République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche Scientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**
Filière : **Automatique**
Spécialité : **Automatique et Informatique Industrielle**

Présenté par
Djedjiga NECHAT
Malia TEMZI

Thème

Implémentation sur FPGA d'une commande neuro-floue : Application à la commande d'un pendule inversé.

Mémoire soutenu publiquement le 26/09/ 2024 devant le jury composé de :

M Ahcene LAKHLEF
MCA, UMMTO, **Président**

M Houcine KHATI
MCB, UMMTO, **Encadrant**

M Ouiza NAIT BELAIDE
MCB, UMMTO, Examinateur

M Prénom AMROUN
MCB, UMMTO, Examinateur

Remerciements

En premier lieu, nous remercions Dieu, le tout puissant, pour nous avoir donné le courage, la patience, la volonté et la force nécessaires, pour affronter toutes les difficultés et les obstacles, qui se sont hissés au travers de notre chemin, durant toutes nos années d'études.

Nous tenons à exprimer nos vifs remerciements et nos sincères reconnaissances à Mr KHATI Hocine, pour sa disponibilité, ses conseils judicieux, ses directives et ses orientations concernant notre projet de fin d'étude, tout en nous accordant sa confiance et en nous faisant profiter de sa large expérience tout au long de la réalisation de ce projet de fin d'étude.

Nos sincères remerciements vont également aux membres du jury pour l'honneur qu'ils nous ont fait en assistant à notre soutenance et en évaluant ce travail.

Nos remerciements s'adressent, autant, au chef de département, M. TOUAT Mohan Achour, ainsi qu'à tous nos professeurs pour leur générosité et leur précieuse contribution à notre réussite tout au long de nos études.

Nous adressons nos remerciements sincères à nos familles et à nos ami(e)s pour leurs encouragements, leur patience, et leur grand soutient durant nos études.

Dédicaces

Je dédie ce travail :

À mes chers parents,Mouloud & Chabha, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À mes chers frères, Ahcen et Ghilas, pour leur appui , leur encouragement,

À mes chères amies ,Chabha,Sadia et thanina pour leurs encouragements permanents, et leur soutien moral,

À Mati, Ninou et Félixe, compagnons de chaque moment de réflexion et de travail, votre présence silencieuse mais réconfortante a rendu ce chemin plus doux et plus joyeux.

À ma binôme,Malia, avec qui j'ai partagé chaque défi et chaque réussite de ce travail,

À toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible,

Merci d'être toujours là pour moi..❤

NECHAT Djedjiga

Dédicaces

Je dédie ce travail :

À mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À mes chers frères et soeurs, pour leur appui , leur encouragement,

À mon chère ami ,Ali, pour son encouragements permanents, et son soutien moral,

À ma binôme,Djedjiga, avec qui j'ai partagé chaque défi et chaque réussite de ce travail,

À toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible,

Merci d'être toujours là pour moi..❤

TEMZI Malia

Notations

FIL :FPGA in the loop
RNA :réseaux de neurones artificiels
FPGA : Field Programmable Gate Array
PID : Proportional Integral Derivative
ANFIS : Adaptive Neuro-Fuzzy Inference System
ASIC : Application-Specific Integrated Circuit
HDL : Hardware Description Languag
CLB : Configurable Logic Block
IOB : Input Output Block
VHSIC : Very High Speed Integrated Circuit
SRAM : Static Random Access Memory
EPROM : Erasable Programmable Read-Only Memory
EEPROM : Electically Erasable Programmable Read-Only Memory
ROM : Read-Only Memory
DSP : Digital Signal Processor
SoC : System on Chips
DCM : Digital Clock Manager
USB : Universal Serial Bus
I2C : Inter-Integrated Circuit
LUT : Look Up Table
PLL : Phase-Locked Loop
CPLD : Complex Programmable Logic Device
ARM : Advanced RISC Machines
RISC : Reduced Instruction Set Computing
PS : Processor System
PL : Programmable Logic
LE :éléments logiques
SDRAM : Synchronous Dynamic Random Access Memory
DDR : Double Data Rate
JTAG : Joint Test Action Group
PLI :interface de langage de programmation
OVI :Association internationale du Verilog ouvert
IEEE :Institut des ingénieurs électriciens et électroniciens
IDE :environnement de développement intégré

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Dans le domaine du contrôle des systèmes dynamiques, l'intégration des techniques de l'intelligence artificielle a ouvert de nouvelles voies pour le développement de régulateurs performants et adaptatifs. Parmi ces approches révolutionnaires, les systèmes neuro-flous se distinguent en combinant les avantages des réseaux de neurones artificiels et de la logique floue, offrant ainsi une solution robuste pour la commande de systèmes complexes.

Ce mémoire se concentre sur l'implémentation d'un régulateur neuro-flou (ANFIS) sur une plateforme matérielle FPGA, en utilisant la technique FPGA in the loop (FIL). L'objectif principal est d'appliquer cette approche à la commande d'un pendule inversé, un système non linéaire souvent utilisé comme banc d'essai pour évaluer les performances des régulateurs.

Les systèmes neuro-flous présentent un intérêt particulier en raison de leur capacité à traiter des informations incertaines ou imprécises tout en bénéficiant des capacités d'apprentissage et d'adaptation des réseaux de neurones. Cette combinaison unique de méthodes permet d'obtenir des régulateurs capables de s'adapter dynamiquement aux variations et aux perturbations du système en temps réel.

L'utilisation des FPGA offre des avantages significatifs pour l'implémentation de ces régulateurs en temps réel. Les FPGA offrent une vitesse de traitement élevée, un niveau élevé de parallélisme et une reconfigurabilité, ce qui les rend idéaux pour les applications de contrôle en temps réel.

La technique FIL représente une avancée majeure dans le domaine de la conception de systèmes de contrôle. Elle permet de simuler et de tester les algorithmes de commande directement sur le FPGA, en intégrant étroitement le modèle de simulation avec le matériel FPGA. Cette approche assure une validation plus précise des régulateurs, en garantissant que les performances observées en simulation se traduisent fidèlement dans l'implémentation matérielle.

Ce mémoire est structuré en quatre chapitres principaux :

Le premier chapitre introduit les concepts fondamentaux des systèmes neuro-flous, en expliquant les principes de la logique floue et des réseaux de neurones, et comment leur combinaison forme un régulateur neuro-flou performant.

Lle deuxième chapitre décrit les circuits FPGA ainsi que les outils permettant leur programmation, nous avons cité d'abord les technologies de programmation des FPGAs,

par la suite nous avons présenté les FPGAs du constructeur Altéra, plus particulièrement ceux de la famille DE2-115, utilisés dans ce mémoire.

Le troisième chapitre expose le processus d'implémentation du régulateur ANFIS sur la carte FPGA, en utilisant la technique FPGA-in-the-loop.

Ce dernier chapitre présente les résultats obtenus de la simulation et l'implémentation du régulateur sur le système de pendule inversé, suivi d'une analyse des performances .

En combinant les avantages des systèmes neuro-flous avec la puissance de traitement et la flexibilité des FPGA, ce projet vise à fournir une solution innovante et efficace pour la commande de systèmes non linéaires complexes, ouvrant ainsi la voie à de nouvelles avancées dans le domaine du contrôle intelligent.

Problématique

Dans le domaine du contrôle des systèmes dynamiques, la conception de régulateurs efficaces représente un défi majeur, particulièrement pour les systèmes non linéaires et instables comme le pendule inversé. Les approches traditionnelles de contrôle, telles que les régulateurs PID (Proportionnel Intégral Dérivé), montrent souvent leurs limites face à la complexité de ces systèmes et à leur sensibilité aux perturbations. Un régulateur performant doit non seulement maintenir des performances élevées malgré les variations et les perturbations du système, mais aussi être implémenté de manière efficiente en temps réel. Cette mise en œuvre requiert l'utilisation de techniques avancées pour garantir la fiabilité et la réactivité du système dans des situations réelles. Par conséquent, il est important de développer des méthodes de contrôle innovantes et robustes, capables de surmonter les limitations des approches traditionnelles et de répondre aux exigences rigoureuses des applications pratiques. La mise en œuvre pratique nécessite des cartes de contrôle puissantes en termes de calcul en raison de la complexité des opérations requises par les contrôleurs intelligents.

Chapitre 1

Généralités sur la commande neuro-floue

1.1 Introduction

Au cours des dernières décennies, des efforts considérables ont été déployés pour résoudre divers problèmes de commande dans le domaine de l'industrie et de la robotique. L'émergence de techniques d'intelligence artificielle telles que la logique floue et les réseaux de neurones artificiels (RNAs) a joué un rôle crucial pour remédier à ces contraintes de contrôle. Ces avancées ont été motivées par la volonté de comprendre comment le système nerveux biologique traite l'information et comment exploiter les propriétés du cerveau humain dans les systèmes réels.

Les techniques d'intelligence artificielle, telles que la logique floue, les réseaux de neurones et les réseaux neuro-flous, sont largement utilisées pour résoudre des problèmes complexes dans divers domaines industriels et robotiques. Ces techniques sont particulièrement adaptées pour contrôler des systèmes non linéaires et complexes, où il est souvent difficile de disposer d'un modèle mathématique précis. La logique floue, par exemple, permet de contrôler efficacement des systèmes non linéaires à plusieurs entrées-sorties en utilisant des règles floues basées sur un raisonnement humain.

Les RNAs offrent une autre approche puissante en permettant l'ajustement automatique des paramètres du système flou grâce à leur capacité d'apprentissage. En combinant les capacités d'inférence de la logique floue avec la capacité d'apprentissage des réseaux de neurones, les réseaux neuro-flous permettent de former des systèmes de contrôle robustes et adaptatifs.

Dans cette optique, les recherches se sont orientées vers le développement de régulateurs neuro-flous, qui combinent les avantages des RNAs et de la logique floue pour créer des systèmes de commande efficaces et flexibles. Ces systèmes hybrides permettent de tirer parti des capacités d'apprentissage des RNAs tout en exploitant les propriétés de

raisonnement de la logique floue, ouvrant ainsi de nouvelles perspectives dans le domaine du contrôle des systèmes complexes.

Dans la première partie de ce chapitre, nous allons présenter les principes de la logique floue et la commande floue. La deuxième partie sera consacrée à la description des réseaux de neurones artificiels et leurs différentes architectures. Par la suite, dans la troisième partie du chapitre, nous allons aborder les techniques neuro-floues ainsi que les différentes combinaisons les plus utilisées dans le contrôle des systèmes.

1.2 Logique floue

1.2.1 Définition

La logique est l'étude des méthodes de raisonnement, où le raisonnement signifie obtenir de nouvelles propositions à partir de propositions existantes. La logique classique est basée sur des propositions vraies ou fausses, avec une valeur de vérité de 1 ou 0. Cette logique à deux valeurs a dominé le monde scientifique depuis plus d'un siècle. Mais il existe de nombreux problèmes du monde réel où la logique traditionnelle à deux valeurs n'a pas bien fonctionné ou n'a pas pu être appliquée en raison du fait des valeurs de vérité absolue.

Cependant, la logique floue généralise cette approche en permettant des valeurs de vérité partielles entre 0 et 1, ce qui facilite le raisonnement approximatif à partir de prémisses imprécises. Contrairement à ce que son nom suggère, la logique floue n'est pas vague mais plutôt une méthode précise pour traiter l'imprécision et le raisonnement approximatif.

Zadeh qui est le fondateur de la logique floue en 1965, soutient que cette logique peut être considérée comme une tentative de formalisation ou de mécanisation de deux capacités humaines. Premièrement, les humains sont capables de converser, de raisonner et de prendre des décisions rationnelles sur l'imprécision, l'incertitude, l'incomplétude de l'information, les informations contradictoires, la partialité de la vérité et la partialité de la possibilité dans un environnement d'information imparfaite. Deuxièmement, ils sont capables d'effectuer une grande variété de tâches physiques et mentales sans aucune mesure et aucun calcul[1].

1.2.2 Bases générales de la logique floue

1.2.2.1 Sous-ensemble ordinaire et sous-ensemble flou

Un sous-ensemble ordinaire A d'un ensemble U est défini par sa fonction d'appartenance $\mu_A(x)$ qui caractérise chaque élément x appartenant à U [37].

$$\mu_{A(x)=} \begin{cases} 1 & si\ x \epsilon A \\ 0 & sinon \end{cases} \tag{1.1}$$

Cette fonction d'appartenance ne peut prendre que deux valeurs possibles : la valeur 1 si x appartient à A ou la valeur 0 si x n'appartient pas à A.

Un sous-ensemble flou A d'un ensemble U défini par une fonction d'appartenance $\mu_A(x)$ peut prendre plusieurs valeurs comprises entre 0 et 1. Le degré d'appartenance de l'élément x au sous-ensemble flou$A$ appartient à l'intervalle[0 1][1].

$$\mu_A(x) \in [0\ 1] \tag{1.2}$$

Les ensembles flous sont un moyen de réaliser l'interface entre l'information numérique (quantitative) et l'information symbolique (linguistique).

1.2.2.2 Variable et valeurs de variable linguistique

La variable linguistique représente un phénomène ou une grandeur, et pour décrire ces phénomènes, on utilise généralement des expressions floues comme : très grand, grand, moyen, petit, ...etc.Ces expréssions sont appelées des valeurs linguistiques.

1.2.2.3 Fonctions d'appartenance

Les variables linguistiques sont exprimées à travers des fonctions d'appartenance, où chaque variable floue x d'un sous-ensemble flou A est liée à une fonction d'appartenance. Cette fonction représente le degré d'appartenance de x à A[3].

La fonction d'appartenance d'un sous-ensemble flou A est définie par :

$$\mu_A = X \rightarrow \left[\ 0 \quad 1\ \right] x \mapsto \mu_A(x) \tag{1.3}$$

$\mu_A(x)$: est le degré d'appartenance de x au sous-ensemble l'ensemble flou A.

Les fonctions d'appartenance peuvent être représentées sous plusieurs formes selon l'application [4] :

- **Fonction triangulaire** : elle est définie par trois paramètres (a, b, c) (figure (1.1(a)) :

$$\mu_{A(x)=} \begin{cases} \frac{x-a}{b-a} & si\ x \in [a,b] \\ \frac{c-x}{c-b} & si\ x \in [b,c] \\ 0 & ailleurs \end{cases} \tag{1.4}$$

- **Fonction trapézoïdale** : elle est définie par quatre paramètres (a, b, c, d) (figure (1.1(b)) :

$$\mu_{A(x)=} \begin{cases} \frac{x-a}{b-a} & si\ x \in [a,b] \\ 1 & si\ x \in [b,c] \\ \frac{d-x}{d-c} & si\ x \in [c,d] \\ 0 & ailleurs \end{cases} \tag{1.5}$$

- **Fonction gaussienne :** elle est définie par deux paramètres (m, s)(figure (1.1(c)) :

$$\mu(x) = \exp\left(-\left(\frac{x-m}{2s}\right)^2\right) \tag{1.6}$$

où m : centre de la gaussienne et s : sa largeur.

- **Fonction sigmoïde :** La fonction sigmoïde est définie par deux paramètres (a et b) (figure (1.1(d)) :

$$\mu(x) = \left(\frac{1}{1 + \exp(-a(x-c)} \right) \tag{1.7}$$

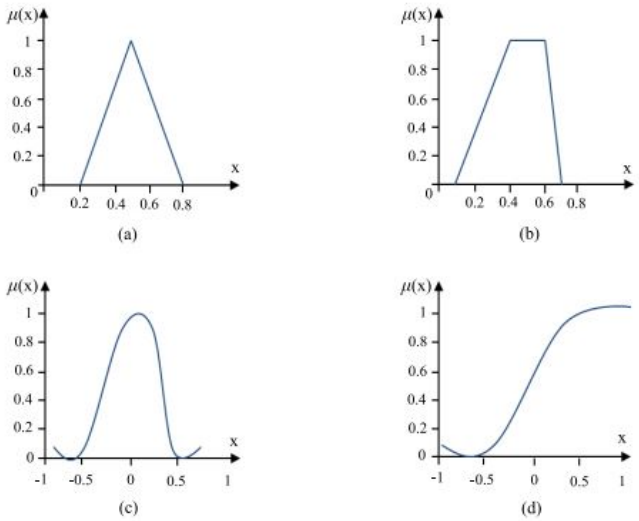


FIGURE 1.1 – Fonctions d'appartenance usuelles.

1.2.2.4 Univers de discour

L'univers de discours est l'ensemble des valeurs possibles qu'une variable linguistique peut prendre. Il représente la plage de fonctionnement du processus que le régulateur flou doit prendre en compte lors de son réglage[5].

1.2.2.5 Opérations sur les ensembles flous

Dans les ensembles classiques, les variables sont liées par des opérateurs logiques comme ET (Intersection), OU (Union) et NON (Complément à1). En revanche, dans la logique floue, ces opérateurs sont interprétés différemment : ils sont représentés respectivement par les opérations de Minimum, Maximum et Complément à 1, et sont définis à travers leurs fonctions d'appartenance.

La représentation de ces opérateurs dans les deux logiques, classique et floue, est présentée dans le tableau suivant [5] :

	Logique classique	Logique floue
$C = A$ ET B	$C = A \cap B$	$\mu_c(x) = \min(\mu_A(x), \mu_B(x))$
$C = A$ OU B	$C = A \cup B$	$\mu_c(x) = \max(\mu_A(x), \mu_B(x))$
$C = NON(A)$	$C = A$	$\mu_c(x) = 1 - \mu_A(x)$

TABLE 1.1 – Les opérateurs dans la logiques classique et floue.

1.2.2.6 Régles d'inférences

Une règle floue définit une relation entre deux énoncés flous, chacun ayant un rôle spécifique. En d'autres termes, elle établit une correspondance entre des conditions imprécises pour déterminer une action ou une conclusion[4].

Ces règles sont décrites sous la forme suivante [5] :

Si condition 1 ET/OU condition 2(ET/OU...) alors action sur les sorties OU

Si condition 3ET/OU condition 4(ET/OU...) alors action sur les sorties OU

...

Si condition n ET/OU condition $n+1$ (ET/OU...) alors action sur les sorties.

1.2.2.7 Commande floue

Les techniques de contrôle classique, bien que largement utilisées, présentent des limites lorsqu'il s'agit de gérer des systèmes complexes et non linéaires. En effet, la modélisation mathématique précise de ces systèmes peut s'avérer difficile, voire impossible, en raison de leur nature incertaine et dynamique. C'est là que le contrôle flou intervient comme une alternative prometteuse. Inspirée du raisonnement humain, la logique floue permet de modéliser les comportements imprévisibles des systèmes complexes en utilisant l'approche linguistique de Zadeh. Cette approche se base sur des règles floues et des variables linguistiques, ce qui la rend facile à comprendre et à mettre en œuvre. De plus, le contrôle flou ne nécessite pas de modèle mathématique précis du système, ce qui le rend particulièrement efficace pour les systèmes non linéaires et partiellement connus[1]. Dans ce qui suit, nous allons présenter les composants essentiels d'un régulateur flou :

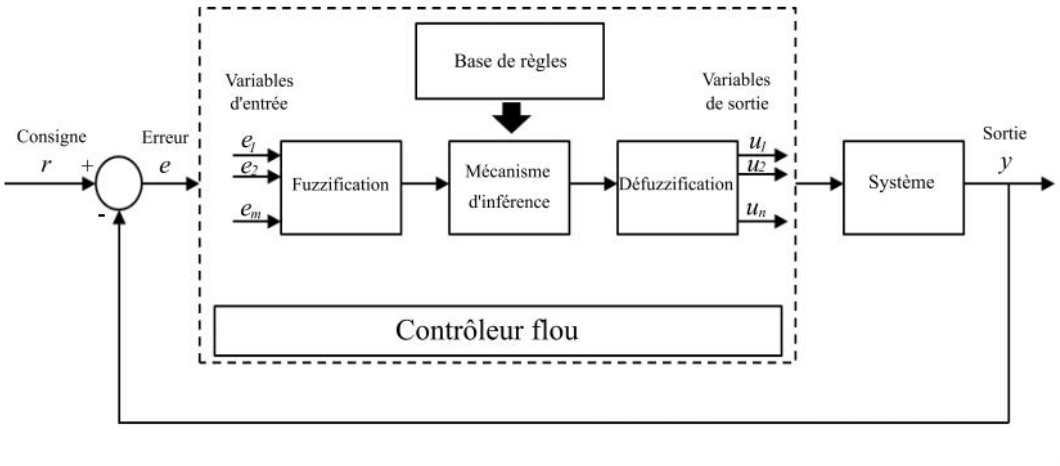


FIGURE 1.2 – Schéma fonctionnel d'un régulateur flou.

- **Fuzzification**

La fuzzification définit la transformation des données d'entrée observées en ensembles flous dans un certain univers de discours d'entrée. Ce processus consiste à associer à

chaque ensemble flou une fonction d'appartenance dont le choix dépend principalement du domaine d'application et de la facilité de calcul[1].

Considérons une température(T_i) définie par l'ensemble des variables linguistiques : F=faible M=moyenne E=elevée.L'univers de discours associé est [14 26].

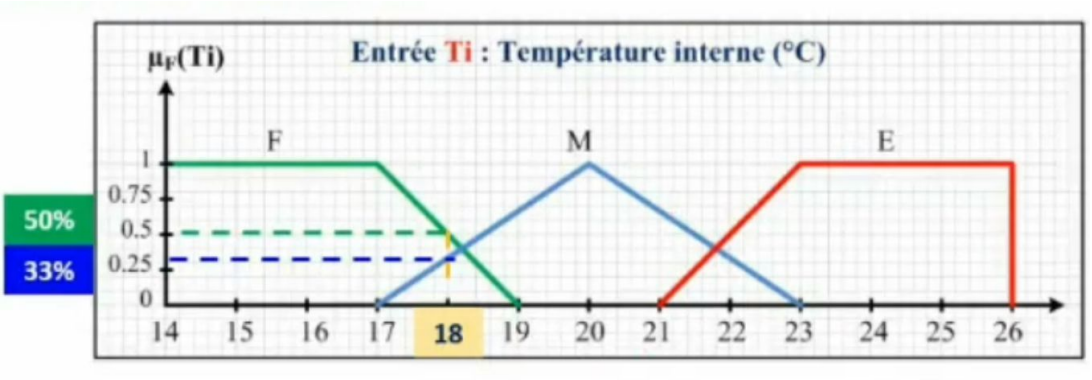


FIGURE 1.3 – Exemple de fuzzification

Les degrés d'appartenance de l'entrée Ti aux sous-ensembles sont :

$\mu_F(Ti) = 0.5$

$\mu_M(Ti) = 0.33$

$\mu_E(Ti) = 0$

- **Base de régles**

La base de règles est un élément central d'un système flou. Elle définit le comportement du système en fonction des différentes entrées possibles.La base de règles est composée d'un ensemble de règles floues sous la forme [1] :

Si condition 1 ET/OU condition 2 (ET/OU. . .) alors action sur les sorties

Si condition 3 ET/OU condition 4 (ET/OU. . .) alors action sur les sorties

. . .

Si condition n ET/OU condition $n+1$ (ET/OU. . .) alors action sur les sorties.

Dans le cas d'un régulateur flou, les entrées peuvent être l'erreur, le changement d'erreur ou la somme de l'erreur, et la sortie peut être l'action de contrôle ou le changement d'action de contrôle.

- **Mécanisme d'inférence**

Cette étape consiste à transformer la partie floue obtenue par la fuzzification en une nouvelle partie floue . Elle comprend trois composants : une base de règles contenant une sélection de règles floues, une base de données qui définit les fonctions d'appartenance utilisées dans les règles floues et un mécanisme de raisonnement qui exécute la procédure d'inférence sur les règles pour obtenir une conclusion raisonnable.

Les modèles flous largement connus utilisés dans les applications de contrôle sont :

1. Les modèles flous de Mamdani
2. Les modèles flous de Takagi-Sugeno

3. Les modèles flous de Tsukamoto

Les modèles flous de Mamdani et de Takagi-Sugeno sont les plus largement utilisés. Cela est dû au fait que le modèle flou de type Mamdani est facile à appliquer sans beaucoup d'informations a priori sur le système, tandis que le modèle flou de type Takagi-Sugeno nécessite l'estimation des paramètres conséquents à partir des données disponibles.

Un autre aspect important du contrôle flou est le choix du mécanisme d'inférence. Il n'y a pas de règle stricte pour le choix d'une inférence floue spécifique. Cela dépend principalement de la préférence du concepteur, de la disponibilité d'informations a priori et du domaine d'application. L'applicabilité de l'analyse de stabilité devrait être un autre critère pour le choix des mécanismes d'inférence. À cet égard, les modèles flous de type Takagi-Sugeno présentent des avantages distincts par rapport aux deux autres types de modèles.Une grande variété de techniques d'analyse de stabilité est facilement applicable aux régulateurs flous de type Takagi-Sugeno[1].

- **Défuzzification**

Cette étape représente l'inverse de la fuzzification.Cette étape consiste à transformer la sortie floue de l'inférence en une valeur numérique pour créer un signal de contrôle pour le système à commander. Plusieurs méthodes de défuzzification sont disponibles, notamment le centre de gravité, la moyenne des maximas et le centre des maximas. Cependant, la méthode la plus couramment utilisée est la défuzzification par centre de gravité, qui calcule l'abscisse du centre de gravité de la surface formée par la fonction d'appartenance résultante[1].

1.2.2.8 Avantages et désavantages du réglage par logique floue

- **Avantages**

Le réglage par logique floue présente les avantages suivants :

La non nécessité de la modélisation du système.

La possibilité d'implémenter des connaissances linguistiques de l'opérateur du processus.

La maîtrise du système à régler avec un comportement complexe (systèmes non linéaires et difficiles à modéliser).

La disponibilité de système de développement efficace, soit par microprocesseur ou PC, soit par circuits intégrés.

- **Désavantages**

Le manque de directives précises pour la conception d'un réglage (choix de grandeur mesurée, détermination de la fuzzification, des inférences et de la défuzzification).

L'approche artisanale et asymptotique (implantation des connaissances de l'opérateur souvent difficile).

L'impossibilité de la démonstration de la stabilité du circuit de réglage en toute généralité (en l'absence d'un modèle valable).

La cohérence des inférences non garanties (possibilité d'apparition de règles d'inférence contradictoires).

1.3 Réseaux de neurones

1.3.1 introduction

Les réseaux de neurones artificiels peuvent être considérés comme des modèles mathématiques simplifiés de systèmes semblables au cerveau[6].

Les structures des réseaux de neurones artificiels ont été développées à partir de modèles connus des systèmes nerveux biologiques et du cerveau humain lui-même. Les composants computationnels ou unités de traitement, appelés neurones artificiels, sont des modèles simplifiés des neurones biologiques. Ces modèles ont été inspirés par l'analyse de la façon dont une membrane cellulaire d'un neurone génère et propage des impulsions électriques[7].

Les réseaux neuronaux artificiels représentent une nouvelle génération de systèmes de traitement de l'information inspirés de la biologie, massivement parallèles et distribués. Ils se composent d'éléments de traitement (également appelés nœuds) et de connexions entre eux avec des coefficients (poids) liés à ces connexions, qui constituent la structure neuronale, ainsi que des algorithmes d'apprentissage attachés à cette structure[8].

L'intérêt pour les réseaux neuronaux artificiels est principalement dû à leurs caractéristiques essentielles telles que l'apprentissage par l'exemple et l'adaptation, la généralisation, le parallélisme massif, le stockage distribué et associatif de l'information, la robustesse et la tolérance aux pannes [8].

Les modèles des réseaux de neurones artificiels sont classés par rapports à leurs méthodes d'apprentissage (supervisé et non supervisé), leurs architectures (bouclés et non bouclés) ou leurs types de sortie (binaire ou continue) [8].

1.3.2 Neurone biologique

Le neurone, élément essentiel du système nerveux central, joue un rôle important dans la transmission des impulsions électriques engendrées par des réactions physico-chimiques dans des conditions spécifiques.

Il se compose de quatre éléments principaux : les dendrites, le corps cellulaire, l'axone et les synapses (figure 1.4).

Les dendrites assurent la réception continue des signaux provenant d'autres neurones.

Le corps cellulaire est chargé du traitement de toutes les informations issues des dendrites.

L'axone guide les signaux vers d'autres neurones connectés.

Les synapses sont des jonctions qui facilitent le transfert des signaux électriques des axones d'un neurone donné vers les dendrites d'autres neurones[7].

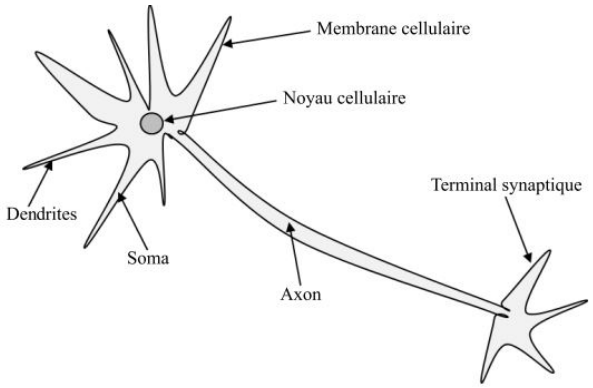


FIGURE 1.4 – Neurone biologique.

1.3.3 Neurone formel

Le neurone formel, qui constitue la pierre angulaire des réseaux neuronaux artificiels, est un automate dont ses principaux éléments peuvent être décrits comme suit (pour un neurone d'indice i) [9] :

- Son état a (aussi appelé activation) , qui peut être une valeur réelle ou booléenne. Cet état est généralement choisi comme valeur de sortie du neurone ;
- Ses connexions d'entrée auxquelles sont associés des poids w_{ij} (j est l'indice du neurone partageant la connexion) ;
- Sa fonction d'entrée réalisant un prétraitement (généralement une somme pondérée) des entrées ;
- Sa fonction d'activation (ou de transfert) , qui calcule à partir du résultat de la fonction d'entrée et l'activation du neurone.

La figure(1.5) représente un neurone formel appliquant une fonction de seuil sur la somme pondérée de ses différentes entrées.

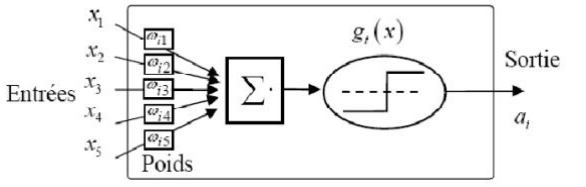


FIGURE 1.5 – Un neurone formel

1.3.4 Architectures des réseaux de neurones artificiels

1.3.4.1 Réseaux à une seule couche

Ce réseau de neurones artificiels comporte une seule couche d'entrée et une seule couche neuronale, qui constitue également la couche de sortie. La figure (1.6) illustre un réseau à une seule couche composé de n entrées et de m sorties.

L'information circule toujours dans une seule direction (donc, unidirectionnelle), c'est-à-dire de la couche d'entrée à la couche de sortie. À partir de la figure (1.6), on peut voir que dans les réseaux appartenant à cette architecture, le nombre de sorties du réseau coïncide toujours avec le nombre de neurones. Ces réseaux sont généralement utilisés dans des problèmes de classification de motifs et de filtrage linéaire[7].

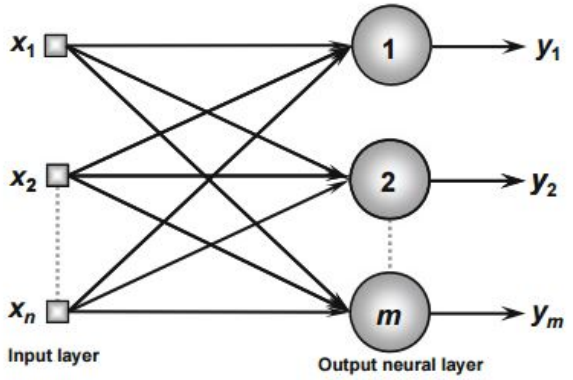


FIGURE 1.6 – Réseaux à une seule couche

1.3.4.2 Réseaux multicouche (MLP)

Contrairement aux réseaux appartenant à l'architecture précédente, les réseaux à propagation avant avec plusieurs couches sont composés d'une ou plusieurs couches neuronales cachées .

À partir de la figure (1.7), il est possible de comprendre que la quantité de neurones composant la première couche cachée est généralement différente du nombre de signaux composant la couche d'entrée du réseau. En effet, le nombre de couches cachées et leur quantité respective de neurones dépendent de la nature et de la complexité du problème que le réseau tente de résoudre, ainsi que de la quantité et de la qualité des données disponibles sur le problème. Néanmoins, tout comme pour les réseaux à propagation avant à couche unique, le nombre de signaux de sortie coïncidera toujours avec le nombre de neurones de cette couche respective[7].

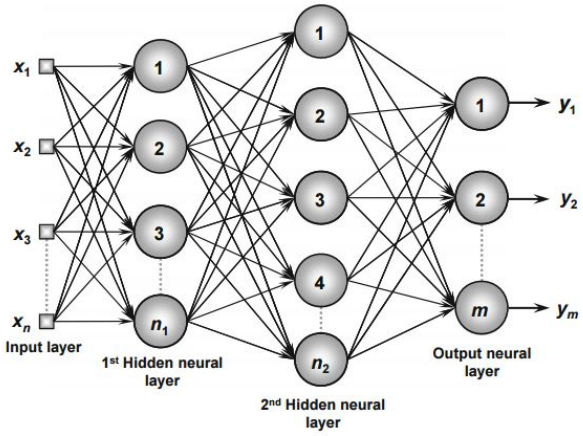


FIGURE 1.7 – Un réseau multicouche

1.3.4.3 Réseaux à connexions récurrentes

Dans ces réseaux, les sorties des neurones sont utilisées comme entrées de rétroaction pour d'autres neurones. La fonction de rétroaction qualifie ces réseaux pour le traitement dynamique de l'information[7].

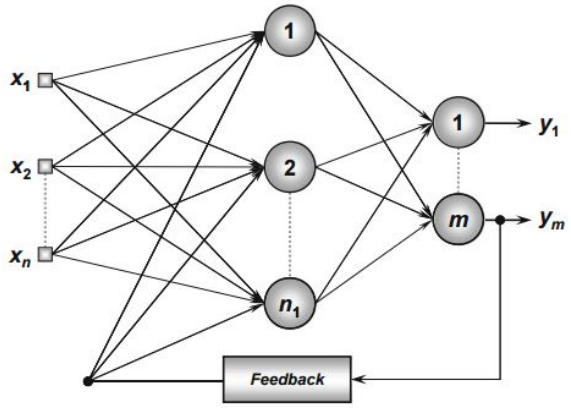


FIGURE 1.8 – Réseau à connexions récurrentes.

1.3.5 Processus d'apprentissage

La capacité à apprendre et à généraliser à partir d'un ensemble de données d'entraînement est l'une des fonctionnalités les plus puissantes des réseaux de neurones artificiels. Les situations d'apprentissage dans les résaux de neurones peuvent être classées en deux types, à savoir supervisé et non supervisé[10].

1.3.5.1 Apprentissage supervisé

Dans l'apprentissage supervisé, à la fois les entrées et les sorties sont fournies. Le réseau traite ensuite les entrées et compare ses sorties résultantes avec les sorties désirées. Une

comparaison est faite entre la sortie calculée par le réseau et la sortie attendue corrigée pour déterminer l'erreur puis ajuster les poids des connexions pour minimiser la différence entre les deux résultats. L'erreur peut ensuite être utilisée pour modifier les paramètres du réseau, ce qui entraîne une amélioration des performances [10].

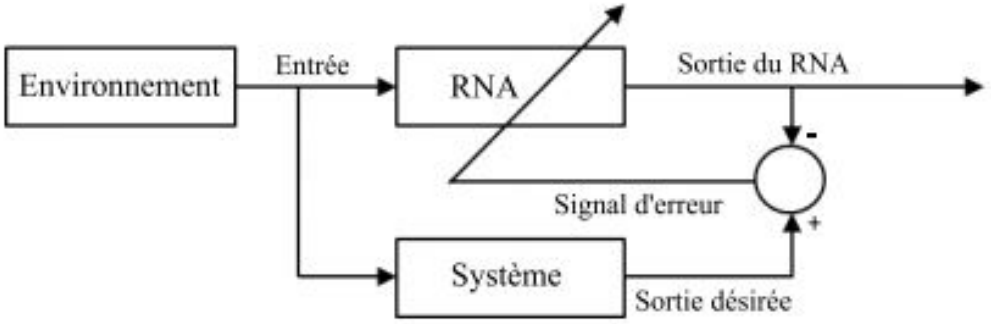


FIGURE 1.9 – Apprentissage supervisé.

1.3.5.2 Apprentissage non supervisé

L'apprentissage non supervisé, le réseau reçoit des entrées mais pas de sorties désirées. Le système lui-même doit alors décider quelles caractéristiques il utilisera pour regrouper les données d'entrée. Cela est souvent appelé auto-organisation ou adaptation. L'apprentissage non supervisé est beaucoup plus difficile que l'apprentissage supervisé [10].

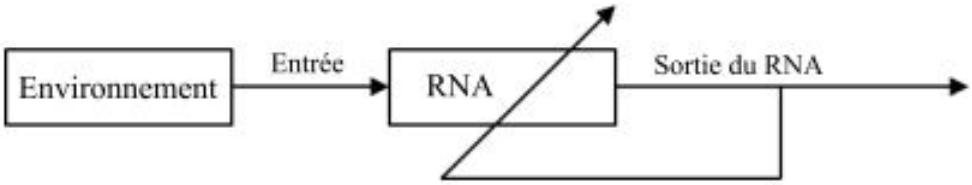


FIGURE 1.10 – Apprentissage non supervisé.

1.4 Réseaux neuro-floues

1.4.1 Système neuro-flou

Les systèmes neuro-flous résultent de la fusion des réseaux de neurones et de la logique floue, combinant ainsi les avantages de ces deux approches. Alors que la logique floue permet une spécification rapide des tâches à partir de connaissances symboliques

disponibles, l'ajustement précis et l'optimisation des paramètres restent souvent difficiles. En revanche, les modèles de réseaux de neurones les plus courants permettent un réglage du comportement du système par l'apprentissage, mais ne permettent pas toujours d'incorporer des connaissances a priori.

Ainsi, la caractéristique principale des systèmes neuro-flous réside dans leur capacité à intégrer à la fois des connaissances numériques et symboliques, exploitant ainsi les capacités d'apprentissage des réseaux de neurones et les capacités de raisonnement de la logique floue.

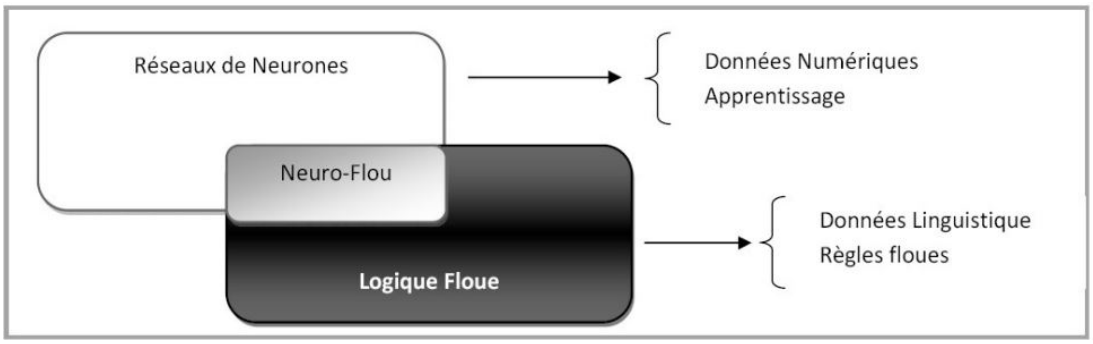


FIGURE 1.11 – Réseaux neuro-flous.

1.4.2 Type de combinaison neuro-flou

1.4.2.1 système flou neuronal

Les systèmes flous neuronaux cherchent à doter les systèmes flous de méthodes de réglage automatique typiques des réseaux de neuronnes, tout en préservant leur fonctionnalité de base (comme la fuzzification, la défuzzification, le moteur d'inférence et la base de logique floue). Ces systèmes exploitent les réseaux neuronaux pour renforcer le traitement numérique des ensembles flous, notamment pour clarifier les fonctions d'appartenance.Étant donné que les systèmes flous neuronaux sont fondamentalement des systèmes de logique floue, ils sont principalement appliqués dans des contextes de contrôle[11].

1.4.2.2 système neuronal flou

Les réseaux neuronaux flous préservent les propriétés fondamentales et les architectures des réseaux neuronaux classiques, mais ils introduisent une "fuzzification" sur certains de leurs composants. Dans ces réseaux, un neurone individuel peut devenir flou, et sa réponse à un signal d'activation de couche inférieure peut prendre la forme d'une relation floue plutôt que d'une fonction sigmoïde.Bien que l'architecture neuronale soit maintenue, ce qui varie sont certains types de poids synaptiques reliant les neurones de niveaux différents. Étant donné que les réseaux neuronals flous demeurent intrinsèquement des

réseaux neuronaux, ils sont principalement utilisés dans des applications de reconnaissance de formes[11].

1.4.2.3 neuro-flou hybride

Les techniques floues et les réseaux neuronaux interviennent conjointement dans des systèmes hybrides, chacun remplissant des rôles spécifiques. En capitalisant sur leurs avantages respectifs, ces techniques se combinent et se renforcent mutuellement pour atteindre un objectif commun. Les architectures des systèmes hybrides flou-neuraux sont spécifiquement conçues pour répondre aux besoins des applications, qu'il s'agisse de contrôle ou de reconnaissance de formes[11].

1.4.3 Architectures de réseaux neuro-flous

On trouve essentiellement deux types d'architectures connexionnistes des systèmes d'inférences floue ; l'une correspondant à un système avec J prémisses et J conclusions et l'autre avec J prémisses et k conclusions[12].

1.4.3.1 Architecture J prémisses J conclusions

Cette architecture de réseau, largement utilisée, repose sur un codage qui associe directement J prémisse à J conclusion. L'implémentation neuronale la plus synthétique de l'inférence des règles J prémisse à J conclusion repose sur cinq couches. La première couche est la couche d'entrée, suivie de la deuxième et de la troisième qui codent les ensembles représentant les prémisses. Ensuite, la quatrième couche code un lien entre un neurone "prémisse" et un neurone "conclusion" en effectuant une simple normalisation des sorties de la couche précédente. Enfin, le neurone de sortie linéaire fournit la sortie défuzzifiée. Parmi les systèmes mettant en œuvre une telle architecture, le plus connu est sans doute AFNIS, développé par Jang[12].

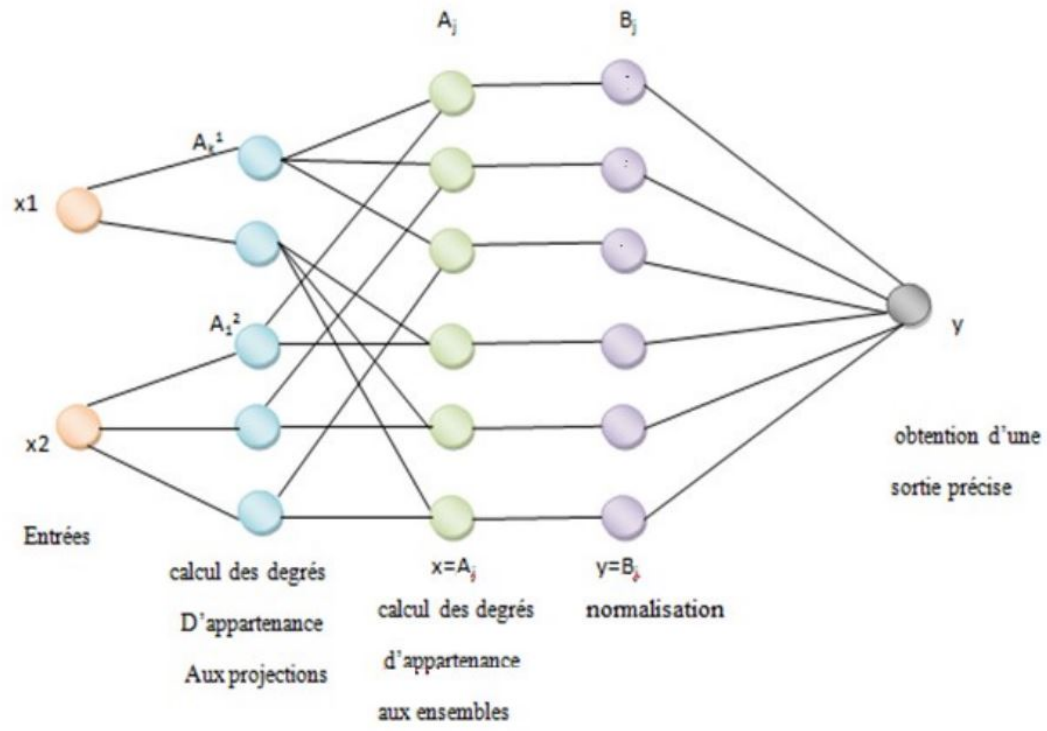


FIGURE 1.12 – Architecture J prémisses J conclusions

1.4.3.2 Architecture J prémisses K conclusions

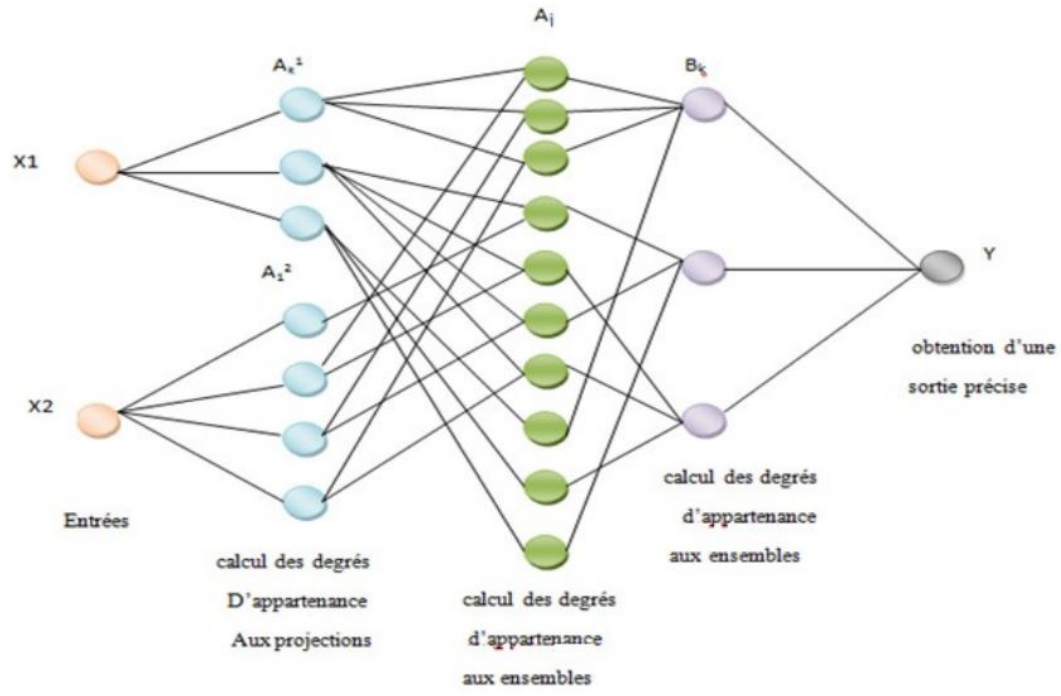


FIGURE 1.13 – Architecture J prémisses K conclusions

On s'intéressera dans ce chapitre a un réseau neuro-flou de type J prémisses J conclusions qui est le système d'inférence flou basé sur les réseaux de neurones ANFIS .

1.4.4 Structure ANFIS

Les systèmes d'inférence neuro-floue adaptatifs appartiennent à la famille des systèmes neuro-flous hybrides, qui combinent les avantages de la logique floue avec ceux des réseaux de neurones dans un seul réseau.La structure ANFIS a été proposée par Jang en 1993 .

ANFIS est une plateforme pour la logique neuro-floue adaptative. Cette plateforme représente le fonctionnement du système flou dans un réseau neuronal, qui comprend cinq couches dont les connexions ne sont pas pondérées, ou ont toutes un poids de 1. Les noeuds sont de deux types différents selon leur fonctionnalité : les nœuds carrés (adaptatifs) contiennent des paramètres, et les nœuds circulaires (fixes) n'ont pas de paramètres. Les règles floues peuvent être de type Takagi-Sugeno ou Mamdani, et la logique peut utiliser soit la règle de multiplication soit la règle de minimum. Les fonctions d'appartenance peuvent être de n'importe quel type (gaussien, trapézoïdal ou triangulaire).

Les modèles ANFIS sont largement utilisés dans diverses applications telles que le traitement du signal, le filtrage adaptatif et le contrôle des systèmes. Ils présentent généralement de bonnes performances, notamment dans les applications de contrôle[11].

Pour simplifier la compréhension et sans perte de généralitées, nous considérons un système à deux entrées x et y et une sortie z.

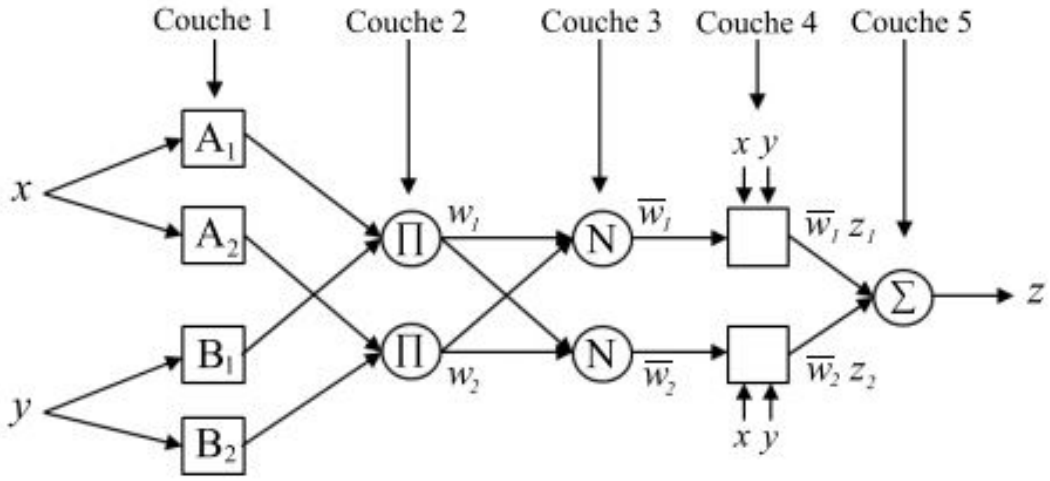


FIGURE 1.14 – Architecture d'un réseau ANFIS

Considérons aussi un modèle flou de type TSK de ce système, composé des deux règles suivantes[13] :

Règle 1 :

$$si\ x\ est\ A_1\ et\ y\ est\ B_1\ alors\ z_1 = p_1x + q_1y + r_1 \tag{1.8}$$

Règle 2 :

$$si\ x\ est\ A_2\ et\ y\ est\ B_2\ alors\ z_2 = p_2x + q_2y + r_2 \tag{1.9}$$

- Couche 1 : Fuzzification

La première couche représente les fonctions d'appartenance floues. Chaque nœud de cette couche est un nœud carré avec une fonction :

$$O_{1,j} = \mu_{A_j}(x), pour j = 1, 2 \tag{1.10}$$

$$O_{1,j} = \mu_{B_{j-2}}(y), pour j = 3, 4 \tag{1.11}$$

$O_{1,j}$ sont les degrés d'appartenance des variables x et y respectivement aux sous-ensembles flous A_j et B_j , qui peuvent être définis par différentes formes de fonctions d'appartenance.

- Couche 2 :Degré d'activation

Chaque nœud i de cette couche est un nœud circulaire appelé π qui engendre en sortie le produit de ses entrées. Ce produit représente le degré d'activation d'une règle :

$$O_{2,j} = \omega_j = \mu_{Aj}(x)\mu_{Bj}(y), j = 1, 2. \tag{1.12}$$

- Couche 3 :Normalisation

Chaque nœud de cette couche est un nœud circulaire fixe appelé N. La sortie du nœud est le degré d'activation normalisé de la j^{ime} règle :

$$O_{3,j} = \bar{\omega}_j = \frac{\omega_j}{\omega_1 + \omega_2},\ j = 1, 2 \tag{1.13}$$

- Couche 4 :Calcul des sorties des règles

La quatrième couche calcule les coefficients de l'équation du premier ordre d'une règle de type Takagi-Sugeno et cela pour chaque règle floue. Chaque nœud de cette couche est un nœud carré avec une fonction réalisant le calcul :

$$O_{4,j} = \bar{\omega}_j z_j = \bar{\omega_j}(p_jx + q_jy + r_j) \tag{1.14}$$

Où ω_j est la sortie de la couche 3 qui représente le degré d'activation normalisé de la règle et $(p_j, q_j\ et r_j)$ sont les paramètres de sortie ajustables de la règle j. Ces paramètres sont appelés paramètres de la conséquence.

- Couche 5 :Défuzzification

La cinquième couche qui représente la couche de sortie, calcule la sortie globale pondérée

du système. Le seul nœud de cette couche est un nœud circulaire qui effectue la somme des signaux provenant de la couche4, c'est-à -dire

$$O_{5,j} = z = \sum_j \bar{\omega}_j z_j = \frac{\sum_j \omega_j z_j}{\sum_j \omega_j} \tag{1.15}$$

Les paramètres de la conséquence ($p_j, q_j \, et \, r_j$) sont identifiés dans le processus d'apprentissage, en utilisant un algorithme basé sur des méthodes d'optimisation telle que la descente du gradient.

La généralisation du réseau à un système à r entrées ne présente aucune difficulté particulière. Le nombre de nœuds dans la couche 1 reste toujours égal au nombre total de termes linguistiques définis. L'apprentissage à partir d'un ensemble de données consiste à identifier les paramètres des prémisses et des conséquences, la structure du réseau étant fixée. L'algorithme d'apprentissage commence par construire un réseau initial, puis applique une méthode de rétro-propagation de l'erreur. Jang a proposé l'utilisation d'une règle hybride d'apprentissage qui combine un algorithme de descente de gradient avec une estimation par moindres carrés.

En ce qui concerne les paramètres des prémisses (couche 1), Jang a suggéré de n'appliquer une technique d'identification à partir des données que si l'ensemble de données est suffisamment grand. Dans ce cas, l'identification permet d'affiner les fonctions d'appartenance proposées par l'expert humain. Dans le cas contraire, il est préférable de conserver les fonctions d'appartenance proposées par l'expert car elles reflètent les connaissances de ce dernier. Jang a démontré l'équivalence entre les réseaux ANFIS et les réseaux neuronaux classiques avec des connexions pondérées[13].

1.5 Conclusion

Dans ce chapitre, nous avons présenté les concepts généraux de la logique floue, des réseaux de neurones et des réseaux neuro-flous. Nous avons d'abord décrit les notions de base de la logique floue ainsi que la structure de la commande floue. Ensuite, nous avons abordé le concept des réseaux de neurones artificiels et leur apprentissage . Finalement, nous avons présenté les structures de base des combinaisons de la logique flou avec les réseaux de neurones, plus particulièrement la structure ANFIS.

Chapitre 2

Description des circuits FPGA

2.1 Introduction

Les FPGAs sont devenus un pilier essentiel dans le domaine de l'ingénierie électronique et des systèmes embarqués. Grâce à leur capacité de reprogrammation et leur flexibilité, les FPGAs permettent aux ingénieurs de concevoir des systèmes numériques complexes et adaptables. Contrairement aux circuits intégrés traditionnels, conçus pour des tâches spécifiques, les FPGAs peuvent être reconfigurés pour exécuter une variété de fonctions. Cette flexibilité les rend particulièrement avantageux pour le prototypage rapide, les applications nécessitant des mises à jour fréquentes, et les environnements de recherche et développement.

Dans ce chapitre, nous présenterons d'abord une vue d'ensemble des architectures des FPGAs ,leurs avantages et leurs limitations, leurs applications dans divers domaines, ainsi que des technologies de programmation associées. Nous nous focaliserons ensuite sur les FPGAs du constructeur Altera, utilisés dans ce travail de thèse. Nous examinerons également les outils de conception et de développement proposés par Altera pour la mise en œuvre des conceptions.

2.2 Définition d'un FPGA

Les FPGA(Field Programmable Gate Array) sont des dispositifs électroniques reconfigurables qui permettent d'exécuter une variété d'applications matérielles. Depuis leur développement en 1985, ces circuits n'ont cessé de progresser grâce aux avancées technologiques, avec de nombreux fabricants tels que Xilinx, Altera, etc.

Ces circuits intégrés préfabriqués en silicium offrent la possibilité d'être programmés électriquement sur site pour former quasiment n'importe quel circuit numérique ou système. Leur conception inclut des blocs logiques et des interconnexions reconfigurables, leur permettant d'intégrer des applications complexes tout en bénéficiant des avancées

technologiques.

La caractéristique principale des FPGA est leur reconfigurabilité au cours du fonctionnement (reconfiguration dynamique), permettant une flexibilité sans limite et réduisant ainsi les délais de développement et de mise sur le marché. Ces dispositifs se déclinent en plusieurs types, tels que les FPGA basés sur la SRAM, sur le flash et sur l'antifusible, selon leur méthode de programmation[14, 15].

2.3 Domaines d'applications des FPGAs

De nos jours, les circuits FPGA sont devenus indispensables dans les systèmes numériques et sont utilisés dans de nombreux domaines d'application en raison des nombreux avantages qu'ils offrent. Parmi ces avantages, on peut notamment citer :

- L'amélioration croissante des performances en temps réel tout en réduisant les coûts et l'encombrement.
- L'augmentation des performances,ce mode d'implantation permet par exemple de réduire le temps d'exécution d'un algorithme afin de permettre au régulateur basé sur un FPGA d'atteindre le niveau de performance des régulateurs analogiques, sans leurs inconvénients tels que la dérive, le manque de souplesse et la compatibilité électromagnétique.
- Leur grande souplesse de programmation qui permet de les réutiliser à volonté pour cibler différents algorithmes en un temps très court.
- La rapidité et la facilité de reconfiguration d'un FPGA autant de fois que nécessaire pour implémenter les fonctionnalités désirées.

Grâce à tous ces avantages, les FPGAs sont aujourd'hui utilisés dans diverses applications nécessitant du traitement numérique telles que le traitement du signal et de l'image, le contrôle/commande des machines électriques, la mesure de vitesse, le contrôle des convertisseurs de puissance, la cryptographie, les équipements médicaux, les télécommunications, l'aéronautique, les transports, la bioinformatique, l'automobile, la robotique, ou encore plus généralement l'accélération de calculs scientifiques[16].

2.4 Technologies de programmation

Il existe plusieurs technologies de programmation qui ont été utilisées pour les FPGA. Chacune de ces technologies présente des caractéristiques différentes qui ont un effet significatif sur l'architecture programmable. Parmi les technologies les plus connues, on trouve les SRAM, les flashs, les anti-fusibles, les EPROM et les EEPROM[14].

2.4.1 Technologie de programmation SRAM

Les cellules de mémoire statique constituent les éléments de base utilisés dans les FPGA basés sur la SRAM. La plupart des fabricants commerciaux optent pour la technologie de programmation basée sur la mémoire statique (SRAM) dans leurs appareils en raison de sa reprogrammabilité et de l'utilisation de la technologie de processus CMOS standard. Cela se traduit par une intégration accrue, une vitesse plus élevée et une consommation d'énergie dynamique moindre pour les nouveaux processus présentant une géométrie plus petite. Cependant, cette approche présente quelques inconvénients. Par exemple, la surface utilisée car les SRAMs nécessitent beaucoup de transistors, ce qui rend cette technologie coûteuse en termes de surface comparativement à d'autres technologies de programmation. De plus, les cellules SRAM étant volatiles, des dispositifs externes sont nécessaires pour stocker de manière permanente les données de configuration, ce qui ajoute au coût et au surdimensionnement des FPGA basés sur la SRAM[14].

2.4.2 Technologie de programmation flash

Une alternative à la technologie de programmation basée sur la SRAM est l'utilisation de la technologie de programmation flash . La technologie de programmation flash présente plusieurs avantages. Par exemple, elle est de nature non volatile, ce qui signifie que les données ne sont pas perdues lorsque l'alimentation est coupée. De plus, elle est plus efficace en termes d'utilisation de l'espace par rapport à la technologie basée sur la SRAM. Cependant, cette technologie présente également des inconvénients. Contrairement à la technologie basée sur la SRAM, les dispositifs basés sur le flash ne peuvent pas être reconfigurés ou reprogrammés un nombre infini de fois. De plus, la technologie basée sur le flash nécessite l'utilisation d'un processus CMOS non standard[14].

2.4.3 Technologie de programmation anti-fusible

Cette technologie repose sur des structures qui, dans des conditions normales, présentent une résistance très élevée (ne conduisent pas le courant) mais peuvent être programmées "brûlés" pour former un lien de faible résistance. Contrairement aux techniques de programmation utilisant les SRAM ou les portes flottantes, ce lien est permanent. L'élément programmable, appelé anti-fusible,une fois qu'il est activé, il ne peut plus être restauré à son état d'origine. Par conséquent, les FPGA basés sur des anti-fusibles ne peuvent être programmés qu'une seule fois[17].

La technologie de programmation à antifusibles présente plusieurs avantages significatifs. Tout d'abord, elle nécessite moins d'espace physique. De plus, les anti-fusibles offrent une non-volatilité, ce qui signifie que le dispositif fonctionne immédiatement après la programmation, réduisant ainsi les coûts du système en éliminant le besoin de mémoire

supplémentaire pour stocker les informations de programmation, néanmoins, elle présente un grand inconvénient, car le fait que la programmation des anti-fusibles soit définitive les rend inappropriés pour les applications nécessitant des modifications de configuration ultérieures[17].

2.4.4 Technologie de programmation EPROM

Cette technologie utilise des transistors de type EPROM ((Erasable Programmable Read-Only Memory)(mémoire morte programmable effaçable).Tout comme avec la technologie de programmation SRAM, un avantage majeur de la technologie EPROM est sa reprogrammabilité.Cependant, son principal inconvénient réside dans le processus de reconfiguration qui nécessite une source de lumière ultraviolette[18].

2.4.5 Technologie de programmation EEPROM

La technologie de programmation basée sur l'EEPROM(Electrically Erasable Programmable Read-Only Memory) partage des similitudes avec l'approche de l'EPROM, à la différence que l'effacement de la charge de grille peut être réalisé électriquement, sur le circuit, sans nécessiter de lumière UV. Cette caractéristique confère un avantage supplémentaire en permettant une reprogrammabilité aisée, ce qui peut s'avérer très utile dans diverses applications, notamment pour les mises à jour matérielles d'équipements dans des endroits éloignés. Cependant, il est important de noter qu'une cellule EEPROM occupe approximativement deux fois plus d'espace qu'une cellule EPROM[18].

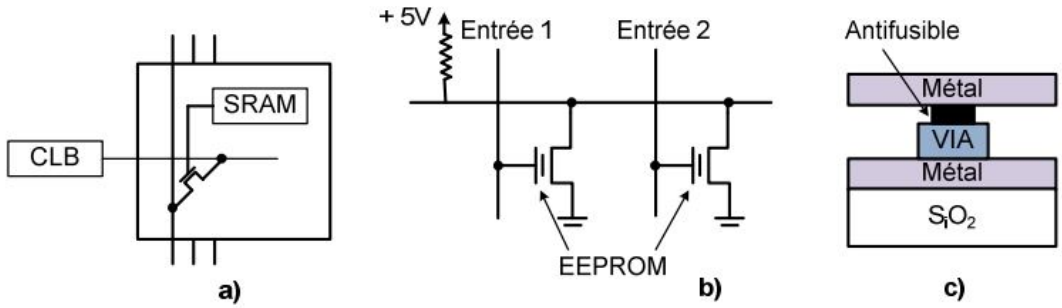


FIGURE 2.1 – Technologie de programmation des FPGA : a) SRAM ; b) EPROM ; c) Antifusible

2.5 Architecture générale d'un circuit FPGA

Les FPGA sont constitués de trois éléments principaux :Les blocs logiques programmables (CLB - Configurable Logic Bloc),Les blocs d'entrées/sorties programmables (IOB - Input Output Bloc) et des routages programmables (interconnexions)[19].

Le marché des FPGAs est dominé par plusieurs fabricant ,cependant l'architecture interne des FPGAs diffère d'un fabricant à l'autre, ce qui impacte la manière dont la configuration matérielle est réalisée. Dans la suite de notre étude, nous nous concentrerons sur l'architecture des FPGAs Altera, qui se compose de deux couches distinctes :la première est la couche du circuit configurable, et la deuxième est une couche de réseau mémoire SRAM[20].

- La première couche (circuit configurable) :

Cette couche est composée d'une matrice de CLBs capables d'implémenter à la fois des fonctions séquentielles et combinatoires et des blocs IOBs qui sont situés autour de ces blocs logiques configurables,ils gèrent les interfaces avec les modules externes[20].

- La seconde couche (réseau mémoire SRAM) :

cette deuxième couche est un réseau de mémoire SRAM qui permet la programmation du circuit.Cette programmation consiste à appliquer les tensions appropriées sur la grille de certains transistors à effet de champ pour interconnecter les éléments des CLBs et des IOBs, permettant ainsi de réaliser les fonctions requises et d'assurer la propagation des signaux.Les blocs SRAM sont volatils[14], ce qui signifie que la configuration est effacée à chaque redémarrage du dispositif. La configuration stockée dans la mémoire SRAM est donc rechargée à chaque mise sous tension depuis une ROM externe[20].

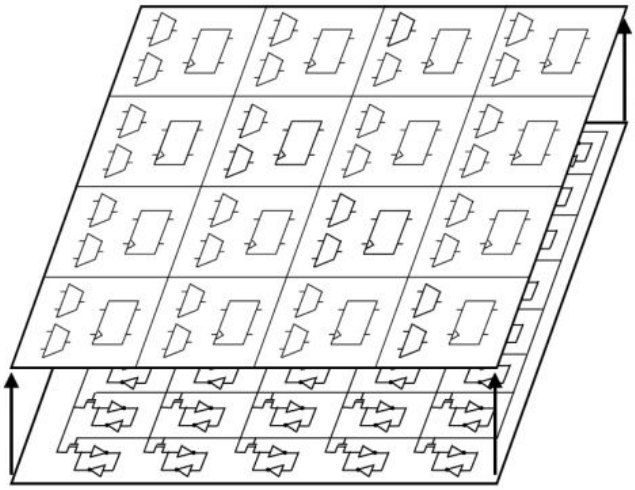


FIGURE 2.2 – Architecture d'un FPGA

2.6 Architecture interne

La structure interne des FPGAs diffère d'un constructeur à un autre. Elle se compose principalement de trois blocs principaux : les blocs logiques configurables, les blocs d'entrées/sorties, et les ressources de communication[21].Pour optimiser les ressources FPGA, des blocs DSP câblés (blocs arithmétiques) comprenant des multiplieurs, des ad-

ditionneurs et des accumulateurs,des gestionnaires d'horloge numérique (DCM) et des processeurs sont inclus. Dans le même but, des blocs de mémoire (RAM, ROM, Flash RAM) sont également intégrés.

Les dispositifs FPGA actuels comprennent également des blocs de communication qui se composent généralement de tampons de transmission et de réception. Divers protocoles de communication sont pris en charge, notamment USB, Ethernet, CAN, PCI, SPI et les protocoles I2C[22].

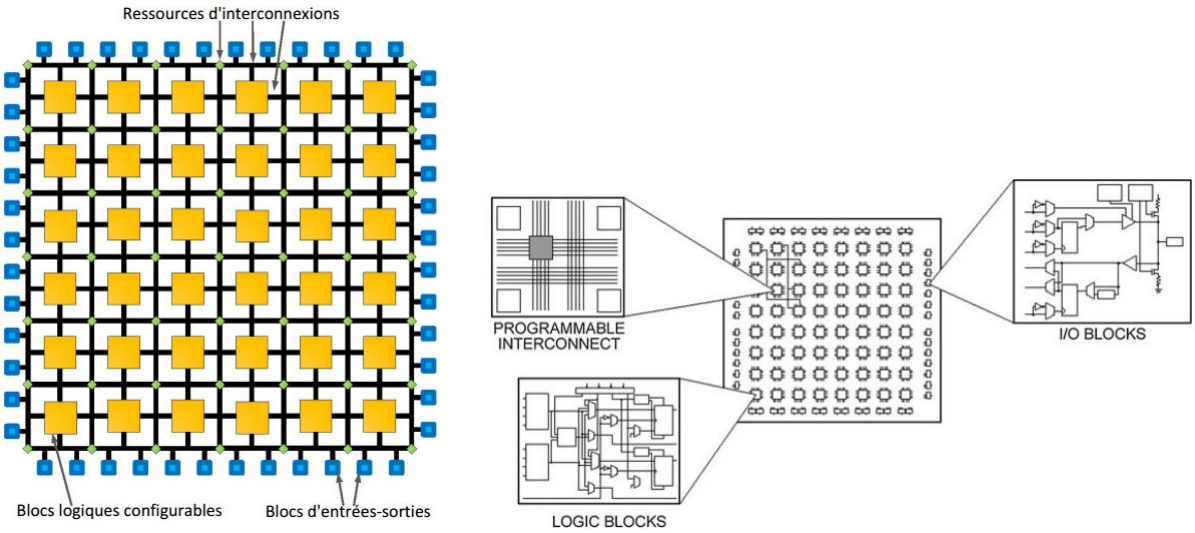


FIGURE 2.3 – Architecture interne d'un FPGA

2.6.1 Blocs élémentaires

2.6.1.1 Blocs logiques configurables (CLBs)

Un bloc logique configurable (CLB) est l'élément de base d'un FPGA, capable de réaliser des fonctions logiques arbitraires[19].Chaque bloc logique configurable est constitué d'éléments logiques de base (BLE), qui sont utilisés pour mettre en œuvre la partie logique du circuit. Ce dernier comprend un ensemble de tables de consultation (Look-Up Table : LUT) et une bascule D pour mettre en œuvre les fonctions de base à l'aide du bloc mémoire (SRAM). Il est suivi d'un multiplexeur utilisé pour acheminer la logique à l'intérieur du bloc et vers des ressources externes. Les multiplexeurs permettent également la sélection de la polarité ainsi que la sélection des entrées de réinitialisation et de suppression.Dans les dispositifs FPGA de la famille Xilinx, une unité CLB contient une paire d'éléments appelés SLICEs. Ces deux SLICEs n'ont pas de connexions directes entre elles, et chaque tranche est organisée en colonne.Chaque SLICE est composé d'un nombre donné de LUTs selon la famille du FPGA, combiné avec des bascules D et des multiplexeurs[15].

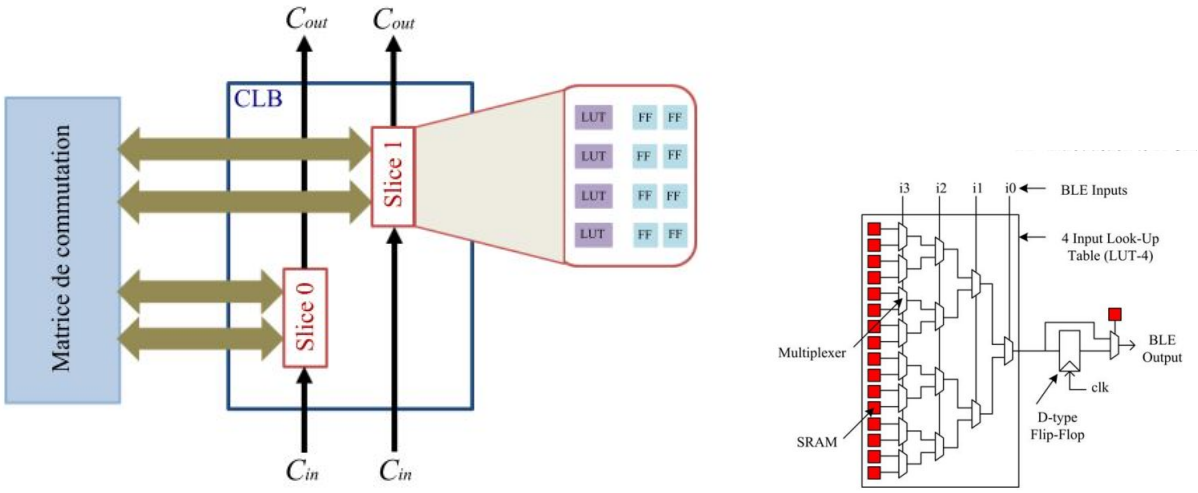


FIGURE 2.4 – Architecture d'un CLB et d'un BLE.

2.6.1.2 Blocs d'entrées/sorties (IOBs)

Les blocs d'entrée-sortie configurables facilitent l'interface entre l'architecture interne et l'environnement externe. Chaque IOB a sa propre mémoire de configuration, stockant les standards de tension et la direction des ports, et est réparti le long de la périphérie du FPGA. Chaque IOB contrôle une broche du composant et peut être configuré en entrée, en sortie, en bidirectionnel ou laissé inutilisé (haute impédance)[20].

2.6.1.3 Interconnexions

Les ressources d'interconnexion utilise des pistes horizontales et verticales avec des boîtes de commutation pour relier les blocs logiques, occupant une grande partie de la surface de l'FPGA, généralement entre 80 et 90 %,.La flexibilité d'un FPGA dépend en grande partie de son réseau d'interconnexions[23].

Les interconnexions jouent un rôle important en connectant de manière efficace les blocs logiques et les entrées/sorties dans un circuit afin d'optimiser le taux d'utilisation. Pour atteindre cet objectif, trois types d'interconnexions adaptées à différentes longueurs et destinations des liaisons sont proposés[20] :

- Les interconnexions directes :permettent d'établir des liaisons entre les blocs configurables (CLB) et les blocs d'entrée/sortie (IOB) dans un FPGA. Elles autorisent également la connexion directe de certaines entrées d'un CLB aux sorties d'un autre.
- Les longues lignes : elles sont des segments métallisés s'étendant sur toute la longueur et la largeur du FPGA. Elles sont conçues pour transmettre les signaux entre les différents éléments avec un minimum de retard, assurant ainsi un synchronisme optimal. En outre, ces longues lignes permettent de limiter le nombre de points d'interconnexion.

- Les matrices d'interconnexion :agissent comme des commutateurs situés à chaque intersection des longues lignes. Leur fonction est de connecter les longues lignes entre elles selon diverses configurations. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre CLB sur le FPGA afin de faciliter la communication des signaux. Pour prévenir l'affaiblissement des signaux traversant les longues lignes, des buffers sont intégrés dans chaque matrice d'interconnexion.

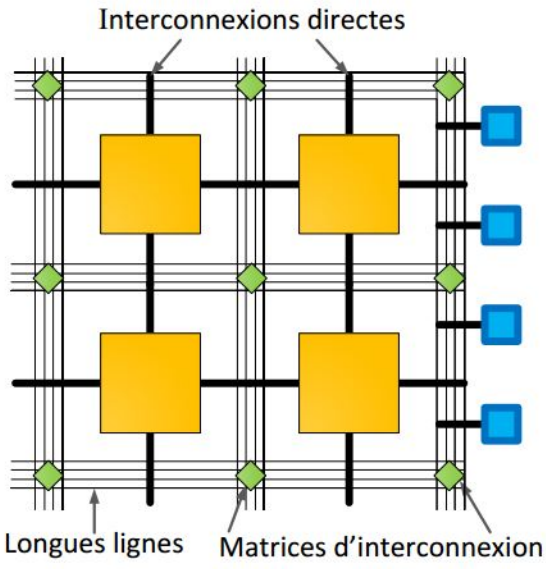


FIGURE 2.5 – Interconnexion interne d'un FPGA

2.6.2 Blocs supplémentaires

2.6.2.1 Blocs mémoires

Ces blocs comprennent des mémoires ROM, RAM et Flash RAM.Ce sont des composants essentiels des FPGAs. Ils constituent une mémoire embarquée sur la puce FPGA et servent à stocker des ensembles de données nécessaires au fonctionnement du circuit implémenté[20].

2.6.2.2 Digital Signal Processing (DSP)

Les blocs DSP, ou blocs de traitement du signal numérique, sont des composants spécialisés présents sur les FPGAs. Ils comprenent des multiplicateurs, des additionneurs et des accumulateurs.Ils permettent d'implémenter des fonctions de traitement du signal plus complexes et plus performantes que ce qui est possible avec les CLBs standard[20].

2.6.2.3 Processeurs embarqués

L'intégration de processeurs embarqués est l'une des innovations majeures des FPGAs modernes. Ces processeurs offrent de nombreux avantages pour la conception de circuits numériques complexes.. Cette technologie permet de simplifier la conception, d'améliorer les performances et de réduire la consommation d'énergie, ouvrant la voie à de nouvelles applications innovantes[20].

2.6.2.4 Gestionnaire d'horloge numérique (DCM)

Le gestionnaire d'horloge numérique (DCM) est un composant intégré aux FPGAs qui permet de générer des horloges internes à partir d'une horloge de référence unique. Cette horloge de référence peut être une horloge externe fournie par un oscillateur ou une horloge interne du FPGA.

Le DCM utilise un processus de bouclage à phase verrouillée (PLL(Phase-Locked Loops)) pour générer des horloges à partir de l'horloge de référence. La PLL compare la phase de l'horloge générée à la phase de l'horloge de référence et ajuste la fréquence de l'horloge générée pour les synchroniser[20].

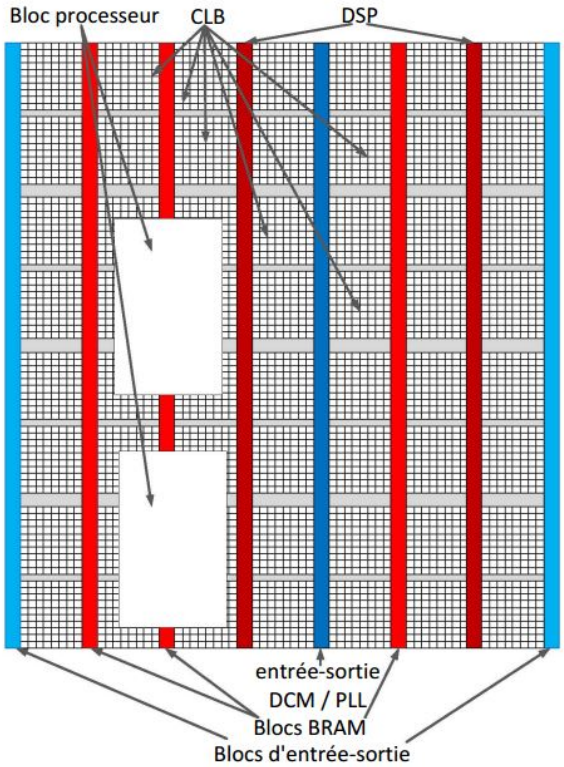


FIGURE 2.6 – Ressources d'un circuit FPGA.

2.7 FPGAs de la société Altera

Altera Corporation, fondée en 1983 par Rodney Smith et Alex Dunlop, s'est établie comme un pionnier des solutions de logique programmable. Son objectif initial était de développer des circuits logiques programmables (FPGA) plus performants et plus abordables que ceux disponibles à l'époque. Au fil des années, l'entreprise a consolidé sa position en offrant une gamme complète de produits destinés à permettre aux entreprises de systèmes et de semi-conducteurs d'innover rapidement et de manière rentable, tout en se différenciant sur leurs marchés respectifs[24].

Les offres d'Altera comprennent une variété de solutions telles que des FPGA, des SoC intégrant des systèmes de processeurs (comme le NIOS II), des CPLD associés à des outils logiciels, des propriétés intellectuelles, des processeurs intégrés et un support client dédié. Cette gamme diversifiée de produits vise à fournir des solutions programmables à haute valeur ajoutée, adaptées aux besoins spécifiques des clients[25].

Avant son acquisition par Intel en 2015, Altera a joué un rôle majeur dans l'évolution de la technologie FPGA, contribuant ainsi à façonner le paysage de l'industrie des semi-conducteurs. Grâce à son engagement envers l'innovation et à son expertise dans le domaine de la logique programmable, Altera a permis aux entreprises de réaliser des avancées significatives dans un large éventail d'applications, renforçant ainsi sa réputation en tant que leader du secteur.

L'une des contributions les plus remarquables d'Altera a été le développement de l'architecture FPGA Cyclone®. Cette avancée majeure a permis de réduire les coûts des FPGA, les rendant ainsi accessibles à un public beaucoup plus large. En conséquence, la famille Cyclone est rapidement devenue l'une des familles de FPGA les plus répandues et appréciées dans le monde entier[26].

En parallèle de la famille Cyclone, Altera a également conçu d'autres familles de FPGA, telles que Stratix®, Arria® et Apex™. Ces différentes gammes ont été développées pour répondre à une variété de besoins, offrant des niveaux de performances et de fonctionnalités plus élevés pour satisfaire aux exigences d'applications toujours plus complexes[26].

Pour accompagner ses produits, Altera a mis en place une suite complète d'outils de développement visant à faciliter la conception et la programmation de ses FPGA. Parmi ces outils, Quartus® Prime a notamment connu un grand succès et est devenu l'un des logiciels de développement FPGA les plus utilisés dans le monde[26].

L'acquisition d'Altera par Intel en 2015 a marqué la fin de son existence en tant que société indépendante. Cependant, les FPGA développés par Altera continuent d'être maintenus et commercialisés par Intel sous la marque Intel Programmable Solutions, garantissant ainsi la continuité et l'évolution de ces technologies au service des utilisateurs à travers le monde[26].

2.8 Présentation de la carte Altera DE2-115 cyclone 4

Le FPGA Cyclone d'Altera est conçu dès le départ pour être économique. Avec une mémoire intégrée, des interfaces mémoire externes et des circuits de gestion d'horloge, les FPGA Cyclone constituent une solution optimale pour les applications à volume élevé et sensible aux coûts. La famille de FPGA Cyclone est basée sur un processus SRAM en cuivre à 1,5 V et 0,13 µm, avec des densités allant jusqu'à 20 060 éléments logiques (LE) et jusqu'à 288 kilo-octets de RAM. Avec des fonctionnalités telles que des boucles à verrouillage de phase (PLL) pour l'horloge et une interface DDR (double débit de données) dédiée pour répondre aux exigences de mémoire DDR SDRAM et FCRAM à cycle rapide, les dispositifs Cyclone sont une solution rentable pour les applications de chemin de données. Les dispositifs Cyclone prennent en charge différents standards d'E/S, y compris le LVDS à des débits de données allant jusqu'à 640 mégabits par seconde (Mbps), ainsi que les interfaces PCI 64 bits et 32 bits à 66 et 33 MHz, pour l'interfaçage avec et la prise en charge des dispositifs ASSP et ASIC. Altera propose également de nouveaux dispositifs de configuration série à faible coût pour configurer les dispositifs Cyclone[27].

La carte sur laquelle se déroule ce mémoire est une ALTERA "DE2 Development and Education Board". Cette carte a été conçue afin de permettre l'apprentissage d'environnements embarqués, plus particulièrement au niveau des FPGAs.

La carte DE2-115 dispose de nombreuses fonctionnalités permettant aux utilisateurs de mettre en œuvre une large gamme de circuits conçus, des circuits simples aux projets multimédias variés. Voici les éléments matériels fournis sur la carte DE2-115 [28] :

- FPGA : Altera Cyclone IV EP4CE115F29C7N
  - ✦ Nombre de logiques : 114,480 éléments logiques
  - ✦ Blocs de mémoire intégrée : 3,888 Kbits
  - ✦ Multiplicateurs intégrés : 266 (18x18)
  - ✦ PLL (Phase-Locked Loops) : 4
- Mémoire
  - ✦ SDRAM : 128 MB
  - ✦ Flash : 8 MB
  - ✦ EEPROM : 2 KB
  - ✦ SD Card Slot : Support pour carte SD
- Interfaces et connecteurs
  - ✦ Ports USB : 2 ports USB (1 hôte, 1 périphérique)
  - ✦ Ethernet : 10/100 Ethernet PHY
  - ✦ VGA : Connecteur VGA pour sortie vidéo

- ✦ Audio : Codec audio avec entrée et sortie audio
- ✦ RS-232 : Port série pour communication
- ✦ GPIO : 40 broches GPIO (General Purpose Input/Output)
- ✦ Switches et LEDs : 18 switches, 4 boutons poussoirs, 27 LEDs

• Périphériques supplémentaires

- ✦ LCD : Écran LCD 16x2 caractères
- ✦ IrDA : Récepteur et émetteur infrarouge
- ✦ PS/2 : Port PS/2 pour clavier et souris
- ✦ Connecteurs d'extension : HSMC (High-Speed Mezzanine Card) pour modules d'extension

En plus de ces fonctionnalités matérielles, la carte DE2-115 dispose d'une prise en charge logicielle pour les interfaces d'E/S standard et d'un panneau de contrôle permettant d'accéder à divers composants. De plus, des logiciels sont fournis pour prendre en charge un certain nombre de démonstrations illustrant les capacités avancées de la carte DE2-115.

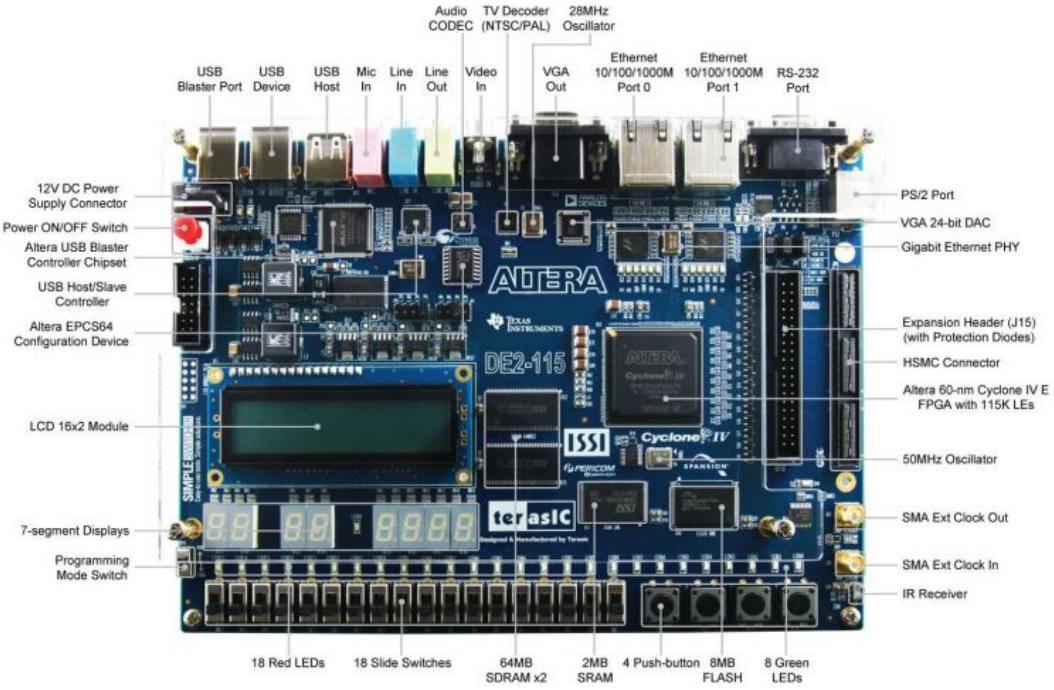


FIGURE 2.7 – La carte DE2-115

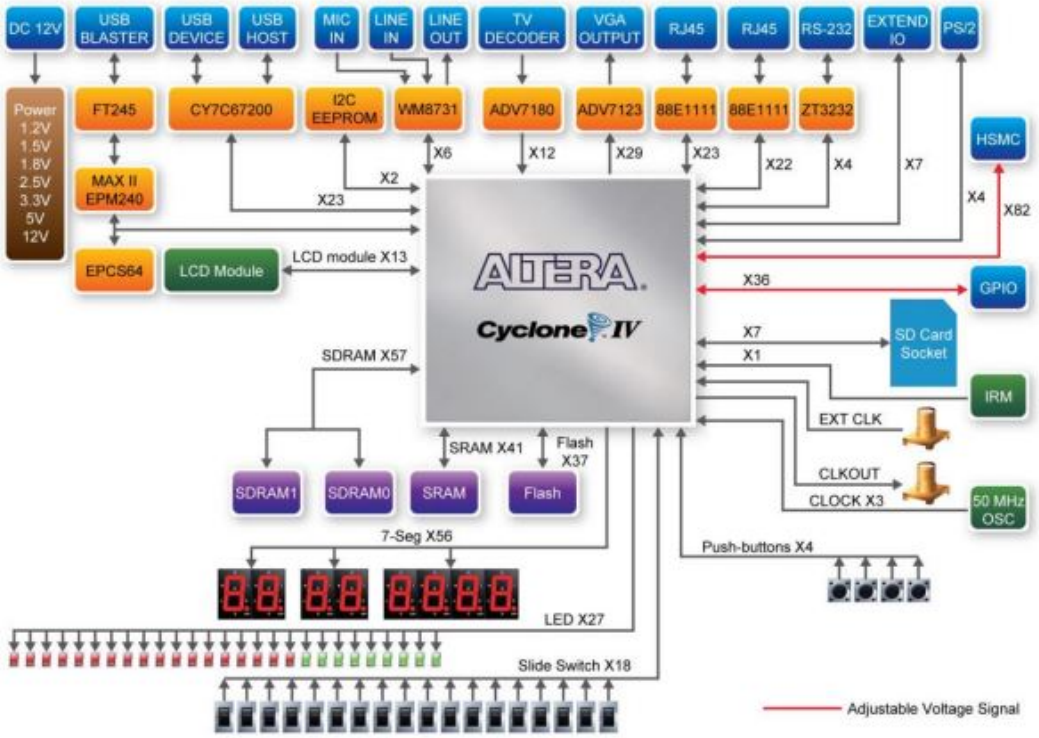


FIGURE 2.8 – Le bloc diagramme de la carte DE2-115 cyclone 4

2.9 Type de données

Dans les circuits numériques, les nombres sont stockés dans des mots binaires. Un mot binaire est une séquence de bits (1 et 0) de longueur fixe. La façon dont les composants matériels interprètent cette séquence de bits est définie par le type de données .

Les nombres binaires sont présentés sous forme de deux types de données : virgule fixe et virgule flottante.

2.9.1 Virgule flottante

Le système des nombres en virgule flottante est un standard de codage des nombres réels largement utilisé lorsqu'une grande précision est nécessaire. Il s'inspire de la notation scientifique et représente un nombre x comme suit[29] :

$$x = (-1)^S M B^E \tag{2.1}$$

S représente le signe de$x$, M est la mantisse ,Best la base et E est l'exposant.

Signe : Le bit de signe indique si le nombre est positif (0) ou négatif (1).

Mantisse : La mantisse est la partie fractionnaire du nombre, codée sur un nombre fixe de bits. Elle détermine la précision du nombre représenté.

précision	largeur totale	signe	exposant	mantisse	biais
simple	32	1	8	23	127
double	64	1	11	52	1023

TABLE 2.1 – Tailles des différentes composantes des deux principaux formats spécifiés par la norme IEEE754, et valeurs du biais pour les précisions simple et double

Exposant : L'exposant est la partie entière du nombre, codée sur un nombre fixe de bits. Il représente un facteur d'échelle qui peut être positif ou négatif, permettant de représenter une large plage de valeurs.

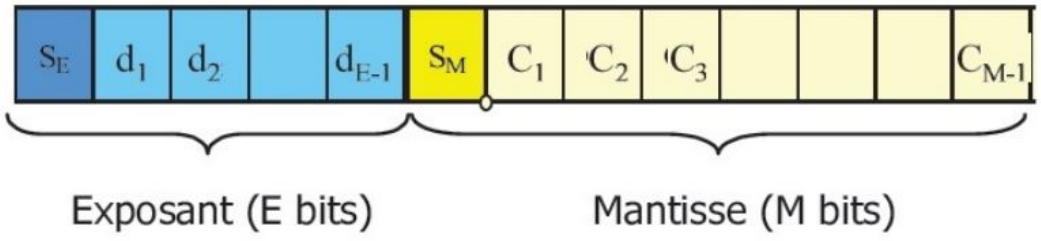


FIGURE 2.9 – Représentation des données en virgule flottante

Comme le nombre de bits accordés à la mantisse et à l'exposant peut prendre un grand nombre de valeurs possibles, un format virgule flottante normalisé a été introduit pour garantir la précision et la cohérence des calculs, un format virgule flottante normalisé a été introduit. La norme IEEE 745-2008 est utilisée dans la plupart des systèmes numériques d'aujourd'hui. Elle spécifie le format virgule flottante, les modes d'arrondis et le traitement des exceptions.L'objectif de cette norme est d'améliorer la compatibilité entre les diverses architectures[29].

La valeur d'un nombre x représenté selon le format virgule flottante de la norme IEEE 745 est calculée :

$$x = (-1)^S M 2^{E-biais} \tag{2.2}$$

Avec $biais = 2^{Ne-1} - 1$.Et La mantisse est normalisée pour représenter une valeur dans l'intervalle$[1, 2]$. Par conséquent, la valeur de son premier bit est fixée à 1 et devient implicite. La valeur de l'exposant est codée comme un nombre non signé.

La norme IEEE 754 définit les deux formats indiqués dans le tableau suivant[30] :

2.9.2 Virgule fixe

Les données en virgule fixe sont un format de codage des nombres réels où la position de la virgule est fixe et prédéfinie. Ce format se distingue de la virgule flottante où la virgule peut se déplacer durant les calculs.Les données en virgule fixe sont composées

d'une partie fractionnaire et d'une partie entières et un bit de signe SR, où 0 désigne un nombre positif tandis que 1 désigne un nombre négatif[31].

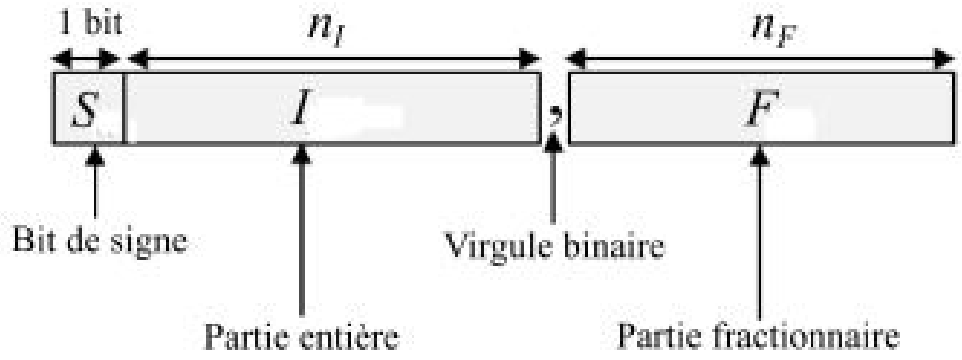


FIGURE 2.10 – Représentation d'un nombre réel en virgule fixe.

Le Nombre réel x peut s'écrire [5] :

$$R = I_x F_x * 2^{n_F} \tag{2.3}$$

2.10 Outils de conception et de développement FPGA de ALTERA

2.10.1 Langages de description matérielle

2.10.1.1 VERILOG : Historique et description

En 1984, la société Gateway Design Automation Inc a introduit le langage Verilog en tant que langage de description matériel. Le langage original, appelé HiLo, présente de fortes similitudes avec le langage C, ce qui a immédiatement suscité l'intérêt de la communauté informatique[32].

À l'origine, Verilog était un langage propriétaire et n'était pas standardisé. En 1989, Cadence a acquis Gateway, devenant ainsi propriétaire du langage Verilog ainsi que de son mécanisme d'interface de langage de programmation (PLI). En 1990, Verilog a été ouvert au domaine public, permettant à quiconque de développer son propre simulateur Verilog et de devenir un concurrent potentiel de Cadence. Cette ouverture a conduit à une augmentation de la popularité de Verilog, en faisant le langage de description matériel le plus répandu et utilisé.

Pour accompagner cette croissance, le groupe de travail Open Verilog International (OVI) a été formé pour standardiser Verilog et son PLI. En 1993, ce groupe a créé un comité de travail pour standardiser Verilog sous l'égide de l'IEEE, aboutissant à la norme 1364-1995, qui combine la syntaxe de Verilog et son PLI.

Les équipes de conception ont trouvé plusieurs avantages à utiliser Verilog, notamment sa similitude avec le langage C, ce qui facilite la prise en main pour les utilisateurs habitués à ce langage. De plus, l'assignation des signaux est moins contraignante que dans VHDL. Cependant, l'absence de prise en charge des types de signaux tels que les bus ou les signaux signés peut être une source de pièges. Verilog ne propose pas non plus de concept de package ou de bibliothèques, ce qui signifie que chaque fonction et procédure d'un module doit être définie dans ce module.

Verilog permet également la description de systèmes mixtes, appelée Verilog-AMS HDL, qui combine la partie numérique du Verilog 1364-1995 avec le Verilog-A pour les systèmes analogiques[33].

2.10.1.2 VHDL : Historique et description

VHDL (VHSIC Hardware Description Language) est un langage de description matériel utilisé pour décrire le comportement et la structure des systèmes numériques. C'est un langage de description matériel polyvalent qui peut être utilisé pour décrire et simuler le fonctionnement d'une grande variété de systèmes numériques, allant de quelques portes logiques à une interconnexion de nombreux circuits intégrés complexes. VHDL a été développé à l'origine dans les années 80 en réponse à un appel d'offres du département de la défense américain pour un langage de description matérielle unique. Lorsque VHDL a été développé, le principal objectif était d'avoir un mécanisme pour décrire et documenter le matériel de manière non ambiguë. La synthèse matérielle à partir de descriptions de haut niveau n'était pas l'un des objectifs initiaux. Le langage VHDL est depuis devenu une norme de l'IEEE (Institute of Electronic and Electrical Engineers) et il est largement utilisé dans l'industrie[33].

Le VHDL est considéré comme un langage de description de haut niveau, offrant une abstraction maximale des composants et des structures. Cela permet une portabilité des descriptions VHDL, offrant la possibilité de cibler différents composants ou structures avec différents outils. Son utilisation est préférable pour décrire des fonctions numériques complexes, offrant une alternative plus claire et plus efficace aux schémas structurels traditionnels[33].

2.10.2 Logiciel de conception et de développement FPGA de ALTERA

Les circuits FPGA ont longtemps été perçus comme des dispositifs difficiles à programmer en raison de leur nature matérielle complexe. Pour simplifier la programmation de ces dispositifs, les outils de synthèse ont connu un développement important. Les fabricants de FPGA ont ainsi lancé plusieurs outils dans ce but.

2.10.2.1 Quartus II

Quartus II est un environnement de développement intégré (IDE) développé par Altera pour la conception, la simulation et l'implémentation de circuits logiques programmables, tels que les FPGA . Ce logiciel offre une solution complète pour gérer toutes les étapes du flot de conception, du schéma initial jusqu'à la programmation du matériel[34].

Quartus II permet aux utilisateurs de créer des designs en utilisant différentes méthodes de saisie :

Saisie graphique : Les utilisateurs peuvent dessiner des schémas en utilisant des symboles et des blocs logiques.

Description HDL (Hardware Description Language) : Les conceptions peuvent être décrites textuellement en utilisant VHDL, Verilog ou AHDL (Altera Hardware Description Language).

Le logiciel inclut des outils de simulation qui permettent de vérifier le comportement fonctionnel et temporel des designs avant leur implémentation physique. Cela aide à détecter et corriger les erreurs dès les premières étapes du développement.

2.10.2.2 Modelsim

Moldelsim développé par Mentor Graphics, est un environnement de simulation de circuits électroniques largement utilisé dans l'industrie. Il permet aux ingénieurs de modéliser, simuler et déboguer des circuits numériques complexes, en particulier ceux conçus à l'aide de langages de description de matériel (HDL) comme VHDL et Verilog.

ModelSim est un outil puissant et polyvalent qui s'avère indispensable pour la conception et la vérification de circuits numériques complexes. Sa large gamme de fonctionnalités, ses capacités de débogage avancées et ses options de vérification formelle en font un choix privilégié pour les ingénieurs et les entreprises du monde entier.

2.10.2.3 HDL Coder de l'environnement MATLAB-SIMULINK

HDL Coder est un outil développé par MathWorks pour générer du code HDL synthétisable (VHDL et Verilog) à partir de fonctions MATLAB et de modèles Simulink. Il propose un flux de travail qui analyse un modèle MATLAB/Simulink et convertit automatiquement le système du point flottant au point fixe, offrant un niveau d'abstraction élevé. Cela permet aux utilisateurs de se concentrer sur le développement d'algorithmes et de modèles sans se soucier des complexités de bas niveau associées à la conception HDL.

Le flux de travail de HDL Coder inclut des outils pour la vérification du code HDL, permettant de tester le code généré en parallèle avec le modèle MATLAB/Simulink original. Cela facilite l'identification et la correction des erreurs potentielles dans le code HDL généré.

L'optimisation du code HDL dans ce flux de travail permet de spécifier le dispositif FPGA cible, offrant ainsi un contrôle important sur l'implémentation. Cela permet de mettre en évidence les chemins critiques, de contrôler l'architecture HDL et de générer des estimations de l'utilisation des ressources matérielles[35].

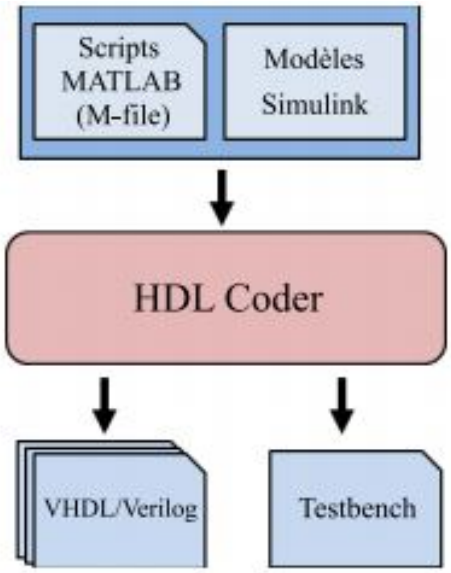


FIGURE 2.11 – Conception avec HDL Coder

2.11 Conclusion

Ce chapitre a été une description générale des circuits FPGA, de leurs technologies de programmation ainsi que de leurs architectures internes. Nous avons présenté les FPGAs du constructeur Altéra, plus particulièrement ceux de la famille DE2-115, utilisés dans ce travail de thèse.

La carte FPGA DE2-115 se distingue par sa flexibilité, sa puissance et sa polyvalence, ce qui en fait une plateforme idéale pour une variété d'applications allant de l'éducation et de la recherche aux projets industriels complexes. Équipée d'un FPGA Cyclone IV, de multiples interfaces de communication, de capacités de mémoire étendues et de riches interfaces utilisateur, elle offre aux étudiants et aux ingénieurs un outil robuste pour l'apprentissage et le développement de systèmes numériques avancés. Les connecteurs d'extension HSMC et GPIO permettent une intégration facile de modules supplémentaires, augmentant ainsi les possibilités d'application.

L'utilisation des FPGA présente des avantages considérables en termes de vitesse de traitement, de parallélisme et de reconfigurabilité, ce qui en fait une plateforme idéale pour l'implémentation en temps réel de régulateurs neuro-flous.

Chapitre 3

Implémentation d'un régulateur ANFIS sur FPGA

3.1 Introduction

Dans ce chapitre, nous allons présenter les techniques utilisées pour concevoir et implémenter un régulateur ANFIS sur la carte DE2-115 en utilisant l'environnement Simulink de MATLAB. Ce régulateur sera développé afin de commander la position d'un pendule inversé.Le schéma de commande ANFIS en utilisant la thechnique FPGA in the loop sera implémenté sur une carte de développement FPGA de type DE2-115 de la famille Altera, en utilisant les outils Fixed-Point Tool et HDL Coder de l'environnement Simulink de MATLAB.

3.2 Algorithme de commande ANFIS

Dans cette section, nous concevons un régulateur ANFIS destiné à commander la position du pendule inversé.

Prenons en compte un réseau neuro-flou de premier ordre utilisant un système d'inférence floue de type Takagi-Sugeno, avec deux entrées,$x_1$etx_2, et une seule sortie,f.

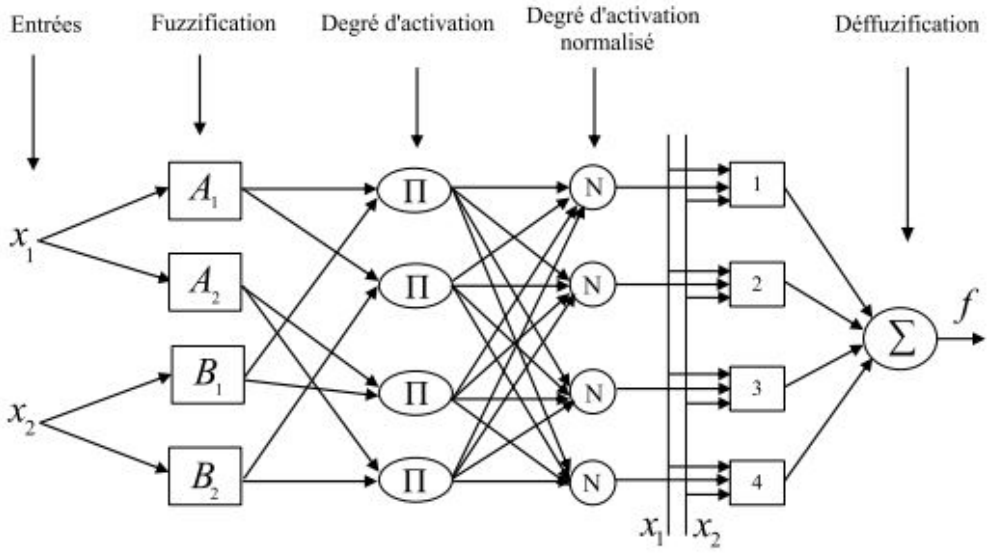


FIGURE 3.1 – Structure du régulateur ANFIS

Les règles floues sont définies comme suit :

$$Règle1 : si\ x_1\ est\ A_1\ et\ x_2\ est\ B_1\ alors\ f_1 = a_1x_1 + b_1x_2 + c_1 \tag{3.1}$$

$$Règle2 : si\ x_1\ est\ A_1\ et\ x_2\ est\ B_2\ alors\ f_2 = a_2x_1 + b_2x_2 + c_2 \tag{3.2}$$

$$Règle3 : si\ x_1\ est\ A_2\ et\ x_2\ est\ B_1\ alors\ f_3 = a_3x_1 + b_3x_2 + c_3 \tag{3.3}$$

$$Règle4 : si\ x_1\ est\ A_2\ et\ x_2\ est\ B_2\ alors\ f_4 = a_4x_1 + b_4x_2 + c_4 \tag{3.4}$$

Où$f_j = a_jx1 + b_jx2 + c_j$ pour$j = 1, 2, ..., 4$, x_1 *et* x_2 sont l'erreur de position et sa dérivée :$[x1, x2] = [e, \Delta e]$.A_i etB_i sont des sous-ensembles flousa_j, b_j *et* c_j sont les paramètres de la conséquence de la règle$j$ déterminés dans le processus d'apprentissage.

Nous attribuons deux ensembles flous à chacune des entrées x_1 etx_2, appelés N (Négatif) et$P$(Positif). Les degrés d'appartenance des variablesx_i aux sous-ensembles flous A_i et B_i, notésμ_P *et* μ_N, sont définis par les fonctions d'appartenance suivantes :

Pour$i = 1, 2$:

$$\mu_P(x_i) = \begin{cases} 0 & si\ x_i < -1 \\ 0.5x_i + 0.5 & si\ -1 < xi < 1 \\ 1 & x_i > 1 \end{cases} \tag{3.5}$$

$$\mu_N(x_i) = \begin{cases} 0 & si\ x_i < -1 \\ -0.5x_i + 0.5 & si\ -1 < xi < 1 \\ 1 & x_i > 1 \end{cases} \tag{3.6}$$

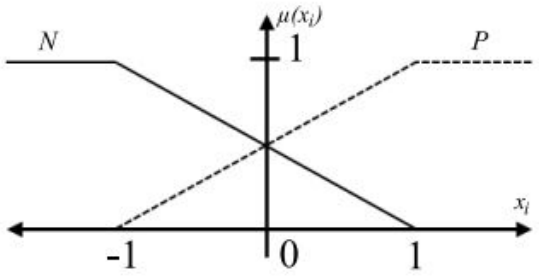


FIGURE 3.2 – Fonctions d'appartenance

En appliquant la méthode de défluzzification par centre de gravité, la valeur numérique de la sortie u est déterminée par :

$$\mu = \frac{\sum_{j=1}^{4} f_j \omega_j}{\sum_{j=1}^{4} \omega_j}$$

Où :

$$\begin{cases} w1 = \mu_P(x_1).\mu_P(x_2) = \mu_{A_1}(x_1).\mu_{B_1}(x_2) \\ w2 = \mu_P(x_1).\mu_N(x_2) = \mu_{A_1}(x_1).\mu_{B_2}(x_2) \\ w3 = \mu_N(x_1).\mu_P(x_2) = \mu_{A_2}(x_1).\mu_{B_1}(x_2) \\ w4 = \mu_N(x_1).\mu_N(x_2) = \mu_{A_2}(x_1).\mu_{B_2}(x_2) \end{cases} \tag{3.7}$$

3.2.1 Algorithme d'apprentissage

Le processus d'apprentissage vise à identifier les paramètres de la prémisse et de la conséquence. Dans cette section, nous élaborons l'algorithme d'apprentissage pour le pendule inversé. Définissons yd et y comme les sorties désirée et réelle du pendule inversé. Dans ce contexte, nous supposons que les paramètres de la prémisse sont constants, tandis que ceux de la conséquence sont ajustés en minimisant la fonction objectif suivante :

$$J = \frac{1}{2}e^2 \tag{3.8}$$

Où $e = y_d - y$

Soit Φj le vecteur des paramètres à ajuster. L'objectif est de trouver les paramètres$a_j, b_j etc_j$ du vecteur Φj en utilisant la méthode de la descente du gradient comme suit :

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k)\frac{\partial J}{\partial \Phi_j} \tag{3.9}$$

Nous avons :

$$\frac{\partial J}{\partial \Phi_j} = -e\frac{\partial y}{\partial \Phi_j} = -e\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j} \tag{3.10}$$

A partir des équations 3.9 et 3.10, nous aurons :

$$\Phi_j(k+1) = \Phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}e \tag{3.11}$$

Dans notre cas, le terme $\frac{\partial y}{\partial u}$ ne peut pas être calculé, mais il peut être estimé en utilisant les équations du filtre de Kalman étendu. L'équation 3.11 peut être écrite comme suit :

$$\Phi_j(k+1) = \Phi_j(k) + K_j^{'}(\Psi_j)^T e \tag{3.12}$$

Où :

$$(\Psi_j)^T = \frac{\partial u}{\partial \Phi j} = \begin{bmatrix} \frac{\partial u}{\partial a_j} \\ \frac{\partial u}{\partial b_j} \\ \frac{\partial u}{\partial c_j} \end{bmatrix} \tag{3.13}$$

$$K_j^{'} = \alpha(k)\frac{\partial y}{\partial u} \tag{3.14}$$

L'équation3.12 peut être identifiée à l'équation du filtre de Kalman étendu :

$$\Phi_j(k+1) = \Phi_j(k) + K(k)e \tag{3.15}$$

Où K (k) est le gain de Kalman défini par :

$$K(k) = \frac{P(k)H^T(k)}{H(k)P(k)H^T(k) + R(k)} \tag{3.16}$$

Avec $H(k)$ est la matrice jacobienne (la matrice d'observation de système),$P(k)$ est la matrice d'estimation de covariance de l'erreur et $R(k)$ est la matrice de covariance du bruit de processus.

En prenant $H^T(k) = (\Psi_j)^T$, $P(k) = \lambda_1$ et $R(k) = \lambda_2$, le gain $K(k)$ peut être écrit sous la forme :

$$K(k) = \frac{\lambda 1}{(\Psi_j)\lambda_1(\Psi j)^T + \lambda_2}(\Psi_j)^T \tag{3.17}$$

Donc l'équation 3.15devient :

$$\Phi_j(k+1) = \Phi_j(k) + \frac{\lambda_1}{(\Psi_j)\lambda_1(\Psi_j)^T + \lambda_2}(\Psi_j)^T e \tag{3.18}$$

Par identification entre les équations 3.12 et 3.18, nous obtenons :

$$K_j^{'} = \frac{\lambda_1}{(\Psi_j)\lambda_1(\Psi_j)^T + \lambda_2} \tag{3.19}$$

Par conséquent, le vecteur des paramètres Φ_j peut être estimé par la formule suivante :

$$\Phi_j(k+1) = \Phi_j(k) + K_j^{'}(\Psi_j)^T e \tag{3.20}$$

3.3 Implimentation du régulateur ANFIS sur la carte DE2-115

Dans cette section, nous allons implémenté le régulteur ANFIS sur la carte FPGA DE2-115 d'Altera en utilisant la technique FPGA in the loop (FiL),en utilisant les outils Fixed-Point Tool et HDL Coder de l'environnement Simulink de MATLAB,afin de commander la position du pendule inversé. L'objectif est de tirer parti des capacités de parallélisme et de traitement rapide des FPGA pour améliorer les performances du système ANFIS.

3.3.1 Modèle du pendule

Le pendule inversé est un système non linéaire qui possède des non-linéarités non négligeables dues à sa structure dynamique et aux forces de friction .Il fait l'objet d'études théoriques et d'expérimentations dans le domaine de l'ingénierie du contrôle des systèmes. Il possède une tige à l'extrémité de laquelle est montée une masse, pivotée sur un chariot qui peut être déplacé horizontalement.Il possède fondamentalement deux états d'équilibre, à savoir un état d'équilibre stable et un état d'équilibre instable. Dans un état d'équilibre stable, le pendule est orienté verticalement vers le bas, tandis qu'en équilibre instable, le pendule est orienté verticalement vers le haut. et une force contraire est donc nécessaire pour le maintenir dans cette position . L'objectif du régulateur est d'amener le pôle à la position d'équilibre instable supérieure[36].

Les techniques de contrôle classiques, souvent inefficaces face à la non-linéarité spécifique du pendule inversé, ont ouvert la voie à l'exploration de solutions plus intelligentes. C'est dans ce contexte que les techniques de contrôle intelligent, telles que les réseaux de neurones flous adaptatifs (ANFIS), ont émergé comme des outils prometteurs[36].

Un chariot et un pendule inversé sont montrés dans la figure (3.3) . $g = 9,8m/s^2$ est la constante de gravité,m est la masse du pendule, $2L$ est la longueur du pendule, M est la masse du chariot, $y(t)$ désigne la distance par rapport à la position initiale, $\theta(t)$est l'angle

du pendule par rapport à l'axe vertical, et $u(t)$ est la force appliquée sur le chariot.

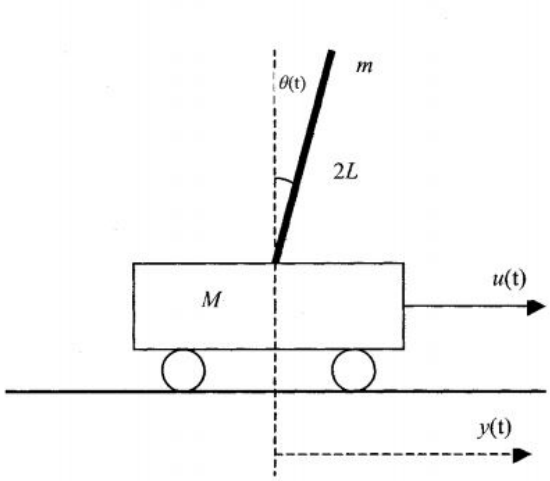


FIGURE 3.3 – un pendule inversé sur un chariot

Pour modèliser ce système ,on utilise la méthode de Lagrange définie par les équations suivantes :

$$L = T - V \tag{3.21}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{3.22}$$

où
T : est l'énergie cinétique ;
V :est l'énergie potentielle ;
q_i : sont les coordonnées généralisées ;
Q_i :sont les forces généralisées ;

- Energie cinétique du système
  - ✦ Énergie cinétique du chariot :

$$T_c = \frac{1}{2}M\dot{x}^2 \tag{3.23}$$

  - ✦ Énergie cinétique du pendule :

$$T_p = \frac{1}{2}m\left(\left(\dot{x} + l\dot{\theta}\cos\theta\right)^2 + \left(l\dot{\theta}\sin\theta\right)^2\right) \tag{3.24}$$

Aprés simplification de l'équation 3.24,l'énergie cinétique prend alors l'expression :

$$T_p = \frac{1}{2}m\left(\dot{x}^2 + 2l\dot{x}\dot{\theta}\cos\theta + l^2\dot{\theta}^2\right) \tag{3.25}$$

✦ l'énergie cinétique totale :

$$T = T_c + T_p = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + 2l\dot{x}\dot{\theta}\cos\theta + l^2\dot{\theta}^2\right) \tag{3.26}$$

En simplifiant :

$$T = \frac{1}{2}(M+m)\dot{x}^2 + ml\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 \tag{3.27}$$

- Énergie potentielle du système :

$$V = -mgl\cos\theta \tag{3.28}$$

- Fonction de Lagrange :

En remplaçant dans l'équation 3.21,la fonction de Lagrange devient :

$$L = \frac{1}{2}(M+m)\dot{x}^2 + ml\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta \tag{3.29}$$

- Équations de Lagrange :

✦ L'équation de Lagrange pour le degré de liberté $\xi(t) = x(t)$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = \tau \Longrightarrow (M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = \tau \tag{3.30}$$

$$(M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = \tau \tag{3.31}$$

✦ L'équation de Lagrange pour le degré de liberté $\xi(t) = \theta(t)$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \Longrightarrow ml\ddot{x}\cos\theta + ml^2\ddot{\theta} = mgl\sin\theta \tag{3.32}$$

$$ml\ddot{x}\cos\theta + ml^2\ddot{\theta} = mgl\sin\theta \tag{3.33}$$

Nous avons donc le système :

$$\begin{cases} (M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = \tau \\ ml\ddot{x}\cos\theta + ml^2\ddot{\theta} = mgl\sin\theta \end{cases} \tag{3.34}$$

En simplifiant, nous obtenons les équations d'état du système :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{g\sin x_1 - \cos x_1\left(\frac{ml}{M+m}x_2^2\sin x_1 - \frac{1}{M+m}\tau(t)\right)}{\frac{4}{3}l - \frac{ml\cos^2 x_1}{M+m}} \end{cases} \tag{3.35}$$

3.3.2 Technique d'implémentation FIL de Simulink

L'approche FPGA-in-the-loop (FiL) est une méthode qui associe des plates-formes logicielles et matérielles pour créer un prototype efficace dans la vérification des systèmes de contrôle, tout en minimisant les risques de dommages au système. Dans cette approche, les algorithmes de contrôle sont implémentés sur une plateforme FPGA, permettant un contrôle en temps réel, tandis que les composants du système tels que les capteurs et les actionneurs mécaniques sont simulés dans un environnement logiciel tel que MATLAB/Simulink. Cette combinaison offre la flexibilité du logiciel avec la précision en temps réel et la vitesse d'exécution matérielle, facilitant ainsi le réglage et la validation du système de manière sûre et efficace[37].

La co-simulation FPGA-in-the-loop (FiL) se divise en deux parties distinctes : l'hôte et la cible FPGA. Dans la première partie, l'hôte, les composants du système sont simulés et analysés à l'aide d'un outil d'analyse tel que MATLAB/Simulink. La seconde partie consiste en la cible FPGA, sur laquelle est implémenté l'algorithme de contrôle. La liaison entre ces deux parties se fait par le biais d'un câble USB-JTAG. L'implémentation de la méthode FiL offre pratiquement les mêmes fonctionnalités que la simulation classique, mais avec l'exécution en temps réel de l'algorithme de contrôle sur la cible FPGA, dans divers scénarios, sans risque de dommages au système[37].

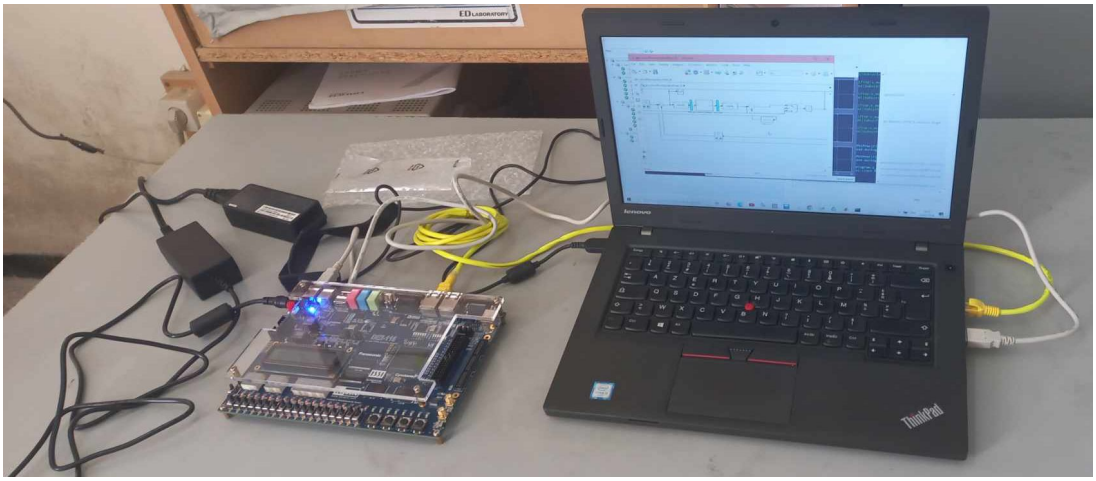


FIGURE 3.4 – Environnement de Co-simulation en utilisant FIL

3.3.3 Préparation du modèle en virgule fixe

Le régulateur neuro-flous (ANFIS) a été développé dans l'environnement Simulink de MATLAB en utilisant des blocs pris en charge par HDL Coder. Ces blocs sont initialement configurés avec des données à virgule flottante à double précision. Ensuite, l'outil Fixed-Point Tool est utilisé pour convertir ces modèles synthétisés en virgule flottante en des modèles en virgule fixe en raison de la rapidité des opérations en virgule fixe, de leur

efficacité en ressources, de leur simplicité de conception.

Cet outil utilise les données de simulation pour suggérer les longueurs des mots binaires ou des parties fractionnaires, en fonction des préférences de l'utilisateur et des exigences de précision pour chaque opération de conception. L'utilisateur a la possibilité de valider les suggestions ou de les ajuster selon ses propres critères.

En outre, Fixed-Point Tool offre la possibilité d'utiliser l'outil "Simulation Data Inspector" pour vérifier et comparer les résultats obtenus en virgule fixe avec ceux obtenus à partir des modèles initiaux en virgule flottante développés sur Simulink. Cette comparaison permet de valider les résultats avant leur implémentation sur FPGA.

Dans notre cas,le régulateur été programmés avec des blocs de Simulink pris en charge par HDL Coder.Les sorties du régulateur ont été calculées à l'aide de l'outil Fixed-Point Tool en combinant les parties entière et fractionnaire sur 20 bits pour garantir la précision nécessaire.

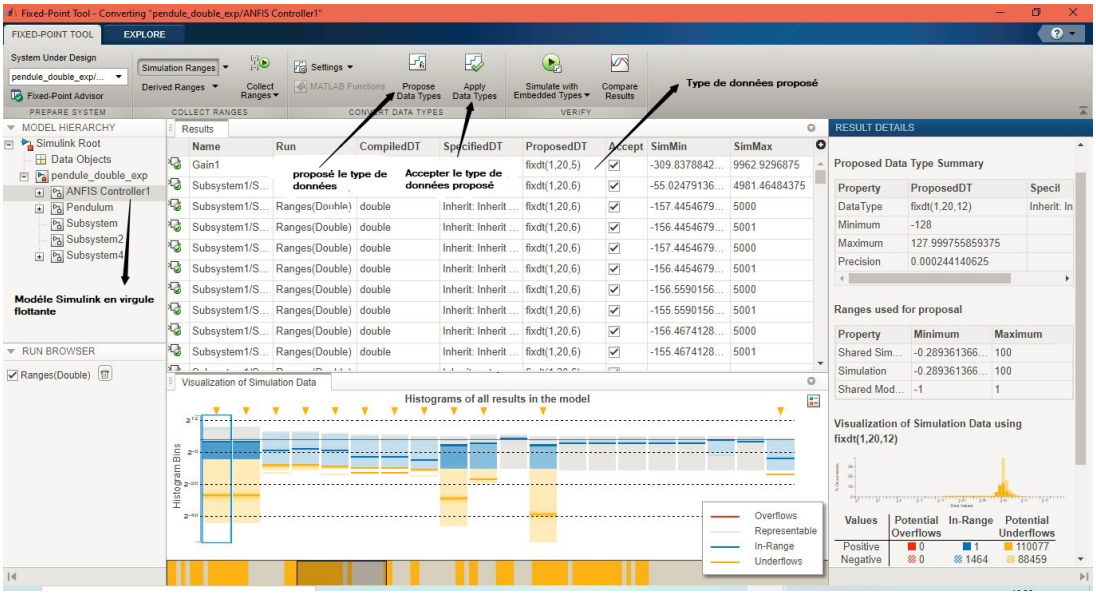


FIGURE 3.5 – L'interface de Fixed-Point Tool sur Simulink.

3.3.4 Implémentation du régulateur sur la carte FPGA via HDL Coder

Le régulateur ANFIS proposé a été implémenté sur la carte DE2-115 en utilisant l'outil HDL Coder de MATLAB via l'interface "Workflow Advisor". Cette interface offre plusieurs fonctionnalités pour la mise en œuvre des conceptions sur FPGA, notamment le choix de la méthode d'implémentation, la sélection de la plateforme cible, le logiciel de synthèse HDL, ainsi que des outils d'optimisation et de simulation.

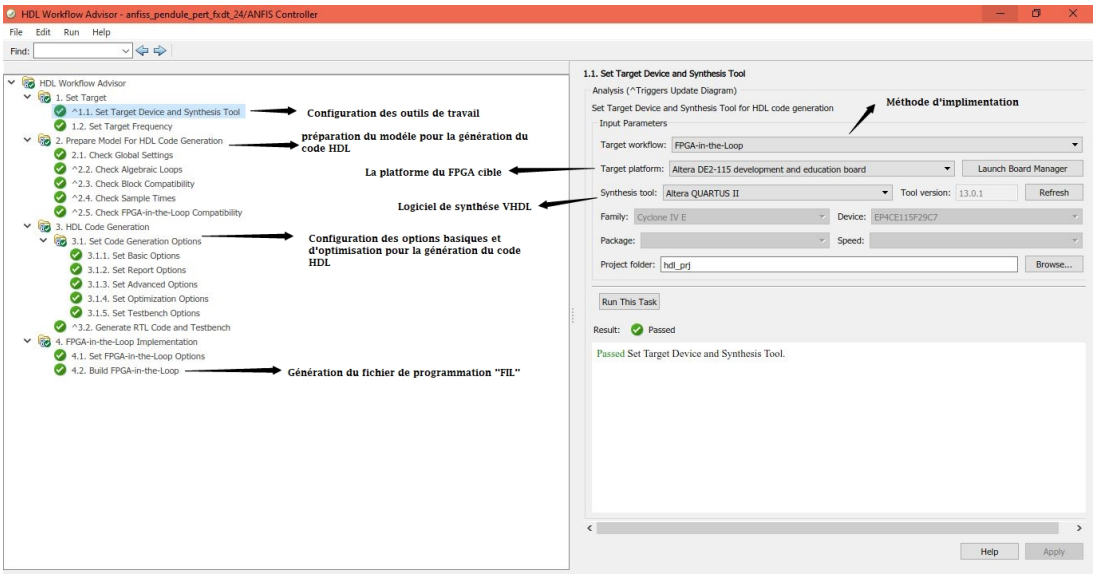


FIGURE 3.6 – Configuration de la plateforme FPGA sur l'interface de HDL Coder.

HDL Coder est utilisé pour générer le code VHDL du système de contrôle à partir du modèle Simulink obtenu précédemment. Chaque bloc du modèle est synthétisé en fichiers VHDL séparés, importés ensuite comme composants dans un fichier VHDL de niveau supérieur.

Après avoir généré les codes VHDL de l'algorithme de contrôle, le logiciel "Quartus ", conçu pour les FPGA de la famille ALTERA, est utilisé pour réaliser les étapes de synthèse logique et d'ajustement . Ces étapes permettent de générer le bitstream nécessaire à la programmation du FPGA.

Une fois le fichier de programmation du FPGA (.sof) généré, tous les blocs de l'algorithme de contrôle dans MATLAB/SIMULINK sont remplacés par un bloc unique FIL correspondant à l'algorithme implémenté sur FPGA. La connexion entre la carte cible et l'ordinateur se fait via un câble JTAG-USB, également utilisé pour la programmation du FPGA.

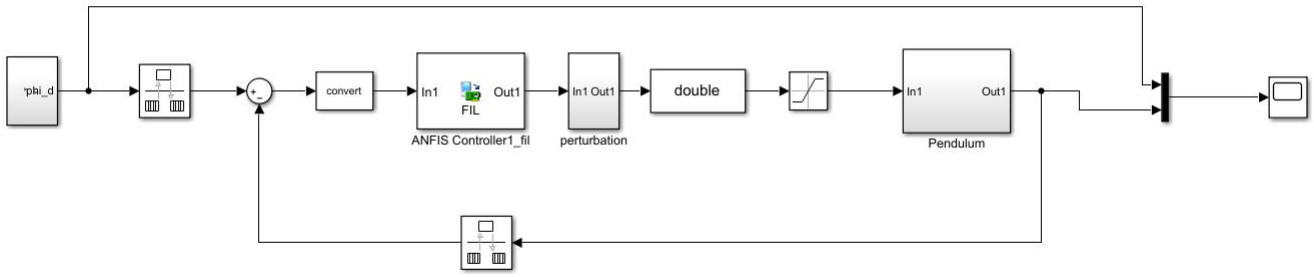
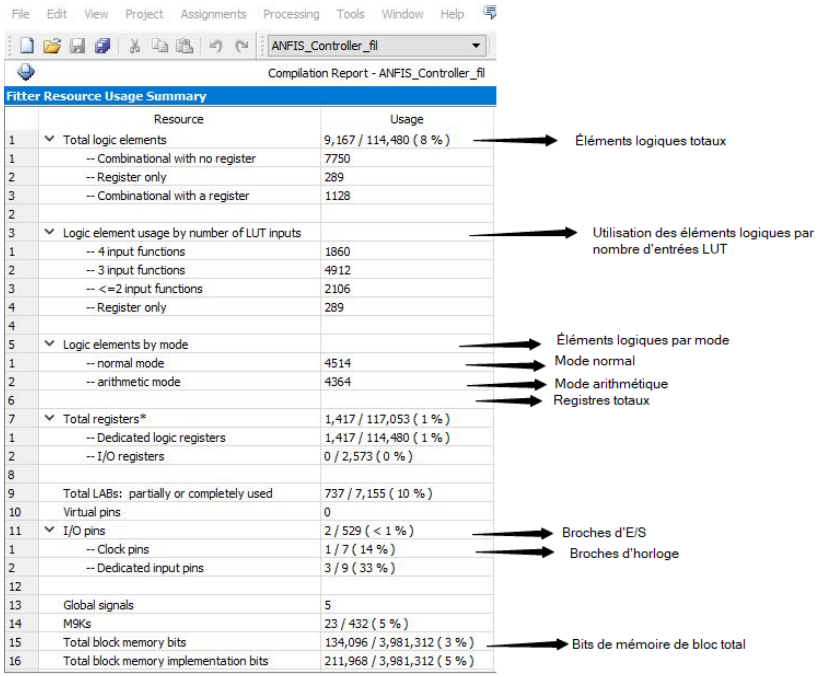


FIGURE 3.7 – Vue globale du modèle de Co-simulation FIL

3.3.5 Ressources matérielles consommées

Grâce aux techniques d'optimisation intégrées à HDL Coder, la solution de contrôle pour le pendule inversé est à la fois robuste et efficace. Ces techniques permettent de répondre aux exigences de performances tout en optimisant l'utilisation des ressources matérielles disponibles sur le FPGA.

La figure (3.8) illustre les ressources matérielles consommées sur le FPGA lors de l'implémentation de l'algorithmes de contrôle Neuro-flou en utilisant la référence 1 et la figure (3.9) illustre les ressources matérielles utilisées lors de l'implémentation de algorithmes de contrôle Neuro-flou en utilisant la référence 2. A partir de ces figures, on peut constater que les ressources matérielles consommées lors de la mise en œuvre des deux algorithmes sont considérablement optimisées en raison de la méthodologie de conception adoptée.


File Edit View Project Assignments Processing Tools Window Help

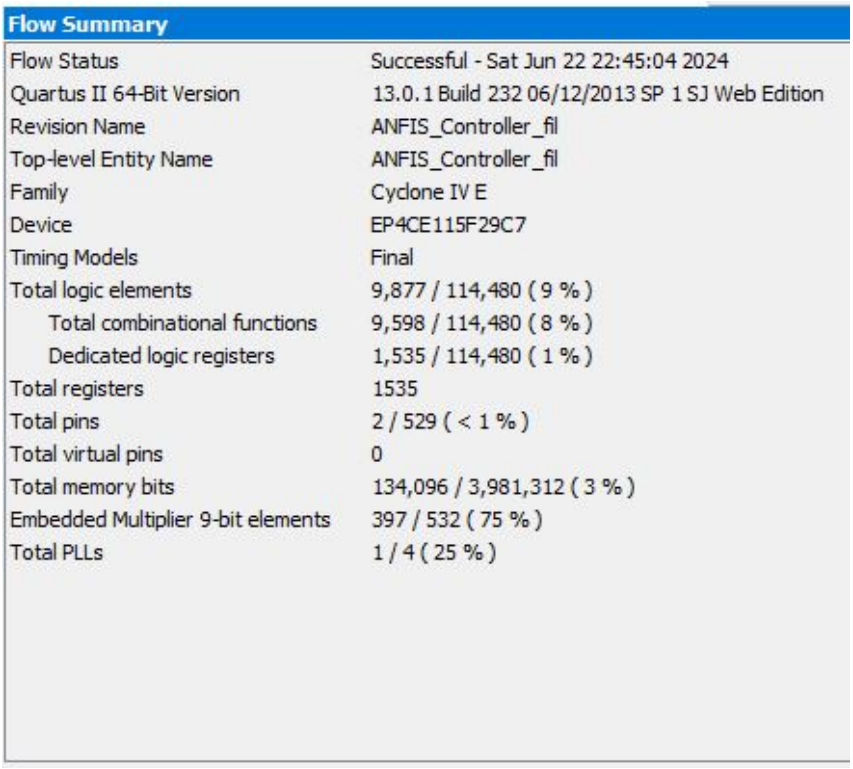
ANFIS_Controller_fil

Compilation Report - ANFIS_Controller_fil

Fitter Resource Usage Summary

	Resource	Usage	
1	Total logic elements	9,167 / 114,480 (8 %)	Éléments logiques totaux
1	-- Combinational with no register	7750	
2	-- Register only	289	
3	-- Combinational with a register	1128	
2			
3	Logic element usage by number of LUT inputs		Utilisation des éléments logiques par nombre d'entrées LUT
1	-- 4 input functions	1860	
2	-- 3 input functions	4912	
3	-- <=2 input functions	2106	
4	-- Register only	289	
4			
5	Logic elements by mode		Éléments logiques par mode
1	-- normal mode	4514	Mode normal
2	-- arithmetic mode	4364	Mode arithmétique
6			Registres totaux
7	Total registers*	1,417 / 117,053 (1 %)	
1	-- Dedicated logic registers	1,417 / 114,480 (1 %)	
2	-- I/O registers	0 / 2,573 (0 %)	
8			
9	Total LABs: partially or completely used	737 / 7,155 (10 %)	
10	Virtual pins	0	
11	I/O pins	2 / 529 (< 1 %)	Broches d'E/S
1	-- Clock pins	1 / 7 (14 %)	Broches d'horloge
2	-- Dedicated input pins	3 / 9 (33 %)	
12			
13	Global signals	5	
14	M9Ks	23 / 432 (5 %)	
15	Total block memory bits	134,096 / 3,981,312 (3 %)	Bits de mémoire de bloc total
16	Total block memory implementation bits	211,968 / 3,981,312 (5 %)	

FIGURE 3.8 – Ressources matérielles consommées sur le FPGA en utilisant la référence 1.



Flow Summary	
Flow Status	Successful - Sat Jun 22 22:45:04 2024
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ANFIS_Controller_fil
Top-level Entity Name	ANFIS_Controller_fil
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	9,877 / 114,480 (9 %)
Total combinational functions	9,598 / 114,480 (8 %)
Dedicated logic registers	1,535 / 114,480 (1 %)
Total registers	1535
Total pins	2 / 529 (< 1 %)
Total virtual pins	0
Total memory bits	134,096 / 3,981,312 (3 %)
Embedded Multiplier 9-bit elements	397 / 532 (75 %)
Total PLLs	1 / 4 (25 %)

FIGURE 3.9 – Ressources matérielles consommées sur le FPGA en utilisant la référence 2.

3.4 Conclusion

Dans ce chapitre, nous avons développé le régulateur ANFIS proposé. Ensuite, nous avons présenté la méthodologie de conception et d'implémentation de ce régulateur sur la carte FPGA DE2-115. Nous avons expliqué la méthodologie adoptée à l'aide de MATLAB-Simulink pour concevoir et implémenter le régulateur sur FPGA, en utilisant l'outil Fixed-Point Tool pour la conversion des types de données en virgule flottante en virgule fixe, et l'outil HDL Coder pour charger les programmes sur la carte FPGA. Cette méthodologie nous a permis de réduire le temps de conception des algorithmes et d'obtenir un régulateur préci et robuste, avec un code VHDL optimal en termes de ressources matérielles consommées sur le FPGA.

Chapitre 4

Résultats et déscussions

4.1 Introduction

Dans ce dernier chapitre ,nous allons présenter les résultats de la simulation et de l'implémentation du régulateur ANFIS proposé précédemment en utilisant la technique FPGA-in-the-Loop. L'objectif principal est de démontrer la performance et la robustesse du régulateur ANFIS appliqué à la stabilisation du pendule inversé qui un système reconnu pour sa dynamique non linéaire complexe.

4.2 Résultats de simulation

Dans cette section, nous présentons les résultats de simulation du régulateur ANFIS pour le contrôle de la position du pendule inversé, en utilisant deux références distinctes : $r1(t) = 0.2exp(-4t)$ et $r2(t) = 0.052(sin(t) + 3sin(0.3t))$. Les performances du régulateur ont été évaluées dans des simulations avec présence et absence de perturbation. La perturbation, représentée par $p(t) = 150sin(t)$, a été introduite pour examiner la réponse du régulateur face à des conditions dynamiques variées.

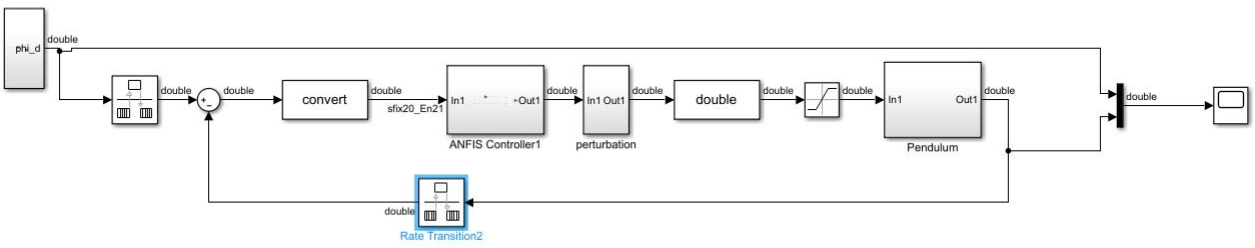


FIGURE 4.1 – Simulation du régulateur sur MATLAB/SIMULINK

4.2.1 $1^{ére}$référence

Les courbes suivantes montrent la performance du système dans des conditions variées, avec et sans perturbations, en utilisant la $1^{ére}$référence $r_1(t) = 0.2exp(-4t)$ pour la stabilisation du pendule.

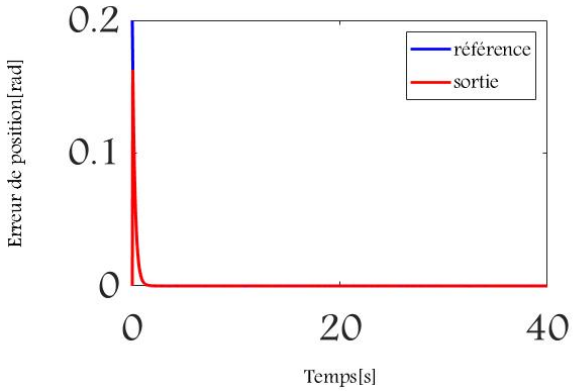


FIGURE 4.2 – Résultats de poursuite de position en absence de perturbations

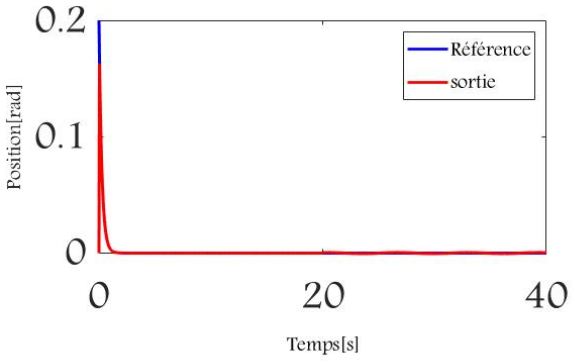


FIGURE 4.3 – Résultats de poursuite de position en présence de perturbations

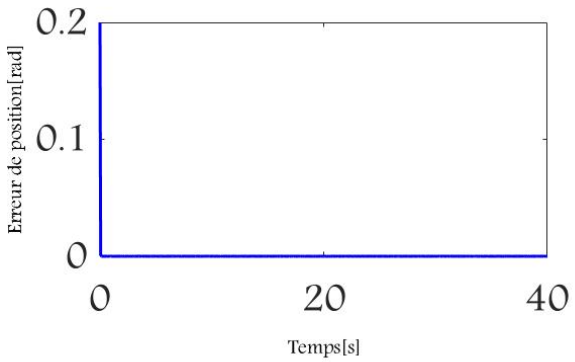


FIGURE 4.4 – Résultats de l'erreur de position en absence perturbations

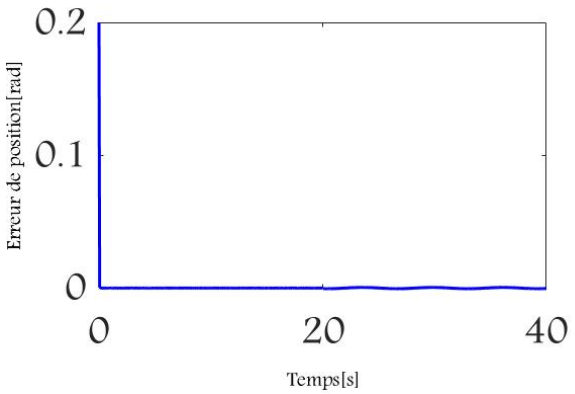


FIGURE 4.5 – Résultats de l'erreur de position en présence de perturbations

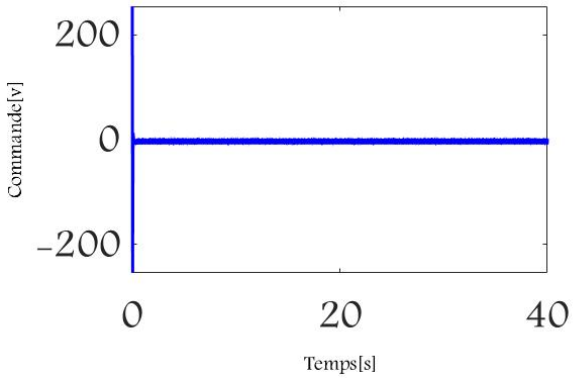


FIGURE 4.6 – visualisation du signal de commande en absence de perturbations

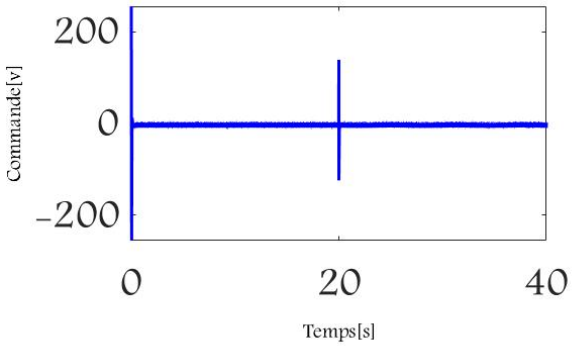


FIGURE 4.7 – visualisation du signal de commande en présence de perturbations

4.2.2 Discussion des résultats

D'après les résultats de simulation présentés sur les figures, il est évident que les objectifs de stabilisation ont été atteints avec succès. Les courbes montrent clairement que la commande assure une stabilisation rapide et efficace du système, même en présence de perturbations extérieures.

Les figures illustrent que cette stratégie de commande parvient à rejeter rapidement les perturbations, garantissant ainsi que les performances de poursuite restent pratiquement intactes. En effet, même lorsque des perturbations significatives sont introduites, le système parvient à réguler les écarts et à revenir à un état stable, démontrant ainsi une résilience remarquable.

De plus, l'analyse de l'erreur révèle qu'elle est négligeable, voire nulle. Cette faible erreur témoigne de l'efficacité du régulateur ANFIS à maintenir le système proche de sa trajectoire désirée en tout temps, renforçant ainsi sa capacité à atteindre et à maintenir la stabilisation.

La capacité du régulateur à maintenir une haute performance et à stabiliser le système malgré les perturbations extérieures témoigne de son efficacité et de sa robustesse. Ces résultats mettent en évidence que la commande par ANFIS est particulièrement adaptée aux systèmes dynamiques non linéaires comme le pendule inversé.

4.2.3 $2^{éme}$référence

Les résultats de simulation présentés dans cette section utilisent une deuxième référence $r_2(t) = 0.052(sin(t) + 3sin(0.3t))$, caractérisée par une non-linéarité intense . Cette référence permet de tester plus rigoureusement la robustesse et l'efficacité de la commande ANFIS dans des conditions dynamiques complexes.

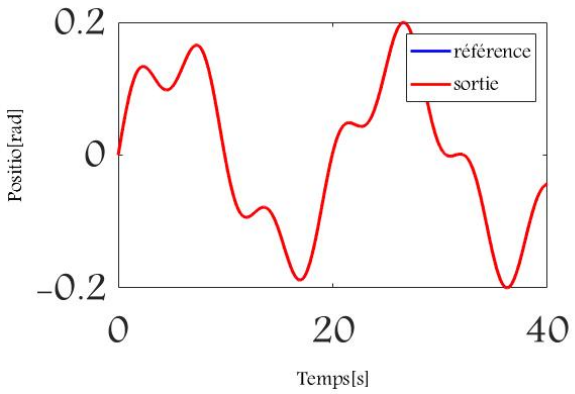


FIGURE 4.8 – Résultats de poursuite de position en absence de perturbations

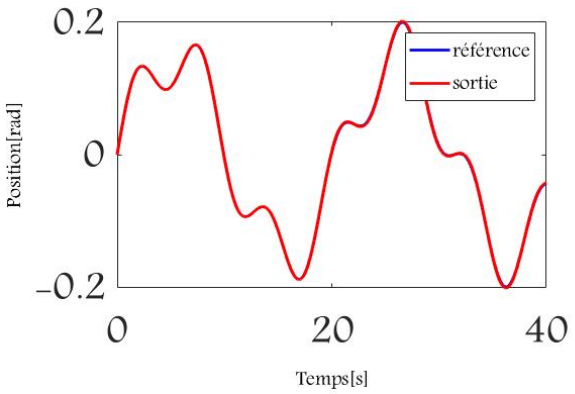


FIGURE 4.9 – Résultats de poursuite de position en présence de perturbations

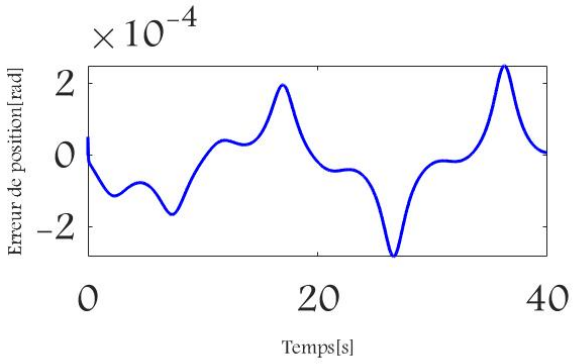


FIGURE 4.10 – Résultats de l'erreur de position en absence de perturbations

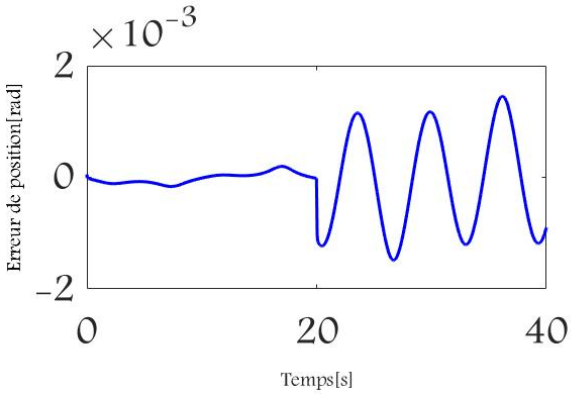


FIGURE 4.11 – Résultats de l'erreur de position en présence de perturbations

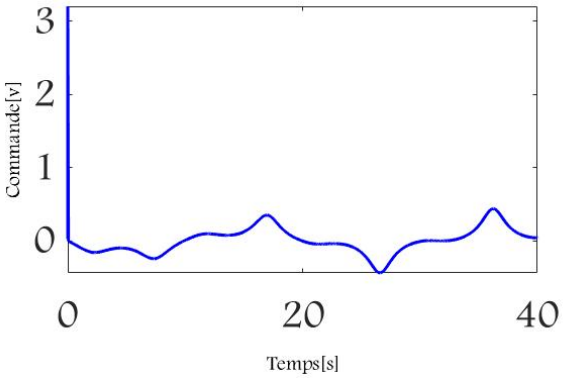


FIGURE 4.12 – visualisation du signal de commande en absence de perturbations

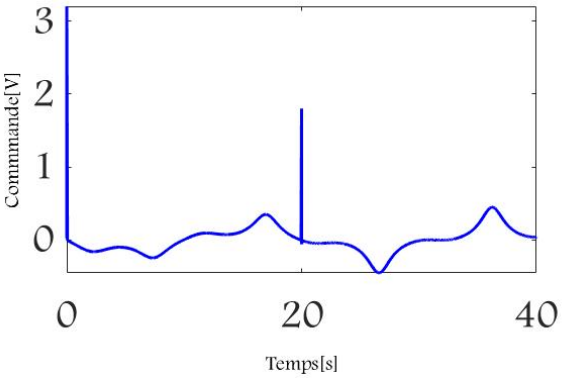


FIGURE 4.13 – visualisation du signal de commande en présence de perturbations

4.2.4 Discussion des résultats

D'après les résultats de simulation présentés sur les figures, la commande basée sur la référence strictement non linéaire démontre une remarquable capacité à stabiliser efficacement le pendule inversé malgré la présence d'une non linéarité intense dans le système.

Les courbes mettent en évidence la capacité remarquable de la commande ANFIS à maintenir le pendule sur sa trajectoire souhaitée face aux variations dynamiques complexes. Même confrontée à des perturbations importantes, la commande parvient à maintenir la stabilité du système, illustrant ainsi sa robustesse et son adaptabilité dans des environnements dynamiques et imprévisibles.

De plus, l'analyse de l'erreur révèle des niveaux d'erreur minimes, soulignant ainsi l'efficacité de la commande ANFIS à gérer efficacement la non linéarité intense du système. Cette performance exceptionnelle renforce la confiance en la capacité du régulateur à maintenir la stabilité du système dans des conditions réelles.

En termes de robustesse, la commande ANFIS se distingue par sa capacité à stabiliser le pendule inversé de manière cohérente et fiable, même lorsque le système est soumis à des conditions dynamiques extrêmes. Cette capacité à maintenir la stabilité du système face à des défis variés confirme la fiabilité et la robustesse de la commande basée sur la référence strictement non linéaire.

En conclusion, les résultats de simulation confirment que la commande basée sur la référence strictement non linéaire offre une solution robuste et fiable pour la stabilisation du pendule inversé, garantissant ainsi des performances optimales même dans des environnements dynamiques et non linéaires.

4.3 Résultats d'implémentation

Nous présentons dans cette section les résultats d'implémentation du régulateur ANFIS développé précédemment sur une carte FPGA à l'aide de la méthode 'FPGA in the loop'. Ces résultats permettent d'évaluer les performances du régulateur dans un environnement matériel réel et de comparer ses performances avec celles obtenues lors des simulations.

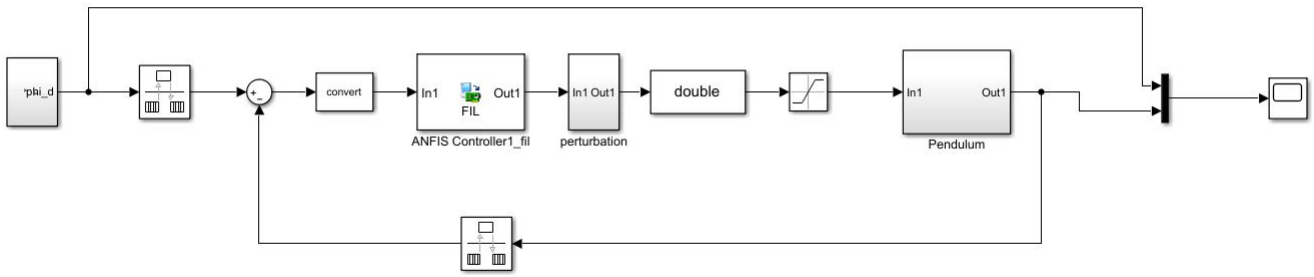


FIGURE 4.14 – : Vue globale du modèle de Co-simulation FIL

4.3.1 $1^{ére}$référence

Les courbes suivantes montrent les résultats de l'implémentation du régulateur ANFIS dans des conditions variées, avec et sans perturbations, en utilisant la $1^{ére}$référence $r_1(t) = 0.2exp(-4t)$ pour la stabilisation du pendule.

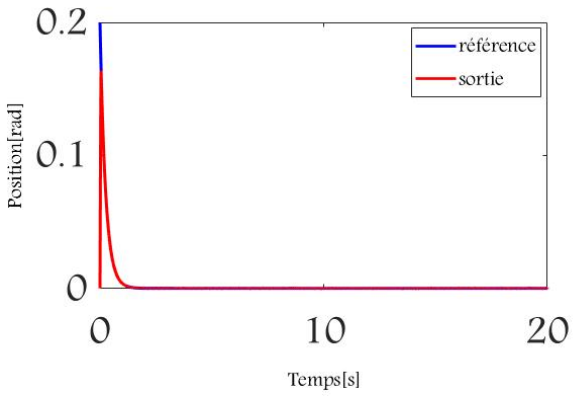


FIGURE 4.15 – Résultats de poursuite de position en absence de perturbations

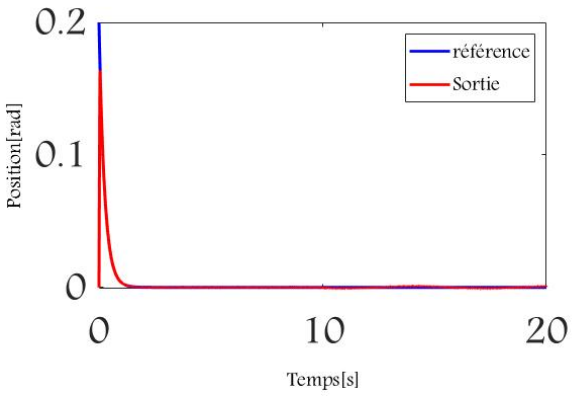


FIGURE 4.16 – Résultats de poursuite de position en présence de perturbations

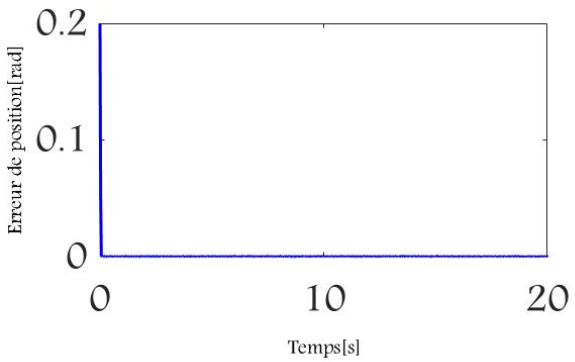


FIGURE 4.17 – Résultats de l'erreur de position en absence de perturbations

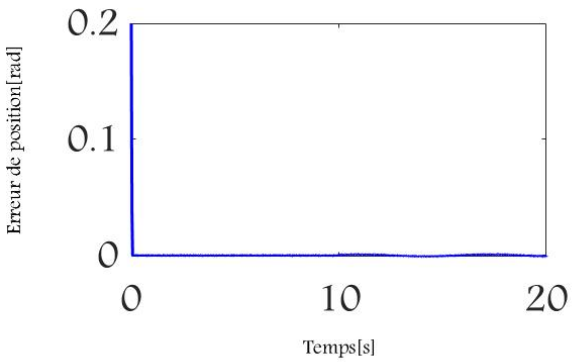


FIGURE 4.18 – Résultats de l'erreur de position en présence de perturbations

4.3.2 Discussion des résultats

Les résultats de l'implémentation du régulateur ANFIS sur une carte FPGA avec la méthode "FPGA in the loop" pour la $1^{ére}$ référence confirment son efficacité dans un environnement matériel réel. Les courbes obtenues après l'implémentation démontrent une stabilisation efficace du pendule inversé, en accord avec les résultats de simulation. Le régulateur ANFIS réagit de manière robuste aux perturbations, maintenant ainsi la stabilité du système. Ces résultats confirment la capacité du régulateur à fonctionner de manière fiable dans des conditions pratiques, ce qui est crucial pour son déploiement réel.

4.3.3 $2^{éme}$référence

Les résultats d'implimentation présentés dans cette section utilisent une deuxième référence $r_2(t) = 0.052(sin(t) + 3sin(0.3t))$.

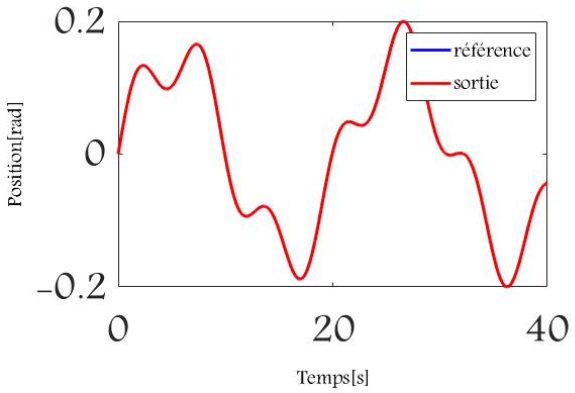


FIGURE 4.19 – Résultats de poursuite de position en absence de perturbations

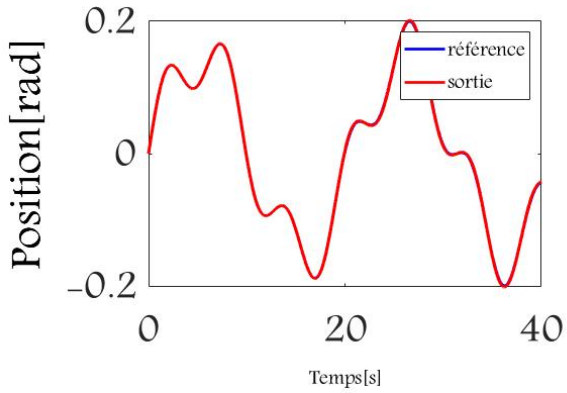


FIGURE 4.20 – Résultats de poursuite de position en présence de perturbations

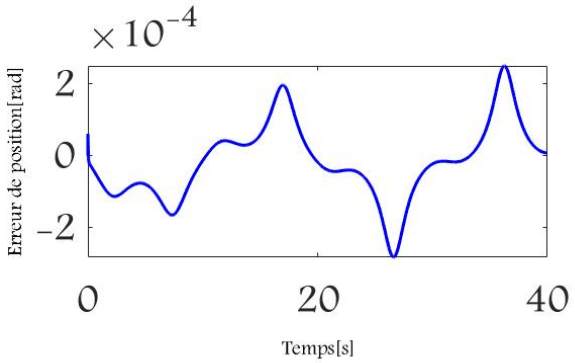


FIGURE 4.21 – Résultats de l'erreur de position en absence de perturbations

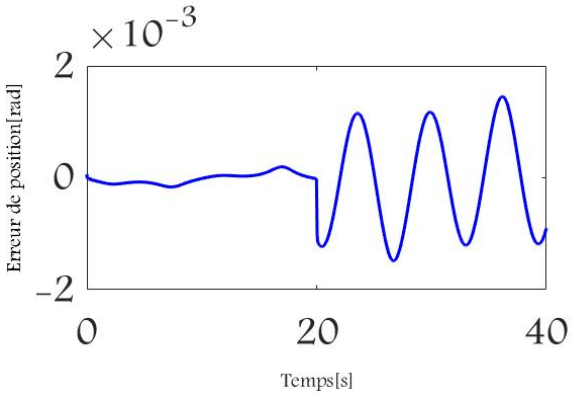


FIGURE 4.22 – Résultats de l'erreur de position en présence de perturbations

4.3.4 Discussion des résultats

L'implémentation du régulateur ANFIS sur une carte FPGA avec la méthode "FPGA in the loop" pour la $2^{éme}$ référence strictement non linéaire montre également des performances encourageantes. Malgré la complexité de la référence non linéaire, le régulateur ANFIS parvient à maintenir la stabilité du pendule inversé dans un environnement matériel réel. Les courbes obtenues confirment la capacité du régulateur à s'adapter à des références de stabilisation variées, même dans des conditions non linéaires intensives. Cette capacité témoigne de la robustesse et de la flexibilité du régulateur ANFIS dans des environnements pratiques, renforçant ainsi sa pertinence pour des applications réelles.

4.4 Conclusion

Dans ce chapitre, nous avons présenté les résultats de simulation et d'implémentation obtenus dans le cadre de l'étude sur la stabilisation du pendule inversé à l'aide du régulateur ANFIS, en mettant particulièrement l'accent sur la technique "FPGA in the loop". Cette approche nous a permis de valider la performance et la robustesse du régulateur dans des conditions pratiques en temps réel.

Les résultats de simulation ont révélé la capacité du régulateur ANFIS à stabiliser efficacement le pendule inversé, avec une réactivité notable aux perturbations et une capacité à maintenir des performances élevées. L'analyse des courbes a confirmé la cohérence des résultats entre la simulation et l'implémentation sur FPGA, soulignant ainsi la fiabilité du régulateur dans un environnement matériel réel.

L'implémentation sur FPGA avec la méthode "FPGA in the loop" a été essentielle pour valider la performance du régulateur dans des conditions pratiques. Les résultats obtenus ont confirmé sa capacité à maintenir la stabilité du pendule inversé et sa robustesse face aux perturbations, même dans des contextes non linéaires intenses. Cette approche a démontré la faisabilité de l'utilisation du régulateur ANFIS dans des applications réelles.

Conclusion générale

Ce mémoire a exploré les possibilités offertes par l'implémentation d'un régulateur neuro-flou sur une plateforme matérielle FPGA, en utilisant la technique FPGA-in-the-loop , et son application à la commande d'un pendule inversé. En combinant les avantages des systèmes neuro-flous avec la puissance de traitement et la flexibilité des FPGA, ce projet a ouvert de nouvelles perspectives dans le domaine du contrôle des systèmes dynamiques.

L'étude présentée dans ce mémoire a pour objectif examiner les possibilités offertes par l'implémentation d'un régulateur neuro-flou sur une plateforme matérielle FPGA, en utilisant la technique FPGA-in-the-loop , et son application à la commande de position d'un pendule inversé.

Dans la première partie de ce travail, nous avons donné quelques notions de base sur la logique floue, les réseaux de neurones ainsi que les réseaux neuro-flous. Cette approche combine la capacité d'apprentissage des réseaux de neurones avec la flexibilité de la logique floue, offrant ainsi un contrôleur adaptatif et performant pour des systèmes dynamiques complexes comme le pendule inversé.

La deuxième partie a été consacrée à la présentation de l'architecture des circuits FPGA du constructeur Altera, en mettant l'accent sur ceux de la famille DE2-115 utilisés dans ce travail. Nous avons également exposé les particularités de la carte de développement que nous avons utilisée, ainsi que les différents logiciels et outils permettant sa programmation. L'utilisation des FPGA offre des avantages significatifs en termes de vitesse de traitement, de parallélisme et de reconfigurabilité, ce qui en fait une plateforme idéale pour l'implémentation en temps réel de régulateurs neuro-flous.

La troisième partie a été dédiée au développement d'un régulateur neuro-flou adaptatif de type ANFIS pour le système du pendule inversé. Nous avons utilisé un algorithme d'apprentissage basé sur le filtre de Kalman étendu pour adapter les paramètres de la conséquence du réseau neuro-flou en ligne, profitant ainsi de la capacité de calcul du FPGA et de sa fréquence d'échantillonnage élevée. La conception et l'implémentation des algorithmes de contrôle sur la carte DE2-115 ont été effectuées en utilisant l'environnement Simulink de MATLAB. La préparation pour l'implémentation des modèles développés sur Simulink a été faite en utilisant l'outil Fixed-Point Tool, où les modèles ont été convertis en virgule fixe avant d'être implémentés sur la carte FPGA via l'outil HDL Coder. Une telle méthodologie de conception et d'implémentation a permis de réduire considérablement le temps de conception, et d'obtenir un algorithme rapide et précis avec un code VHDL optimal en terme de ressources matérielles consommées sur la carte fpga.L'utilisation de la technique FPGA-in-the-loop a permis de simuler et de tester efficacement les algorithmes de commande, garantissant une validation précise des performances du système.

Enfin, la dernière partie a été consacrée à la présentation et la discussion des résultats

de simulation et de l'implémentation. Ces résultats ont montré une amélioration significative des performances de commande , mettant en évidence le potentiel des régulateurs neuro-flous sur FPGA pour résoudre des problèmes complexes de commande.

En conclusion, ce projet a contribué à l'avancement des connaissances dans le domaine du contrôle des systèmes dynamiques en proposant une solution novatrice et efficace basée sur l'intégration des techniques de l'intelligence artificielle, du FPGA-in-the-loop et des plates-formes matérielles FPGA.

Bibliographie

[1] Siddique, N. (2014). Intelligent Control : A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms. Springer.

[2] Nguyen, H. T., Walker, C. L., Walker, E. A. (2018). A First Course in Fuzzy Logic (4th ed.). Chapman and Hall/CRC.

[3] Nguyen, H. T., Prasad, N. R., Walker, C. L., Walker, E. A. (2002). A First Course in Fuzzy and Neural Control. Chapman and Hall/CRC.

[4] Doudou, S. (2013). Contribution à la commande moderne des systèmes non linéaires multivariables non affines. Thèse de doctorat, Université Sétif 1.

[5] Khati, H. (2020). Commande d'une Architecture de Téléopération par la Carte FPGA. Université Mouloud Mammeri de Tizi-Ouzou.

[6] Fuller, R. (2000). Introduction to Neuro-Fuzzy Systems. Springer.

[7] Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., Alves, S. F. D. R. (2017). Artificial Neural Networks : A Practical Course. Springer.

[8] Gorzalczany, M. B. (2002). Computational Intelligence Systems and Applications : NeuroFuzzy and Fuzzy Neural Synergisms. Physica-Verlag Heidelberg.

[9] Benchaabane, A. (2011). Commandes hybrides neuro-glissantes et neuro-floues appliquées au simulateur d'hélicoptère TRMS. Mémoire de Magister, École Nationale Polytechnique d'Alger.

[10] Chakraverty, S., Mall, S. (2017). Artificial Neural Networks for Engineers and Scientists : Solving Ordinary Differential Equations. CRC Press.

[11] Stavroulakis, P. (2004). Neuro-Fuzzy and Fuzzy-Neural Applications in Telecommunications. Springer.

[12] Saidoun, O. (2008). Commande floue et neuro-floue du niveau de liquide d'un réservoir. Université de Mouloud Mammeri de Tizi Ouzou.

[13] Jang, J. S. R., Sun, C. T., Mizutani, E. (1997). Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence. Prentice Hall.

[14] Farooq, U., Marrakchi, Z., Mehrez, H. (2012). Tree-Based Heterogeneous FPGA Architectures : Application-Specific Exploration and Optimization. Springer Science and Business Media.

[15] Blanchardon, A. (2015). Synthèse d'architectures de circuits FPGA tolérants aux défauts. Thèse de doctorat, Université Pierre et Marie Curie - Paris VI.

[16] Abdusalam, M. M. (2008). Structures et stratégies de commande des filtres actifs parallèle et hybride avec validations expérimentales. Thèse de Doctorat, Université Henri Poincaré, Nancy-I.

[17] Kuon, I., Tessier, R., Rose, J. (2008). "FPGA Architecture : Survey and Challenges". Foundations and Trends in Electronic Design Automation, 2(2), 135–253.

[18] Rose, J., El Gamal, A., Sangiovanni-Vincentelli, A. (1993). "Architecture of Field-Programmable Gate Arrays". Proceedings of the IEEE, 81(7).

[19] Ahmed, C. B. (2016). Fault-Mitigation Strategies for Reliable FPGA Architecture. Thèse de doctorat, Université de Rennes 1.

[20] Marques, N. (2012). Méthodologie et architecture adaptative pour le placement efficace de tâches matérielles de tailles variables sur des partitions reconfigurables. Thèse de doctorat, Université de Lorraine.

[21] Chen, D., Cong, J., Pan, P. (2006). "FPGA Design Automation : A Survey". Foundations and Trends in Electronic Design Automation, 1(3), 195–330.

[22] Monmasson, E., Idkhajine, L., Cirstea, M. N., Bahri, I., Alin, T. (2016). "FPGAs in industrial control applications". IEEE Transactions on Industrial Informatics, 7(2), 224-243.

[23] Parvez, H., Mehrez, H. (2011). Application-Specific Mesh-based Heterogeneous FPGA Architectures. Springer.

[24] Antonik, P. (2018). Application of FPGA to Real-Time Machine Learning : Hardware Reservoir Computers and Software Image Processing. Thèse de doctorat, Université libre de Bruxelles.

[25] Parab, J. S., Gad, R. S. (2018). Hands-on Experience with Altera FPGA Development Boards. Springer.

[26] Terasic Technologies Inc. (2012). DE2-115 User Manual. World Leading FPGA Based Products and Design Services.

[27] Liu, N. (2013). "The Research of Direct Digital Frequency Synthesis Based on FPGA". In 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC).

[28] Chakhari, A. (2014). Évaluation analytique de la précision des systèmes en virgule fixe pour des applications de communication numérique. Thèse de doctorat, Université de Rennes.

[29] Lopez, B. (2014). Implémentation optimale de filtres linéaires en arithmétique virgule fixe. Thèse de doctorat, Université Pierre et Marie Curie - Paris VI.

[30] Rocher, R. (2006). Évaluation analytique de la précision des systèmes en virgule fixe. Thèse de doctorat, Université Rennes 1.

[31] Roth, C. H., Jr., John, L. K. (2008). Digital Systems Design Using VHDL (2nd ed.). Thomson Learning.

[32] Hwang, E. O. (2016). Digital Logic and Microprocessor Design with Interfacing. La Sierra University.

[33] Perisse, T. (2009). Manuel d'utilisation de Quartus II.

[34] Crockett, L. H., Elliot, R. A., Enderwitz, M. A., Stewart, R. W. (2014). The Zynq Book : Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000, All Programmable SoC. Department of Electronic and Electrical Engineering, University of Strathclyde.

[35] Kharola, A. (2016). "A PID Based ANFIS & Fuzzy Control of Inverted Pendulum on Inclined Plane (IPIP)". International Journal on Smart Sensing and Intelligent Systems, 9(2).

[36] Jaiswal, S., Ballal, M. S. (2017). "FDST-based PQ event detection and energy metering implementation on FPGA-in-the-loop and NI-LabVIEW". IET Science, Measurement and Technology, 11(4), 453-463.

[37] ZOGHBI,A.(2022).Développement et implémentation d'un algorithme adaptatif : Application au filtrage actif de puissance.Thèse de doctorat, UniversitéEcole Nationale Polytechnique-Alger.

Résumé

Résumé :Dans ce mémoire, un régulateur neuro-flou adaptatif est implémenté sur une carte FPGA en utilisant la technique FIL pour commander la position d'un pendule inversé. Dans cette architecture, le régulateur est exécuté sur le FPGA, tandis que le reste de la boucle de régulation est réalisé dans l'environnement Simulink de MATLAB. L'apprentissage du réseau neuro-flou se déroule en ligne, ajustant les paramètres de la conséquence des règles floues à l'aide d'un algorithme basé sur les méthodes de la descente du gradient et du filtre de Kalman étendu. Cette approche tire parti du parallélisme de calcul du FPGA et de sa haute fréquence d'échantillonnage.

Le régulateur proposé est développé dans l'environnement Simulink de MATLAB et implémenté en utilisant les outils "Fixed-Point Tool" et "HDL Coder". Cette méthodologie de conception permet d'obtenir un algorithme précis avec un code VHDL optimal en termes de ressources matérielles utilisées, tout en réduisant le temps de conception de l'algorithme. Les résultats obtenus ont démontré l'efficacité des contrôleurs proposés.

Mots-clés : FPGA, HDL Coder, ANFIS, Commande neuro-floue, Filtre de Kalman étendu, Fixed-Point Too,FPGA in the loop, Pendule inversé.

Abstract : In this thesis, an adaptive neuro-fuzzy controller is implemented on an FPGA board using the FIL technique to control the position of an inverted pendulum. In this setup, the controller runs on the FPGA, while the rest of the control loop operates within the Simulink environment of MATLAB. The online learning of the neuro-fuzzy network adjusts the parameters of the fuzzy rule consequent using an algorithm based on gradient descent and extended Kalman filtering methods. This approach leverages the FPGA's computational parallelism and high sampling frequency.

The proposed controller is developed in MATLAB's Simulink environment and implemented using the "Fixed-Point Tool" and "HDL Coder". This design methodology achieves a precise algorithm with optimal VHDL code in terms of hardware resources, while reducing algorithm design time. The results demonstrate the effectiveness of the proposed controllers.

Keywords : FPGA, HDL Coder, ANFIS, Neuro-fuzzy control, Extended Kalman Filter, Fixed-Point Tool, FPGA in the loop, Inverted pendulum.