Université Mohamed Boudiaf - M'sila

FACULTÉ DE TECHNOLOGIE

DEPARTEMENT DE GENIE ELECTRIQUE

Numéro de série :……
Numéro d'inscription : **DGE/04/16**

Thèse

Présentée pour l'obtention du diplôme de

DOCTORAT EN SCIENCES

Spécialité: Génie Electrique
Option: Génie Electrique

THEME

Implémentation des techniques d'intelligence artificielles sur FPGA en vue de contrôle des systèmes d'entrainements électriques

Présenté Par

AIB Abdelghani

Soutenue le : 15/01/ 2022

Devant le jury composé de:

Nom & Prénom	Grade	Etablissement	Qualité
RAHALI Hilal	MCA	Uni. de M'sila	Président
KHODJA Djalal Eddine	Professeur	Uni. de M'sila	Rapporteur
BENYETTOU Loutfi	MCA	Uni. de M'sila	Co- Rapporteur
MAROUANI Khoudir	Professeur	EMP. Alger	Examinateur
KHEMLICHE Mabrouk	Professeur	Uni. de Sétif	Examinateur
LATRACHE Samia	MCA	Uni. de Sétif	Examinateur

Année Universitaire : **2021/2022**

الملخص

الهدف الرئيسي من هذا العمل هو تحسين أداء التحكم في الماكينة غير المتزامنة باستخدام العديد من أدوات التنفيذ والمحاكاة التكنولوجية مثل Xilinx ISE , Xilinx System Generator ,Matlab / Simulink و الدوائر القابلة للبرمجة FPGA.

إن تنفيذ وحدات التحكم التقليدية والذكية (الضبابية والعصبية) على مصفوفة البوابات القابلة للبرمجة (FPGA) يوفر العديد من المزايا مثل: إعادة برمجة هذه الدوائر، أداة برمجية عملية، كفاءة عالية، وكثافة تكامل عالية جدًا.

علاوة على ذلك، تُعرف الآلة غير المتزامنة بنموذجها المعقد و غير الخطي . تمثل وحدات التحكم القائمة على الذكاء الاصطناعي حلاً فعالاً للمشاكل المرتبطة بالتحكم المباشر للعزم الكهرومغناطيسي في الآلة غير المتزامنة مثل تموجات العزم الكهرومغناطيسي وتموجات التدفق وتردد التبديل. تحقيقا لهذه الغاية، تم إعادة صياغة خوارزميات التحكم المقترحة في هذا العمل وإعادة هيكلتها من أجل الحصول على بنية أجهزة بسيطة ومنظمة بشكل جيد وبأقل قدر من الموارد على بطاقة مصفوفة البوابة القابلة للبرمجة. أخيرًا، تم التحقق من صحة البنى والمخططات المقترحة من خلال عملية المحاكاة المشتركة للأجهزة بين بطاقة ML402 المزودة بدائرة Virtex-4 FPGAمن نوع Xilinx وبرامج المحاكاة والتنفيذ XSG, Matlab / Simulink.

الكلمات المفتاحية : تقنيات الذكاء الاصطناعي، مصفوفة البوابات القابلة للبرمجة الميدانية،التحكم للعزم الكهرومغناطيسي ، الآلة غير المتزامنة ، المحاكاة المشتركة للأجهزة.

Abstract

The main goal of this work is to improve the performance of the asynchronous machine control using several technological implementation and simulation tools namely: Matlab / Simulink, Xilinx System Generator, Xilinx ISE and Xilinx ISE Simulator and circuits reconfigurable FPGAs.

However, the implementation on FPGA of conventional and intelligent controllers (fuzzy and neural), provides advantages, namely: the reprogramming of these circuits, a flexible software tool, good performance, and very high integration density.

Moreover, the asynchronous machine is known by its complex and non-linear model. Artificial intelligence based controllers represent an effective solution to problems associated with DTC control of MAS such as torque ripples, flux ripples and switching frequency. To this end, the control algorithms proposed in this work have been reworked and restructured in order to have simple hardware architecture, well structured and with the minimum of resources on the FPGA card. The modeling and simulation of this diagram are done in Matlab / Simulink with Xilinx System Generator and Xilinx ISE Simulator. Finally, the validation of the proposed architectures was carried out by a Co-Simulation hardware process between the ML402 card equipped with a Virtex-4 FPGA circuit of Xilinx type and XSG under Matlab / Simulink.

Keywords: Artificial intelligence techniques, FPGA (Field Programmable GateArray), DTC control, Asynchronous Machine (MAS), Hardware Co-Simulation.

Résumé

Le but principal de ce travail est l'amélioration des performances de la commande de la machine asynchrone en utilisant plusieurs outils technologiques d'implémentation et de simulation à savoir : Matlab/ Simulink, Xilinx System Generator, Xilinx ISE et Xilinx ISE Simulator et les circuits reconfigurables FPGA.

Cependant, l'implémentation sur FPGA des contrôleurs classiques et intelligents (flou et neuronal), fournit des avantages à savoir : la reprogrammation de ces circuits, outil logiciel souple, bon rendement, et densité d'intégration très élevée.

Par ailleurs, la machine asynchrone est connue par son modèle complexe et non linéaire. Les contrôleurs à base d'intelligence artificielle qui représentent une solution efficace face aux problèmes associés à la commande DTC de la MAS telles que les ondulations de couple, les ondulations de flux et la fréquence de commutation. A cet effet, les algorithmes de commande proposés dans ce travail ont été retravaillés et restructurés dans le but d' en avoir une architecture hardware simple, bien structuré et avec le minimum des ressources sur la carte FPGA. La modélisation et la simulation de ce schéma sont faites en Matlab/ Simulink avec Xilinx System Generator et Xilinx ISE Simulator. Enfin, La validation des architectures proposées a été effectuée par un processus de hardware Co-Simulation entre la carte ML402 dotée d'un circuit FPGA Virtex-4 de type Xilinx et XSG sous Matlab/ Simulink.

Mots clés: Les techniques d'intelligence artificielle, FPGA (Field Programmable Gate Array), La commande DTC, Machine Asynchrone (MAS)**,** Co-Simulation hardware.

Table des Notations et symboles

ASIC	Application Specific Integrated Circuits
FPGA	Field Programmable Gate Array
LUT	Lock UpTable
RAM	Random Access Memory
ROM	Read Only Mémory
SRAM	Static Random Access Memory
VHDL	Very High Speed Integrated Circuit Hardware Description Langage
DSP	Digital Signal Processor
CLB	Configurable Logic Block
SOC	System On Chip
DTC	Direct Torque Control
MAS	Machine Asynchrone
RNA	Réseau de Neurones Artificiels
CORDIC	Coordinate Rotation Digital Computer
XSG	Xilinx System Generator
DCM	Digital Clock Managers
IOB	Input Output Block
dSPACE	Digital Signal Processing and Control engineering
HIL	Hardware In the Loop
RMS	Root Mean Square
FFT	Fast Fourier Transform
PWM	Pulse-Width Modulation
IM	Induction Motor
$V_{s\alpha}, V_{s\beta}$	Les tensions statoriques dans le plan α-β
$I_{s\alpha}, I_{s\beta}$	Les courants statoriques dans le plan α-β
$\emptyset_{s\alpha}, \emptyset_{s\beta}$	Les flux statoriques dans le plan α-β
$\|\|\emptyset_s\|\|$	L'amplitude du flux statorique
$\theta_{\emptyset_s}$	L'argumente du flux statorique
T_e	Le couple électromagnétique
I_{sa}, I_{sb}	Les courants statoriques dans le plan abc
T_{eref}	Le couple électromagnétique de référence
$\emptyset_{sref}$	Le flux statorique de référence
N	Le numéro de Secteur
S_a, S_b, S_c	Les commandes de commutation
ω	La vitesse
ω_{ref}	La vitesse de référence
R_s	La résistance statorique
Ts	Temps d'échantillonnage
P	Le nombre des pairs des pôles de la machine
$\bar{V}_s$	Le vecteur commande de commutation

LISTE DES FIGURES

LISTE DES TABLEAUX

TABLE DES MATIERES

Introduction générale

Chapitre 1: Les techniques d'intelligence artificielle et leurs implémentations sur FPGA

Chapitre 2: Méthodologie d'implémentation des commandes numériques sur FPGA

Chapitre 3: Implémentation de la commande DTC à base des techniques intelligentes sur FPGA

Chapitre 4: Résultats, Simulations et Validations des architectures d'implémentation sur FPGA

Conclusion générale

Bibliographie

Introduction générale

Introduction générale

Grâce au développement technologique de la micro-électronique et de l'électronique de puissance, le domaine d'entraînement électrique à vitesse variable, a connu ces dernières années une évolution remarquable. Ceci est en faveur de la machine asynchrone (MAS), qui lui a permis d'être la plus utilisée dans l'industrie pour les entraînements à vitesse variable.

Dans le souci d'une amélioration continue des performances relatives à la commande des machines asynchrones, des techniques de plus en plus laborieuses sont vu le jour passant par la commande basant sur le rapport Volt/Hertz (scalaire), la commande par orientation du flux (vectorielle) et enfin la commande directe du couple DTC. L'évolution des techniques de commande appliquées aux moteurs asynchrone sont directement reliée aux avancées réalisées dans les domaines de micro-électronique, de l'électronique et de l'électronique de puissance. La complexité dont le model des moteurs asynchrones fait figure, la nécessité d'assurer un contrôle découplé des grandeurs électriques assurant le fonctionnement du moteur. En plus, les algorithmes élaborées sont appliqués afin de nous permettre d'avoir le plus de contrôle possible, et c'est dans cette optique que la DTC a été améliorée [11].

Par ailleurs, la commande directe du couple développée par TAKAHASHI et DEPENBROCK [1-2] est très populaire en industrie et intéresse beaucoup les scientifiques dans l'entrainement à vitesse variable [3]. Toutefois, cette commande présente deux importunités majeures : la fréquence de commutation est strictement imprévisible d'une part, et d'une autre part, les ondulations au niveau des amplitudes du flux et du couple restent non maitrisables dans tout l'intervalle de vitesse du fonctionnement considérée [4]. En plus, ces ondulations du couple créent des bruits et des vibrations additionnelles, ce qui provoque la faiblesse de l'arbre de la machine en rotation [5]. Pour réduire l'influence de ces phénomènes sur la durée de vie de la machine électriques on opte pour les techniques intelligentes dans l'optique d'améliorer la commande.

En outre, l'avancement technologique en conception hardware telles que les ASICs (Application Specific Integrated Circuit) et les FPGAs (Field Programmable Gate Array) et en micro-électronique permettrai des applications en temps réel gérées par des techniques intelligentes [6-7]. Cette technologique peut être utilisée pour l'implémentation numérique des lois de commande sur des cibles hardwares [8]. En plus, les avantages d'utilisation de ces circuits sont multiples : les traitements parallèles pour la minimisation du temps d'exécution, la confidentialité de l'implémentation et la rapidité du prototypage sur FPGA. D'autre part, les applications basées sur des techniques intelligentes nécessitent des procédés lourds en termes de temps de calcul et coûteuses en termes des ressources hardwares [9].

En fait, le développement de la technologie des semi-conducteurs, la micro-électronique et l'évolution des solutions de contrôle rapide tels que les DSP et les FPGA, permettent d'effectuer des commandes très complexes de la machine asynchrone malgré la non-linéarité de leur modèle mathématique [10].

Il est à signaler que les quantités des ressources hardwares dans un circuit FPGA sont limitées, des efforts doivent être fait pour minimiser la consommation de ces ressources à travers une méthodologie d'implémentation présentée dans cette thèse.

Cette méthodologie est fondée sur l'illustration d'un compromis qui réalise une optimisation de l'architecture hardware avec la préservation des performances de système d'entrainement électrique constitué d'un onduleur de tension et une machine asynchrone.

Dans ce contexte, l'objectif de ce travail est d'améliorer les performances de la commande DTC à base des techniques intelligentes vis-à-vis la DTC classique à base des comparateurs à hystérisés et la table de commutation, pour la commande des machines asynchrones. Ces commandes sont implémentés sur un circuit logique programmable de type FPGA tout en minimisant les ressources hardwares occupées dans la carte FPGA en utilisant les outils académiques disponibles (Matlab/ Simulink, Xilinx Systhem Generator, Xilinx ISE, ModeISim). Ensuite, Après les étapes de développement, synthèse et simulation sous Matlab/ Simulink avec Xilinx Systhem Generator et Xilinx ISE les architectures proposées sont validées par une procédure de hardware Co-Simulation avec le kit de développement ML402 (basé sur circuit FPGA de type Xilinx Virtex-4).

Ce travail de thèse est divisé en quatre chapitres organisés comme suit:

Le premier chapitre fournit une étude complète sur les techniques d'intelligence artificielle et les contraintes de leurs implémentations sur des circuits logiques programmables de type FPGA.

Au deuxième chapitre, nous allons exposer les FPGAs, leurs Contributions dans la commande des systèmes électriques et la méthodologie de conception par prototypage " Hardware in the loop".

Au troisième chapitre, nous allons présenter l'implémentation de la commande DTC et leurs versions à base des techniques intelligentes sur FPGA.

Nous consacrons le dernier chapitre à la validation des architectures proposées par hardware Co-Simulation.

Cette thèse se termine par une conclusion générale sur l'ensemble de nos travaux de recherches, en présentant les perspectives proposées.

Chapitre 1:

Les techniques d'intelligence artificielle et leurs implémentations sur FPGA

1.1. Introduction

Le développement des techniques d'intelligence artificielle se fait à travers les procédés par lesquelles l'homme essaye de copier leurs modes de raisonnement et de comportement. La logique floue et les réseaux de neurones représentent les approches intelligentes les plus populaires et leurs usages sont multiples du traitement de l'image à la gestion financière, en passant par les domaines électrotechniques et industriels. Généralement, elles sont utilisées pour résoudre les problèmes d'optimisation, de classification, d'identification, de régulation industrielle, de détection des défauts et la prise des décisions **[11].**

Sachant que la machine asynchrone et l'onduleur associé ont des difficultés dans leur étude et leur commande, nous proposons que les techniques d'intelligence artificielles puissent apporter des solutions pour ces problèmes.

Par conséquent, nous avons proposé dans un premier temps de présenter ces techniques et de clarifier les moyens les plus simples de les mettre en œuvre.

Notre travail se limite principalement à l'objectif fixé ici : la commande directe de couple de la machine asynchrone, et d'évaluer les avantages liés à l'utilisation des techniques intelligentes pour une commande DTC implémentée sur un circuit logique programmable de type FPGA.

1.2. Logique floue

1.2.1. Définitions et principe de la logique floue

L'étude de la logique floue basée principalement sur les théories des ensembles flous proposés par Zadeh [12]. En parallèles avec des formalismes mathématiques fortement développés, nous avons abordé la présentation de la logique floue d'une manière intuitive. Les concepts de courant moyen ou de température faible sont partiellement pénibles à spécifier de manière précise. Alors, on doit fixer des seuils pour considérer que l'on qualifie tel ou tel attribut en fonction de la valeur de la proposition par rapport à ces seuils (figure 1.1) **[11].** Le degré d'appartenance de la variable température à l'ensemble "moyen" peut être exprimé par "degré de vérité "de la proposition" la température est moyenne".

Dans la logique booléenne, les valeurs des variables ne peuvent prendre que deux valeurs (0 ou 1). La variable température par exemple peut être : élevée, moyenne ou faible exclusivement. Elle ne peut pas avoir deux qualificatifs à la fois.

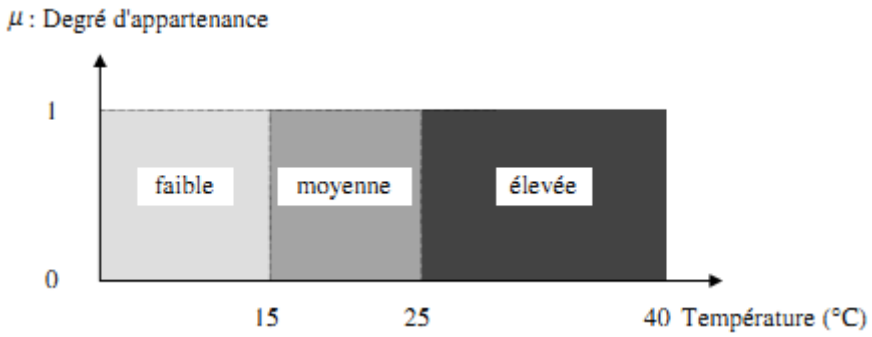


Figure 1.1 Les ensembles dans la logique booléenne

Dans le raisonnement flou, le degré d'appartenance présente une fonction mathématique qui peut avoir une valeur réelle comprise entre 0 et 1.

Par exemple la fonction μ_{faible} (T) permet de quantifier l'appartenance de la variable température à l'ensemble floue faible.

Dans la figure suivante, la variable température peut être considérée moyenne avec un degré d'appartenance de 0,8 et faible avec un degré d'appartenance de 0,2 (figure 1.2). **[11].**

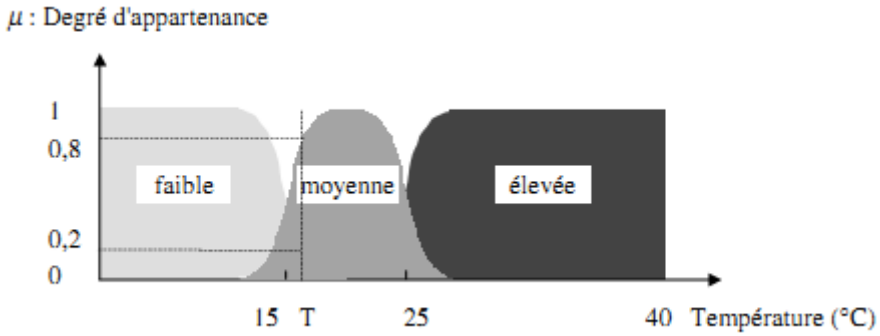


Figure 1.2 Les ensembles dans la logique floue

Considérant une variable floue x, on peut définir l'ensemble flou A par une fonction d'appartenance $\mu_A(x)$ sur l'univers de discours X, tel que L'univers de discours est présenté par l'ensemble des valeurs que peut prendre la variable floue x (figure 1.3).

D'une manière générale, le domaine de définition d'une fonction d'appartenance peut être réduit à un sous-ensemble flou [12]. On peut avoir ainsi plusieurs fonctions d'appartenance pour une variable floue [13].

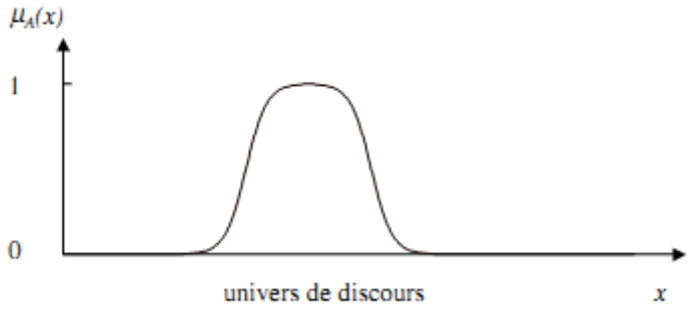


Figure 1.3 Représentation d'une fonction d'appartenance floue

1.2.2. Les opérateurs flous

Similairement aux théories des ensembles traditionnels, on peut définir pour les ensembles flous les relations telles que : Complémentaire d'un ensemble flou, l'union et l'intersection des ensembles flous. Ces opérations peuvent être présentés par les opérateurs "non", "et" et "ou".

L'intersection de deux ensembles flous traduit par l'opérateur "et" et peut être réalisé par :

- La fonction "Min": $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$
- La fonction arithmétique "Produit": $\mu_{A\cap B}(x) = \mu_A(x).\mu_B(x)$

L'union de deux ensembles flous traduit par l'opérateur "ou" et peut être réalisé par :

- La fonction "Max": $\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$
- La fonction arithmétique "Somme": $\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x)$

Le complémentaire d'un ensemble flou traduit par l'opérateur "non" et peut être réalisé par :

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

1.2.3. Inférence

Dans les régulations industrielles et les commandes des systèmes les variables flous ont plusieurs ensembles d'appartenance donc plusieurs règles floues donc plusieurs règles peuvent être activées simultanément. L'inférence floue est le processus d'agrégation des règles activées en même temps. On peut décrire les règles d'inférences floues par plusieurs manières **[11]** :

- Linguistiquement

Dans ce cas les règles sont écrites explicitement :

si (la tension est élevée **et** la température est faible) **alors** la vitesse est grande
ou
si (la tension est moyenne **et** la température est faible) **alors** la vitesse est positive

- Symboliquement :

Dans ce cas, on va remplacer par des abréviations la désignation des ensembles flous.

- En utilisant une matrice d'inférence :

Dans ce cas les règles floues sont présentées sous la forme d'un tableau à deux dimensions, tel que les variables floues d'entrées sont représentées par les entrées du tableau. La valeur floue de la variable de sortie définie par la règle est donnée par l'intersection d'une ligne et d'une colonne.

Exemple:

U		*T*		
		F	M	E
V	F	Z	P	GP
	E	Z	Z	P

Tableau 1.1 Matrice d'inférence pour des règles floues

Dans le cas où la table d'inférence n'est pas entièrement remplie l'inférence floue est dite incomplète, dans ce cas la sortie existe mais avec degré d'appartenance nul pour cette règle.

Dans les paragraphes suivants nous présentons les méthodes d'inférence les plus utilisées.

1.2.3.1. L'inférence floue par la méthode Min/Max

Cette méthode est appelée l'implication de Mamdani dans laquelle l'opérateur "ET" et la conclusion "ALORS" pour chaque règle sont réalisées par une fonction "Min" et l'agrégation des règles (opérateur "OU") est réalisée par une fonction "Max"

Dans l'exemple suivant deux règles sont activées simultanément **[11]** :

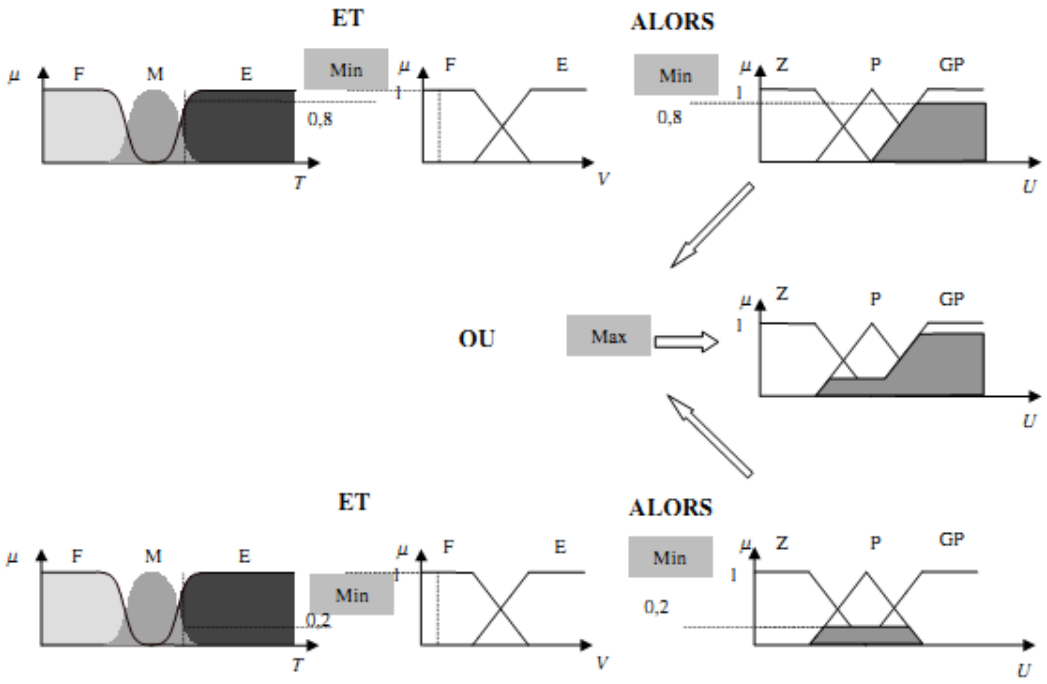


Figure 1.4 L'inférence Min/Max

1.2.3.2. L'inférence floue par la méthode Max/Produit

Cette méthode est appelée l'implication de Larsen dans laquelle l'opérateur "ET" est réalisée par une fonction "Min" et la conclusion "ALORS" pour chaque règle est réalisée par une fonction " Produit " et l'agrégation des règles (opérateur "OU") est réalisée par une fonction "Max".

Dans l'exemple suivant deux règles sont activées simultanément **[11]** :

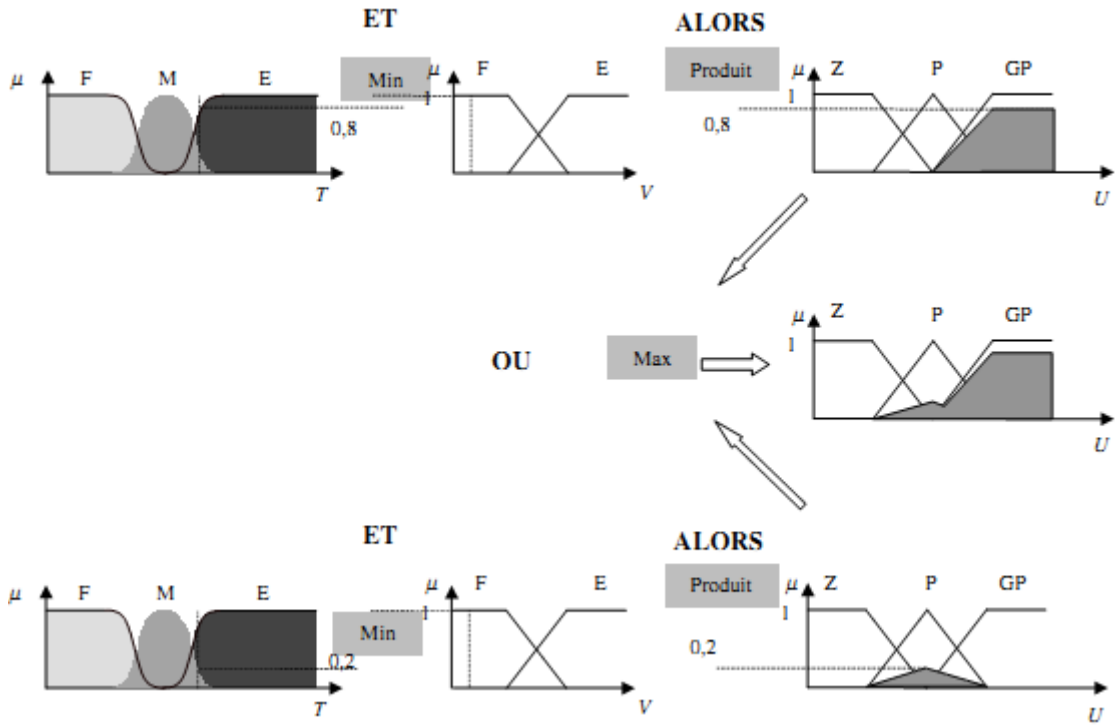


Figure 1.5 L'inférence Max/Produit

1.2.3.3. L'inférence floue par la méthode Somme/Produit

Dans cette méthode, l'opérateur "ET" et la conclusion "ALORS" sont réalisés par une fonction " Produit ". Cependant, l'opérateur "OU" est réalisé par une fonction " **Somme**".

À cause de sa simplicité la méthode **d'inférence** Min/Max est la plus utilisée. Dans ce travail on a utilisé la méthode **d'inférence** Min/Max.

1.2.4. La structure d'un régulateur flou

1.2.4.1. Introduction

Dans cette partie, nous nous intéressons à l'implémentation des systèmes flous dans les algorithmes de commande des systèmes et la conception des régulateurs.

Un régulateur flou (Figure 1.6) ne diffère pas beaucoup d'un régulateur classique. On retrouve à chaque fois un bloc d'entrée, un bloc de traitement et un bloc de sortie pour la détermination de la commande.

Deux blocs supplémentaires apparaissent pour le contrôleur flou: le bloc de fuzzification et le bloc de défuzzification.

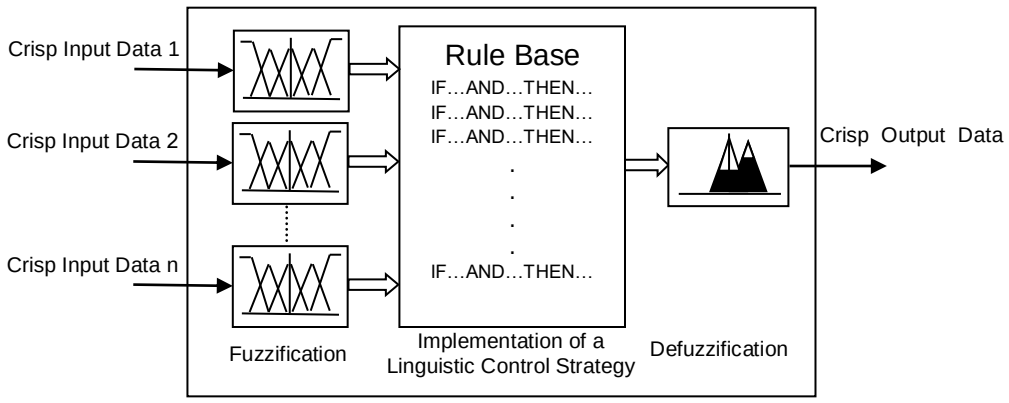


Figure 1.6 Structure du régulateur flou

1.2.4.2. Fuzzification

Dans les problèmes du control, les données d'entrées sont généralement physiques (réelles). Mais le traitement dans un régulateur flou est basé sur la théorie des ensembles flous; ceci nécessite donc une étape de fuzzification.

La procédure de fuzzification représente le passage des grandeurs physiques (ou réelles) aux valeurs floues. Cette procédure nécessite généralement une conversion analogique/numérique, et aussi le traitement des grandeurs entrées et leur conversion en variables linguistiques avec la détermination des fonctions d'appartenance [14].

Dans ce travail nous avons choisi les fonctions d'appartenance trapézoïdales et triangulaires pour la fuzzification les variables d'entrées (figure 1.7). Ces fonctions permettent une fuzzification rapide et une implantation facile.

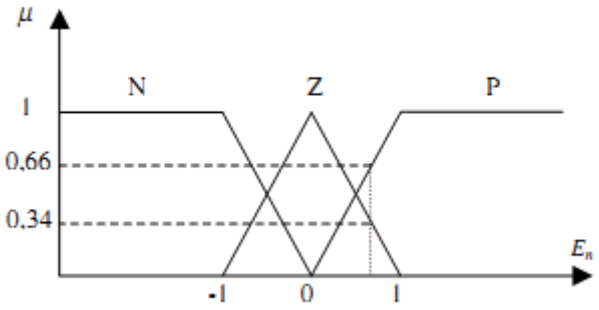


Figure 1.7 Fuzzification de l'erreur

La fuzzification de la variable de sortie est indispensable surtout lors de l'inférence et de la defuzzification. Dans ces étapes on doit connaître les ensembles flous de la variable de sortie ainsi que leurs fonctions d'appartenance [13].

1.2.4.3. Inférence [11]

Dans cette phase du régulateur flou les valeurs des variables linguistiques d'entrée et de sortie sont liées par certains règles flou qui doivent tenir compte du comportement statique et dynamique du processus à régler ainsi que des buts de réglage aperçus en particulier le circuit de réglage doit être stable et bien amorti. La stratégie de réglage dépend

essentiellement des inférences choisies. Dans cette étape l'expérience et les connaissances antérieures du système jouent un rôle important pour la détermination de ces règles.

Les régulateurs flous les plus usuels sont :

• Le régulateur de **Mamdani :**

Dans ce type de régulateur les entrées et les sorties du régulateur flou sont des variables floues, pour obtenir la valeur réelle de la commande à appliquer on doit passer par une étape de "defuzzification"[13].

• Le régulateur de **Sugeno :**

Dans ce type de régulateur seules les entrées sont des variables floues, la sortie du régulateur flou qui correspond à la commande est une variable réelle ou une expression polynomiale. [15].

1.2.4.4. La phase de Defuzzification

Cette phase consiste à calculer à partir des degrés d'appartenance de tous les ensembles flous de la variable de sortie, la valeur déterministe de cette sortie qui correspond à la commande fourni par le régulateur flou proposé.

Plusieurs méthodes de Defuzzification sont utilisées **[11]** :

• **Defuzzification par centre de gravité :**

La commande fourni par le régulateur flou est donnée par l'abscisse correspond au centre de gravité de la fonction d'appartenance résultant pour la variable de sortie.

$$dU_n = \frac{\int x u_R(x) dx}{\int u_R(x) dx} \tag{1.1}$$

C'est la méthode la plus utilisé. Néanmoins leur performance en temps de calcul peut être largement dégradée si la fonction d'appartenance résultante est compliquée.

• **Defuzzification par valeur maximum :**

Cette méthode de Defuzzification est très simple. La commande du régulateur est optée comme l'abscisse de la valeur maximale de la fonction d'appartenance résultante.

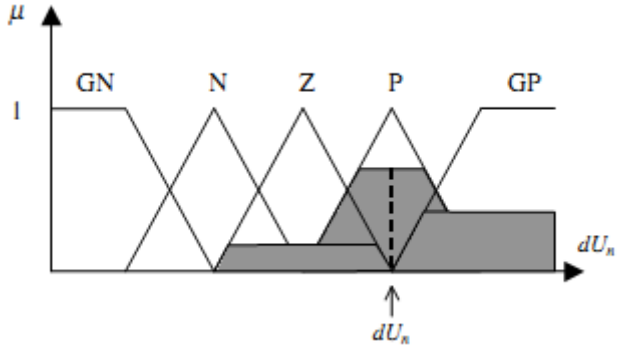


Figure 1.8 Méthode de Défuzzification par maximum

- **Defuzzification par hauteurs pondérées :**

Cette méthode présente un cas particulier de la Défuzzification par centre de gravité, lorsque les fonctions d'appartenance de la variable de sortie ne se recouvrent pas.

$$dU_n = \frac{\sum x u_{Ri}(x)}{\sum u_{Ri}(x)} \tag{1.2}$$

Cette méthode est très utile et simplifie grandement le calcul du centre de gravité si le régulateur est de type Sugueno.

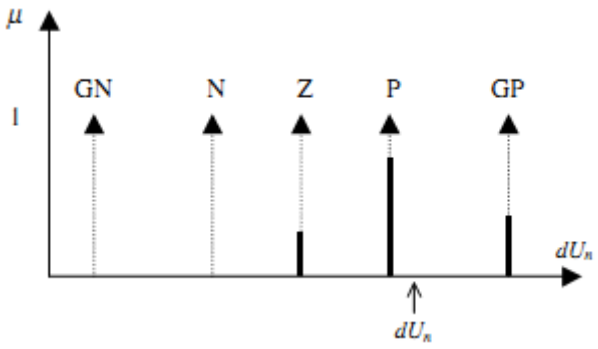


Figure 1.9 Défuzzification par la formule des hauteurs pondérées

1.3. Les réseaux de neurones artificiels

1.3.1. Définitions et principe des réseaux de neurones artificiels

L'origine des réseaux de neurones artificiels vient de l'essai de modélisation mathématique du cerveau humain par les travaux de McCulloch et Pitts **[16].** Dans ces travaux l'impulsion nerveuse est obtenue par un calcul simple effectué par chaque neurone et la pensée est née grâce à l'effet collectif d'un réseau de neurones interconnectés.

Un neurone formel est présenté dans le schéma suivant **[11]**:

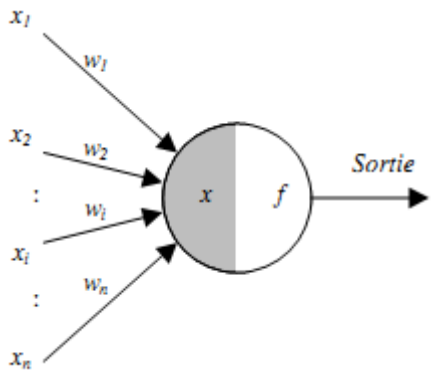


Figure 1.10 Schéma d'un neurone formel

La sortie du neurone formel est donnée par :

$$Sortie = f(x) \text{avec } x = \sum_{i=1}^{n} w_i x_i \quad (1.3)$$

Tel que:

x_i: les entrées du neurone.

x : l'état d'activation ou l'activité du neurone.

Wi: les poids synaptiques.

f : la fonction d'activation du neurone.

Les fonctions d'activation doivent être croissantes et bornées conformément au modèle du neurone biologique. Les fonctions d'activation les plus utilisées sont la fonction linéaire saturée, la fonction sigmoïde et la fonction signe.

1.3.2. Les Perceptrons multicouches [11]

Le perceptron est le réseau de neurones artificiel le plus connu. C'est est un réseau de neurones à propagation directe (feed forward). La figure suivante présente un perceptron multicouche à trois couches. La première couche est la couche des entrées. La deuxième couche est dite couche intermédiaire (ou couche cachée) avec des fonctions d'activation du type sigmoïde, cette couche présente le cœur du réseau de neurones. Quant à la troisième c'est la couche de sortie et elle est caractérisée par une fonction d'activation est du type linéaire bornée [17].

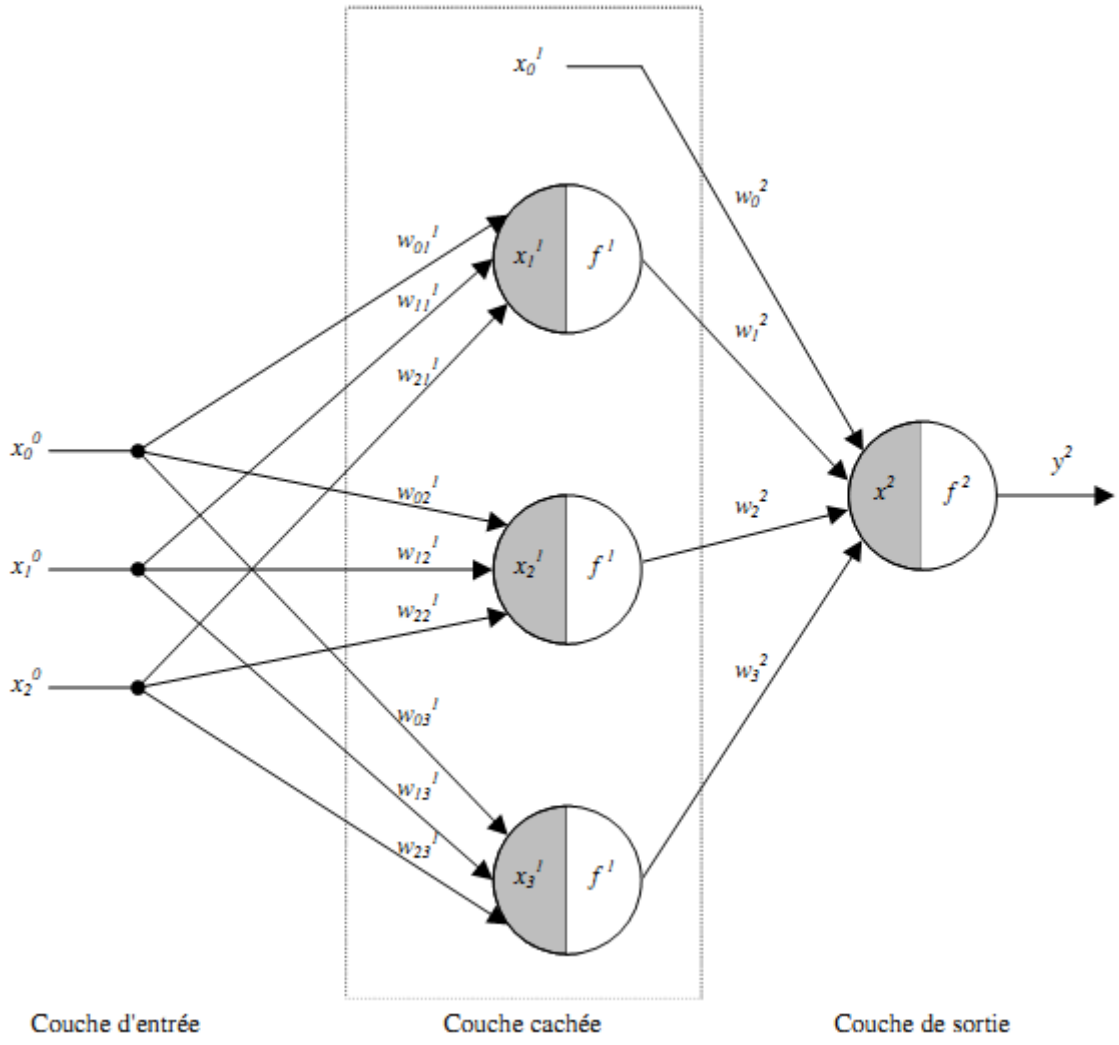


Figure 1.11 Réseau de neurones artificielle avec une couche cachée

La difficulté majeure de l'utilisation des réseaux de neurones réside dans le choix de la topologie de réseau de neurones (le nombre des couches cachées et le nombre de neurones pour chaque couche). Généralement il n'existe pas une méthode générale pour la détermination de la topologie de réseau de neurones et ce choix dépend souvent de la nature d'application.

1.3.3 Les fonctions d'activation d'un réseau de neurones

Dans un perceptron multicouche existe une variété des fonctions d'activation. Ces fonctions doivent être croissantes et bornées conformément au modèle du neurone biologique. Les fonctions d'activation les plus utilisées sont la fonction rampe avec saturation, la fonction sigmoïde et la fonction binaire à seuil [18].

a) La fonction d'activation binaire à seuil :

Cette fonction est caractérisée par un modèle tout ou rien. Une non-linéarité engendrée par le seuil introduit dans le comportement du neurone.

La fonction d'activation binaire à seuil est présentée dans La figure 1.12.

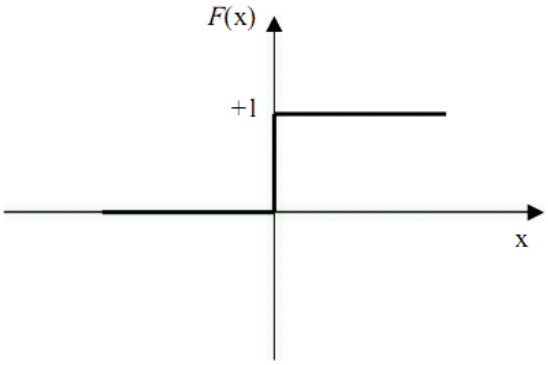


Figure. 1.12 : La fonction d'activation binaire à seuil.

b) La fonction d'activation rampe avec saturation :

La fonction d'activation rampe avec saturation connue par la fonction linéaire saturée représente un compromis entre la fonction seuil et la fonction linéaire.

La figure 1.13 présente La fonction d'activation rampe avec saturation :

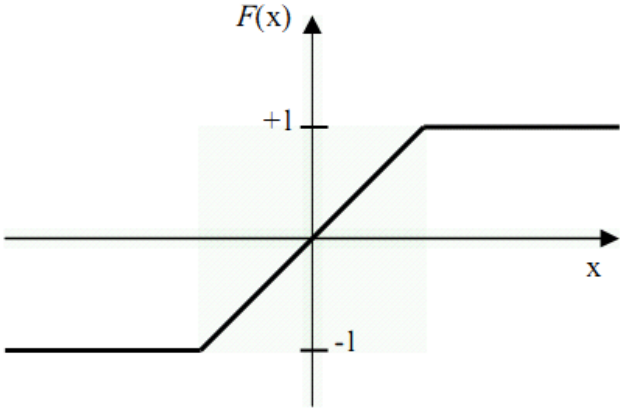


Figure. 1.13 : La fonction d'activation rampe avec saturation.

c) La fonction d'activation sigmoïde

C'est la fonction d'activation la plus utilisée surtout dans les perceptrons multicouches à cause de la continuité et la dérivabilité de cette fonction en tout point. Leur sortie maintenue dans l'intervalle [0,1] (Figure 1.14).

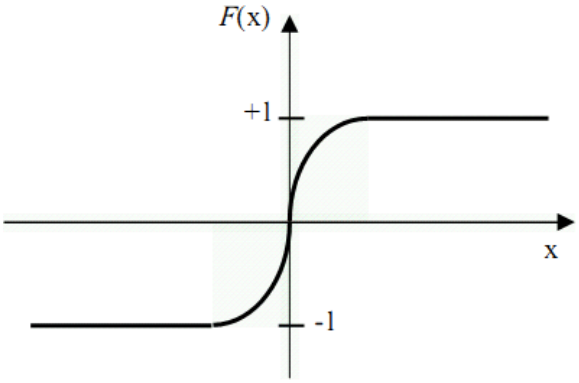


Figure. 1.14 : La fonction d'activation sigmoïde.

1.3.4. Apprentissage d'un RNA

Une fois la topologie du réseau de neurones est fixée, il faut passer le réseau par un algorithme d'apprentissage. Ce processus consiste à ajuster les poids synaptiques du réseau de neurones pour satisfaire un critère d'optimisation donné.

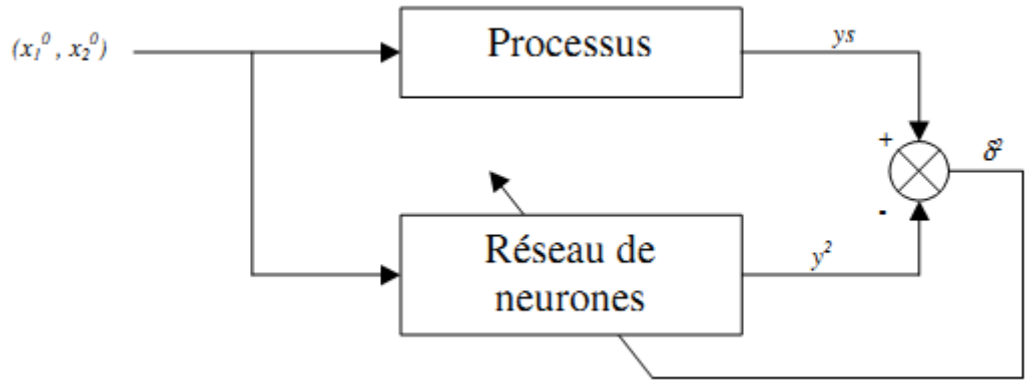


Figure 1.15 Schéma d'apprentissage du réseau de neurones artificielle

L'apprentissage est une procédure répétitive dans la quelle un ensemble de données a été utilisé pour permettre au réseau d'apprendre à résoudre ce problème en ajustant conformément ses poids.

Dans la littérature, il y a trois catégories des algorithmes d'apprentissage : apprentissage supervisé, apprentissage non supervisé et apprentissage avec des poids prédéterminés [18].

L'algorithme de rétropropagation de l'erreur est l'algorithme le plus populaire parmi les méthodes d'apprentissage des réseaux multicouches. Il utilise une fonction d'activation sigmoïde et il est basé sur la généralisation de la règle de Pineda [19].

1.3.5. Les applications des réseaux de neurones

- ☐ La classification
- ☐ La reconnaissance de formes
- ☐ L'identification des systèmes
- ☐ La commande des processus
- ☐ La robotique mobile

1.4. Les contraintes d'implémentation des fonctions complexes utilisées dans les techniques d'intelligence artificielle sur FPGA.

Les algorithmes des techniques d'intelligence artificielles exigent une grande quantité des opérations arithmétiques et logiques pendant leurs fonctionnement d'où le besoin de disposer des opérateurs mathématiques très performants capable de supporter la puissance de calculs exigée par ces opérations [5].

Pour les l'algorithmes de commande en temps réel, l'implémentation logicielle présente une limitation en vitesse de traitement des calculs à cause de leur fonctionnement séquentielles dans le traitement en termes de calcul. Pour ce la, l'implémentation matérielle tels que ASIC et FPGA remplacent de plus en plus les systèmes à microprocesseurs car ils sont très rapide et hautement parallèles. les circuits FPGA programmables présentent une solution alternative qui permettent de diminuer les temps de conception et de produire des prototypes à moindre coût de fabrication [31].

Il est à signaler que les quantités des ressources hardwares dans un circuit FPGA sont limitées [5], des efforts doivent être fait pour minimiser la consommation de ces ressources à travers une méthodologie d'implémentation présentée dans cette thèse.

Cette méthodologie est fondée sur l'illustration d'un compromis qui réalise une optimisation de l'architecture hardware. Ce qui préserve les performances de système d'entrainement électrique constitué d'un onduleur de tension et une machine asynchrone.

Dans le but de développer une architecture hardware simple et efficace, nous avons choisi l'utilisation des nombres avec le format point fixe et une approximation optimisée de la fonction sigmoïde pour la fonction d'activation dans les réseaux de neurones artificiels. En logique floue nous avons implémenté les valeurs linguistiques d'une fonction d'appartenance par un registre simple, les operateurs de Zadeh Max-Min sont utilisés pour l'inférence floue à cause de leurs simplicités d'implémentation matérielles, de plus les régulateurs flous implémentés sont de type Sugeno pour éviter la défuzzification par des formules mathématiques complexes telle que la defuzzification par centre de gravite.

La commande de la machine synchrone exige l'estimation de certaines grandeurs électriques telles que le couple électromagnétique et le flux statorique de la machine [42]. Cependant cette estimation demande le calcul des fonctions couteuses en ressources et en temps de calcul comme l'intégration des fonctions, la division binaire, l'arc tangent et la racine carrée. Dans ce travail nous avons implémenté l'algorithme "Backward Euler Approach" pour calculer des intégrales en temps discret pour approximer les intégrales d'estimation de flux statoriques, de plus on a utilisé l'algorithme de CORDIC pour le passage de coordonnées cartésiennes vers les coordonnées polaires par des rotations simples en évitant les calculs des fonctions classiques tels que la division binaire, l'arc tangent et la racine carrée. Nous avons utilisé aussi une approximation optimisée de la fonction d'activation sigmoïde non linéaire dans l'implémentation des réseaux de neurones.

Enfin, la réalisation de l'architecture hardware a été effectuée simultanément par les fonctions de la bibliothèque XSG et le langage VHDL pour le développement et la

simulation des architectures hardware proposées. Pour ce la, une plate forme a été introduit par les fondateurs des circuits FPGA pour l'insertion des codes, la simulation, la synthèse et le placement et routage des architectures matériels dans les circuits FPGA.

1.4.1 La méthode d'intégration d'Euler [20]

La méthode d'Euler utilise une approximation linéaire d'ordre 1. Elle suppose qu'une fonction y(t) évolue entre deux instants de mesure discrets selon l'équation :

$$\dot{y}(t) = \frac{dy}{dt} = f \Leftrightarrow y(t) = ft + f_0 \tag{1.4}$$

A $t = t_n = 0s$, (on prend t_n comme origine de temps) :

$$y(t_n) = y_n \tag{1.5}$$

A l'instant $t = t_n + T_s$, on a :

$$y(t_n + T_s) = f(t_n + T_s) + f_0 = ft_n + f_0 + fT_s \tag{1.6}$$

$$y_{n+1} = y_n + fT_s \tag{1.7}$$

Où T_s représente le pas de calcul.

Afin d'atteindre la solution y(t) de $\dot{y}(t) = f(t)$, sur l'intervalle [0, t]

Puisque $\frac{dy(t_k)}{dt} = f\big(t_k, y(t_k)\big)$, on obtient ainsi le schéma numérique d'Euler:

$$\begin{cases} y_0 = valeur\ initiales \\ y_{n+1} = y_n + T_s f\big(t_n, y(t_n)\big) \end{cases} \tag{1.8}$$

Cette méthode est d'ordre 1, cela veut dire que l'erreur est proportionnelle au carré du pas de discrétisation. L'avantage de la méthode d'Euler, tire son origine du fait qu'elle réclame uniquement l'évaluation de la fonction f pour chaque pas d'intégration.

1.4.2 L'algorithme de CORDIC

L'algorithme CORDIC (acronyme de coordinate rotation digital computing) a été conçu initialement par J.E.Volder [21]. Cet algorithme est utilisé d'une part, comme un mécanisme pour le calcul numérique, et d'autre part pour effectuer des opérations géométriques tels que les rotations de vecteurs ou les changements de coordonnées polaires/cartésiennes et cartésiennes/polaires et dans le plan euclidien.

Nous nous intéressons spécialement ici à l'implémentation des commandes pour machines électriques sur des architectures FPGA. En effet, il est considéré comme un opérateur générique qui peut être utilisé dans un grand nombre des algorithmes de control et de traitement du signal.

Principe de l'algorithme CORDIC

Le principe de l'algorithme CORDIC est basé sur des calculs trigonométriques comme la rotation d'un vecteur de base pour un angle donné. Une transformation de rotation élémentaire peut être exprimée sous une forme matricielle par: [22]

$$\begin{cases} x' = (x\,cos(\theta) - y\,sin(\theta)) \\ y' = (y\,cos(\theta) + x\,sin(\theta)) \end{cases} \tag{1.9}$$

L'algorithme CORDIC a été initialement conçu pour effectuer une rotation vectorielle, où le vecteur (x, y) est tourné de l'angle θ donnant un nouveau vecteur (x ', y') [22].

La force de la méthode CORDIC découle du principe qu'on peut obtenir $\boldsymbol{v}$ en appliquant au vecteur $\boldsymbol{v_0} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix}$une série des rotations des angle$\boldsymbol{\theta_i}$. (Théoriquement infinie, mais réellement finie).

Ici, prenons une série telle que $0 \leq |\boldsymbol{\theta_{i+1}}| < |\boldsymbol{\theta_i}| \leq \pi/2$ et que:

$$\sum_{i=0}^{n} \theta_i = \theta \tag{1.10}$$

Notons que les $\boldsymbol{\theta_i}$ peuvent aussi être négatifs.

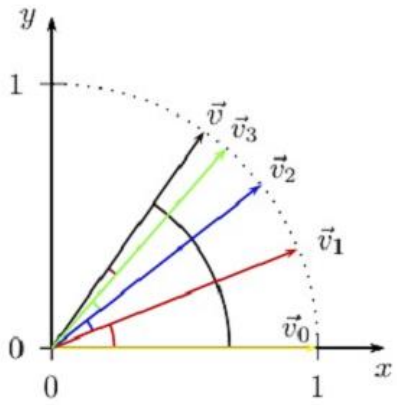


Figure 1.16 Principe de l'algorithme CORDIC

Si $\boldsymbol{v_i} = \begin{bmatrix} \boldsymbol{x_i} \\ \boldsymbol{y_i} \end{bmatrix}$ est le vecteur position à la i[ème] rotation.

Alors $\boldsymbol{v_{i+1}}$ est donné par la rotation d'angle $\boldsymbol{\theta_i}$ du $\boldsymbol{v_i}$ qui peut être calculé par le produit matriciel suivant :

$$v_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{1.11}$$

En mettant en évidence le cos dans (1.11). Nous arrivons à l'équation suivante [23] :

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = cos(\theta_i) \begin{bmatrix} 1 & -tan(\theta_i) \\ tan(\theta_i) & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (1.12)$$

Nous abandonnerons donc le cos de côté pour l'instant, puisque comme cité avant, les récurrences suivantes nous conduiront vers les bienfaisants rapports trigonométriques.

Marquons par $\boldsymbol{x_i^*}$ et $\boldsymbol{y_i^*}$ les composantes de ces vecteurs :

$$v_{i+1}^* = \begin{bmatrix} x_{i+1}^* \\ y_{i+1}^* \end{bmatrix} = \begin{bmatrix} 1 & -tan(\theta_i) \\ tan(\theta_i) & 1 \end{bmatrix} \begin{bmatrix} x_i^* \\ y_i^* \end{bmatrix} = \begin{bmatrix} x_i^* - tan(\theta_i) y_i^* \\ y_i^* + tan(\theta_i) x_i^* \end{bmatrix} \quad (1.13)$$

Il reste toujours deux difficultés dans ces récurrences au niveau de la vitesse du calcul informatique : le calcul de $\boldsymbol{tan(\theta_i)}$ et celui du produit.

Si nous prenons les $\boldsymbol{\theta_i}$ tels que $tan(\theta_i) = \pm 2^{-i}, i = 0, 1, 2, ..$ nous enlevons la problématique de la multiplication, puisque une multiplication par une puissance de deux en binaire revient à déplacer une virgule.

En réécrivant l'équation (1.13) de façon scalaire, nous obtenons : [23]

$$\begin{cases} x_{i+1}^* = x_i^* - \sigma_i 2^{-i} y_i^* \\ y_{i+1}^* = y_i^* + \sigma_i 2^{-i} x_i^* \\ \theta_{i+1} = \theta_i + \sigma_i atan(2^{-i}) \end{cases} \quad (1.14)$$

$\sigma_i = (+1 \text{ or } -1)$, tel que α_i est la direction de rotation

Si nous exécutons n itérations à l'aide de la méthode de CORDIC, le vecteur terminal est multiplié par le résultat de: $K_n = cos(\theta_0).cos(\theta_1).cos(\theta_2) \dots \dots \dots cos(\theta_n)$. Puisque $tan(|\theta_i|) = 2^{-i}$, il est possible de précalculer les divers $cos(\theta_i)$. En effet,

$$cos(\theta_i) = \frac{1}{\sqrt{1 + tan^2(\theta_i)}} = \frac{1}{\sqrt{1 + 2^{-2i}}} \quad (1.15)$$

Nous avons donc :

$$K_n = \prod_{i=1}^{n} cos(\theta_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{1 + 2^{-2i}}} \quad (1.16)$$

Le facteur d'échelle CORDIC de Kn ne dépend que du nombre d'itérations; 'n' Seules les configurations fonctionnelles Rotation, Translation, Rectangulaire vers Polaire et Polaire vers Rectangulaire sont affectées par le facteur d'échelle CORDIC. Lorsque ces configurations fonctionnelles sont sélectionnées, le noyau CORDIC offre la possibilité de multiplier par Kn pour annuler le facteur d'échelle. Pour n = 10, Kn = 0,6073.

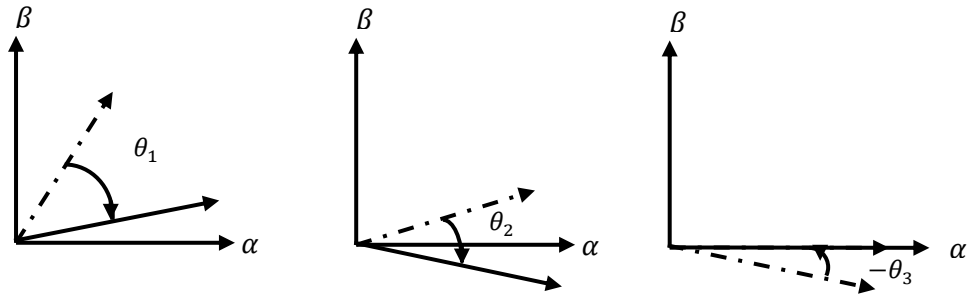


Figure 1.17 Example l'algorithme de CORDIC pour vector rotation

1.4.3 L'approximation optimisée de la fonction d'activation sigmoïde [6]

La fonction sigmoïde est une fonction mathématique ayant une courbe caractéristique en «S», qui transforme les valeurs comprises entre 0 et 1. La fonction sigmoïde est également appelée courbe sigmoïde ou fonction logistique. C'est l'une des fonctions d'activation non linéaires les plus utilisées.

L'expression mathématique pour la fonction sigmoïde:

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{1.17}$$

En règle générale, une fonction sigmoïde est une fonction continue et dérivable, dont il a une dérivée première non-négative et équipé d'un minimum local et maximum local.

Graphique

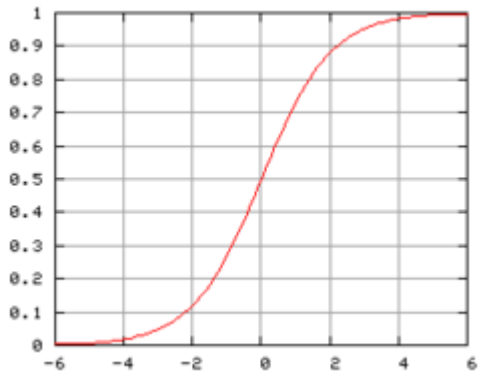


Figure 1.18. Graphique de la fonction sigmoïde [6]

Les fonctions sigmoïdes sont souvent utilisés dans les réseaux de neurones pour introduire une non-linéarité dans le modèle et / ou pour faire en sorte que certains signaux restent dans des limites spécifiques. Un élément neuronal artificiel populaire calcule la combinaison linéaire de ses signaux d'entrée et applique une fonction sigmoïde limitée à la suite; ce modèle peut être considéré comme une variante de « régulière » seuil de neurone classique [6].

Cependant l'implémentation de cette fonction sur une architecture hardware n'est pas facile et parfois impossible à cause des fonctions complexes constituant la fonction sigmoïde telles que

la division et l'exponentielle. Dans [06] D. E. Khodja et al proposent une approximation optimisée de la fonction sigmoïde pour l'implémentation sur FPGA de cette fonction.

La fonction sigmoïde peut être approximée par un polynôme de deuxième ordre de la forme: $f(x) = ax^2 + bx + c$

Les seules inconnues dans cette équation sont a, b et c qui sont déterminées à l'aide de l'algorithme de Vendermond [24] une fois que on prend trois points de la fonction sigmoïde {(xi,, yi), i=1, 2 et 3} qui remplissent les équations suivantes [06]:

$$\begin{cases} y_1 = \frac{1}{1+e^{-x_1}} = ax_1^2 + bx_1 + c \\ y_2 = \frac{1}{1+e^{-x_2}} = ax_2^2 + bx_2 + c \\ y_3 = \frac{1}{1+e^{-x_3}} = ax_3^2 + bx_3 + c \end{cases} \quad (1.18)$$

L'équation(3) peut être réécrite sous la forme matricielle comme suite:

$$Y = X * A \text{ Où } Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, X = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix} et\ A = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (1.19)$$

A la suite, les coefficients sont calculés par l'équation suivante:

$$A = X^{-1} * Y \quad (1.20)$$

En fin:

$$f(x) = \begin{cases} -0.0332x^2 + 0.2549x + 0.5\ pour - 4 < x < 0 \\ 0.0332x^2 + 0.2549x + 0.5\ \ pour\ 0 \leq x < 4 \end{cases} \quad (1.21)$$

1.5. Conclusion

Dans ce chapitre, nous avons présenté les éléments de base des techniques d'intelligences artificielles basées sur la logique floue et les réseaux de neurones. Cependant ces techniques font recours à des fonctions très pénibles en termes de temps de calcul, et des opérations arithmétiques difficiles à implémenter sur des plateformes hardwares. En fin de ce chapitre on a présenté des techniques d'approximation et d'optimisation pour quelques fonctions dont nous avons besoin dans les prochains chapitres.

Chapitre 2:
Méthodologie d'implémentation des commandes numériques sur FPGA

2.1 Introduction:

Le développement de système sur FPGA est différent du développement de système traditionnel car il implique une conception physique et une conception logicielle. En effet, il n'est pas facile d'utiliser un langage de description de matériel qui partage certains aspects avec les langages de programmation traditionnels. Tandis que, ce langage permet de définir physiquement des circuits au niveau d'abstraction choisi par le concepteur. Par conséquent, la stratégie de développement est différente de la conception logicielle qui décrit généralement un traitement séquentiel. Toutefois, lors du développement de systèmes sur puce et de systèmes basés sur des circuits logiques programmables, la frontière entre développement matériel et développement logiciel est encore moins évidente.

Dans ce chapitre, nous présentons l'implémentation des algorithmes de contrôle de la machine asynchrone sur FPGA et leurs plates-formes d'implémentation.

2.2 La technologie des circuits FPGA

Les circuits intégrés FPGA contiennent un grand nombre d'éléments de base, qui sont utilisés pour synthétiser des fonctions logiques ou mathématiques et sont connectés les uns aux autres via une matrice d'interconnexion programmable. D'un autre coté, par rapport aux circuits intégrés spécifiques à l'application ASIC, les FPGA sont plus lents, plus énergivores et nécessitent une plus grande surface de silicium. Par contre les FPGAs sont reprogrammables et moins coûteux en coût et en temps de développement. Cependant, une fois le produit développé, le FPGA est toujours plus cher que l'ASIC [25]. A cet effet, nous nous concentrons principalement sur la structure du FPGA Xilinx, car c'est le Virtex-4 de Xilinx, qui est utilisé dans notre travail.

2.2.1 La structure interne des circuits FPGA

Les FPGAs sont des circuits intégrés numériques composés d'un grand nombre de composants ou de blocs logiques programmables et reconfigurables sans modifications matérielles majeures. Le FPGA est programmé via sa cellule mémoire de type SRAM (Static Random Access Memory). Cette technologie SRAM permet d'économiser la configuration à mettre en œuvre, et le FPGA peut être reconfiguré autant de fois que nécessaire pour réaliser les fonctions requises. Cette souplesse de programmation est très utile lorsque des erreurs de conception sont diagnostiquées, car la reprogrammation du FPGA avec la version corrigée est suffisante. La figure 2.1 montre l'architecture générale du FPGA, basée sur des cellules mémoire SRAM [25].

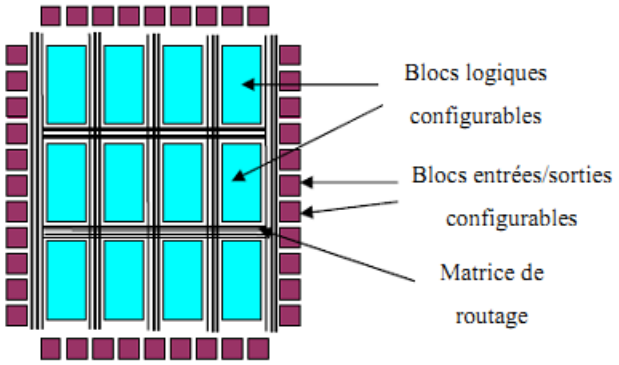


Figure 2.1 L'architecture interne d'un circuit FPGA.

Les éléments caractéristiques d'un circuit FPGA sont toujours les mêmes quelle qu'en soit la structure ou le concepteur. Nous mentionnons dans ce qui suit quelque nombre de ces éléments :

- **Les composants logiques**

Ce sont les éléments constitutifs de tout circuit FPGA. Des opérations de type logique combinatoire peuvent être mises en œuvre dans ces blocs. Tandis que les fabricants et les architectures varient, ces blocs ont généralement la même composition. La structure la plus courante est la figure 2.2, qui montre la structure de base. Ces structures sont généralement constituées d'une ou plusieurs tables ou LUT (look-up tables). Elles contiennent des tables de vérité ou un ensemble de valeurs stockées de fonctions logiques qui doivent être exécutées après la configuration du FPGA. Ensuite, la table LUT se trouve dans le registre de sortie permet de synchroniser la sortie avec l'horloge si nécessaire. Bien que la plupart des blocs logiques de base fournissent une chaîne de report de propagation rapide pour former un petit additionneur rapide.

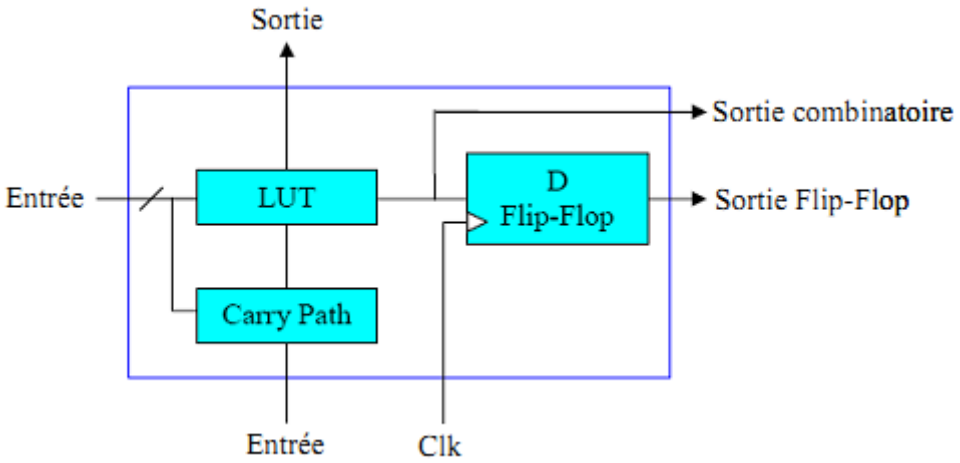


Figure 2.2 Architecture de base d'un élément logique [38].

- Les blocs de mémorisation

Actuellement, les FPGA sont utilisés dans des applications qui nécessitent souvent une capacité de stockage. En plus, le besoin d'intégrer des blocs mémoire directement dans l'architecture FPGA est rapidement devenu indispensable et donc devenu opérationnel. Ainsi, puisqu'il n'est plus nécessaire de communiquer avec des composants extérieurs au circuit, le temps d'accès mémoire est réduit.

- Les blocs de routage

Les composants de routage sont les éléments les plus influents dans les FPGA. En fait, ces composants représentent la plus grande partie du silicium consommé sur la carte FPGA. Ces ressources sont composées de segments (de différentes longueurs) qui admettent de relier entre eux d'autres éléments à travers une matrice de liaison. De plus, le routage de ces ressources est le point clé pour développer des applications sur FPGA. Alors que, ces composants sont très influents puisqu'ils sont directement attachés à la surface utilisée et à la fréquence maximale de fonctionnement.

- Les blocs Entrés/Sorties

Les blocs d'entrées/sorties sont utilisés pour relier le circuit FPGA à son environnement extérieur. Ces éléments peuvent bénéficier de mémorisation, de protections et des autres éléments pour le control des entrées et des sorties. Il faut noter que les cartes FPGA actuelles utilisent différentes normes pour les niveaux d'entrées et de sorties qui peuvent êtres choisies par configuration pour s'adapter à l'environnement [25].

2.2.2 Ressources logiques

Plusieurs types de ressources hardwares sont embarqués, même la puce de silicium pour l'amélioration des performances des circuits FPGA pour des applications spécifiques. Fréquemment, ces ressources sont faisables à partir d'éléments logiques de base mais elles en nécessitent un très grand nombre et les répits de routage sont alors très grands. Pour cela, les fabricants des circuits FPGA incluent ces éléments directement sur la puce de silicium. En conséquence, la quantité et le type d'éléments embarqués dépendent du marché visé.

La figure 2.1 montre quelques éléments dérivant de l'architecture du Virtex-4 de Xilinx :

- les fonctions d'entrées/sorties du FPGA sont implémentées par les Input/Output Blocks (IOB)
- les fonctions relatives à l'horloge sont implémentées par les Digital Clock Managers (DCM)
- Les mémoires RAM à double port sont implémentées par les blocs SelectRAM
- un multiplicateur 18 bits par 18 bits dédié est implémenté par le bloc Multiplier

2.2.3 Routage

En observant la figure 2.1, on constate que les ressources d'interconnexions de la carte FPGA Xilinx sont regroupées sous la forme des matrices d'interconnexions programmables (Switch Matrix). Il est important de noter que ces ressources logiques sont entrecroisées avec les ressources de routage.

La mémoire SRAM (Static Random Access Memory) est la base des interconnexions programmables des FPGA de type Xilinx et Altéra, ces avantages principaux sont dus à leur capacité infinie de reprogrammation et leur rapidité.

Pour chaque connexion une case mémoire à été allouée et le contenu de cette mémoire montre si la connexion inactive (non connectée) ou active (connectée).

Ces mémoires ne nécessitent pas des rafraîchissements périodiques et sont des mémoires volatiles qui maintiennent les données exclusivement s'ils sont alimentés.

Donc il est nécessaire de reconfigurer le FPGA pour chaque remis sous tension. Pour les clients finaux, les fichiers de configuration sont fréquemment sauvegardés dans une mémoire ROM à proximité du circuit FPGA [26].

2.3 Technologie des systèmes sur puce

L'accroissement de la compacité d'intégration a autorisé le développement des systèmes intégralement embarqué sur une seule puce. L'unité de traitement, les bus de communication, les mémoires et autres éléments spécialisées se retrouvent sur la même carte, qui permettent un traitement physique plus compact. Généralement, un système SoC est plus fiable, moins coûteux et moins énergivore qu'un système semblable conçu avec plusieurs puces. Par contre le développement d'un système sur une seule puce réclame le développement parallèle des aspects logiciels et matériels et une expertise dans les deux domaines est indispensable [26].

2.4 Le langage VHDL pour la description hardware

Les langages de description **hardware** les plus utilisés pour la conception sur FPGA sont le VHDL et le Verilog. Les deux font l'objet de standards EEEE pour uniformiser ces langages pareillement pour les manufacturiers que les concepteurs. le VHDL est le langage utilisé dans le cadre de ce travail.

Le langage VHDL possède une grammaire semblable à celle d'Ada, permet la description par trois niveaux d'abstraction : le niveau comportemental, le plus élevé, décrit l'algorithme réalisé par le circuit; le niveau des flots de données décrit comment les données sont traitées et où elles sont stockées et le niveau structural définit exactement la structure du circuit à l'aide d'éléments de base tels que les portes logiques ou même des transistors [27].

2.5. Contributions des FPGAs dans la commande des systèmes électriques

Actuellement les circuits FPGA sont utilisés dans diverses applications qui nécessitant une quantité importante des traitements arithmétiques et logiques tels que la télécommunication, le traitement de l'image et du signal, le contrôle des convertisseurs statiques de puissance, l'aéronautique, la bio-informatique, les équipements médicaux, l'automobile, les transports, la robotique, le contrôle des machines électriques et/ou encore plus souvent dans toute application requérant des calculs numériques extrêmement rapides [26].

Les avantages des composants FPGA sont nombreux et dépendent de leurs utilisations [28]. Parmi les avantages les plus courants des FPGA on peut citer :

- Niveau de performance temps réel.
- La réduction de coût et d'encombrement ;
- La diminution du temps d'exécution des algorithmes.
- La souplesse de programmation.
- Reconfiguration rapide et facile.

Les cartes FPGAs ont été utilisées avec succès dans les applications du contrôle des systèmes électriques, pour la commande de convertisseurs multi-niveaux, le contrôle de dispositifs d'électronique de puissance, pour la commande des filtres actifs, le contrôle d'onduleurs et ainsi pour la commande des machines électriques [29-32].

Dans le contexte de cette thèse, dont l'objectif majeur est l'évaluation des performances de l'utilisation des techniques intelligentes (Les réseaux de neurones et La logique floue) versus les techniques classiques, pour le contrôle d'une machine asynchrone basé sur un circuit FPGA avec les outils conventionnels disponibles (Matlab/ Simulink, Xilinx ISE Simulator, Xilinx ISE, Xilinx System Generator).

Dans le paragraphe suivant, nous exposons le principe de " Co-Simulation Hardware" pour le prototypage et la validation des architectures hardwares proposées pour la commande intelligente des machines électriques, d'une manière générale.

2.6. La conception par prototypage " Co-Simulation Hardware"

Dans cette partie, nous exposons le principe du prototypage " Co-Simulation Hardware" des implémentations hardwares des techniques d'intelligence artificielle pour la commande d'une machine asynchrone à vitesse variable.

Le prototypage " Co-Simulation Hardware" n'est bien sûr pas limité à ce cas d'application. Le système étudier comporte une partie commande et une partie puissance.

La partie puissance utilise un convertisseur statique avec une machine asynchrone commandée par la partie commande (La figure 2.3). La partie commande communique

avec la partie puissance pour connaître l'état du système et changer cet état en agissant de manière appropriée sur les semi-conducteurs de l'onduleur pour assurer la fonctionnalité désirée.

Tous les blocs de la partie commande sont présentées dans la figure 2.3. On peut remarquer toutes les interactions possibles entre la partie puissance et celles de la commande.

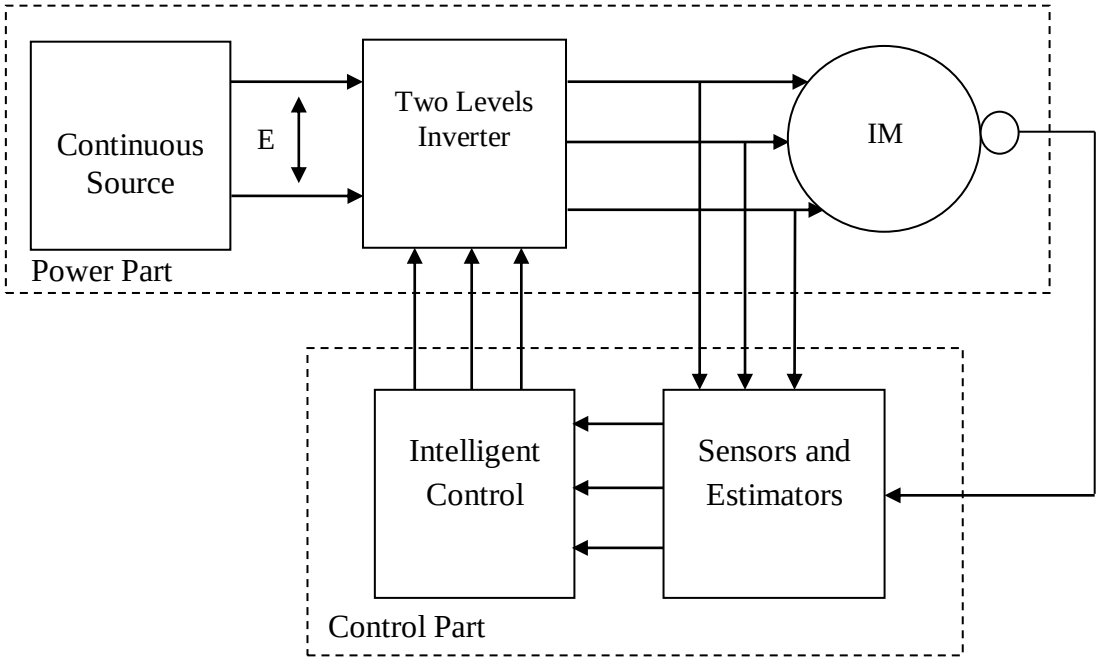


Figure 2.3 Système électrique pour le contrôle d'une machine asynchrone

Les fonctions coûteuses en temps de calcul doit nécessairement être implémentées sur un composant FPGA pour réduire le temps d'exécution global. Par contre, les fonctions avec des contraintes mineures en termes de vitesse de traitement des taches on peut les exécuter sur un microprocesseur ou un DSP, par exemple.

Pour limiter les interfaces, simplifier l'implémentation et réduire les problèmes de comptabilité engendrés par l'utilisation de plusieurs systèmes numériques, tous les blocs de la partie commande seront implantés dans le circuit FPGA.

2.6.1 Prototypage par " Co-Simulation Hardware"

Le but du prototypage par **" Co-Simulation Hardware"** est la validation des architectures hardware des contrôles de la partie commande tout en simulant la partie puissance sur ordinateur. Donc le prototypage par **" Co-Simulation Hardware"** permet d'évaluer les architectures hardware des commandes dans un environnement virtuel tel que la modification des architectures de commande est simplement réalisables par reprogrammation et sans redondance matérielle coûteuse. Alors, le temps de développement et le coût d'un projet sont réduits.

Les motivations majeures pour l'utilisation d'un prototypage **" Co-Simulation Hardware"** sont [33]:

- L'indisponibilité de la partie puissance.
- les parties puissances sont élevées,
- les risques importants pour la sûreté globale de l'opération.
- les dégâts irréversibles sur la partie puissance qui pouvant entrainer par le disfonctionnement des architectures de commande.

Dans un prototypage **" Co-Simulation Hardware"**, à toute étape de simulation, la partie puissance du système électrique est simulée sur ordinateur et les sorties sont envoyés au circuit FPGA. Lorsque le circuit FPGA reçoit les signaux d'ordinateur, il réalise les algorithmes des architectures implémentés.

La carte retourne ensuite à l'ordinateur les commandes des interrupteurs du convertisseur de puissance établis dans cette étape.

A ce stade, une étape de simulation **" Co-Simulation Hardware"** est réalisé. L'échange des informations entre l'ordinateur et la carte est synchronisé par une interface de type Ethernet Point to Point, cette interface est utilisée pour connecter la carte à l'ordinateur.

Le prototypage **" Co-Simulation Hardware"** permet la validation et la vérification de l'implémentation numérique des architectures de commande antérieurement cités avec l'utilisation d'une carte FPGA réelle avant l'intégration de ces commandes à un environnement de puissance réel.

2.6.2 Les logiciels mis en œuvre dans le prototypage " Co-Simulation Hardware"

L'intégration et la conception des architectures de commande sur circuits programmables de type FPGA demande des outils de développement particuliers. Dans notre travail, on a choisi l'utilisation de l'outils logiciels Xilinx système générateur (XSG) de la société Xilinx. Généralement, les outils Xilinx visent spécialement des cartes FPGA de la même famille. Le couplage possible entre Xilinx système générateur (XSG) et les outils Matlab/Simulink est la motivation majeure de ce choix.

La librairie Xilinx système générateur(XSG) permet de produire le code VHDL synthétisable des architectures de commande à partir d'un bloc Matlab/Simulink.

Les fichiers Matlab/Simulink sont encore compatibles avec les codes de Xilinx système générateur (XSG) ; cette alternative permet la modélisation des systèmes mixtes (hardware/software).

2.6.3 Le flot de conception pour prototypage " Co-Simulation Hardware"

Dans notre travail de recherche nous avons utilisé la méthodologie de conception par prototypage **" Co-Simulation Hardware"** [34], qui est montrée à la figure 2.4.

Cette méthodologie utilise Matlab/Simulink et ISE Xilinx système générateur de la société Xilinx.

Le flot de conception présenté et affecté au prototypage **" Co-Simulation Hardware"** permet d'une étude fonctionnelle du système électrique de la figure 2.3, à l'aide de Matlab/ Simulink et des librairies Fuzzy Logic et Neural Network.

Ensuite, les architectures de commande sont modélisés et simulés étape par étape, jusqu'à l'obtention d'un schéma synthétisable. Ce schéma sera effectivement implanté sur la carte FPGA cible.

Une fois la carte FPGA est programmée, il sera alors testé dans son circuit par prototypage " Co-Simulation Hardware".

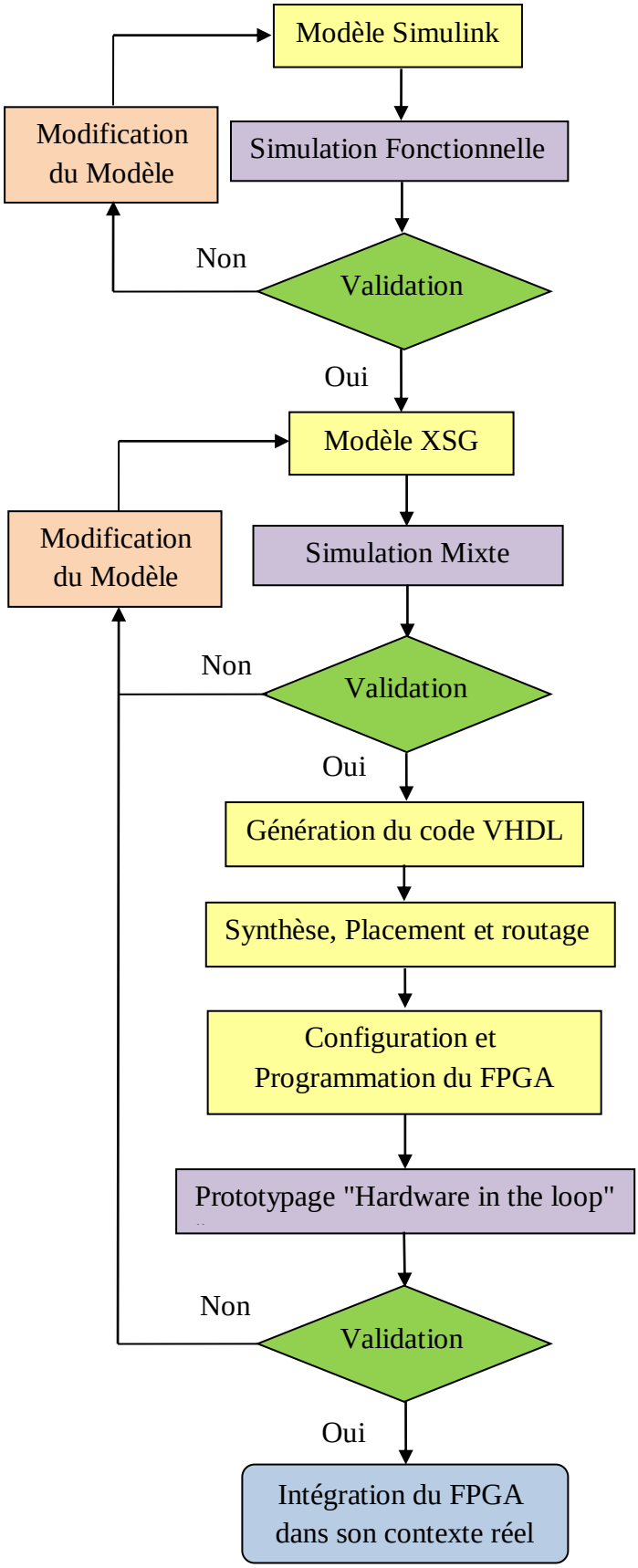


Figure 2.4. Le flot de conception pour le prototypage " Co-Simulation Hardware".

2.6.3.1 La simulation fonctionnelle

Dans la première phase (Figure 2.4), la fonctionnalité des architectures de commande est étudiée et validée après la définition des spécifications du système électrique.

Cette étape utilise généralement les outils conventionnels de simulation, pour notre domaine d'application sont Matlab/Simulink, Matlab/FuzzyLogic et Matlab/NeuralNetwork.

La bibliothèque Simulink a été utilisée pour la modélisation et la simulation des algorithmes de commande et la partie puissance du système électrique.

Les architectures de commande sont simulées et validées en mode continu avec un pas de simulation variable et à partir de modèles continus.

Par la suite, les architectures de commande sont simulées et validées en mode continu avec un pas de simulation fixe. Ce dernier doit être choisi convenablement petit pour assurer une précision tolérable.

La Figure 2.5 suivante présente cette première phase de conception.

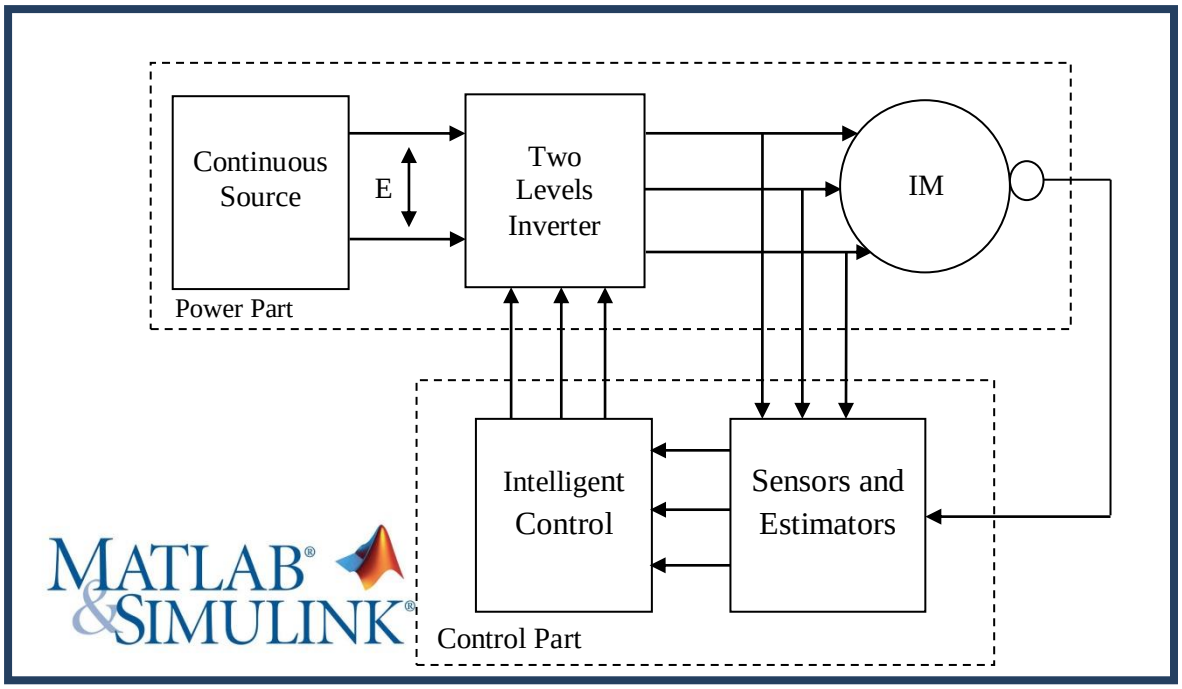


Figure 2.5 Simulation fonctionnelle

2.6.3.2 La simulation mixte

Une fois la fonctionnalité des architectures de commande est validée en mode continu et au format réel dans l'étape précédente, on doit maintenant choisi un format binaire spécifique de codage des données pour l'implémentation des algorithmes de commande sur la carte FPGA.

Dans cette deuxième phase (Figure 2.4), la partie puissance reste identique au modèle décrit et utilisé dans l'étape précédente.

Tandis que les architectures de commande doivent être modifiés et remplacés par des architectures basées sur des éléments de la bibliothèque XSG.

L'environnement de simulation reste celui de Matlab/Simulink. Mais Il faut marquer que le passage d'un modèle Simulink à un modèle en XSG n'est pas automatique.
Les entrées et les sorties des modèles de la bibliothèque XSG nécessitent un format entier ou réel à virgule fixe. Et doivent être obligatoirement relié à des blocs d'interfaçages Gateways In/Out.
Notre objectif final est d'implémenter des algorithmes intelligents pour la commande des machine asynchrone sur un circuit FPGA pour les tester dans un premier temps par une simulation mixtes (hardware/software) ensuite dans un contexte " Co-Simulation Hardware".

Néanmoins, le nombre de bits pour le codage des données doit être choisi pour chaque élément par le créateur. Un agrandissement arbitraire de la taille du format binaire agrandir inutilement le temps d'exécution et la surface occupée sur la carte FPGA. Par contre, on peut avoir une saturation des signaux et une perte de précision avec le choix d'une taille trop faible du format binaire qui peuvent conduire à la dégradation des performances des architectures de commande. Donc le nombre de bits des formats binaires doit être choisi avec grande attention.

La figure 2.6 présente la deuxième étape du flot de conception.

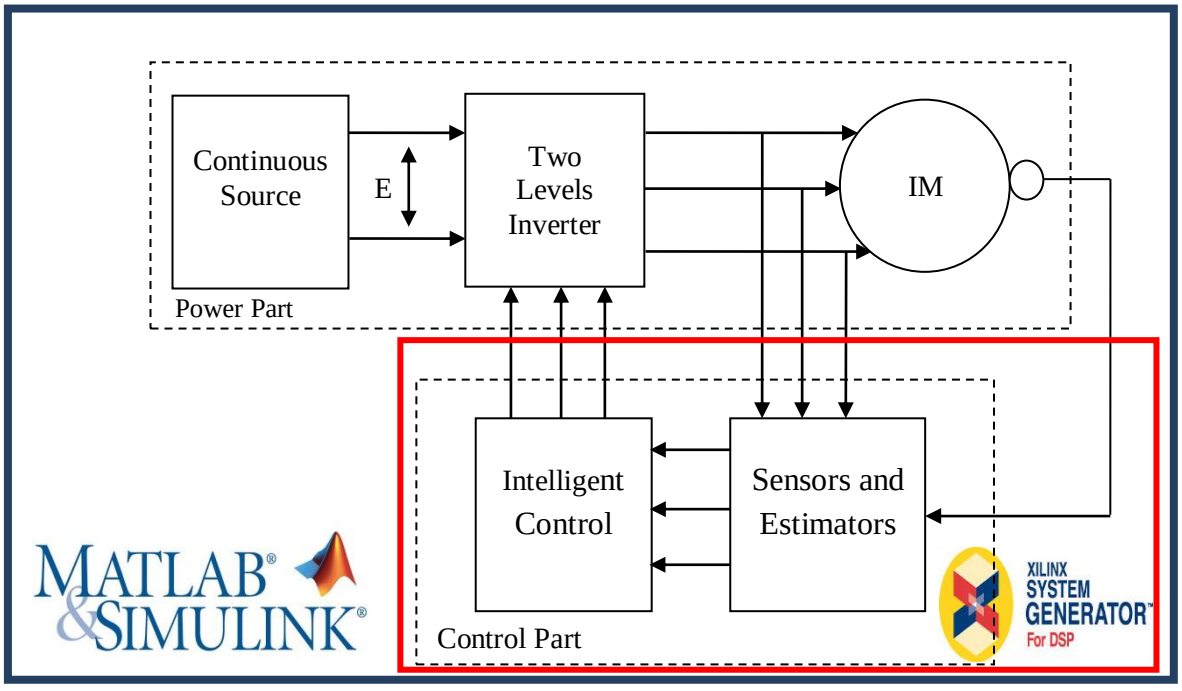


Figure 2.6 Simulation mixte

2.6.3.3. Hardware Co-Simulation

Cette étape est consacrée à l'implémentation des architectures de commande sur une carte FPGA (voir Figure 2.4). Elle est particulièrement destinée à la validation et à la vérification de l'implémentation numérique des architectures de commande sur une carte FPGA cible dans un environnement de prototypage " Co-Simulation Hardware".

Durant cette phase un fichier permettant la programmation du circuit est généré par l'outil de synthèse, dans notre cas appelé Bitstream File.

Un bloc nommé " Hardware Co-Simulation Bloc " modélisant l'architecture de commande proposée a été crée par l'outil XSG. Ce bloc peut remplacer l'ensemble de l'architecture de commande de l'étape précédente. La compilation de "Hardware Co-Simulation Bloc" permettre la programmation de la carte FPGA cible à travers une interface Ethernet point to point.

La figure 2.7 présente cette phase de conception, celle du prototypage " Hardware Co-Simulation" Une fois les architectures de commande sont validées, la carte FPGA cible intégrante les algorithmes de commande peut être utilisée dans un environnement de puissance réelle.

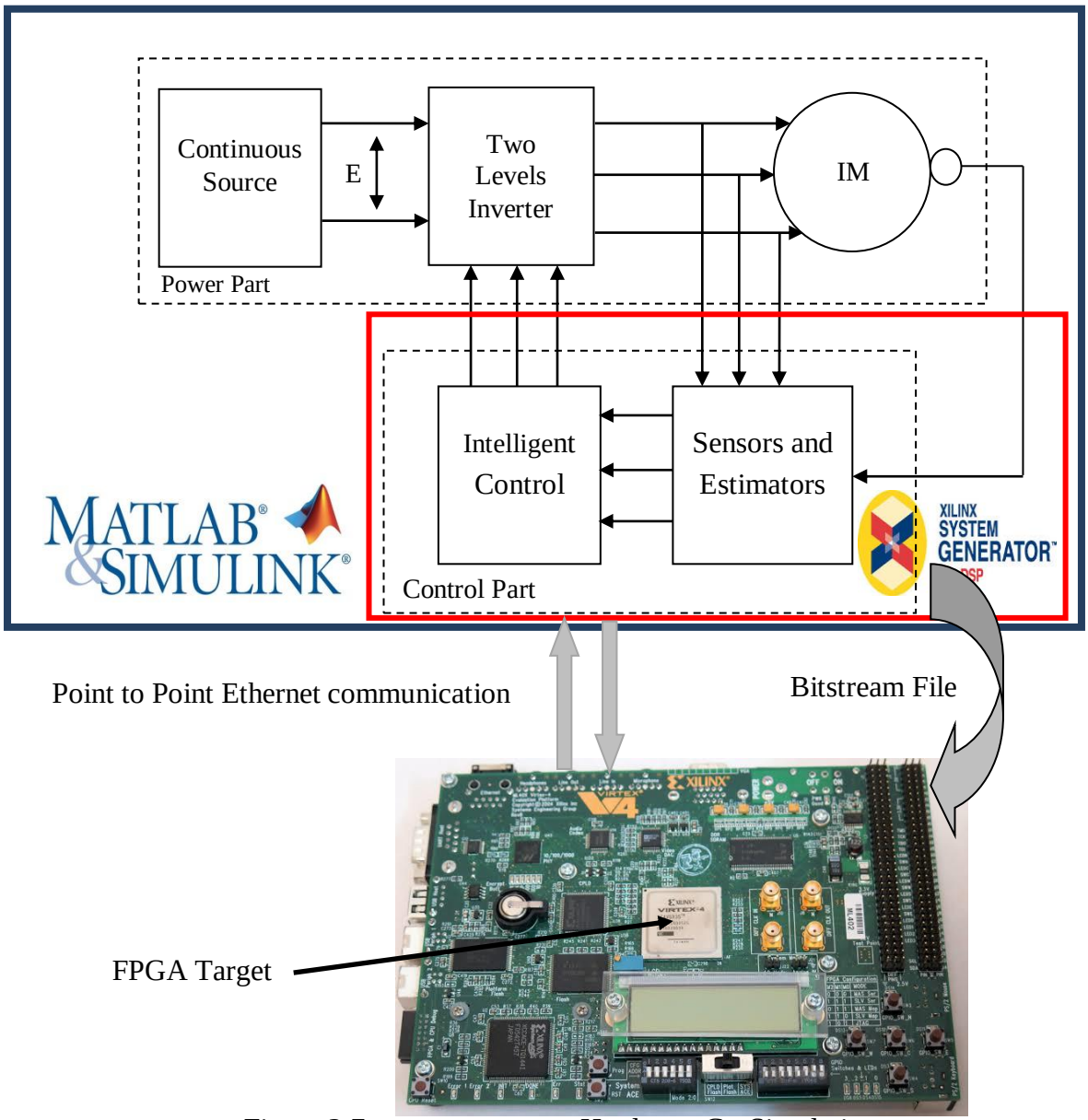


Figure 2.7 prototypage par Hardware Co-Simulation

La carte FPGA cible utilisée dans ce travail est la carte de développement ML402 de Xilinx. Elle est occupée par un circuit FPGA de type Virtex-4 de la famille Xilinx.

La figure 2.8 présente la carte ML402 [35].

Figure 2.8. Carte de développement ML402.

Spécifications de la carte ML402

La carte de développement ML402 comprend les éléments suivants [35]:

- un composant Virtex-4 FPGA: XC4VSX35-FF668-10
- une mémoire SRAM 64-MB, 32-bit/ 266-MHz
- une entrée et une sortie d'horloge différentielles.
- un oscillateur à quartz de fréquence 100-MHz.
- Interrupteurs DIP, LEDs et boutons poussoirs à usage général.
- AC97 codec audio.
- Port série RS-232.
- Afficheur LCD 16-caractères x 2-lignes
- Un EEPROM 4-Kb
- Sortie VGA: 140 MHz / 24-bit video DAC.
- Connecteurs clavier et sourie PS/2.
- Contrôleur de configuration Système ACE™ CompactFlash.
- SRAM synchrone 9 Mb.
- Puce flash linéaire Intel StrataFlash (8 MB).
- Trois vitesses Ethernet 10/100/1000.
- Puce d'interface USB.
- Xilinx XC95144XL CPLD
- Platform de configuration Flash Xilinx XCF32P.

- Port de configuration JTAG.
- Adaptateur 5V @ 3A AC.
- LED indicateur d'alimentation.

2.6.3.4. La programmation de la carte FPGA cible

Afin d'implémenter sur FPGA des architecteurs proposées pour l'implémentation hardware des techniques de commandes modélisé à l'aide XSG, il faut réaliser successivement les étapes suivantes [34]:

- la génération du code VHDL synthétisable, à partir des composantes de la bibliothèque XSG.
- la synthèse logique, un outil appelé synthétiseur prend en entrée du code VHDL synthétisable et des contraintes (Temps et Espace) convertit le code en modules disponibles sur la puce utilisée.
- L'implémentation qui se fait en deux étapes (Le placement et Le routage). Le placement consiste à choisir des structures spécifiques sur un modèle de la puce et à y fixer des modules produits par la synthèse. Le routage consiste à établir des connexions entre les structures choisis par le processus de placement.

On peut extraire les métriques d'implémentation suit à la synthèse ou l'implémentation (Le nombre de blocs logiques et autres ressources utilisés ; Le délai et la fréquence maximale d'horloge estimés ; La puissance consommée estimée). Si ces métriques ne rencontrent pas les spécifications, il faut modifier la description du circuit.

- La dernière étape consiste à Générer un fichier de configuration pour la puce utilisée.

Une fois que le fichier de programmation du FPGA a été généré, tous les blocs XSG sont remplacés dans l'environnement Matlab/Simulink par un unique bloc "Hardware Co-Simulation" correspondant aux algorithmes implémentés sur le FPGA. La connexion entre la carte cible et l'ordinateur est réalisée à l'aide d'un câble Ethernet, également utilisé lors de la programmation du FPGA. Après la configuration de la carte FPGA, on peut alors effectuer le prototypage par "Hardware Co-Simulation" (voir Figure 2.7). Dans ce cas, pour chaque pas de simulation, le composant FPGA reçoit les signaux d'entrés puis il exécute les algorithmes implémentés et transmet en retour les ordres de commande des interrupteurs de l'onduleur.

2.7 Conclusion

Dans ce chapitre, et après la présentation de l'architecture des circuits FPGA et leur contribution dans la commande des systèmes électriques, une méthodologie de conception basée sur le prototypage " Hardware in the loop" a été élaboré et adapté à nos travaux de recherches. Les différentes phases de conception ont alors été détaillées et illustrées. La méthodologie proposée est capable de guider le concepteur durant toutes les phases de conception. La dernière étape du processus de conception consiste à valider les architectures proposées par une procédure de hardware Co-Simulation.

Chapitre 3:

Implémentation de la commande DTC à base des techniques intelligentes sur FPGA

3.1 Introduction

Compte tenu des avantages de la machine asynchrone en termes de robustesse et de moindre coût, c'est la machine la plus utilisée dans les entraînements électriques. Les progrès de l'électronique de puissance et de la microélectronique en ont fait un concurrent redoutable dans les domaines du contrôle de vitesse variable et de couple rapide [36].

Le contrôle direct du couple (DTC) a été introduit pour la première fois par Depenbrock et Takahashi [1-2]. C'est l'une des méthodes utilisées dans les entraînements à fréquence variable tels que la commande des machines asynchrones. Sa renommée est due à la simplicité de sa structure, son rendement élevé et son faible coût [37].

Une DTC efficace pour les systèmes d'entraînement d'un moteur asynchrone nécessite des moyennes de calcul rapides. Les processeurs de traitement numérique du signal (DSP) et les microprocesseurs sont fréquemment utilisés dans de telles applications [32][38]. Mais ces appareils ont une vitesse de traitement limitée en raison des calculs en série ; cela affecte ses performances, en particulier dans les applications en temps réel. L'utilisation des circuits FPGA offre une solution appropriée pour des calculs rapides. Une stratégie pour la programmation FPGA et le contrôle des onduleurs multi niveaux a été présentée dans [39]. Dans [29][40], une implémentation DTC basée sur FPGA a été développée. Un estimateur de couple et de flux basé sur FPGA pour la commande DTC a été implémenté dans [10]. Dans [06][41], un processus de Co-Simulation hardware sur FPGA pour la commande des machines asynchrones a été présenté.

3.2 Objectif et principe de la commande DTC

L'objectif de la commande DTC est le contrôle du couple et du flux statorique de la machine en appliquant différents vecteurs de tension à travers un onduleur multi niveaux. Le contrôle est généralement réalisé par des comparateurs à hystérésis. Le but de cette commande est de maintenir les variables contrôlées dans une bande d'hystérésis spécifiée. Le contrôleur fournit les impulsions de commutation nécessaires à l'onduleur PWM pour générer le vecteur de tension optimal alimentant le moteur asynchrone pour une condition de fonctionnement prédéfinie. Le modèle du moteur asynchrone est utilisé avec les grandeurs mesurées pour estimer le flux statorique et le couple électromagnétique requis par le schéma de commande (Fig. 3.1). L'élément le plus important qui pouvant garantir une performance DTC satisfaisante est l'estimateur du flux statorique et du couple électromagnétique.

De plus, plusieurs paramètres doivent être déterminés afin d'estimer le flux statorique et le couple électromagnétique. Leurs modèles sont adaptés aux besoins d'un entraînement contrôlé. Tout d'abord, les courants statoriques a-b-c sont transformés en coordonnées α-β, et les tensions statoriques dans le référentiel α-β sont déterminées en fonction de l'état de commutation (Sa, Sb et Sc) produit par la table de commutation [42].

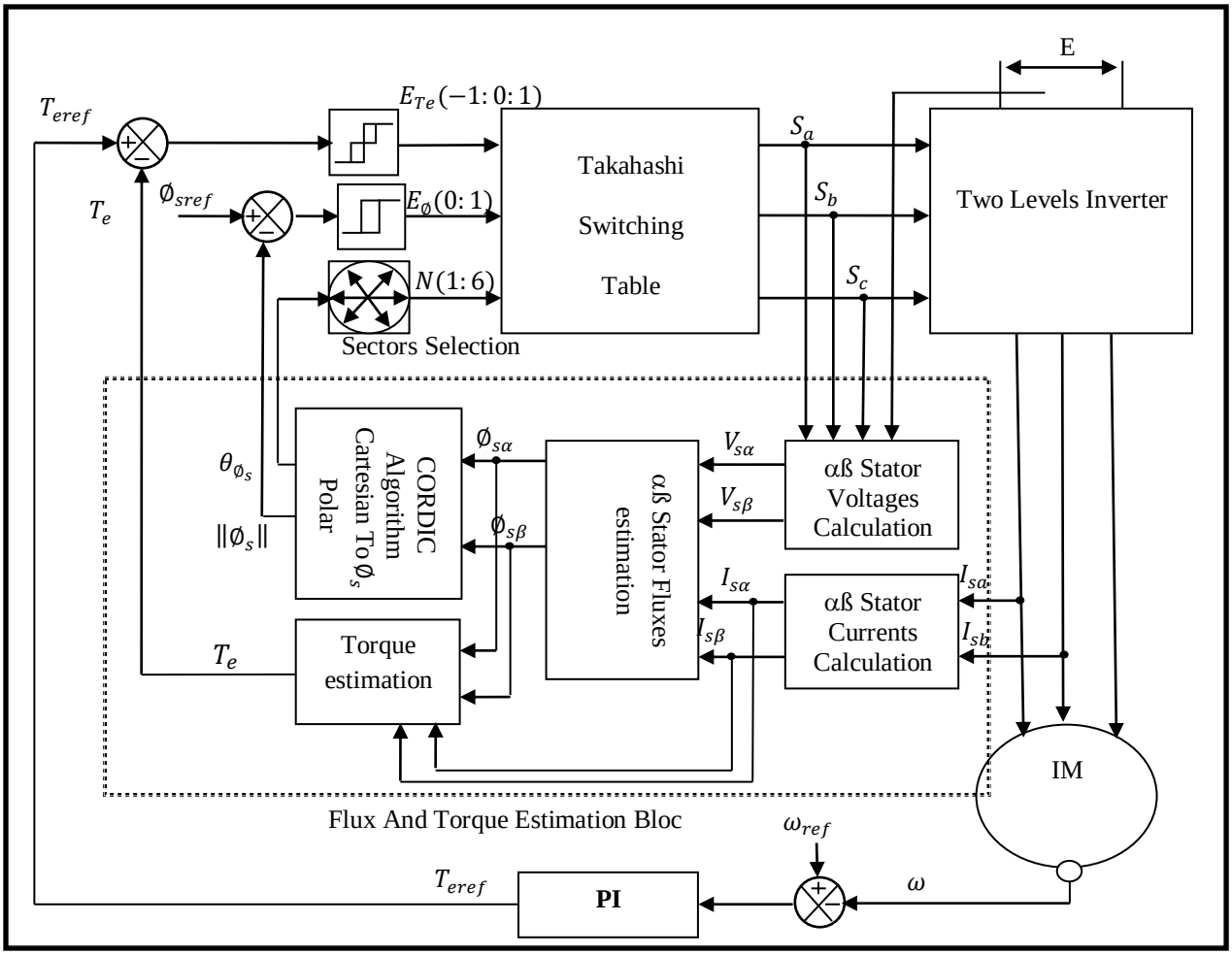


Figure 3.1 Schéma fonctionnel de la commande DTC

En utilisant le courant et la tension statoriques, le flux statorique peut être estimé en coordonnées (α-β) comme suit [42] :

$$\emptyset_{s\alpha} = \int (V_{s\alpha} - R_s i_{s\alpha}) dt \tag{3.1}$$

$$\emptyset_{s\beta} = \int \left(V_{s\beta} - R_s i_{s\beta}\right) dt \tag{3.2}$$

Avec: $(\emptyset_{s\alpha}, \emptyset_{s\beta})$: Les flux statoriques dans le plan (α-β).

$(i_{s\alpha}, i_{s\beta})$: Les courants statoriques dans le plan (α-β).

$(V_{s\alpha}, V_{s\beta})$: Les tensions statoriques dans le plan (α-β).

R_s: La resistance statorique.

L'approche d'Euler est utilisée pour effectuer l'intégrale, avec un temps d'échantillonnage (Ts), le système (3.1) et (3.2) deviennent comme suit :

$$\emptyset_{s\alpha} = \emptyset_{s\alpha Old} + T_s(V_{s\alpha} - R_s i_{s\alpha}) \tag{3.3}$$

$$\emptyset_{s\beta} = \emptyset_{s\beta Old} + T_s\left(V_{s\beta} - R_s i_{s\beta}\right) \quad (3.4)$$

Enfin, avec (p) est le nombre de paires de pôles de la machine. Le couple électromagnétique (Te) peut être estimé par [42] :

$$T_e = P\left(\emptyset_{s\alpha} i_{s\beta} - \emptyset_{s\beta} i_{s\alpha}\right) \quad (3.5)$$

Pour l'amplitude et l'angle de flux statorique ($\|\emptyset_s\|$ et $\theta_{\emptyset_s}$) peuvent être obtenu par une transformation des coordonnées cartésiennes en coordonnées polaires :

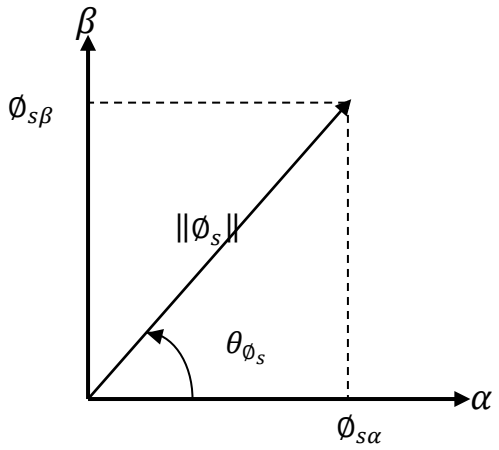


Figure 3.2 Transformation des coordonnées cartésiennes en coordonnées polaires pour le flux statorique

Malheureusement, les transformations mathématiques standards basées sur les formules: $\|\emptyset_s\| = \sqrt{{\emptyset_{s\alpha}}^2 + {\emptyset_{s\beta}}^2}$ et $\theta_{\emptyset_s} = ArcTan(\frac{\emptyset_{s\beta}}{\|\emptyset_s\|}, \frac{\emptyset_{s\alpha}}{\|\emptyset_s\|})$ ne sont pas toujours implantables en hardware et nécessitent souvent beaucoup de ressources sur la carte FPGA à cause des fonctions coûteuses en termes de consommation des ressources hardwares, telles que la multiplication et la division binaires, la racine carrée et la fonction ArcTan. Pour surmonter cette contrainte, notre estimateur utilise une version de l'algorithme CORDIC (coordinate rotational digital computer) [21] pour calculer l'amplitude et l'angle du flux statorique, en utilisant des opérateurs simplement implémentables sur FPGA comme l'addition, la soustraction et le décalage des chiffres binaires [22].

Dans [43], les auteurs ont utilisé deux multiplicateurs binaires et une racine carrée sans restauration de 62 bits [44] pour calculer l'amplitude du flux statorique. Et en [31] ont utilisé deux multiplicateurs binaires, un bloc CORDIC généralisé racine carrée et un bloc CORDIC généralisé ArcTan pour estimer l'amplitude et l'angle du flux statorique. Deux multiplicateurs binaires et un bloc CORDIC généralisé racine carrée ont été utilisés pour estimer l'amplitude du flux statorique dans [42].

Notre solution utilise uniquement un bloc CORDIC réduit et un multiplicateur pour estimer simultanément l'amplitude et l'angle du flux statorique.

3.3 Implémentation d'un bloc d'estimation de la DTC sur FPGA

Ce travail présente un estimateur amélioré de couple et de flux statorique basé sur FPGA pour la commande DTC d'un moteur asynchrone, qui permet des calculs très rapides. Ces améliorations sont effectuées par le calcul de l'opération d'intégration discrète du flux statorique en utilisant l'approche d'Euler et le calcul de l'angle et de l'amplitude du flux statorique en utilisant l'algorithme CORDIC (Transformation des coordonnées cartésiennes en coordonnées polaires). Les modèles du diagramme d'estimation sont simplifiés et organisés en sous-blocs pour faciliter la description hardware (Figure 3.3)[45].

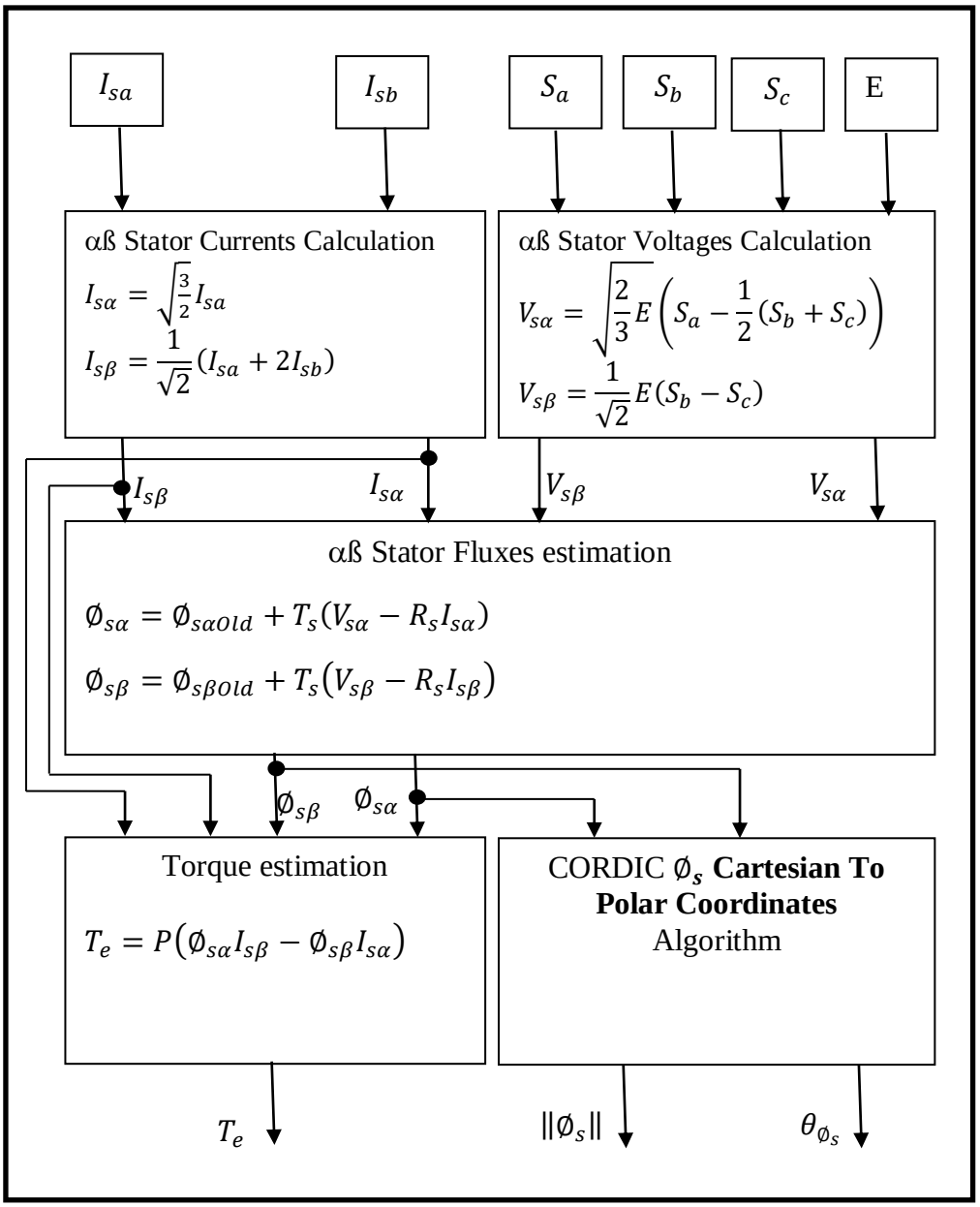


Figure 3.3 Flot de conception hardware de l'estimateur de flux et de couple

Le XSG Xilinx System Generator est un outil de modélisation développé par Xilinx pour le package MATLAB/SIMULINK est largement utilisé à des fins de vérification et de développement d'algorithmes dans FPGA et DSP. Le principal avantage de cet outil est de

traduire la modélisation d'un schéma conceptuelle en une implémentation matérielle et de générer le code VHDL sans revenir à une programmation difficile.

De plus, les ressources hardwares FPGA sont représentées par des blocs de l'outil XSG développés pour le package MATLAB/SIMULINK. Par conséquent, le temps de mise en œuvre est réduit car l'algorithme n'a besoin d'être simulé et modélisé qu'une seule fois. La conception de notre estimateur de flux et de couple utilisant le XSG est basée sur un modèle mathématique (Figure 3.3).

3.3.1 Implémentation du calculateur de courant et de tension statorique dans le plan α-β

La conception XSG du calculateur de tension et de courant statorique dans le plan (α-β) est illustrée à la Figure 3.4.

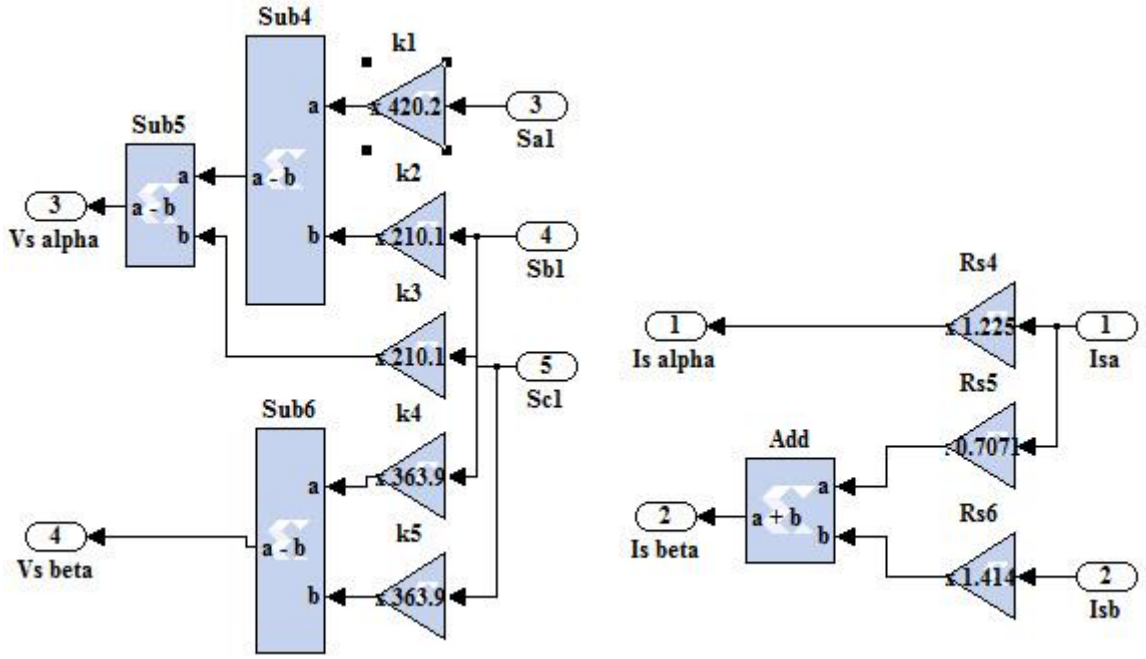


Figure 3.4 Implémentation du calculateur de courant et de tension statorique dans le plan (α-β)

3.3.2 Implémentation de l'estimateur des flux statoriques dans le plan (α-β)

La figure 3.5 montre la description du hardware pour l'estimation du flux statorique dans le plan α-β à l'aide de la boîte à outils XSG et à partir des équations (3.3) et (3.4).

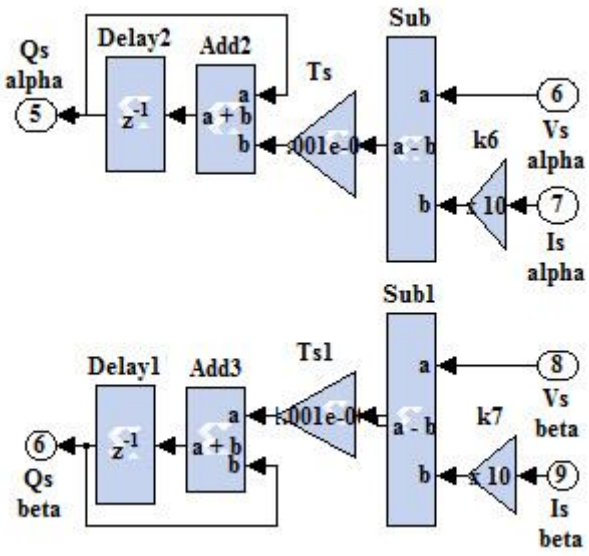


Figure 3.5 Description hardware pour l'estimation du flux statorique dans le plan α-β à l'aide du XSG

3.3.3 Implémentation de l'estimateur de couple électromagnétique

La conception XSG du module d'estimation de couple électromagnétique propose dans l'équation (3.5) est illustrée à la Figure 3.6.

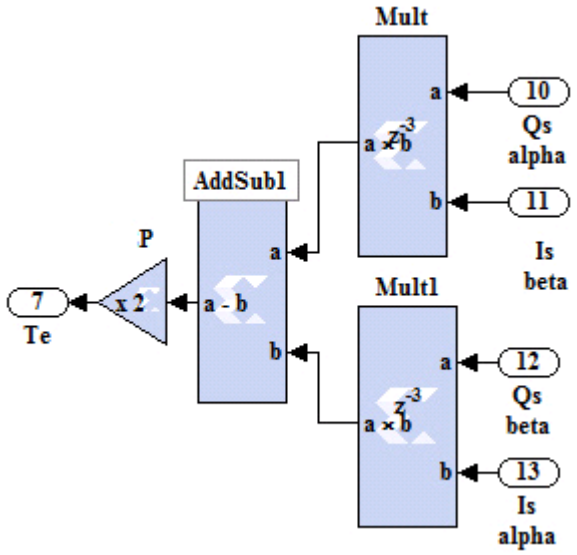


Figure 3.6 Description hardware de l'estimateur de couple électromagnétique utilisant le XSG

3.3.4 Implémentation de la Transformation des coordonnées cartésiennes en coordonnées polaires pour le flux statorique

La Transformation des coordonnées cartésiennes en coordonnées polaires pour le flux statorique consiste à transformer les coordonnées cartésiennes du flux statorique dans le plan (α-β) précédemment implémentés (Figure 3.5) en coordonnées polaires, l'amplitude et l'angle

de flux statorique ($\|\emptyset_s\|$ et $\theta_{\emptyset_s}$) en utilisant le mode vecteur de l'algorithme de CORDIC qui permet de calculer le module et la phase d'un vecteur à partir de ces coordonnées cartésiennes.

L'algorithme CORDIC présenté dans le paragraphe 4.2 de premier chapitre par les systèmes d'équations (1.14) et (1.16) est implémenté en hardware par une architecture VHDL de la forme :

```
Entity CORDIC is
        Port ( Qsd : in  STD_LOGIC_VECTOR (15 downto 0);
               Qsq : in  STD_LOGIC_VECTOR (15 downto 0);
        Magnitude : out  STD_LOGIC_VECTOR (15 downto 0);
           AngleQ : out  STD_LOGIC_VECTOR (15 downto 0));
End  CORDIC;
```

Le code VHDL de l'architecture CORDIC est implémenté dans une Black Box avec XSG:

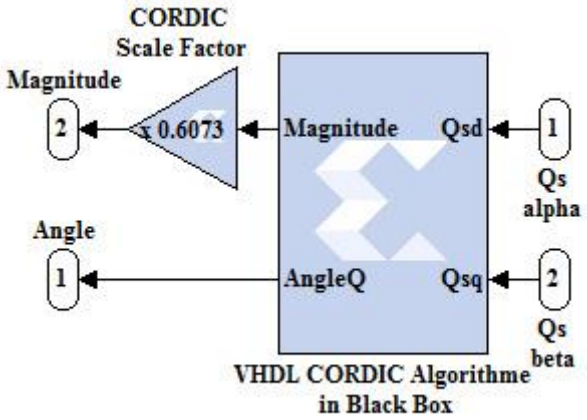


Figure 3.7 Implementation de l'algorithme CORDIC avec XSG

3.4 Implémentation de la DTC classique sur FPGA

La commande DTC conventionnelle (Figure 3.1) se compose de deux parties, une partie d'estimation et une autre partie pour le control. La partie du control utilise les grandeurs calculées par la partie d'estimation (flux statorique et couple électromagnétique) pour généré le vecteur des commandes de l'onduleur, cette partie se compose principalement d'un sélectionneur des secteurs Nj donnant l'information sur la position du vecteur de flux statorique, un comparateur à hystérisais de deux niveaux pour le contrôle du vecteur de flux statorique, un comparateur à hystérisais de trois niveaux pour le contrôle du couple électromagnétique et une table de commutation qui permet l'élaboration du vecteur de commande en sortie en fonction de l'état des variables booléennes à la sortie des deux correcteurs de flux et du couple électromagnétique et le sélectionneur des secteurs.

3.4.1 Implémentation du sélectionneur des secteurs

Le plan de coordonnées α-β est divisé en 6 secteurs. Ce bloc détermine dans quel numéro de secteur se situe le vecteur de flux statorique à un instant d'échantillonnage donné. Cela se fait en comparant l'angle de flux avec les limites de chaque secteur. Pour cela nous utilisons deux comparateurs, deux additionneurs et un multiplexeur en XSG :

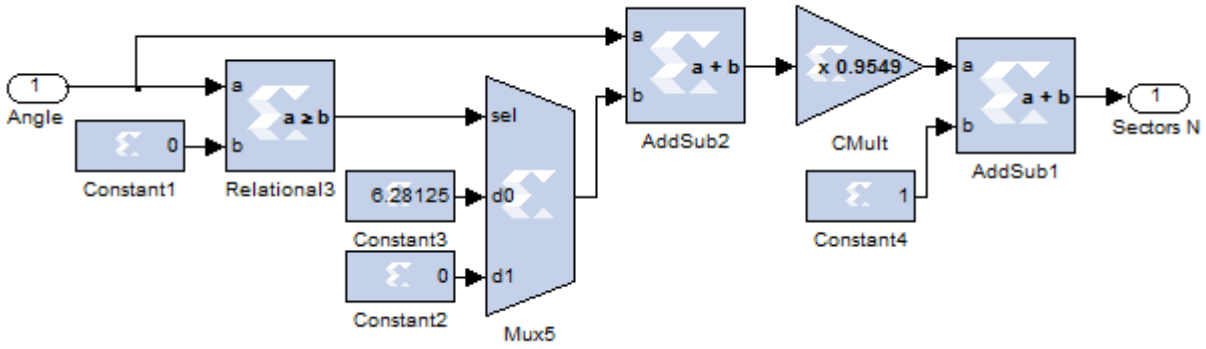


Figure 3.8 Implémentation de la sélection des secteurs avec XSG

3.4.2 Implémentation des contrôleurs d'hystérésis

Les valeurs de couple et de flux statorique estimées par le bloc d'estimation sont comparées à leurs valeurs de références à l'aide des comparateurs de couple et de flux d'hystérésis à trois et deux niveaux, respectivement, la mise en œuvre XSG de ces comparateurs comme illustré à la Fig.3.9.

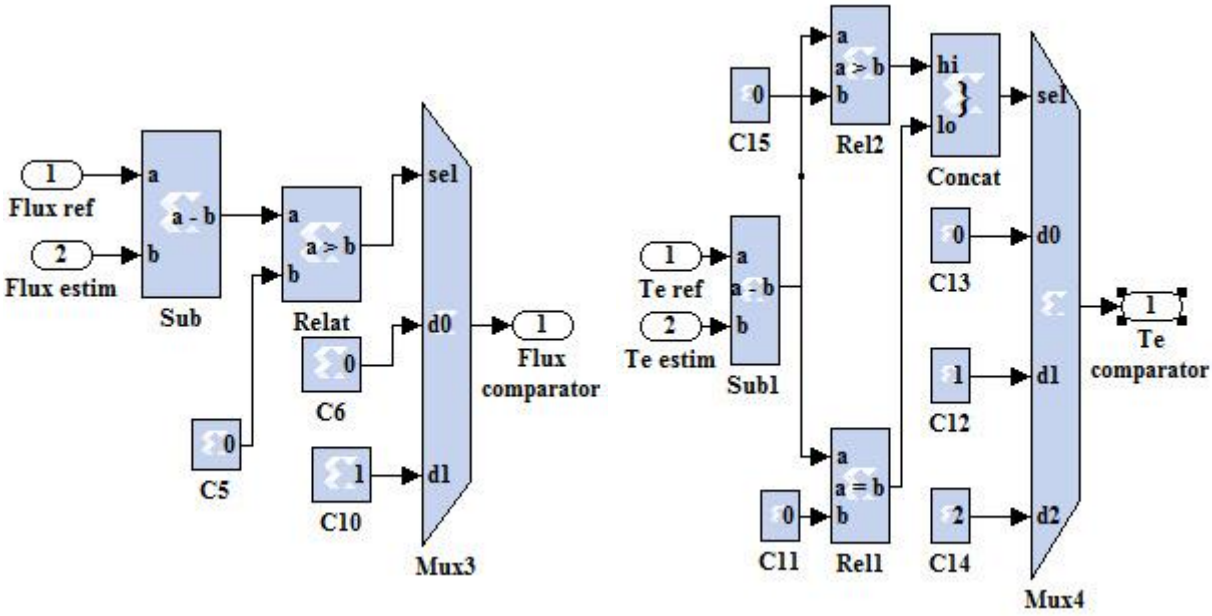


Figure 3.9 Implémentation des comparateurs à hystérésis avec XSG

3.4.2 Implémentation de la table de commutation de Takahashi

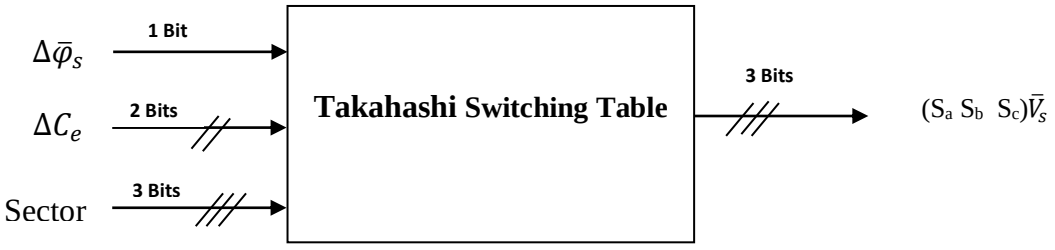
Les signaux de commutation de l'onduleur sont décidés sur la base du tableau de commutation de Takahashi présenté dans le tableau 3.I. Le tableau se compose de vecteurs nuls et non nuls (actifs). Les vecteurs zéro sont (0,0,0) et (1,1,1) qui arrêtent le vecteur de champ, réduisant ainsi le couple. D'autre part, tous les six autres vecteurs connus sous le nom de vecteurs actifs qui font avancer le champ, entraînant une augmentation du couple [30].

$V_i = (S_a\ S_b\ S_c)$							
$\Delta\bar{\varphi}_s, \Delta T_e, N$		N_1	N_2	N_3	N_4	N_5	N_6
$\Delta\bar{\varphi}_s=1$	ΔT_e= 1	(110)	(010)	(011)	(001)	(101)	(100)
	ΔT_e= 0	(111)	(000)	(111)	(000)	(111)	(000)
	ΔT_e= -1	(101)	(100)	(110)	(010)	(011)	(001)
$\Delta\bar{\varphi}_s=0$	ΔT_e=1	(010)	(011)	(001)	(101)	(100)	(110)
	ΔT_e= 0	(000)	(111)	(000)	(111)	(000)	(111)
	ΔT_e= -1	(001)	(101)	(100)	(110)	(010)	(011)

Tableau 3.1. La table de Takahashi pour la commande DTC

La table de commutation Takahashi est implémentée par une architecture hardware en VHDL. Le langage de description hardware VHDL nous permet de définir les entrées/sorties de notre architecture. Un bit d'entrée pour l'erreur de flux $\Delta\bar{\varphi}_s$(0 ou 1), Deux bits d'entrée pour l'erreur de couple ΔT_e(-1,0 ou 1), Trois bits d'entrée pour le sélectionneur des secteurs (N_1, N_2, N_3, N_4, N_5 ou N_6) et Trois bits de sortie pour le vecteur de commande $\bar{V}_s$(S_a, S_b et S_c).

La figure suivante présente les entrées/sorties de l'architecture proposée et le code VHDL correspondant :




```
entity DTC is
    Port ( Selec : in  STD_LOGIC_VECTOR (2 downto 0);
           Cflex : in  STD_LOGIC;
           Ccpl : in  STD_LOGIC_VECTOR (1 downto 0);
           S135 : out STD_LOGIC_VECTOR (2 downto 0) );
end DTC;
```

Figure 3.10 Architecture VHDL de la table de commutation de Takahashi

L'architecture de la table de commutation Takahashi est implémentée en VHDL par une affectation en mode compétitif de la forme:

$$\begin{aligned}\bar{V}_{Sortie} <= \ & \bar{V}_1 \ when \ Condition1 \\ & else \ \bar{V}_2 \ when \ Condition2 \\ & else \ \bar{V}_3 \ when \ Condition3 \\ & else \ \bar{V}_4 \ when \ Condition4 \\ & else \ \bar{V}_5 \ when \ Condition5 \\ & else \ \bar{V}_6 \ when \ Condition6 \\ & else \ \bar{V}_7 \ when \ Condition7 \\ & else \ \bar{V}_0\end{aligned}$$

```
architecture Behavioral of DTC is

begin
	S135 <= "100" when  ((Cflex='0') and  (Selec="101") and  (Ccpl="10")) or ((Cflex='0') and  (Selec="011") and  (Ccpl="00"))
	else    "110" when  ((Cflex='0') and  (Selec="110") and  (Ccpl="10")) or ((Cflex='0') and  (Selec="100") and  (Ccpl="00"))
	else    "010" when  ((Cflex='0') and  (Selec="001") and  (Ccpl="10")) or ((Cflex='1') and  (Selec="010") and  (Ccpl="10"))
	else    "011" when  ((Cflex='0') and  (Selec="010") and  (Ccpl="10")) or ((Cflex='0') and  (Selec="110") and  (Ccpl="00"))
	else    "001" when  ((Cflex='0') and  (Selec="001") and  (Ccpl="00")) or ((Cflex='0') and  (Selec="011") and  (Ccpl="10"))
	else    "101" when  ((Cflex='0') and  (Selec="100") and  (Ccpl="10")) or ((Cflex='0') and  (Selec="010") and  (Ccpl="00"))
	else    "000" when  ((Cflex='0') and  (Selec="001") and  (Ccpl="01")) or ((Cflex='0') and  (Selec="011") and  (Ccpl="01"))
	else    "111" when  ((Cflex='0') and  (Selec="010") and  (Ccpl="01")) or ((Cflex='0') and  (Selec="100") and  (Ccpl="01"))
	else    "000";
end Behavioral;
```

Figure 3.11 L'implémentation VHDL de la table de commutation de Takahashi

Le vecteur de signal de sortie $\bar{V}_s = \begin{pmatrix} Sa \\ Sb \\ Sc \end{pmatrix}$ divisé par des Slices pour générer les trois signaux (Sa, Sb et Sc) de la commande de l'onduleur. Une Slice extrait une plage donnée de bits de chaque échantillon d'entrée et la présente à la sortie.

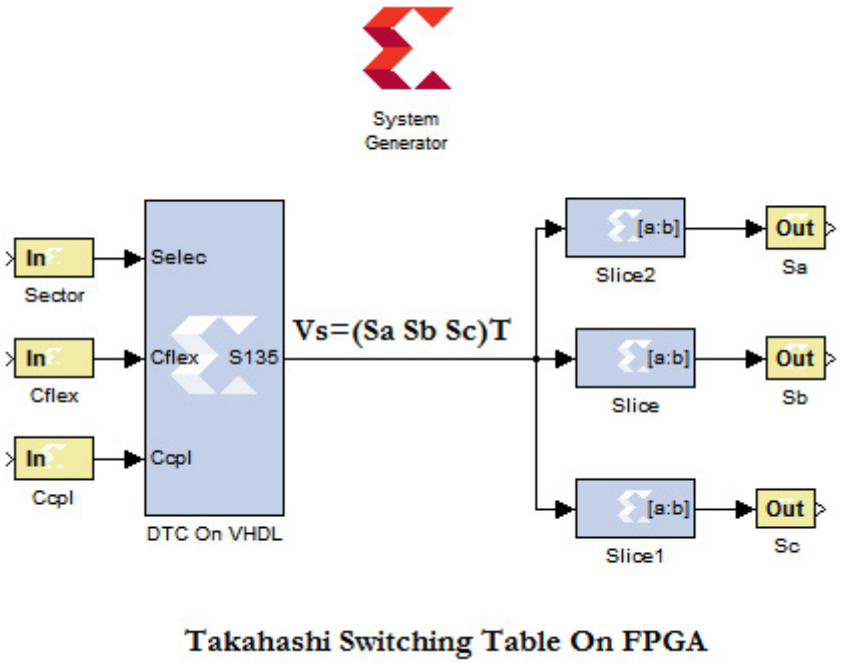


Figure 3.12 Table de commutation Takahashi avec XSG

3.5 Implémentation d'une Commande DTC Floue sur FPGA

Le sujet développé dans ce travail de recherche s'articule principalement autour de l'exploitation des nouvelles solutions technologiques pour la réalisation d'une Commande Directe de Couple pour la machine asynchrone à base des techniques intelligentes dans un environnement hardware basé sur un circuit FPGA. Le but principal de cette implémentation est la réduction des ondulations du flux statorique et du couple électromagnétique. A cet effet, dans l'architecture hardware proposée, la table de Takahashi et les deux comparateurs à hystérésis (Table 3.1) seront remplacés par un système d'inférence flou. La figure suivante présente la structure de contrôle de la machine à induction basée sur DTC floue.

.

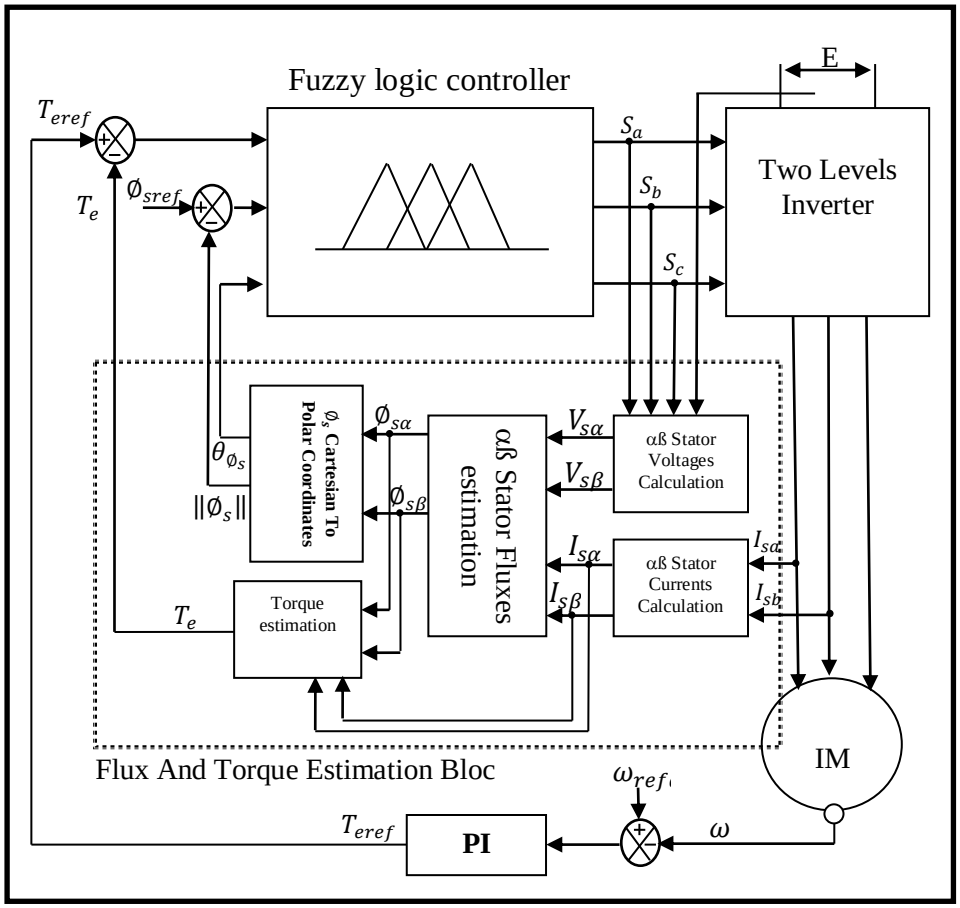


Figure 3.13 Structure de contrôle de la machine à induction basée sur DTC floue

L'implémentation hardware d'un système d'inférence flou consiste à implémenter les trois phases d'une régulation par logique floue: Fuzzification, Inférences floues et Défuzzification.

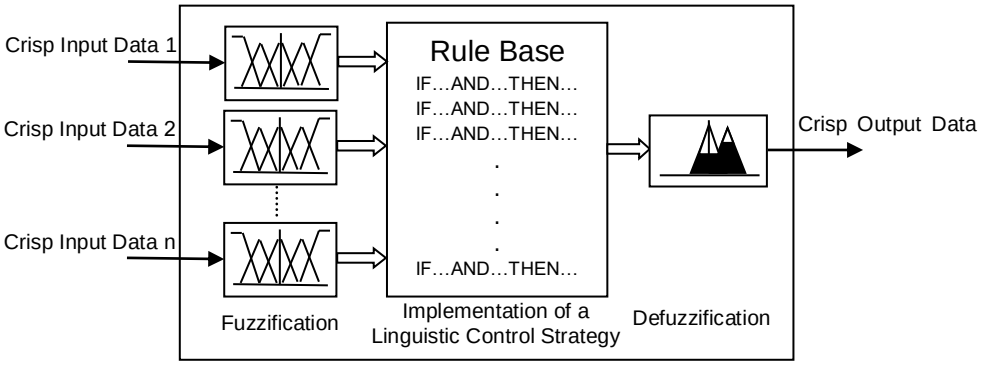


Figure 3.14 Les composants du système d'inférence floue

3.5.1 Implémentation de l'unité "Fuzzification"

Dans le processus de fuzzification les données d'entrée sont converties en valeurs linguistiques floues. Pour la détermination de degré d'appartenance à un ensemble flou à partir d'une fonction d'appartenance Il y a deux solutions hardwares dans la littérature. La première est la solution orientée mémoire où pour chaque valeur d'entrées les valeurs de sortie sont calculées et sauvegardées dans un bloc mémoire en off-line afin de simplifier le changement d'une fonction d'appartenance. L'autre solution est la méthode orientée calcul, dans laquelle, uniquement les caractéristiques des fonctions d'appartenances sont sauvegardées dans les mémoires. L'avantage de cette solution est la simplification des calculs on-line des valeurs de sortie pour chaque fonction d'appartenance. Dans ce travail, on a choisit la solution orientée mémoire.

Chaque variable d'entrée/sortie linguistique est représentée par des tables, une table pour le degré d'appartenance de chaque valeur linguistique. Ces tables sont implémentées en hardware par des blocs mémoires ROMs adressables par une seule entrée, tel que les casées mémoires contient le degré d'appartenance de la valeur linguistique et l'espace d'adressage mémoire donne une image sur l'univers du discours discrétisé par exemple pour un univers du discours normalisé [0, 1] discrétisé en 512 points on utilise un espace d'adressage [0:511].

Les fonctions d'appartenances Erreur flux, Erreur couple, les secteurs et les vecteurs de sortie sont illustrées par les figures suivantes :

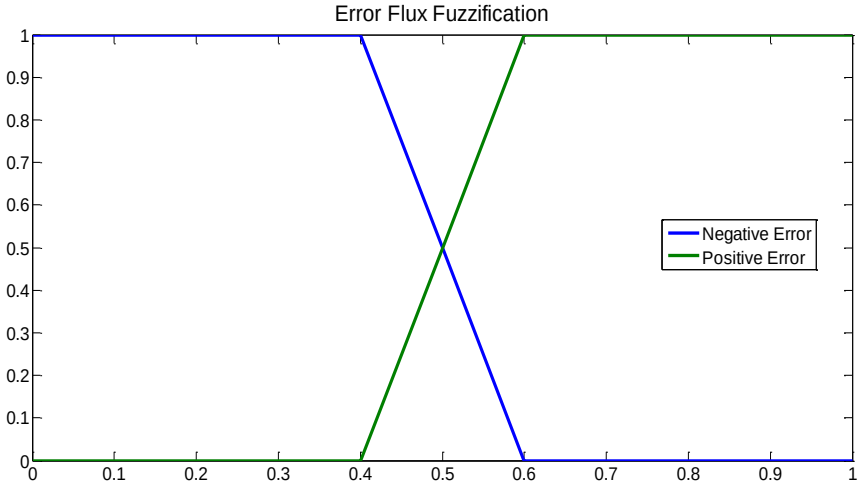


Figure 3.15 Fuzzification de l'erreur du flux statorique

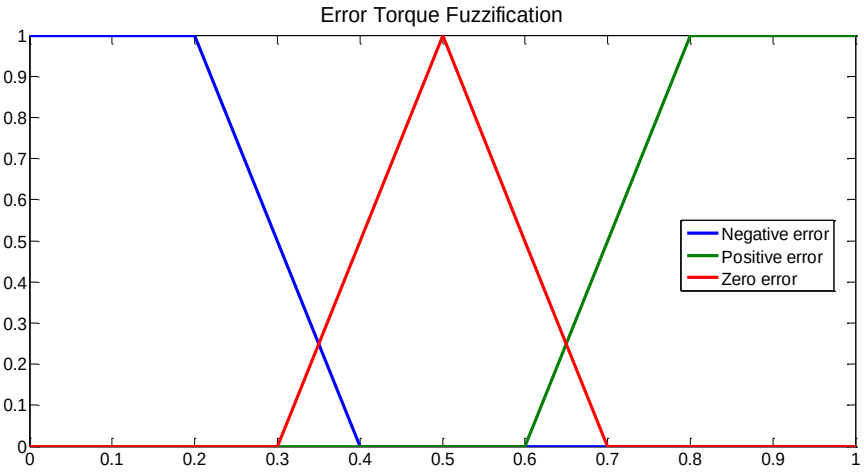


Figure 3.16 Fuzzification de l'erreur du couple électromagnétique

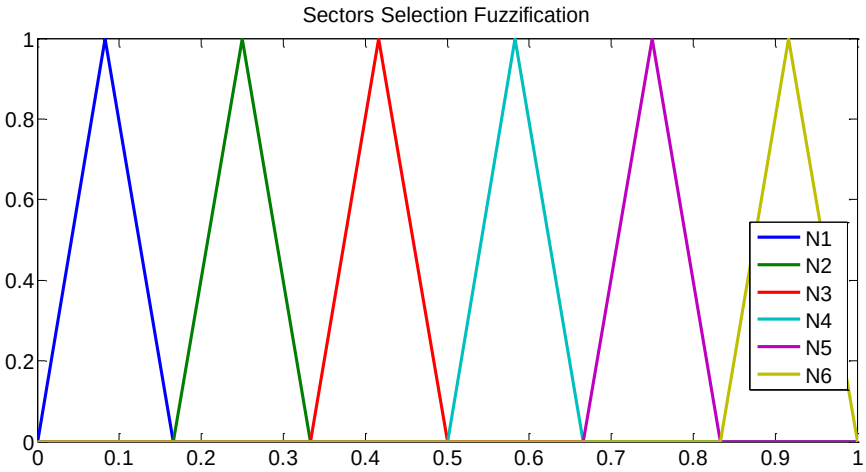


Figure 3.17 Fuzzification de l'angle du flux statorique

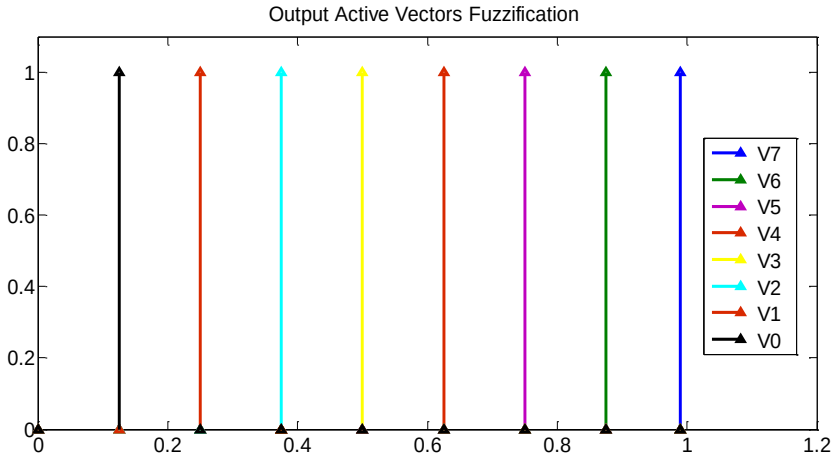


Figure 3.18. Fuzzification de la sortie

L'implémentation matérielle de ces fonctions est présentée par les architectures hardwares des figures 3.19, 3.20, 3.21 et 3.22.

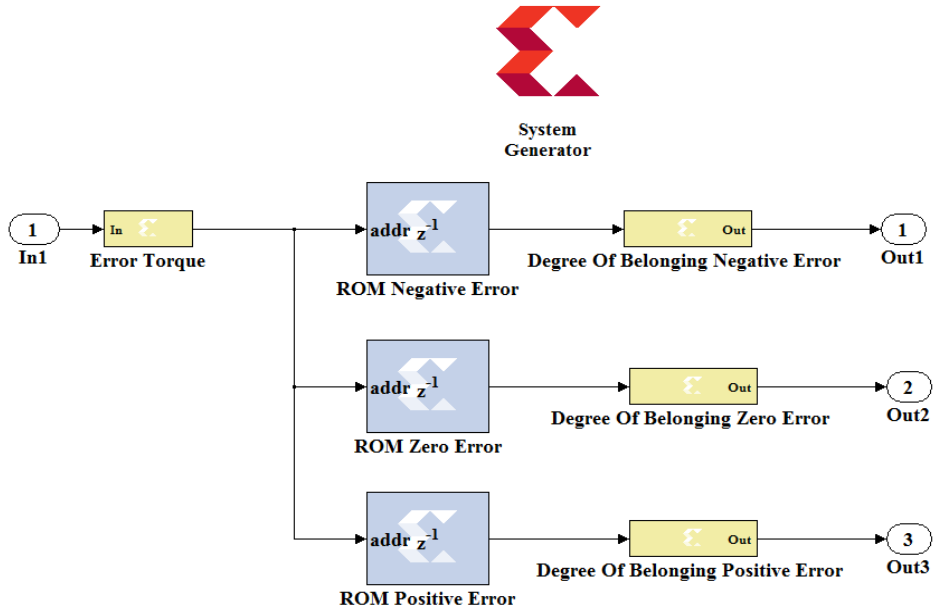


Figure 3.19 L'architecture hardware de la variable linguistique " Erreur de Couple"

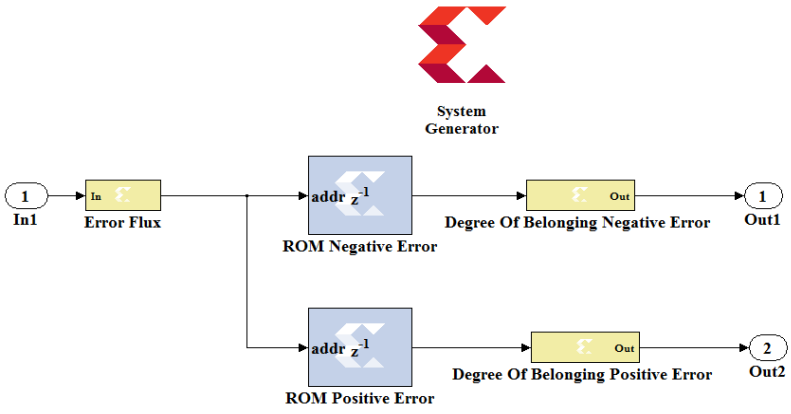


Figure 3.20 L'architecture hardware de la variable linguistique " Erreur de Flux "

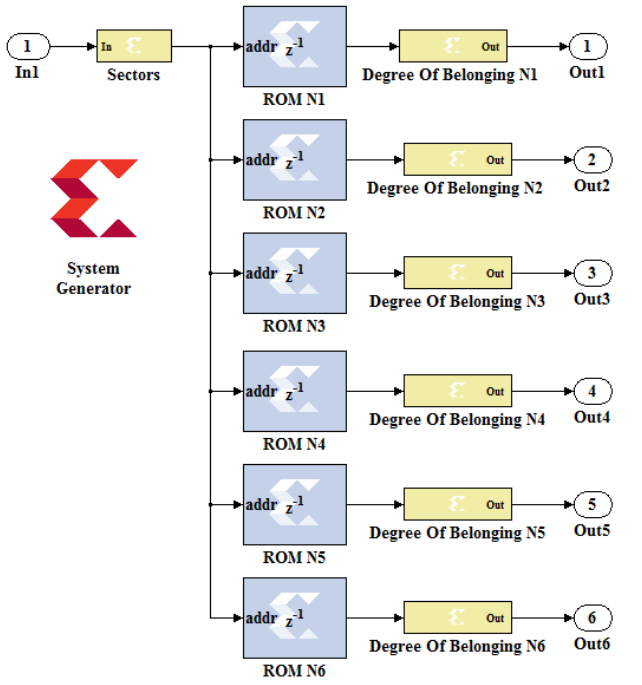


Figure 3.21 L'architecture hardware de la variable linguistique " Sectors "

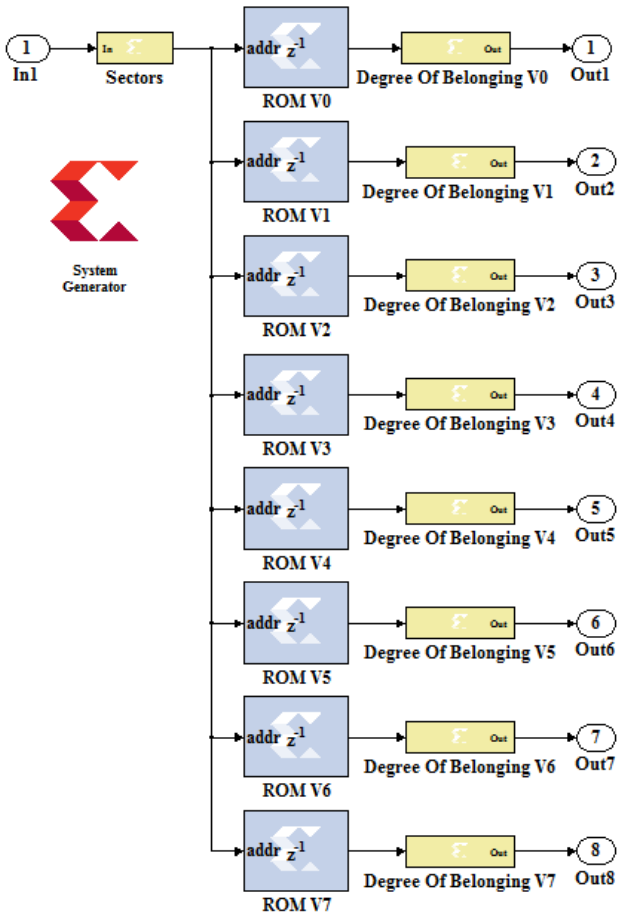


Figure 3.22 L'architecture hardware de la variable linguistique " Vecteur de la sortie "

L'outil Resource Estimator de Xilinx permet d'estimer les ressources hardwares nécessaires pour implémenter chaque variable linguistique, la figure suivante présente les ressources estimées pour la variable linguistique " Erreur de Couple" :

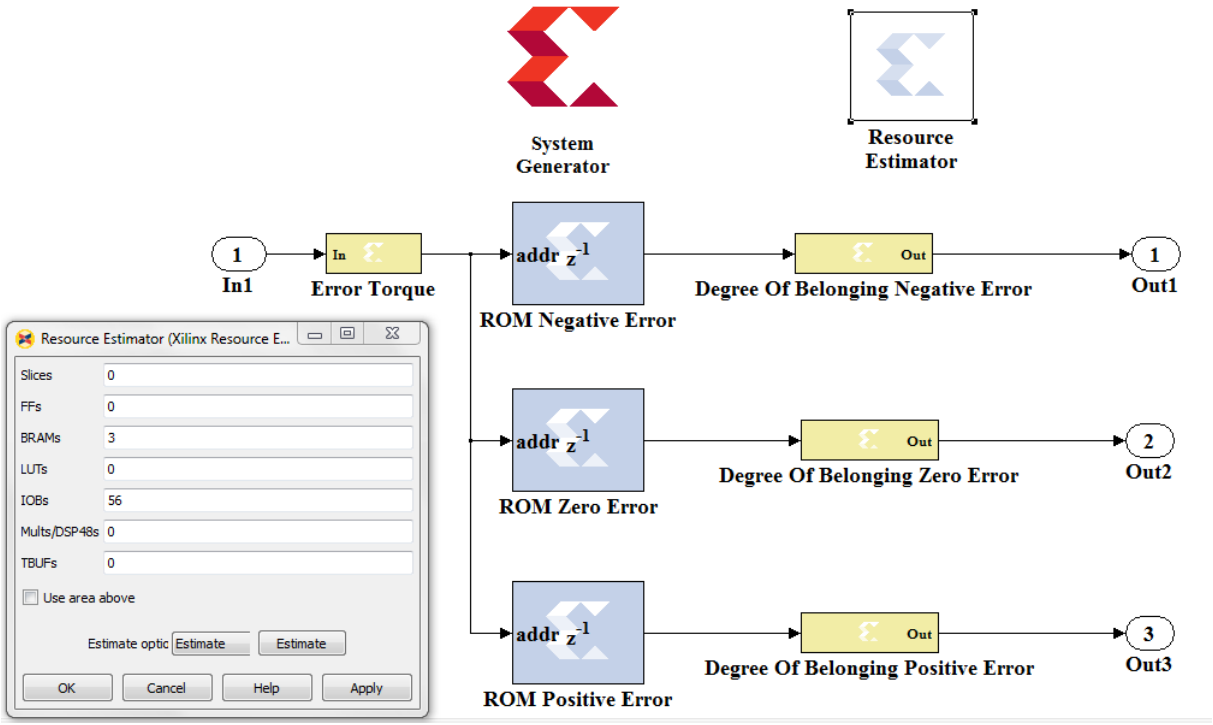


Figure 3.23 Les ressources hardwares consommées par la variable linguistique " Erreur de Couple"

3.5.2 Implémentation de l'unité "Moteur d'inférence"

La représentation hardware de l'unité "Moteur d'inférence" est donnée par la figure 3.24. Ce bloc reçoit comme entrée les trois sorties de l'unité "Fuzzification", le bloc de sélection des règles permet la création d'une base des règles formée par 36 règles. Ces règles sont obtenues par la réalisation de toutes les combinaisons possibles entre les valeurs floues d'entrée (Erreur flux, Erreur couple et les secteurs de l'angle du flux statorique).

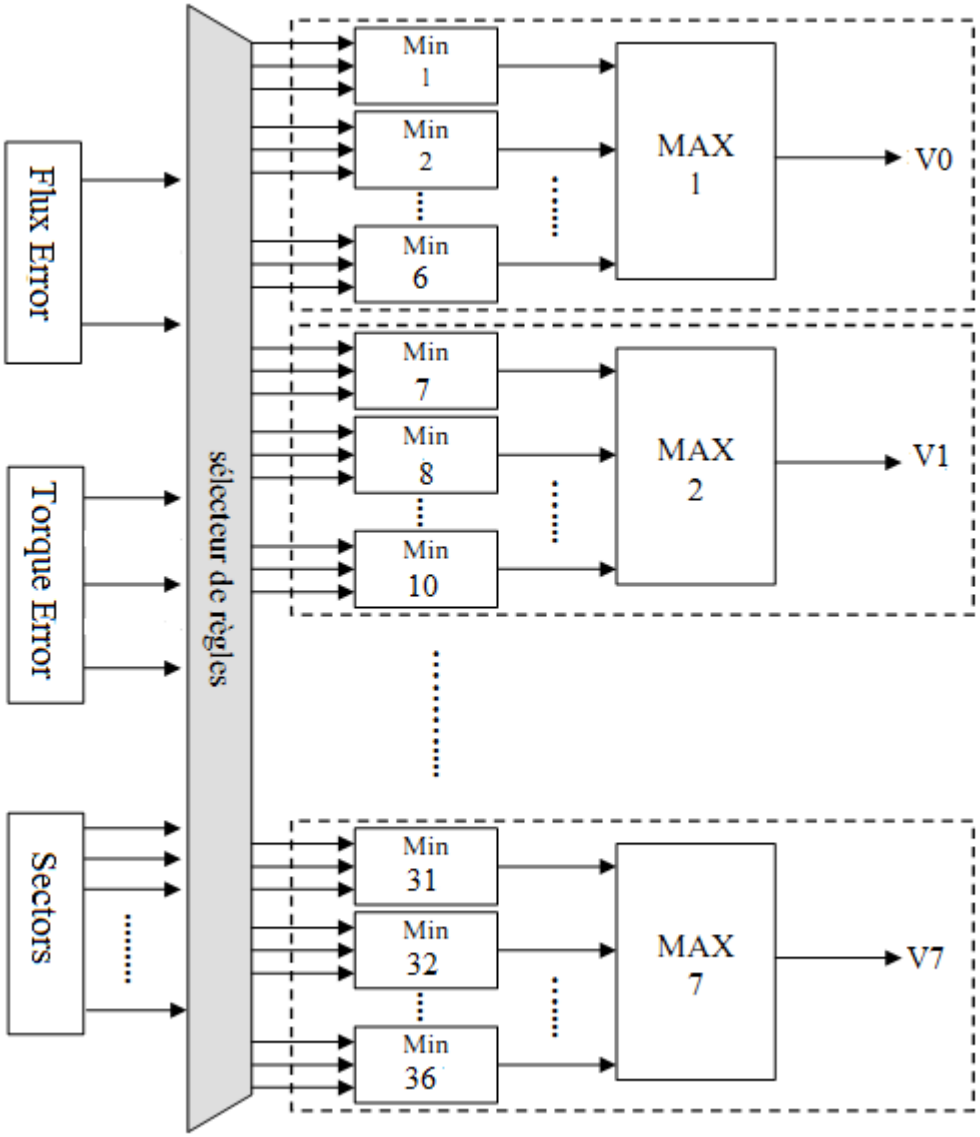


Figure 3.24 Architecture du module moteur d'inférence

L'implémentation hardware sur "Xilinx System Generator" des opérateurs (min/max) à deux entrées se fait par un comparateur et un multiplexeur 2-1. La figure suivante montre le câblage des fonctions min/max :

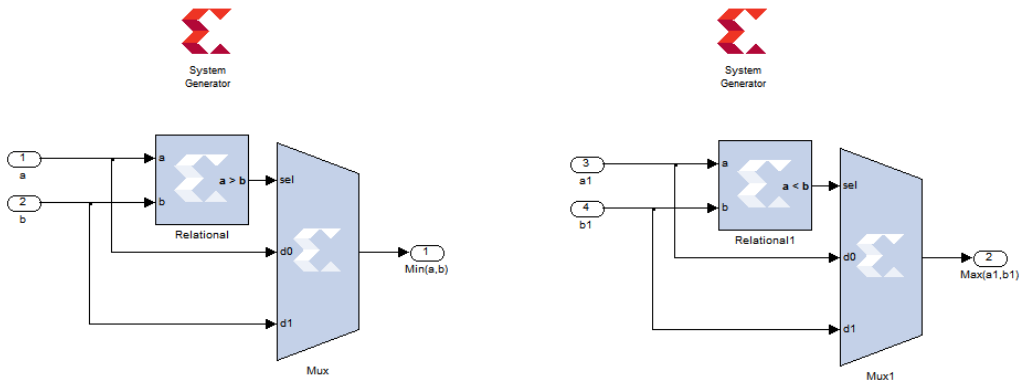


Figure 3.25 L'implémentation des fonctions Min et Max sur XSG

L'utilisation de l'outil Resource Estimateur nous permet d'estimer les ressources hardware consommées par un operateur Min à deux entées:

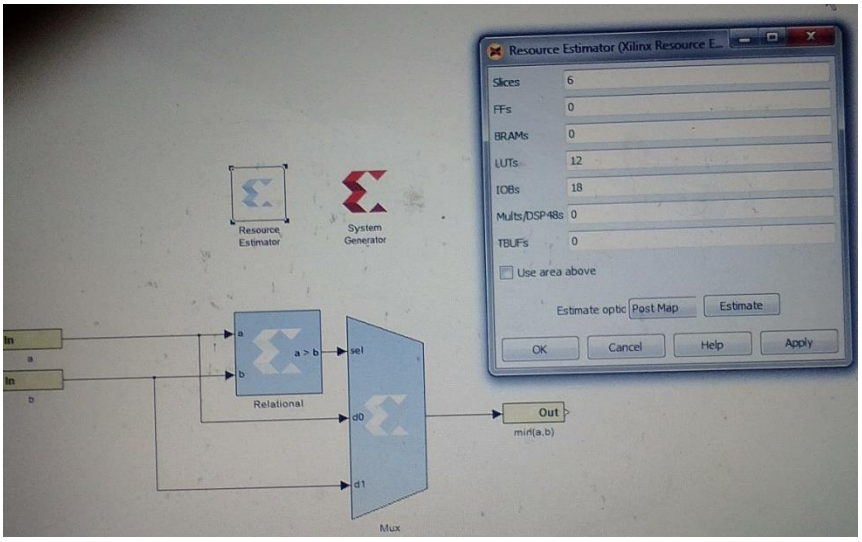


Figure 3.26 Les ressources hardwares consommées par un operateur Min à deux entées

Pour les operateurs (min/max) qu'ont plus de deux entrées on utilise des operateurs (min/max) à deux entrées pour l'implémentation de ces operateurs. Par exemple, pour implémenter un opérateur (Min) à trois entées, on utilise deux operateurs (Min) à deux entées:

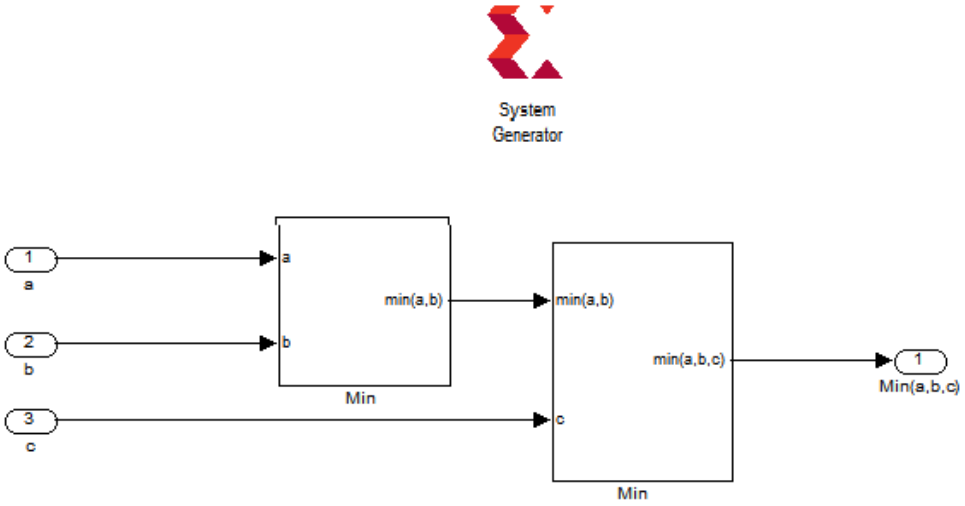


Figure 3.27 L'implémentation d'une fonction Min à 3 entées

Composition des règles: si plusieurs règles peuvent être activées simultanément et préconiser des actions avec différents degrés de validités sur la même sortie.

On suppose que les règles sont attachées par un opérateur OU.

$$\mu_B(y) = MAX\left[\mu_{B_i}(y)\right] \qquad i \in \{indices\ des\ règles\ activées\}$$

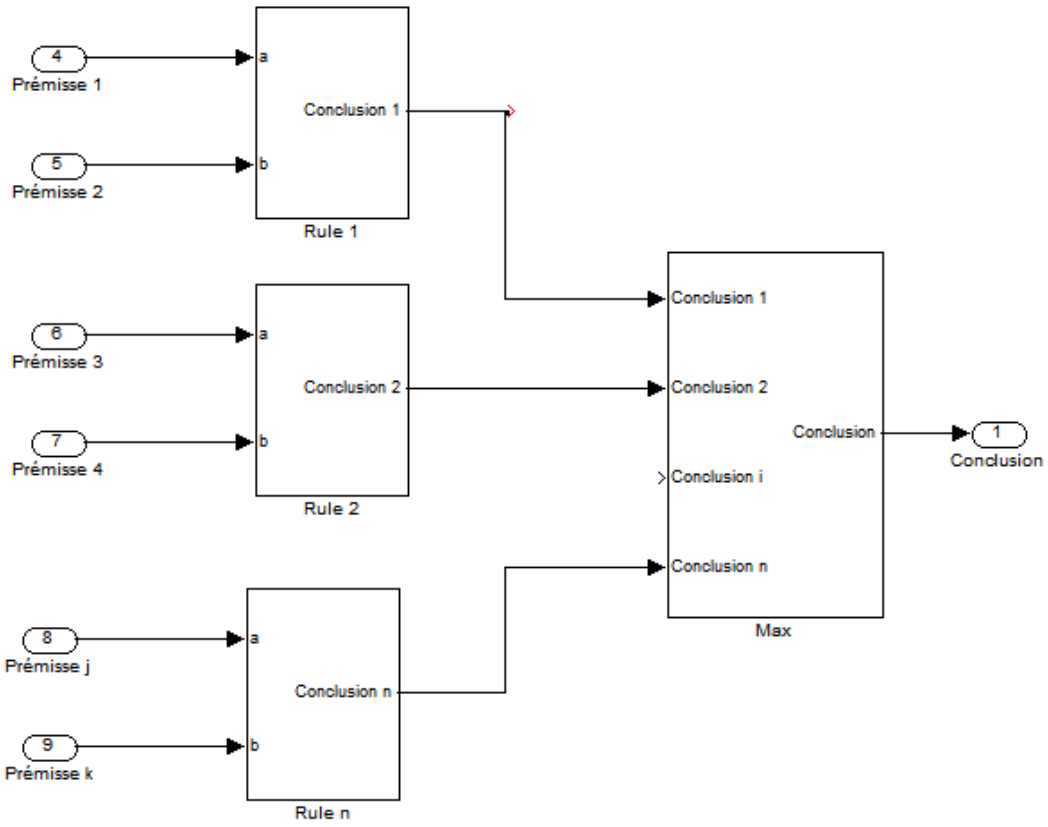


Figure 3.28 L'implémentation d'une Composition des règles pour une sortie

3.5.3 Implémentation de l'unité "Defuzzification"

L'implémentation hardware de l'unité "Defuzzification" est effectuée par un opérateur MAX comme le montre la figure 3.29. Les entrées de ce bloc sont les sorties de l'unité "Moteur d'inférence". Comme il a été indiqué dans les sections antérieurs, les Vi (i=0..7) représentent les degrés d'activation des tensions V. La sortie de l'unité "Moteur d'inférence" représentant la sortie de système d'inférence flou, correspond aux états des interrupteurs qu'on doit appliquer à l'onduleur pour donner la tension nécessaire aux bornes de la machine.

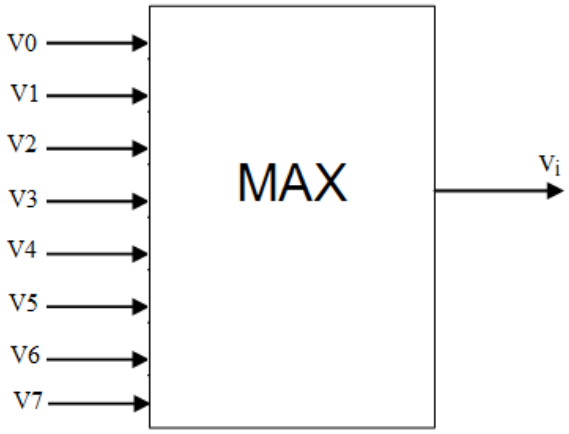


Figure 3.29 Architecture de la defuzzification

3.6 Implémentation d'une Commande DTC Neuronale sur FPGA

En ce qui concerne la commande DTC neuronale proposée, les deux comparateurs à hystérésis et la table de Takahashi (Figure 3.1) seront remplacé par un RNA. La figure suivante présente la structure de contrôle de la machine asynchrone basée sur une commande DTC Neuronale.

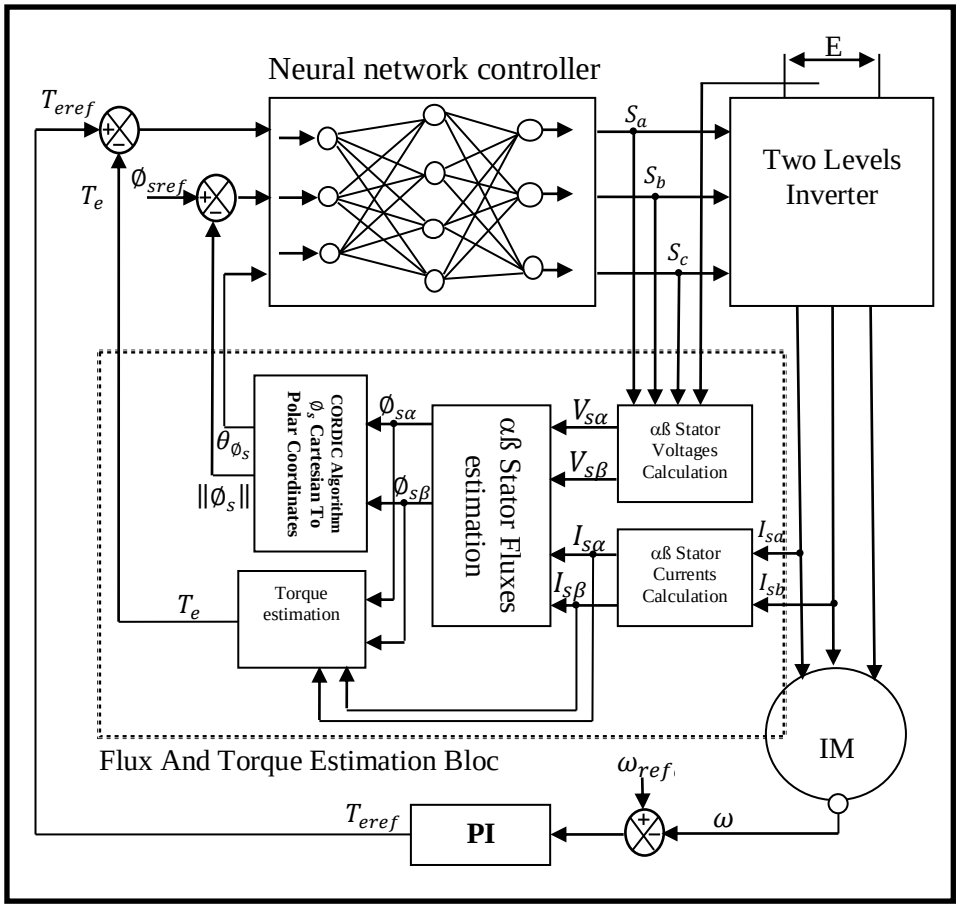


Figure 3.30 Structure de contrôle de la machine à induction basée sur DTC Neuronale

Ce travail propose de construire un sélectionneur neuronal qui va générer des séquences direct pour commander l'onduleur par un réseau de neurones qui a l'architecture suivante: une couche d'entrée avec 03 neurones, une couche cachée avec 06 neurones et une couche de sortie avec 03 neurones.

La structure de réseau de neurones artificiels RNA proposée est montrée dans la figure: Fig.3.31, les entrées du sélectionneur neuronal sont les états du flux, du couple et le secteur calculé par l'angle de flux statorique. La sortie de ce sélectionneur représente les états des commandes de l'onduleur de tension (S_a, S_b et S_c).

L'implémentation de l'architecture neuronale proposée sur FPGA consiste à implémenter tous les opérations arithmétiques et les fonctions mathématiques dont il est formé le réseau de neurones artificiels. Pratiquement il y a deux types des fonctions mathématiques dans un RNA, la fonction d'état ou d'activité calculée à partir des autres états et les poids synaptiques

par : $x = \sum_{i=1}^{n} w_i x_i$ et la fonction d'activation du neurone qui est la fonction sigmoïde dans notre étude. (Voir le paragraphe 3.1 et l'équation 1.3 en premier chapitre).

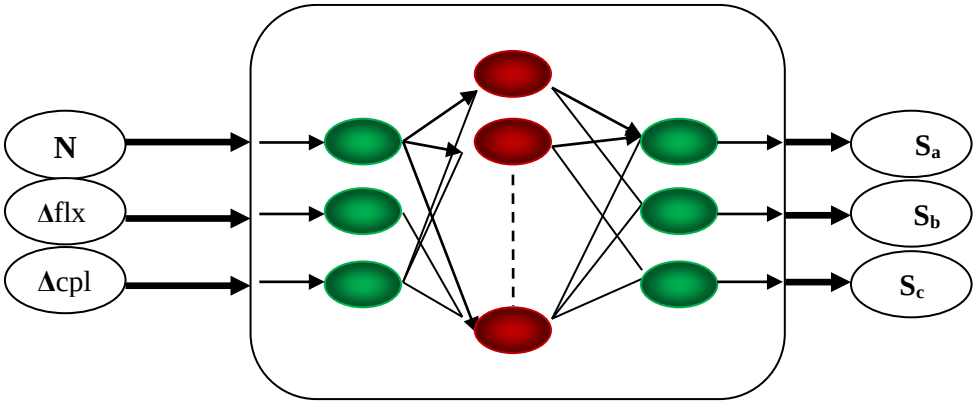


Figure 3.31 La structure de réseau de neurones artificiels RNA proposée

Puisque le nombre des neurones est fixe et les coefficients des poids synaptiques w_i sont fixés après l'apprentissage de notre RNA, les fonctions des états du RNA proposé sont simplement implémentées sur FPGA par des operateurs d'addition et de multiplication avec XSG.

L'implémentation de la fonction d'état ou d'activité d'un neurone de la couche cachée est présentée dans la figure suivante:

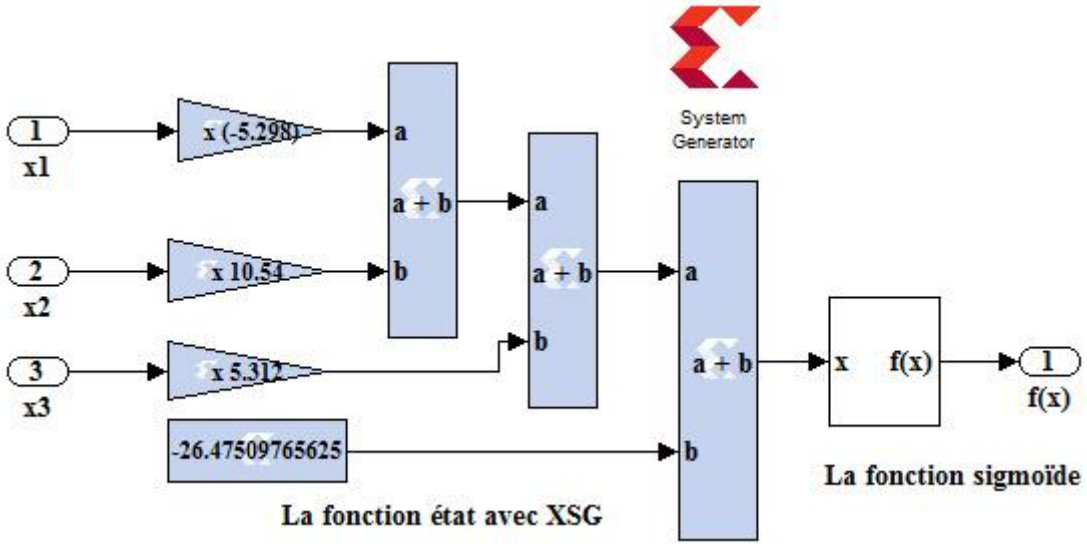


Figure 3.32 Implémentation de la fonction d'état avec XSG

Pour l'implémentation de la fonction d'activation on a utilisé l'approximation optimisée de la fonction sigmoïde présentée dans le paragraphe 4.3 du premier chapitre proposée par [06].

La figure 3.33 présente l'implémentation hardware sur FPGA de l'approximation optimisée de la fonction sigmoïde avec XSG sous Matlab/Simulink.

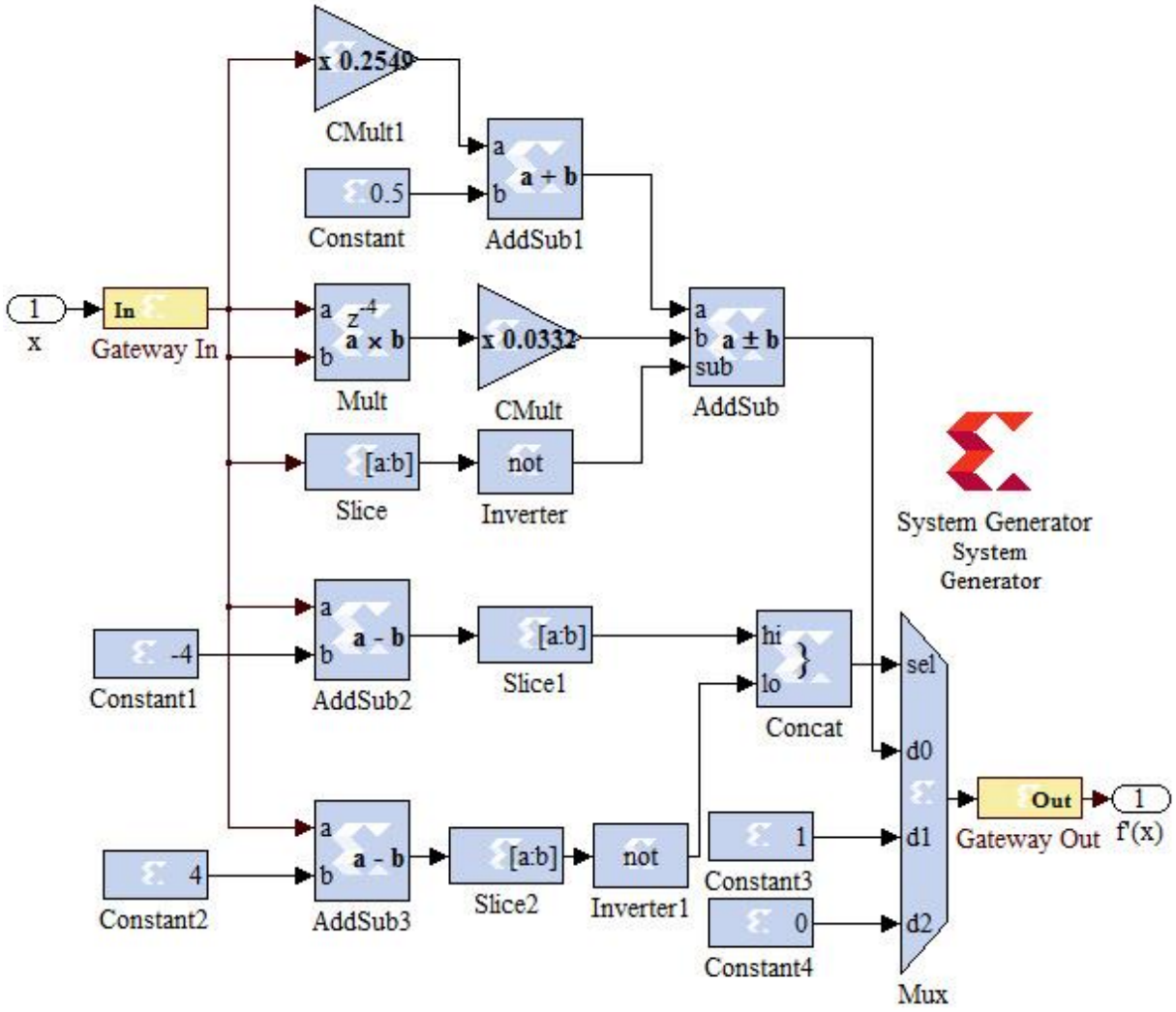


Figure 3.33 L'implémentation sur FPGA de l'approximation optimisée de la fonction sigmoïde

Les coefficients (les poids synaptiques w_i et les biais) du RNA proposé sont estimés par un processus d'apprentissage, l'algorithme de rétropropagation de l'erreur est utilisé pour l'apprentissage de notre RNA.

La base de données d'apprentissage est constituée des entrées (N, Δflx et Δcpl) appliquées au RNA et les sorties désirées (S_a, S_b et S_c).

Les coefficients sont estimés de manière à minimiser une erreur d'approximation définie à partir de l'écart entre les sorties désirées et les sorties réelles à l'aide de l'algorithme d'apprentissage utilisé.

Le déroulement d'apprentissage est fait selon les critères suivants:

- L'apprentissage est supervisé.
- Les fonctions d'activation utilisées pour la couche cachée et la couche de sortie est la fonction sigmoïde.
- L'algorithme d'apprentissage utilisé est l'algorithme de rétropropagation de l'erreur.

3.7 Conclusion

Dans ce chapitre on a présenté en premier lieu le principe de la commande directe de couple pour l'entrainement d'un ensemble machine asynchrone-onduleur de tension deux niveaux. En deuxième lieu, on a exposé l'implémentation de la DTC sur un circuit logique configurable de type FPGA avec les langages de description hardware VHDL et Xilinx System Generator XSG. L'architecture proposée est basée sur une implémentation séparée entre la partie contrôle et la partie estimation de la commande DTC pour une éventuelle réutilisation de l'estimateur développé.

D'autre part, nous avons effectué l'intégration des techniques intelligentes dans la commande DTC afin de réduire les inconvénients liés à l'utilisation des régulateurs à hystérésis. Dans cette partie, on a présenté une architecture hardware optimisée d'une commande DTC floue, ensuite on a présenté aussi une implémentation hardware sur FPGA d'une commande DTC basée sur un réseau de neurones artificiels.

Dans le chapitre suivant les résultats de l'implémentation des architectures proposées dans ce chapitre sont détaillées et discutées.

Chapitre 4:
Résultats, Simulations et Validations des architectures d'implémentation sur FPGA

4.1 Introduction

Le but de ce chapitre est la validation par Co-Simulation hardware des architectures de la commande DTC basée sur des techniques intelligentes proposées et développées dans les chapitres précédentes.

A cette effet, nous avons choisi l'implémentation sur une cible hardware reconfigurable de type FPGA les architectures des commandes DTC proposées pour réduire le temps d'exécution de ces méthodes. Il est à noter que les FPGA et leurs conceptions en logique câblée peuvent fournir des performances très élevées en quantité des traitements arithmétiques et temps d'exécution des tâches.

Dans un premier temps, nous présenterons la simulation mixte (Simulink/XSG) des architectures hardwares proposées afin de tester le fonctionnement des architectures de commande proposées en mode discontinue et au format binaire. Nous détaillerons ensuite les résultats de la synthèse et la validation par Co-Simulation hardware des algorithmes de commande proposées. Une fois l'implémentation hardware sur FPGA est validée, une étude comparative entre les algorithmes proposés sera exposée.

4.2 Simulation mixte des commandes d'une machine asynchrone

Cette partie consiste à l'intégration d'une technique intelligente dans la structure de la commande DTC. Pendant cette phase et d'une façon similaire à la simulation avec MATLAB, on va simuler avec MATLAB/Simulink et Xilinx système générateur les architectures hardware de la commande DTC conventionnelle et celle de la commande DTC à base des techniques intelligentes.

On applique la commande DTC à une machine asynchrone dont les spécifications sont données par le tableau suivant :

U_n	**220 V**
I_n	**2.35 A**
P_n	**1.08 kW**
N	**1430 Tr/mn**
f	**50 Hz**
R_S	**10 Ω**
R_r	**6.3 Ω**
L_S	**0.4642 H**
L_r	**0.4612 H**
L_m	**0.4212 H**
J	**0.02 kg.m^2**
P	**2**

Tableau 4.1 Paramètres de la machine asynchrone

Le schéma de la commande DTC conventionnelle est présenté dans la **Figure 3.1.** pour les différentes commandes DTC à base des techniques intelligentes exposées dans la figure 4.1, on va remplacer les comparateurs à hystérésis et la Table de commutation de la commande

DTC par un algorithme intelligent (un système d'inférence flou ou un RNA) qu'on a réalisé et tester. Pour vérifier ces méthodes de commande, on a simulé ces architectures sur MATLAB/Simulink avec Xilinx système générateur. Les résultats sont présentées dans les figures 4.2 et 4.3.

Les coefficients du régulateur proportionnel intégral PI de la vitesse sont K_p=1 et K_i=8. Les bandes d'hystérésis des comparateurs du flux et du couple sont fixés à ±0.05Wb et ±0.5 Nm respectivement.

On observe qu'il y ait une amélioration au niveau des résultats du couple électromagnétique donnée par la commande DTC floue par rapport aux résultats obtenus par la commande DTC conventionnelle et DTC neuronal avec une diminution des ondulations importantes.

Par contre au niveau du flux statorique une amélioration a été obtenue observée dans les résultats de la commande DTC neuronal par rapport à celles obtenues par la commande DTC conventionnelle et DTC floue avec une diminution remarquable des ondulations.

Dans tous les cas le remplacement des comparateurs à hystérésis par des techniques intelligentes apporte des améliorations significatives dans les ondulations du flux statorique et du couple électromagnétique.

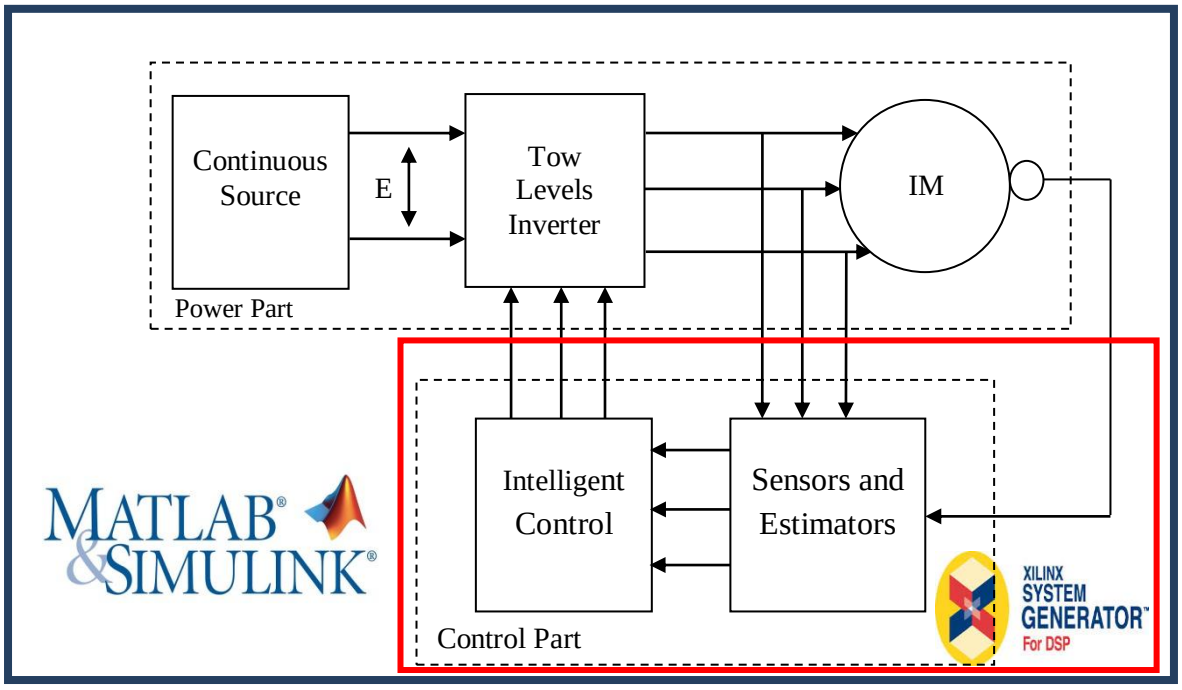
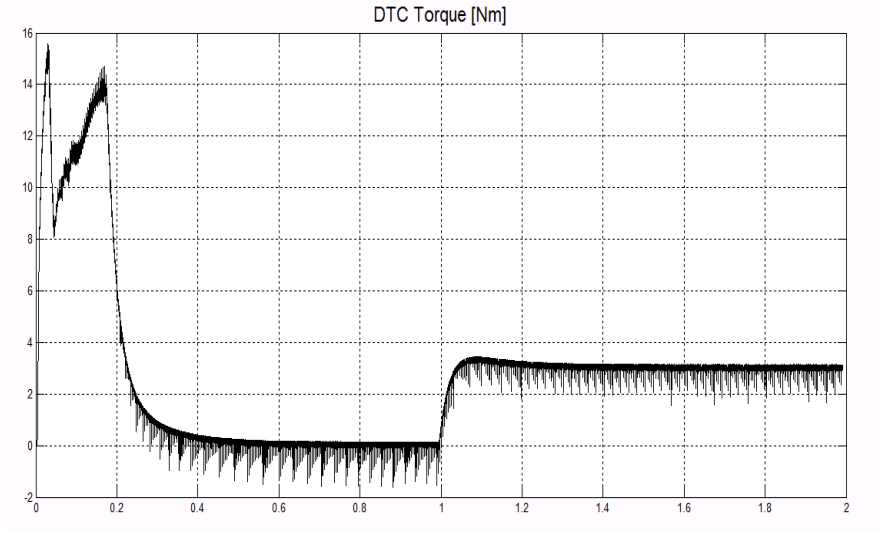
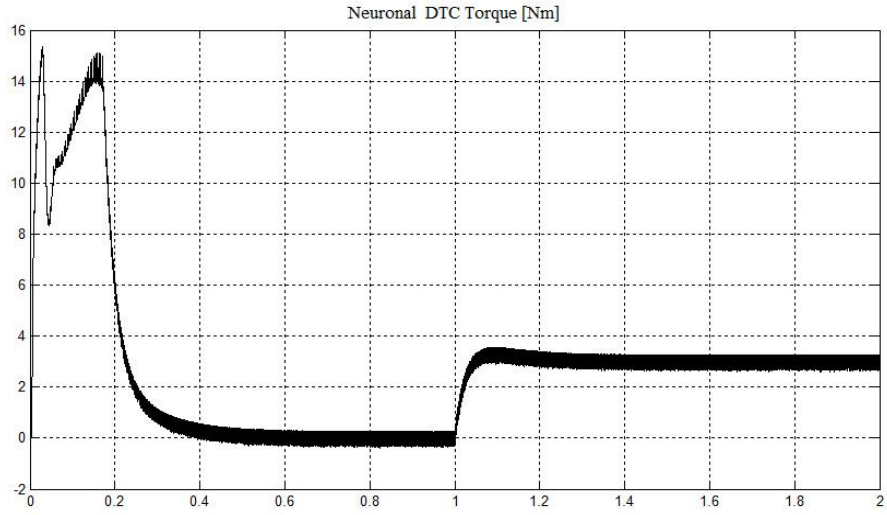


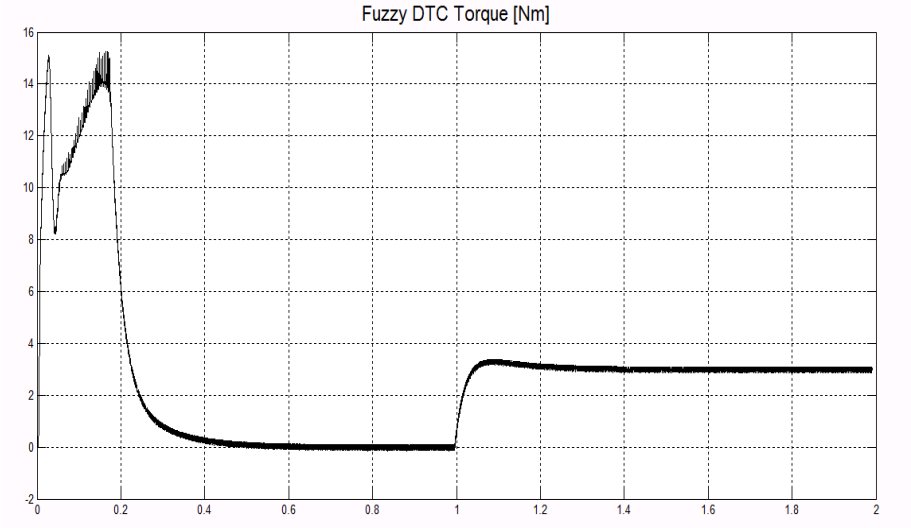
Figure 4.1 Simulation mixte de la commande DTC à base des techniques intelligentes



(a)

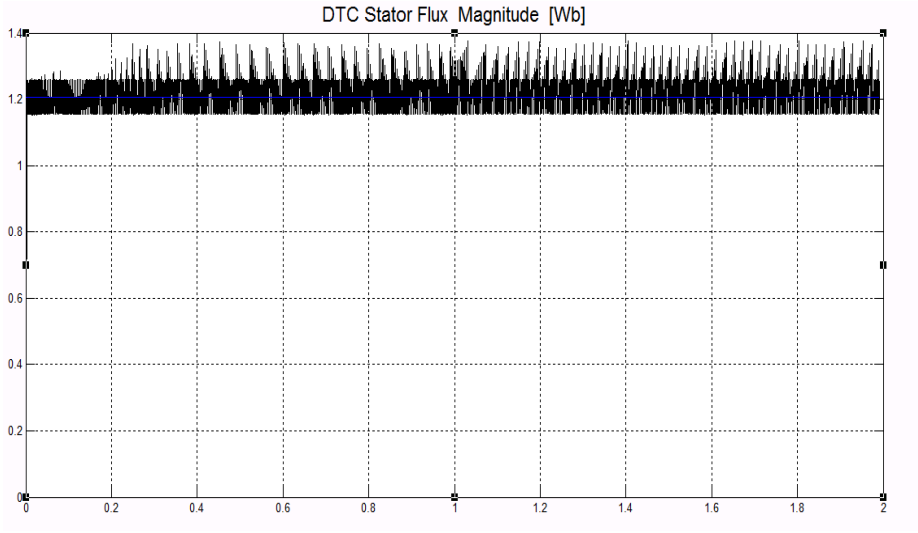


(b)

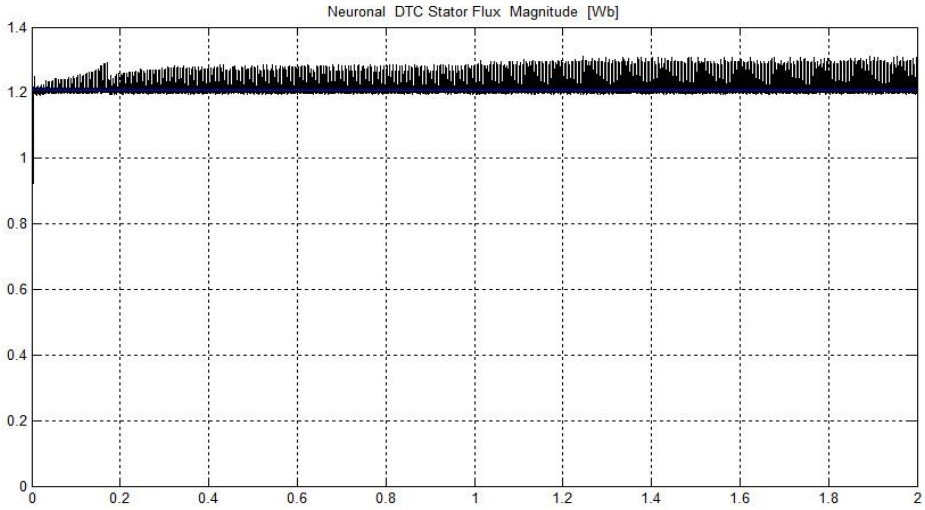


(c)

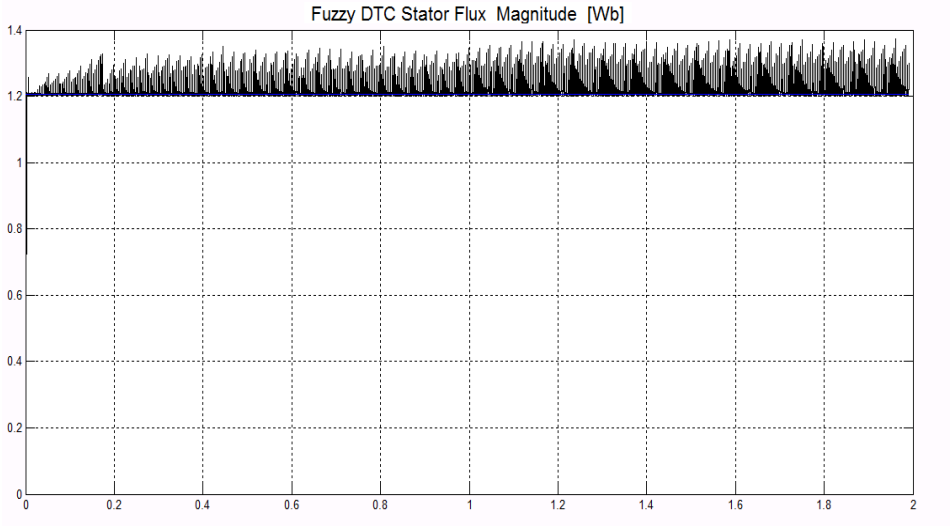
Figure 4.2 Couple électromagnétique par XSG simulateur pour DTC conventionnelle, Neuronal DTC et Fuzzy DTC respectivement



(a)



(b)



(c)

Figure 4.3 Flux statorique obtenu par XSG simulateur pour DTC conventionnelle, Neuronal DTC et Fuzzy DTC respectivement

4.3 Synthèse et Ressources FPGA utilisées

Les tableaux 4.2, 4.3, 4.4 et 4.5 présentes les occupations des Ressources FPGA données par l'implémentation des architectures proposées pour l'estimateur de flux et de couple, la commande DTC conventionnelle, la commande DTC neuronale et la commande DTC floue sur FPGA VIRTEX 4 données par l'outil de synthèse pour les architectures proposées et présentées dans le chapitre précédent.

La carte cible: ML402 Virtex-4 xc4vsx35-10ff668			
Utilisation logique	**Utilisé**	**Disponible**	**Utilisation**
Nombre des tranches de bascules	1,471	30,720	4%
Nombre de tranches occupées	3,417	15,360	22%
Nombre total des LUTs 4 entrée	5,233	30,720	17%
Nombre d' IOBs cautionnés	58	448	12%
Nombre de FIFO16/RAMB16s	12	192	6%
Nombre de BUFG/BUFGCTRLs	8	32	25%

Tableau 4.2 Occupation des Ressources FPGA par le bloc estimateur d'une commande DTC

La carte cible: ML402 Virtex-4 xc4vsx35-10ff668			
Utilisation logique	**Utilisé**	**Disponible**	**Utilisation**
Nombre des tranches de bascules	1,438	30,720	4%
Nombre de tranches occupées	1,720	15,360	11%
Nombre total des LUTs 4 entrée	2,178	30,720	7%
Nombre d' IOBs cautionnés	58	448	12%
Nombre de FIFO16/RAMB16s	12	192	6%
Nombre de BUFG/BUFGCTRLs	7	32	21%

Tableau 4.3 Occupation des Ressources FPGA par la commande DTC conventionnelle

La carte cible: ML402 Virtex-4 xc4vsx35-10ff668			
Utilisation logique	**Utilisé**	**Disponible**	**Utilisation**
Nombre des tranches de bascules	8,875	30,720	28%
Nombre de tranches occupées	8,155	15,360	53%
Nombre total des LUTs 4 entrée	13,361	30,720	43%
Nombre d' IOBs cautionnés	58	448	12%
Nombre de FIFO16/RAMB16s	12	192	6%
Nombre de BUFG/BUFGCTRLs	08	32	25%

Tableau 4.4 Occupation des Ressources FPGA par la commande DTC neuronale

La carte cible: ML402 Virtex-4 xc4vsx35-10ff668			
Utilisation logique	**Utilisé**	**Disponible**	**Utilisation**
Nombre des tranches de bascules	1,365	30,720	04%
Nombre de tranches occupées	1,620	15,360	10%
Nombre total des LUTs 4 entrée	2,453	30,720	07%
Nombre d' IOBs cautionnés	58	448	12%
Nombre de FIFO16/RAMB16s	12	192	06%
Nombre de BUFG/BUFGCTRLs	04	32	12%

Tableau 4.5 Occupation des Ressources FPGA par la commande DTC floue

On remarque bien que l'architecture proposée pour la commande DTC floue optimise réduite l'utilisation des ressources hardware de la carte FPGA (10% des Slices et 07% des LUTs), de plus cette architecture réduit considérablement les composants logiques utilisés par rapport à l'architecture proposée pour la commande DTC conventionnelle, l'architecture proposée pour la commande DTC neuronale et les architectures présentées dans [30] et [31].

L'utilisation des operateurs simplement implémentables en hardware telles que les fonctions MIN et MAX qui se basent uniquement sur des comparateurs et des multiplexeurs, et la méthode de CORDIC qui se basent sur des operateurs élémentaires tels que l'addition, la

soustraction et le décalage des chiffres binaires, nous permettent de réduire considérablement les ressources utilisées sur le FPGA par le bloc d'estimation et l'algorithme de commande DTC floue. Par contre, dans l'algorithme de commande DTC neuronale et malgré l'utilisation d'une fonction d'activation sigmoïde optimisée les ressources utilisées sont très élevées et ceci est principalement dû à la complexité des RNAs et de leur fonction d'activité qui utilise fréquemment la multiplication.

La fréquence de l'horloge maximale donnée par l'outil de synthèse est égale à 231.64MHz, qui présente une période de fonctionnement minimale de 4.317ns. Par contre dans l'article [30] La fréquence de l'horloge est égale à 54MHz. Dans l'article [32] la période minimale est égale à 50 ns en utilisant dSPACE (digital signal processing and control engineering). On voit que le temps d'exécution est trop important par rapport au FPGA en raison des traitements séquentiels du dSPACE.

Le tableau suivant présente les ressources matérielles les plus importantes consommées dans les contrôleurs proposés dans ce travail, compte tenu de la partie estimation par rapport aux travaux antérieurs dans le même axe de recherche:

Utilisation logique	Références					
	[30]	[46]	[47]	DTC conventionnelle	DTC neuronale	DTC floue
Famille de la carte FPGA	Xilinx Virtex-5	Altera DE-115	Altera CYCLONE II	Xilinx Virtex-4	Xilinx Virtex-4	Xilinx Virtex-4
Multiplicateur intégré 9 bits	/	80	57	/	/	/
Éléments logiques totaux	2.502	6.931	3.256	**2.909**	**10.346**	**2.836**
Fonctions combinatoire totales	21.758	6.491	2.549	**7.411**	**18.594**	**7.686**

Tableau 4.6 Comparaison des ressources utilisées par différents algorithmes

Le tableau montre que les architectures proposées pour la DTC conventionnelle et la DTC floue minimisent l'utilisation des ressources par rapport aux implémentations précédentes de la commande DTC. La commande DTC neuronale a besoin d'une plus grande quantité de ressources en raison de sa structure complexe et ces fonctions des activités basées sur l'operateur de multiplication binaire.

4.4 Validation des architectures proposées par Co-Simulation hardware

Après les étapes de simulation et de synthèse, les architectures hardwares proposées ont été validées par un hardware Co-Simulation sur le périphérique cible ML402 dotée d'un circuit FPGA VIRTEX4.

Cette phase est consacrée à l'implémentation hardware sur la carte FPGA des architectures des commandes développées et simulées dans les sections précédentes. Elle est particulièrement destinée à la validation et à la vérification de l'implémentation hardware des architectures des commandes sur une carte FPGA cible et dans un processus de " Co-Simulation hardware".

La figure suivante montre le principe de validation des architectures proposées par hardware Co-Simulation:

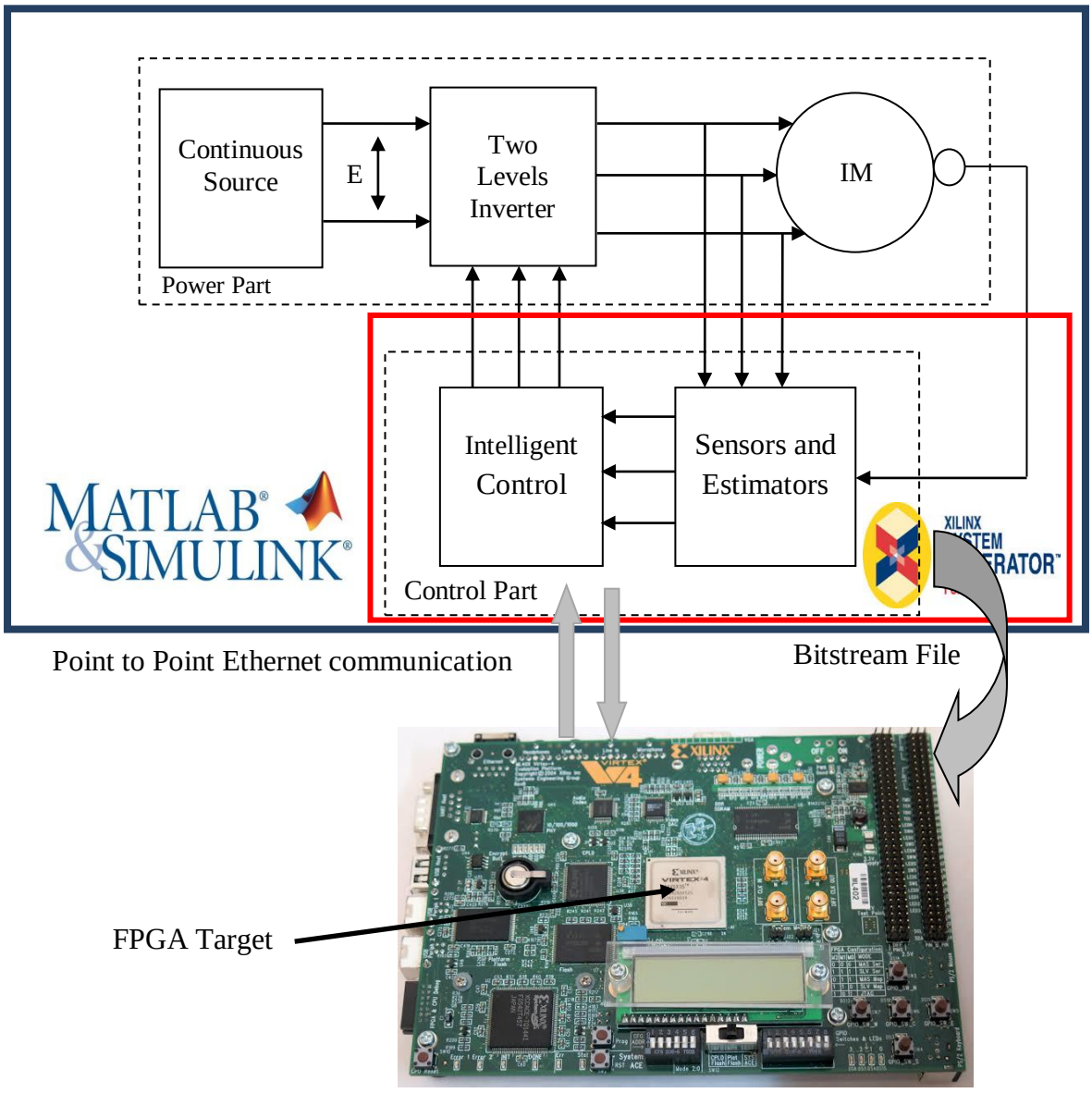


Figure 4.4 Co-Simulation hardware de la commande DTC à base des techniques intelligentes

Une fois la simulation et l'analyse temporelle terminées, la procédure de la Co-Simulation hardware dans XSG crée un fichier bitstream à partir du prototype hardware et un bloc Ethernet point à point pour la procédure **Hardware-In-the-Loop**. Le bloc créé remplace l'architecture hardware qui a été construit avant.

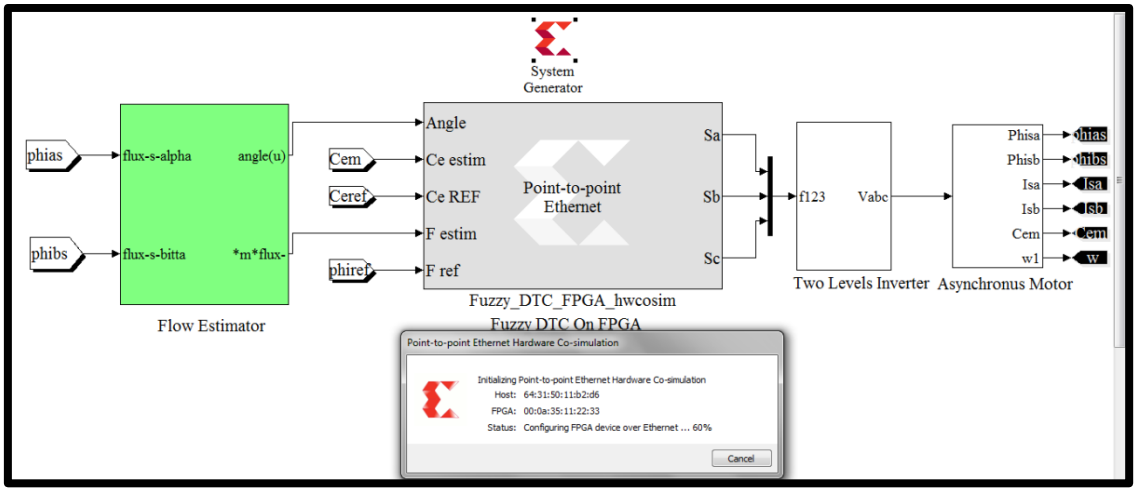


Figure 4.5 Le bloc Ethernet point à point dans une procédure Hardware-In-the-Loop

Les blocs Ethernet point à point sont reliés à l'onduleur et à la machine asynchrone pour faire fonctionner une procédure HIL. Dans cette situation les modèles de la MAS et de l'onduleur de tension sont simulés dans un environnement Matlab/Simulink, et les architectures XSG sont réalisées dans le dispositif FPGA ML402.

La validation par Hardware-In-the-Loop est exécutée en connectant la carte cible au PC via un câble Ethernet.

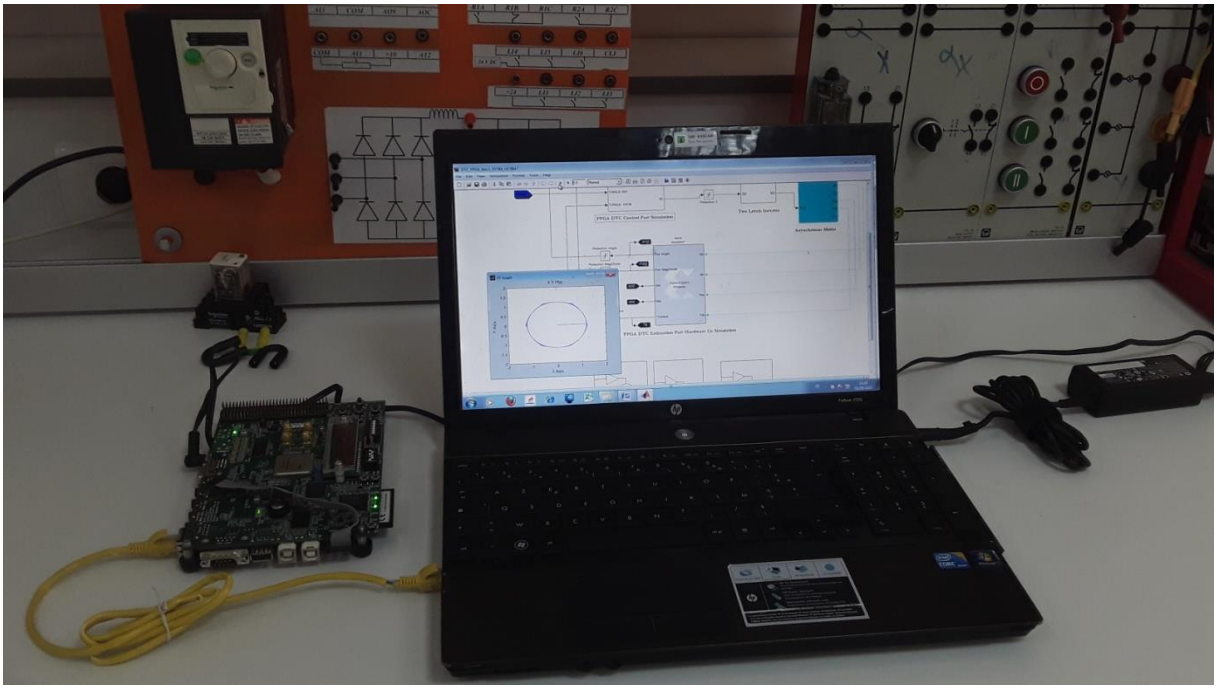


Figure 4.6 Banc d'essai pour validation par Hardware Co-Simulation

4.4.1 Validation de l'architecture proposée pour l'estimateur de flux et de couple

L'architecture hardware de l'estimateur de flux et de couple est validée par un processus de Co-Simulation hardware dans l'environnement Matlab/Simulink avec l'outil XSG et la carte FPGA ML402.

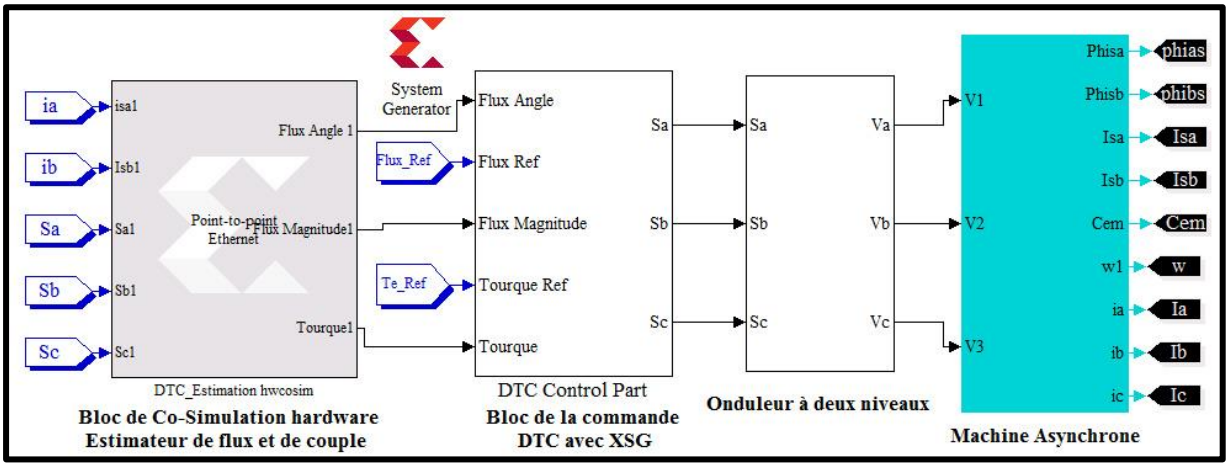


Figure 4.7 Bloc de Co-Simulation hardware pour L'architecture de l'estimateur de flux et de couple

L'estimation du flux statorique en plan α/β est présentée par la figure suivante:

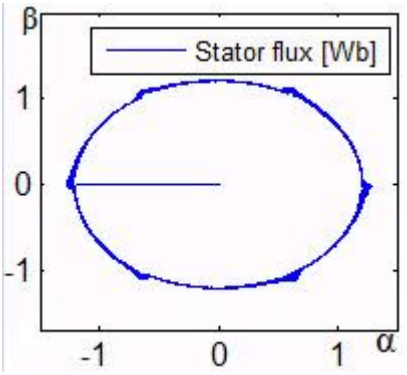


Figure 4.8 L'estimation du flux statorique en plan α/β

Dans la figure 4.9, le couple, l'angle et l'amplitude de flux statorique calculés sont présentés pour la phase de démarrage d'une machine asynchrone [45].

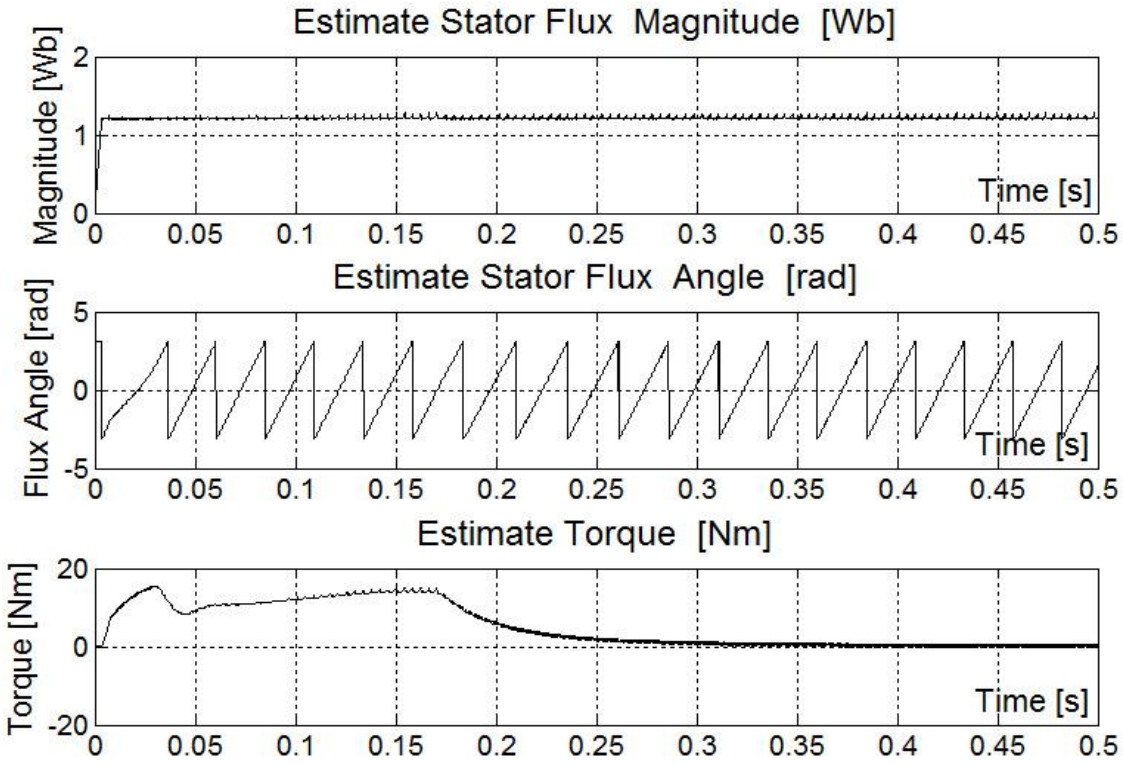


Figure 4.9 L'amplitude de flux statorique, l'angle et le couple estimés pour la phase de démarrage d'une machine asynchrone

Le tableau 4.7 montre l'erreur quadratique moyenne et l'erreur maximale entre les valeurs théoriques et les valeurs calculées par notre estimateur de couple électromagnétique, angle et amplitude de flux statorique [45]:

	Angle de flux statorique [rad]	Amplitude de flux statorique [Wb]	Couple électromagnétique [Nm]
Erreur RMS	0.01	0.0002	0.0005
Erreur MAX	0.03	0.02	0.04

Tableau 4.7 Les erreurs d'estimation entre les valeurs théoriques et les valeurs estimées

Le travail présenté dans cette section propose un estimateur universel de flux et de couple qu'on peut réutiliser pour toutes les commandes de la machine asynchrone nécessite ces quantités estimées sans modif0.ications significatives.

L'estimateur proposé est développé avec une erreur d'estimation négligeable (erreur RMS de l'estimation de flux statorique égale à 0.0002 [Wb] et erreur RMS de l'estimation de couple électromagnétique égale à 0.0005 [Nm]). En plus, cet estimateur est implémentée sur la carte FPGA Virtex-4 avec minimum des ressources (moins 5% des slices registers, 5% des DSP48Es et environ 17% des LUTS) [45].

4.4.2 Validation de l'architecture proposée pour la DTC conventionnelle

Après l'étape de validation de notre estimateur, il a été utilisé pour compléter l'architecture DTC conventionnelle proposée dans un processus de Co-Simulation hardware.

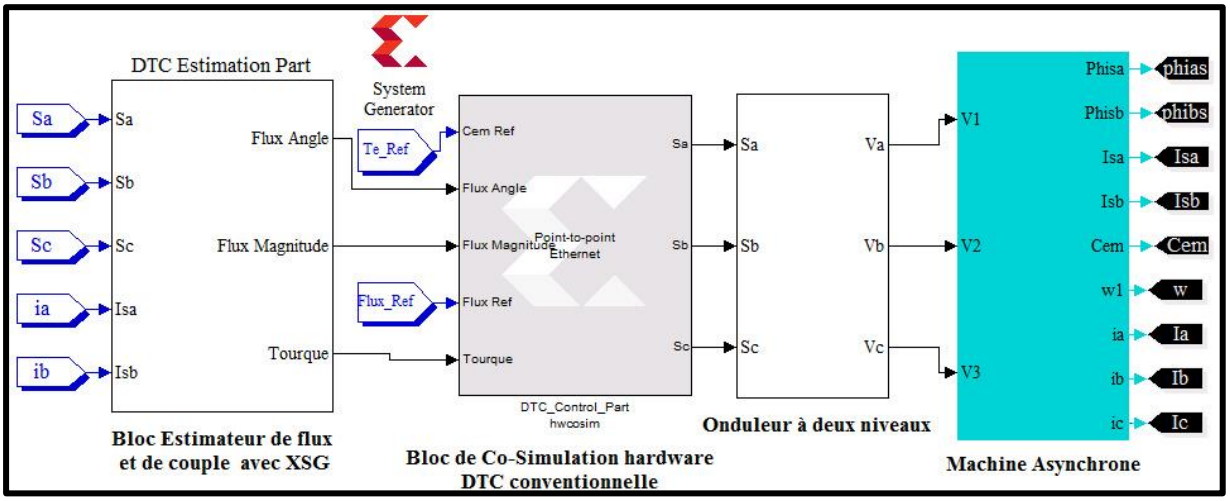


Figure 4.10 Bloc de Co-Simulation hardware pour L'architecture de DTC conventionnelle

Les formes d'ondes de réponse de la vitesse et du couple de la machine asynchrone par rapport différentes références de la vitesse et du couple résistant sont indiquées dans la figure 4.11.

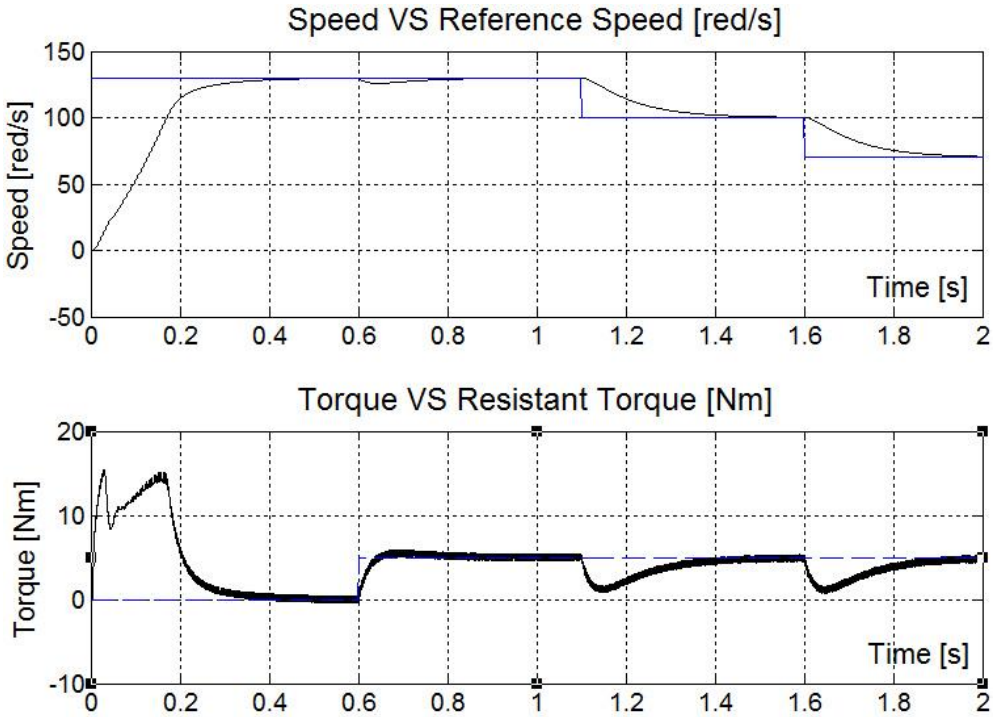


Figure 4.11 Courbe de la vitesse et du couple d'une machine asynchrone pour la commande DTC conventionnelle

L'évolution de la vitesse, du courant et du flux de phase sont présentés sur la figure 4.12.

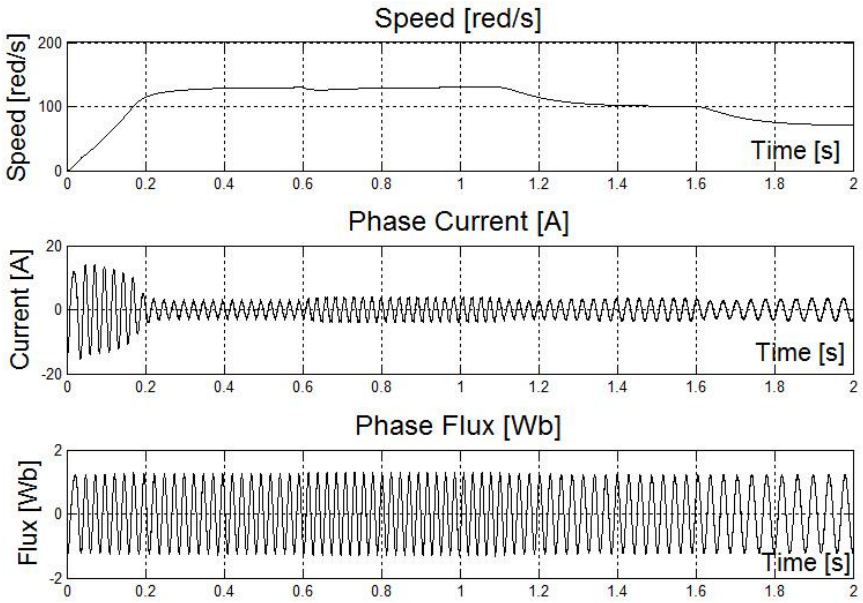


Figure 4.12 Courbe de la vitesse, le courant et le couple de phase d'une machine asynchrone pour la commande DTC conventionnelle

Les résultats obtenus lors de la phase de Co-Simulation hardware ont prouvé que l'implémentation était réussie. Néanmoins, il y a l'apparence de certaines fluctuations au niveau des courbes de paramètres tels que le courant, le couple et le flux. Cela est dû à l'utilisation des comparateurs à hystérésis au niveau de la structure de la commande DTC conventionnelle et à la tentative de choisir la taille optimale des données afin d'obtenir un calcul précis et des ressources internes minimales du circuit FPGA. De plus, la fluctuation du couple et du flux estimés n'a pas d'une influence importante sur les réponses dynamiques du moteur ; même la courbe de la vitesse n'a pas été affectée [45].

La figure suivante montre l'analyse spectrale de la courbe de vitesse obtenue par "Powergui FFT AnalysisTool" de Matlab.

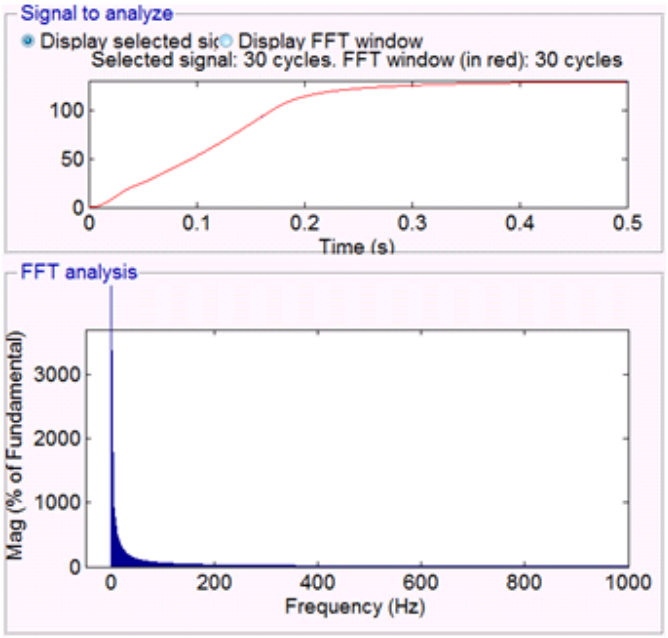


Figure 4.13 L'analyse spectrale de la courbe de vitesse

L'analyse spectrale par FFT de la courbe de vitesse montre que l'ondulation du couple et du flux générée par les erreurs d'estimation n'influence pas la réponse dynamique de la vitesse.

4.4.3 Validation de l'architecture proposée pour la DTC neuronale

L'estimateur de flux et de couple développé précédemment et valider dans la section 4.4.1 a été utilisé pour valider l'architecture DTC neuronale proposée par Co-Simulation hardware.

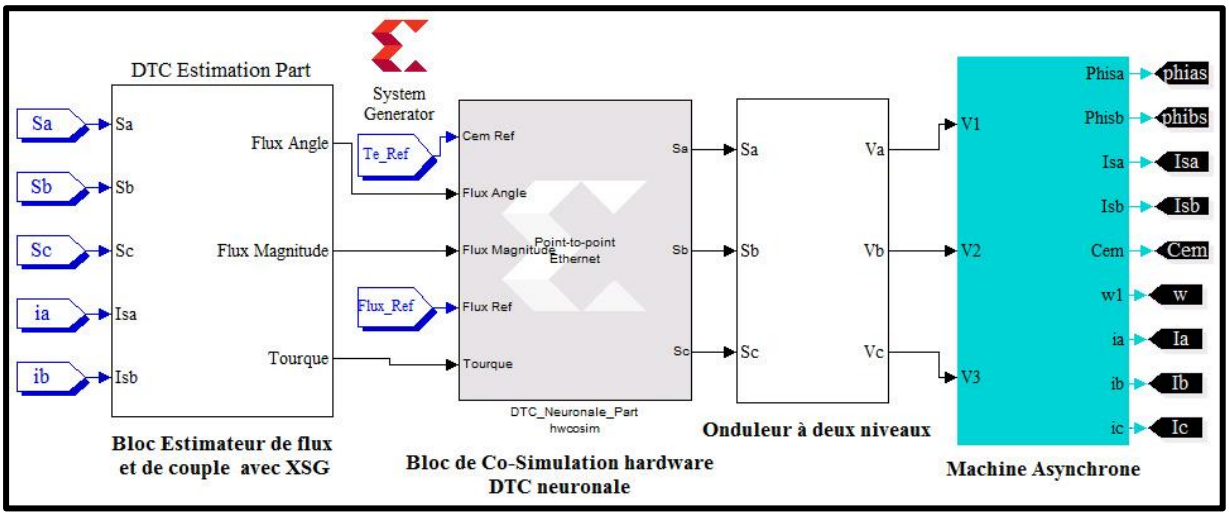


Figure 4.14 Bloc de Co-Simulation hardware pour L'architecture de DTC neuronale

Tout d'abord on a testé et valider l'architecture hardware de la fonction sigmoïde optimisée présentée dans la section **3.6** et proposée dans [06]. On a aussi utilisé le format des données réelles binaires en virgule fixe. Le format binaire la plus adéquate et que représente une compromis entre la précision d'approximation et le coût en terme d'occupation des ressources hardware est la formate (27,10) (27 bits avec 10 bits virgule binaires).

La figure suivante présente une Comparaison entre la fonction sigmoïde et la fonction sigmoïde optimisée de Xilinx (27,10) [06].

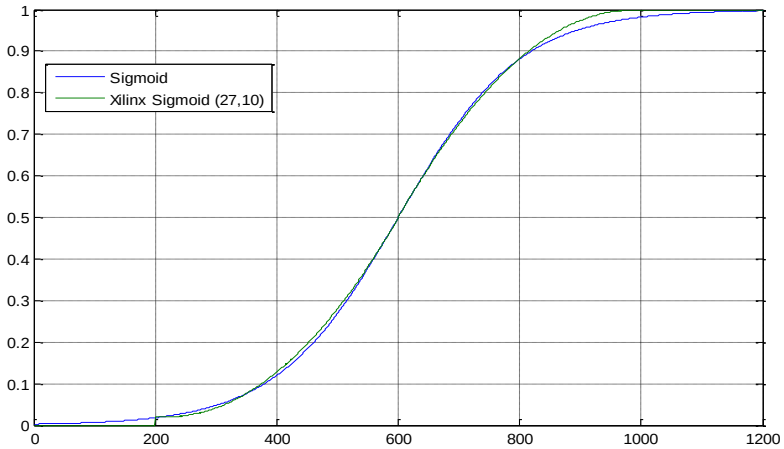


Figure 4.15 Comparaison entre la fonction sigmoïde et la fonction sigmoïde optimise de Xilinx (27,10).

L'erreur d'approximation entre la fonction sigmoïde et la fonction sigmoïde optimisée de Xilinx (27,10) est présentée dans la figure suivante:

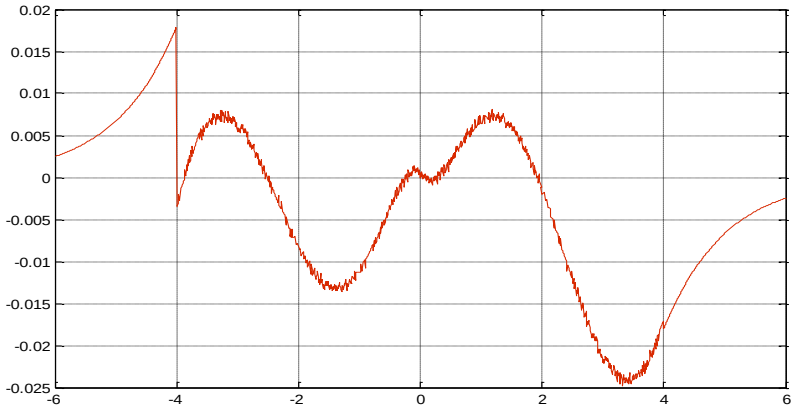


Figure 4.16 Erreur d'approximation entre la fonction sigmoïde et la fonction sigmoïde optimisée de Xilinx (27,10).

Les formes d'ondes de réponse de la vitesse, du flux statorique et du couple électromagnétique d'une machine asynchrone par rapport à la vitesse, le couple et le flux de références sont indiquées dans la figure 4.17.

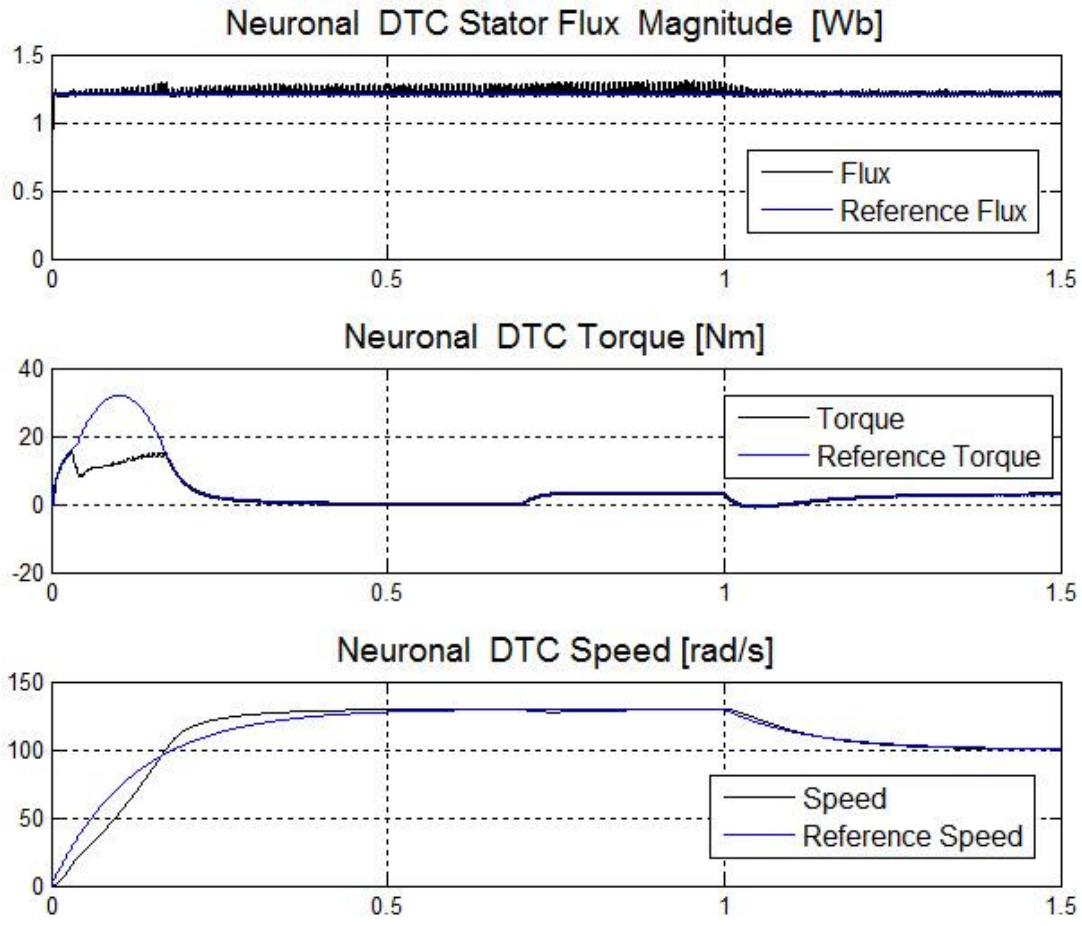


Figure 4.17 Réponse de la vitesse, du flux statorique et du couple électromagnétique pour DTC neuronale

4.4.4 Validation de l'architecture proposée pour la DTC floue

La figure suivante présente le schéma bloc de la validation de l'architecture DTC floue proposée par Co-Simulation hardware basée sur notre estimateur de flux et de couple.

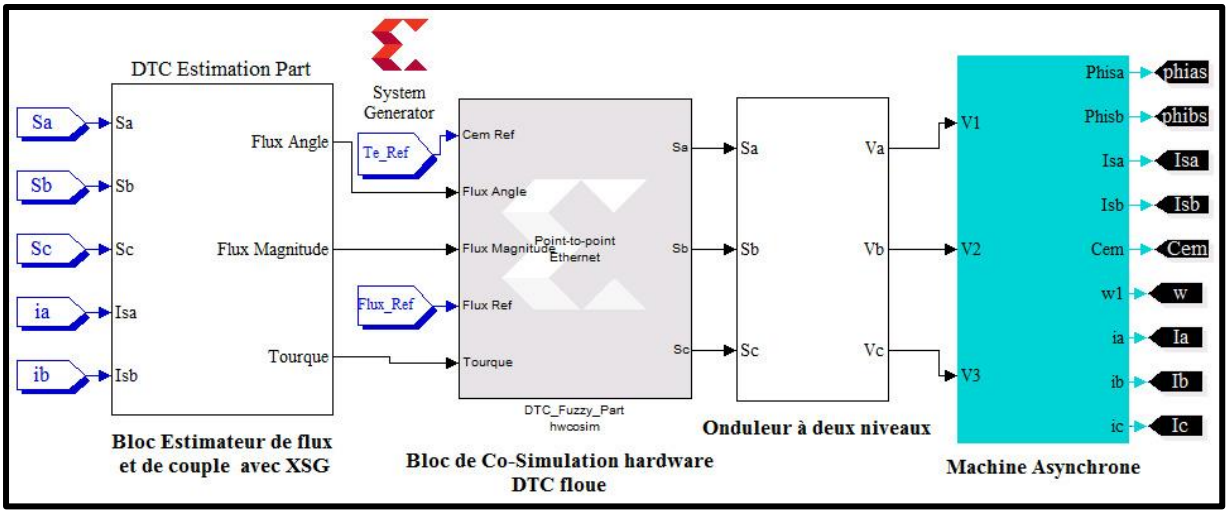


Figure 4.18 Bloc de Co-Simulation hardware pour L'architecture de DTC floue

Les formes d'ondes de réponse de la vitesse, du flux statorique et du couple électromagnétique d'une machine asynchrone versus leurs grandeurs de références sont présentées dans la figure suivante:

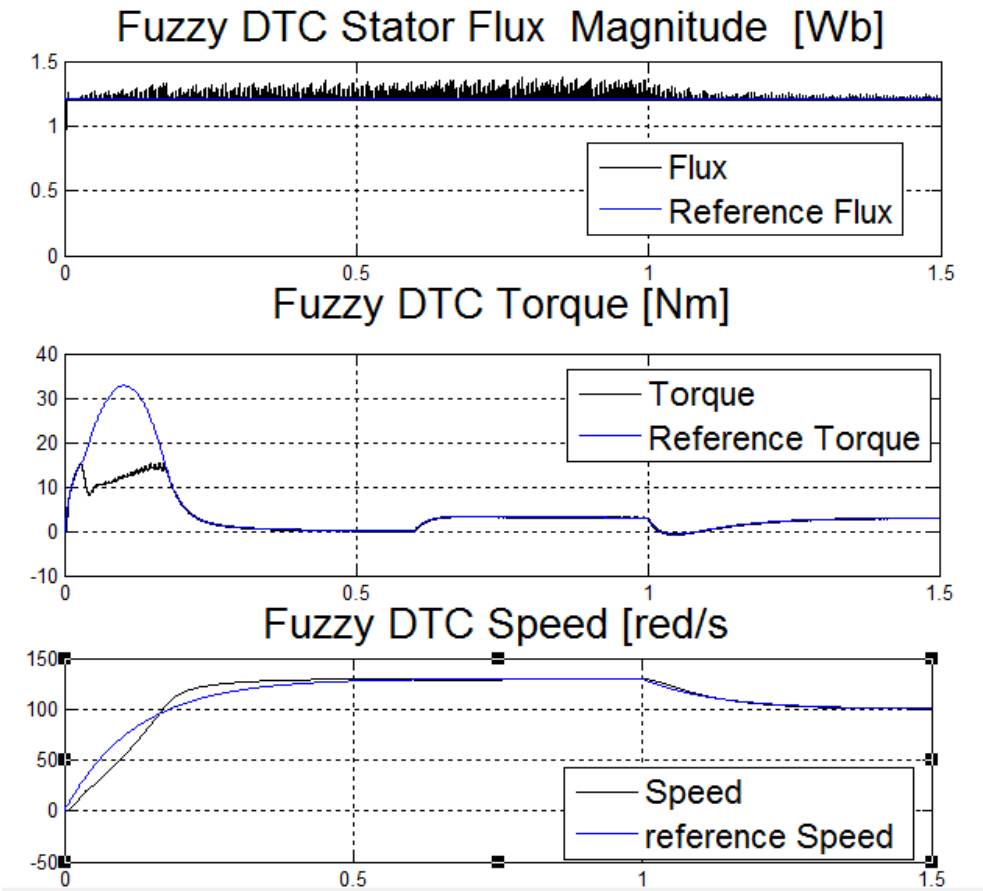


Figure 4.19 Réponse de la vitesse, du flux statorique et du couple électromagnétique pour DTC floue

4.5 Étude comparative entre les différentes architectures proposées

L'analyse spectrale par "Powergui FFT AnalysisTool" de Matlab du couple électromagnétique obtenu par DTC conventionnelle, DTC neuronale et DTC floue (Figures 4.20, 4.21 et 4.22) en régime permanant montre l'existence des harmoniques le long du spectre du couple électromagnétique obtenu par DTC conventionnelle contrairement au couple électromagnétique obtenu par DTC floue et DTC neuronale.

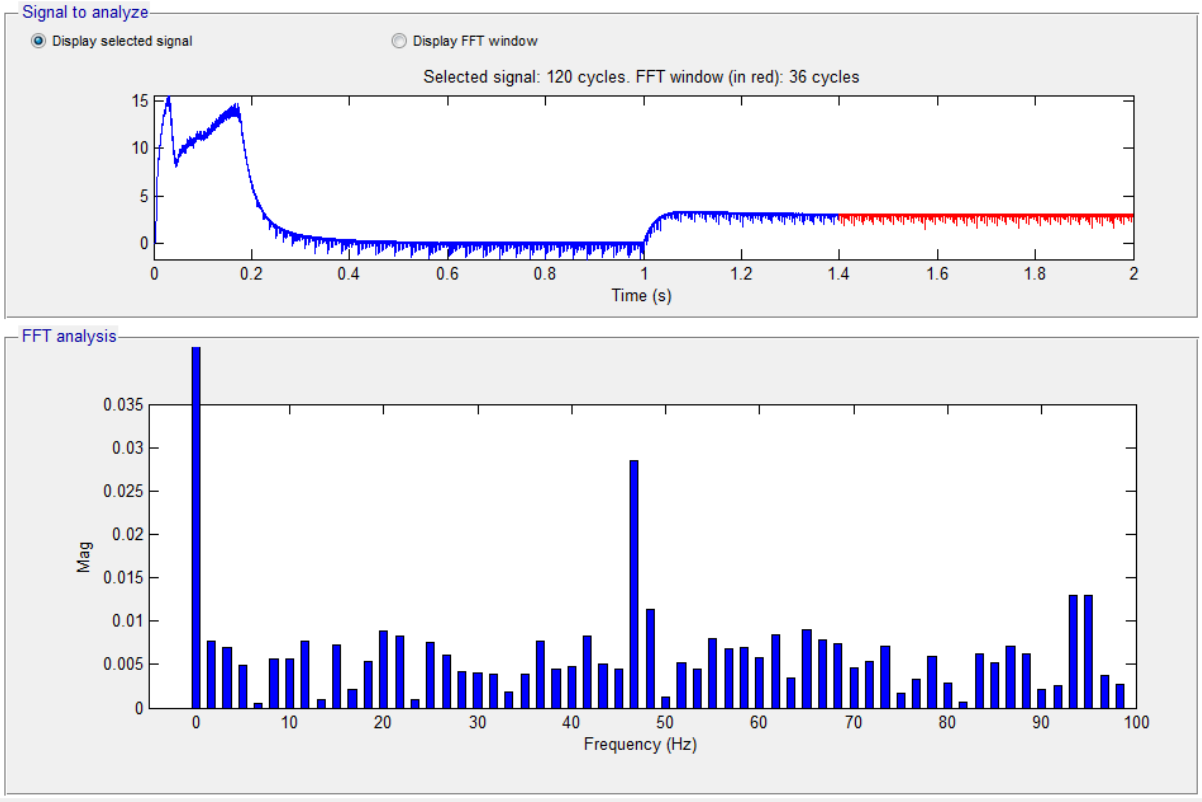


Figure 4.20 L'analyse spectrale du couple électromagnétique obtenu par DTC conventionnelle

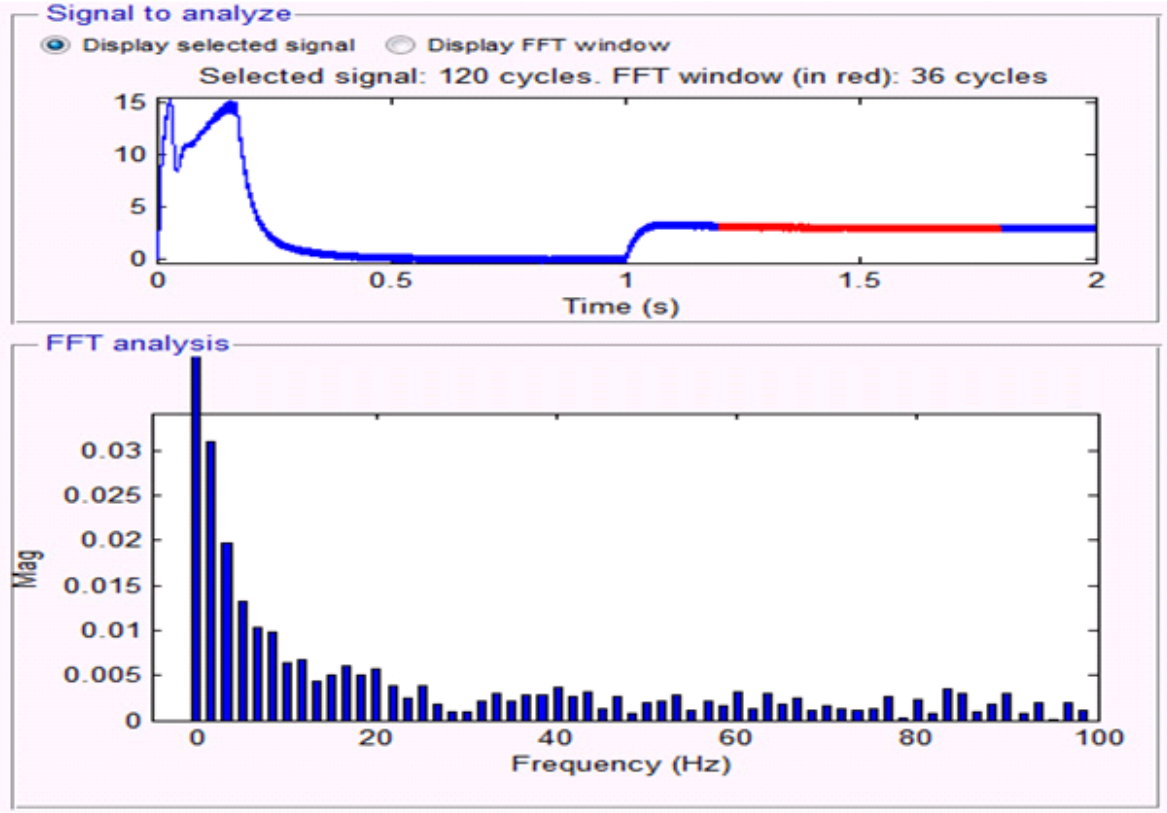


Figure 4.21 L'analyse spectrale du couple électromagnétique obtenu par DTC neuronale

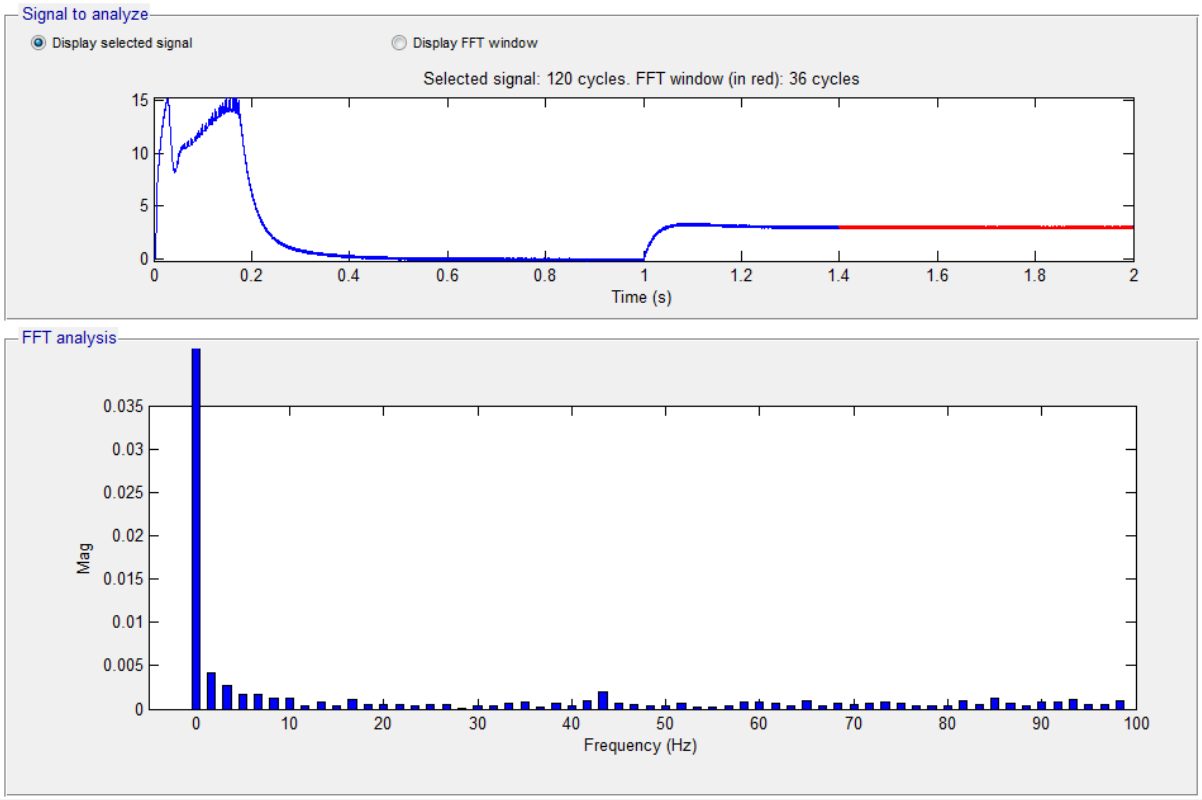


Figure 4.22 L'analyse spectrale du couple électromagnétique obtenu par Fuzzy DTC

L'analyse spectrale par FFT du couple électromagnétique montre aussi l'élimination des ondulations du couple par la DTC floue et dans une moindre mesure par la DTC neuronale par rapport la DTC conventionnelle.

La Table 4.8 présente l'erreur quadratique moyenne et la bande d'ondulation maximale du couple électromagnétique et le flux statorique pour les approches DTC conventionnelle, DTC neuronale et DTC floue.

		Erreur RMS	Max-Min
Couple Electromagnétique (N · m)	DTC Conventionnelle	0.0367	2.164
	DTC Floue	**0.0096**	**0.827**
	DTC Neuronale	0.0314	0.955
Amplitude du flux statorique (Wb)	DTC Conventionnelle	0.0024	0.250
	DTC Floue	0.0017	0.171
	DTC Neuronale	**0.0011**	**0.090**

Tableau 4.8 Ondulations du flux statorique et du couple électromagnétique

Les résultats du tableau 4.8 montrent bien que la commande DTC à base de techniques intelligentes réduit considérablement les ondulations du couple électromagnétique et du flux statorique par rapport la commande DTC Conventionnelle.

L'architecture de la commande DTC floue donne les meilleurs résultats en termes de consommation des ressources hardware et en termes d'élimination des ondulations du couple électromagnétique. La DTC Neuronale et malgré son coût élevé en termes de consommation des ressources hardwares, elle donne des résultats satisfaisants en termes d'élimination des ondulations du couple électromagnétique et des meilleurs résultats pour d'élimination des ondulations au niveau de l'amplitude de flux statorique.

4.6 Conclusion

Dans ce chapitre on a présenté les résultats d'implémentation des architectures proposées pour l'estimateur du flux et du couple, la commande DTC Conventionnelle, la commande DTC Neuronale et la commande DTC floue. En premier lieu on a exposé les résultats de simulation mixte (XSG/Simulink) des architectures proposées. En suite, les architectures hardware proposées sont synthétisées par ISE Xilinx synthétiseur et les résultats de cette phase tel que l'occupation des ressources FPGA et le temps d'exécution sont données et discutées. Apres synthèse on a validé ces architectures hardware par un processus de Co-Simulation hardware. En fin une étude comparative entre les différentes commandes a été faite. Les résultats obtenus montrent que l'implémentation était réussie et que notre contribution au développement de la commande DTC a donné des résultats encourageants, notamment en termes d'économie des ressources hardwares et de minimisation des ondulations du flux et du couple.

Conclusion générale

Conclusion générale

Dans cette thèse, nous avons intégré de nouvelles solutions techniques et technologiques dans le contrôle des machines électriques. Plus précisément, nous avons combiné les techniques de l'intelligence artificielle, tels que les réseaux des neurones artificielles et la logique floue dans la réalisation de la commande DTC de la machine asynchrone. Par la suite, nous avons montré une méthodologie pour l'implémentation hardware de ces commandes sur une carte FPGA cible.

Le but de ce travail à été, en premier lieu l'amélioration des performances dynamiques de la commande direct du couple appliqué sur une machine asynchrone alimenté par un onduleur de tension par l'introduction des techniques de l'intelligence artificielle. Dans un second lieu, de matérialiser la faisabilité et de juger la qualité de la commande proposée.

Dans cette thèse, nous avons décrits essentiellement le développement, l'implémentation et la validation d'une architecture hardware sur FPGA pour une commande DTC à base des techniques intelligentes d'une machine asynchrone.

L'originalité de notre travail a été de combiner les performances des techniques d'intelligence artificielles et la puissance d'exécution des circuits logiques programmables, pour la définition d'une structure de contrôle réalisant les meilleurs rapports simplicité/performance et rapidité/performance. Nous avons fait appel à des outils de commande non conventionnelles pour implanter une stratégie de commutation sans avoir besoin de la table de commutation et les comparateurs à hystérésis utilisés dans la commande DTC classique.

La commande DTC est basée particulièrement sur le contrôle des variables principales tel que le couple électromagnétique et le flux statorique de la machine asynchrone par la sélection directe des vecteurs de tension de la sortie de l'onduleur dans une table de commutation. Ces sélections sont choisies pour qu'ils puissent garder le couple électromagnétique et le flux statorique au voisinage de leurs valeurs de référence dans une bande d'hystérésis bien définie. Néanmoins, l'utilisation des comparateurs à hystérésis dans la commande DTC traditionnel produit des fluctuations du flux statorique et du couple électromagnétique à cause de la largeur des bandes des comparateurs à hystérésis.

Étant donné que notre travail est principalement axé sur la réduction du taux d'ondulation du flux statorique et du couple électromagnétique pour une commande DTC de la machine asynchrone. La contribution de ce travail se place surtout dans l'élaboration d'une stratégie de commutation basée la DTC et les techniques d'intelligentes et sans l'utilisation des comparateurs à hystérésis.

Enfin, nous estimons que la solution proposée a amélioré les performances dynamiques du moteur à induction et réduit considérablement les importunités de la commande DTC conventionnelle telles que les ondulations de couple, les ondulations de flux et la fréquence de commutation.

Bibliographie

[1] I. Takahashi and Y. Ohmori, "High-performance direct torque control of an induction motor,"IEEE Transaction on Industrial Application., vol. 25, pp. 257–264, Mar./Apr. 1989.

[2] M. Depenbrock, "Direct Self Control of Inverter-Fed Induction Machines", IEEE Transaction on Power Electronics, vol. PE-3, no 4, Oct 1988, pp 420-429.

[3] B.BENDJAIMA, DJ.SAIGAAS, DJ.KHODJA, "Fault Tolerant Control Based on Adaptive Fuzzy Sliding Mode Controller for Induction-Motors", International Journal of Intelligent Engineering and System, INASS 2017. PP.39-48, Vol.10, No.6, 2017.

[4] N. HOUICHE, " Commande DTC flou d'un moteur synchrone à aimant permanent", Thèse de Doctorat, Université Mohamed Boudiaf d'M'sila, 2016.

[5] S. GDAIM, A. Matibaa, M. F. Mimouni, " Design and Experimental Implementation of DTC of an Induction Machine Based on Fuzzy Logic Control on FPGA",IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 23, NO. 3, JUNE 2015.

[6] D.E. Khodja, S. Simard, R. Beguenan: «Implementation of Optimized Approximate Sigmoid Function on FPGA Circuit to use in ANN for Control and Monitoring», Control Engineering and Applied Informatics Journal, Vol.17, No.2, pp. 64-72, 2015.

[7] M. Nasir Uddin, M. Hafeez, "FLC-Based DTC Scheme to Improve the Dynamic Performance of an IM Drive". IEEE Transactions on Industry Applications, Vol. 48, No. 2, March/April 2012.

[8] Nassir Mansouri, « Commande par DTC à base de technique d'intelligence artificielle », Mémoire de Master, Université Mohamed Boudiaf d'M'sila, 2015.

[9] B.BENDJAIMA, " Commande tolérante de la machine asynchrone en tenant compte des défauts statoriques et rotoriques", Thèse de Doctorat, Université Mohamed Boudiaf d'M'sila, 2018.

[10] T. Sutikno, N. Nik Idris, A. Zakwan Jidin, M. Zaki Daud, "FPGA Based High Precision Torque and Flux Estimator of Direct Torque Control Drives", Applied Power Electronics Colloquim (IAPEC), 2011 IEEE.

[11] L. BAGHLI, "Contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques", thèse Doctorat, Université Henri Poincaré, Nancy-I, 1999.

[12] Zadeh, L. A., "Fuzzy Sets," Information and Control, vol. 8, pp. 338-353, 1965.

[13] Mamdani, E. H.; Assilian, S., "An experiment in linguistic synthesis with a fuzzy logic controller," International Journal of Man-Machine Studies, vol. 7, pp. 311-323, 1975.

[14] Bühler, H., Réglages par logique floue, Presses Polytechniques Romandes, 1994, 181p.

Bibliographie

[15] Takagi, T.; Sugeno, M., "Derivation of fuzzy control rules from human operator's control actions," in Proc. IFAC Symp. On Fuzzy Information, Knowledge Representation and Decision Analysis, July 1983, pp. 55-60.

[16] Jodouin, J-F., Les réseaux de neurones; principes et définition, Hermes, 1994,124p.

[17] Thiria, S.; Lechevallier, Y.; Gascuel, O.; Canu, S., Statistique et méthodes neuronales, Ed. Dunod, 1997.

[18] H. Chaoui, " Implantation Sur Fpga D'une Loi De Commande Adaptative Neuronale Supervisée Pour Une Articulation Flexible", Mémoire Pour L'obtention Du Grade De Maître Es Science, Université Du Québec, Outaouais, 2005.

[19] F.J. Pineda «Generalization of Back-Propagation to Recurent Neural Networks»,Physical Review letters, 59(19), pp 2229-2232, 1987.

[20] A. M. Rakotozafy," Simulation temps réel de dispositifs électrotechniques", Thèse Doctorat de l'Université de Lorraine, Lorraine, 2014.

[21] J. E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Transaction on Electronic Computers*, vol. EC-8, no.3, pp. 330-334, Sept. 1959.

[22] J.S. Walther, "A Unified Algorithm for Elementary Functions", In: *Proc Of Spring Joint computer conference*, New Jersey, USA, pp 379-385, 1971.

[23] E. Brunelle, M. A. Désautels, "L'algorithme CORDIC", Article de synthèse Bulletin AMQ , Vol. LV, n o 4, décembre 2015

[24] Armato, L. Fanucci, E.P. Scilingo, D. De Rossi, "Ow-error digital hardware implementation of artificial neuron activation functions and their derivative", Microprocessors and Microsystems, Elsevier 35, pp. 557–567, 2011.

[25] Arnaud GAILLARD, " Système éolien basé sur une MADA : contribution à l'étude de la qualité de l'énergie électrique et de la continuité de service", Thèse Doctorat, Université Henri Poincaré, Nancy-I, France, 2010.

[26] Philippe Le-Huy, " Simulation Temps Réel De Convertisseurs De Puissance A L'aide De FPGA", Mémoire pour l'obtention du grade de maître es sciences, Université Laval, QUÉBEC, 2006.

[27] P.J. Ashenden, The Designer's Guide to VHDL Second Edition, Morgan Kaufmann Publishers, ISBN : 1-55860-674-2, 2002.

[28] J. Detrey, "Arithmétiques réelles sur FPGA, virgule fixe, virguleflottante et système logarithmique", Thèse de doctorat de l'École Normale Supérieure de Lyon, France, 2007.

Bibliographie

[29] C. T. Kowalski, J. Lis, T. Orlowska-Kowalska, "FPGA Implementation of DTC Control Method for the Induction Motor Drive", The International Conference on Computer as a Tool EUROCON, Warsaw, September 9-12 2007, pp 1916-1921.

[30] M. A. Zare, R. G. Kavasseri and C. Ababei, " FPGA-based Design and Implementation of Direct Torque Control for Induction Machines", 2014 Internationl Conference on ReConfigurable Computing and FPGAs(ReConFig14), Cancun, 2014, pp. 1-6.

[31] S. Boukadida, S. Gdaim and A. Mtibaa, " Hardware Implementation of FTC of Induction Machine on FPGA", Electronics, Vol. 20, No. 2, December 2016, pp 76-84.

[32] L. Gil-Su, L. Dong-Hyun, Y. Tae-Woong, L. Kyo-Beum, S. Joong-Ho and I. Choy, «Speed and Flux Estimation for an Induction Motor Using a Parameter Estimation Technique», International Journal of Control, Automation, and Systems, Vol.3, N.1, March 2005, pp 79-86.

[33] S. Karimi, P. Poure, S. Saadate, E. Ghlipour, "FPGA-based fully-digital-controller for three phase shunt active filters", International Journal of Electronics, Vol. 95, No. 8, pp. 805-818, August 2008.

[34] S. Karimi, P. Poure, S. Saadate, "An HIL-based reconfigurable platform for design, implementation and verification of electrical systems digital controllers", In IEEE Transactions on Industrial Electronics, vol. 57, no. 4, pp. 1226-1236, 2010.

[35] http://www.xilinx.com/ml402

[36] H. Sudheer, S. Kodad, and B. Sarvesh, **"**Improvements in direct torque control of induction motor for wide range of speed operation using fuzzy logic**"**, *journal of electrical systems and information technology*, vol. 5, no. 3, pp.813-828, 2018.

[37] B. Bossoufi, M. Karim, A. Lagrioui and S. Lonitta, "Performance Analysis of Direct Torque Control (DTC) for Synchronous Machine Permanent Magnet (PMSM)", In: *Proc Of IEEE 16th International Symposium for Design and Technology in Electronic Packaging (SIITME),* Pitesti, Romania, pp. 237-242, 2010.

[38] N. *Farah, M. H. N. Talib,* Z. Ibrahim, Q. Abdullah, O. Aydogdu, Z. Rasin, A. Jidin and J. M. Lazi, "Analysis and investigation of different advanced control strategies for high-performance induction motor drives", *TELKOMNIKA*, vol. 18, no. 6, pp. 3303-3314, December 2020.

[39] T. Porselvi, K. Deepa and R. Mutha, "FPGA Based Selective Harmonic Elimination Technique for Multilevel Inverter", *International Journal of Power Electronics and Drive System (IJPEDS),* Vol. 9, no. 1, pp. 166-173, March 2018.

[40] M. Selvarathinam, T. Deepika, D. Devipriya and P. Muthukumar, "A Novel FPGA Based Direct Torque Control for Induction Motor", *Middle-East Journal of Scientific Research,* vol. 24, no. 3, pp.787-793, 2016.

Bibliographie

[41] N. Kabache, S. Moulahoum and H. Houassine, “Hardware Co-Simulation of an Adaptive Field Oriented Control of Induction Motor”, *Journal of International Conference on Electrical Machines and Systems*, vol. 3, no 2, pp. 116-120, 2014.

[42] A. Mohammedi, N. Kabache, S. Moulahoum and H. Houassine, "FPGA Hardware In the Loop Validation of Direct Torque Control for Induction Motor", 2015 20th International Conference on Method and Models in Automation and Robotics (MMAR), Miedzyzdroje, 2015, pp 812-816.

[43] T. Sutikno A. Z. Jidin, A. Jidin, N. R. N. Idris, "FPGA Based High Performance Torque and Flux Estimator" ,International Review of Electrical Engineering(I.R.E.E), Vol. 6, No.1, February 2011, pp 207-214.

[44] T. Sutikno ,N. R. N. Idris, A. Jidin and N. Cirstea, “An Improved FPGA Implementation of Direct TorqueControl for Induction Machines”, IEEE Transactions on Industrial Informatics, VOL. 9, NO. 3, 2013, pp. 1280-1290.

[45] A. Aib, D. E. Khodja, L. Benyettou, S. Chakroune, " FPGA Hardware in the Loop Validation of Torque and Flux Estimators for Direct Torque Control (DTC) of an Induction Motor (IM) ", International Journal of Intelligent Engineering and Systems, Vol.14, No.5, pp 583-594, 2021.

[46] O. Sandre-Hernandez, J. Ranglel-Magdaleno, R. Morales-Caporal and E. Bonilla-Huerta, “HIL simulation of the DTC for a three-level inverter fed a PMSM with neutral-point balancing control based on FPGA”, *Electrical Engineering*, Issue 3/2018, pp. 1441-1454, 2018.

[47] E. Lotfi, M. Elharoussi and E. Abdelmounim, “VHDL design and FPGA implementation of direct torque control for induction machines”, *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1220-1231, June 2021.