mu_B}$.

$$\mu_{A\cup B} = \max(\mu_A, \mu_B) \tag{3.1}$$

III.7.2. L'intersection floue

L'intersection de ***A*** et ***B***, que l'on note $\boldsymbol{A \cap B}$, est le sous-ensemble flou constitué des éléments de ***X*** affectés du plus petit des deux degrés d'appartenance$\boldsymbol{\mu_A}$, $\boldsymbol{\mu_B}$.

$$\mu_{A\cap B} = min\ (\mu_A, \mu_B) \tag{3.2}$$

III.7.3. Le complément flou

Le complément de ***A***, que l'on note $\overline{A}$, est le sous-ensemble flou de ***X*** constitué des éléments ***x*** lui appartenant d'autant plus qu'ils appartiennent peu à ***A***:

$$\mu_{\bar{A}} = 1 - \mu_A \tag{3.3}$$

Exemple : Supposons que ***A*** (bleu) et ***B*** (rouge) sont deux sous-ensembles flous définis dans un univers de discours ***X*** par les fonctions d'appartenance $\boldsymbol{\mu_A}$ et $\boldsymbol{\mu_B}$.

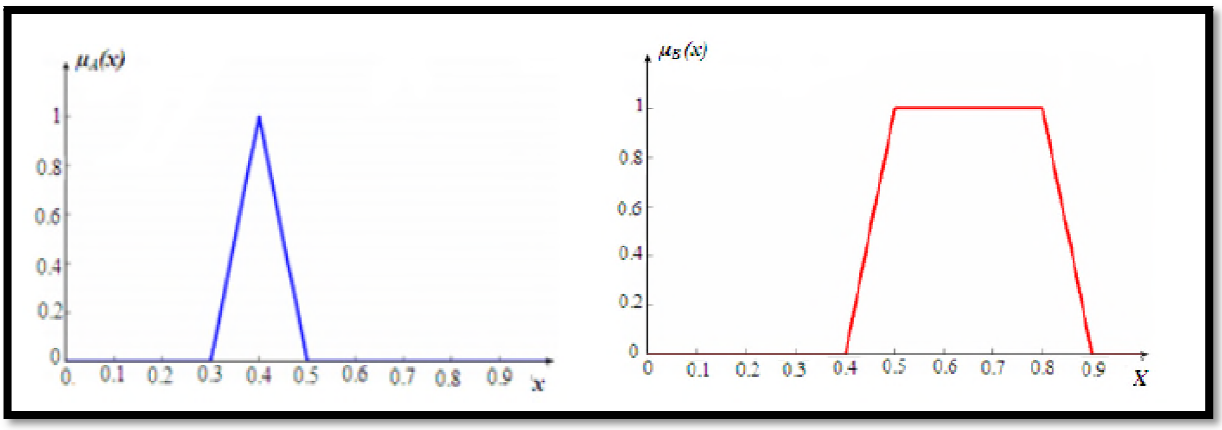


Figure 3-2 : Fonction d'appartenance de A et B.

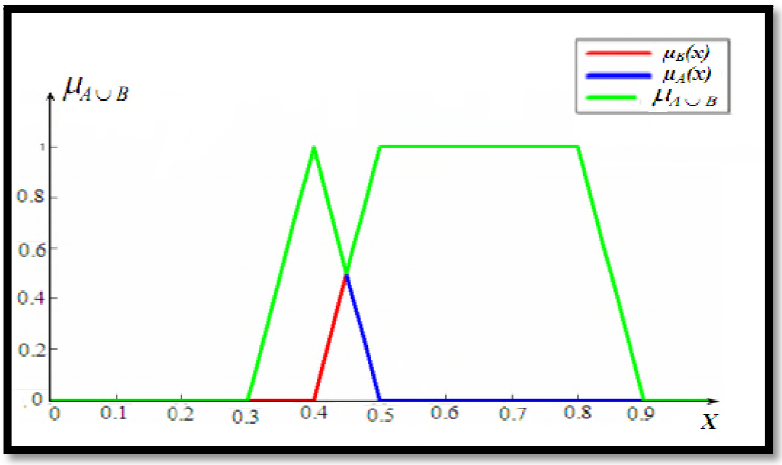


Figure 3-3 : Fonction d'appartenance de l'union.

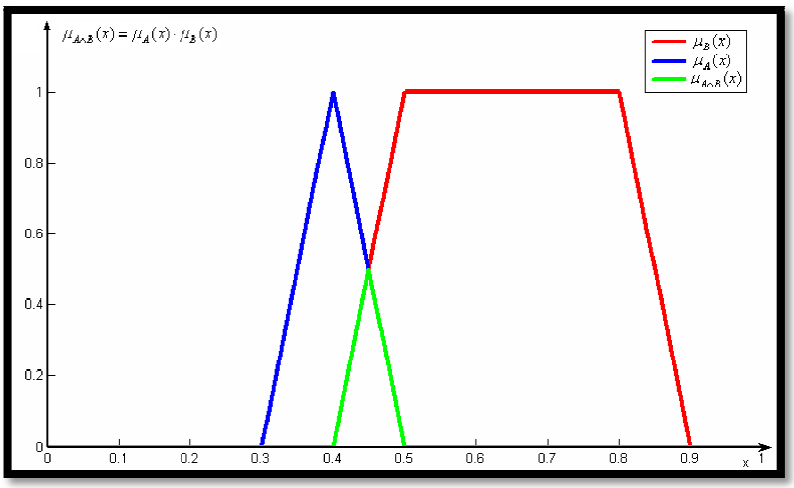


Figure 3-4 : Fonction d'appartenance d'intersection.

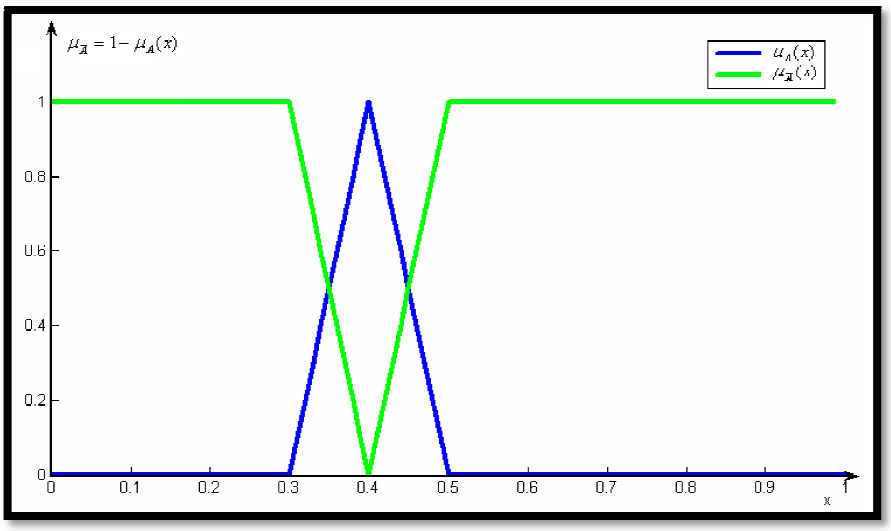


Figure 3-5 : Opérateur de complémentation de la fonction d'appartenance ***A.***

III.8. Règles linguistiques [21]

L'idée principale des systèmes basés sur la logique floue et d'exprimer la connaissance humaine sous la forme de règles linguistiques de forme ***si***, ***alors***. Chaque règle à deux parties :

1. Partie antécédente (prémisse ou condition), exprimée par ***Si***…
2. La partie conséquente (conclusion), exprimée par ***alors***…

La Partie antécédente est la description de l'état du système. La partie conséquente exprime l'action que l'opérateur qui contrôle le système doit exécuter. Chaque règle floue est basée sur l'implication floue. Zadeh a été le premier à introduire la notion de règle floue sous la forme :

Règle : Si ***x*** est ***A*** alors ***y*** est ***B***

Si (un ensemble de conditions est satisfait) **alors** (un ensemble de conséquences peut être exécuté).

III.9. Le contrôleur flou [18]

Lorsque le nouveau concept de la logique floue a été proposé, ses premières applications étaient dans le domaine du contrôle des systèmes fait des experts humains. Le contrôle de ces systèmes fait apparaitre deux types d'information :

- ☑ Des informations numériques obtenues par les mesures des capteurs.
- ☑ Des informations linguistiques obtenues par les experts humains.

Le contrôle flou utilise la logique floue comme une démarche qui peut couvrir la stratégie du contrôle linguistique. Il est intégré dans la partie qui gère les données de commande et de contrôle de la partie opérative du système, appelée contrôleur.

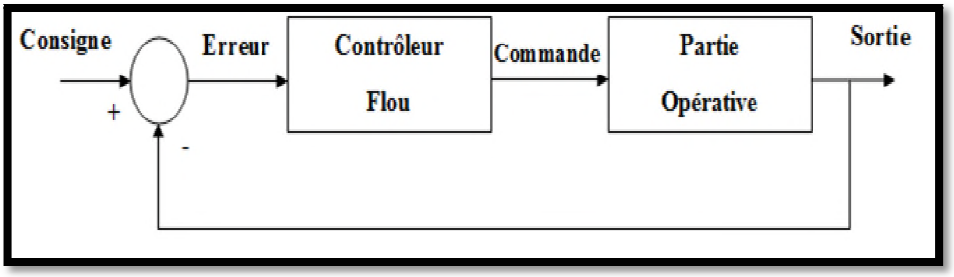


Figure 3-6 : Système à commande floue.

III.9.1. Structure d'un contrôleur flou [18]

La logique floue est souvent associée à la notion de régulateur et plus particulièrement de régulateur flou (en anglais Fuzzy Logic Controller ou FLC). Le modèle flou permet de relier à la fois les observations scientifiques et l'expérience de l'opérateur à des représentations symboliques et qualitatives. Un contrôleur flou est composé de quatre blocs principaux qui sont représenté dans la **figure 3-7.**

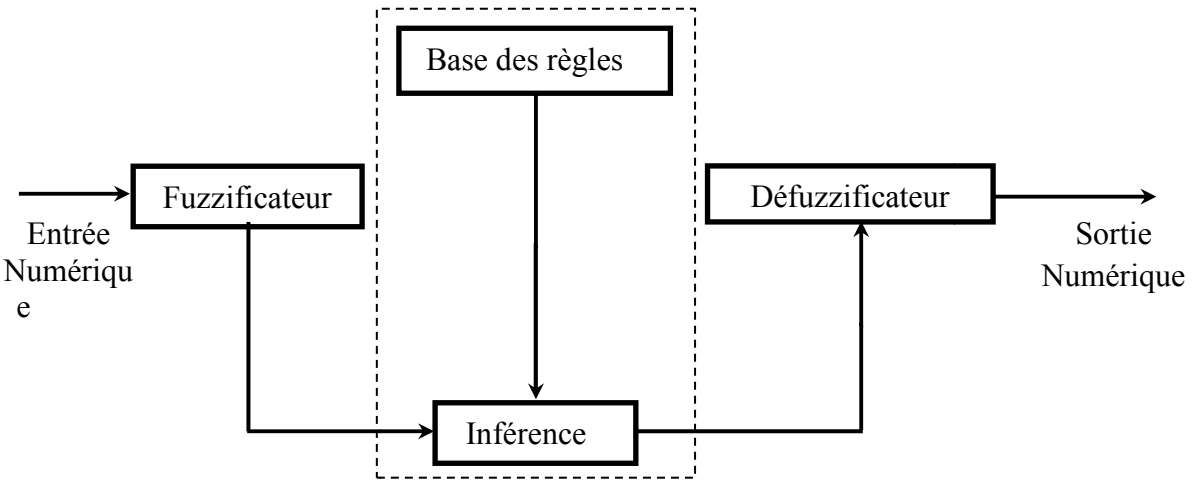


Figure 3-7 : Architecture d'un contrôleur par logique floue.

a. Fuzzification[20]

C'est le mécanisme réalisant l'interface ''Numérique-linguistique''. Les variables d'entrée et de sortie choisies pour modéliser ou commander un système sont des grandeurs numériques. L'étape de fuzzification consiste à transformer ces grandeurs réelles en variables linguistiques en vue d'un traitement d'inférence. Ainsi, à chaque variable d'entrée et de sortie est associé des ensembles caractérisant les termes linguistiques pris par ces variables.

Ces termes seront utilisés pour écrire les règles d'inférence. Le bloc de fuzzification effectue les fonctions suivantes :

- Définition des fonctions d'appartenance de toutes les variables d'entrées.
- Transformation des grandeurs physiques (réelles ou numériques) à des grandeurs linguistiques ou floues.
- Représentation d'échelle transférant la plage des variables d'entrées aux univers de discours correspondants.

b. Mécanisme d'inférence [20]

Considéré comme le « cerveau » du contrôleur, il permet de lier les degrés d'appartenance des fonctions d'appartenance d'entrée aux fonctions d'appartenance de sortie. Le degré d'appartenance de la fonction de sortie peut être calculé par différentes méthodes :

- La méthode d'inférence Min-Max.
- La méthode d'inférence Max-Prod.
- La méthode d'inférence Somme-Prod

b.1. La méthode Min-Max

C'est la méthode la plus universelle mais qui n'est guère applicable en raison du temps de calcul très long. Cette méthode réalise, au niveau de la condition, l'opérateur « OU » par la formation maximum et l'opérateur « ET » par la formation du minimum.

La conclusion dans chaque règle, introduite par ALORS, lie le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie ***X*** par l'opérateur « ET », réalisé dans le cas présent par la formation du minimum. Enfin l'opérateur « OU » qui lie les différentes règles est réalisé par la formation du maximum.

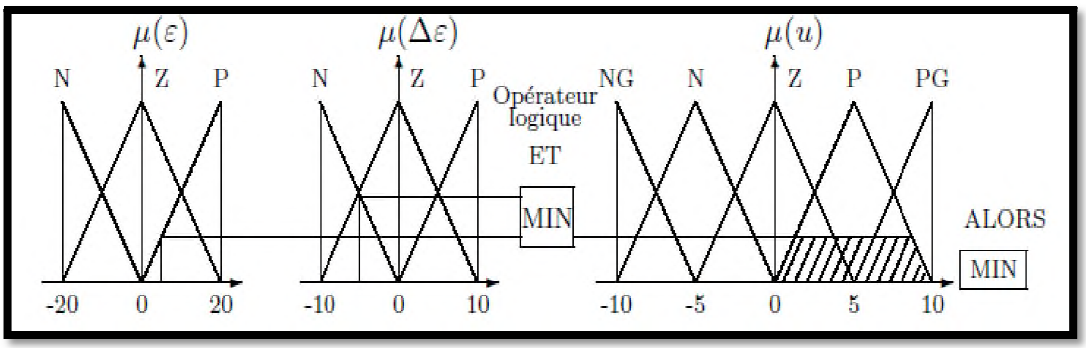


Figure 3-8 : Définition de Alors dans la méthode d'inférence max-min

Par ailleurs l'interaction entre les règles sera nommée OU ainsi, on considère que les deux règles suivantes :

- Si ε est Z et $\Delta\varepsilon$ est N ALORS *u* est N
- Si ε est P et $\Delta\varepsilon$ est N ALORS *u* est P

L'opérateur « OU » se traduit par l'opération max, Nous obtenons ainsi la fonction d'appartenance résultante.

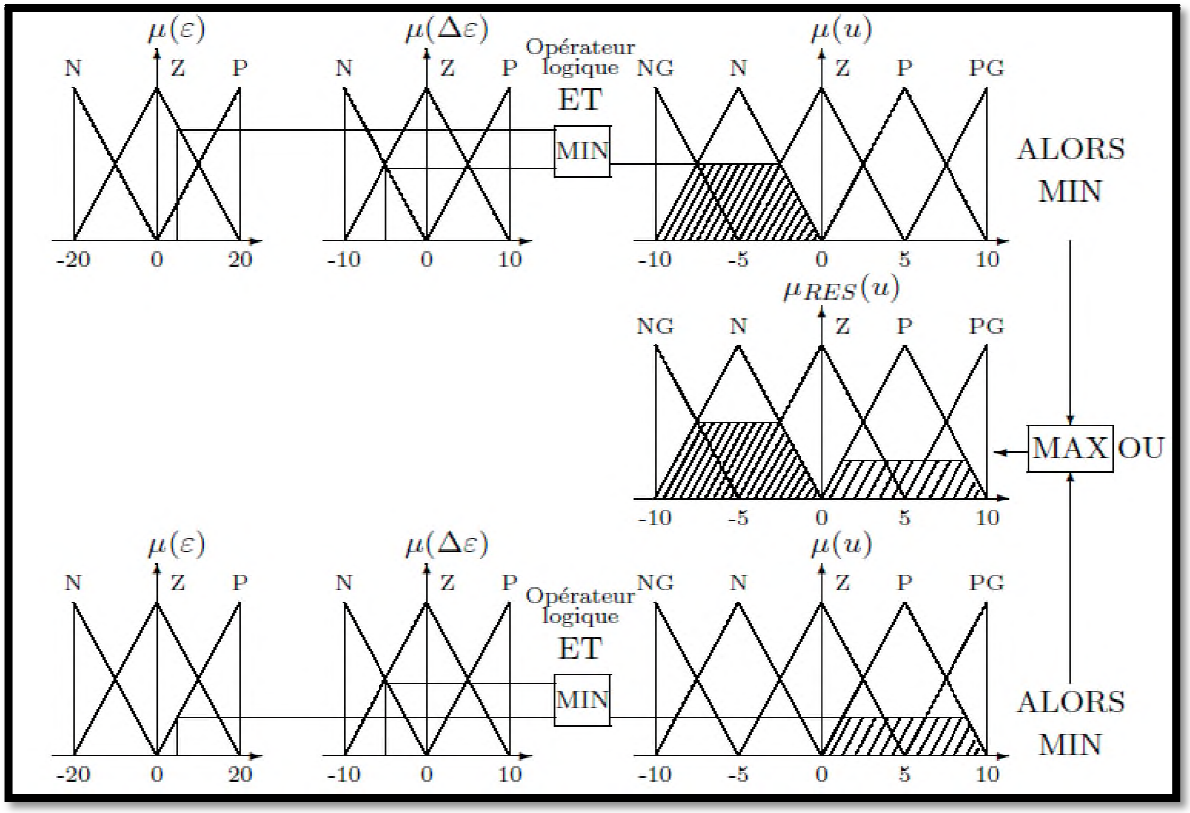


Figure 3-9 : Définition de OU dans la méthode d'inférence Min – Max.

L'application de l'ensemble des règles donne la fonction d'appartenance partielle représentée sur la figure 3-10.

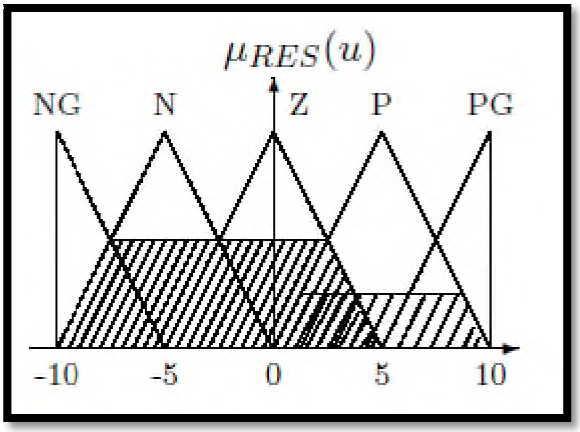


Figure 3-10 : Fonction d'appartenance résultante.

b.2. La méthode d'inférence Max-Prod

La différence avec la méthode précédente est la réalisation d'ALORS qui se traduit par la multiplication de la fonction d'appartenance considérée par la valeur de la règle. Ce résultat est illustré sur la figure 3-11.

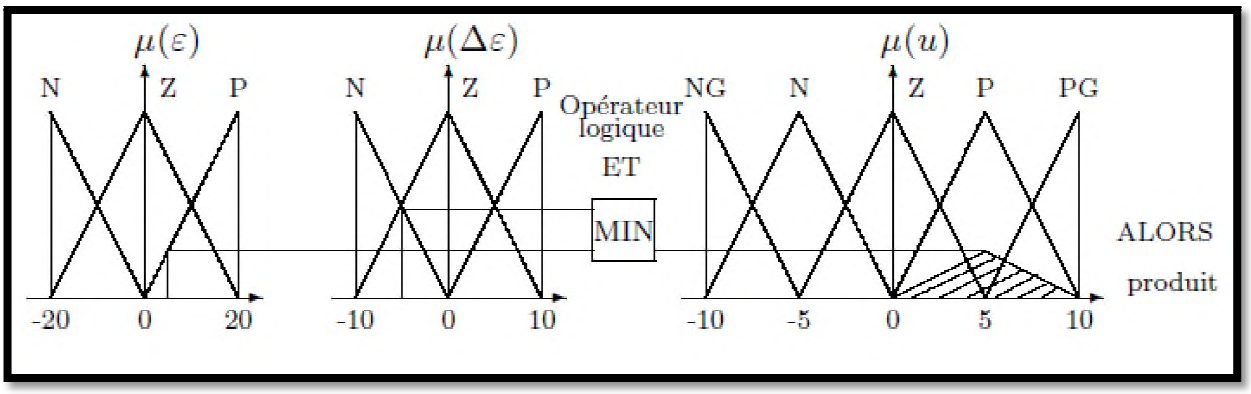


Figure 3-11 : Définition de ALORS dans la méthode d'inférence Max-Prod.

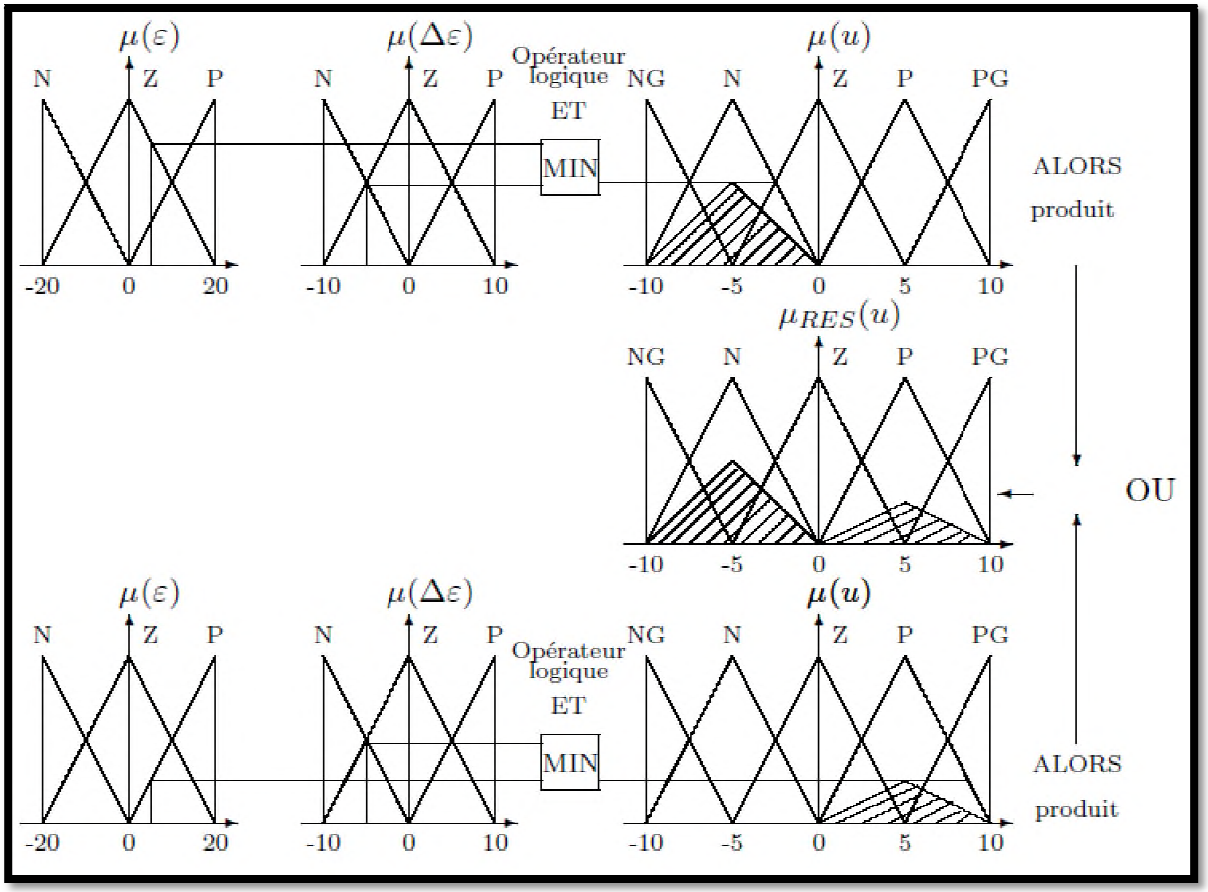


Figure 3-12 : Définition de OU dans la méthode d'inférence Max-Prod.

b.3. La Méthode d'inférence Somme-Prod

Proche de la méthode précédente, il suffit de remplacer la traduction de « ALORS » par la somme des fonctions d'appartenance partielles.

c. Base de connaissances [21]

La conception d'une base de connaissances représente la phase dans la conception des systèmes experts. Elle comprend la base de données et la base des règles floues.

c.1. La base de données [21]

Contient la définition des ensembles flous, les facteurs d'échelle pour la normalisation des ensembles de référence et la partition de l'espace flou d'entrée et de sortie.

c.2. La base des règles floues [21]

Elle rassemble l'ensemble des règles floues de type « Si-Alors » décrivant en termes linguistiques basés sur la connaissance d'un expert le comportement dynamique du système. L'ensemble des règles se présente sous la forme d'une énumération de type :

- Si condition 1 et/ou condition 2 (et/ou) alors action sur les sorties.
- Si condition 3 et/ou condition 4 (et/ou) alors action sur les sorties.
- Si condition 5 et/ou condition 6 (et/ou) alors action sur les sorties.

Dans le domaine de **l'automatique** deux types de règles sont utilisé :

- Modèle de Mamdani :

$$\boldsymbol{R^i : Si\ x_1\ est\ F_1^i\ et\ x_2\ est\ F_2^i\ et\ \dots x_n\ est\ F_n^i\ Alors\ y_j\ est\ G_j}$$

Dans le contrôleur flou de type MAMDANI les conséquences sont floues

- Modèle de Takagi Sugeno :

$$\boldsymbol{R^i : Si\ x_1\ est\ F_1^i\ et\ x_2\ est\ F_2^i\ et\ \dots x_n\ est\ F_n^i\ Alors}$$

$$y_i = a_0^i + a_1^i x_1 + \cdots + a_n^i x_n$$

Dans le contrôleur flou de type TS les conséquences sont numériques

Où : $i = 1,\dots, m$.

m : Nombre total des règles.

$F_1, F_2,\dots, F_n$: Ensembles flous (valeurs linguistique) des entrées ($x_1, x_2,\dots, x_n$)

G_j : Ensemble flou correspondant à la sortie y_j.

$a_0^i, a_1^i, \cdots, a_n^i$: Paramètres ajustables des conséquences de la règle R^i.

On peut écrire les règles d'inférence sous forme d'une matrice appelée Matrice d'inférence, qui est généralement antisymétrique. A titre d'exemple, si on considère un contrôleur flou à

x_1

deux entrées caractérisées par trois ensembles flous et une sortie, alors la matrice d'inférence peut prendre la forme suivante :

x_2 \ x_1	F_1^1	F_1^2	F_1^3
F_2^1	G_1	G_4	G_7
F_2^2	G_2	G_5	G_8
F_2^3	G_3	G_6	G_9

Tableau 3-1 : Matrice d'inférence floue.

d. Défuzzification [19]

Le résultat d'une inférence floue est une fonction d'appartenance. C'est un sous-ensemble flou. Un organe de commande nécessite un signal de commande précis. La transformation floue en une information déterminée est la défuzzification (concrétisation). De plus, on doit souvent prévoir un traitement du signal et la conversion digitale/analogique. Il y a plusieurs méthodes de défuzzification proposée dans la littérature. Il n'y a pas de stratégie systématique pour choisir parmi l'une de ces méthodes, la méthode la plus utilisée c'est la méthode du centre de la gravité.

L'abscisse du centre de gravité peut être déterminée en utilisant la formule générale :

$$\overline{y} = \frac{\sum_{i=1}^{N_c} \mu_{ci}(y)\,\omega_i}{\sum_{i=1}^{N_c} \mu_{ci}(y)} \qquad (4.3)$$

Ou :

$\overline{y}$: la valeur numérique de sortie.

N_c : le nombre de valeur linguistique de conséquence.

ω_i : les sommets des ensembles flous de conséquence.

μ : la valeur d'appartenance

III.10. Avantages et inconvénients de la logique floue [18]

Évidemment, le réglage par la logique floue réunit un certain nombre d'avantages et inconvénients. Les avantages essentiels sont :

- ✓ La théorie est simple et s'applique à des systèmes complexes. Robustesse de la commande floue vis-à-vis des incertitudes.
- ✓ Possibilités de commande auto-adaptative aux variations du procédé.
- ✓ La maitrise du système à régler avec un comportement complexe.
- ✓ L'obtention fréquente de meilleures prestations dynamiques (régulateur non linéaire).

Par contre les inconvénients sont :

- ✓ Technique de réglage essentiellement empirique.
- ✓ Performances dépendent de l'expertise.
- ✓ Le manque de directives précises pour la conception d'un réglage (choix des grandeurs à mesurer, détermination de la fuzzification, des inférences et de la défuzzification).
- ✓ L'approche artisanale et non systématique (implémentation des connaissances de l'opérateur, est souvent difficile).

III.11. Conclusion

Dans ce chapitre nous avon abordé principalement une téchnique de commande intéllegente, c'est la commande par logique floue, qui permet la représentation et le traitement de connaissances imprécises ou approximatives. Cette méthode se base sur des variables, des valeurs et des règles linguistiques.

IV.1 Introduction

L'idée principale de ce travaille et d'appliquer deux types de commandes différentes (le contrôleur flou FLC et le contrôleur PID classique) pour contrôler la vitesse et la position du moteur a courant continu. Après avoir énoncé les conceptions de base de logique floue, le FLC et LabVIEW, nous nous somme intéressé à implémenter l'algorithme de commande dans la carte de développement « digital electronic FPGA board de la firme National Instrument ». Nous présentons dans ce chapitre le circuit électronique ainsi le programme nécessaire a la mise en marche de l'application.

IV.2 Schéma synoptique principal du projet

Le schéma synoptique de notre projet est déterminé dans la figure suivante.

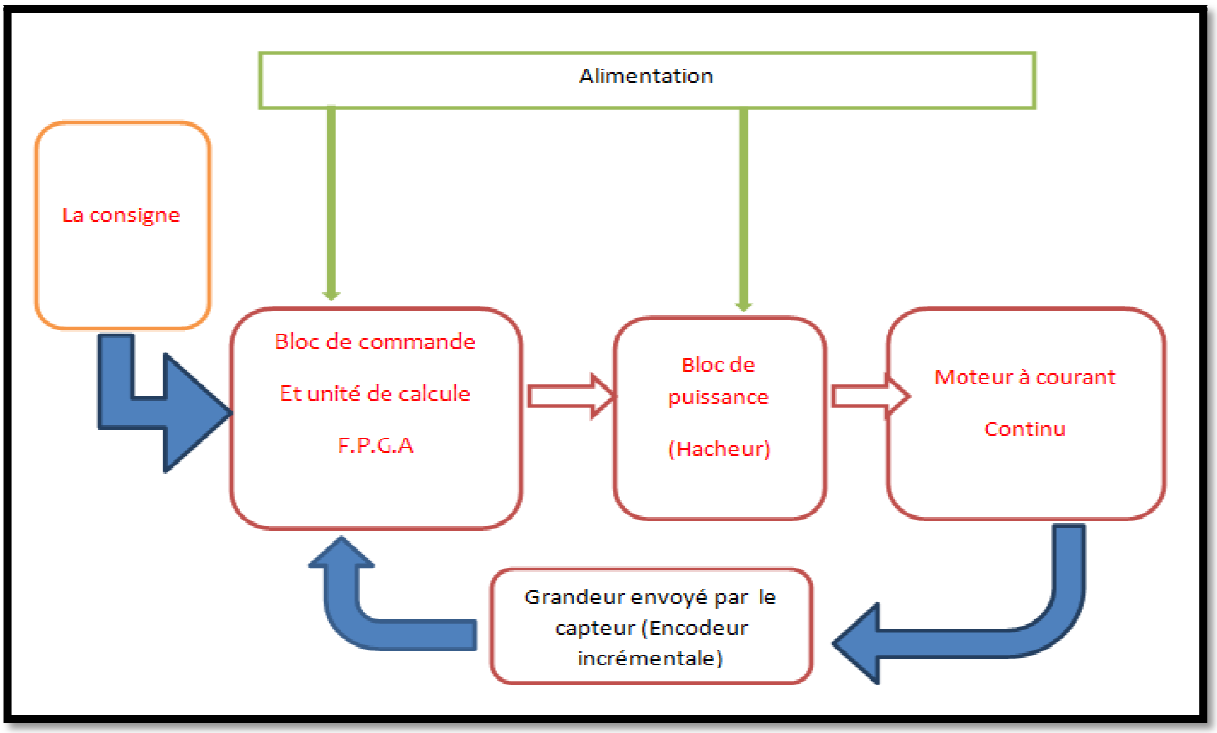


Figure 4-1 : Schéma synoptique du dispositif réalisé.

IV.3 Circuits des puissances

IV.3.1 Hacheur

Les hacheurs sont des convertisseurs statiques de type DC-DC, la nécessité de les utiliser est pour obtenir une tension continu réglable à partir d'une tension continu fixe. Ils ont

souvent utilisés pour la commande des MCC et les alimentations à découpage et d'autres applications.

a. Principe de fonctionnement

Le principe de fonctionnement d'un hacheur est simple, il consiste à établir une connexion source-charge pendant un laps de temps (***αT***), ensuite il coupe cette connexion pendant un α laps de temps (**1** - ***αT***) et il refait ça périodiquement (chaque période ***T***).

Dans le hacheur à quatre quadrants, le changement du sens de rotation du MCC est lié au changement du courant alors que le changement de la vitesse est lié au changement de la tension aux bornes du moteur. L'interrupteur est formé de deux composants. Le premier est un composant commandé à l'amorçage et au blocage (transistor, IGBT, GTO), alors que le second est une diode, ils sont montés en antiparallèles. La figure 4-2 présente un schéma simplifié du hacheur quatre quadrants (pont en H).

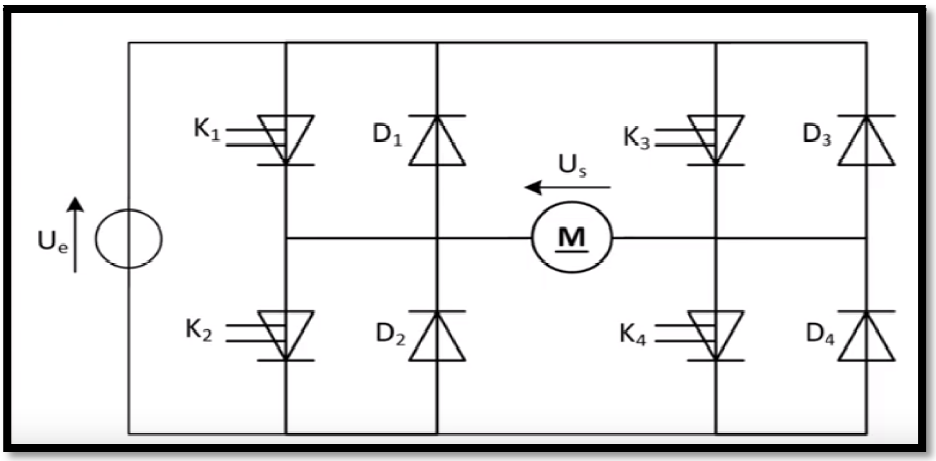


Figure 4-2 : Schéma structurel d'un hacheur à quatre quadrants.

Les interrupteur K1, K2, K3 et K4 sont commandés par deux signaux MLI (Modulation de Largeur d'Impulsion) ou PWM (Pulse Width Modulation) complémentaires avec une zone morte. Lorsque les interrupteur K1, K4 sont commandés par le premier signal PWM, ils sont donc fermés et les interrupteurs K2 et k3, sont commandés par le deuxième signal PWM, doivent être ouverts et inversement.

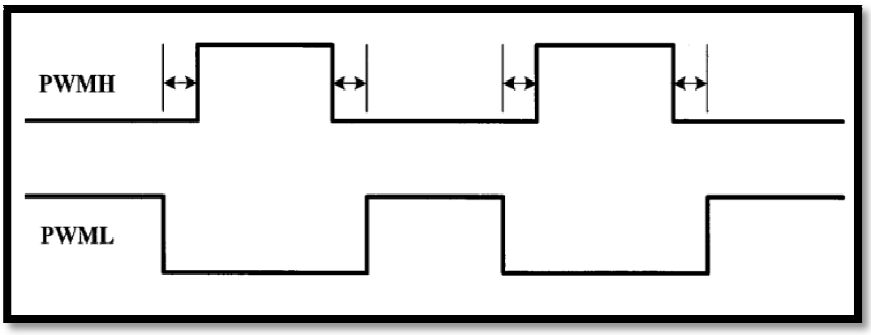


Figure 4-3 : Signal PWM et son complément avec une zone morte.

Ce montage a deux modes de fonctionnement :

1. Le mode moteur.
2. Le mode génératrice.

La figure suivante nous résume les modes de fonctionnement de ce montage.

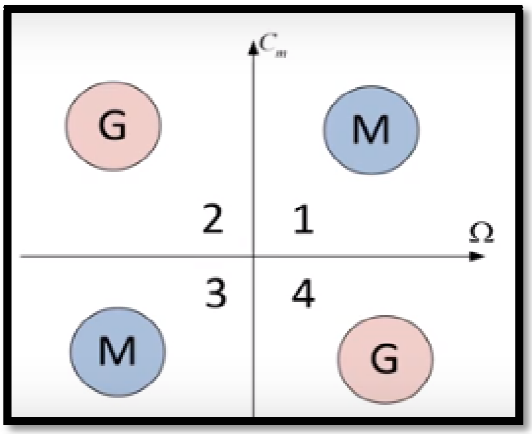


Figure 4-4 : Modes de fonctionnement dans les quatre quadrants

Prenant le premier quadrant c'est le mode moteur avec le sens de rotation positif et le couple positif, on va voire une phase d'alimentation avec la commande des interrupteurs K1, K4. Et dans la phase de roue libre c'est la diode D3 et le interrupteur K1 qui sont commandés.

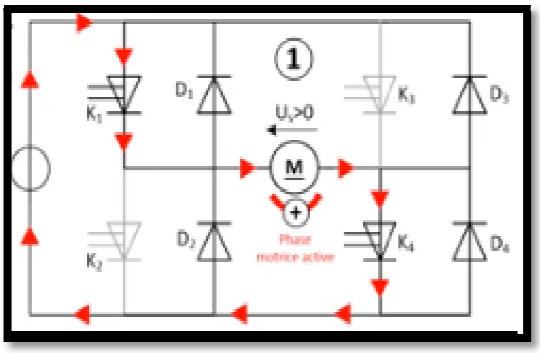


Figure 4-5 : le mode moteur

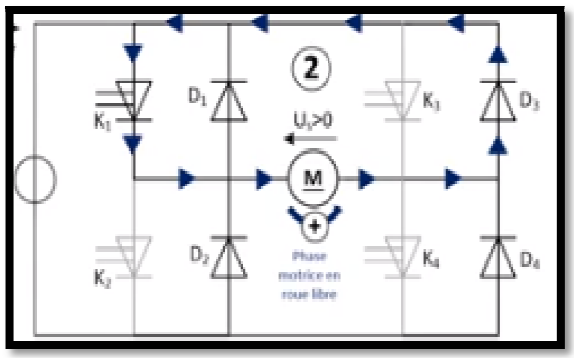


Figure 4-6 : Phase de roue libre

Dans le deuxième cas un quadrant génératrice avec le couple moteur positif et la vitesse négatif, on va dans un premier temps commandé les interrupteur K2, K3 et dans la phase de roue libre K4 et D2 qui seront passant.

Dans le quadrant numéro 3 c'est un quadrant moteur avec la vitesse et le couple moteur sont les deux négatif, on peut donc commander K3, K2 avec les deux transistors qui seront passant. Puis après la phase de roue libre on commande K3 et D1 qui seront passant.

Et enfin si on veut fonctionner en mode génératrice avec une vitesse positif et un couple moteur négatif on pourra commander K4 et K1 et dans ce cas là ça sera donc D4 et D1 qui seront passante ou alors dans une autre phase de roue libre on commande K3 et K4 donc K3 et D1 qui seront passant

b. Réalisation du l'Hacheur à quatre quadrants

S'en basent sur le circuit intégré L6203 qui est principalement un Hacheur à quatre quadrants ce circuit peut supporter jusqu'à 48V et un courant d'intensité de 5 A. A l'aide du logiciel EAGLE on a pu réalise le PCB de ce circuit.

Les valeurs des composants choisissent pour réaliser le circuit de puissance :

Composants	Valeurs
C7, C5	15 ***ηF***
C6	22 ***ηF***
C8	220 ***ηF***
R1	10 **Ω**
D1, D2	BYV28

Tableau 4-1 : Valeurs des composants utilisés dans l'Hacheur.

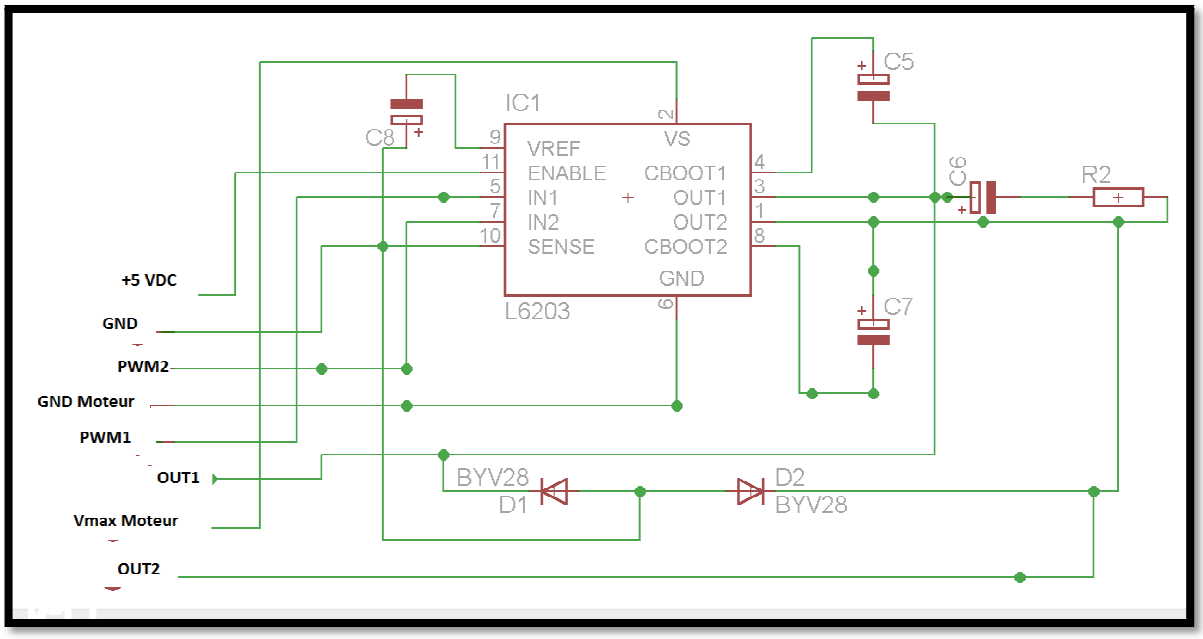


Figure 4-7 : Schéma électrique du hacheur réalisé.

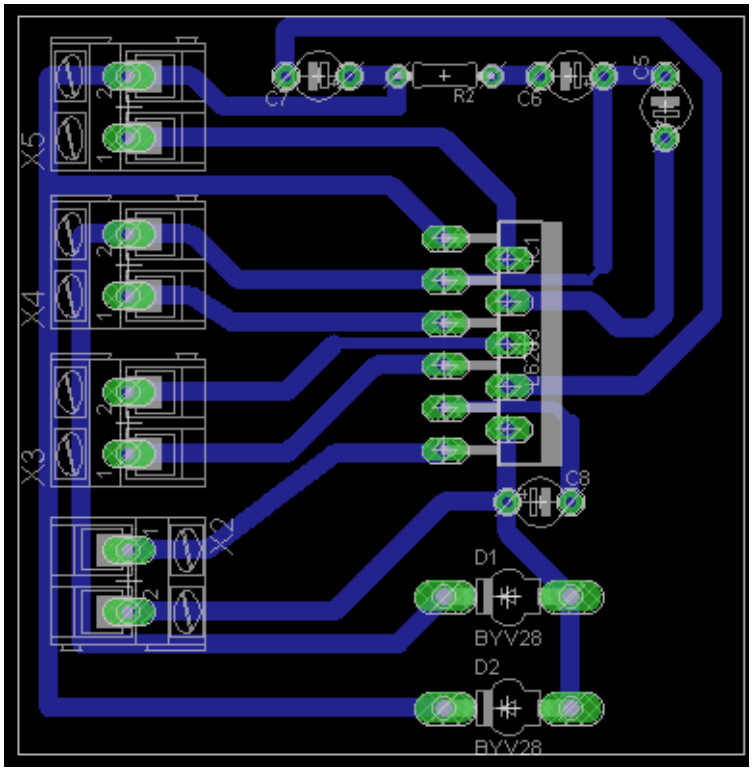


Figure 4-8 : Schéma final du PCB.

IV.3.2 Circuit d'isolation

L'isolation galvanique sert à séparer deux circuit électronique, de tel sort qu'aucun courant ne puisse passer d'une partie a l'autre, a cause de la sensibilité élevée des pins de la carte de développement nous allons réaliser un circuit d'isolation a base d'un optocoupleur

6N137 afin d'isoler la carte de développement et la carte de puissance constituée d'un Hacheur.

IV.3.3 Alimentation stabilisée

Pour alimenter les circuits intégrés qui sont utilisés dans ce projet tel que le Hacheur L6203, l'optocoupleur 6N137 et l'inverseur 74LS14N, nous allons besoin d'une alimentation de 5V continu. En utilisant le régulateur LM7805 nous arrivons à baisser la tension du 20V continu au 5V continu. Dans la Figure 4-9 nous représentons le schéma électrique global de l'alimentation stabilisée. Les valeurs des composants utilisés dans ce circuit est illustré dans le tableau ci-dessous :

Composants	**C1**	**C2**	**C4**	**C3**	**C7, C5**	**C6**	**C8**	**R1**	**R2, R4**	**R3, R5**	**D1, D2**	**D3**
Valeurs	**100 µF**	**0.33 µF**	**0.1µ F**	**100 0µF**	**15 ηF**	**22η F**	**220 ηF**	**10Ω**	**150Ω**	**4.7K Ω**	**BYV28**	**1N400**

Tableau 4-2 : Valeurs des composants utilisés pour l'alimentation stabilisée

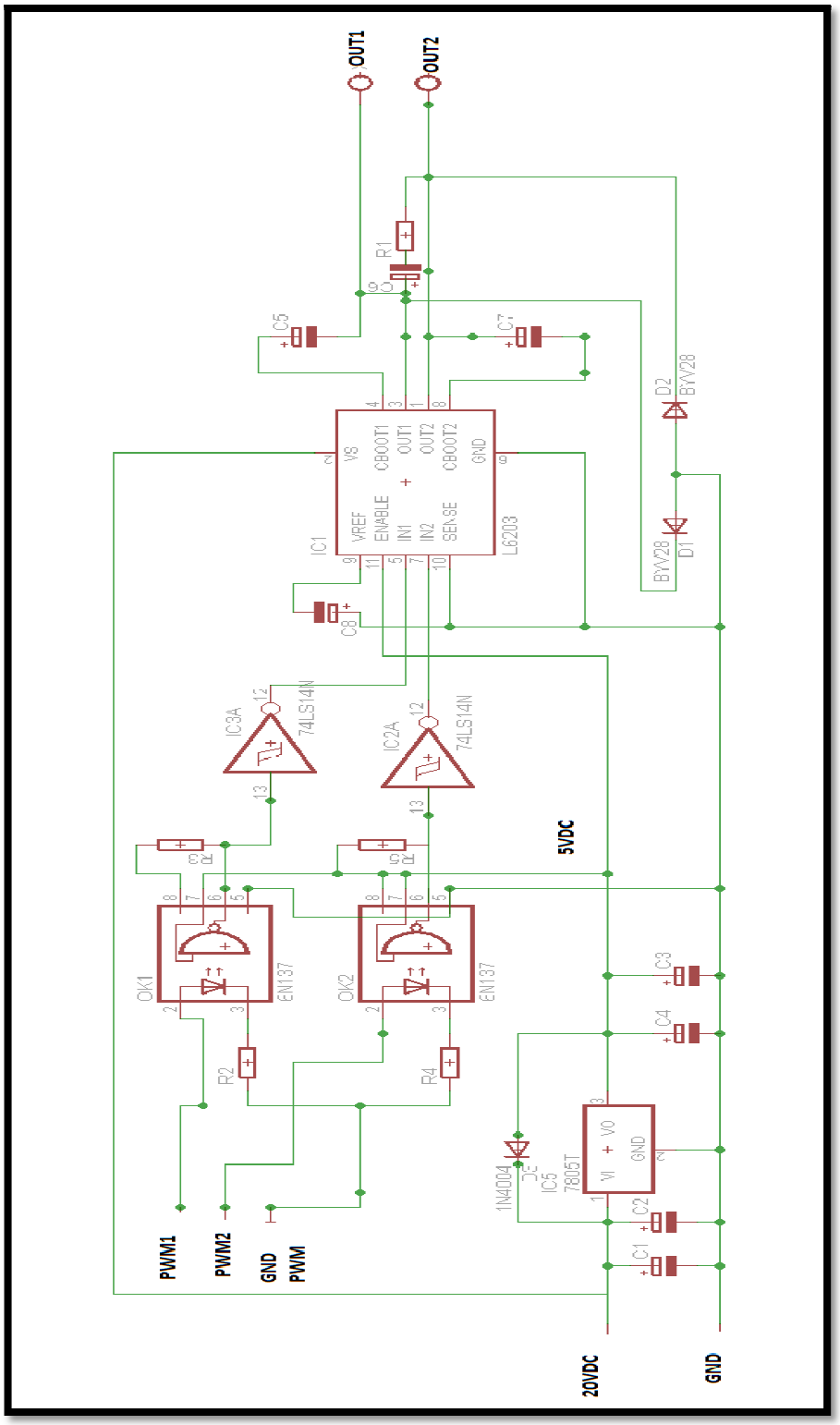


Figure 4-9 : Schéma électrique globale

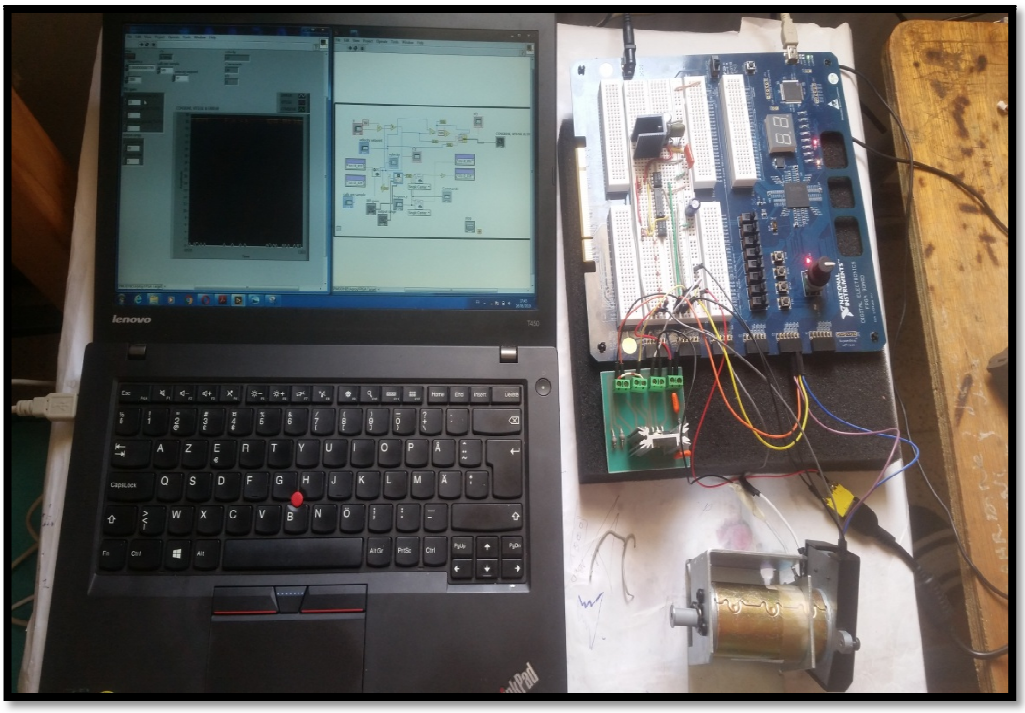


Figure 4-10 : réalisation final du projet

IV.4 Contrôle de la vitesse du moteur à courant continu

Notre objectif est de contrôler la vitesse du moteur a courant continu (aiment permanent) sur une consigne fixe par deux contrôleurs, le premier c'est un contrôleur classique PID et le seconde c'est un contrôleur intelligent flou.

IV.4.1 Le contrôleur PID

Pour commander la vitesse du moteur MCC avec le contrôleur PID on utilise le schéma bloc d'asservissement ci-dessous.

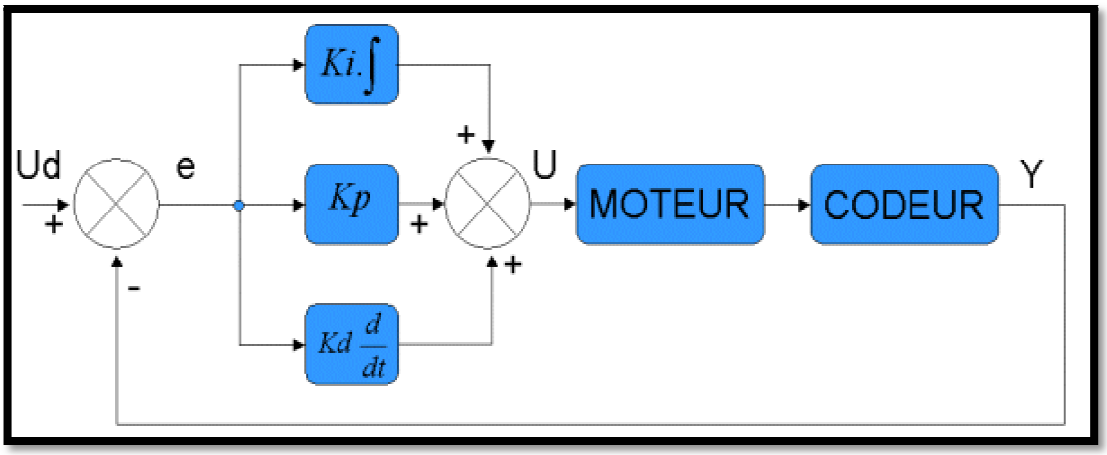


Figure 4-11 : Schéma d'asservissement PID

a. Le programme PID implémenté sous LabVIEW

Description du programme

Notre programme est constitué par :

- ✓ Une boucle principale (boucle while).
- ✓ Deux entrées numériques incluses dans la boucle while pour l'acquisition des données (les donnés envoyées par l'encodeur).
- ✓ Deux sorties numériques pour générer les signaux PWM.
- ✓ Bloc du contrôleur PID.
- ✓ Bloc de l'encodeur pour calculer la vitesse de rotation du moteur.

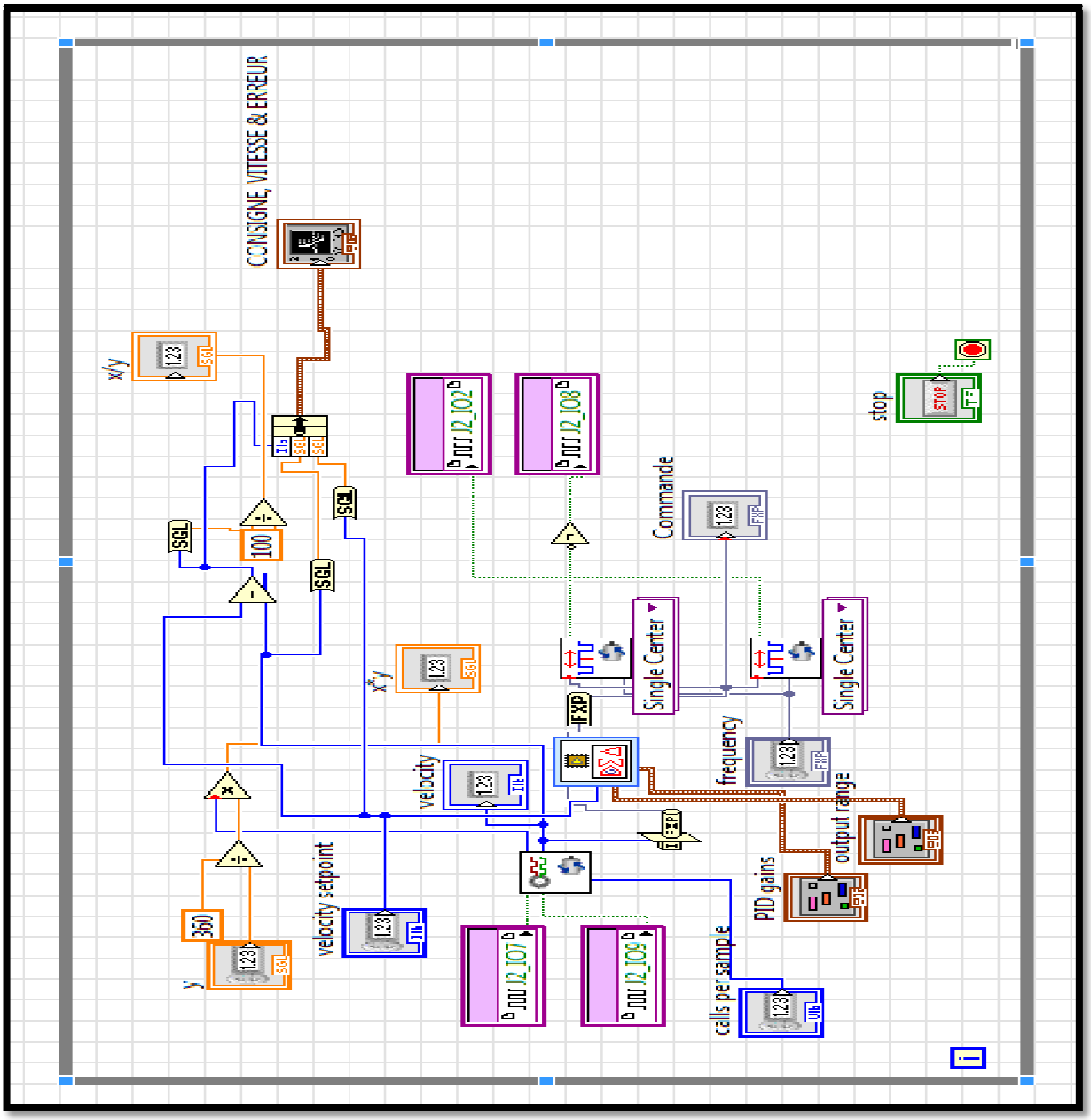
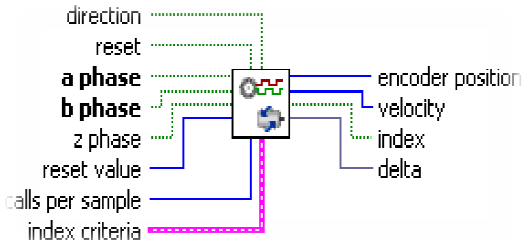
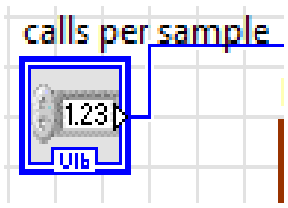
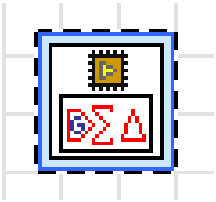


Figure 4-12 : Le contrôleur PID implémenté sous LabVIEW.

b. Description des différents terminaux utilisés

Terminaux	Fonctionnement
Décodeur incrémental 	Calcule la position, la vitesse et la transition d'index du codeur à partir des entrées de codeur incrémentielles (en quadrature)
	Spécifie le nombre d'impulsions d'horloge entre les quelles le VI incrémente le nombre de codeurs pour le calcul de la vitesse. Pour assurer un rapport de vitesse précis, cette valeur doit être égale au taux de la boucle de contrôle de vitesse.
Bloc PID 	Implémente un algorithme PID à point fixe pour les applications PID avec contrôle à haute vitesse et / ou nombre de canaux élevé sur une cible FPGA. Vous pouvez utiliser ce VI Express avec des configurations monocanal ou multicanal. L'algorithme PID présente une plage d'action de contrôle et utilise un calcul d'anti-enroulement d'intégrateur pour limiter l'effet de l'action intégrale pendant les transitoires. L'algorithme PID comporte également une sortie de contrôleur sans à-coups pour les changements de gain PID.

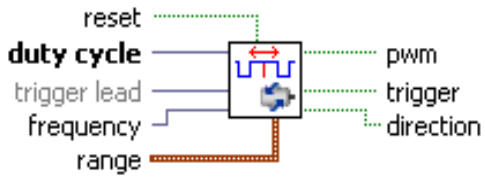
Modulation de largeur impulsion 	Génère un signal PWM pour contrôler la sortie d'un moteur. Le comportement de sortie PWM par défaut est aligné au centre.
	Les entrées pour l'acquisition des signaux de l'encodeur (phase A et phase B).
	Les sorties PWM pour commander le hacheur

Tableau 4-3 : Représentation des terminaux et blocs utilisés dans le programme

c. Résultat obtenus de réalisation avec le contrôleur PID

Dans cette figure nous allons présenter les résultats obtenus lors de la réalisation pratique.

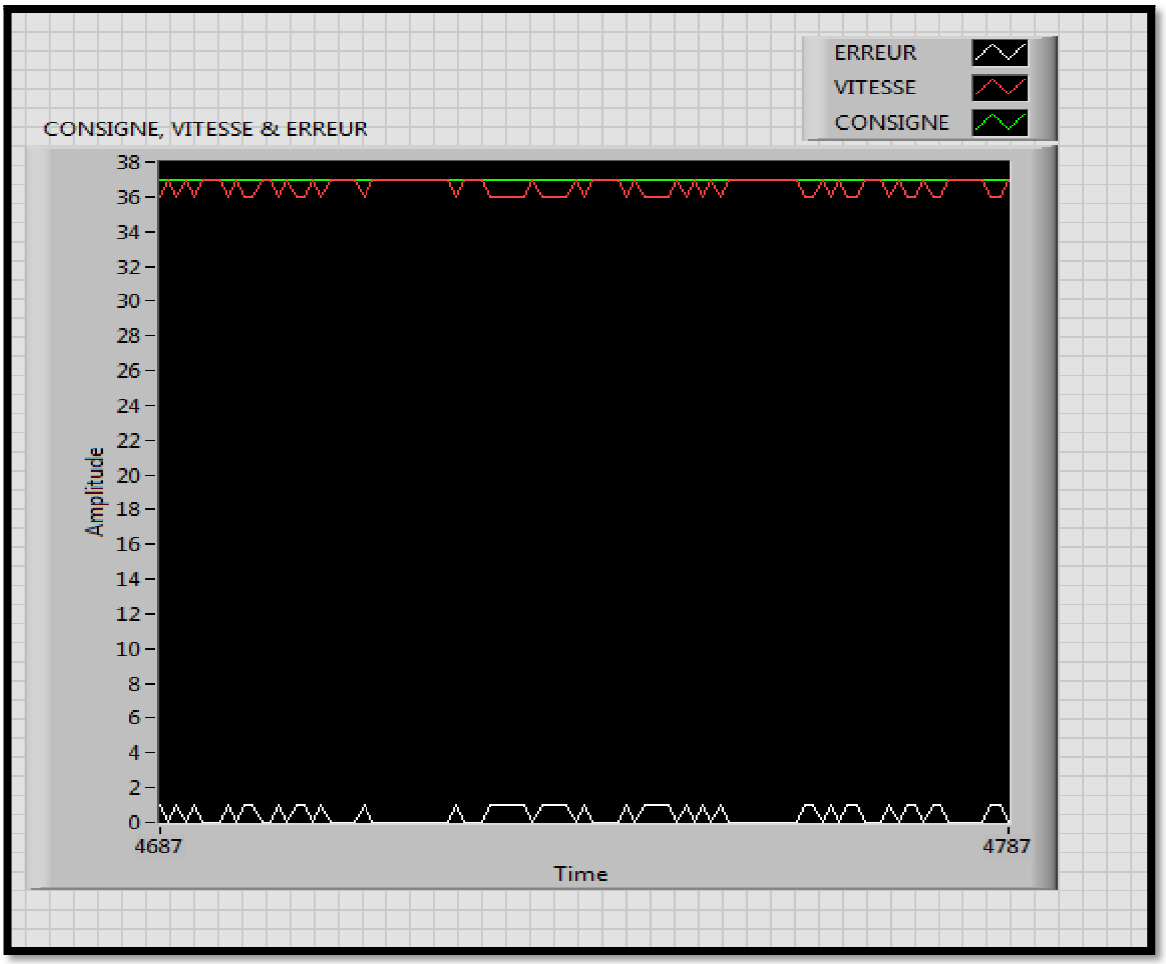


Figure 4-13 : Représentation de la vitesse réelle, la consigne et l'erreur.

Pour tester les performances du programme développé on applique une perturbation externe sur le moteur et en verra comment réagit le système, dans la figure suivante nous présentons la sortie du système avec une perturbation.

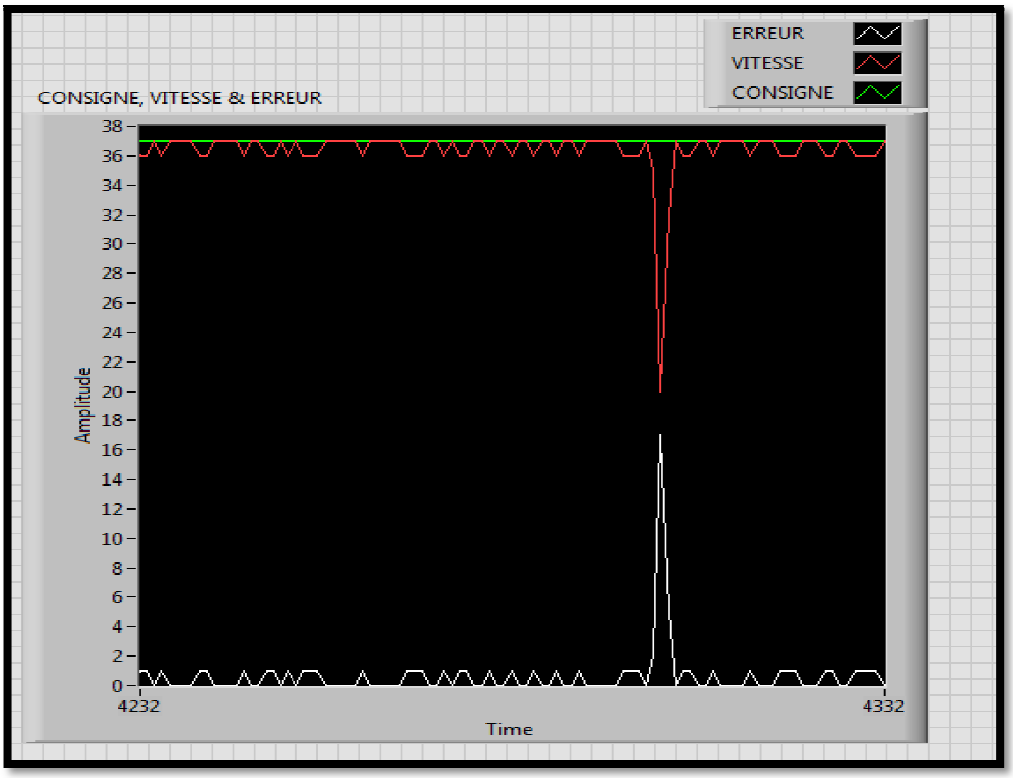


Figure 4-14 : Régulation de la vitesse du MCC avec perturbation

D'après la figure ci-dessus on remarque que lorsqu'on applique une perturbation externe (une charge) sur le moteur, la vitesse diverge sur la consigne cela implique une augmentation d'erreur, puis après un certain temps court la vitesse revient de suivi la consigne.

IV.4.2 Le contrôleur flou

Le schéma de commande du MCC avec le contrôleur flou est représenté par la figure ci-dessous.

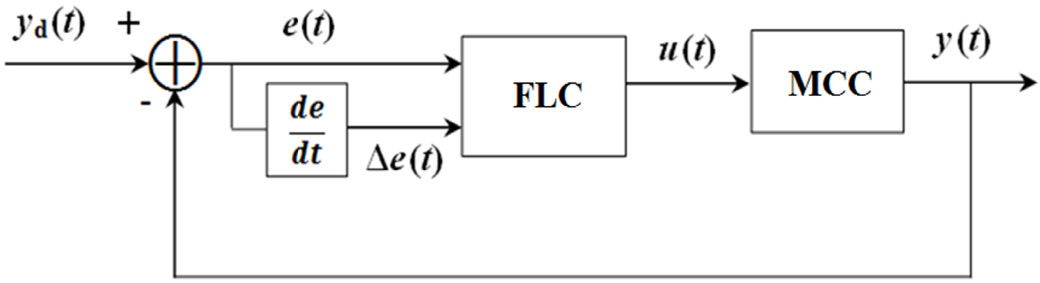


Figure 4-15 : Schéma de commande du MCC avec le contrôleur flou

IV.4.3 La conception du contrôleur flou

a. Choix des variables d'entrées et de sorties

Pour réaliser un contrôleur flou il faut bien choisir ces variables d'entrées et de sortie, dans le cas ou on veut contrôler la vitesse d'un moteur a courant continu nous avons choisit

deux variable d'entrées, la première nommée ***e(t)*** est l'erreur entre la vitesse mesurée et la consigne, la deuxième est la variation d'erreur c'est-à-dire sa dérivée $\frac{de}{dt}$. La variable de sortie est la tension ***u(t)*** appliquée aux bornes du MCC.

b. Les univers de discoure des variables d'entrées et sorties

Selon les caractéristiques de notre système (le moteur) on a pu choisir les univers de discoure de l'erreur, la variation d'erreur et la commande.

- La tension d'alimentation du moteur : 12V ⇒ $u \in [-12, 12]$
- La vitesse du moteur : 288 tr/min ⇒ $e(t) \in [-288, 288]$ d'où $\frac{de}{dt} \in [-576, 576]$

c. Les règles d'inférences

La commande floue est basée sur un expert du système pour déterminer les règles d'inférence, dans notre cas on peut déterminer les règles d'inférence en utilisant la figure suivante qui représente le comportement de système dans le temps.

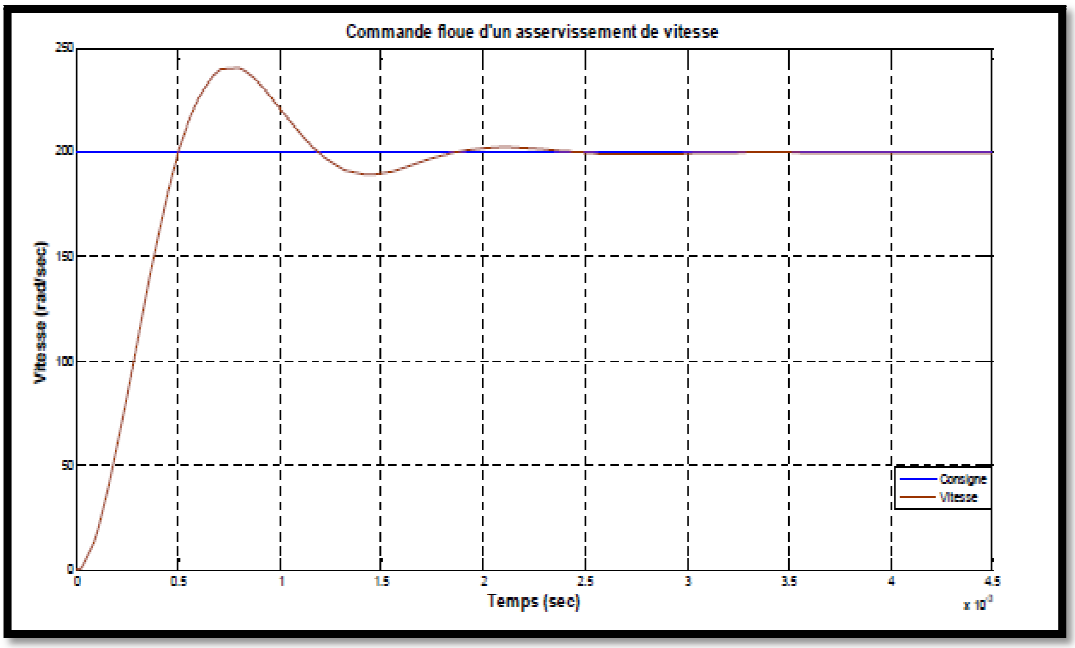


Figure 4-16 : Comportement du système dans le temps

Nous avons choisit trois sous ensembles pour chaque variable d'entrée, le régulateur flou synthétisé dans cette partie est de type MAMDANI donc les conséquences sont flou. Le tableau suivant représente les 9 règles.

de/dt \ *e*	N	Z	P
N	GN	N	Z
Z	N	Z	P
P	Z	P	GP

Tableau 4-4 : Matrice d'inférence

d. Création d'un système flou dans LabVIEW

Dans LabVIEW, un système flou est défini dans la barre de menu du diagramme, Tools(Outils) ⟶ Control Design and Simulation ⟶ Fuzzy System Designer…

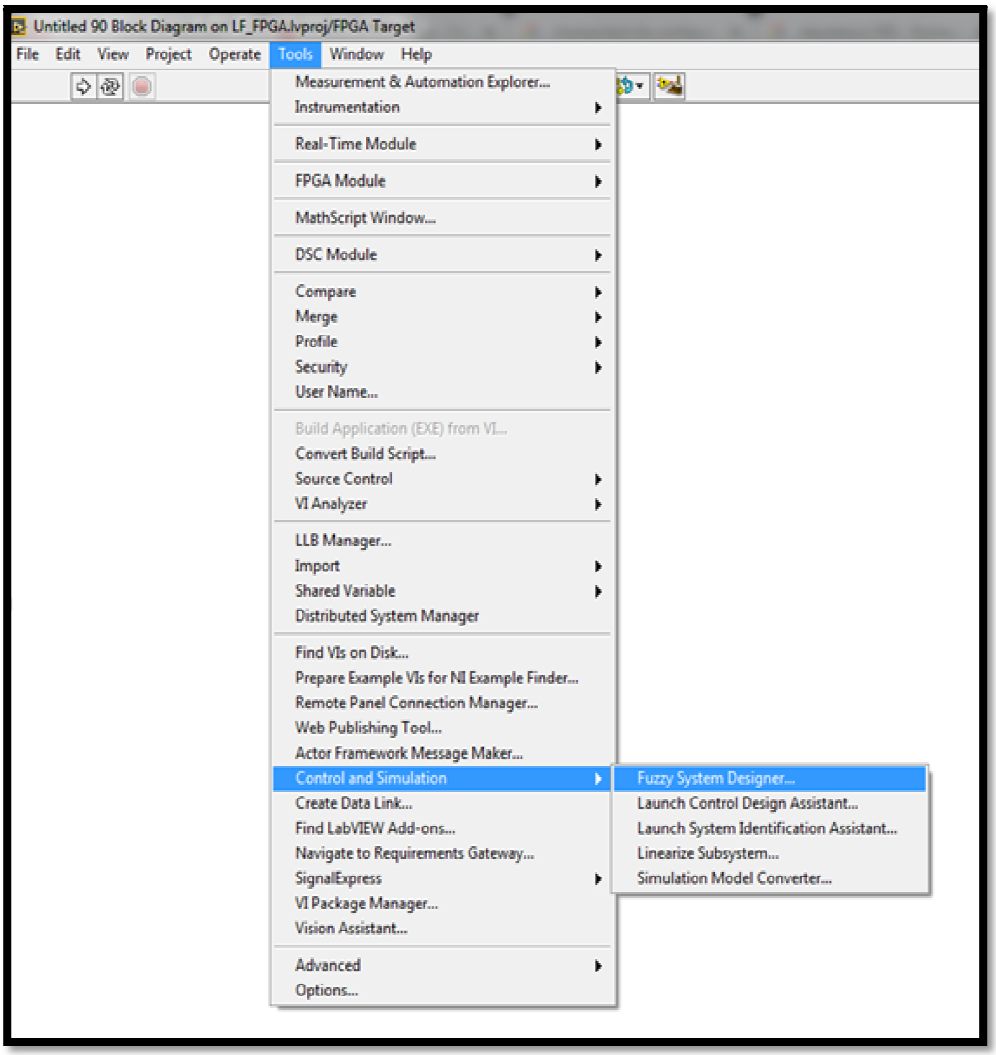


Figure 4-17 : Création du contrôleur flou dans le LabVIEW

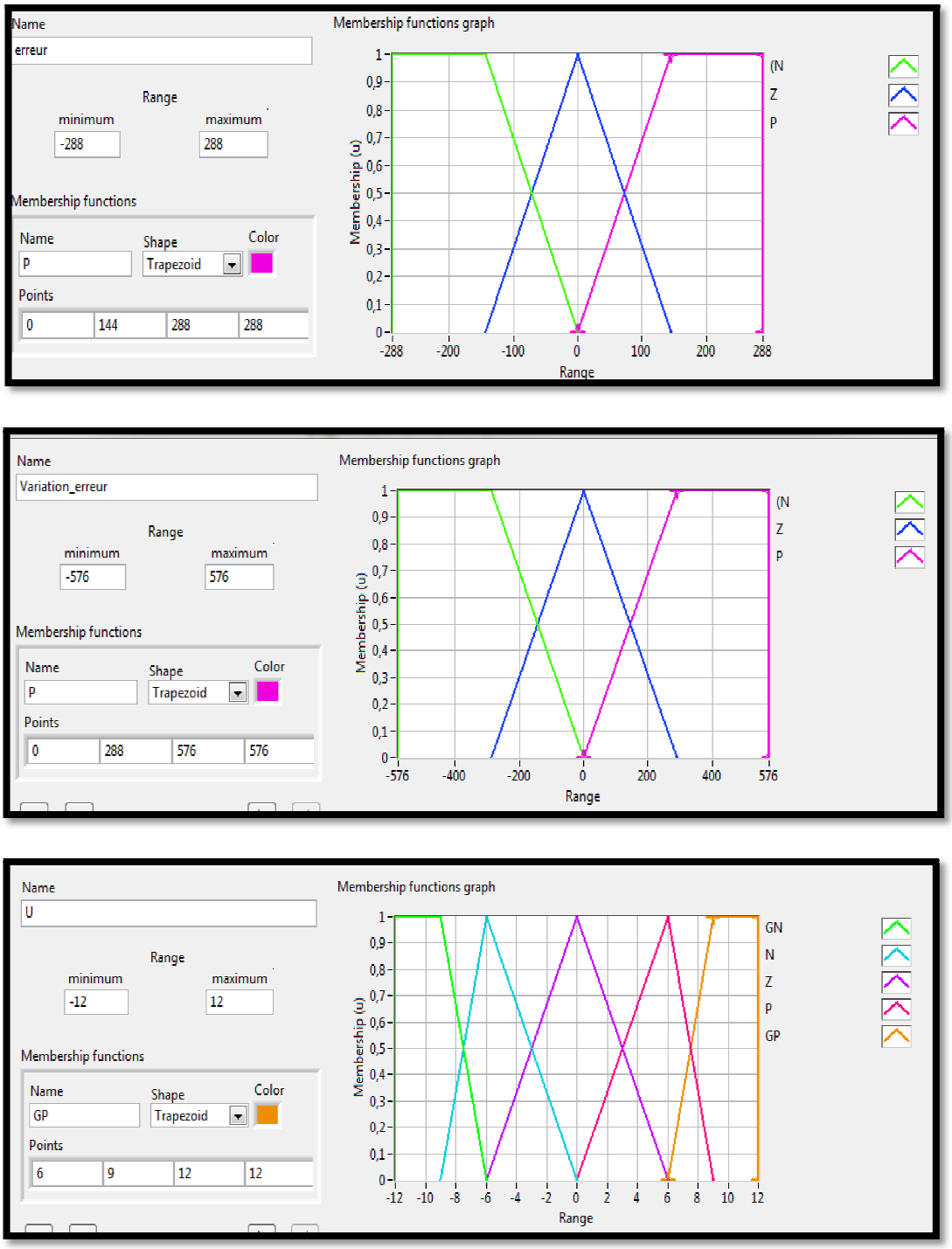


Figure 4-18 : Définition des fonctions d'appartenances
pour les entrées « ***e*** » et « **Δ*e*** » et la sortie « ***u*** »

Maintenant nous pouvons établir les lois de bloc flou, le principe celui d'actions conditionnelles « ***if*** » « ***then*** » représenté sur la figure ci-dessus.

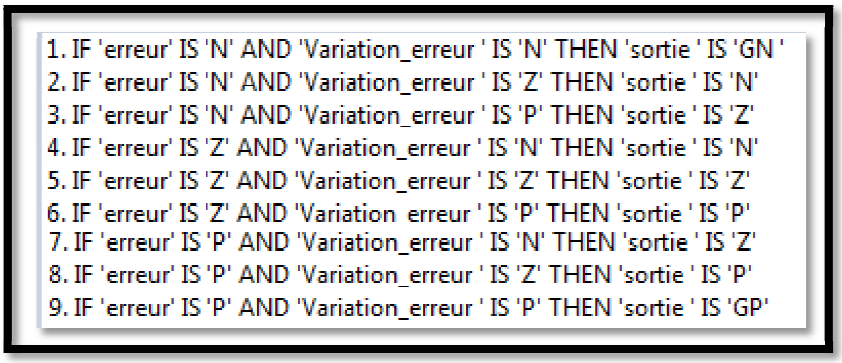


Figure 4-19 : Les règles d'inférence

Après la fuzzification des entrées du système flou, le contrôleur flou utilise les termes linguistiques de ces entrées et les règles pour déterminer les variables linguistiques de sortie.

Lorsqu'une règle possède plus qu'un antécédent, nous devons spécifier la méthode de conjonction de ces antécédents, à savoir comment calculer la valeur vraie de la sortie à partir de ces antécédents.

Le toolkit possède les types de conjonction suivants :

1. AND (Minimum) : Minimum des degrés d'appartenance.
2. AND (Produit) : Produit des degrés d'appartenance.
3. OR (Maximum) : Maximum des degrés d'appartenance.

Pour la défuzzification on peut choisir la méthode de centre de gravité.

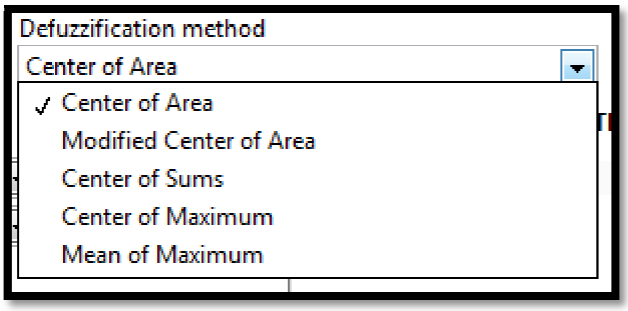


Figure 4-20 : Choix de la méthode de défuzzification

e. L'implémentation du contrôleur flou dans LabVIEW

Nous considérons le système flou que nous avons réalisé précédemment. Le système flou est lu dans le répertoire défini par la commande le chemin qu'on applique à l'entrée file path du VI FL Load Fuzzy System.vi. Ce système est appliqué à l'entrée Fuzzy system in du VI FL Fuzzy Controller.vi. Pour choisir plusieurs entrées et une seul sortie nous sélectionnons le type MISO. La variation d'erreur est réalisée à l'aide d'un nœud de rétroaction. Les deux

variables d'entrées sont regroupées dans un tableau pour être appliquée au contrôleur. Nous affichons la consigne et la sortie du système dans le même graphe XY. Le contrôleur flou et la fonction de transfert et les autres VIs sont insérés dans une boucle de contrôle et de simulation, les valeurs des gains du régulateur sont choisies après plusieurs tests d'ajustement Ge=25, Gve =10^{-5} et Gc =200.

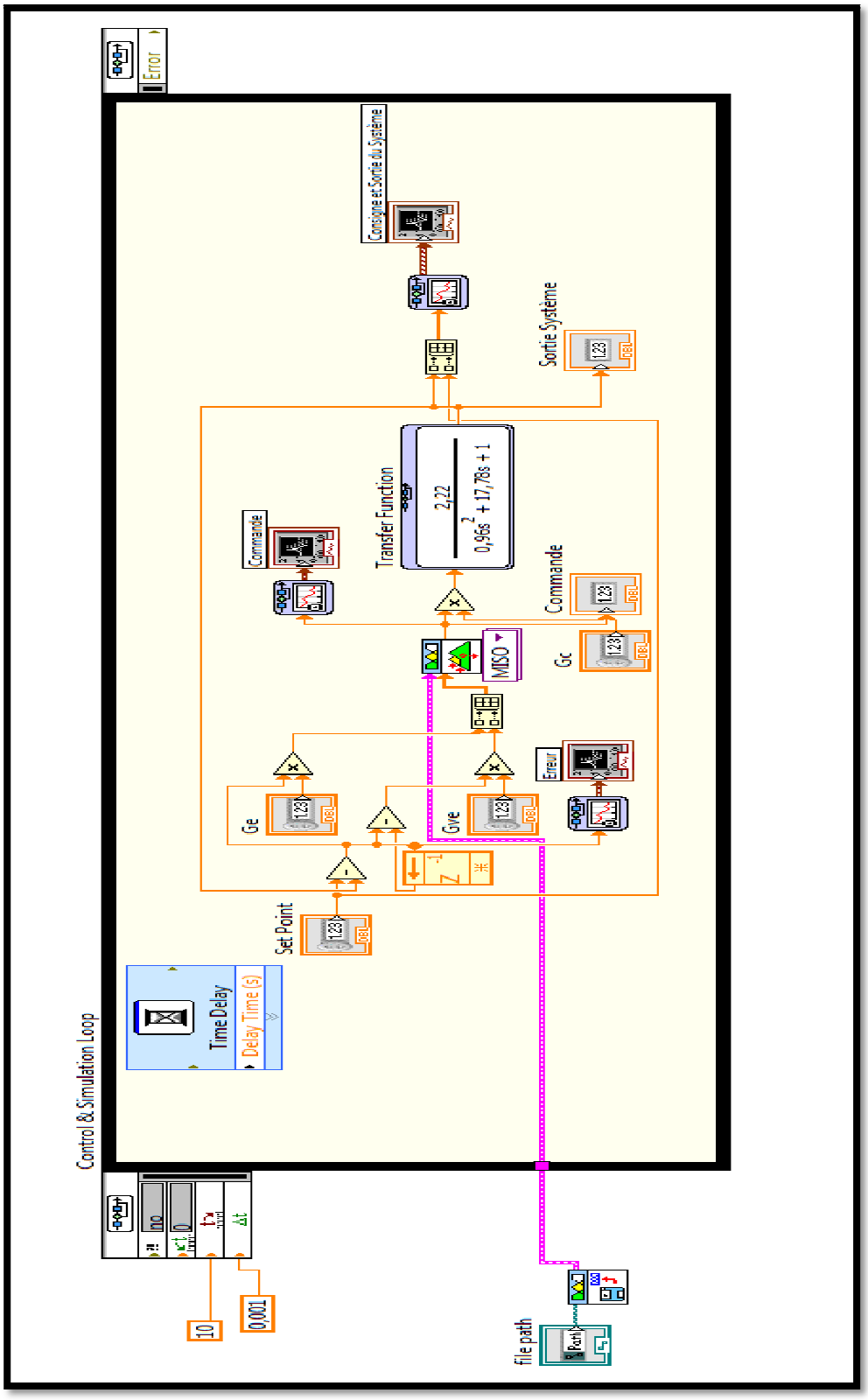


Figure 4- 21 : Programme du contrôleur flou

IV.4.4 Résultat de la simulation avec le contrôleur flou

Dans la face avant du programme nous affichons la consigne et la vitesse dans le même graphe XY. Pour comparer les résultats on doit afficher le graphe de l'erreur.

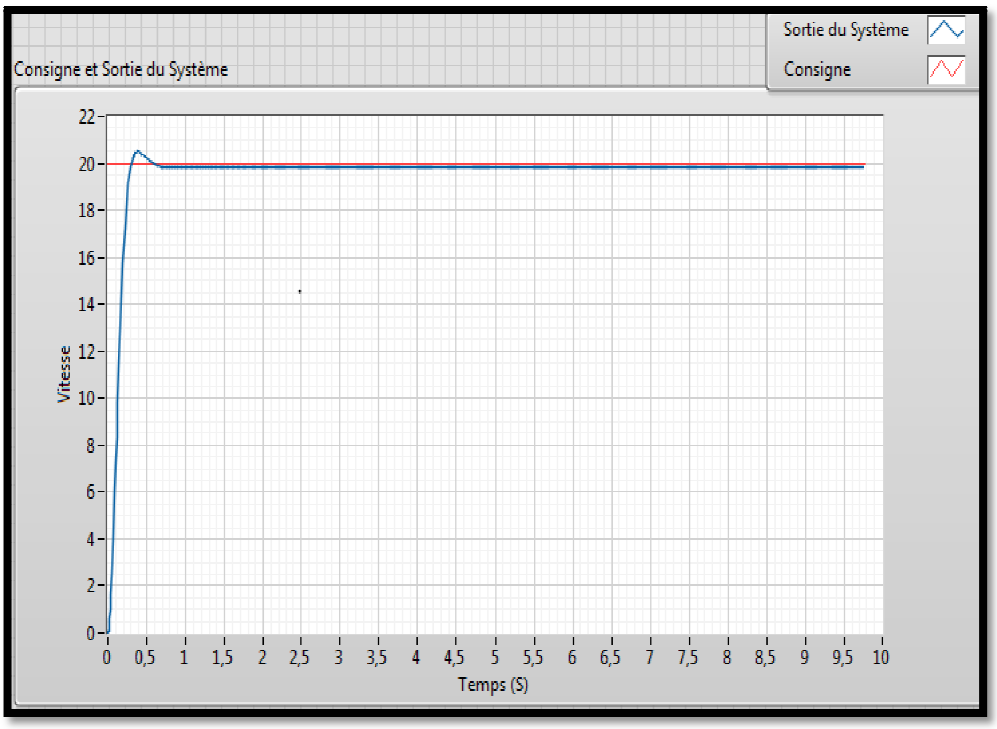


Figure 4- 22 : Représentation de la consigne et la sortie du système

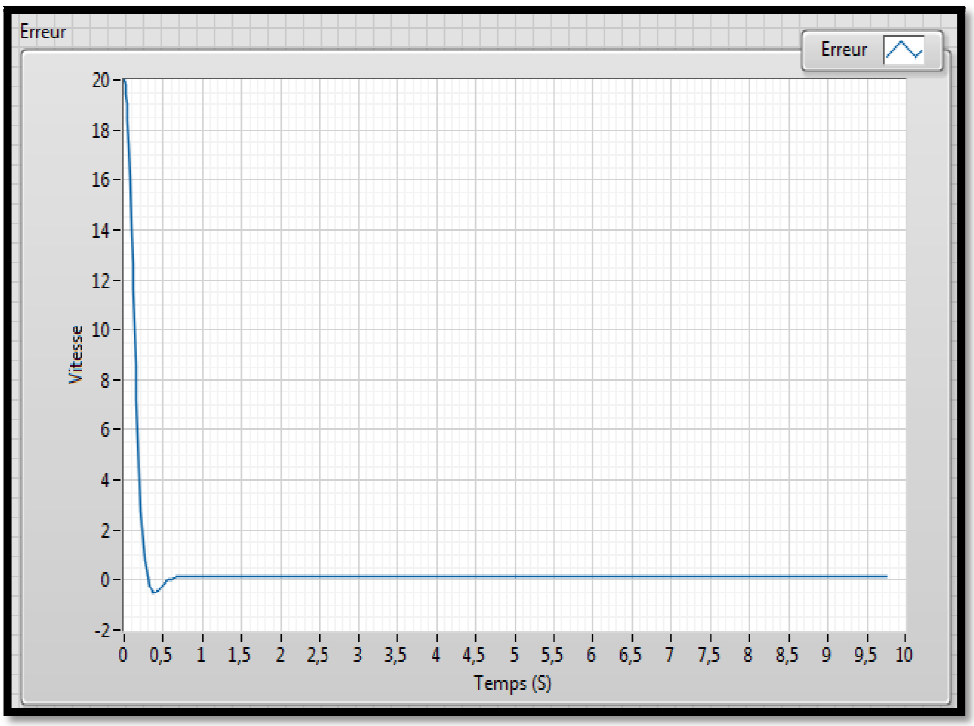


Figure 4-23 : Représentation de l'erreur

On remarque qu'à chaque fois on intervient sur les gains d'entrées de notre régulateur flou on agit sur la rapidité du système et à chaque fois qu'on intervient sur le gain de la sortie du système on agit sur la précision.

D'après les graphes représentés ci-dessus nous remarquons que la sortie du système converge vers la consigne avec un temps de monté qui est égale à 0.6s.

IV.5 Conclusion

Dans ce chapitre nous avons présenté la commande de la vitesse du moteur à courant continu par le contrôleur PID, l'algorithme de commande a été implémenter dans notre carte FPGA. Le choix des paramètres PID sont choisit après plusieurs teste d'ajustement. D'après les résultats obtenu lors des essayas on constat que le contrôleur PID est un contrôleurs robuste.

Liste des abbreviations

ASIC: Application-Specific Integrated Circuit.

CLB: Bloc Logique programmable

CLK: Clock

CPLD: complex programmable logic device.

DSP : Digital Signal Processor (Processeur de traitement numérique du signal).

FLC : fuzzy logic Controller

FPGA: Field Programmable Gate Array

GAL: Generic Array Logic.

IOB : input output bloc

Kt : Constante de couple

LabVIEW: Laboratory Virtual Instrument Engineering Workbench

LUT: Look Up Table.

LF : logique floue.

MCC: Machine à Courant Continu.

MLI : Modulation de largeur d'impulsion.

NI: National Instrument.

Open CL: Open Computing Language

Pa : puissance absorbé.

Perm : puissance électromagnétique.

PAL: Programmable Array Logic.

PLA: Programmable Logic Array..

PROM: Programmable Read Only memory

PWM : Pulse Width Modulation.

Perm : puissance électromagnétique.

PJ : Les pertes Joules.

PF : pertes fer.

PM : pertes mécanique

ROM : read only memory.

SPI: Serial Peripheral Interface

SRAM: mémoire vive statique (Static Random Access Memory)

VHDL: Very High Scale Integrated Circuit Hardware Description Language

Introduction Générale

Table des matières

Chapitre 01

Généralité

Chapitre 02

Initiation sur logiciel LabVIEW

Chapitre 03

La commande par logique floue

Chapitre 04

Réalisation de la commande du moteur MCC

Dédicace

Coclusion Générale

Remerciement

Annexes

Bibliographie

Liste des figures

Résumé

ملخص:

في هذه المذكرة،تصميم وتنفيذ نظام تحكم غامض ووحدة تحكم PID التقليدية للتحكم في سرعة وموضع محرك التيار المستمر. إستراتيجية بسيطة، تستند الفكرة العامة إلى التباين في دورة العمل لإشارة PWM، يتم استخدام مجموعة من القواعد غامض لتصميم وحدة تحكم غامض. تم تنفيذ الاستراتيجية المقترحة في LabView لبرمجة هدف بسهولة باستخدام دائرة F.P.G.A قابلة لإعادة التشكيل تنتجها National Instruments .

الكلمات الدلالية: نظام تحكم غامض ; ووحدة تحكم PID لإشارة pwmر; labVIEW; F.P.G.A

Résumé :

Dans ce mémoire, la conception et la mise en œuvre d'un système de contrôle flou et le contrôleur PID classique pour contrôler la vitesse et la position du moteur à courant continu. Une stratégie simple, l'idée générale est basée sur la variation du rapport cyclique du signale PWM,unensemble des règles floues est utilisé pour concevoir le contrôleur flou. La stratégie proposée a été mise en œuvre dansLabView afin de programmer facilement une cible dotée d'un circuit reconfigurable F.P.G.A produit par NationalInstruments.

Mots clés : Contrôleur flou ; PID Classique ; Signal PWM ; FPGA; LabVIEW.

Abstract:

In thismemory, the design and implementation of a fuzzy control system and the conventional PID controller to control the speed and position of the DC motor. A simple strategy, the general idea is based on the variation in the duty cycle of the PWM signal; a set of fuzzy rules is used to design the fuzzy controller. The proposed strategy has been implemented in LabView to easily program a target with a reconfigurable F.P.G.A circuit produced by National Instruments.

Keywords: Fuzzy control; Conventional PID; PWM Signal; FPGA; LabVIEW.

Remerciements

Avant tout, Elhamedo l'ALLAH Le tout puissant de nous avoir donné le courage, et la santé durant toutes ces années et que grâce à lui ce travail a pu être réalisé.

A travers ce modeste travail, je remercie mon encadreur Dr. FAS Mohamed Lamine pour son encadrement pendant celui-ci. Pour l'intéressante documentation qu'il a mis à notre disposition, pour ses conseils précieux et pour toutes les commodités qu'il nous a apportées durant notre étude et réalisation de ce mémoire.

Je remercie les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail. J'exprime également notre gratitude à tous les professeurs et enseignants qui ont collaboré à notre formation depuis notre premier cycle d'étude jusqu'à la fin de notre cycle universitaire.

Sans omettre bien sûr de remercier profondément à tous ceux qui ont Contribué de près ou de loin à la réalisation du présent travail.

Dédicace

C'est avec profonde gratitude et sincères mots, que nous dédions ce modeste travail de fin d'étude à nos chers parents ; qui ont sacrifié leur vie pour notre réussite et nous ont éclairé le chemin parleurs conseils judicieux. Nous espérons qu'un jour, nous pourrons leurs rendre un peu de ce qu'ils ont fait pour nous, que ALLAH leur prête bonheur et longue vie.

Mon binôme abdessattar, Je ne peux trouver les mots justes et sincèrespour t'exprimer mon affection et mes pensées, tu es pour moi un frère et un ami sur qui je peux compter.

Nous dédions aussi ce travail à nos frères et sœurs, nos familles, nos amis, tous nos professeurs qui nous ont enseigné et à tous ceux qui nous sont chers.

AHFIR Abdelaziz

[1] "Introduction au FPGA"

https://www.youtube.com/watch?v=XuexRKbes2w&t=154s (consulté le 10/05/2019).

[2] KARA, Kamel, "circuit logique programmable", Algérie : université Blida 1,2013.

[3] "Module labVIEW FPGA"

https://fr.scribd.com/doc/81089467/Projet-labview-fpga (consulté le 24/05/2019).

[4] "Advantages of FPGA, disadvantages of FPGA"

https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of .html?fbclid=IwAR1LUeR3u3H8gZS0K_Wn7Q3RaOl0ONMjFRpnIJwqfdlcIVhiqfivoryfr4

[5] Doru Ursutiu, Marius Ghercioiu, Cornel Samoila1, Petru Cotfas : "FPGA LABVIEW PROGRAMMING, MONITORING AND REMOTE CONTROL",mai 2009.

[6] " Le moteur électrique".

https://www.youtube.com/watch?v=mUpNwZKheBU

[7] FAS Mohamed Lamine, " actionneur", Algérie : université Blida 1, 2015.

[8] " quel est le principe de fonctionnement d'un moteur a courant continu ?".

https://www.youtube.com/watch?v=JV50zqHvqAM

[9] BENRABAH Mohamed, DEHEBI Ali, " Etude et implémentation sur DSP de la commande PID neuronal", Master automatique, Algérie université Blida1, 2016.

[10] " Les encodeurs pour la robotique mobile"

https://www.generationrobots.com/blog/fr/encodeurs-robotique-mobile/

[11] " asservissement d'un moteur électrique avec un capteur incrémental"

https://www.youtube.com/watch?v=prun24bIIEA&t=16s

[12] FAS Mohamed Lamine, Concepts et langage de programmation graphique « LabVIEW », Algérie : université Blida 1, 2016.

[13] Francis Cottet. Michel Pinard, Luc Desruelle, " labVIEW programmation et application", 3 éme édition,mai 2015.

[14] "Initiation sur LabVIEW "

http://hebergement.u-psud.fr/projetsdephysiquestatistique

[15] Gharmoul Ayoub, Zaabout Seif Elislam, " Réalisation d'une maison Intelligente à base d'Arduino commandée par LabVIEW", Licence professionnel Communication/transmition, Algérie Kolea école supérieur de transmission.

[16] SAHRAOUI Fodhil,KADA ALI Naziha , " Réalisation d'une imprimante 2D à base d'Arduino commandée par LABVIEW " , Licence professionnel Radio, Algérie Kolea école supérieur de transmission.

[17] FAS Mohamed Lamine, Commande intelligente, Algérie: université Blida1, 2018.

[18] BENZENATI Siham, RABIAI Meriem, " Commande floue d'un moteur à courant continu à excitation séparée", Master automatique, Algérie université Abderrahmane Mira Bejaia.

[19] TOUKITI Nacira, "Optimisation des systèmes photovoltaïque connectes au réseau par logique floue ", Ingénieur d'état en Electricité Industrielle, Algérie université de Biskra,2004.

[20] Ferhat Lahouazi,"mise en œuvre d'une stratégie de commande neuro floue application à un Pendule inversé", mémoire magister en Automatique, Université Mouloud Mammeri, Tizi-Ouzou, 2011.

[21] OUMAYA Mohamed,LIMAM Mohammed Lakhdar, " Commande par réseaux d'ondelette-floue", Master en automatique, Algérie Université Kasdi Merbah–Ouargla, 2012.

I.1. Introduction

La densité croissante des circuits programmables actuels, notamment des FPGA (Field Programmable GateArray), permet le prototypage rapide des circuits numériques à grande complexité. Aussi, il est possible de tester rapidement la validité de concepts architecturaux nouveaux:l'implémentation complète d'un processeur sur des circuits FPGA est aujourd'hui à notre portée, entraînant ainsi plus de possibilités d'évaluation que celles offertes par des simulateurs logiciels. Les évolutions technologiques et architecturales qui ont eu successivement lieu depuis le début des années 2000 ont fait de ces circuits de réels et rentables alternatives aux classiques ASIC. Avec ces évolutions c'est tout un nouveau domaine de l'électronique numérique qui s'est ouvert. Aujourd'hui les FPGA sont utilisés dans tous les domaines, des systèmes embarqués aux systèmes de communications, ils sont au cœur d'un important champ de recherche académique et industrielle.

Le but de ce chapitre est de découvrir les circuits FPGA en générale et on terminera cette première partie par une étude détaillée de la carte de développement « Digital Electronics FPGA board» de la firme National Instrument (NI), et dans la suite de ce chapitre on va s'appuis essentiellement sur la partie théorique de la machine a courant continu.

I.2. Les FPGA [1]

Le FPGA en anglais l'acronyme signifie« Field Programmable Gate Array », ou un circuit intégré pré diffusé programmable. Pour mieux comprendre on peut donner une autre définition pour les FPGA, ce sont des circuits intégré composé de réseaux de porte logique configurable par logiciel et donc reconfigurable a volonté, d'une façon simplifier on a donc de foule de bronche d'entrée sortie et entre les deux plusieurs portes logique que l'on configure pour accomplir des taches. Cette partie nous présente l'évolution, les caractéristiques principales et les applications typiques du FPGA.

I.3. Historique [1]

Avant d'aller plus loin dans le fonctionnement interne du FPGA regardant un peu plus son histoire, la logique programmable existe depuis le début des années soixante-dix mais au début il n'y'avait que quelques portes logiques dans un boitier et on les programmés qu' une seule fois comme des mémoires ROM(read only memory). Par la suite on va parler des PLA et des PAL pour les utiliser on écrit quelques équations logiques désire le tout est compilé

habituellement par un ordinateur et on brûle les fusible interne de notre circuit intégré pour configurer l'agencement des portes logiques si on constate une erreur dans les équations on jette la puce, il fallait recommencer tout simplement.**La figure 1-1** nous montre le schéma simplifié du PAL(PLD).

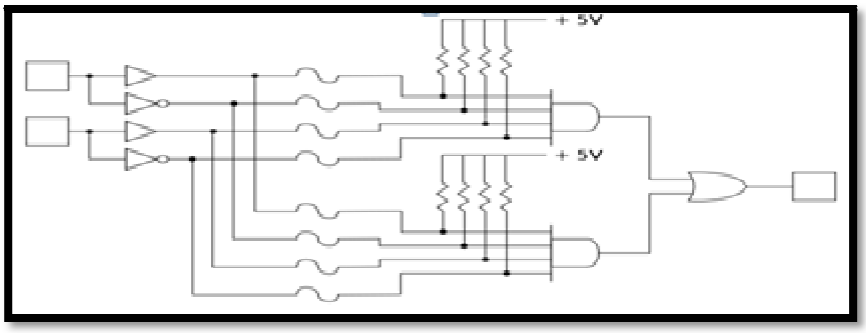


Figure 1-1: schéma simplifié du PAL(PLD).

L'arrivé du GAL qu'est effaçable électriquement et réutilisable pour améliorer un peu le processus de développement c'est « latticesemi-conducteurs» qui a lance le premier GAL.
La vraie révolution est apparu avec les PLD et plus tard les CPLD (ComplexProgrammableLogicDevice), ou dispositif de logique complexe programmable, ceci contient l'équivalent du plusieurs PAL relie entre eux par des interconnexions programmable le tout dans le même circuit intégré, les CPLD d'aujourd'hui peuvent remplacer des centaines de millier des portes logiques.

Au milieu des années quatre-vingt la compagnie américaine XILINXa lancé le premier FPGA qui a une architecture interne complètement différente celle-ci du PAL ou CPLD.

I.4. Architecture interne du FPGA [1]

Au lieu d'avoir un réseau de porte logique fixe et des interconnexions programmables les FPGA présente une série de bloc logique configurable et un réseau d'interconnexion aussi configurable. Les blocs logique peuvent être configuré pour exécuter des fonctions combinatoire complexe, dans la plus part des FPGA comprenne également des éléments de mémoire qui peuvent être des simple bascule de type Flip-Flop ou des blocs de mémoire plus complexe. Les FPGA peuvent être programmé pour implémenter différente fonctions logiques ce qui permet en informatique reconfigurable flexible similaire a celle exécuter par un logicielle informatique, finalement il y'a des blocs d'entrée sortie dont les caractéristiques électriques sont souvent programmable.

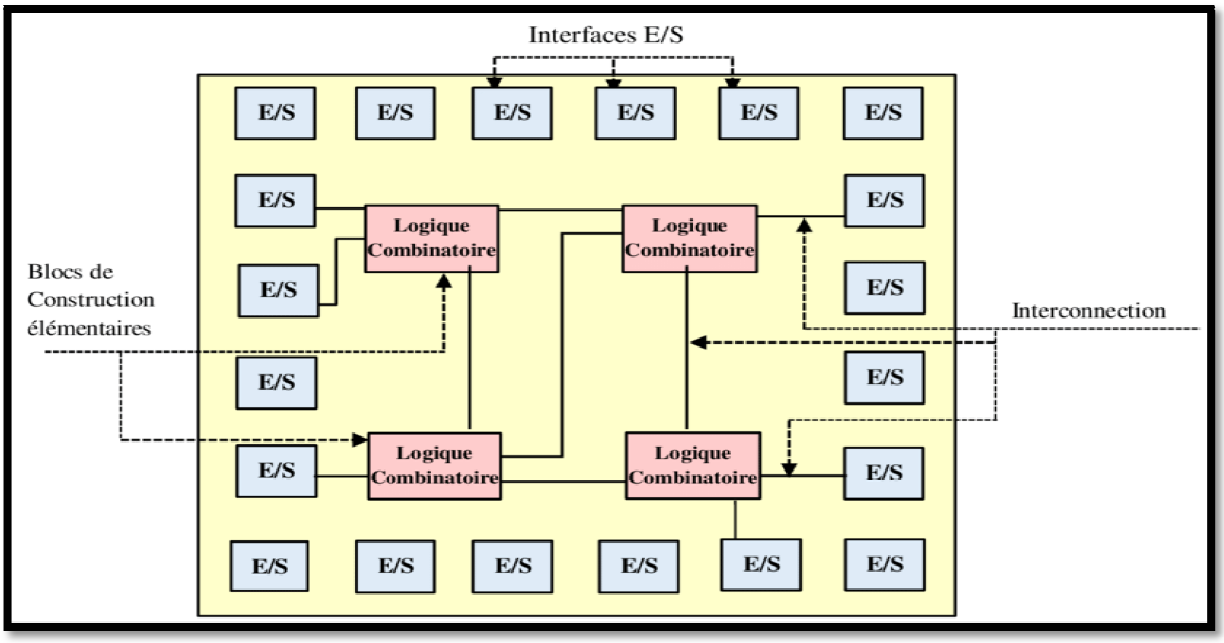


Figure 1-2 : Architecture interne du FPGA.

Regardent le bloc logique de plus près, en générale un bloc logique est constitué de quelques cellules logique une cellule typique consiste a une table de conversion ou une table de vérité qu'on appelle LUT (Look Up Table) a plusieurs entrées, le bloc contient aussi un additionneur complet et une bascule de type D. la table dans la figure 1-3 est divisée en deux tables a trois entrées, une table LUT consiste a une combinaison de porte logique programmé qui fourniront des sorties au niveaux spécifique en fonction des entrées fourni.

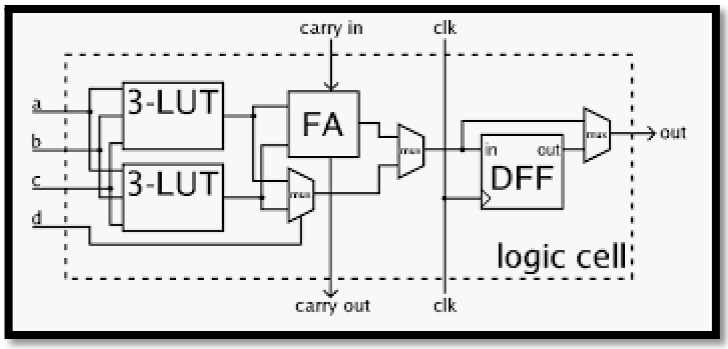


Figure 1-3 : Schéma simplifier d'une cellule FPGA.

En mode normale les sorties des tables contourne l'additionneur via un multiplexeur, en mode arithmétique les sorties des tables sont envoyée a l'additionneur. La sélection du mode est programmée dans le multiplexeur central. La sortie peut être synchrone donc cadencé par l'horloge (CLK) par l'entrée mise à la bascule ***D*** ou simplement asynchrone on contourne la bascule, la sélection synchrone-asynchrone est faite en fonction de la programmation du multiplexeur adroit. En pratique tout ou une partie de l'additionneur est stocké en tant que fonction afin de gagner la place.

I.5. Mécanisme de chargement de configuration dans un circuit FPGA [2]

Un FPGA n'est pas une pièce programmé en permanence, a chaque mise sous tension le FPGA doit charger sa configuration d'une façon ou d'une autre. Il existe donc des mécanismes de chargement de configuration. Le processus de chargement la conception dans la SRAM afin de programmer les fonctions des différents blocs et de réaliser leur interconnexion. Toutes les informations concernant la programmation des CLB et des IOB ainsi que l'état de la connexion sont stockée dans la mémoire de configuration. La configuration du FPGA est réaliser a la mise sous tension par le fichier binaire dont la taille varie en fonction du nombre de porte du circuit. A la mise sous tension la mémoire de configuration est effacée puis le circuit est initialisé.

Le circuit peut se configurer en série (bit par bit) ou en parallèle (octet par octet). Il peut être maitre ou bien esclave en cas de configuration de plusieurs boitiers. En mode maitre c'est lui qui fournit les signaux nécessaires à sa propre configuration mais il peut aussi configurer un ou plusieurs circuits esclaves montés en cascade. Le fichier de configuration contient successivement tout les fichiers de configuration des circuits à programmer.

La méthode privilégiée est l'emploi d'une mémoire flash série (PROM) externe au FPGA, dans ce cas le FPGA génère une horloge CLCK pour lire les bits en série c'est le mode série maitre. Souvent les FPGA ont un bus SPI pour connecter une simple puce de mémoire flash série.

En mode parallèle esclave depuis un processeur, c'est un microcontrôleur externe qui charge la configuration octet par octet. L'horloge est fournie par le microcontrôleur.

Il y'a une autre méthode de transfert la configuration est l'usage du bus JTAG via une interface JTAG/USB. C'est la méthode privilégiée durant le développement car c'est facile de faire le transfert à partir d'un PC, donc le bus JTAG est considéré comme un bus de teste car il donne l'accès tout les bronche FPGAa partir de l'intérieure. La figure 1-4 représente le chargement de configuration dans un circuit FPGA.

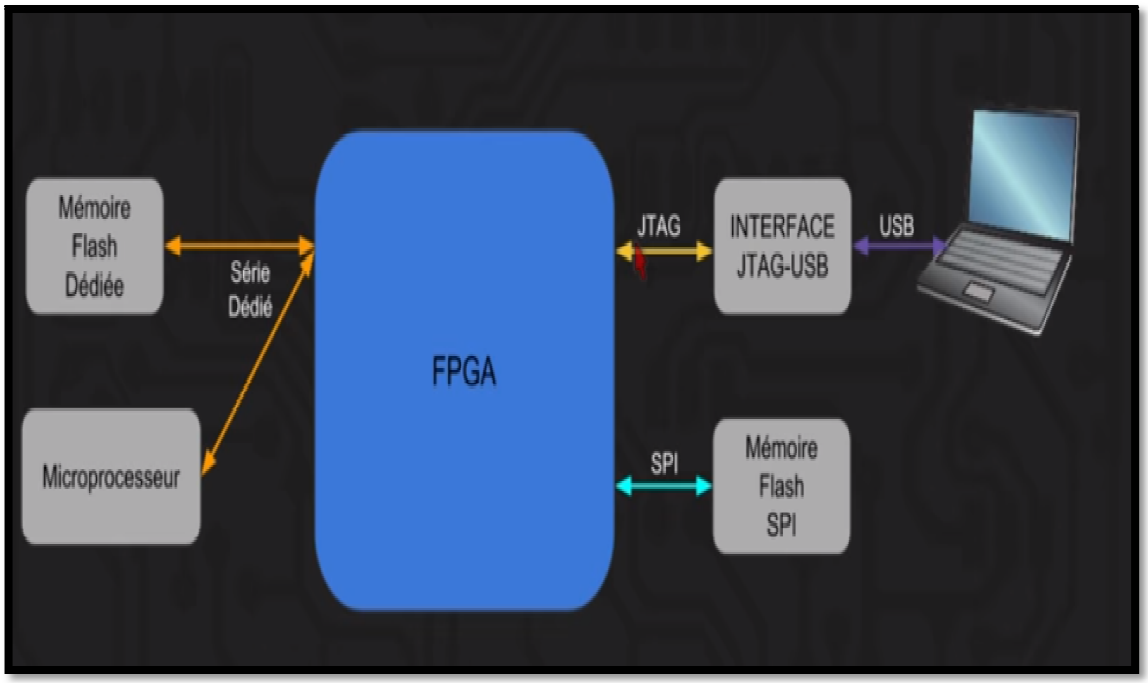


Figure 1-4 : Chargement de configuration dans un circuit FPGA.

I.6. Les étapes pour coder un FPGA [1]

Pour définir le comportement désiré d'un FPGA l'utilisateur fournit une conception soit dans un langage de description matérielle qu'on appelle HDL, soit avec une conception schématique. La forme de langage HDL est plus adapté au travaille de grand structure car il est possible de spécifier le comportement fonctionnelle de haut niveaux au lieu de dessiner chaque élément a la main par contre la saisie schématique peut permettre de visualiser plus facilement une conception et ces modules, plus récemment la conception open CL qui veut dire open Computing Languge est utilisé par les programmeurs pour tirer partie de performance de l'efficacité énergétique fourni par le FPGA comparé au microprocesseur. OpenCL permet aux programmeurs de développer du code dans langage de programmation. C'est a l'aide de logicielle électronique spécifique au FPGA une Netliste (liste de connexion logique) est généré a partir de code source, cette dernière est adapté a l'architecture réelle de FPGA ce processus appelle place and route et généralement exécuté par logiciel propriétaire de fournisseur de l'FPGA. L'utilisateur en suite validera les résultat du lieu et l'itinéraire place and route et les signaux dans le FPGA via une analyse de synchronisation une simulation et d'autres méthodes de vérification et de validation s'il y'a une erreur ou une correction nécessaire on retournera a la source ou a l'état de place and route, une fois le processus de conception et de validation et terminé une vérification finale est faite et le fichier binaire est généré encore une autre fois a l'aide du logicielle propriétaire de fournisseur de

FPGA, finalement ce fichier est utilisé pour configurer le F.PG.A.La figure 1-5 résume les étape de développement du FPGA.

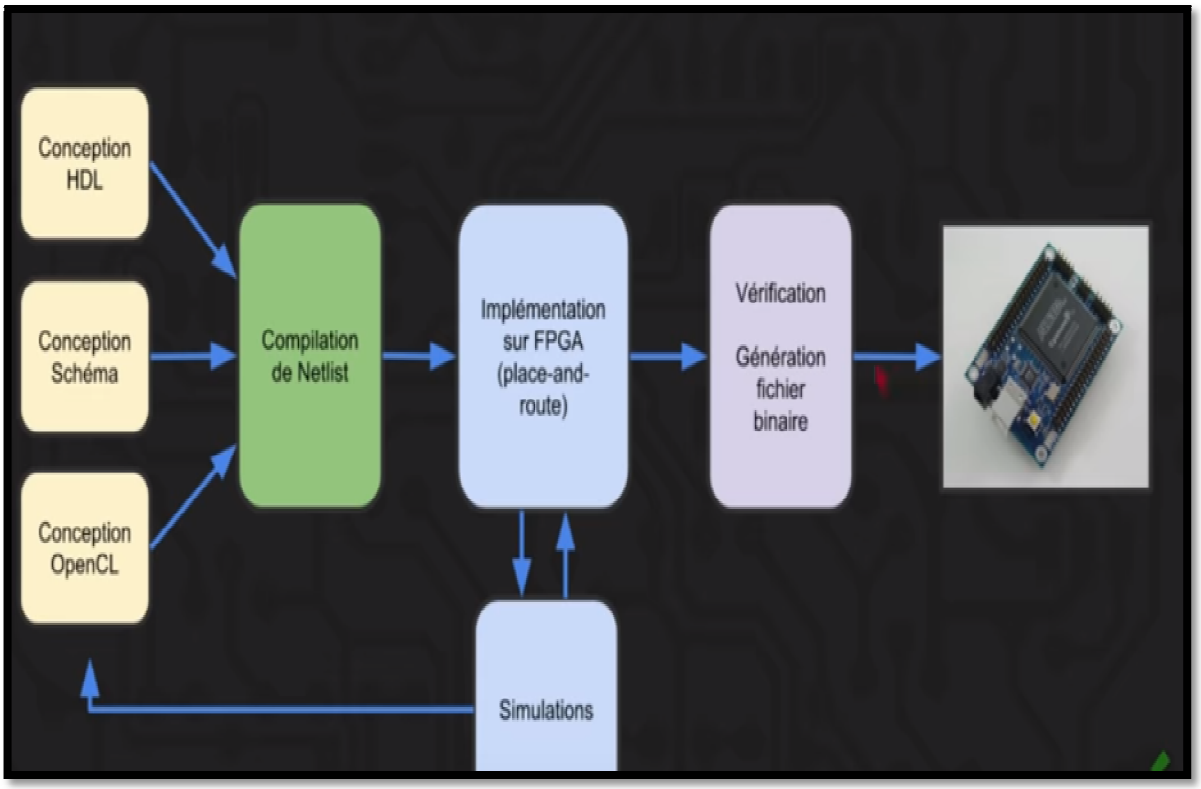


Figure 1-5 : Flux de développement du FPGA.

I.7. Les principaux atouts de la technologie FPGA [3]

Il y a cinq atouts de la technologie FPGA :

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

I.7.1Performances [3]

Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. BDTI, une importante société d'analyse et de « Benchmarking », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau

matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application.

I.7.2. Temps de mise sur le marché [3]

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, vous pouvez tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé. Par la suite, vous pourrez apporter les éventuelles modifications nécessaires à votre FPGA, en quelques heures au lieu de quelques semaines. Le matériel « sur étagère » actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cours de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux.

I.7.3.Coût [3]

Les coûts d'ingénierie non récurrents (NRE) des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC se justifie largement pour les OEM, par exemple, qui peuvent livrer des circuits par milliers chaque année. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps.

I.7.4Fiabilité [3]

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

I.7.5.Maintenance à long terme [3]

Comme nous l'avons vu, les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la préconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité.

Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. A mesure qu'un produit ou qu'un système évolue, vous pouvez y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit.

I.8. Les avantages et inconvénients du FPGA [4]

I.8.1.Les Avantages du FPGA [4]

Les avantages du FPGA sont les suivants :

- ✓ Les FPGA peuvent être programmés au niveau logique. Par conséquent, il peut implémenter un traitement plus rapide et parallèle des signaux. Ceci est difficile à exécuter par le processeur.
- ✓ Contrairement aux ASIC qui sont fixés une fois programmés, les FPGA sont programmables à tout moment au niveau logiciel. Ainsi, les circuits intégrés FPGA peuvent être reprogrammés ou réutilisés autant de fois que nécessaire. Les FPGA peuvent également être programmés à distance.
- ✓ Les circuits intégrés FPGA sont facilement disponibles et peuvent être programmés à l'aide du code HDL en un rien de temps. Par conséquent, la solution est disponible plus rapidement sur le marché.
- ✓ Contrairement à l'ASIC qui nécessite d'énormes NRE (dépenses non récurrentes) et des outils coûteux, le développement de FPGA est moins cher en raison d'outils moins coûteux et de l'absence de NRE.
- ✓ Dans la conception FPGA, les logiciels s'occupent du routage, du placement et de la synchronisation. Cela fait une intervention manuelle moindre. Le flux de conception élimine les tâches complexes et fastidieuses en matière de lieux et de routeurs, ainsi que les analyses de planification et de synchronisation.

I.8.2.Inconvénients du FPGA [4]

Les inconvénients du FPGA sontlessuivants :

- ✓ La programmation du FPGA nécessite la connaissance des langages de programmation VHDL / Verilog ainsi que des bases du système numérique. La programmation n'est pas aussi simple que la programmation en ***C*** utilisée dans un matériel à base de processeur. De plus, les ingénieurs doivent apprendre à utiliser les outils de simulation.
- ✓ La consommation électrique est plus importante et les programmeurs n'ont aucun contrôle sur l'optimisation de la puissance dans les FPGA. Aucun de ces problèmes dans ASIC.
- ✓ Une fois qu'un FPGA particulier est sélectionné et utilisé dans la conception, les programmeurs doivent utiliser les ressources disponibles sur le circuit intégré FPGA. Cela limitera la taille et les caractéristiques de la conception. Pour éviter une telle situation, un FPGA approprié doit être choisi au début.
- ✓ Les FPGA conviennent mieux au prototypage et à la production en petites quantités. Lorsque le nombre de FPGA à fabriquer augmente, le coût par produit augmente également. Ce n'est pas le cas avec la mise en œuvre ASIC.

I.9. La carte électronique FPGA de Digital Electroniques [5]

NIElvis est une conception pédagogique et plate-forme de prototypage de national instruments Corp. Basé sur le logiciel de conception des systèmes graphiques ni LabVIEW. La plate-forme est utilisée pour enseigner des concepts dans des domaines tels que l'instrumentation, circuits, contrôle, communication et conception intégrée de manière pratique et interactive. NIElvis contient une suite intégrée des 12 logiciels les plus instruments ;couramment utilisés multimètre numérique, oscilloscope, fonction générateur, compteur.

Un nouvel instrument, la carte électronique FPGA de Digital Electroniques a été ajouté à la plate-forme NI-ELVIS pour aider les éducateurs enseignent les concepts de la programmation FPGA.

Le Digital Electroniquesest un circuit deplate-forme de développement basée sur XilinxSpartan 3EFPGA. La clé du tableau de bord électronique numérique Les composants et fonctionnalités sont:

- FPGA Xilinx XC3S500E Spartan-3^E
- Jusqu'à 232 E/S utilisateur
- Boîtier FPGA 320 broches
- Plus de 10 000 cellules logiques

- PROM de configuration Flash de la plate-forme 4 Mbit
- 16 Mbits de Flash série SPI (ST Micro) pour FPGA Stockage de configuration
- Téléchargement / débogage de FPGA / CPLD sur carte USB embarqué interface
- Oscillateur d'horloge 50 MHz
- 6 connecteurs d'extension Digilent à 12 broches (PMOD)
- DAC (convertisseur numérique-analogique) basé sur SPI 4 canaux
- CAN (convertisseur analogique-numérique) basé sur SPI à 2 canauxavec gain programmablepréamplificateur
- Affichage à LED à 2 segments et 7 chiffres
- Codeur rotatif avec tige à bouton-poussoir
- Huit LED discrètes
- Huit interrupteurs à glissière
- Quatre boutons poussoirs
- Zone de planche à pain FPGA
- Zone de planche à pain NI-Elvis
- Zone de planche à pain à usage général
- Interface de connecteur NI-ELVIS
- Fiche du baril de puissance + 15VDC

La carte FPGA Digital Electroniques peut être utilisée commecarte enfichable dans la plate-forme d'enseignement NI-ELVIS. La carte est alimentée par la ligne d'alimentation + 15V générée parNI-ELVIS et il est connecté au PC hôte via un port USBCâble.

Figure 1-6 : Plate-forme NI-ELVIS II avec la carte FPGA Digital Electroniques.

La carte FPGA Digital Electroniques peut également être utilisé en tant que conseil autonome posé sur la table, il est alimenté adaptateur secteur de 15V externe (minimum 500mA) et est connecté au PC hôte via un câble USB.

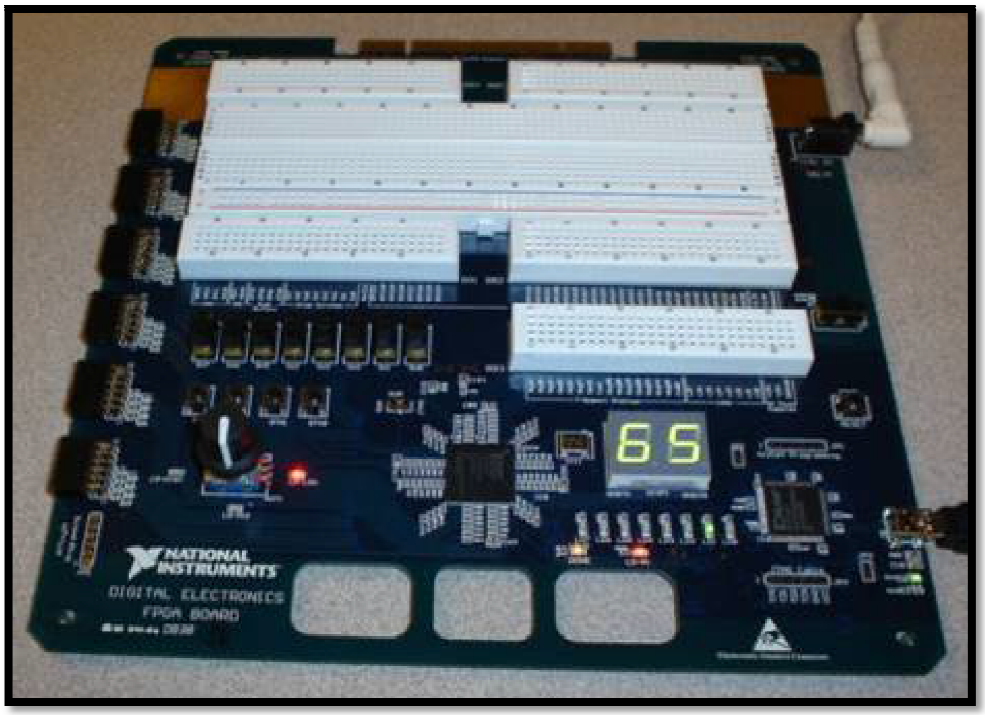


Figure 1-7 : Carte FPGA Digital Electroniques utilisée en mode autonome.

Le nouveau module NI LabVIEW FPGA utilise LabVIEWtechnologie intégrée pour cibler la porte programmable sur sitetableaux (FPGA). Avec le module LabVIEW FPGA, nous avons pu programmer le FPGA Spartan-3E sur la carte électronique FPGA sans matériel de bas niveaulangages de description ou conception au niveau du conseil.

Digital Electronics FPGA Board présente les caractéristiques suivantes:

- ✓ Périphériques d'E/S FPGA
- ✓ 6 connecteurs d'extension Digilent à 12 broches (PMOD)
- ✓ DAC (convertisseur numérique-analogique) basé sur SPI à 4 canaux
- ✓ CAN (convertisseur analogique-numérique) basé sur SPI à 2 canaux
- ✓ préamplificateur à gain programmable
- ✓ Affichage à LED à 2 segments et 7 chiffres
- ✓ Huit LED discrètes
- ✓ Huit interrupteurs à glissière
- ✓ Quatre boutons poussoirs

- ✓ Zone de planche à pain FPGA
- ✓ Zone de planche à pain NI-Elvis
- ✓ Zone de planche à pain à usage général

LabVIEW FPGA contient des objets graphiques nommés E/S élémentaires qui représentent une E/S physique FPGApériphériques de la carte FPGA Digital Electronics dans leschéma d'application (ou code). Un objet d'E/S FPGA peut être activé pour une certaine application en le plaçant dans leprojet clic droit sur le périphérique cible FPGA:

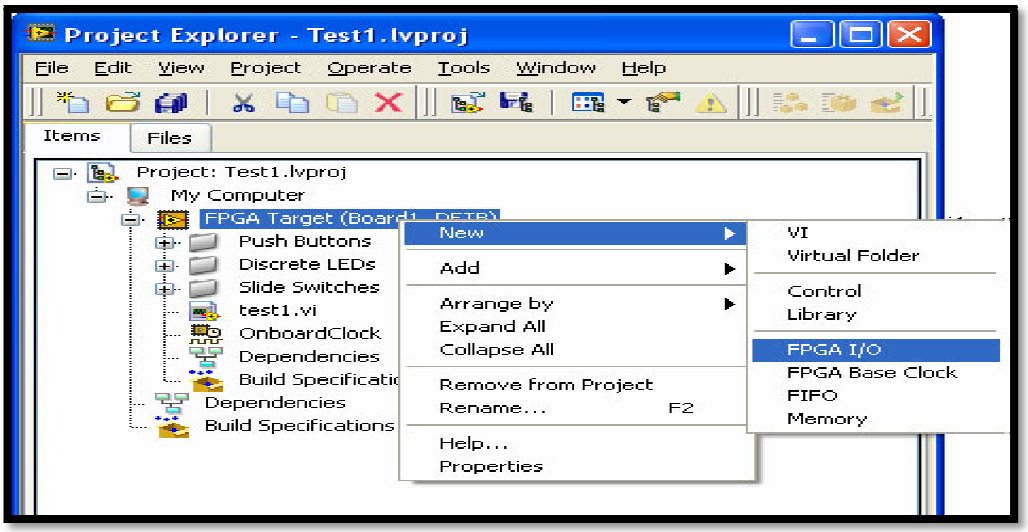


Figure 1-8 : Sélection d'E/S FPGA.

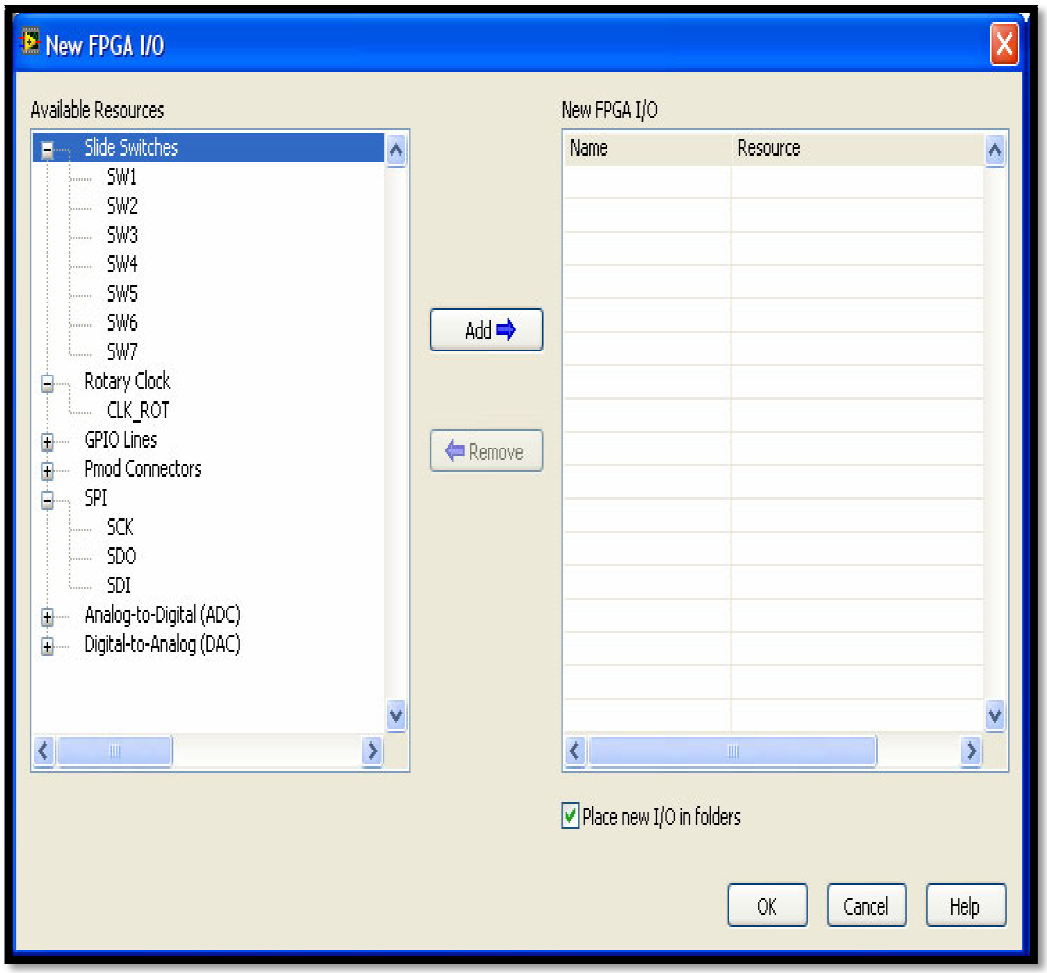


Figure 1-9 : Sélection d'E /S FPGA.

I.10. La Machine à Courant Continu (MCC)

I.10.1. Définition [6]

Les moteurs à courant continue sont les premiers convertisseurs électromécaniques, c'est-à-dire ils transforment l'énergie électrique en puissante rotation capable de faire tourner des machines comme des convoyeurs ou des pompes. Il est tellement plus fort qu'il peut soulever des objets deux fois plus lourds que lui à trois mètres dans les aires en une seconde, mais ce qui est vraiment incroyable ce qu'il n'y a aucune composante mécanique à l'intérieur du moteur. Pour créer cette puissante rotation le moteur DC utilise seulement des aimants.

I.10.2. Représentation schématique de MCC [7]

Le moteur à courant continu est représenté dans le schéma électrique par des symboles normalisées, la **figure 1-10** représente le moteur à courant continu en générale, la **figure 1-11** nous montre le moteur à inducteur bobiné et le moteur à aimant permanent.

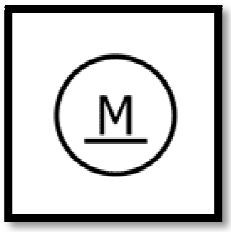


Figure 1-10: Symbole du moteur à courant continu en général.

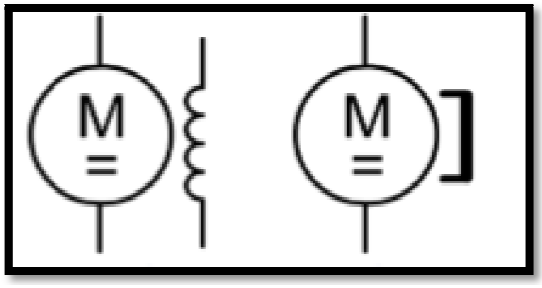


Figure 1-11 : Moteur à inducteur bobiné et moteur à aimant permanent.

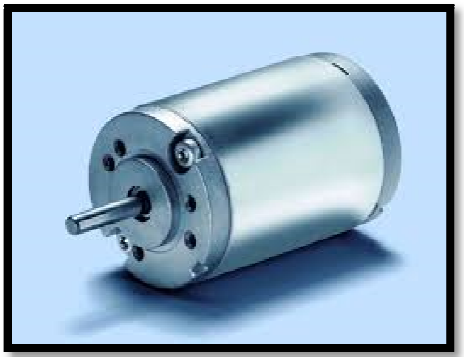


Figure 1-12 : Moteur à courant continu.

I.10.3. Morphologie [7]

Cette machine est constituée de deux composants principaux, qui ont le rôle le plus important dans le fonctionnement de la machine à courant continu sont l'inducteur et l'induit.

L'inducteur : appelé aussi le stator, l'inducteur est composé soit d'un aimant permanent, soit d'enroulement bobiné autour d'un élément immobile du stator. Il crée un champ magnétique dit statorique, c'est la partie fixe du moteur.

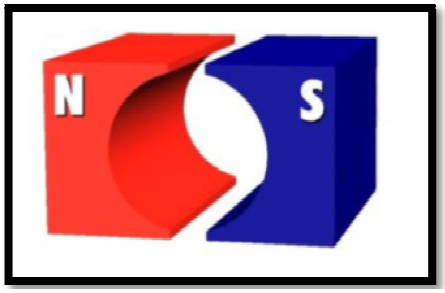


Figure 1-13 : l'inducteur.

L'induit : appelé aussi le rotor c'est la partie tournante du moteur. Le rotor cylindrique est composé de tôle isolée entre elles et munies d'encoches dans lesquelles sont réparties les conducteurs. Parcourus par un courant, ceux-ci créent un champ magnétique dit rotorique. [1]

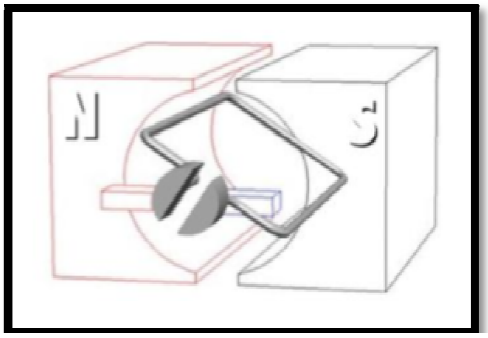


Figure 1-14 : l'induit.

Le collecteur : fixé à l'induit il est en contact avec les balais.

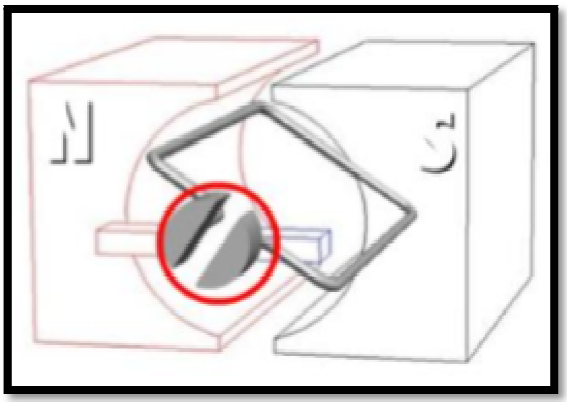


Figure 1-15 : Le collecteur.

Les balais : appelés aussi des charbons ils alimentent l'induit par le collecteur sur le quelle ils frottent.

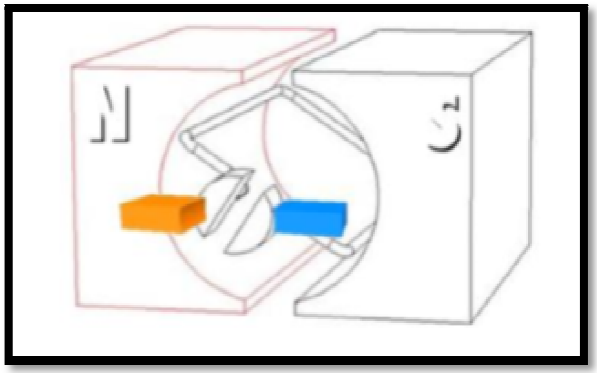


Figure 1-16 : Les balais.

I.10.4. Principe de fonctionnement [8]

Avant même d'aborder le principe de fonctionnement d'un MCC il est très important de comprendre ce qu'est la force de la place, c'est grâce à un aimant nous pouvons créer un champ magnétique et que on place a l'intérieure de ce champ magnétique un fil conducteur

traverse par un courant électrique, ce champ magnétique va créer une force qu'est appelée la force de la place. Ce phénomène est connu sous le nom de l'électromagnétique.

Pour connaitre la direction de cette force vous pouvez utiliser la règle des trois doigts de la main droite qui fonctionne comme ceci

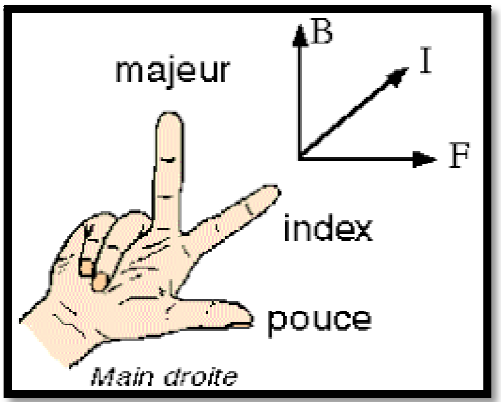


Figure 1-17 : La règle de la main droite.

Maintenant on passe à la suite, lorsqu'un bobinage (enroulement) est alimenté par un courant continu alors cela crée un champ magnétique de direction nord-sud on appelle ce bobinage le stator, le nord correspond au potentielle positif et le sud au potentielle négatif, en suite on place dans ce champ magnétique une spire capable d'effectuer un mouvement de rotation (c'est le rotor). Ce dernier laisse passer un courant électrique via un mécanisme assis ingénieux composé d'un collecteur et deux balais.

D'après la loi de la place tous conducteurs parcourus par un courant et placé dans un champ magnétique sont soumis à une force, alors la spire est soumise à deux forces de direction opposée entrainent le début de sa rotation. **La figure 1-18** représente le phénomène électromagnétique.

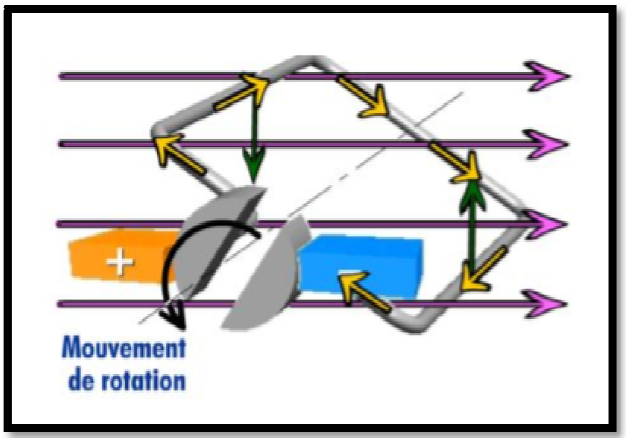


Figure 1-18 : Phénomène électromagnétique.

Le problème ici c'est que si le courant ne change pas de direction alors la spire restera en position horizontale mais toute la subtilité du MCC réside dans un détaille c'est la forme

du collecteur. Si vous faites bien attention vous pouvez observer que le collecteur en réalité est en deux demi-collecteur, c'est logique il faut bien que la spire soit connecté à un collecteur positif et un collecteur négatif. Ces deux demi-collecteur se sépare exactement là où la spire attient sa position horizontale, donc à cet instant précis le courant est inversé, le curant s'inverse la spire continu sa rotation dans le même sens.

I.10.5. Modélisation du moteur à courant continu [9]

La modélisation sert à établir le modèle mathématique d'un système dynamique linéaire ou non linéaire, ce modèle nous donne la relation entre l'entrée du système et la sortie du système pour qu'on puisse faire l'étude et l'analyse des systèmes, dans notre cas le moteur à courant continu peut être modélise comme un système dont l'entrée est la tension de commande de l'induit $\boldsymbol{u(t)}$ et la sortie la vitesse de rotation de l'arbre moteur $\boldsymbol{\omega_m(t)}$. L'induit est modélisé par une résistance en série avec une inductance et une force contre électromotrice. On donne dans la **figure 1.19** le modèle de connaissance du MCC.

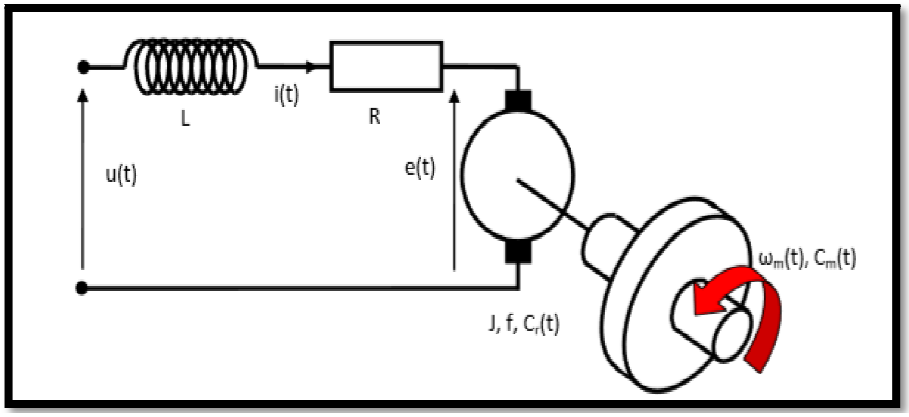


Figure 1-19: Modèle de connaissance du MCC.

En utilisant la loi des mailles :

$$\boldsymbol{u(t) = e(t) + R\, i(t) + L\, \frac{di(t)}{dt}} \qquad (1.1)$$

D'autre part l'équation de la force électromotrice est donnée par l'équation suivante : La force électromotrice E(t) est une tension induite par la variation de champ magnétique reçu par les bobinages. On retiendra que cette variation est proportionnelle à la vitesse de rotation

$$\boldsymbol{e(t) = K_e \omega_m(t)} \qquad (1.2)$$

L'équation de la dynamique de l'arbre moteur :

$$\boldsymbol{L\, \frac{d\omega_m(t)}{dt} = C_m(t) - C_r(t) + f\omega_m(t)} \qquad (1.3)$$

L'équation de l'électromagnétisme :

$$C_m(t) = K_t\, i(t) \qquad (1.4)$$

Remarque : en l'absence de perte K_t et K_e ont la même valeur

Avec :

$u(t)$: Tension du moteur [V]

$e(t)$: Force contre électromotrice du moteur [V]

$i(t)$: Intensité dans le moteur [A]

$C_m(t)$: Couple exercé par le moteur [N.m]

$C_r(t)$: Couple résistant sur l'axe moteur [N.m]

$\omega_m(t)$: Vitesse angulaire du moteur [rad/s]

R : Valeur de la résistance [Ω]

L : Valeur de l'inductance [H]

K_e : Coefficient de la force contre électromotrice [V/ (rad/s)]

J : Inertie équivalente ramenée sur l'arbre moteur [kg.m²]

f = 0,01 : Paramètre de « frottement fluide » total [N.m.s]

K_t : Constante de couple [N.m/A]

I.10.6. Aspect énergétique [7]

a. Puissance absorbée

La puissance absorbée est la puissance électrique utilisée par le moteur, vue depuis l'extérieur du moteur. Si u est la tension d'alimentation du moteur et i est le courant qui le traverse, alors:

$$P_a(t) = ui \qquad (1.5)$$

b. Puissance électromagnétique

C'est la puissance que le moteur est théoriquement capable de convertir, lorsque l'on néglige les pertes.

$$P_{em}(t) = e \times i(t) \qquad (1.6)$$

c. Puissance utile

La puissance utile est la puissance mécanique réellement récoltée à la sortie du moteur. Si ω est la vitesse de rotation du rotor, et Cu le couple généré par le moteur, alors :

$$P_u = \omega \times C_u\, i \qquad (1.7)$$

d. Pertes et rendement

Lors de sa transformation, toute la puissance n'est donc pas transmise; il y a des pertes :

d.1. Les pertes Joules

C'est la puissance dissipée par la résistance électrique du rotor (bobinage...).

$$P_j = R \times i^2(t) \qquad (1.8)$$

d.2. Les pertes collectives

Cette puissance perdue et la somme des pertes fer et les pertes mécanique:

d.3. Les pertes fer PF

Ce sont les pertes dues aux effetsdes champs magnétiques sur les noyaux du rotor (autre que le bobinage). Les courants de Foucault créent, entre autre, un échauffement du noyau, et donc une perte sous forme de chaleur.

d.4. Les pertes mécaniques PM

Ce sont les pertes liées aux frottements du rotor, notamment dans la liaison pivot, dans les frottements de l'air. Les pertes collectives peuvent être déterminées par un essai à vide (sans charge) : la puissance de sortie est donc nulle; les pertes collectives correspondent à la puissance électromagnétique. Tout cela est représenté clairement dans la figure 1.20.

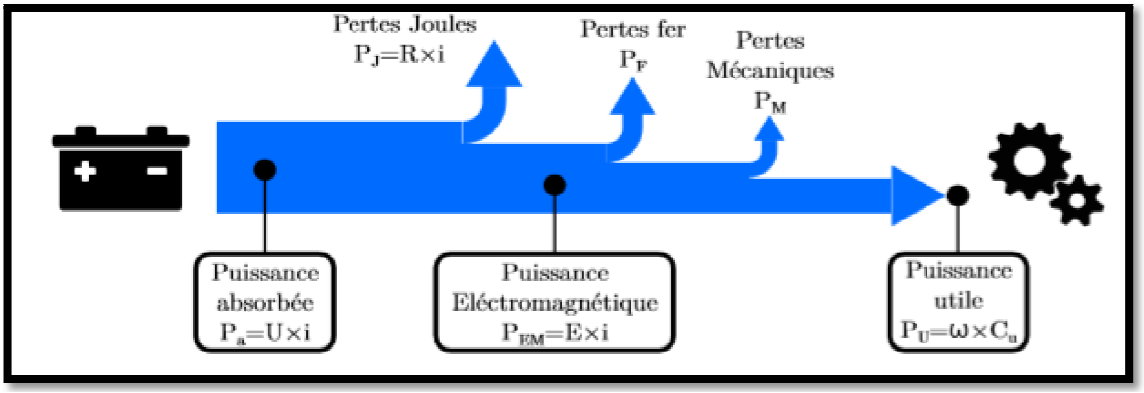


Figure 1-20 : Bilan énergétique dans la MCC.

d.5. Rendement

On appelle rendement du moteur à courant continu, le coefficient η qui est donné par l'expression suivante. Sa valeur numérique est limitée sur cet intervalle [0 1].

$$\eta = \frac{P_u}{P_a} \qquad (1.9)$$

I.11. Les encodeurs [10]

Un encodeur est un dispositif électromécanique qui génère un signal électrique en fonction de la position ou du déplacement de l'élément mesuré. Dans la commande des MCC, les encodeurs rotatifs sont utilisés pour mesurer le déplacement (sens et vitesse de rotation du) du moteur a courant continu.

I.11.1. Les différents types d'encodeurs [10]

On distingue deux grands types des encodeurs incrémentaux et absolus. Un encodeur incrémental génère un signal permettant de déterminer sens et vitesse de rotation tandis qu'un encodeur absolu génère une information absolue indiquant la position du capteur

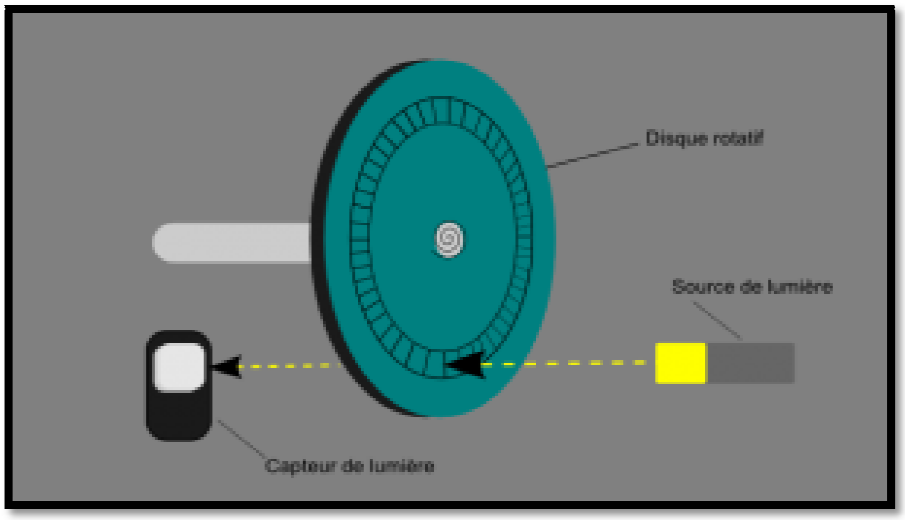


Figure 1-21 : Encodeur incrémental.

I.11. 2. Principe de fonctionnement de l'encodeur incrémental [11]

Dans notre application on a besoin d'utiliser un encodeur de roue incrémentale son principe de fonctionnement est plutôt simple, une diode infrarouge émet un rayon celui-ci est reçu par un phototransistor et lorsque la roue tourne, elle coupe le faisceau et crée des interruptions détectée par le phototransistor ainsi avec une roue codeuse de 500 révolution on peut compter 500 impulsion par tour. Il existe des roues incrémentale notamment est les plus utilisées celle qui ont deux voie grâce au déphasage (90°) entre les interruptions elles permettent de connaitre le sens de rotation de la roue.

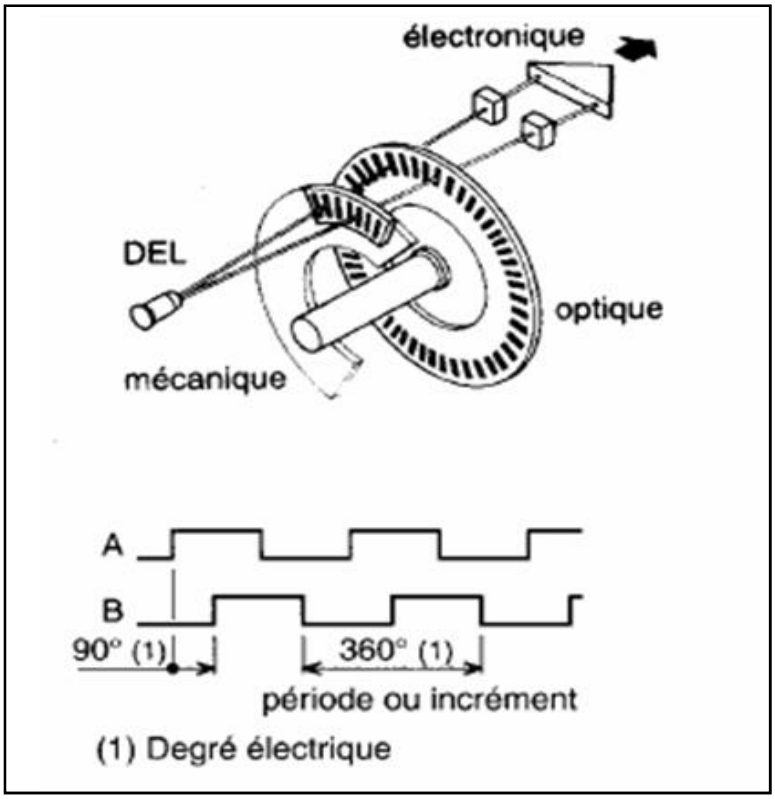


Figure 1-22 : Encodeur incrémental en quadrature.

Les signaux électriques obtenus par l'encodeur à deux voies sont illustré dans la figure suivante.

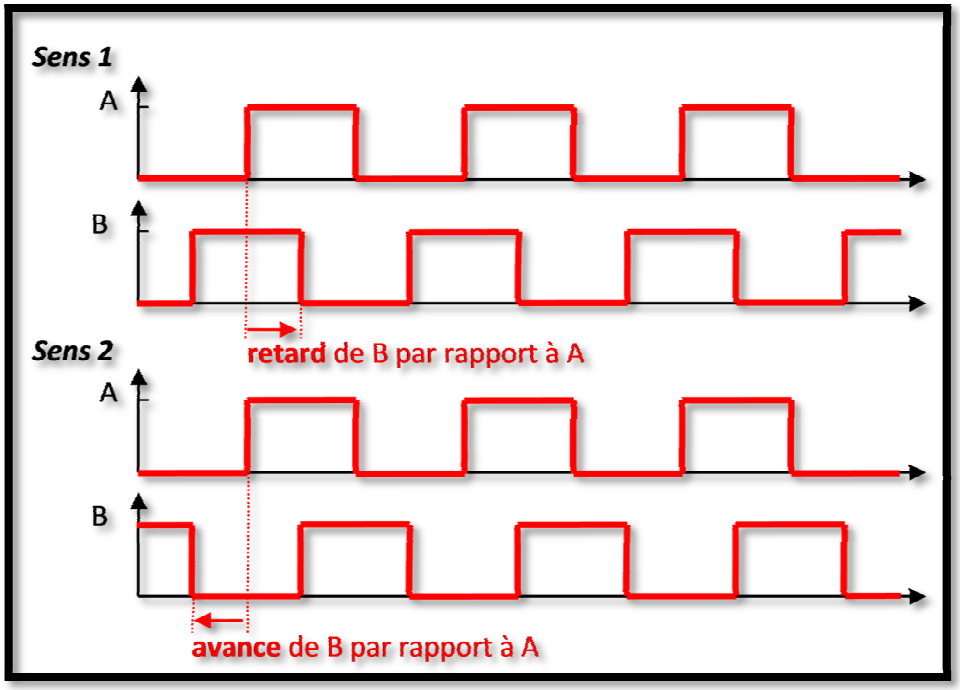


Figure 1-23 : Les signaux électriques de la voie A et B.

I.12. Conclusion

Dans ce chapitre nous avons présenté les matériels utilisés dans notre projet, tel que la carte FPGA, le moteur à courant continu et leur capteur de vitesse et de position.

Dans le début de ce chapitre, nous avons fait une étude des circuits logique programmable en commençant par sont historique puis on s'est base beaucoup plus sur les circuits logique programmable de type FPGA en général et par la carte électronique FPGA de Digital Electroniques en particulier, Les FPGA permettent aux systèmes d'être configurés et reconfigurés pour de nombreuses applications.

Dans la seconde partie, une étude théorique consacrée sur le moteur à courant continu et le capteur incrémental, cette étude nous permette d'établir le modèle mathématique du MCC.
Le chapitre suivant est réservé pour définir l'environnement de programmation LabVIEW qui est utilisé pour la programmation de notre FPGA.